Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination

R. Alur[⋆], J. Esposito[⋆⋆], M. Kim[⋆], V. Kumar[⋆⋆], and I. Lee[⋆]

Abstract. The design of controllers for hybrid systems (i.e. mixed discrete-continuous systems) in a systematic manner remains a challenging task. In this case study, we apply formal modeling to the design of communication and control strategies for a team of autonomous robots to attain specified goals in a coordinated manner. The model of linear hybrid automata is used to describe the high-level design, and the verification software HyTech is used for symbolic analysis of the description. The goal of the project is to understand tradeoffs among various design alternatives by generating constraints among parameters using symbolic analysis. In this paper, we report on difficulties in modeling a team of robots using linear hybrid automata, results of analysis experiments, promise of the approach, and challenges for future research.

1 Introduction

A hybrid system typically consists of a collection of digital programs that interact with each other and with an analog environment. Examples of hybrid systems include manufacturing controllers, automotive and flight controllers, medical equipment, micro-electro-mechanical systems, and robots.

Designing reliable hybrid systems is a challenging task. Control theoretic tools enable the design of continuous controllers in a single mode of operation. While nonlinear switching controllers have been designed for systems with several modes of operation (see [Bra95,Utk77,ZB98]), the techniques are generally only applicable for simple systems with relatively few modes. Nonlinear systems are, in general, solved on a case by case basis. Even when solutions exist, the properties of the design such as stability, convergence and reachability only apply within certain limited operating regimes. Typically the state-space for hybrid systems can be partitioned into many regions so that within each region a different control law with predictable performance can be designed. Variations on this theme can be found in the literature on variable structure systems [Utk77] and on multimodal systems [NBC95]. 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

[⋆] Department of Computer and Information Science, University of Pennsylvania. Email: `alur,moonjoo,lee@cis.upenn.edu`

[⋆⋆] Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania. Email: `jme,kumar@grip.cis.upenn.edu`

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 [LGS95,TPS97]. However, the the hierarchy is designed manually, and the solution often requires very strict assumptions and constraints.

Inspired by the success of automated formal methods in discovering subtle errors in hardware designs (cf. [CK96]), a current trend is to investigate if these techniques can be generalized to obtain design aids for hybrid systems (the proceedings of the workshops on hybrid systems provide an excellent survey of emerging trends [GNRR93,AKNS95,AHS96,Mal97,HS98]).

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.

The formal analysis of the mixed digital-analog nature of hybrid systems requires a mathematical model that incorporates the discrete behavior of computer programs with the continuous behavior of environment variables such as time, position, and temperature. The model of our choice is the *hybrid automaton*—a finite automaton augmented with a finite number of real-valued variables that change continuously, as specified by differential equations and differential inequalities [ACH+95]. Algorithmic analysis of hybrid systems is a challenging problem since the presence of continuous variables results in an infinite state-space, and even the simplest analysis problems turn out to be undecidable. However, 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 [AHH96,HHW97], and has been applied to case studies such as audio-control protocol [HW95] and steam boiler [HW96].

In this paper, we explore the application of formal modeling and analysis to design of multi-robot coordination based upon an experimental testbed of a system of autonomous, mobile robots. We consider a task that involves exploring a room with unknown geometry and with obstacles, and identifying and locating an obstacle. The experimental system and the task is motivated by military applications (scouting, reconnaisance, and surveillance) and the need to minimize human intervention in hazardous environments a wide range of civilian applications including space and nuclear facilities. 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. Our robots are equipped with omnidirectional cameras, and they operate in a two-dimensional world. We make the realistic assumption that the errors in estimates of the position and geometry of a candidate object (target or obstacle) decrease as the robot gets closer to the object. The control of a single robot requires image processing algorithms for interpreting the visual information, computer vision algorithms for building a model of the objects around the robot and for localizing the position of the robot, and planning algorithms for selecting a local direction for steering or for selecting a control law. We can abstract this problem and reduce it to the two subproblems of building estimates of the positions and shapes of objects around the robot and of planning a path around the objects to a target goal. When there are multiple robots that can communicate with one another, they can communicate and share knowledge about the world. For instance, two robots can exchange their individual estimates about a specific obstacle, conclude that the obstacle must lie within the region consistent with both the estimates, and employ this information for better path planning. 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. This design problem is not unlike the design problems encountered in a wide range of hybrid systems including intelligent vehicle systems and flight management systems.

Traditionally, verification tools such as KRONOS [DOTY96], UPPAAL [LPY97] and HYTECH have been used to detect logical errors by checking whether a high-level model satisfies a temporal logic requirement. For us, the goal of formal modeling and analysis is to explore and compare various design alternatives. For instance, even for simple communication and control strategies, we need to set various parameters such as 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. Standard simulation, using a tool like MATLAB (see `www.mathworks.com`), requires fixed setting of all such parameters. In a verification tool such as HYTECH, such parameters can be left unspecified, and the 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.

After reviewing the basics of formal verification of hybrid automata and the state of the art in design of hybrid controllers in Section 2, we explain the general scenario of experimental testbed of coordinating robots in Section 3. The main effort in this case study concerns modeling the application scenario using linear hybrid automata. The modeling issues are discussed in detail in Section 4. Linear hybrid automata require all expressions to be linear, and all differential constraints to be constant rectangular inclusions. 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 is quite simple, 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.

The results of the analysis experiments are reported in Section 5. Since the analysis is computationally expensive, we could successfully analyze only special cases of the scenario. In particular, for two robots and one obstacle, HyTech could synthesize the region of the possible positions of the target for which communication improves the distance traveled. While modest, this experiment does yield information that is computed automatically by a general-purpose tool.

In summary, the paper serves two purposes. First, it shows the promise, and the road-map, for applying tools for formal modeling and analysis to the problem domain of multi-robot coordination. Second, we believe that the outlined scenario will serve as a challenge problem to guide the research in formal verification of hybrid systems.

2 Verification of Hybrid Systems

A hybrid automaton [ACH$^+$95] is a formal model to describe reactive systems with discrete and continuous components. Formally, a hybrid automaton H consists of the following components.

- A finite directed multi-graph (V, E). The vertices are called the *control modes* while the edges are called the *control switches.*
- 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 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 Σ. A *region* is a subset of Σ.
- A function *init*, that assigns to each mode v, a set $\mathit{init}(v) \subseteq \Sigma_X$ of valuations. This describes the initialization of the system: a state (v, ν) is initial if $\nu \in \mathit{init}(v)$. The region containing all initial states is denoted σ^I.
- A function *flow*, that assigns to each mode v, a set $\mathit{flow}(v)$ of C^∞-functions from R^+ to Σ_X. This describes the way variables can evolve in a mode.
- A function *inv*, that assigns to each mode v, a set $\mathit{inv}(v) \subseteq \Sigma_X$ of valuations. The system can stay in mode v only as long as the state is within $\mathit{inv}(v)$, and a switch must be taken before the invariant gets violated.
- A function *jump*, that assigns to each switch e, a set $\mathit{jump}(e) \subseteq \Sigma_X \times \Sigma_X$. This describes the enabling condition for a switch, together with the discrete update of the variables as a result of the switch.
- A function *syn*, that assigns to each switch e, a label $\mathit{syn}(e)$ 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 automaton H starts in an initial state. During its execution, its state can change in one of the two ways. A *discrete* change causes the automaton 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 automaton are captured by defining transition relations over the state space Σ. For a switch $e = (v, v')$, we write $(v, \nu) \to_e (v', \nu')$ if $(\nu, \nu') \in \mathit{jump}(e)$. For a mode v and a time increment $\delta \in R^+$, we write $(v, \nu) \to_\delta (v, \nu')$ if there exists a function $f \in \mathit{flow}(v)$ such that $f(0) = \nu$, $f(\delta) = \nu'$, and $f(\delta') \in \mathit{inv}(v)$ for all $0 \leq \delta' \leq \delta$. The transition relations $\to_e$ capture the discrete dynamics, while the transition relations $\to_\delta$ capture the continuous dynamics.

A key operation on hybrid automata is the *product* operation. For two hybrid automata H_1 and H_2, the product $H_1 \| H_2$ is defined to be another hybrid automaton that describes the behavior of the composite system with two concurrent components. The formal definition of product can be found in [AHH96]. Note that the component automata can have shared variables to communicate, and in addition, can synchronize transitions using common labels on jumps.

Algorithms for symbolic reachability analysis of hybrid automata manipulate regions. Let σ be a region of H. The *successor region of* σ, denoted $\mathit{post}(\sigma)$, contains states q' such that $q \to_e q'$ for some $q \in \sigma$ and some switch e, or $q \to_\delta q'$ for some $q \in \sigma$ and some $\delta \in R^+$. A state q is said to be reachable if $q \in \mathit{post}^i(\sigma^I)$ for some natural number i. The set of reachable states of a hybrid automaton H is denoted $\mathit{reach}(H)$.

The central problem in algorithmic formal verification of hybrid systems is to compute the set of reachable states of a given hybrid automaton. The set of all reachable states of a hybrid automaton can be computed by repeatedly applying *post* to the initial region. For a special class of automata, called *linear hybrid automata*, all the regions encountered during the computation can be described by boolean combinations of linear inequalities over automaton variables.

A hybrid automaton $H = (V, E, X, \mathit{init}, \mathit{flow}, \mathit{jump}, \mathit{syn})$ is called *linear* if

1. For each mode v, the sets $\mathit{init}(v)$ and $\mathit{inv}(v)$ are described by boolean combinations of linear inequalities over the variables X.
2. For each switch e, $\mathit{jump}(e)$ 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 $\mathit{flow}(v)$ iff the first derivative $\dot{f}$ of f with respect to time satisfies each linear inequality for all times $\delta \in R^+$.

The above requirements ensure that for each i, the set $\mathit{post}^i(\sigma^I)$ can be described by a boolean combination of linear inequalities [ACH+95]. Furthermore, such sets can be computed effectively. The software HyTech [AHH96,HHW97] 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: system description section and analysis section. The system description section has a textual representation of 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. This section contains definitions of regions, each of which represents a set of states in the system and commands for manipulating these regions. Properties are checked by reachability test of region. 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 4.

3 Multirobot coordination

3.1 A typical scenario

Consider the scenario in which a small team of robots enters a room with obstacles, identifies and locates a target object, and retrieves this object from the room as shown in Figure 1. In the figure, the team consists of five robots, designated as R_1 through R_5. The obstacles are denoted by A through E and the target is denoted by T. In order to keep the problem simple we assume that the geometry is two dimensional. Each robot is equipped with a camera that allows it to identify other robots, obstacles and the target. The sensor has errors in estimating the position of objects (obstacles, targets and other robots) that decrease as the robot approaches the object. Typically, it is necessary for the robot to get within some threshold distance before it can make a positive identification. In addition, each robot has the ability to determine its own position and orientation. This may come from an independent sensor or from the camera and landmarks in the environment. The 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. Each robot is controlled independently and is able to move in the room. Although there is no central controller, it is possible to have a situation in which a robot assumes the role of the leader, issues commands to and obtains reports from all other robots in the team, and thus all the decision making is centralized. Finally, the target's shape and size are such that it requires at least two robots to cooperatively transport the object.

As shown in the initial configuration in Figure 1, the team of small robots enters the room. They may or may not have a nominal model of the room. Even if they have a nominal model, there will always be a need to identify and locate features (obstacles, targets and the walls) in the room. The robots disperse to search the room. As they search the room, they identify and locate obstacles. The communication protocol allows the robots to exchange information about their position and orientation, the identity and the position of objects in the room. They may also be able to communicate with each other, and thus, the actions of one can depend on the plans of others.

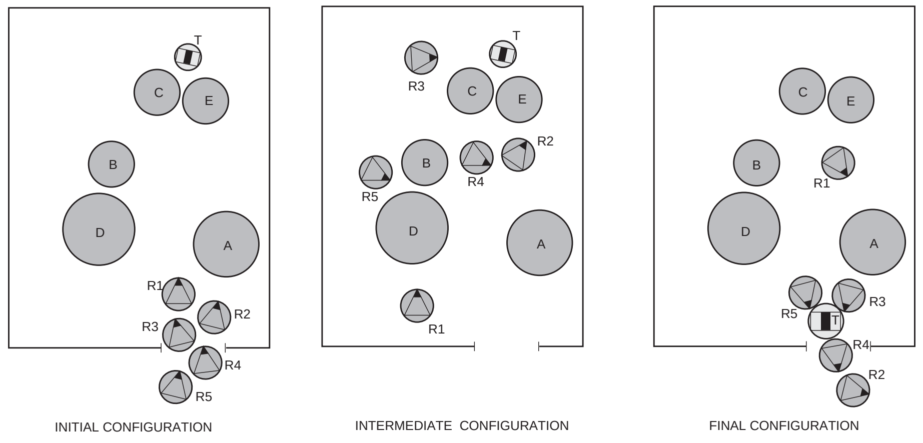


Fig. 1. A robot team searching a room with obstacles, to locate and retrieve a target

One of the robots (R_3) eventually locates the target as shown in the intermediate configuration in the figure. When the other robots get this information, they stop searching for the target and organize themselves into a formation. Two or three robots (R_4, R_3, and R_5 in the figure) organize themselves around the target object in order to move the object. The other robot(s) (R_2 in the figure) may lead the robots with the target object and act as scouts as shown in the possible final configuration in the figure. It is quite possible that one or more robots may fail. As shown in the figure, robot R_1, the team leader in the initial configuration, has an actuator failure and is left stranded. However, the team, which now consists of the four remaining robots, successfully completes the mission.

In this scenario, each robot is driven by actuators and sensors that are intrinsically continuous. The dynamics are derived from laws of physics and are represented by continuous mathematics. Therefore the robot behavior is continuous. However, this behavior changes, possibly discontinuously, as new information becomes available or as new events occur. For example, a robot pursuing the identification of an object that is possibly the target object, abandons this course of action when it is told that another robot has identified the target. The exchange of information changes the behavior of the robot. A robot approaching a target changes its control strategy when it makes contact with the target. The event of making contact changes the behavior (dynamics) of the robot.

3.2 The specific example

In the specific case used in formal modeling, we consider two robots, two static convex obstacles and a goal target position for both robots as shown in Figure 2. The solid lines indicate the actual obstacle location while the dotted lines indicate the initial obstacle model. Note that in the initial model the two obstacles

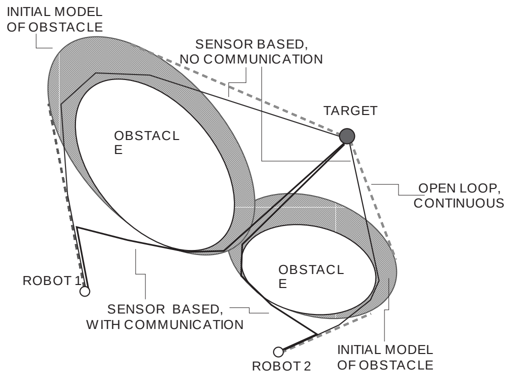

INITIAL MODEL OF OBSTACLE
SENSOR BASED, NO COMMUNICATION
TARGET
OBSTACL E
OPEN LOOP, CONTINUOUS
OBSTACL E
ROBOT 1
SENSOR BASED, WITH COMMUNICATION
INITIAL MODEL OF OBSTACLE
ROBOT 2

Fig. 2. A 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.

appear to overlap. An open loop control for each robot without any communication or sensing yields the dotted paths labeled "open loop: continuous." Each robot follows a continuous control law. There is only a single control mode. 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 black lines labeled "sensor based: no communication." 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 the third case, the two robots exchange information about their world models at discrete intervals. The corresponding paths followed by the robots are labeled "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.

3.3 Assumptions

While it is possible to analyze the problems discussed above via simulation, our goal is to pursue symbolic methods and to develop verified control strategies. In order to keep the analysis of the problem simple, we will make a number of simplifying assumptions.

First, we will restrict our attention to mobile robots operating in planar environments. Robots will modeled as points in $\Re^2$. In other words, we ignore the orientation of the robot. We also ignore the nonholonomic constraints that may limit the robots' direction of motion, as in the case of wheeled carts. Further, we will model the dynamics by a set of first order differential equations:

$$\dot{x} = u_1$$
$$\dot{y} = u_2$$

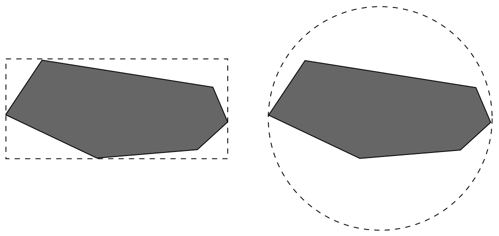


Fig. 3. A rectangle provides a reasonable approximation to most convex polygons, as compared to a circle.

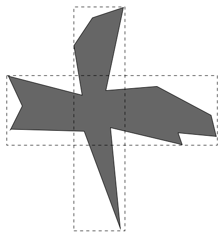


Fig. 4. Non-convex shapes can be well approximated with several overlapping rectangles

where (x, y) are the coordinates of a robot and (u_1, u_2) are the control inputs, in this case velocities. Note the equations are linear, which would not be the case if the orientation of the robot is included in the description.

Second, we will discretize our model of the robots' environment. We will superimpose a grid on the $x-y$ plane and assume that robots can only move along the grid. Thus, 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 successively finer directional discretizations could be made to describe complicated trajectories at the price of computation time.

Another implication of this assumption is that models of obstacles are limited to rectangles or unions of rectangles. As shown in Figures 3 and 4, the use of rectangles to describe objects in the workspace is not necessarily a limitation. Most general polygons can be reasonably approximated by a rectangle. Note that non-convex obstacles can be approximated using multiple overlapping rectangles.

In the grid world, we will use the so-called Manhattan metric or the L_1 norm to measure the distance between two points. The Euclidean metric makes the distance a nonlinear function of the state, but the Manhattan distance is linear. The Manhattan distance, D_M, from Point A to Point B is simply:

$$D_M(\overline{AB}) = \mid X_A - X_B \mid + \mid Y_A - Y_B \mid . \tag{1}$$

Finally, we make some assumptions about the model for sensing and estimation. 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 unknown. In other words, we assume that 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. Such uncertainty models are reasonable approximations of sensor systems where errors are primarily geometric in origin.

For example, in vision applications in a two-dimensional world without occlusions and problems due to segmentation, the accuracy is limited by CCD resolution, especially at long ranges, and the estimates improve as the distance to target decreases. This is also true with sonars. In Figure 5, 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 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.

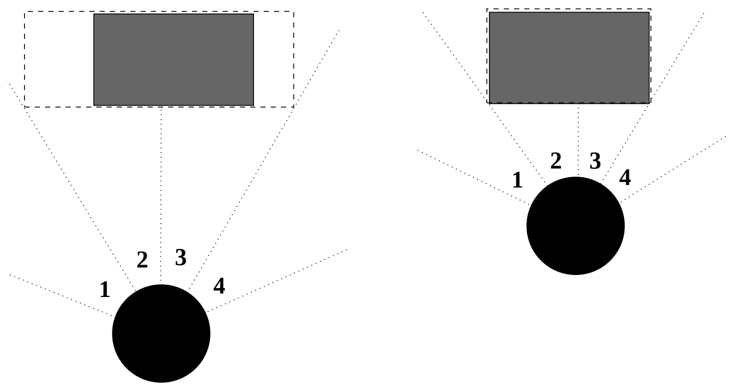


Fig. 5. An overhead view of a mobile robot equipped with sonar arrays detecting a rectangular obstacle. The sonars return the closest distance to an object which lies somewhere within the ensonification cone. At greater distances (left figure) the uncertainty can be rather large since the robot only 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.

4 Modeling

4.1 Robots

Each robot is described by coordinates (x, y). In our model of the robot and the grid world, the robot only has four distinct modes of travel:

$$\begin{array}{lll} \textit{right}: & \dot{x} = v_{max}, & \dot{y} = 0 \\ \textit{left}: & \dot{x} = -v_{max}, & \dot{y} = 0 \\ \textit{forward}: & \dot{x} = 0, & \dot{y} = v_{max} \\ \textit{back}: & \dot{x} = 0, & \dot{y} = -v_{max} \end{array} \tag{2}$$

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

4.2 Obstacles and Workspace

Consider the problem of a team of such mobile robots R_i, for $i = 1, \ldots, N$ navigating an environment toward their respective goal configurations (G_x^i, G_y^i). The environment is populated with multiple polygonal obstacles O_j, for $j = 1, \ldots, M$ which occupy closed sets in $\Re^2$. Note that these obstacles are assumed to be in fixed positions. Therefore, the collision free space through which the robot is permitted to move is:

$$\mathcal{W} = \Re^2 - \cup_j O_j. \tag{3}$$

As discussed in Section 3.3, to ensure linearity and computational feasibility, the obstacles are modeled as rectangles 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.

4.3 Sensor Model

Let $Y_j^i(t)$ be a closed set in the plane which represents the i^{th} robot's estimate of the j^{th} obstacle at time t. As discussed in Section 3.3, the initial map, $Y_j^i(0)$, and any subsequent estimates, $Y_j^i(t)$, are strictly worst case representations with no stochastic uncertainty. In other words, the uncertainty in a given estimate is bounded in such a way that

$$Y_j^i(t) \supseteq O_j, \forall t \geq 0. \tag{4}$$

Although it is not known *where* O_j lies in $Y_j^i(t)$, it is certain that $O_j \cap (\neg Y_j^i(t)) = \emptyset$. 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, $Y_j^i \longrightarrow O_j$.

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

$$X_L^O(t) = X_L^A + \underbrace{(X_L^O(0) - X_L^A)\frac{\mathrm{d}}{\mathrm{d}_0}}_{error}, \forall t \geq 0. \tag{5}$$

$$X_R^O(t) = X_R^A + \underbrace{(X_R^O(0) - X_R^A)\frac{\mathrm{d}}{\mathrm{d}_0}}_{error}, \forall t \geq 0. \tag{6}$$

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 distance from the robot to the obstacle at time zero.

One limitation of the sensor model is its inability to capture the nonlinearities that appear in real life. However, we argue that the above model captures the essential characteristics of sensor model and higher dimensional linear approximations to sensor models can be used to improve the accuracy. A more serious limitation is due to the use of the Manhattan distance. 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 the robot's estimate will not strictly improve as the robot approaches an obstacle along certain paths.

4.4 Coordination

The robots collectively represent a team and therefore it may be advantageous for them to exchange information periodically. For instance, at discrete time intervals, robot R_i may send its current map of the environment to robot R_k. 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,

$$Y_j^k new = Y_j^i \cap Y_j^k. \tag{7}$$

R_k's resulting estimate of obstacle j, $Y_j^k new$, 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. Note that the parameters describing the updated estimates are linear functions of the parameters describing the two old estimates.

4.5 Control strategy

The term control strategy 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 [Lat91] . For this scenario, the workspace decomposition used is shown in Figure 6.

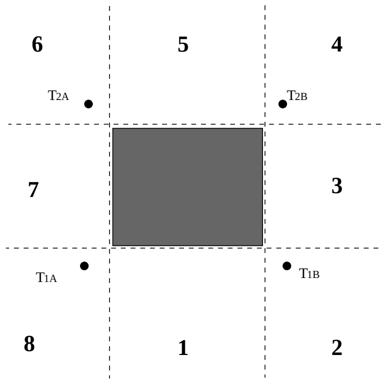


Fig. 6. Illustration of the exact cell decomposition planning method. The dark rectangle represents the obstacle while the numbered regions are free cells in the workspace.

For a point robot navigating amidst rectangular obstacles, only two separate cases need to be considered. First, suppose the robot is currently in cell 1 (the cases for cell 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_{1A} (or T_{1B}) 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_{1A} (or T_{1B}), then it sets a new temporary goal T_{2A} (or T_{2B}) based on which intermediate point will result in the shortest overall path. Once it reaches T_{2A} (or T_{2B}), 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 anti-clockwise 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. [1]

A complete description of the robot's behavioral algorithm as a finite state machine appears in Figure 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 some time period
}
```

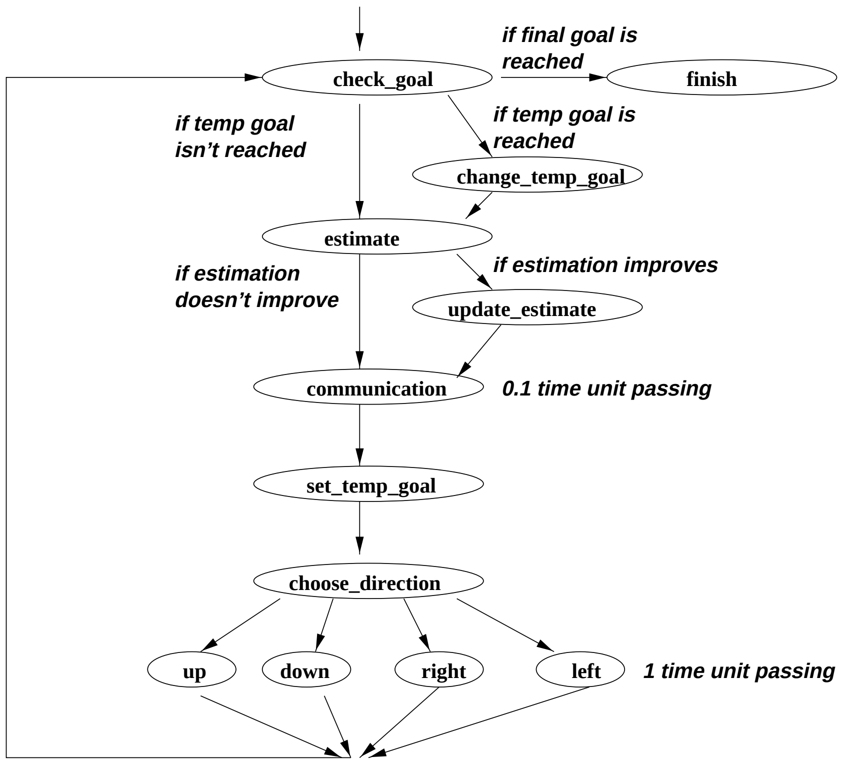


Fig. 7. Representation of the robot's behavioral algorithm as a Finite State Machine.

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

4.6 Cost model

As mentioned in the previous section the robots attempt to choose behaviors which minimize some type of cost function. In this model, the cost function indicates sum of the time taken to travel a path and the time taken to communicate.

The shortest path between two points is not unique when using the Manhattan distance even in the absence of obstacles, 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 equi-spaced grid. In this case, when traveling from grid point (X_A, Y_A) to grid point (X_B, Y_B) there are

$$\frac{(\mathrm{N}+\mathrm{M})!}{\mathrm{N}! \cdot \mathrm{M}!} \tag{8}$$

distinct shortest paths, provided there are no obstacles in the region $[X_A, X_B] \times [Y_A, Y_B]$. 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.

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 Figure 8 these bounds can be expressed as

$$D_l = \frac{\sqrt{2}}{2} D_M \leq D_E \leq D_M = D_u \tag{9}$$

where D_E is the Euclidean distance. Note that $\frac{\sqrt{2}}{2} \approx 0.707$, which implies that the Manhattan distance, at most, over estimates the actual distance by approximately 41 percent.

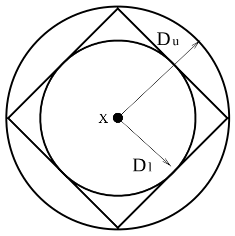


Fig. 8. The diamond shaped line represents the set of points equidistant from X. The circles indicate the upper and lower bounds on the actual distance measured in the Euclidean sense

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.

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

$$J^i = \frac{D_M^i}{v_i} + \rho_{comm} \cdot f \cdot \frac{D_M^i}{v_i} \tag{10}$$

where D_M^i is a total distance traveled by R^i in the sense of the Manhattan metric and v_i is the speed of R^i.

5 Results

5.1 Example

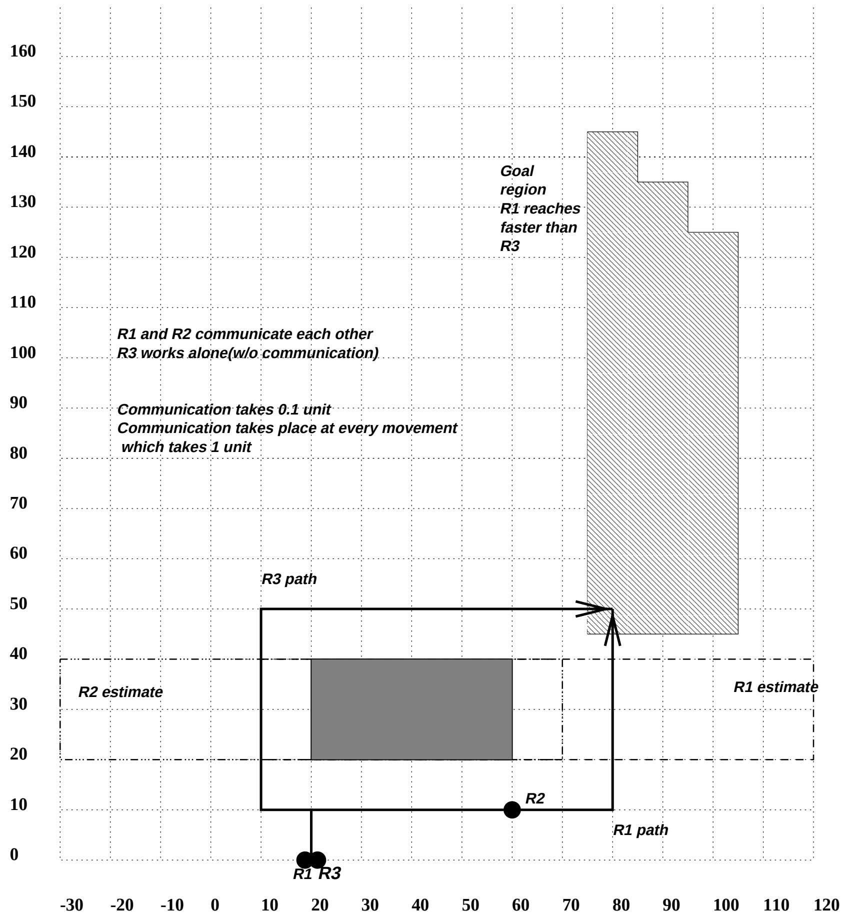
Our scenario contains three identical robots (R_1, R_2 and R_3), one fixed obstacle and one fixed goal. R_1 and R_2 collaborate via communication, while R_3 works by itself. The initial positions of R_1 and R_3 are the same. Communication takes a fixed amount of time for each time message exchanged. 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.

Initially R_1 and R_3 are located at (20,0). R_2 is located at (60,10). The target is located at (80,50). 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 right hand side, but R_2 estimates the obstacle far larger toward 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 as estimates). The direction of movement is determined at the end of each iteration. A robot moves for 1 time unit once direction is determined (see Figure 7). A robot moves 10 units of distance in 1 time unit. R_1 and R_2 communicate every time unit. Communication has a cost of 0.1 time unit.

Initially R_3 sets up a temporary goal as (0,10). It is because the estimated length of left path toward the goal is shorter than the length of right path. It chooses left path for reaching the goal. 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 a 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 collaboration between R_1 and R_2 helps R_1 to reach the goal faster than R_3 in this scenario.

5.2 Parametric analysis

Setting x and y positions of the goal as parameters, we can compute the geometric region in which R_1 reaches faster than R_3 with the help of communication



Goal region R1 reaches faster than R3
R1 and R2 communicate each other
R3 works alone(w/o communication)
Communication takes 0.1 unit
Communication takes place at every movement which takes 1 unit
R3 path
R2 estimate
R1 estimate
R2
R1 path
R1 R3

Fig. 9. The scenario of robot modeling

(see figure 9). With the help of communication, R_1 can choose a shorter path which saves two time units compared to R_3, when the goal lies in the shaded region shown. However, R_1 constantly communicates with R_2 and this communication overhead accumulates so that this overhead cancels out the saving after 20 movements. That is why the region has a stair-like shape.

Also, we can compute the optimal frequency of communication for reaching the goal. Setting the *period* of communication as a parameter, we modified the model so that a robot communicates only every *period* number of iterations. A domain of *period* is finite because period should be less than time for robots to reach the goal. Thus, we can choose the frequency which leads to the minimal total elapsed time.

We get a result that when R_1 and R_2 communicate once in two unit times, R_1 takes 11.5 time units to reach the goal (80,50).

In addition, two safety properties for a robot controller are verified. First, a robot never collides with the obstacle while it navigates to reach any valid goal position. A valid goal position is any position outside of the 5 units extension of the initially estimated obstacle. We added a monitor automaton to the description so that the monitor can check whether a position of a robot is overlapped with the estimate of the obstacle. Second, the verification establishes that a robot *does* reach any valid goal position in the work space

5.3 Limitations of the analysis

We had to make several simplifications in order to make the analysis tractable. 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 could not allow more than three robots in the verification. We limited the range of x between -30 and 120 and the range of y between 0 and 160. Furthermore, we had to divide this region into 21 partitions to make the analysis tractable. The robot model description is around 1700 lines. We verified this description using Sun Enterprise 3000 (4 X 250 Mhz UltraSPARC) with 1GB physical memory. Verifying each region took up to 1.3 GB memory space and one hour[2].

Another limitation is the internal arithmetic overflow error of HyTech when HyTech manipulates a complex region. A region is defined by a set of linear constraints. A linear region becomes more and more complex at each iteration of computation. Although linear constraints in the model description do not look complex, HyTech can eventually generate an overflow error after many iterations. Therefore, we have to be careful to make the linear equations as simple as possible. In modeling a sensor, linear equations can be complex depending on which number we choose for a initial position of robot and obstacle. For example, estimated x position of the obstacle is formulated as

$$x = x_{real} + (x_{init} - x_{real}) \times \frac{|(x_r - x_{real})| + |(y_r - y_{real})|}{d_{init}}$$

where x_r, x_y are position of a robot. Let us see the equation of estimation of x_1 by R_1, where x_1 is a x position of left end of the obstacle. In the scenario $x_{real} = 20, y_{real} = 20, x_{init} = 10, d_{init} = 20$. It generates a simple equation $x = \frac{x_{r1}}{2} + \frac{y_{r1}}{2}$ when the obstacle is at the upper right side of R_1. However, if this equation becomes a little more complex, HyTech generates "Library overflow" error even with enough free memory.

6 Conclusions

We have reported a case study in applying formal modeling and analysis aimed at exploring alternatives in the design of multi-robot coordination systems. Simultaneous design of control strategies and coordination protocols for interacting

[2] We used `top` for checking memory usage.

dynamical systems is a significant challenge. We believe that tools for high-level design and analysis can greatly aid the design process.

The gist of our approach is to describe the system as interacting hybrid automata, and then employ symbolic analysis to compute the constraints among various parameters for a given objective. Note the generality of this approach compared to prevalent methods in simulation in which either the parameters need to be set to specific values or the computation of the constraints for the specific problem needs to be programmed. Even though we have reported only modest success in the goals of the exercise, we hope that it illustrates the potential of the approach.

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 are

Computational requirements. As reported in Section 5, all the parameters had be scaled down to be able to get feedback from HyTech. Improving the efficiency of polyhedra-based analysis remains a significant challenge.

Expressiveness. As described in Section 4, the linearity requirement forces us to do a variety of approximations. While the issue of approximating complex dynamics by rectangular inclusions has received attention in literature, for us, issues such as approximating Euclidean distance by Manhattan distance were particularly unsatisfactory. This problem suggests directions for further research and tool development for more general classes of problems.

We conclude by noting that this is, to our knowledge, the first use of formal methods in the analysis of multirobot coordination and communication. This problem is particularly complicated because of the use of continuous controllers and planners with discrete protocols for communication and for refining world models. Our approach points a way to analyze the role of communication in multirobot coordination, and to establish the dependence of the team performance on the number of team members and on the rate of communication.

Acknowledgements We gratefully acknowledge the support of NSF grant CISE CDS 97-03220-001, NSF CAREER award CCR-9734115,NSF CCR-9619910, ARO DAAG55-98-1-0393, ARO DAAG55-98-1-0466, ONR N00014-97-1-0505 and the DARPA ITO/MARS program. J. Esposito was supported by a Department of Education fellowship.

References

[ACH+95] 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. *Theoretical Computer Science*, 138:3–34, 1995.

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[AHS96] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III: Verification and Control.* LNCS 1066. Springer-Verlag, 1996.

[AKNS95] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems II.* LNCS 999. Springer-Verlag, 1995.

[Bra95] M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Massachusetts Institute of Technology, 1995.

[Bro93] R. W. Brockett. Hybrid models for motion control systems. In H. L. Trentelman and J. C. Willems, editors, *Essays in Control: Perspectives in the Theory and its Applications*, pages 29–53. Birkhäuser, 1993.

[CK96] E.M. Clarke and R.P. Kurshan. Computer-aided verification. *IEEE Spectrum*, 33(6):61–67, 1996.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III: Verification and Control*, LNCS 1066, pages 208–219. Springer-Verlag, 1996.

[GNRR93] R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors. *Hybrid Systems.* LNCS 736. Springer-Verlag, 1993.

[HHW97] T.A. Henzinger, P. Ho, and H. Wong-Toi. HyTech: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1, 1997.

[HS98] T. Henzinger and S. Sastry, editors. *Hybrid Systems: Computation and Control.* LNCS 1386. Springer-Verlag, 1998.

[HW95] P.H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *Proceedings of the Seventh Conference on Computer-Aided Verification*, LNCS 939, pages 381–394. Springer-Verlag, 1995.

[HW96] 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*, LNCS 1165, pages 265–282. Springer-Verlag, 1996.

[Lat91] J.-C. Latombe. *Robot motion planning.* Kluwer Academic Publishers, 1991.

[LGS95] J. Lygeros, D. N. Godbole, and S. Sastry. A game-theoretic approach to hybrid system design. In *Hybrid Systems III. Verification and Control*, pages 1–12. Springer-Verlag, 1995.

[LPY97] K. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1, 1997.

[Mal97] O. Maler, editor. *Hybrid and Real-Time Systems.* LNCS 1201. Springer-Verlag, 1997.

[NBC95] K. S. Narendra, J. Balakrishnan, and K. Ciliz. Adaptation and learning using multiple models, switching and tuning. *IEEE Control Systems Magazine*, pages 37–51, 1995.

[TPS97] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, August 1997. Accepted as a regular paper.

[Utk77] V. I. Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*, Vol. 22(2):212–222, 1977.

[ZB98] M. Zefran and J. Burdick. Stabilization of systems with changing dynamics. In *Hybrid Systems*, 1998.

[ZDK96] M. Zefran, J. Desai, and V. Kumar. Continuous motion plans for robotic systems with changing dynamic behavior. In *2nd Int. Workshop on Algorithmic Foundations of Robotics*, 1996.