Bisimulation Based Hierarchical System Architecture For Single-Agent Multi-Modal Systems

T. John Koo and Shankar Sastry

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720
{koo,sastry}@eecs.berkeley.edu

Abstract. In this paper, a hierarchical system architecture for single-agent multi-modal systems is proposed. The layered system is designed to promote proof obligations so that system specification at one level of granularity conforms with that at another level and vice versa. The design principle for the construction of the hierarchy is based on bisimulation with respect to reachability specifications. Therefore, a higher-level system and a lower-level system are bisimilar. Our approach is illustrated by designing a system architecture for controlling an autonomous agent.

1 Introduction

Control of multi-agent systems focus on the control of individual agents to accomplish a mission collectively, while satisfying their dynamic equations and inter-agent formation constraints, for an underlying communication protocol being deployed. Advances in embedded software, computation, communication, and new methods of distributed sensing and actuation are revolutionizing the development of advanced control technologies for distributed, multi-agent systems. These advances also enable the conduct of missions deemed impossible in the recent past.

Imposing a hierarchical structure on the system architecture has been used for solving the control problem of large-scale systems[5, 15, 17, 19]. A desired hierarchical structure should not only provide manageable complexity but also promote verification. There are several approaches to understanding a hierarchy depending on the design perspective. In particular, two distinct approaches have been shown in [20] for the design and analysis of AHS [19]. One approach to the meaning of hierarchy is to adopt *one-world* semantics, and the other approach is referred to as *multi-world* semantics.

In one-world semantics for hierarchical systems as shown in [18], a higher-level expression is interpreted in a process called *semantic flattening*: the expression is first compiled into lower-level expression and then interpreted. In other words, an interpretation at each level is semantically complied into a single interpretation at the lowest-level in the *imperative* world. Furthermore, semantic flattening

implies that checking any high-level truth-claim can be performed by an automatic procedure if there is a facility for automatically verifying the lowest-level interpretation. This approach provides a unique interpretation to system description and verification. However, a major drawback to one-world semantics is that higher-level syntax and truth-claims have to be reformulated if there are some changes at any of the lower levels. The advantage of using one-world semantics for hierarchical systems is gained at the heavy cost of a rigidity of framework that makes it unsuitable for most complex and heterogeneous system. On the other hand, in multi-world semantics for hierarchical systems, an expression at each level is interpreted at the same level. Therefore, checking the truth-claim at that level is performed in its own *declarative* world. This approach conforms with common system design practice. However, relating these disconnected worlds together is a nontrivial task. In the following, we will present a multi-agent system to motivate our discussion on the design of hierarchical system architecture.

Consider a mission of controlling a group of autonomous agents in the pursuit of multiple evaders. Assume that each agent is a UAV equipped with necessary computation, communication, and sensing capabilities to accomplish the mission. Different approaches have been proposed in solving the pursuit-evasion game either in deterministic [14, 16] or probabilistic framework [7, 8] based on complete or partial information about the environment. In the common setting of the game, the game is performed on a finite graph $G = (S, E)$ with node $s \in S$ and all allowed motions for the players are represented by edges $e \in E \subseteq S \times S$ connecting the nodes. Each node may be occupied by more than one agent. The game is then performed on the discrete graph G, and each action of an agent depends on a discrete event generated from a given strategy. Depending on the level of centralization and the nature of the game, a specific set of strategies can be selected to accomplish the mission. In the game, an evader is *captured* if the evader and one of the pursuers occupy the same node.

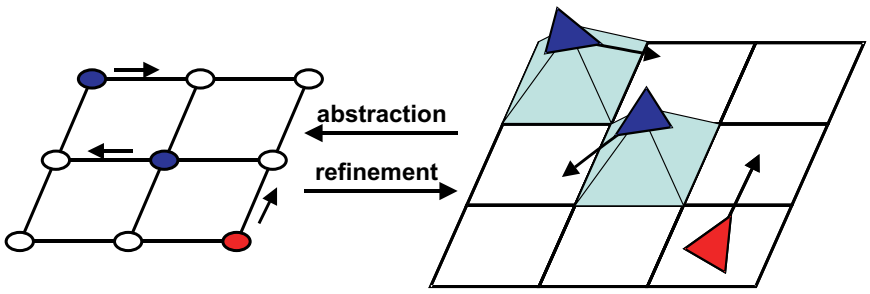

Fig. 1. A hierarchical system for solving the Pursuit-Evasion game is designed as a two-layered system in which decision-making procedures for solving the game on the discrete graph are performed at the top level and motions being generated in the physical world by agents are exhibited at the bottom level.

However, the actual game is taken place in a continuous space $X \subseteq \mathbb{R}^3$ by the agents. In order to implement the discrete game in the continuous space X, one has to construct a partition of X which contains a finite number of cells X_j for $j = 1, \ldots, M$ with each cell corresponding to a node on the graph G. Furthermore, for an agent, each allowed motion on the graph has to be refined to feasible motions by exhibiting multi-modal behaviors. Hence, the system is designed as a two-layered system in which decision-making procedures for solving the game on the discrete graph are performed at the top level and motions being generated in the physical world by agents are exhibited at the bottom level. Between these two worlds, state information at the bottom level is being abstracted at the top level and control information at the top level is being refined at the bottom level.

To achieve high level of mission reliability, it is desirable that the layered system is designed to promote proof obligations so that system specification at one level of granularity *conforms* with system specification at another level. Consider two levels of a system and system specification at a lower-level conforms with system specification at a higher-level. Hence, there is a tight relation between the levels since each detailed state at the lower-level corresponds to an abstract state at the higher-level, and each transition at the lower-level corresponds to a transition at the higher-level. This relation is captured mathematically by the notion of *simulation*. If system specification at the higher-level conforms with system specification at the lower-level, the detailed system at the lower level and the abstracted system at the higher level are called *bisimilar*.

In this paper, we are interested in the design of a hierarchical system architecture for a single-agent multi-modal system based on bisimulation with respect to reachability specification. Therefore, if the abstracted system at the top level, which utilizes the graph G, and the detailed system at the bottom level, which is a hybrid system containing a collection of control modes, are bisimilar with respect to reachability specification, then the reachability problem for the hybrid system can be converted to an equivalent reachability problem on the finite graph G. If, in addition, the equivalent problem can be performed in a computationally feasible way, then the reachability problem for the hybrid system is *decidable*. An agent model inspired by the motion capability of helicopter is used as an example to demonstrate the effectiveness of the proposed concepts for solving the control system design problem.

2 Single-Agent Multi-Modal Systems

Given the motion capability of an agent, we assume that there exist control strategies such that a finite number of directions of motion can be generated. Any control strategy may utilize a single controller or a sequence of controllers for generating the motion directions. In this paper, we consider that an agent can only move in a horizontal plane and assume that it has five possible motion directions. In general, depending on the choice on a control strategy, one can have different sets of motion directions. However, we will show that the feasible

motion directions would affect the construction of the partition which are used for abstracting state from detailed system to abstracted system.

2.1 Hybrid Automaton

Motivated by an design example of a helicopter based UAV shown in [10], we consider that the detailed system is modeled as a multi-modal system with five control modes [9]. In each control mode, there is a closed-loop dynamics embedded. The system can further be modeled as a hybrid automaton [13]. As depicted in Figure 2, the hybrid automaton H which models the multi-modal system is defined as a collection $H = (Q \times X, \Sigma, Y, Init, f, h, I, G, R)$ where $Q = \{q_1, q_2, q_3, q_4, q_5\}$, $X \subseteq \mathbb{R}^3$ and $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$, $Y \subseteq \mathbb{R}^3$, with the hybrid state $(q, x) \in Q \times X$, the input $\sigma \in \Sigma$, and the output $y \in Y$. Let

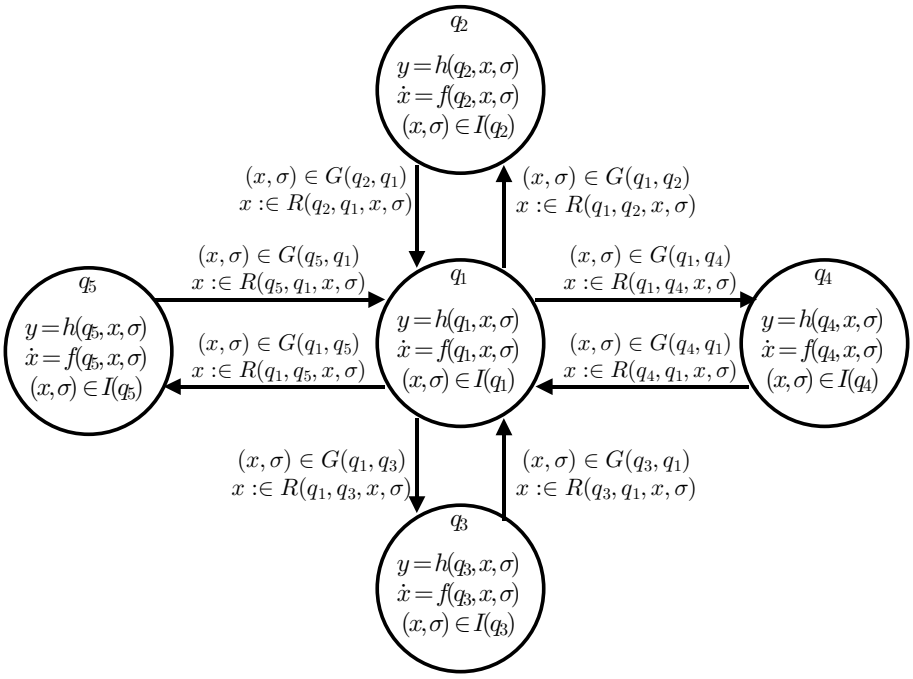

Fig. 2. Hybrid automaton models the multi-modal behaviors of an agent.

$x = [p_x, p_y, p_z]$ to specify the location of an agent in the state space X. The

vector field is defined by

$$f(q,x,\sigma) = \begin{cases} (0,0,0) & \text{if } q = q_1, \\ (\epsilon_2,0,0) & \text{if } q = q_2, \\ (-\epsilon_3,0,0) & \text{if } q = q_3, \\ (0,\epsilon_4,0) & \text{if } q = q_4, \\ (0,-\epsilon_5,0) & \text{if } q = q_5 \end{cases}$$

where $\epsilon_i > 0$ for $i = 2,3,4,5$. The output map is defined by $h(q,x,\sigma) = x$ for $q \in Q$. The invariant is defined by

$$I(q_i) = X \times \{\sigma_i\} \quad \text{for } i = 1,\ldots,5.$$

The guard and the reset relation are defined by

$$\begin{cases} G(q_i,q_j) = X \times \{\sigma_j\} \\ R(q_i,q_j,x) = \{x\} \end{cases}, \qquad \text{for } (i,j) \in \{(1,2),(2,1),(1,3),(3,1) \\ (1,4),(4,1),(1,5),(5,1)\}.$$

The initial set is defined by $Init = \{q_1\} \times X$. When the multi-modal system is in q_1 mode, since the vector field is a zero vector the continuous state x remains the same. If there is an input σ_2, the guard $G(q_1,q_2)$ is enabled and the discrete state becomes q_2 and p_x keeps increasing while p_y, p_z remain the same. This is because in control mode q_2, the first component of the vector field is a positive number and the other components are zero. If we use the North-East-Down coordinate system for defining x,y,z axes, then when the system is in q_2 mode the agent is moving in north direction and hence when the system is in q_4 mode the agent is moving in east direction. The situations are similarly defined for the system being in q_3 and q_5 modes.

After defining the motion capability of an agent, we are interested in the issues related to reachability. Consider $x', x'' \in X$, σ_i-*labeled transition* is defined as $x' \xrightarrow{\sigma_i} x''$ iff there exists $\delta \geq 0$, and a curve $x : [0,\delta] \to \mathbb{R}^n$ with $x(0) = x', x(\delta) = x''$ and for all $t \in [0,\delta]$ it satisfies $\dot{x}(t) = f(q_i, x(t), \sigma_i)$. Notice that the continuous transitions are time-abstract transitions, in the sense that the time it takes to reach one state from another is ignored. Now, we define another transition relation which is used for taking a transition from q_1 to q_i then back to q_1. Therefore, after taking the transition, the state of the multi-modal system is always q_1. Consider $x', x'' \in X$, σ_i-*labeled cyclic transition* is defined as $x' \overset{\sigma_i}{\Rightarrow} x''$ iff $x' \xrightarrow{\sigma_i} x'' \xrightarrow{\sigma_1} x''$. The above definitions of transition relations are motivated by the similar definitions defined in [1] for timed automata.

2.2 Partition and Its Induced Equivalence Relation

Having the transition relations defined, we can start discussing the partition of the continuous space. To address this issue, we consider an equivalence relation $\sim$ over the state space X. Consider that the continuous space X is decomposed into a finite number of cells X_j for $j = 1,\ldots,m$ and we denote the family of subsets X as $\pi = \{X_j\}$. Define $\mathcal{I} = \{1,\ldots,m\}$. If we require that each location

of an agent in the space can belong to exactly one cell, π should be a *partition* of X which satisfies the following properties:

$$X = \bigcup_{j \in \mathcal{I}} X_j, \tag{2.1}$$

$$X_i \cap X_j = \emptyset, \qquad \forall i \neq j. \tag{2.2}$$

Therefore, the cells of the partition π *cover* the space X and *do not overlap*. Here, the induced equivalence relation is called *cell equivalence*, which is defined over the space X. For two locations $x', x'' \in X$, $x' \sim x''$ if $\exists j \in \mathcal{I}$ such that $x', x'' \in X_j$.

However, in order to obtain a *stable* partition [1], the motion capability of an agent has to be taken into consideration. Consider the multi-modal system, the partition is designed by putting a two-dimensional grid over the state space and each boundary of a cell is parallel to exactly one possible motion direction. Therefore, we obtain a partition π composed of rectangular cells and each cell is a Cartesian product of two half-open intervals. The partition is depicted in Figure 3.

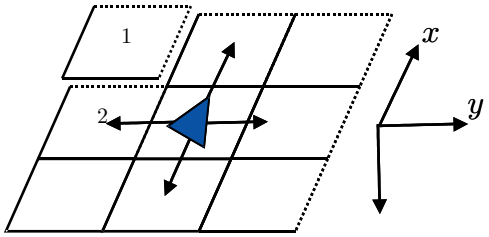

Fig. 3. Graphical illustration of the partition and the possible agent motion directions.

Given a cell $X_j \in \pi$, define $\mathcal{I}_j = \{i \in \mathcal{I} \setminus \{j\} | \ |\partial X_j \cap \partial X_i| > 1\}$. Thus, every $X_i \in \pi$ is *adjacent* to X_j for $i \in \mathcal{I}_j$. Hence we have the following lemma concerning the local motion capability of the agent.

Lemma 1 (Local Motion Capability). *Given a cell $X_j \in \pi$ and an adjacent cell $X_i \in \pi$ with $i \in \mathcal{I}_j$, $\exists \sigma \in \Sigma$ $\forall x' \in X_j$ $\exists x'' \in X_i$ such that $x' \overset{\sigma}{\Rightarrow} x''$.*
Proof: Given the partition, due to the definition of adjacent cells, there are only four possible adjacent cells for each cell. For an adjacent cell, since there exists exactly one motion direction that parallel to each boundary, one can simply pick a motion direction that could make an agent go towards the adjacent cell. Due to the simple reachability property of the multi-modal system, one can easily show that an agent could start from any where within the cell and could reach some where inside the adjacent cell in some time.

Therefore, if an agent starts at any location $x' \in X_j$, then it can move to any adjacent cells X_i of cell X_j and reach some location $x'' \in X_i$ in some time. By

construction, the reachability computation of the hybrid automaton is greatly simplified. Therefore, sophisticated reach set computation is avoided.

3 Bisimulation

Consider an agent starting from a location in a cell $X_S \in \pi$ and we are interested in determining whether it can reach a final cell $X_F \in \pi$. Now, we define a transition system which preserve the reachability property of the multi-modal system. A transition system $T = (X, \Sigma, \Rightarrow, X_S, X_F)$ is constructed to consist of a set X of states, an alphabet Σ of events, a transition relation $\Rightarrow \subseteq X \times \Sigma \times X$, a set $X_S \subseteq X$ of initial states, and a set $X_F \subseteq X$ of final states. The σ_i-labeled cyclic transition $(x', \sigma_i, x'') \in \Rightarrow$ is simply denoted as $x' \Rightarrow x''$. The transition system is infinite since the cardinality of X is infinite. Given an equivalence relation $\sim \subseteq X \times X$ which partitions the state space into a number of equivalence classes. Let $X/\sim = \{X_j\}$ denote the quotient space. For a region X' we denote by $X'/\sim$ the collection of all equivalence classes which intersect X'. If a set is a union of equivalence classes, it is called a $\sim$-block.

Definition 1 (Bisimulation). *Given $T = (X, \Sigma, \Rightarrow, X_S, X_F)$, and $\sim$ an equivalence relation over X, $\sim$ is called a bisimulation if:*

1. *X_S is a union of equivalence classes;*
2. *X_F is a union of equivalence classes;*
3. *For all $\sigma \in \Sigma$, if X' is a $\sim$-block, $Pre_\sigma(X') = \{x' \in X | \exists x'' \in X' : x' \Rightarrow x''\}$ is a $\sim$-block.*

By using Definition 1, we can show that the equivalence relation defined in previous section is a bisimulation.

Theorem 1 (Bisimulation). *The equivalence relation $\sim$ is a* bisimulation*.*
Proof: By construction, each cell in the partition π is an equivalence class. Therefore, X_S, X_F are $\sim$-blocks since $X_S, X_F \in \pi$. For all $\sigma \in \Sigma$ and for every $\sim$-block X' , the predecessor set defined by $Pre_\sigma(X')$ is a $\sim$-block since by Lemma 1 the predecessor can only be the union of all adjacent cells of X'. Hence the result.

The complexity of the reachability problem is reduced by using special quotient transition systems. The quotient transition system is defined as $T/\sim \; = (X/\sim, \Sigma, \Rightarrow_\sim, X_S/\sim, X_F/\sim)$ where the transition relation $\Rightarrow_\sim$ on the quotient space is defined as follows: for $X_1, X_2 \in X/\sim$, $X_1 \Rightarrow_\sim X_2$ iff there exists $x' \in X_1$ and $x'' \in X_2$ such that $x' \Rightarrow x''$. $T/\sim$ is a reachability preserving quotient system. Since the cardinality of $X/\sim$ is finite, $T/\sim$ is called *finite*. Furthermore, we have the following result.

Theorem 2 (Bisimular Systems). *T and $T/\sim$ are* bisimilar*.*

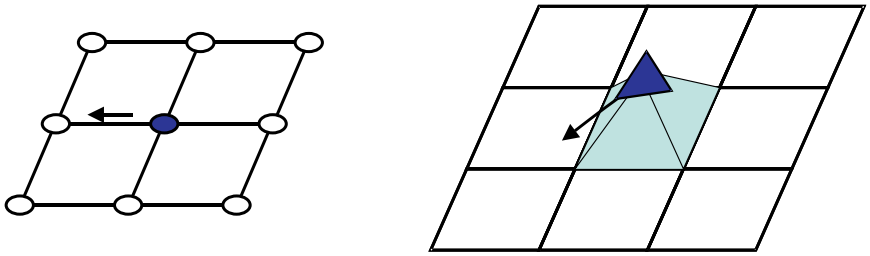

Fig. 4. $T/\sim$ and T are bisimilar.

T and $T/\sim$ accept the same language [6]. Furthermore, the checking reachability for the detailed system T can be equivalently performed on the finite, discrete, quotient graph. Since the quotient graph is finite, the reachability algorithm will terminate. Given the partition, since the equivalent reachability problem on the finite graph can be performed in a computationally feasible way, the reachability problem for the hybrid system is decidable.

In the next section, we will show how to use the quotient transition system to construct a hierarchical system architecture for the single-agent multi-modal system.

4 System Architecture

In this section, we are interested in the design of a hierarchical system architecture for the single-agent multi-modal system by considering the transition system T and its reachability preserving quotient transition system $T/\sim$. Hence, a two-layered system architecture is naturally suggested. The abstracted system, N, on the top layer is associated with $T/\sim$ whereas the detailed system, M, on the bottom layer is associated with T. The hierarchical system F is the composition of the two systems, *i.e.* $F = N \| M$.

However, there are two technical issues have to be addressed before the design of the hierarchical system can take place. First, two transition systems have different notions of time. This is because the continuous transitions defined are *time-abstract*. However, it does take some time for the continuous state to make a transition. Therefore, although T and $T/\sim$ accept the same language, the time accepting an event could be badly mismatched. Second, the use of transition systems is mainly for reachability analysis but they are not suitable to be used as system models to perform actual computation.

For the first issue, we suggest to use a synchronization scheme for keeping a close correspondence between two layers so that state transitions can be synchronized. Next, since $T/\sim$ is a purely discrete transition system, it literary suggests

that finite automaton would be sufficient to be used to represent the quotient transition system for performing actual computation, and since T exhibits hybrid behaviors, hybrid automaton could be used to model the transition system. Therefore, the proposed hierarchical system is composed of a finite automaton and a hybrid automaton. To simplify future discussion on system composition, we will use the hybrid formalism to express the finite automaton. Regarding the formal treatment of the composition of hybrid automata, please refer to [2].

4.1 A Design Example

In this design example, we assume that there are only 4 equivalence classes in a partition and initially an agent is located at $x \in X_S = X_1$.

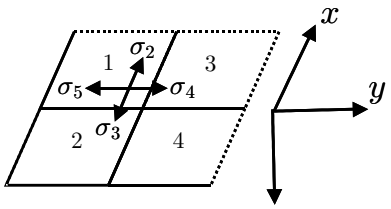

Fig. 5. Four equivalence classes, namely X_1, X_2, X_3, X_4, are illustrated. An agent is located in X_1 and it is capable of moving in four different directions

For the quotient transition system $T/\sim$, we associate it with a hybrid automaton N which is a collection $N = (S, P{\times}\Sigma, S{\times}\Sigma, Init_N, f_N, h_N, I_N, G_N, R_N)$ where the state $s \in S = \{s_1, s_2, s_3, s_4\}$, the input $(p, \sigma) \in P \times \Sigma$ with $P = \{p_1, p_2, p_3, p_4\}$, the output $(s, \delta) \in S \times \Sigma$, the initial set $Init_N = \{s_1\}$, the vector field $f_N(s, p, \sigma) = \emptyset$, the output map $h_N(s, p, \sigma) = (s, \sigma)$, the invariant $I_N(s) = \emptyset$, the reset map $R_N(s_i, s_j) = \emptyset\ \forall i, j \in \{1, 2, 3, 4\}$, and the guard $G_N(s_i, s_j) = \{p_i\} \times \{\delta_{ij}\}$ with

$$\delta_{ij} = \begin{cases} \sigma_2 & \text{for } (i,j) \in \{(2,1),(4,3)\} \\ \sigma_3 & \text{for } (i,j) \in \{(1,2),(3,4)\} \\ \sigma_4 & \text{for } (i,j) \in \{(1,3),(2,4)\} \\ \sigma_5 & \text{for } (i,j) \in \{(3,1),(4,2)\} \end{cases}.$$

Each state s_i represents an equivalence class X_i for $i \in \{1, 2, 3, 4\}$. For each s_i, there is also a corresponding state p_i provided from the bottom layer as an input to N for synchronizing the two layers.

For the transition system T, we associate it with a hybrid automaton M. Since *σ_i-labeled cyclic transition* is introduced in the construction of the transition system, the hybrid automaton H has to be augmented in order to be able to accept the same language as T does. Consider M is composed of two

hybrid automata K and H, *i.e.* $M = K \| H$. Hybrid automaton K is a collection $K = (P, X \times \Sigma, P \times \Sigma, Init_K, f_K, h_K, I_K, G_K, R_K)$ where the state $p \in P = \{p_1, p_2, p_3, p_4, p_{12}, p_{21}, p_{13}, p_{31}, p_{42}, p_{24}, p_{43}, p_{34}\}$, the input $(x, \delta) \in X \times \Sigma$, the output $(p, \sigma) \in P \times \Sigma$, the initial set $Init_K = \{p_1\}$, the vector field $f_K(p, x, \delta) = \emptyset$, the invariant $I_K(p) = \emptyset$, the reset map $R_K(p_i, p_j) = \emptyset$ $\forall i, j \in \{1, 2, 3, 4\}$, the output map $h_K(p_i, x, \delta) = (p_i, \sigma_1)$ $\forall i \in \{1, 2, 3, 4\}$,

$$\begin{array}{l} h_K(p_{12}, x, \delta) = (p_1, \sigma_3),\ h_K(p_{21}, x, \delta) = (p_2, \sigma_2), \\ h_K(p_{13}, x, \delta) = (p_1, \sigma_4),\ h_K(p_{31}, x, \delta) = (p_3, \sigma_5), \\ h_K(p_{42}, x, \delta) = (p_4, \sigma_5),\ h_K(p_{24}, x, \delta) = (p_2, \sigma_4), \\ h_K(p_{43}, x, \delta) = (p_4, \sigma_2),\ h_K(p_{34}, x, \delta) = (p_3, \sigma_3), \end{array}$$

and the guard $G_K(s_i, s_{ij}) = \{\delta_{ij}\}$ with

$$\delta_{ij} = \begin{cases} \sigma_2 & \text{for } (s_i, s_{ij}) \in \{(p_2, p_{21}), (p_4, p_{43})\} \\ \sigma_3 & \text{for } (s_i, s_{ij}) \in \{(p_1, p_{12}), (p_3, p_{34})\} \\ \sigma_4 & \text{for } (s_i, s_{ij}) \in \{(p_1, p_{13}), (p_2, p_{24})\} \\ \sigma_5 & \text{for } (s_i, s_{ij}) \in \{(p_3, p_{31}), (p_4, p_{42})\} \end{cases},$$

or the guard $G_K(s_{ij}, s_j) = X_j$ $\forall i, j \in \{1, 2, 3, 4\}$.

To implement the *σ-labeled cyclic transition*, for each possible transition from p_i to p_j an intermediate state p_{ij} are introduced so that two different symbols σ and σ_1 can be emitted via the output to the hybrid automaton H. To explain the idea, the following scenario is considered. Assume that the system starts from p_1 and the output is (p_1, σ_1). Now, there is an input σ_4 and hence an transition from p_1 to an intermediate state p_{13} is enabled. At p_{13}, the output is (p_1, σ_4). Therefore, the symbol σ_4 can enable the evolution of the continuous state in H. Before the condition $x \in X_3$, p_1 is still being indicated at the output. When $x \in X_3$, a transition from p_{13} to p_3 is enabled and the output becomes (p_3, σ_1). Hence, $M = K \| H$ can be used to generate the transition system T.

We have shown the construction of a hierarchical system $F = N \| M$. Since N and M accept the same language, every sequence accepted by N would also be accepted by M. Hence, checking reachability of the system can be equivalently performed on the finite, discrete graph. Furthermore, in execution, the architecture guarantees that two different worlds residing in two different layers are synchronized. Between these two worlds, state information at the bottom level is abstracted at the top level and control information at the top level is refined at the bottom level.

4.2 System Realization

We have described the construction of a hierarchical system architecture which promotes proof obligations while conforming with common design practice. Here, we are interested in the realization of the system design. Control laws and decision-making procedures can be considered as the basic components for the construction of the hierarchical system. To realize a system design, component-based design provides a clean way to integrate heterogeneous models by hierarchical nesting of parallel and serial composition of components. A semantics

gives meaning to components and their interconnections. A collection of semantics models which are useful for system design have been codified in [12] as *models of computation* (MOCs). Here, we outline some of the most useful MOCs, such as continuous-time (CT), finite-sate machine (FSM), and discrete-event (DE). CT models represented by differential equations are excellent for modeling physical systems. Execution in FSM is a strictly ordered sequence of state transitions and FSM models are amenable to in-depth formal analysis. In DE model, an event consists of a value and a time stamp. There is no global clock tick in DE, but there is a globally consistent notion of time.

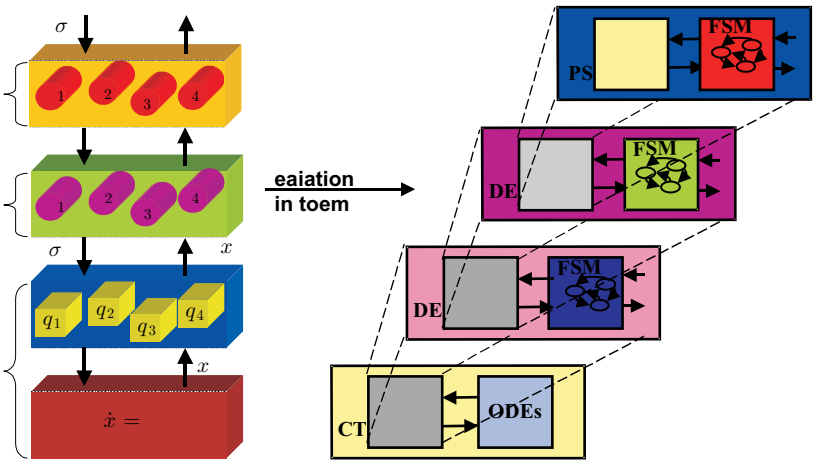

Fig. 6. Hierarchical system architecture for single-agent multi-modal systems and its realization in Ptolemy II [11] by hierarchical aggregating components.

For the proposed hierarchical system architecture for single-vehicle multi-modal systems, we suggest a realization of the system by hierarchical aggregating components such that components are hierarchically refined. As depicted in Figure 6, there are four levels of hierarchy introduced. At each level of hierarchy, a MOC is chosen to govern the interaction between components. The hybrid automaton H is realized by mixing FSM and CT together to exhibit multi-modal behaviors. The interaction between H and K is governed by DE since events

consisting of values and time stamps are exchanged between components. The FSM models the abstracted system N which interacts with the detailed system M in DE domain. Since the interaction between N and any component at higher levels is pure asynchronous, a specific MOC such as publisher-subscriber (PS) could be used.

5 Conclusion

In this paper, a hierarchical system architecture for single-agent multi-modal systems has been proposed. The design principle for the construction of the hierarchy is based on bisimulation and therefore a higher-level system and a lower-level system are bisimilar. The layered system is designed to promote proof obligations so that system specification at one level of granularity *conforms* with system specification at another level and vice versa. Hence, it can be guaranteed that the system is correct by construction with respect to a given set of specifications. Our approach is illustrated on designing a system architecture for executing a mission of controlling a single autonomous agent. We have shown that the construction of a transition system and the corresponding quotient transition system. They both capture the reachability properties of the agent within the environment but at different levels of granularity. Furthermore, we have shown that both systems bisimilar. Thus, one can guarantee that for every action made on the abstracted system there is a refined action on the detailed system; for every event occurred on the detailed system there is an abstracted event on the abstracted system. However, for any concurrent game being played by multiple players, one may have to use alternating simulation[3, 4] for obtaining a useful abstraction. Therefore, with this simulation notion, one can guarantee that for any game being played at the higher-level there is a corresponding game played at the lower-level.

Acknowledgments

The authors would like to thank G. J. Pappas, H. J. Kim, and R. Majumdar for stimulating discussions and valuable comments. This work is supported by the DARPA SEC grant, F33615-98-C-3614.

References

1. R. Alur and D. Dill. A theory of time automata. *Theoretical Computer Science*, 126:183-235, 1994.
2. R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR)*, pages 74-88, 1997.
3. R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *Proceedings of the Tenth International Conference on Concurrency Theory (CONCUR)*, pages 163-178, 1998.

4. L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR)*, 2001.
5. Datta N. Godbole, John Lygeros, and Shankar S. Sastry. Hierarchical hybrid control: an IVHS case study. In *Proceedings of the 33th IEEE Conference on Decision and Control*, pages 1592-1597, 1994.
6. T.A. Henzinger. Hybrid automaton with finite bisimulstions. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, pages 324-335, Springer-Verlag, 1995.
7. J. P. Hespanha, H. J. Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *Proceedings of IEEE Conference on Decision and Control*, pages 2432-2437, Phoenix, Arizona, December 1999.
8. H. J. Kim, R. Vidal, H. Shim, O. Shakernia, and S. Sastry. A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings of IEEE Conference on Control and Decision*, Orlando, Florida, December 2001.
9. T. J. Koo, G. Pappas, and S.Sastry. Mode switching synthesis for reachability specifications. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Vol. 2034, pages 333-346, Springer Verlag, 2001.
10. T. J. Koo, B. Sinopoli, A. Sangiovanni-Vincentelli, and S. Sastry. A formal approach to reactive system design: a UAV flight management system design example. In *Proceedings of IEEE International symposium on Computer-Aided Control System Design*, pages 522-7, Kohala Coast, Hawaii, September 1999.
11. E. A. Lee. Overview of the ptolemy project. Technical Report UCB/ERL M01/11, University of California, Berkeley, 2001.
12. E. A. Lee and A. Sangiovanni-Vincentelli. A Framework for Comparing Models of Computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217-1229, December 1998.
13. J. Lygeros, C. Tomlin, S. Sastry. Controllers for reachability specifications for hybrid systems, Automatica, Volume 35, Number 3, March 1999.
14. N. Megiddo, S. L. Hakimi, M. R. Garey, D.S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18-44, January 1988.
15. A. Pant, P. Seiler, T. J. Koo, and J. K. Hedrick. Mesh stability of unmanned aerial vehicle clusters. In *Proceedings of American Control Conference*, pages 62-68, Arlington, Virginia, June, 2001.
16. T. D. Parsons. Pursuit-evasion in a graph. In Y. Alani and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426-441, Springer-Verlag, 1976.
17. S. Sastry, G. Meyer, C. Tomlin, J. Lygeros, D. Godbole, and G. Pappas. Hybrid control in air traffic management systems. In *Proceedings of the 1995 IEEE Conference in Decision and Control*, pages 1478-1483, New Orleans, LA, December 1995.
18. M.P. Singh. Multiagent systems. A theoretical framework for intentions, know-how, and communications. Berlin, Germany: Springer-Verlag, 1994.
19. P. Varaiya. Smart Cars on Smart Roads: Problems of Control, *IEEE Transactions on Automatic Control*, 38(2):195-207, February 1993.
20. P. Varaiya. A question about hierarchical systems, *System Theory: Modeling, Analysis and Control*, T. Djaferis and I. Schick (eds), Kluwer, 2000.