Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

Proposal for Thesis Research in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

TITLE: A Framework for Modelling and Analysis of Hybrid Systems

SUBMITTED BY: Sayan Mitra
32-G630, 32 Vassar Street
Cambridge, MA 02139

(SIGNATURE OF AUTHOR)

DATE OF SUBMISSION: September 18, 2005

EXPECTED DATE OF COMPLETION: September 2006

LABORATORY: Computer Science and Artificial Intelligence Laboratory

BRIEF STATEMENT OF THE PROBLEM:

In this thesis, we build a formal framework for modeling and analysis of hybrid systems based on the Hybrid Input/Output Automaton (HIOA) model of Lynch, Segala, and Vaandrager. To systematically describe the trajectories of HIOA, we add new structure to it. The result is the *Structured Hybrid I/O Automaton (SHIOA)* model—a specialization of HIOA that allows the description of its trajectories using differential and algebraic equations and inequalities, mode invariants, and stopping conditions. To model probabilistic discrete behavior in hybrid systems we introduce another related model, the *Probabilistic Timed I/O Automaton (PTIOA)*. For each model, we define the notions of invariance, stability, and external or visible behavior of automata. For a pair of automata, we define what it means for one to be an abstraction of the other, and for both to be equivalent. We define composition of a pair of automata, and prove theorems that show that abstraction and equivalence, are preserved under composition.

Based on the SHIOA model we present a formal specification language for non-probabilistic hybrid systems. This language is the basis for the development of a set of software tools that includes a language frontend for checking specifications and a theorem prover interface.

For hybrid systems modeled as SHIOA and PTIOA, we present a set of methods for proving invariance and stability properties of automata, and abstraction and equivalence

relationships between a pair of automata. Invariant properties and abstraction relations can be used to prove safety of hybrid systems; our proof methods for these properties rely on induction on the length of the executions of automata models. Owing to the structure of the models, we can reason about the discrete and the continuous parts of the system independently, within a single inductive proof. We demonstrate these proof techniques with several case studies including a supervisory controller for a helicopter system and a motion coordination algorithm that uses virtual nodes.

For proving stability of SHIOA, we first show that the standard techniques based on Common and Multiple Lyapunov functions can be applied in our framework. Secondly, to prove the stability of a SHIOA whose individual modes of are known to be stable, that is, their Lyapunov functions are known, we use the *Average Dwell Time (ADT)* criterion of Hespanha and Morse. We present two complementary methods—one based on proving invariants and the other based on solving an optimization problem—for proving the ADT-like properties. We show that Lyapunov function based arguments and probabilistic variants of the ADT criteria can be applied to prove stability of PTIOAs. We demonstrate these methods by analysing stability of several hybrid systems, including a scale-independent hysteresis switch.

We develop connections to several software tools for supporting the above proof methods. We present a scheme for translating HIOA specifications to the language of the PVS theorem prover, and a methodology for designing PVS proof strategies. By partially automating proofs, these strategies aid the process of proving invariants and simulation relations. We also present a method for formulating the ADT verification problem for certain classes of SHIOA and PTIOA as mixed integer linear programs, which we solve using the Gnu LP solver kit.

Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
Cambridge, Massachusetts 02139

Doctoral Thesis Supervision Agreement

To: Department Graduate Committee
From: Professor Nancy Lynch

The program outlined in the proposal:

Title: A Framework for Modelling and Analysis of Hybrid Systems
Author: Sayan Mitra
Date: September 18, 2005

is adequate for a Doctoral thesis. I believe that appropriate readers for this thesis would be:

Reader 1: Professor Sanjoy Mitter
Reader 2: Professor Gerald Jay Sussman
Reader 3: Professor Daniel Liberzon

Facilities and support for the research outlined in the proposal are available. I am willing to supervise the thesis and evaluate the thesis report.

Signed: _____
Professor of Electrical Engineering
and Computer Science

Date: _____

Comments:

Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
Cambridge, Massachusetts 02139

Doctoral Thesis Reader Agreement

To: Department Graduate Committee
From: Professor Sanjoy Mitter

The program outlined in the proposal:

Title:	A Framework for Modelling and Analysis of Hybrid Systems
Author:	Sayan Mitra
Date:	September 18, 2005
Supervisor:	Professor Nancy Lynch
Other Reader:	Professor Gerald Jay Sussman
Other Reader:	Professor Daniel Liberzon

is adequate for a Doctoral thesis. I am willing to aid in guiding the research and in evaluating the thesis report as a reader.

Signed: _____
Professor of Electrical Engineering
and Computer Science

Date: _____

Comments:

Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
Cambridge, Massachusetts 02139

Doctoral Thesis Reader Agreement

To: Department Graduate Committee
From: Professor Gerald Jay Sussman

The program outlined in the proposal:

Title: A Framework for Modelling and Analysis of Hybrid Systems
Author: Sayan Mitra
Date: September 18, 2005
Supervisor: Professor Nancy Lynch
Other Reader: Professor Sanjoy Mitter
Other Reader: Professor Daniel Liberzon

is adequate for a Doctoral thesis. I am willing to aid in guiding the research and in evaluating the thesis report as a reader.

Signed: _____
Professor of Electrical Engineering
and Computer Science

Date: _____

Comments:

Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
Cambridge, Massachusetts 02139

Doctoral Thesis Reader Agreement

To: Department Graduate Committee
From: Professor Daniel Liberzon

The program outlined in the proposal:

Title:	A Framework for Modelling and Analysis of Hybrid Systems
Author:	Sayan Mitra
Date:	September 18, 2005
Supervisor:	Professor Nancy Lynch
Other Reader:	Professor Sanjoy Mitter
Other Reader:	Professor Gerald Jay Sussman

is adequate for a Doctoral thesis. I am willing to aid in guiding the research and in evaluating the thesis report as a reader.

Signed: _____

Assistant Professor of Electrical and
Computer Engineering Department
University of Illinois at Urbana-Champaign

Date: _____

Comments:

1 Overview

A formal framework is developed for modeling and analysis of hybrid systems, consisting of mathematical models, proof methods for invariance, stability, abstraction, and equivalence of these models, and a set of software tools for supporting these proof methods. The next section motivates this work. Section 3 provides an overview of previous research in modelling hybrid system. Section 4 summarizes the key contributions of the thesis and Section 5 gives an overview of the thesis. The Hybrid Input/Output Automaton (HIOA) model of Lynch, Segala, and Vaandrager—the mathematical basis of this work is described in Section 5.1. The *Structured Hybrid I/O Automaton (SHIOA)* model is developed by adding new structure to the HIOA model, which allows systematic description of the trajectories. Another related model, the *Probabilistic Timed I/O Automaton (PTIOA)* is introduced to model probabilistic discrete behavior in hybrid systems. These new automata models are described in Section 5.2. Based on the SHIOA model a new formal language for specifying non-probabilistic hybrid systems is described in Section 5.3. Inductive methods for proving invariants and simulation relations for SHIOA along with application of these methods to verification of hybrid systems are presented in Section 5.4. Methods for proving stability of ordinary and probabilistic hybrid systems based on Lyapunov functions, Average Dwell Time criterion of Hespanha and Morse, and the probabilistic extensions of the latter, are presented in Section 5.5. Finally, the software tool support for the above proof techniques are described in Section 5.6. These include an interface to the PVS theorem prover for partially automating invariant and simulation proofs, and a method for verifying ADT properties of certain classes of SHIOA and PTIOA by solving a mixed integer linear programs using the Gnu LP solver kit.

2 Motivation

Real-time and embedded systems are now pervasive, ranging from wrist-top devices to on-board instruments of spacecrafts. Fueled by the growth of the wireless communications and faster CPUs, embedded devices are rapidly growing in complexity, and are being deployed to perform increasingly more critical and sophisticated tasks. Embedded systems are often called *hybrid systems*, because the models used to represent them involve both discrete and continuous time phenomena. For system designers to guarantee a high level of assurance for embedded or hybrid systems a formal framework is necessary. A framework for handling design and analysis of complex hybrid systems should have the following features:

- The mathematical models for representing hybrid systems, the primary objects of

the framework, should be capable of capturing the range of discrete and continuous phenomena that arise in typical applications.

- There should be a well defined notion of external behavior or visible behavior for the models. The notion of external behavior allows us to define what it means for one component to be an implementation of another component, or for two components to be equivalent.

- The framework should be *compositional*. That is, one should be able to build larger and more complex systems by putting together smaller components. These components should be allowed to communicate both discretely and continuously with each other and with the environment. The composition operation should respect the notions of equivalence and implementation. That is, if component A is an implementation of B, then for every environment C, $A||C$ should be an implementation of $B||C$.

- The framework should provide methods for proving the commonly encountered types of properties for the models, like invariant properties, implementation and equivalence relations, and stability. In the course of such proofs, wherever existing analysis techniques from computer science and control theory are useful in practice, it should be possible to apply those techniques within the framework.

- The framework should be suitable as a basis for supporting software tools, and wherever possible, should facilitate automatic analysis.

The traditional literatures in control theory and computer science study hybrid systems from two different perspectives. Computer scientists have modeled hybrid systems as state transition systems augmented with simple differential equations (see e.g., [3, 37]), and have concentrated on safety verification. The safety of an automaton, is specified by a set of invariant properties or as a more abstract automaton. So, safety verification consists of proving invariant properties or abstraction relations, sometimes both. Control theorists, on the other hand, have viewed hybrid systems as dynamical systems with special boolean variables, and have developed techniques for proving stability, controllability, and for synthesizing controllers (see e.g., [48, 80]). We argue that there is a need for a general framwork for hybrid systems that draws from the strengths and insights of both these disciplines and offers all the above features.

In this thesis we develop a framework for specification and analysis of hybrid systems based on the interactive state machine model *Hybrid Input/Output Automaton (HIOA)* of Lynch, Segala, and Vaandrager [55]. A hybrid I/O automaton consists of sets of actions and

variables—both partitioned into input, output, and internal categories, and sets of transitions and trajectories. Hybrid behavior of a system is modeled as an alternating sequence of actions and trajectories, which is called an *execution* of the system. The input and output actions (variables) are used to model discrete (resp. continuous) communication between components. This partitioning of the variables and actions leads to a natural definition for the external behavior or *trace* corresponding to a given execution of the automaton, which is the sequence obtained by removing the internal variables and actions from the execution. An automaton A is said to be an implementation of automaton B if the set of traces of A is contained in that of B. The HIOA framework provides composition theorems which allow us to reason about components of hybrid systems separately and then put them together in a meaningful way. In the thesis we develop methods for proving invariant and stability properties, and for proving abstraction and equivalence relationships between automata.

Apart from the applications to analysis of embedded systems, the theory of hybrid I/O automata and its probabilistic extensions are of independent interest because they pave the way towards a theory of complicated, interacting systems, encompassing discrete, continuous, and probabilistic behavior. These models are essential for improving our understanding of complex systems (selected references: [9, 10, 23, 39, 67]).

3 Related work : Hybrid system models

We give a brief overview of the models for hybrid systems that are available in the literature. We first describe state machine based models and then dercribe a representative model developed form the dynamical systems perspective. In discussing these models we follow the conceptual developments of the models rather than the chronological order in which the models appeared.

3.1 Timed and hybrid automata

Models for qualitative reasoning about concurrent systems have been studied in great detail. Examples include ω-automata [21], modal logics [58, 74], I/O automata [31, 52]. These formalisms either abstract away time completely, retaining only the sequence of events, or assume the time sequence to be a monotonically increasing sequence of integers. These are not entirely satisfactory settings for reasoning about systems that must interact with physical processes and critically depend upon *real-time* constraints.

An earlier timed version of the I/O automata model [54], addressed this problem by associating lower and upper time bounds with the discrete transitions of the automaton. In [59] Merritt, Modugno, and Tuttle generalized this idea to associate time bounds on sets of

actions instead of individual transitions. The result was the *MMT-automaton* model which is a special kind of I/O automaton that has an equivalence relation on the set of internal and output actions. The equivalence classes are called *tasks*, and time bounds are associated with each task. From any point in an execution where a task gets enabled, some action in the task occurs within the corresponding time bounds.

Alur and Dill introduced the *Timed Automaton* model [4], in which it is possible to directly express the time taken by an execution of an automaton, and not just the interval between the successive transitions. A timed automaton is a finite automaton with a finite set of real valued clocks that evolve at constant rates. The clocks can be reset to 0 with the transitions of the automaton, and keep track of the time elapsed since the last reset. The discrete transitions are associated with constraints on the clock values called *guards*. A transition may be taken when the corresponding guard constraint is satisfied by the current clock values.

The Alur, Henzinger, et. al. *Hybrid Automaton (AH)* model [3, 34] generalizes the Alur-Dill timed automaton model so that the continuous variables do not necessarily evolve at constant rates. The AH model is defined by a set of continuous variables and a graph with vertices (called *locations*) and edges. The state of a AH hybrid automaton consists of the location and the values of the continuous variables. Each edge defines a discrete transition and is associated with a label, a guard, and a reset map. Each location is associated with an invariant and a set of differential equations. At a given location the values of the continuous variables evolve according to the differential equations of that location. A discrete transition corresponding to an outgoing edge of the location may be taken when the guard of the edge is satisfied by the current values of the continuous variables. The result of a discrete transition is a change of the location and a resetting of the continuous variables according to the reset map of the edge. The AH framework provides a parallel composition operator for putting together pairs of automata.

The Alur-Dill timed automaton and the Alur-Henzinger hybrid automaton models have been the basis for a large body of research on formal-language theoretic study of hybrid systems, and for the development of algorithms for automatic analysis (selected references: [2, 3, 36, 38]). Several successful software tools have been built based on the developed algorithms (see, e.g., [10, 35]).

On the other hand, the focus of the timed and hybrid I/O automaton models have been to develop a general framework with well defined notion of external behavior, parallel composition and abstraction. Several timed extensions of the basic I/O automaton model had been presented in [24, 57, 59] before the final version of the *timed I/O automaton (TIOA)* model appeared in [45, 46]. A TIOA is a nondeterministic, possibly infinite state, state

machine. The state of a TIOA is described by a valuation of a set of *internal* variables. The state of a TIOA can change in two ways: instantaneously by the occurrence of a discrete transition, which is labeled by a discrete action, or according to a *trajectory* which is a function that describes the evolution of the variables over intervals of time. An *execution* of the system is given by an alternating sequence of actions and trajectories.

The actions of a TIOA are classified as input, output, or internal, which provides a natural definition for external behavior. The external behavior of a TIOA is defined by a *trace*—essentially, a sequence of the input/output actions interspersed with trajectories that only capture the passage of time. For abstraction, the TIOA framework defines what it means for one TIOA, $\mathcal{A}$, to *implement* another TIOA, $\mathcal{B}$, namely, that any trace of $\mathcal{A}$ is also a trace of $\mathcal{B}$. The notion of a *simulation relation* from $\mathcal{A}$ to $\mathcal{B}$ provides a sufficient condition for proving the $\mathcal{A}$ implements $\mathcal{B}$. The framework provides a *composition operation*, by which TIOAs modelling individual timed components can be combined to produce a model for a larger timed system. The model for the composed system can describe interaction among components through simultaneous participation in input/output actions. The *Hybrid I/O automaton (HIOA)* [55, 56] model generalizes the TIOA model by allowing components to communicate continuously through shared input and output variables. Accordingly, a trace of a HIOA contains input/ouput actions and trajectories of the input/ouput variables. Composition, implementation, and simulation relations for HIOA are defined analogously to accomodate this more general notion of traces. The timed and hybrid I/O automaton models have been used to verify safety of several hybrid systems through inductive invariant proofs and simulation relations (see, for example, [30, 47, 50, 53, 81, 82], and [33]). Supporting software tools for TIOAs and composed HIOAs are available [43], which include a frontend for the specification language, a simulator, and an interface to the PVS theorem prover [49].

The Alur-Dill timed automaton, AH, TIOA, and HIOA models are all closely related. AH generalizes the Alur-Dill timed automaton model and HIOA generalizes TIOA. In [46] it has been shown that the Alur-Dill timed automaton model is a specialization of the timed I/O automaton model. In this thesis we show that, AH and TIOA, in fact have the same expressive power and therefore HIOA subsumes AH. In particular, (1) HIOA provides enough structure to capture interesting discrete behavior and not just mode changes of a hybrid system. The discrete state of a HIOA may have more information than just the mode of the system. And the discrete transitions of HIOA may modify its discrete state to perform interesting computations and not just bring about mode changes. (2) Component HIOAs may communicate not only discretely through shared actions but also continuously through shared variables. In the AH model, automaton components communicate through shared

actions and may evolve over a common set of variables, but there is no structure on the variables to model continuous communication between components. The extra expressive power of HIOAs does not necessarily imply that HIOA is a better model than AH. For example, the algorithmic analysis techniques that have been developed for AH are not applicable to general HIOA. Regarding external behavior and abstractions, there is no inherent notion of external behavior for the AH model: AH automata do not distinguish between internal state variables and externally visible variables, nor does it distinguish between internal and external transitions. There are however, several types of abstractions for AH automata, for example, discrete abstraction [5], predicate abstraction [79], each of which rely on some implicit notion of external behavior. For example, in [5] it has been shown that for certain classes of AH automata, a discrete abstraction may be obtained that is equivalent to the original with respect to temporal logic formulas.

In this thesis, our focus is on developing a variety of proof methods, not just automatic ones, for general classes of hybrid systems that arise in practice. Therefore the HIOA model is suitable for our purpose because of its expressive power, inherent notion of external behavior, and the availability of simulation relation based proof methods for implementation relations.

3.2 General hybrid dynamical systems and switched systems

The *General Hybrid Dynamical System (GHDS)* and *Controlled GHDS (CGHDS)* models were introduced by Branicky in [13]. These models subsume and unify several other models for hybrid systems, including those proposed in [6, 8, 16, 68], and the AH hybrid automata model. GHDS represents a hybrid system as an interacting collection of dynamical systems, each evolving on continuous-variable state spaces, with switching among systems occurring at autonomous jump times, when the state is in a specified subset of the constituent state spaces. The set of continuous variables of each constituent dynamical system may be different. In each of the constituent dynamical system, the dynamics of the continuous variables may be continuous time, discrete time, or mixed, and are generally given by difference or differential equations. The *switched system* model [41, 48] is a special case of the GHDS model where all the constituent dynamical systems have the same state space and the right hand side of the differential equations defining the dynamical systems are globally Lipschitz continuous. Switched systems may also be viewed as higher-level abstractions of hybrid systems, obtained by neglecting the details of the discrete behavior and instead considering switching signals from a suitable class, that brings about the switching between the different dynamical systems. CGHDS adds the possibility of continuous control of each constituent dynamical system, and the ability to discontinuously reset the state variables when the state

is in another specified subset of the constituent state spaces.

The GHDS model has been used to compare computational capabilities of discrete, continuous, and hybrid systems, and for developing analysis tools for limit cycle existence, robustness, and stability. In [15] the CGHDS model is used to develop algorithms for controller synthesis in an optimal control framework. The switched system model has been widely used to obtain many results related to stability of hybrid systems (see, e.g., [48]). The basic tool for studying stability of switched systems relies on the existence of a *Common Lyapunov function* whose derivative along the trajectories of all the modes satisfies the suitable inequalities. When such a function is not known or does not exist, *Multiple Lyapunov* functions [14] are useful for proving stability of a chosen execution. The *dwell time* [65] and the *average dwell time* criteria of Morse and Hespanha [41] define restricted classes of switching signals, based on switching speeds, and one can conclude the stability of a system with respect to these restricted classes. More recently, Liberzon and Chatterjee [17, 18] have applied Lyapunov function type arguments to derive stability conditions for stochastic hybrid systems.

The GHDS framework does not explicitly provide a notion of external behavior, though it has been pointed out that a GHDS can be refined by adding inputs and outputs. We are not aware of any study that develops the theory of abstraction relations for GHDS. One advantage of the hybrid I/O automaton model over GHDS, is that it provides enough structure to inductively prove invariants and simulation relations. From an applicability point of view, the mathematical machinery and tools based on the GHDS framework have not been translated to supporting linguistic or software tools.

3.3 Models for probabilistic hybrid systems

Probabilistic or stochastic models are used to capture uncertainties about the system model. Uncertainties may affect the behavior of a hybrid system model, in many different ways. For example, there might be uncertainties about the outcome of the discrete transitions, the parameters of the differential equations might be uncertain, or there may be some white-noise-like disturbance affecting the continuous evolution. Consequently, various models for probabilistic hybrid systems are possible.

In [42] the transitions between the modes of the system are guided by a discrete time Markov chain, and within each mode the evolution of the continuous variables is described by stochastic differential equations. In [39], a model for stochastic hybrid systems is proposed where transitions between discrete modes are triggered by transitions between states of a continuous-time Markov chain and the rate at which transitions occur is allowed to depend both on the continuous and the discrete states. Both these models develop the theory for

finding invariant distributions over the state space. These models do not define external behavior or composition of model components.

Probabilistic extensions of I/O automata were presented by Segala in [75–77]. The notion of traces is generalized to *trace distributions* that define the external behavior of a probabilistic automaton. Each trace distribution is induced by a probabilistic scheduler which resolves all nondeterministic choices. These extensions are natural, but the resulting abstraction relations are not compositional. Difficulties arise from the interaction between probabilistic choice and the resolution of nondeterminism of the model. This is because nondeterministic choices are resolved by a powerful global scheduler, which can use arbitrary information about the execution so far in resolving nondeterministic choices. For example, such a scheduler may resolve nondeterministic choices in one automaton component in a composed system based on internal state information of the other component.

More recently, a special case of PIOAs, *switched PIOA*s, has been proposed in [19] in which these difficulties have been overcome by carefully defining a local scheduler for each automaton, which resolves local nondeterministic choices using local information only.

4 Key contributions of the thesis

We believe that the HIOA framework provides suitable compositionality and abstraction theorems and proof techniques, for hierarchical and component-wise analysis of hybrid systems. Based on the HIOA model we propose a unified framework for modelling and analysis of both ordinary and probabilistic hybrid systems. The framework provides new mathematical models for ordinary and probabilistic hybrid systems. It gives methods for proving invariance, stability, abstraction, and equivalence of hybrid systems, and a set of software tools supporting these proof techniques.

Mathematical models. To systematize the description of the trajectories of a hybrid I/O automaton, we specialize the HIOA model by adding extra structure to it. This allows us to describe the trajectories using *state models* consisting of differential and algebraic equations. We call the resulting model *Structured HIOA*. A special case of this model that does not have input and output variables, and does not distinguish between input, output, and internal actions, is the *Structured Hybrid Automaton (SHA)*. We use the SHA model to develop methods for proving internal stability of non-probabilistic hybrid systems.

We will also present a new model for probabilistic hybrid systems called *Probabilistic Timed I/O Automata (PTIOA)*. This is a compositional model that captures non-probabilistic continuous behavior, and both probabilistic and non-deterministic discrete behavior. Analogous to the developments for non-probabilistic hybrid systems, for inter-

nal stability analysis of probabilistic hybrid systems, we will work with a special case of PTIOA—*Probabilistic Timed Automaton (PTA)*—which is purely probabilistic (all nondeterministic choices are resolved) and does not distinguish among input, output, and internal actions.

We establish the relationship among all the new models and the existing models in the I/O automaton family. For the new models we formally define the following notions:

- *Invariant property:* A condition that is satisfied in all reachable states.
- *Stability property:* Starting form any state, every execution converges to some equilibrium state.
- *Abstraction and equivalence relations:* One system is an implementation of another; two systems are equivalent with respect to external behavior.
- *Composition operation:* An operation for putting two system components together.

And we will prove the following types of theorems:

- *Abstraction or equivalence through simulation relations:* We define *simulation relations* on the state spaces of a pair of automata A and B, that provide sufficient conditions for establishing that A is an implementation of B, or that A and B are equivalent with respect to their external behavior.
- *Closure under composition:* If A and B are automata of a particular type, then their composition $A||B$ is also an automaton of the same type.
- *Composition preserves abstraction:* If A is an implementation of B, then for every environment C, $A||C$ is an implementation of $B||C$.

HIOA language. We present a formal language called HIOA for specifying non-probabilistic hybrid systems. HIOA is based on the SHIOA mathematical model, and is an extension of the IOA language [31]. It provides constructs for describing the trajectories of a hybrid system in a natural way using differential and algebraic equations, mode invariants, and stopping conditions. We give the syntax and the semantics of the HIOA language and illustrate its usage with several examples. This language is used throughout the thesis for describing the hybrid systems we use as case studies.

Proof methods for invariant properties and simulation relations. We present methods for proving invariant properties of SHIOA and implementation relationships between a pair of automata, SHIOA or PTIOA. The proof methods rely on induction over the length

of the executions of the automaton (or the pair of automata) in question, followed by a case analysis of actions and trajectories. Such inductive methods allow us to reason about the discrete and the continuous parts of the execution independently of each other. This feature of the model facilitates the use of available techniques from both control theory and computer science, within a single proof. For example, to prove that a certain closed set is an invariant for a given automaton, we show that the continuous part of the induction can be proved by invoking a well known theorem from control theory on subtangential relationships between the boundary of the invariant set and the vector field of the trajectories.

We illustrate the proof methods with several small examples. We also apply these techniques to verify the correctness of two interesting hybrid systems: a supervisory controller for a model helicopter system, and a distributed motion coordination algorithm that uses *virtual nodes*.

Proof methods for stability. A system is said to be stable if it converges to an equilibrium state starting from *any* state. So, the inductive proof techniques that are available for proving invariance cannot be directly applied to prove stability. In this thesis we focus on internal stability analysis of hybrid systems and therefore work with the SHA and PTA models that distinguish among input, output, and internal actions. We show that the standard techniques based on Common and Multiple Lyapunov functions can be applied to the SHA model.

In cases where the individual modes of the system are known to be stable, that is, their Lyapunov functions are known, we prove the stability of the complete system by verifying an *Average Dwell Time (ADT)* property. We have developed two complementary techniques for proving ADT properties. The first technique relies on transforming the given automaton A to a new automaton A', such that A satisfies the ADT property in question if and only if A' has a certain invariant property. Then the techniques developed above for proving invariant properties can be used on A' to verify the ADT of A. The second technique relies on solving an optimization problem over the set of executions of an automaton, to search for a counterexample execution that violates the candidate ADT property. We show that for *finitely initialized* SHA and certain classes *linear* SHA, it is necessary and sufficient to optimize over a small set of execution fragments, and therefore it is possible to solve the optimization problem efficiently. We apply both the techniques to verify the ADT of several hybrid systems, including a scale-independent hysteresis switch.

We extend both the invariant and the optimization based techniques to verify probabilistic variants of the ADT property for PTA.

Software tools. We have developed the following connections to software tools to support

the various proof methods developed in the thesis. The tools have been applied to analyze several illustrative examples.

- We have designed a scheme for translating HIOA specifications to the language of the PVS theorem prover [49]. This allows us to construct invariant and abstraction proofs using the PVS theorem prover, without having to rewrite the specifications of the hybrid systems in the language of PVS.
- We have developed several PVS strategies for proving two kinds of abstraction relations, refinements and forward simulations, for SHIOA specifications [**?**]. These strategies exploit the known structure of inductive proofs to partially automate the proofs process. The infrastructure we have laid down for developing these strategies will simplify the development of other strategies.
- We present a methodology for formulating the ADT verification problem of a certain classes of SHA and PTA as Mixed Integer Linear Programs (MILP), which allows us to verify ADT properties automatically using MILP solvers.

5 Thesis overview

In this section we provide an overview of the proposed thesis. We describe the hybrid I/O automaton framework of Lynch, Segala, and Vaandrager, and introduce the new mathematical models that are derived from it. We highlight the important features of the HIOA specification language. We present the proof methods for invariant properties and abstraction relations, and describe the two major case studies that rely on these methods. We describe the proof methods for stability based on ADT properties of ordinary and probabilistic hybrid systems. Finally, we describe the connections to the various software tools.

5.1 Hybrid I/O automata framework

We fix the *time axis* T to be $\mathsf{R}_{\geq 0}$. A variable in this framework represents either a state component of the system, or a named channel through which information flows from one system component to another. Let V be a set of state variables. Each variable $v \in V$ is associated with a *static type*, denoted by $type(v)$, which is the set of values that v can assume, and a *dynamic type*, denoted by $dtype(v)$, which is a set of functions from left-closed intervals in T to $type(v)$ that is closed under time-shift, restriction to subintervals, and pasting. We say that a variable $v \in V$ is it is *discrete* if $dtype(v)$ is the pasting closure of the set of constant functions; it is *continuous* if it is not discrete and $dtype(v)$ is the

pasting closure of the set of continuous functions; Given a set of variables V, the discrete and the continuous subsets will be denoted by V_d and V_c.

A *valuation* $\mathbf{v}$ for the set of variables V is a function that associates each variable $v \in V$ to a value in $type(v)$. The set of all valuations of V is denoted by $val(V)$. A *trajectory* τ of V is a mapping $\tau : J \to val(V)$, where J is a left closed interval of time with left endpoint equal to 0, such that for each $v \in V$, $\tau \downarrow v \in dtype(v)$. In other words, a trajectory specifies the values of all state variables on a time interval, with the constraint that their evolution be consistent with their dynamic types. We denote the domain of trajectory τ by $\tau.dom$. The first time of τ is is written as $\tau.ftime$. If $\tau.dom$ is right closed then τ is *closed* and its *limit time* is the supremum of $\tau.dom$ and is written as $\tau.ltime$. The *first valuation* of τ, $\tau.fval$ is $\tau(0)$, and if τ is closed, then the *last valuation* of τ, $\tau.lval$, is $\tau(\tau.ltime)$.

Definition 5.1 *A* hybrid Input/Output automaton *is a tuple $\mathcal{A}$ consists of :*

1. *a set V of variables, partitioned into* internal X, input U, *and* output variables Y.
2. *a set A of actions, partitioned into* internal H, input I, *and* output actions O.
3. *a set of states $Q \subseteq val(X)$,*
4. *a non-empty set of* start states $\Theta \subseteq Q$,
5. *a set of* discrete transitions $\mathcal{D} \subseteq Q \times A \times Q$.
6. *a set of* trajectories $\mathcal{T}$ *for V, such that $\mathcal{T}$ is closed under prefix, suffix, and concatenation.*

A hybrid I/O automaton satisfies two axioms: **E1** input action enabling*: input actions cannot be blocked, and* **E2** input trajectory enabling*: the automaton is able to accept any trajectory of the input variables either by allowing time to progress for the entire length of the trajectory or by reacting with some action in $H \cup O$.*

Executions, traces, invariance, and stability. An execution of an automaton models a run or a particular hybrid behavior of the system. An *execution fragment* of $\mathcal{A}$ is an alternating sequence of actions and trajectories $\alpha = \tau_0, a_1, \tau_1, a_2 \ldots$, such that the first and last states of the trajectories "match up" with the transitions. The *first state* of an execution fragment α, $\alpha.fstate$, is $\tau_0.fstate$. An execution fragment is *closed* if the sequence is finite and the domain of the final trajectory is a finite closed interval. The *length* of an execution fragment is the number of actions and trajectories in the sequence. The *limit time* of α, $\alpha.ltime$, is defined as $\sum_i \tau_i.ltime$. An execution fragment α is an *execution* of $\mathcal{A}$

if $\alpha.fstate \in \Theta$. A state is said to be $\mathbf{x} \in Q$ of $\mathcal{A}$ is *reachable* if it is the last state of some execution of $\mathcal{A}$. An *invariant* $\mathcal{I}$ of $\mathcal{A}$ is a condition on X that remains true in all reachable states of $\mathcal{A}$.

Stability is a property of the continuous variables X_c of $\mathcal{A}$. Given a state $\mathbf{x} \in Q$, we define $|\mathbf{x}|$ to be the standard Euclidean norm of the valuation of X_c at $\mathbf{x}$. We say that $\mathcal{A}$ is *uniformly stable with origin as the equilibrium point* if for every $\epsilon > 0$, there exists a $\delta > 0$, such that for every execution fragment α, $|\alpha.fstate| \leq \delta$ implies $|\alpha.lstate| \leq \epsilon$.

The external behavior of a hybrid I/O automaton is captured by the set of *traces* of its executions, which record input/output actions and the trajectories that describe the of input/output variables. Two hybrid I/O automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to *comparable* if they have the same external interface, that is, $U_1 = U_2$, $Y_1 = Y_2$, $I_1 = I_2$, and $O_1 = O_2$. An important feature of this framework is the formal notion of abstraction or implementation through behavioral containment. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable then we say that $\mathcal{A}_1$ *implements* $\mathcal{A}_2$ if the traces of $\mathcal{A}_1$ are included among those of $\mathcal{A}_2$.

The hybrid I/O automaton framework derives its strength from powerful compositionality results. For example, two "well behaved" hybrid I/O automata $\mathcal{A}_1$ and $\mathcal{A}_2$ can be composed and the resulting object is also an automaton of the same type. Moreover, if $\mathcal{A}_1$ implements $\mathcal{A}_2$, and $\mathcal{B}$ is well behaved with respect of both $\mathcal{A}_1$ and $\mathcal{A}_2$, then A_1 composed with $\mathcal{B}$ implements A_2 composed of $\mathcal{B}$.

5.2 New mathematical models

The hybrid I/O automata model of [55] does not specify any means for describing the trajectories of an automaton. For developing analysis techniques that rely on precise nature of the trajectories, we incorporate extra structure in the model to allow specification of the trajectories in a systematic way. The result is the *Structured HIOA (SHIOA)* model which specializes the hybrid I/O automaton model, by adding *state models* for describing the trajectories of the automaton. In this proposal, for brevity and to avoid extra notation, we present the *Structured Hybrid Automaton (SHA)* SHA model which is a version of the SHIOA, that does not have input and output variables, and does not distinguish between input, output, and internal actions. We also present a brief overview of the new models for probabilistic hybrid systems.

5.2.1 Structured Hybrid Automata.

For a trajectory τ for a set of variables V, with some abuse of notation we use the variable name $v \in V$ to denote the function $\tau \downarrow v : \tau.dom \rightarrow type(v)$. So, for any $t \in \tau.dom$, $v(t)$ denotes $(\tau \downarrow v)(t)$. For some $v \in V$, and a function $f : val(V) \rightarrow \mathsf{R}$, τ *satisfies* the

algebraic equation $v = f$ if for every $t \in \tau.dom$, $v(t) = f(t)$, where $f(t)$ is the value of the function f at $\tau(t)$. If f is integrable when viewed as a function of time over the domain of τ, then we say, that τ *satisfies* the differential equation $d(v) = f$ if for every $t \in \tau.dom$, $v(t) = v(0) + \int_0^t f(t')dt'$. This notation can be extended to inequalities and differential inclusions.

Definition 5.2 *A* state model *F for a set of variables X, is a set of differential equations for X_c, of the form $\dot{\mathbf{x}}_c = f(\mathbf{x}_c)$, such that for every $\mathbf{x} \in val(X)$, there exists a trajectory τ starting from $\mathbf{x}$, with the property that $\tau \downarrow X_c$ satisfies F, and for all $t \in \tau.dom$, $\tau \downarrow X_d(t) = \tau \downarrow X_d(0)$. The prefix and suffix closure*[1] *of the set of trajectories of X that satisfy the above conditions is denoted by $traj(X, F)$.*

Definition 5.3 *A* Structured Hybrid Automaton *(SHA) is a tuple $\mathcal{H} = (X, Q, \Theta, A, \mathcal{D}, P)$, where*

1. *X is a set of* variables*, including a special discrete variable called* mode*.*
2. *$Q \subseteq val(X)$ is a of states.*
3. *$\Theta \subseteq Q$ is a non-empty set of* start states*.*
4. *A is a set of* actions*.*
5. *A set $\mathcal{D} \subseteq Q \times A \times Q$ of* discrete transitions*. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is a* mode switch *if $\mathbf{x}.mode \neq \mathbf{x}'.mode$. The set of* mode switching transitions *is denoted by M. The* enabling predicate *of action $a \in A$ is $Pre_a \triangleq \{\mathbf{x} \in Q \mid \exists \mathbf{x}', (\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}\}$.*
6. *An indexed family P of state models F_i, $i \in \mathcal{P}$, for X. For each F_i, $i \in \mathcal{P}$ and $\mathbf{x} \in Q$ with $\mathbf{x}.mode = i$, there exists a trajectory τ such that if $\tau.dom$ is finite then $\tau.lstate \in Pre_a$ for some $a \in A$.*

The set of *trajectories* $\mathcal{T}_\mathcal{A}$ of SHA $\mathcal{A}$ is defined in terms of the state models $P_\mathcal{A}$ as $\bigcup_{i \in \mathcal{P}_\mathcal{A}} traj(X_\mathcal{A}, F_i)$. Assuming that the right hand sides of the differential equations in the state models are locally Lipschitz, the differential equations have solutions defined globally in time. As a consequence the set of trajectories $\mathcal{T}_\mathcal{A}$ satisfies the input enabling condition **E2**. An SHA is *initialized* if for each mode switching action $a \in M$, there exists a set $R_a \subseteq Q$, such that for every $\mathbf{x} \xrightarrow{a} \mathbf{x}' \in \mathcal{D}$, $\mathbf{x}' \in R_a$. An SHA is *finitely initialized* if $\bigcup_{a \in M} R_a$ is finite. In Section 5.5 we use the SHA model for internal stability analysis of hybrid systems.

[1] A *prefix* of trajectory τ, is obtained by restricting τ to a subset of its domain. A *suffix* of τ is obtained by restricting its domain to a subset $[t, \infty)$ and shifting the restricted trajectory to 0.

A Structured HIOA classifies the actions of a SHA, into input, output, and internal subclasses, and adds input and output variables. The SHIOA model is suitable for input-to-state stability analysis and is also the basis for the development of the HIOA language, which is discussed in Section 5.3. Executions, traces, invariant properties, and implementation relationships of SHIOA are defined in a way that is analogous to the definitions for HIOA. We prove the following composition theorems for SHIOA.

Theorem 5.1 *If A and B are SHIOA, then their parallel composition $A||B$ is also a SHIOA.*

Theorem 5.2 *Given a pair of SHIOAs A and B, If A is an implementation of B, then for every context C, $A||C$ is an implementation for $B||C$.*

5.2.2 Models for probabilistic hybrid systems.

As a compositional model for probabilistic hybrid systems, we develop probabilistic extensions of the timed I/O automaton following the models presented in [76] and [19]. We propose a new model, *Probabilistic Timed I/O Automaton (PTIOA)*, in which the component automata communicate discretely only (i.e., all continuous variables are internal), and the trajectories of the continuous variables are nondeterministic (not probabilistic). We define the notion *trace distributions* for PTIOA. We show how invariant distributions can be proved inductively. Extending the notion of abstraction (resp. equivalence) relation between a pair of PTIOAs, A and B, to be containment (resp. equality) of two sets of trace distributions, we define simulation relations on the state spaces A and B, that are sufficient to inductively prove that A is an implementation of (resp. equivalent to) B. For composition of a pair of PTIOAs, as in [19], the nondeterministic choices within a component are resolved by a local scheduler that relies only on local history of the component, and the inter-component choices are resolved by a global scheduler, that uses externally available information of each component. Using these ideas we prove composition theorems for PTIOAs.

For developing methods for proving internal stability of probabilistic hybrid systems, we work with a special case of PTIOA, which we call *Probabilistic Timed Automaton (PTA)*. A PTA does not distinguish among input, output, and internal actions, and all nondeterministic choices are resolved by a local scheduler, making it a purely probabilistic hybrid automaton.

5.3 Specification language: HIOA

The HIOA language is a formal language for specifying hybrid systems, based on the SHIOA model. An earlier version of the language was developed in [60]. The variables, actions and transitions of a hybrid I/O automata are specified in HIOA in the same way as in the IOA language [31]. Each variable has an explicitly defined static type, and an implicitly defined dynamic type. All `Real` variables are considered to be continuous, and variables of any other simple type or of the type `discrete Real` are discrete. The trajectories are defined using a set of trajectory definitions. A *trajectory definition* w is defined by an invariant $inv(w)$, a stopping condition $stop(w)$, and a state model described by a set of differential and algebraic equations $daes(w)$. $W_{\mathcal{A}}$ denotes the set of trajectory definitions of $\mathcal{A}$. Each $w \in W_{\mathcal{A}}$ defines a set of trajectories, denoted by $traj(w)$. A trajectory τ belongs to $traj(w)$ if the following conditions hold: for each $t \in \tau.dom$:

1. $\tau(t) \in inv(w)$.
2. If $\tau(t) \in stop(w)$, then $t = \tau.ltime$.
3. For each $v \in X_c$, $\tau \downarrow v$ satisfies $daes(w)$.
4. For each discrete variable v, $(\tau \downarrow v)(t) = (\tau \downarrow v)(0)$; that is, v remains constant over τ.

The set of trajectories $\mathcal{T}_{\mathcal{A}}$ of automaton $\mathcal{A}$ is the concatenation closure of the functions in $\bigcup_{w \in W_{\mathcal{A}}} traj(w)$. We show that the entities described by this specification language are indeed structued hybrid I/O automata. The HIOA language has been used to specify many hybrid systems [63, 64], including all the examples in this thesis.

A subset of the HIOA language without input and output variables, called the TIOA language, has been identified and a frontend for this language has been implemented [44]. Based on this language, a set of software tools is being developed in the TDS group; these include a language front-end, a simulator, and an interface to the Prototype Verification System (PVS) theorem prover [71].

5.4 Proof methods for invariant properties and simulation relations

An invariant property $\mathcal{I}$ of a SHIOA $\mathcal{A}$ is proved by induction on the length of closed executions of $\mathcal{A}$. The induction consists of a base case: $\mathcal{I}$ holds for all start states, and an induction step which consists of: (1) A discrete part—which tests that for every transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$, if $\mathbf{x}$ satisfies $\mathcal{I}$ then $\mathbf{x}'$ also satisfies $\mathcal{I}$, (2) A continuous part—which tests that for every closed trajectory $\tau \in \mathcal{T}$, if the first state of τ satisfies $\mathcal{I}$ then the last state of

τ also satisfies $\mathcal{I}$. Thus, this method breaks down invariant proofs into a discrete part and a continuous part. This structure, allows us to seamlessly apply results from the computer science and powerful theorems from systems theory within the same proof.

Simulation relations provide sufficient conditions to show that one automaton is an implementation of another one. A *simulation* from $\mathcal{A}$ to $\mathcal{B}$ is a relation $R \subseteq Q_{\mathcal{A}} \times Q_{\mathcal{B}}$, satisfying the following conditions: (1) For every start state of $\mathcal{A}$, there exists a related start state of $\mathcal{B}$, (2) For every pair of related states, and every action (or trajectory) α of $\mathcal{A}$, there exists a closed execution fragment β of $\mathcal{B}$, such $trace(\alpha) = trace(\beta)$, and $\alpha.lstate$ and $\beta.lstate$ are related.

We have verified several hybrid systems using composition theorems to decompose the system, and then using invariant and simulation proofs to derive the properties of interest. In the next two sections we describe two of our major case studies; the thesis covers other case studies including proving timing properties of distributed algorithms.

5.4.1 Supervisory Controller for Model Helicopter

In this section we describe the safety verification of a supervisory controller of a model helicopter system. The helicopter is driven by two rotors mounted at the two ends of its frame. The rotor inputs are either controlled by the user and the pitch dynamics is described by

$$\ddot{\theta_p} + \Omega^2 \cos\theta_p = U(t) \tag{1}$$

where Ω is the characteristic frequency of the system and U is the net input for the pitch axis ranging over U_{min} and U_{max}. User input may damage the equipment by pitching the helicopter too high or too low. The safety condition to be verified is that the pitch angle θ_p^0 is within θ_{min} and θ_{max}, which are the lower and the upper safety bounds corresponding to the helicopter hitting the ground and a fragile mechanical stop. So, the set of safe states $\mathbf{S} \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max}\}$. The `Sensor` automaton conveys the state (θ_s^0, θ_s^1) to the controllers every Δ time with arbitrary bounded errors, ϵ_0 for pitch angle and ϵ_1 for velocity. The `UsrCtrl` automaton reads the sensor outputs and triggers an output *control(u_d)* action, which communicates the user's output U_u to the supervisor. The `Actuator` automaton models the actuator and the D/A converter. There is a delay $\tau_{\texttt{act}}$ in the actuator response which is modeled as a fixed-length FIFO *buffer* of `Supervisor` outputs. The supervisory controller is designed to prevent the helicopter from reaching unsafe states. It periodically observes the position and the velocity of the helicopter and overrides the user controller by conservatively estimating the worst that might happen if the latter is allowed to remain in control. The design of the supervisor is limited by the actuator bandwidth, the sampling

rate, and sensor inaccuracies. In particular, the supervisor operated in to modes: in the `user` mode the sampled state (θ_s^0, θ_s^1) is within a region $\mathbf{U}$, and the user's output U_u is fed to the `Actuator`. If the sampled state is not in $\mathbf{U}$ then the supervisor goes into the `sup` mode, and overrides the user's output by its own recovery action, U_{min} or U_{max}, and gives back control to the user when the state is once again in $\mathbf{I}$. The variable rt is a reset timer to measure, how long the system has been in `sup` mode most recently.

The Safety of the system is verified by and proving a sequence of invariants for the composed system automaton to establish that all the reachable states are contained in the set $\mathbf{C} \subset \mathbf{S}$. We use a standard technique to prove this by induction, which is to strengthen the invariants. We define a sequence of regions $\mathbf{B}_t, 0 \leq t \leq \tau_{\texttt{act}}$, such that, $\mathbf{B}_0 = \mathbf{R}$, $\mathbf{B}_{\tau_{\texttt{act}}} = \mathbf{C}$, and $t \leq t'$ implies $\mathbf{B}_t \subseteq \mathbf{B}_{t'}$. Some of the key invariants we used to establish safety are listed below. Here, $s.v$ denotes the valuation of variable v in state s.

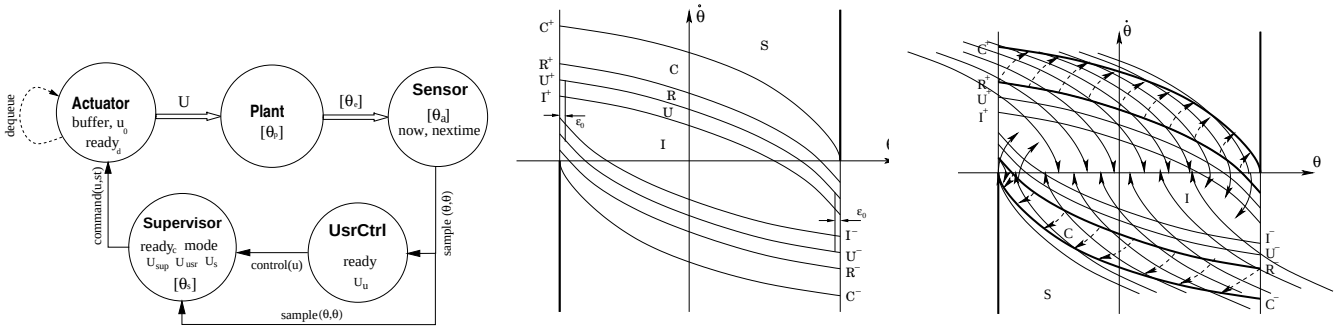

Figure 1: *left*: HIOA model of the Quanser System. *center:* Important regions in the pitch-velocity space. *right:* Trajectories of the system in the supervisor mode after $\tau_{\texttt{act}}$.

1. $s.mode = \texttt{usr} \;\Rightarrow\; s \in \mathbf{R}$.
2. $s.mode = \texttt{sup} \;\wedge\; s.rt \leq \tau_{\texttt{act}} \;\Rightarrow\; s \in \mathbf{B}_{s.rt}$.
3. $s.mode = \texttt{sup} \;\wedge\; s.rt > \tau_{\texttt{act}} \;\wedge\; s.\theta_s^1 > I^+(s.\theta_s^0) \;\Rightarrow\; U = U_{min}$.
4. $s.mode = \texttt{sup} \;\wedge\; s.rt \geq \tau_{\texttt{act}} \;\Rightarrow\; s \in \mathbf{C}$.

The proofs of these invariants illustrate one of the strengths of the HIOA framework, which is that it allows us to reason about the discrete and the continuous mechanisms in the system, independently, using the tools that are most suited for each part. For example, to prove the continuous part of the last invariant we invoke the following well known result from control theory.

Theorem 5.3 *(Theorem 3.2.21 of [11]) Consider an autonomous system $\dot{x} = g(x)$. The relatively closed set W is positively (or negatively) weakly invariant if and only if $g(x)$ (or $-g(x)$) is subtangential to W at x for all $x \in W$.*

Now, using this theorem, to show that a trajectory of the hybrid I/O automaton preserves the invariant **C**, we have to show that the tangent to the trajectory points inwards at every point on the boundary of **C**. The boundary of **C** is defined by four curves, and the tangent to every trajectory satisfying $s.mode = \texttt{sup}$ and $s.rt \geq \tau_{\texttt{act}}$, can be obtained by replacing $U(t)$ with U_{min} or U_{max} in Equation (1). So, we can check that the tangent indeed points inwards at every point on the boundary.

5.4.2 Mobile Robot Coordination using Virtual Nodes

We model a framework for distributed motion coordination for mobile robots and use formal techniques to prove the correctness of a simple coordination algorithm. Interest in such algorithms stem from the developments in the field of mobile sensor networks. The framework uses the notion of "virtual nodes" for mobile ad hoc networks. A virtual node is an abstract, relatively well-behaved active node that is implemented using less well-behaved real nodes. Virtual nodes have been be used to solve problems such as providing atomic memory [29], geographic routing [27], and point-to-point routing [28]. The goal of the motion coordination problem we study is to uniformly positioning mobile nodes on a known differentiable curve Γ.

Our physical model of the system consists of a finite but unknown number of communicating physical nodes in a bounded square $\mathcal{B}$ in R^2. We assume that each node has a unique identifier from a set $\mathcal{I}$. Formally, our physical layer model consists of three types of hybrid I/O automata: (1) automata PN_i to model physical nodes with identifiers $i \in \mathcal{I}$, (2) an automaton that models the communication service between the physical nodes, and (3) a "real world" automaton to model the physical locations of all the nodes and the real time.

The bounded square $\mathcal{B}$ is partitioned into a $m \times m$ square grid, with each grid square corresponding to a zone. Our virtual layer abstraction consists of: (1) client node automata CN_i with identifiers $i \in \mathcal{I}$, (2) one stationary virtual node automaton VN for each zone, (3) a virtual communication service for the VNs and the CNs, and (4) an automaton RW to model the physical locations of all the CNs and the real time. The Virtual Node abstraction is used as a means to coordinate the movement of client nodes in a zone. A VN controls the motion of the CNs in its zone by broadcasting target waypoints for the CNs: VN, periodically receives information from clients in its zone, exchanges information with its neighbors, and sends out a message containing a calculated target point for each client node "assigned" to its zone. VN performs two tasks when setting the target points: (1) it reassigns some of the CNs that are assigned to itself to neighboring VNs, and (2) it sends a target position on Γ to each CN that is assigned to itself.

We model each component of this system as a hybrid I/O automaton, and prove the

following properties of the algorithm: if there are no failures or recoveries of physical nodes after a certain point in time, then: (1) within finite time the set of nodes in each zone becomes fixed, and the size of the set is "approximately" proportional to the length of Γ in the zone, (2) within finite time all physical nodes in the zones that contain some part of Γ are located on Γ, and (3) in the limit, all the nodes in each zone are evenly spaced on Γ.

5.5 Proof methods for stability

In control theory literature, many different notions of stability are used—asymptotic, uniform, bounded-input-output, global [48, 51]—which we show, can be adopted to the HIOA framework. However, all these definitions of stability are concerned only with the continuous state of hybrid systems. On the other hand, the notion of stability has been widely used in designing self-stabilizing distributed algorithms [26, 69]. In the thesis we propose a general definition of stability of hybrid systems, that encompasses both its discrete and its continuous variables.

In general, stability is a stronger notion than invariance, because a system is stable only if it converges to an equilibrium state starting from *any* state; whereas, an invariant property is *preserved* when the system starts from a state that already satisfies the invariant. So, the inductive proof techniques that are available for proving invariance cannot be directly applied to prove stability. For developing proof methods, we are primarily concerned with internal stability of (probabilistic) hybrid systems under the usual meaning of stability, that concerns only the continuous state of the system. Therefore we work with the SHA and PTA models, and with the definition of stability given in Section 5.1.

Lyapunov methods. The basic tool for studying stability of hybrid systems relies on the existence of a *Common Lyapunov function* whose derivative along the trajectories of all the modes satisfies the suitable inequalities. When such a function is not known or does not exist, *Multiple Lyapunov* functions [14] is an useful tool for proving stability of a chosen execution. We show that both these techniques can be applied in the SHA model. We illustrate the application of these techniques with examples.

Stability under slow switching. We develop techniques for proving stability of hybrid systems that switch "slowly". The *Average Dwell Time* [41] criteria define restricted classes of switching signals, based on switching speeds, and one can conclude the stability of a system with respect to these restricted classes. In the SHA model, the discrete transitions of the automaton define the speed of switching, and so the average dwell time property can be formalized as a property of the automaton.

Definition 5.4 *Let $N(\alpha)$ denote the number of switches over an execution fragment α of a*

SHA $\mathcal{A}$. The execution fragment has an average dwell time $\tau_a > 0$ if there exists a positive number N_0 such that:

$$N(\alpha) \leq N_0 + \frac{\alpha.ltime}{\tau_a}. \tag{2}$$

If all reachable execution fragments of $\mathcal{A}$ have average dwell times of at least τ_a then $\mathcal{A}$ has an average dwell time τ_a.

A large ADT means that the system spends enough time in each mode, so as to dissipate the transient energy gained through mode switches. Having a large average dwell time itself is not sufficient for stability, in addition, the individual modes of the automaton must also be stable. In fact, the problem of testing the stability of a hybrid system can be broken down into, (a) finding Lyapunov functions for the individual modes, and (b) checking the average dwell time property. We assume that a solution to part (a)— a set of Lyapunov functions for the individual modes — is known from existing techniques, and we present two complementary methods for part (b), that is, verifying the ADT property.

5.5.1 Invariant Approach

A given automaton $\mathcal{A}$ has average dwell time τ_a, if all its executions have average dwell time of at least τ_a. In general, properties of executions are harder to check than invariant properties which are properties of the state. Our first method transforms the given automaton by adding some extra state to it, so that the original automaton has ADT τ_a if and only if the corresponding transformed automaton $\mathcal{A}'$ satisfies a particular invariant property $\mathcal{I}$. This transformation allows us to prove ADT for automaton $\mathcal{A}$ by checking the invariant $\mathcal{I}$ for automaton $\mathcal{A}'$ which can be done using the techniques described in Sections 5.4.

The application of this method is illustrated by analyzing the hysteresis switching logic unit of a supervisory control system taken from [40]. For certain restricted types of hybrid I/O automata, invariants can be checked automatically by a model checking tool like HyTech [35]. So, for such systems, the the ADT property an also be checked automatically.

5.5.2 Optimization based approach.

This method is complementary to the previous one; it searches for an execution of the given automaton $\mathcal{A}$, that violates the candidate ADT property τ_a, failing which, it declares that the ADT property is satisfied. From Definition 5.4 it follows that a given $\tau_a > 0$ is *not* an average dwell time of a given SHA $\mathcal{A}$ if for every N_0 there exists an execution α of $\mathcal{A}$ such that

$$N(\alpha) > N_0 + \frac{\alpha.ltime}{\tau_a}, \quad or \quad S_{\tau_a}(\alpha) \triangleq N(\alpha) - \frac{\alpha.ltime}{\tau_a} > N_0.$$

This formula can be verified by optimizing the following function:

$$\mathsf{OPT1}: \quad \alpha^* \in \arg\max S_{\tau_a}(\alpha)$$

over all the executions of $\mathcal{A}$. If the optimal value $S_{\tau_a}(\alpha^*)$ is bounded then $\mathcal{A}$ has average dwell time τ_a with the optimal value as the corresponding constant N_0. Otherwise, if the optimal value is unbounded then τ_a is not an ADT for $\mathcal{A}$.

In general, the optimization problem OPT1 may not be solvable using standard mathematical programming tools because, among other things, the optimal executions of $\mathcal{A}$ may not have finite descriptions. However, for certain useful classes of SHA, OPT1 is solvable. For example, we show that for rectangular, finitely initialized SHA, it is necessary and sufficient to optimize the objective over the simple cyclic execution fragments of the automaton. This reduces the search space dramatically, and makes it possible to formulate and solve this, more constrained, optimization problem as a Mixed Integer Linear Program. Similarly, we show that for a useful subclass of linear hybrid systems OPT1 reduces to detecting a negative-cost cycle in a weighted graph, which can be done efficiently using Bellman-Ford algorithm [22]. Also, we note that, if there exists an efficient algorithm for solving OPT1, then we can use this algorithm with binary search, and *find* the ADT of the given automaton. We illustrate the application of this method by verifying and finding the ADT of several hybrid systems, including a hysteresis switch with monitoring signals that are generated by linear differential equations.

Stability of PTA through ADT. We will develop techniques for proving stability of probabilistic hybrid systems modeled as PTA (see Section 5.2.2), using probabilistic variants of ADT. Stability conditions for probabilistic switched systems using Lyapunov function like arguments have been derived in [17, 18]. As in the deterministic case, we will develop an approach for proving ADT properties through invariant distributions.

We will show that for PTAs, the problem of proving *almost sure stability*, using Lyapunov arguments can also be formulated as an optimization problem that can be solved efficiently. For special classes of PTA that arise naturally, we will show that the optimization based approach leads to automatic verification techniques. For example, we first consider a model in which the mode switch transitions occur periodically, every Δ time, and leads to a probability distribution over the modes of the automaton. This corresponds to a hybrid system in which the discrete transitions are triggered by a discrete time Markov chain. In this simple setting we show that the ADT property reduces to detecting the minimum cost cycle in a graph, and therefore can be solved using the Bellman-Ford algorithm. Next, we consider a slightly more general model in which mode transitions occur at any rate, not just at a fixed rate Δ, depending on both the continuous and the discrete states of the

automaton. We will show how the ADT verification problem for this class of PTA can be formulated and solved as an optimization problem.

5.6 Software Tools

Theorem provers are used to construct proofs interactively, thus giving the highest level of assurance. Also, with a theorem prover, saved proof scripts can be executed quickly to re-check the validity of a proof after changes have been made to a specification. Various tools have been developed to translate IOA specifications to different theorem provers, for example, Larch [12, 32], PVS [25], and Isabelle [66, 73]. In this work, we have chosen to use the PVS theorem [71, 78] prover, because it provides an expressive specification language and an interactive prover with powerful decision procedures. PVS also provides a way of developing special strategies or tactics for partially automating proofs, and it has been used in many real life verification projects [72].

To verify a property in PVS the user invokes the theorem prover and supplies proof commands to manipulate the proof obligations or subgoals, until all the subgoals are resolved. This procedure may involve a lot of interaction, at a very low level of detail between the user and the theorem prover. The *Timed Automaton Modeling Environment (TAME)* [7] demonstrates that for stylized proofs, like proof of invariants in the MMT-automaton model [59], it is possible to minimize this interaction and to allow the user to interact with the prover at a more abstract level that embodies the natural high-level steps typically needed in hand proofs.

SHIOA to PVS Translation. We have designed a scheme for translating SHIOA specifications to the language of the PVS theorem prover. A translator based on this scheme will enable users to verify SHIOA specifications using the theorem prover, without having to re-write them in PVS. The translation scheme is based on a PVS theory template, like the one provided by Timed Automata Modeling Environment (TAME) [7]. This *template* is instantiated with the state variables, actions, transitions, and trajectories, of a particular hybrid I/O automata, to obtain the description of the automaton in PVS. In the TAME template theory, passage of time is written as a *time pasage* action, with the task time bounds of the automaton encoded in the enabling condition of this action. This approach, however, if applied directly to translate trajectories of SHIOA, does not allow assertion of properties that must hold throughout the duration of the trajectory. Herein lies the main challenge in designing the translation scheme: to translate the trajectories without imposing too many restrictions on the nature of the trajectories. Our translation scheme solves this problem by embedding the trajectory as a functional parameter of the time passage action.

This translation scheme has been used to implement a TIOA to PVS translator [49].

Framework for developing PVS Strategies. TAME provides several PVS strategies for proving invariant properties of timed I/O automata. Abstraction properties involve a pair of automata, and hence to express them generally, one needs a way to represent abstract automaton objects in PVS. The most convenient way to represent abstract automaton objects would be to make them instances of a type "automaton". This proved to be challenging because no general automaton type can be defined in PVS, and unlike Isabelle/HOL [70], PVS does not support parametric polymorphism. With the theory instantiation feature of PVS, we have designed a framework that supports defining abstraction relations between two automata that is both straightforward for a user and clean from the point of view of the strategy developer. This support relies on a new `automaton` theory and a library of *property theories*.

The theory `automaton`, is an abstract declaration of the elements that specify an automaton in PVS. The property theories define abstraction properties, for example, forward simulation, refinement, etc., in terms of two parameters A and B that are theories of type `automaton`. A particular PVS automaton theory when interpreted as an instance of the `automaton` theory, provides concrete definitions of `automaton`'s elements. Abstraction properties are stated in a separate PVS theory as follows: Two PVS automaton theories specifying the abstract and the concrete machines are imported. These theories are interpreted as instances of the `automaton` theory. The abstraction relation between the variables of the automata that is to be proved is defined. Then, the relevant property theory is imported to state the abstraction property as a theorem to be proved.

Strategies for abstraction proofs. We have developed PVS strategies for proving forward simulation and refinement, between a pair of timed I/O automaton. Strategies are macros made up of many PVS proof commands embedded in Lisp code. Our strategies exploit the known structure of invariant and abstraction proof of hybrid I/O automata, to partially automate proofs of abstractions.

We have developed two generic strategies for aiding forward simulation proofs: **PROVE_FWD_SIM** and **FWD_SIM_ACTION_REC**. These strategies rely on several substrategies together with some of the strategies that were developed earlier. **PROVE_FWD_SIM** is **PROVE_REFINEMENT** first breaks down a forward simulation theorem into its base case and induction step, and then splits the induction step into cases for the individual action subtypes of the concrete automaton. The **FWD_SIM_ACTION_REC** strategy is meant to be applied to the individual action branches produced by **PROVE_FWD_SIM**; it is used to prove the induction step where an input or output action of the concrete automaton is simulated by

an sequence of actions of the abstract automaton. This strategy takes as input an action sequence $\sigma = a_1, a_2, \ldots, a_n$, a starting state s_1, and a known target state s_2, and produces the following set of subgoals:

- $s_2 = \texttt{trans}(a_n, \texttt{trans}(a_{n-1}, \texttt{trans}(a_{n-2}, \ldots, \texttt{trans}(a_1, s_1)\ldots)))$,
- For each action a_i in σ, a_i is `internal`, and
- For each $i \in 1, \ldots, n$, a_i is `enabled` in $\texttt{trans}(a_{i-1}, \texttt{trans}(a_{i-2}, \ldots, s)\ldots)$.

The strategy then discharges some of these subgoals by expanding definitions and simplifying. The remaining non-trivial subgoals are then presented to the user with properly labeled sequents. We have used these strategies in verifying several case studies, including the a two task race system, a tree based leader election algorithm, a memory module [20, 61, 62].

Solving MILPs for verifying ADT properties. In Section 5.5.2 we have discussed how ADT properties of SHA and PTA can be viewed as optimization problems. To automatically verify ADT properties using standard tools for solving Mixed integer linear programs, one has to carefully formulate and express the optimization problem in a way that can be effectively solved by a tool. For certain classes of SHA and PTA, we present a methodology for expressing the ADT verification problem in the Gnu LP Kit (GLPK) [1], and solve the resulting MILP using the GLPK solver.

References

[1] GLPK - GNU linear programming kit. http://www.gnu.org/directory/libs/glpk.html.

[2] R. Alur. *Techniques for automatic verification of real-time systems.* PhD thesis, Stanfor University, 1991.

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[4] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] R. Alur, T. A. Henzinger, G. Lafferriere, and George Pappas. Discrete abstractions of hybrid systems. In *Proceedings of the IEEE*, 2000.

[6] P. J. Antsakis, J. A. Stiver, and M. D. Lemmon. Hybrid system modeling and autonomous control systems. In R. L. Grossman et al., editors, *Hybrid Systems*, volume 736 of *LNCS*, New York, 1993.

[7] M. Archer. TAME: PVS Strategies for special purpose theorem proving. *Annals of Mathematics and Artificial Intelligence*, 29(1/4), February 2001.

[8] A. Back, J. Guckenheimer, and M. Myers. A dynamical simulation facility for hybrid systems. In R. L. Grossman et al., editors, *Hybrid Systems*, volume 736 of *LNCS*, New York, 1993.

[9] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, G. M. Miconi, U. Pozzi, T. Villa, H. Wong-Toi, and A. L. Sangiovanni-Vincentelli. Maximal safe set computation for idle speed control of an automotive engine. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control (HSCC 2000)*, volume 1790 of *Lecture Notes in Computer Science*, pages 32–44, New York. Springer-Verlag.

[10] A. M. Bayen, E. Cruck, and C. Tomlin. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: solving the hamilton-jacobi equation using viability techniques. In *Hybrid Systems: Computation and Control 2002*, volume 2289 of *LNCS*, pages 90–104. Springer, March 2002.

[11] N. P. Bhatia and G. P. Szegö. *Dynamical Systems: Stability Theory and Applications*, volume 35 of *Lecture notes in mathematics*. Springer-Verlag, Berlin; New York, 1967.

[12] A. Bogdanov, S. Garland, and N. Lynch. Mechanical translation of I/O automaton specifications into first-order logic. In *Formal Techniques for Networked and Distributed Sytems - FORTE 2002 : 22nd IFIP WG 6.1 International Conference*, pages 364–368, Texas, Houston, USA, November 2002.

[13] M. Branicky. *Studies in hybrid systems: modeling, analysis, and control*. PhD thesis, MIT, Cambridge, MA, June 1995.

[14] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43:475–482, 1998.

[15] M. Branicky, V. Borkar, and S. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[16] 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 20–53, Boston, 1994. Birkhauser.

[17] D. Chatterjee and D. Liberzon. On stability of stochastic switched systems. In *in Proceedings of the 43rd Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

[18] D. Chatterjee and D. Liberzon. Stability analysis of deterministic and stochastic switched systems via a comparison principle and multiple lyapunov functions. *SIAM Journal on Control and Optimization*, 2005.

[19] L. Cheung, N. A. Lynch, R. Segala, and F. Vaandrager. Switched probabilistic i/o automata. In *First International Colloquium on Theoretical Aspects of Computing*, July 2004.

[20] G. Chockler, N. A. Lynch, S. Mitra, and J. Tauber. Proving atomicity: an assertional approach. In *Proceedings of International Symposium on Distributed Computing (DISC'05)*, Cracow, Poland, September 2005.

[21] E. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[22] T. H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[23] H. de Jong, J.-L. Gouz, C. Hernandez, M. Page, and J. Geiselmann T. Sari. Hybrid modeling and simulation of genetic regulatory networks: A qualitative approach. In *Hybrid Systems: Computation and Control, HSCC 2003*, volume 2623 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2003.

[24] R. DePrisco, Butler Lampson, and Nancy Lynch. Revisitng the paxos algorithm. In *Distributed Algorithms 11th Workshop WDAG'97*, pages 111–125, Berlin-Heidelberg, 1997.

[25] M. Devillers. Translating IOA automata to PVS. Technical Report CSI-R9903, Computing Science Institute, University of Nijmegen, February 1999. Available at `http://www.cs.ru.nl/research/reports/info/CSI-R9903.html`.

[26] S. Dolev. *Self-stabilization*. MIT Press, Cambridge, MA, USA, 2000.

[27] S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. A. Nolte. Virtual stationary automata for mobile networks. *Technical Report MIT-LCS-TR-979*, 2005.

[28] S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvartsman, and Jennifer L. Welch. Virtual mobile nodes for mobile ad hoc networks. In *18th International Symposium on Distributed Computing (DISC)*, pages 230–244, 2004.

[29] S. Dolev, S. Gilbert, N. A. Lynch, Alex Shvartsman, and Jennifer Welch. Geoquorums: Implementing atomic memory in mobile ad hoc networks. In *GeoQuorums: Implementing Atomic Memory In Mobile Ad Hoc Networks*, 2003.

[30] E. Dolginova and N. A. Lynch. Safety verification for automated platoon maneuvers: A case study. In *HART'97 (International Workshop on Hybrid and Real-Time Systems)*, volume 1201 of *Lecture Notes in Computer Science series*, Grenoble, France, March 1997. Springer Verlag.

[31] S. Garland, N. A. Lynch, J. Tauber, and M. Vaziri. *IOA User Guide and Reference Manual*. MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2003. Available at `http://theory.lcs.mit.edu/tds/ioa.html`.

[32] S. J. Garland and J. V. Guttag. A guide to LP, the Larch prover. Technical report, DEC Systems Research Center, 1991. Available at `http://nms.lcs.mit.edu/Larch/LP`.

[33] C. Heitmeyer and N. A. Lynch. The generalized railroad crossing: A case study in formal verification of real-time system. In *Proceedings of the 15th IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico, December 1994. IEEE Computer Society Press.

[34] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, pages 278–292, New Brunswick, New Jersey, 1996.

[35] T. A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. In *Computer Aided Verification (CAV '97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–483, 1997.

[36] T. A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.

[37] T. A. Henzinger and Rupak Majumdar. Symbolic model checking for rectangular hybrid systems. In *Proceedings of the Sixth International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, pages 142–156. Lecture Notes in Computer Science 1785, Springer-Verlag, 2000.

[38] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[39] J. P. Hespanha. Stochastic hybrid systems: Application to communication networks. In G. J. Pappas R. Alur, editor, *Hybrid Systems: Computation and Control: 7th International Workshop, HSCC 2004*, volume 2993, pages 387 – 401, Philadelphia, PA, USA, 2004. Lecture Notes in Computer Science.

[40] J.P. Hespanha, D. Liberzon, and A.S. Morse. Hysteresis-based switching algorithms for supervisory control of uncertain systems. *Automatica*, 39:263–272, 2003.

[41] J.P. Hespanha and A. Morse. Stability of switched systems with average dwell-time. In *Proceedings of 38th IEEE Conference on Decision and Control*, pages 2655–2660, 1999.

[42] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In N. A. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control, 3rd Int. Workshop (HSCC 2000)*, Lecture Notes in Computer Science, pages 160–173, 2000, 2000.

[43] D. Kaynar, N. A. Lynch, and S. Mitra. Specifying and proving timing properties with TIOA tools. In *In the work in progress session of the 25th IEEE International Real-Time Systems Symposium (RTSS-WIP)*, Lisbon, Portugal, December 2004.

[44] D. Kaynar, N. A. Lynch, S. Mitra, and S. Garland. *TIOA Language*. MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2005.

[45] D. Kaynar, N. A. Lynch, R. Segala, and F. Vaandrager. The theory of timed I/O automata. Technical Report MIT/LCS/TR-917a, MIT Laboratory for Computer Science, 2004. Available at `http://theory.lcs.mit.edu/tds/reflist.html`.

[46] D. K. Kaynar, N. A. Lynch, R. Segala, and F. Vaandrager. Timed I/O automata: A mathematical framework for modeling and analyzing real-time system. In *RTSS 2003: The 24th IEEE International Real-Time Systems Symposium*, Cancun,Mexico, December 2003.

[47] G. Leeb and N. A. Lynch. Proving safety properties of the steam boiler controller. In E Boerger J. Raymond Abrial and H. Langmaack, editors, *Proceedings of Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume 11654 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[48] D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.

[49] H. Lim, D. Kaynar, N. A. Lynch, and S. Mitra. Translating timed i/o automata specifications for theorem proving in pvs. In *Proceedings of Formal Modelling and*

Analysis of Timed Systems (FORMATS'05), LNCS, Uppsala, Sweden, September 2005. Springer.

[50] C. Livadas, J. Lygeros, and N. A. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99),Phoenix, Arizona*, pages 115–125, December 1999.

[51] D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. John Wiley and Sons, Inc., New York, 1979.

[52] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.

[53] N. A. Lynch. A three-level analysis of a simple acceleration maneuver, with uncertainties. In *Proceedings of the Third AMAST Workshop on Real-Time Systems*, pages 1–22, Salt Lake City, Utah, March 1996. World Scientific Publishing Company.

[54] N. A. Lynch and Hagit Attiya. Using mappings to prove timing properties. *Distributed Computing*, 6(2):121–139, 1992.

[55] N. A. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, August 2003.

[56] N. A. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. Technical Memo MIT/LCS/TR-827d, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, January 13 2003.

[57] N. A. Lynch and F. Vaandrager. Forward and backward simulations - part II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996.

[58] Z. Manna and A. Pnueli. *The Correctness problem in Computer Science*, chapter The temporal framework for concurrent programs. Academic Press, 1981.

[59] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In J. C. M. Baeten andJ. F. Goote, editor, *CONCUR '91: 2nd International Conference of Concurrency Theory*, volume 527, pages 408–423, 1991.

[60] S. Mitra. HIOA - a specification language for hybrid Input/Output automata. Master's thesis, Department of Computer Science and Automation, IISc, Indian Institute of Science, Bangalore, 2001. http://theory.lcs.mit.edu/mitras/research/mastersthesis.ps.gz.

[61] S. Mitra and M. Archer. Reusable pvs proof strategies for proving abstraction properties of I/O automata. In *STRATEGIES 2004, IJCAR Workshop on strategies in automated deduction*, Cork, Ireland, July 2004.

[62] S. Mitra and M. Archer. Pvs strategies for proving abstraction properties of automata. *Electronic Notes in Theoretical Computer Science*, 125(2):45–65, 2005.

[63] S. Mitra and D. Liberzon. Stability of hybrid automata with average dwell time: an invariant approach. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, February 2005.

[64] S. Mitra, Y. Wang, N. A. Lynch, and E. Feron. Safety verification of model helicopter controller using hybrid Input/Output automata. In *HSCC'03, Hybrid System: Computation and Control*, Prague, the Czech Republic, April 3-5 2003.

[65] A. S. Morse. Supervisory control of families of linear set-point controllers, part 1: exact matching. *IEEE Transactions on Automatic Control*, 41:1413–1431, 1996.

[66] T. Ne Win. Theorem-proving distributed algorithms with dynamic analysis. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2003.

[67] N. A. Neogi. Dynamic partitioning of large discrete event biological systems for hybrid simulation and analysis. In *HSCC*, pages 463–476, 2004.

[68] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, stability. In R. L. Grossman et al., editors, *Hybrid Systems*, volume 736 of *LNCS*, New York, 1993.

[69] M. Nesterenko and A. Arora. Stabilization-preserving atomicity refinement. *J. Parallel Distrib. Comput.*, 62(5):766–791, 2002.

[70] O. Müller. *A Verification Environment for I/O Automata Based on Formalized Meta-Theory*. PhD thesis, Technische Universität München, Sept. 1998.

[71] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Computer-Aided Verification, CAV '96*, number 1102 in Lecture Notes in Computer Science, pages 411–414, New Brunswick, NJ, July/August 1996. Springer-Verlag.

[72] S. Owre, J. Rushby, N. Shankar, and D. Stringer-Calvert. PVS: an experience report. In Dieter Hutter, Werner Stephan, Paolo Traverso, and M. Ullman, editors, *Applied Formal Methods—FM-Trends 98*, volume 1641 of *Lecture Notes in Computer Science*, pages 338–345, Boppard, Germany, oct 1998. Springer-Verlag.

[73] L. C. Paulson. The Isabelle reference manual. Technical Report 283, University of Cambridge, 1993.

[74] A. Pnueli. The temporal logic of programs. In *IEEE Symposium on Foundations of Computer Science*, pages 46–77, 1977.

[75] R. Segala. A compositional trace-based semantics for probabilistic automata. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR '95)*, volume 962 of *LNCS*, pages 234–248, Philadelphia, PA, USA, August 1995.

[76] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, June 1995.

[77] R. Segala. Testing probabilistic automata. In *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR '96)*, volume 1119 of *LNCS*, pages 299–314, Pisa, Italy, August 1996.

[78] N. Shankar, S. Owre, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS Prover Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.

[79] Ashish Tiwari and Gaurav Khanna. Series of abstractions for hybrid automata. In Claire Tomlin and Mark R. Greenstreet, editors, *5th International Workshop on Hybrid Systems: Computation and Control,*, volume 2289 of *Lecture Notes in Computer Science*, pages 465–478. Springer, 2002.

[80] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, London, 2000.

[81] H. B. Weinberg and N. A. Lynch. Correctness of vehicle control systems – a case study. In *17th IEEE Real-Time Systems Symposium*, pages 62–72, Washington, D. C., December 1996.

[82] H. B. Weinberg, N. A. Lynch, and Norman Delisle. Verification of automated vehicle protection systems. In T. A. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems)*, volume 1066 of *Lecture Notes in Computer Science*, pages 101–113, New Brunswick, New Jersey, October 1995. Springer-Verlag.