Communication and control of distributed hybrid systems

T. Şimşek, P. Varaiya and J. Borges de Sousa [1]
email: {simsek,varaiya,sousa}@eecs.berkeley.edu
Tel: (510) 642-5649
Fax: (510) 642-6330
Dept. of Electrical Engineering and Computer Science
University of California, Berkeley, CA, 94720

Abstract

The rich and exciting research over the past decade concerning the description, analysis, controller design, simulation, and implementation of distributed systems is reviewed. From control engineering, this research has inherited the concepts and theories of optimality, stability, controlled differential equation models, and the motivation to improve the performance of increasingly complex physical processes. From computer science, the research has incorporated the theories of logical specification and verification, event-driven state machine models, concurrent processes and object-oriented approaches. The review is organized in the framework of dynamic networks of hybrid automata (DNHA). The case study of an automated highway system (AHS) is used to illustrate the challenges posed by a complex distributed system, and the research contributions that address different challenges. There is an equal emphasis on the conceptual and theoretical contributions and on tools and techniques that yield more immediately practical benefits.

Keywords Distributed hybrid systems, communication, control, networked multi-vehicle systems, AHS, Hybrid Systems, SHIFT.

1 Introduction

The last decade has witnessed unprecedented interactions between technological developments in computing, communications and control, and the design and implementation of interacting dynamical systems, such as networked multi-vehicle systems. These developments enable engineers to design new systems, and in turn, the implementation of those systems leads to a better understanding of the underlying technological issues, and to the formulation of new theories. One such example, and the one that we are concerned with in this paper, is the theory of distributed hybrid systems. That theory led to the development of a body of technology and tools for simulation, analysis and de-

sign. The objective of this paper is to present a survey of some of this material.

The control of distributed hybrid systems has presented a new challenge to control theory. The challenge comes from the distributed nature of the problem. For example, in networked multi-vehicle systems, information and commands are exchanged among multiple vehicles, and the roles, relative positions, and dependencies of those vehicles change during operations. In fact, this challenge entails a shift in the focus of control theory – from prescribing and commanding the behavior of isolated systems to prescribing and commanding the behavior of interacting systems.

The control and computer science communities address this challenge in the context of distributed hybrid systems, and contribute complementary views and techniques. The inter-disciplinary nature of hybrid systems requires a new description language. This language is in the process of being developed. Meanwhile, control engineers have developed a collection of idioms, patterns and styles of organization that serves as a shared, semantically rich, vocabulary among them [68]. However, this shared vocabulary is still deeply rooted in the underlying mathematical framework – differential equations and dynamic optimization – and lacks some semantically rich concepts invoked by distributed computing. The cause may be that experience and functionality in computing are acquired at a rate unmatched by the rate of evolution of concepts in control systems. For example, it was only recently that the expressiveness of the language of differential equations and dynamic optimization was enlarged with concepts from mathematical logic, under the denomination of hybrid control (see for example [66], [57], [16], [34]). In this paper we will attempt to provide an overview of the complementary views and perspectives from computer science and control systems.

The paper is organized as follows. In Section 2, we present the Automated Highway System (AHS) as the motivation for our discussion on distributed hybrid systems. In Section 3, we discuss and survey distributed

hybrid systems research. In Section 4, we use the Smart Automated Highway System (Smart-AHS) case study to illustrate some the concepts discussed before. In Section 5 we draw some conclusions and discuss open questions.

2 Case Study

Recent advances in hybrid systems are partly due to the diversity of their application domains ranging from control systems to bioinformatics [8]. In this section we focus on the control of distributed dynamic systems that have played a central role. Novel examples of distributed control systems include the control and coordination of Unmanned Aerial Vehicles (UAVs), autonomous underwater vehicles [13, 30, 27], Mobile Offshore Bases (MOB) [29, 28] and Intelligent Vehicle/Highway Systems (IVHS). To fix ideas we will discuss selected topics from the Partners for Advanced Transit and Highways (PATH) Automated Highway System (AHS) – a fully automated IVHS policy.

The AHS is a distributed and hierarchical control system that aims to increase highway capacity, safety and efficiency without building more roads. Due to its size and impact on everyday life, the design of a safe, efficient and practical AHS has been the source of challenges leading to the development of control, communication, traffic flow theories and enabling sensor and actuator technologies. The hybrid systems field is no exception. After a brief description of the AHS we provide examples that promote the use of hybrid systems for AHS.

2.1 AHS concepts
Traffic is organized into groups of tightly spaced vehicles called *platoons*. In this case the capacity of the highway is given by [79]:

$$Capacity = \frac{vn}{ns + \Delta x(n-1) + d} \text{ vehicles/lane/hour}$$

where the quantities v, n, s, Δx and d represent the average velocity, platoon size, vehicle length, inter-vehicle spacing and inter-platoon spacing respectively. A simple calculation shows that for an average platoon size of 20 with inter-vehicle spacing of 1m the capacity of the highway increases by a factor of 4 with respect to free-flow traffic:

v (kph)	n	s (m)	Δx (m)	d (m)	$Capacity$ (vehicles/lane/hour)
72	1	5	free-flow	30	2,100
72	5	5	2	60	3,840
72	20	5	1	60	8,000

Furthermore, this approximation is conservative in that a large inter-platoon spacing of 60m is assumed. The tight inter-vehicle spacing also accounts for increased safety and efficiency. The former is due to a small relative velocity in the event of a collision (it is assumed that the 60m inter-platoon spacing is sufficiently large to avoid collisions between vehicles in separate platoons) and the latter due a reduction of average aerodynamic drag experienced by a vehicle.

The benefits of the AHS are accounted by the tight formation of vehicles in a platoon traveling at relatively high velocities. It is clear that control policies beyond those of human drivers are needed to realize such platoon configurations. The AHS envisions full automation.

A typical scenario under full automaton is illustrated as follows. Say you are the driver and are in a manual lane. You drive your car into the transition area where it is validated, registered and queued by the AHS entry/exit system. You place your car under automatic control and the car merges into the automated lane. Until your car reaches its destination it performs a dynamic sequence of maneuvers such as joining/leaving platoons, changing lanes and yielding to other vehicles that may merge on to the AHS. At your destination exit the control of your vehicle is returned to you.

In August 1997, the National Automated Highway Systems Consortium (NAHSC) demonstrated AHS technologies, including an eight-vehicle platoon system, on I-15 in San Diego, CA.

2.2 Selected AHS problems
The AHS introduced several challenges that led to the development of tools, techniques and theories for hybrid systems. In this section we provide selected examples and explore the motivation for these advances.

The hierarchical control architecture [79, 81] illustrated in Figure 1 provides a setting for our examples:

1. *Physical layer* – Open-loop vehicle dynamics.
2. *Regulation* – Control of join, split, lane-change maneuvers.
3. *Coordination* – Compute maneuver sequences to fulfill a goal.
4. *Link* – Control vehicle activity for a section of the highway.
5. *Network* – Control of traffic flow amongst links.

The salient feature of this architecture is that it breaks down AHS control into a self-contained hierarchically organized functional layers.[1] That is, it provides us with a semantic framework in which we extract subproblems from AHS and reason about them in self-

---

[1]We will restrict our attention to the regulation, coordination and link layers. Neither the physical nor network layer have been addressed in the context of hybrid system formalisms.

contained theories.

For example, consider again the AHS scenario described above. Assume there is a prescribed set of ideal conditions. Under these conditions the AHS is said to be functioning in the normal mode of operation. The assignment of a vehicle to the enter, exit, or yield operations is a control action taken by the link layer. This action is broken down into a sequence of join, split and lane-change maneuvers by the coordination layer. This sequence of maneuvers is then carried out by closed-loop feedback laws at the regulation layer. For the normal mode of operation the functions of these three layers are described using self-contained theories. The functions of the regulation, coordination and link layers are described using the theories of continuous-time feedback control, Finite State Machines (FSMs) and fluid flows respectively.

A comprehensive control system design must also consider non-ideal conditions. These include conditions resulting from obstacles on the highway, noisy communications links, a flat tire, sensor and actuator failures. The difficulty with non-ideal conditions is that they do not admit a compact representation in the ideal theories.
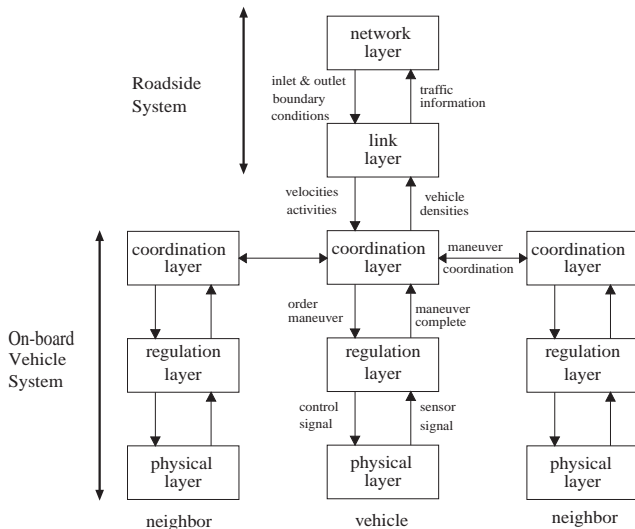

Figure 1: AHS control hierarchy. Figure from [81].

The following observations are immediate.

1. The ideal theories assigned to the regulation and coordination layers naturally form a hybrid system – a dynamic system that evolves in alternating continuous and discrete phases. In fact, we will demonstrate below that the regulation layer functions alone form a special class of hybrid systems called piece-wise continuous dynamic systems.

2. The ideal theories must extend to accommodate mutations and dynamic reconfigurations. For example, to model the join maneuver. Suppose vehicle v is to join platoon p. The control laws of v must be reconfigured dynamically to operate in closed-loop with the vehicles in p and vice-versa.

3. The tools used to describe the system functions must admit a formal methodology, for example, to derive degraded modes of operation from ideal modes. The methodology used should provide tools and techniques to preserve properties of the ideal controllers and reason about degraded modes of operation with respect to these properties.

Item 1 promotes the use of hybrid systems to describe AHS functions for obvious reasons. Items 2 and 3 promote hybrid systems due to their rich syntax and semantics that incorporates an expressive language for differential equations and dynamic optimization endowed in a formal and structured logical framework. For example, in Section 4 we describe the object-oriented programming language Shift. Shift provides structured programming facilities to describe dynamic networks of hybrid automata. The language provides a formal syntax and semantics that incorporates first order predicate logic that is used to dynamically reconfigure the state of a hybrid automaton. On the other hand, its abstraction constructs – such as inheritance – are used to incrementally describe ideal (*eg.* normal) and extended (*eg.* degraded) behaviors for hybrid automata.

2.2.1 Safe controller design example: Consider a merge maneuver with the following specifications. The inter-vehicle displacement Δx, lead vehicle velocity v_l, follower vehicle velocity v_f, relative velocity $\Delta v = \dot{\Delta x}$ and the maximum recommended velocity for a vehicle to travel on the highway v_{max}. The objective is to design an efficient and comfortable control law that decreases the initial relative displacement $\Delta x(0)$ to a desired inter-vehicle spacing Δx_{join} subject to the safety constraint

$$\Delta x(t) \leq 0 \Rightarrow \Delta v(t) \geq -\bar{v} \text{ and } v_l > 0 \qquad (1)$$

for all times t. The quantity $\bar{v}$ is the maximum tolerated impact velocity in case of a collision. The following theorem [54] states the existence of safe control laws and provides a class of maximum braking safe control laws.

Theorem 1 *Suppose that the acceleration of each vehicle is bounded by $[-\underline{a}, \bar{a}]$ and that the maximum deceleration $-\underline{a}$ is achieved and maintained at most d seconds after a maximum braking command is issued. Let X_{MS}, X_{safe} and X_{bound} denote the set of all triples $X = (\Delta x, \Delta v, v_l)$ with $v_l \geq 0$ that satisfy (1), (2) and*

(3) respectively:

$$\Delta v \geq Ad - \max \left\{ \begin{array}{l} \sqrt{2\underline{a}\Delta x + v_l{}^2 + \bar{v}^2 + \underline{a}Ad^2} - v_l, \\ \bar{v} \end{array} \right.$$

$$\tag{2}$$

$$\Delta v \geq -\max \left\{ \begin{array}{l} \sqrt{2\underline{a}\Delta x + v_l{}^2 + \bar{v}^2} - v_l, \\ \bar{v} \end{array} \right. \tag{3}$$

where $A = \underline{a} + \bar{a}$. Then, provided that $X(t) \in X_{safe}$:

1. There exists a control law that is safe for all times $s > t$. ie., $\forall s > t, X(s) \in X_{MS}$.
2. Any control law that applies maximum braking whenever $X(t) \notin X_{safe}$ is safe for all times $s > t$. Furthermore, $\forall s > t, X(s) \in X_{bound}$.
3. $X_{safe} \subset X_{bound} \subset X_{MS}$.

Observe that the maximum braking safe control law naturally defines a class of two-state hybrid automaton as illustrated in Figure 2. In the *Max Braking* state Theorem 1 asserts that $X \in X_{bound}$ provided that maximum braking is applied. In the notation of hybrid automata the former is described as the invariant of the *Max Braking* state and the latter by its flow. In the *Control* state we impose the invariant $X \in X_{safe}$ and do not put any restrictions on the flow. That is, any control law may be applied in this state. The conditions of Theorem 1 are satisfied as soon as we deploy the transition from *Control* to *Max Braking* which is specified to occur as soon as $X \notin X_{safe}$.

See also [58] where a three-state regulation layer controller is designed as a hybrid automaton and its safety properties are verified use game-theoretic methods.
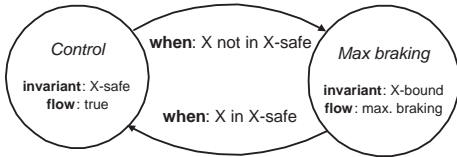
Figure 2: Safe hybrid controller for merge maneuver.

2.2.2 Communications example: The communications requirement for AHS are diverse and a variety of communication protocols and technologies are needed. Traditionally communication and control have been done separately. We will demonstrate below how hybrid systems are used to incorporate communication protocols and control theory in a natural manner.

Two classes of protocols have received the most attention. The first are protocols whose properties – fairness, liveness and deadlocks – are formally verified. These protocols and properties are formulated within a formal framework such as FMSs or timed automata and the delivery of messages are modeled as events. In [72]

a fault-diagnosis protocol is presented. It is claimed that in a platoon consisting of N vehicles with a persistent receiver or transmitter fault of a single vehicle, the protocol will identify the faulty vehicle within a finite number of protocol steps. The protocol is specified with Shift and the correctness of the claim is verified using Kronos [25]. Similarly, in [37] coordination layer communication protocols for a fault-tolerant AHS are specified as FSMs and their properties are verified using Cospan.

The second class of results pertains to protocols that analyze the implications of communication latency for controller performance such as stability. The control system is modeled as a sampled data system and the protocols are parameterized in terms of sampling rate and data interpolation methods. For example, in [82] results are presented for the stability analysis of a class of linear networked control systems with distributed sensors.
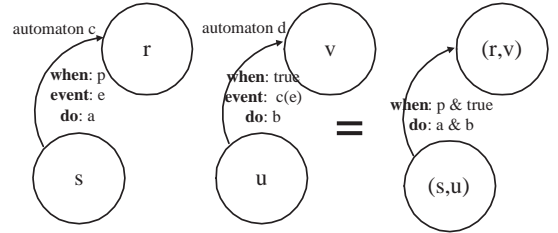
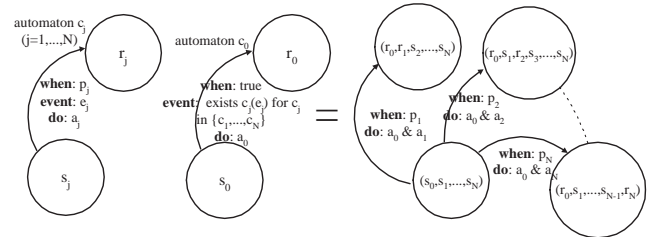**Figure 3:** One-to-one composition of automata.

Figure 4: Existential composition of automata.

Hybrid systems admit a compact representation of both types of protocols mentioned above. The former are described using the inherent FSM of the hybrid automaton and the latter using timed transitions. Furthermore, the DNHA computation model admits a compact representation for dynamic scheduling and network reconfiguration. We demonstrate these concepts in the following example.

Consider again the regulation layer merge maneuver controller of Figure 2. Suppose there are are N such controllers, denoted by $c_1, \ldots, c_N$, interacting in closed loop. Denote the control variable of controller c_j by x_j

and that the control law for c_j is given by

$$\dot{x}_j(t) = f_j(x_j, z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_N)$$
$$z_i(t) = x_i(t), t \in [t_0, t_0 + T). \qquad (4)$$

The objective is to extend the controller c_j with a safe real-time data communications. We demand that N is not fixed – that is, the network may change over time.

The transmission of a message is modeled using composition. This is illustrated in Figure 3. Given two automata c and d that agree on an event e, their composition is an equivalent automaton as illustrated. Notice that the event need not appear in the composite automaton. The actual data delivery is captured by the action of the composite automaton. For example, the actions a and b may stand for "transmit message" and "receive message" respectively. We say that c transmits a message to d on the event e when the enabling conditions p is true (if the enabling condition of d is not necessarily true then d may block c from transmitting).

The 1-to-1 synchronization is sufficient to model a static communications network. However, when the network is dynamic – that is, nodes in the network may appear/disappear – then more sophisticated data structures are needed. DNHA provides such constructs based on first-order predicate logic. Existential composition is illustrated in Figure 4. N similar automata denoted by c_j, $j = 1, \ldots, N$ may transmit a message on the event e_j. Suppose that c_1 is ready to transmit – *ie.* the enabling condition p_1 is true. Then, the distinguished automaton c_0 existentially quantifies c_1 out of the set $\{c_1, \ldots, c_N\}$ and executes a synchronous transition as illustrated in the composite automaton. The important properties are (i) c_1 is made available in the action a_0 and (ii) $c_2, \ldots, c_N$ are blocked during the synchronous transition. Since N is not fixed, it is possible to model dynamic communication networks.

Using these composition constructs we design a time-division medium access protocol as illustrated in Figure 5. Fix a controller c_j and consider its *Control* state. We divide time into slots of length T/N. At the jth time slot the controller broadcasts its control data. This is captured using the self-transition from the *Control* state on the event e_j. All other controllers receive a message as a result of the self-transition on the existentially quantified event e_j. This satisfies the real-time requirements for the controller c_j as given by (4). A similar protocol is applied to the *Max. Braking* state.

To each controller is associated a fault-diagnosis protocol d_j. The object d_j continuously monitors the system and in case of a fault signals the event *abort*. We do not require d_j to be a hybrid automaton. We only require it to agree on the event *abort* with the con-

troller c_j.[2] Thus we are free to use an off-the-shelf fault-diagnosis protocol whose safety properties have already been verified (for example, see [37]). The controller c_j transitions to a state called *Abort* on the event *abort*. The synchrony laws ensure that this transition is not missed. We have not specified the invariant and flow for the *Abort* state. We assume that a suitable control law may be specified. This satisfies the safety requirements for the controller c_j.
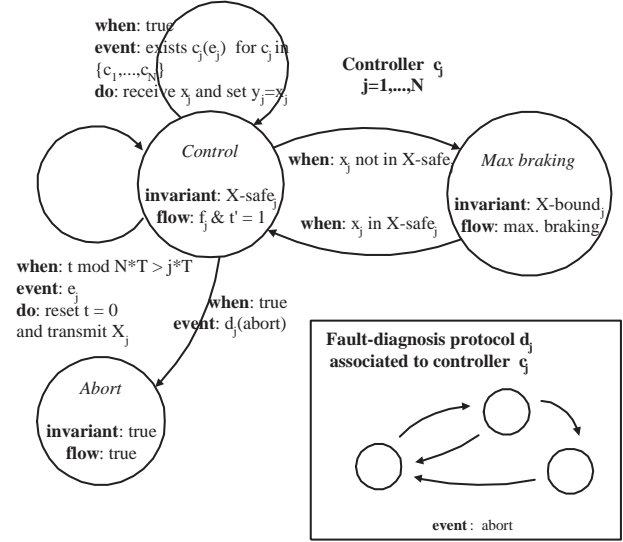

Figure 5: Safe controller with real-time communications.

2.3 Motivations for hybrid systems

The previous sections demonstrated the basic use of hybrid systems to model a safe merge controller design and a safe communications protocol design. We address the question "what is the advantage of the hybrid systems formulation?" under the following topics.

Scope. The controller and protocol design examples demonstrate that hybrid systems model a wide range of problems.

Extension. In the controller design example the flow of the *Control* state is not specified (see Figure 2). We say that the partially specified hybrid automaton is abstract. That is, it defines a class of such hybrid automata. Recent advances in formal methods for hybrid systems provides us with a collection of tools and techniques to extend or specialize this abstract hybrid automaton. See Section 3.3.

Simulation. The structure of hybrid automata allows us to program the desired behavior of controllers. Tools are available to simulate the behavior of a given program. These tools which are summarized in Section 3.3

---

[2]Many fault diagnosis protocols are specified as either FSMs or timed automata.

provide a formal syntax and semantics to describe the behavior (either graphically or programmatically or both) and a means to observe simulation traces.

Implementation. The programmatic nature of FSMs admits the description of hybrid automata as formal programs that can be processed for deployment on real systems. Tools (see Section 3.3) help to generate platform specific code from a simulation model.

Verification. Several results and tools exist for the verification of hybrid automata. That is, given a hybrid automaton and a formal specification, these tools check whether the hybrid automaton satisfies the specification. Note that in the safety design for the communications protocol example we relied on verification techniques for FSMs and timed automata.

Computational techniques. The study of hybrid automata has lead to advances for several computational techniques. These include algorithms that accurately detect guard crossings, synthesis of switching conditions that satisfy a given formal specification and filters to eliminate or reduce jitter – *ie.* excessive switching amongst the states of a hybrid automaton about a nominal state or trajectory.

Presentation. The structure of hybrid automata admits a natural graphical representation as in Figure 2.

3 Distributed hybrid systems

3.1 Introduction
Informally, a distributed hybrid system is a collection of hybrid automata that interact through the exchange of data and messages. Here, we provide an overview of distributed hybrid systems research. Before doing so, we succinctly describe the control systems and computer science backgrounds that paved the way to this research.

Several topics in control theory are of value for the research on hybrid systems:

Optimal control. Optimal control produced a body of theory and results that has been fundamental for hybrid systems research: 1) Witsenhausen [83] produced early ground-breaking work in hybrid systems; 2) the Pontryagin maximum principle – which gives necessary conditions for optimality – provides for an operational characterization of the reach set [80]; 3) the principle of optimality and dynamic programming provide an interpretation of reach sets and solvability sets [3] in terms of the level sets of the value functions of some specific optimization problems (see [49]), and also for a

---

[3] A solvability set [49] is the set of all initial conditions from which a target set can be attained

method for state feedback control synthesis; 4) Bensoussan contributed conditions of optimality for systems where the control space includes measures, thus allowing for discontinuous trajectories [12]; 5) the optimal control formulation also allows to express relations between optimality and several forms of invariance in terms of hamiltonian equations (see [35], [45]); 6) control of an ordinary differential equation subject to positive switching costs is addressed in [20], where it was proved that the corresponding value functions are viscosity solutions to the dynamic programming quasi-variational inequalities (the corresponding differential game formulation was introduced in [84] as a zero-sum differential game).

Viability theory. Viability can be described as follows: given a dynamical system, a set K, and a set of initial conditions that lie within K, synthesize a control law that ensures the state of the system never leaves K [9]. The concepts and techniques from viability theory, that uses techniques from set-valued analysis (see [11]), have an natural extension to hybrid systems.

Differential games. The setting here is that of a dynamic optimization problem where the control inputs are partitioned into at least two classes: 1) those available for controlling the system, 2) those available to the adversary or the disturbance. This setting extends that of deterministic optimal control to the case where a stochastic characterization of the disturbances is not available, preventing the formulation of a stochastic optimal control problem. Some important observations are: 1) the reach set is different under closed and open-loop controls, 2) the principle of optimality does not always apply, 3) there are conditions under which closed-loop and open-loop controls produce the same results but, in general, the results achieved with the class of closed-loop strategies cannot be achieved with open-loop controls ([45], [51]).

Automata and supervisory control theories. These theories are used to model and control the behavior of discrete event dynamical systems (see [46], [21]), an important component of the behavior of hybrid systems.

Switched controls. There is a large research on switched controls that precedes work on hybrid systems. The problem consists in selecting a strategy to switch between several control laws to achieve some goal [55]. On one hand, switching controls arise naturally in systems with discrete actuator settings. The controller has to switch between these settings. On the other hand, switching is sometimes essential. For example, some systems cannot be stabilized with a continuous feedback law [22]. However, the introduction of discontinuous feedback laws is not trivial, since those tend to be quite sensitive to measurement errors [74]. Krasovskii proposed what amounts to be a precursor to hybrid

controllers to solve this problem [45].

Computer science brought new concepts and techniques into play:

Formal methods. for specification, analysis and design. This is an significant topic in computer science research (see [78]), and can be important for hybrid systems: 1) formal methods provide a basis for computer aided verification; 2) computer scientists are developing logical theories within which specific properties can be formalized and analyzed (e.g., [6, 4, 5]); 3) results from game theory have been used and extended to model systems, and to prove properties [26].

Computer-aided verification. The verification problem can be described as follows: given a discrete transition system, a controller, and a set of properties check if the system indeed satisfies these properties. This problem is solved algorithmically. Research resulted in the development of a considerable number of techniques and software tools (e.g., [47],[25]).

Concepts and terminology. Formal methods and logic theories introduced the possibility to express concepts and properties that are of importance for hybrid systems. Examples of those are the notions of fairness – that is studied in the context of infinite runs of transition systems – and of least restrictive control – a control that satisfies some prescribed properties, and which is specified as a set of options from which the one applied to the system is selected.

The developments in hybrid systems research seem now predictable in terms of this background. We propose to discuss these developments in terms of: 1) models, 2) languages and tools, 3) analysis and control synthesis. Space limitations prevent us from presenting an exhaustive survey. We opted to informally organize each subsection into two parts: the first part introduces the baseline research for the topic under discussion, the second part provides pointers to more specific material.

3.2 Models

Any model of the operation of multiple interacting dynamic systems should be able to capture two essential features of these systems: 1) switched mode operation and, 2) dynamic interactions.

Hybrid automata. Hybrid automata [2] are quite convenient to model a switched mode control system. We present a formal definition of a hybrid automaton after [66].

A hybrid automaton consists of control locations with edges between the control locations (see Figure 2). The control locations are the vertices in a graph. A location is labeled with a differential inclusion, and every edge

is labeled with a guard, and a jump and reset relation. A hybrid automaton is $H = (L,D,E)$ where:

- L is a set of control locations.

- $D : L \rightarrow \textit{Inclusions}$ where D(l) is the differential inclusion at location l.

- $E \subseteq L \textit{ X Guard X Jump X L}$ are the edges - an edge $e = (l,g,j,m) \; \epsilon$ E is an edge from location from l to m with guard g and jump relation j.

The state of a Hybrid Automaton is a pair (l, x) where l is the control location and $x \epsilon R^n$ is the continuous state. Hybrid automata are classified according to the characteristics of L, D and E (see [66]).

In more abstract terms, this modeling problem consists in combining continuous time dynamics and discrete event dynamics. There are other ways to do this. Hence the diversity of models for hybrid control systems. For general representative references see [67], [18], [63] and [16]. An attempt to produce a model which subsumes the others appears in [16].

Dynamic interactions. The problem of modeling dynamic interactions is quite difficult. First, we want to be able to express links and exchange of information among different dynamic systems, for example hybrid automata. Second, there are several modalities for interactions: information exchange can take time, and depend on the communication environment. See Section 2.2 for illustrative examples.

Researchers have used DNHA to model dynamic interactions [33]. Informally, DNHS allow for interacting automata to create and destroy links among themselves, and for the creation and destruction of automata. Formally, for each hybrid automaton, there are two types of interactions: 1) the differential inclusions, guards, jump and reset relations are also functions of variables from other automata, 2) exchange of events among automata. Obviously, interactions are mediated by means of communication. Hence, a model for dynamic interactions has to include a description of the mechanisms by which automata interact. At the level of software implementation, the mechanisms by which software modules interact are called models of computation, or semantic frameworks. Hence, models of computation provide the formal basis for dynamic interactions. The choice of the model of computation (or mix of models) for a specific implementation is quite dependent on the properties of the underlying problem domain [53].

When it comes to simulation and deployment, modeling dynamic interactions requires the consideration of a full-fledged programming language that maintains data structures representing links, and their evolution with

time. This modeling problem has received significant attention in computer science, as one can infer, for example, from Milner [60]. In fact, he argues that a rich conceptual development, that gives a distinct character to the principles and concepts underlying computing, is in progress . In his claim, the distinct and unifying theme encompassing the new developments is what he calls "Information flow":

> ...information flow - not only the volume and quantity of flow, but the structure of the items which flow and the structure and the control of the flow itself.

The notion of dynamic reconfiguration is an essential element for the control of the information flow:

> Dynamic reconfiguration is a common feature of communicating systems. The notion of link, not as a fixed part of the system but as a datum that we can manipulate, is essential for understanding such systems. Is there a common notion of link which subsumes pointers, references, channels, variables (in the programming sense), locations, names, addresses, access, rights, ..., and is it possible to take this notion as basic in a computational model?... What is the mathematics of linkage?

The problem is that the realm of interactions and processes is not captured by the formal theories behind programming languages [4] . The theories of computation have to evolve from notions like value, evaluation and function to those of link, interaction and process. The Π-calculus [61] is a first attempt in this direction.

3.3 Languages and tools

The rapid growth of hybrid systems theory, technique and applications is accompanied by a menagerie of languages and tools. Early development of such tools finds origins in automata-theoretic formal methods. In particular, the formal verification tools such as Cospan [39], Spin [43], SMV [62] and PVS [64] has had a large impact in the development of formal methods for hybrid systems. These developments are followed shortly by the development of specification and modeling languages specific to hybrid systems. The structure and purpose of these tools were driven by their application domains. One such example is the Shift language [33]. Shift was designed to meet the complex modeling requirements of AHS. The underlying

---

[4]The character of each programming feature is defined by associating with it a specific space of meanings. These include domains, functions and values, and can be expressed in a specific calculus – the λ-calculus .

model, DNHA, proved adequate in a variety of problem domains , where other models and tools failed. The lessons learned from both the formal methods and specification tools suggest that the hybrid systems paradigm is sufficiently powerful and this observation is reflected in the recent development of hybrid systems based implementation tools and techniques. Such advances include methodologies and real-time code generation for hybrid systems models.

Although hybrid systems tools finds origins in the software and hardware formal methods community it turns out that the mainstream of development has been along the lines of modeling, design and analysis of software related methods. This may be due to the relatively large growth of the software community over the last few decades and the resulting availability of resources. The main trends for hybrid system tools may be summarized under the following software categories: computational packages, modeling and design tools, programming languages and implementation tools.

Computational packages. Computational packages are software packages that perform a specific computation or algorithm for hybrid systems. For example, VeriShift [14] is a C++ package that performs approximate reach-set analysis for DNHA. The Shift in VeriShift is due to underlying model which is borrowed from the programming language Shift. The mainstream for computational packages perform model checking and verification. Most packages are based on automata-theoretic concepts and hence assume some form of hybrid automata as their underlying computational models.

Hybrid automata have an inherently complex structure. Furthermore, specific algorithms rely on various sets of assumptions. For example, the decidability assumption for a computation to terminate. Thus, most computational packages are accompanied by a form of lexical parser or grammar that allows the user to specify hybrid automata while ensuring the desired structure and assumptions by construction. For example, see Kronos [86] and HyTech [76] which perform verification of timed and linear hybrid automata respectively.

Modeling and design tools. The largest trend in software tools for hybrid systems consist of modeling and design tools. These tools are largely developed as subsystems of multidisciplinary tools that support a larger class of application domains based on differential and algebraic equations, sampled systems, discrete-event and real-time systems. Examples include Simulink/Stateflow from Mathworks, Ptolemy II [56] and Omsim/Omola [17]. Such tool sets assume an underlying computational model for hybrid automata and allow the user to rapidly develop and simulate application models. These tools provide state-of-the-

art simulation engines for differential algebraic equation (DAE) and Ordinary Differential Equation (ODE) solvers and data visualization utilities. Extended tool sets provide support for sub-system analysis, synthesis and verification.

All modeling and design tools support bottom-up design methodologies. Support is provided for model libraries and application models are developed using model composition techniques. Furthermore, the nature of the bottom-up methodologies are similar – in the form of block diagrams with input/output composition semantics. This is due largely to their large application domains that inherits the state-space representation for continuous-time dynamic systems. Support is also provided for top-to-bottom design methodologies. That is, a methodology to extend a given model. Stateflow and Ptolemy II support hierarchical construction of input/output blocks and FSMs. Omsim/Omola provides support for object-orientation with inheritance. Design methodologies are enforced by graphical model editors.

A recent advance is presented by Ptolemy II which supports polymorphic models (called actors) that allow the underlying computational model to be changed.

Programming languages. Modeling and design tools allow the rapid development of hybrid systems by making use of various design techniques, model libraries and composition semantics. The emphasis is on the design methodology. Programming languages share the same ultimate goal – to design large hybrid systems. However, their emphasis in on the elementary structure of the hybrid automata. The hybrid automaton is taken as the most elementary programming construct. Its basic properties – flows, invariants, guards, events and actions – are given through a coherent set of elementary language primitives. These primitives are then used to create hybrid system based abstractions and methodologies that are specific to a given application domain. Examples of hybrid systems based programming languages are Shift [33], λ-Shift [71] and Charon [3].

The languages are greatly influenced by conventional object-oriented programming languages. The language primitives are used to describe a class of hybrid automata and support is provided for instantiation, destruction and object-management techniques such as hiding.

The characterizing features of the languages are their support for composition and abstractions. Shift allows hybrid automata to interact through dynamically reconfigurable input/output connections and synchronous composition. λ-Shift provides a generalized set of accessors whose instances include dynamic input/output connections and cause-effect event synchronization. The abstraction mechanisms for these languages include object-orientation, type systems and inheritance.

A coherent collection of language primitives provides a framework to reason about the language as a self-contained theory. This has lead to many important advances for hybrid systems. For example, the first order predicate constructs of Shift (*eg.* existential and universal quantification) were found useful to provide compact representations of dynamic synchronous composition. In Section 2.2 we illustrated the use of this technique to implement a dynamic communication protocol for an AHS merge maneuver controller. Another example is the exception handling/debugging framework of λ-Shift. The basic parts of a hybrid automaton – flows, invariants, guards, events and actions – are given as language primitives the language exploits their atomic properties to define classes of conditions specific to hybrid automata. For example, a class of *flow conditions* may be used to describe the positive definiteness of the state matrix of a symmetric differential Ricatti equation.

Implementation tools. The lessons learned from the computational and modeling tools and programming languages suggests that hybrid systems paradigms are sufficiently rich and understood well to be applicable to real-world systems. Implementation tools attempt to deploy a given algorithm, model or program on a real system. There are two main categories for such tools: code-generation and real-time hybrid automata based programming languages.

The code-generation community is influenced largely by the success of modeling and design tools. Given a model the procedure is to generate software that is capable of executing in an open environment such as an embedded control application. The difficulty is that the models contain many abstractions that do not admit an (efficient) implementation. The technique is to restrict models at design time or to refine them at code-generation time. Simulink/Stateflow and Ptolemy II provide support for real-time code generation for several target architectures.

The programming language community does not envision code generation as a separate process. The primitive features of the language are designed to admit an efficient atomic implementations. The overall implementation of a given application is then derived from the compositional and abstraction properties of these elementary constructs. The advantage of this approach is that the programmer may reason about the efficiency of an implementation in terms of these primitives. This leads to new methodologies for implementing hybrid systems. An example of a hybrid systems based im-

plementation tool is Teja which finds applications for the efficient implementation of communication related protocols on network processors.

3.4 Analysis and control synthesis

Analysis. Researchers have extended Lyapunov theory to accommodate the intricacies of the stability analysis for hybrid systems. Branicky [16], proposes multiple Lyapunov functions, and Sontag et al. nonsmooth control Lyapunov functions [75].

See [32] for recent perspectives on stability of hybrid systems, and ([16], [66]) for discussions on hybrid systems analysis. For a detailed discussion on qualitative properties of hybrid systems see [59]. The control geometric perspective for the analysis of hybrid systems is provided in [70]. Techniques from non-smooth analysis are useful for the analysis of hybrid systems [23].

Control synthesis. The problem of control synthesis can be described as follows: given a dynamic system, or a collection of interacting dynamic systems, synthesize a controller so that the system(s) satisfies(y) some specifications. The type of specification dictates the type of control formulation. Examples include: 1) the problem of invariance; 2) the problem of attaining a given target set while the trajectories of the system remain inside some other set; 3) the problem of optimizing some criteria; 4) the problem of stabilizing a system. These control formulations can be further decomposed in terms of information sets (addressing the problem of decision under complete information), and open and closed-loop control strategies ([44], [49] and [48]). Inherent to most of these control formulations is the problem of reach set computation. In fact, given the reach set, it is quite simple to solve most of these problems.

The problem of reach set computation has attracted the attention of both the control and the computer science communities.

The development of algorithmic approaches to reach set computation was leveraged by the availability of tools for computer aided-verification, such as the ones mentioned before. Hybrid systems in which a problem can be solved algorithmically in a finite number of steps are called decidable. The general problem of reachability of hybrid systems is undecidable. Several techniques for reachability analysis of hybrid systems have been proposed. They can be (roughly) classified in two kinds: 1. Purely symbolic methods based on (a) the existence of analytic solutions to the differential equations and (b) the representation of the state space in a decidable theory of the real numbers. 2. Methods that combine (a) numeric integration of the differential equations and (b) symbolic representations of approximations of state space typically using (unions

of) polyhedra or ellipsoids. These techniques provide the algorithmic foundations for the tools that are available for computer-aided verification of hybrid systems ([85] [25] , [40], [76]). The use of these tools usually presupposes the existence of a controller, whose properties have to be verified, or whose parameters have to be synthesized. From a theoretical point of view, and in order to study control problems for hybrid systems, Henzinger et al. [41] generalize hybrid automata to hybrid games-say, controller vs. plant. The observation is that if we specify the continuous dynamics by constant lower and upper bounds, we obtain rectangular games. They show that, for rectangular games with objectives expressed in LTL (linear temporal logic), the winning states for each player can be computed, and winning strategies can be synthesized.

Reach set computation for hybrid systems has received considerable attention from the control systems community. Varaiya suggests an approach for reach set computation based on the Pontryagin maximum principle of optimal control theory, and illustrates the main concepts for the case of linear systems [80]. Broucke presents an approximate verification method for hybrid systems in which the reach sets of the automaton are over-approximated, while leaving the vector fields intact [19]. The method is based on a geometrically-inspired approach, using tangential and transversal foliations, to obtain bisimulations. Kurzhanskii and Varaiya use dynamic programming techniques to describe reach sets and related problems of forward and backward reachability [50]. These problems are formulated as optimization problems that are solved through the Hamilton-Jacobi-Bellman equations. The reach sets are the level sets of the value function solutions to these equations. This approach also accommodates the problem of reach set computation under uncertainty or under closed and open-loop control [49]. The Verishift tool implements this approach for linear systems [14]. This implementation uses techniques from ellipsoidal calculus to solve the Hamilton-Jacobi-Bellman equations [49].

One of the major difficulties for control synthesis is the complexity of the underlying models. One approach to deal with this complexity is to use simpler abstracted models that propagate desired properties of the original system (see [65], [1]).

The problem of control synthesis is that, in general, the controller is not know in advance – it has to be synthesized. Research in control systems resulted in several approaches for control synthesis.

Branicky [16], and Branicky et al. [15], used the impulse optimal control framework for synthesizing hybrid controllers for hybrid plants. In this framework, they proved the existence of optimal (relaxed) and

near-optimal (precise) controls and derived "generalized quasi-variational inequalities" that the associated value function satisfies. They have also proposed algorithms for solving these inequalities based on a generalized Bellman equation, impulse control, and linear programming.

See [32] for recent perspectives on stabilization of hybrid systems.

Deshpande and Varaiya study the problem of viable control for hybrid systems [34]. Aubin [10] discusses the problem of invariance for hybrid systems under impulse controls in the framework of viability of control systems. He does that with the introduction of impulse differential inclusions. An impulse differential inclusion is a pair (F, R), where the set-valued map $F : X \mapsto X$, mapping the state space $X : \Re^n$ to itself, governs the continuous evolution of the system in some closed set K, and R, the reset map, governs discrete switches to new "initial conditions" to prevent the continuous evolution to leave the set K.

Lygeros et al. [57] and Tomlin et al. [77] have used techniques from optimal control and game theory to design controllers for safety specifications in hybrid systems. Their methodology consists of three phases. First, they translate safety specifications into restrictions on the system's reachable sets of states. Second, they formulate a differential game and derive Hamilton-Jacobi-Bellman equations whose solutions describe the boundaries of reachable sets. Third, they synthesize the hybrid controller from these equations. The controller assumes the form of a feedback control law for the continuous and discrete variables, which guarantees that the hybrid system remains in the "safe subset" of the reachable set. They also discuss issues related to computing solutions to Hamilton-Jacobi equations.

4 Shift and SmartAHS

The AHS is an IVHS policy that aims to increase highway capacity, safety and efficiency through automated control of vehicles (see Section 2). The design of an AHS is a challenging task and several proposals consisting of different control strategies have been made. An important part of the design process is the modeling and simulation of the alternate strategies. In this section we describe the hybrid systems based programming language Shift and its application to the AHS simulation framework SmartAHS.

Shift/SmartAHS was created to provide a software modeling and simulation framework for the evaluation and objective comparison of alternate AHS control strategies. The AHS is a complicated system and this is reflected in the requirements imposed on a simula-

tion framework for AHS. For a detailed description of these requirements see [38]. We include a short summary here:

Modeling. Traffic modeling has traditionally been done using macro and meso-simulators based on fluid flow models. However, due to the reduced role of the human driver, the automated vehicle behaves deterministically. Thus, the most accurate information is obtained if simulation is at the individual vehicle level.

Granularity. The multiple goals of AHS – capacity, safety, efficiency and comfort – require that a given AHS strategy be evaluated at multiple levels of granularity.

Semantics. The AHS is a dynamic network of interacting vehicles. The simulation framework must be capable of creating, destroying and reconfiguring the vehicles dynamically – *ie.* during a simulation (for example, see existential quantification in Section 2.2.2). The framework must provide explicit support for continuous and discrete behaviors of system components.

Abstractions. The simulation framework involves several categories of users including control engineers, system analysts and system planners. Thus, the framework must provide abstraction facilities to associate physical and logical representations of system components.

The Shift/SmartAHS approach is to tackle the AHS simulation requirements in two steps. First, primitive support for the system-theoretic concepts is provided by a single object-oriented programming language with well-defined simulation semantics. Then, the abstraction facilities of this language is used to implement basic classes of system components that may then be extended and combined to realize a particular AHS scenario (for example, in [7] and [36] the Shift/SmartAHS framework is deployed to study Adaptive Cruise Control and Multiple Merge Junctions under AHS operation respectively). The immediate advantage of this approach is that the system-theoretic concepts may be re-used in various other application domains. Shift was used to create simulation frameworks for Autonomous Underwater Vehicles [30] Unmanned Aerial Vehicles and Mobile Offshore Bases [29].

4.1 Shift overview
Shift is an object-oriented programming language with simulation semantics. Its computation model is the DNHA. The world is described by classes of hybrid automata and their interactions. A simulation world is populated by instances of these classes.

Primitives. The basic programming entity is a hybrid automaton. The behavior of a hybrid automaton is

described in discrete modes of operation – by Ordinary Differential Equations and algebraic read-out relations – and discrete transitions amongst these modes – by guards, events and actions:

```
type Sedan : Vehicle {
 input      // what we feed to it
 output     // what we see on the outside
 state      //  what's internal
 discrete   // discrete modes of operation
 export     //  event labels

 flow       //  continuous behavior
 transition // discrete behavior
 setup }    // initialization
```

The inputs, outputs and state variables define the encapsulated variables of the class. Shift is a *strongly typed* programming language. The type system and the type-checking compiler ensures that a user writes correct Shift programs. The class definition implicitly defines an abstract type. The built-in types are numbers, continuous numbers, symbols, links[5] and sets and arrays of these types. The language provides primitives to program the flows, guards, events and actions using typed expressions on these variables. These primitives include the first order predicate calculus on sets and arrays that are used, for example, to realize the dynamic existential composition illustrated in Section 2.2.2.

Combination facilities. The language provides facilities to construct aggregate objects from primitive classes of hybrid automata. There are three mechanisms. The first is given by support for abstract variable types. The expressions in a flow, guard or action may use the output variables of a link:

```
type Vehicle {
 input Vehicle vehicle_in_front;
 output continuous number speed;
 flow default
  { speed' = speed
      - speed(vehicle_in_front); }}
```

The second is given by a mechanism to establish input/output connections as a side-effect of the setup clause or action of a discrete transition. The third is given by the dynamic synchronous composition semantics. In Section 2.2.2 we demonstrated the existential composition construct to implement dynamic communication protocols. Similar constructs are defined for universal composition.

Abstraction facilities. The input, output and state declarations define an input/output encapsulation for hy-

---

[5]Links are generalized references to abstract Shift types.

brid automata. Furthermore, the class system is complemented by an inheritance mechanism that is used to define encapsulated abstract component interfaces. For example, the Sedan type in the example above is specified to be a subtype of Vehicle from which it inherits the input vehicle_in_front and the output speed.

4.2 SmartAHS architecture

SmartAHS is a simulation framework for the specification, simulation and evaluation of AHS strategies. SmartAHS is developed using Shift. The Shift primitives are used to build libraries of abstract classes of hybrid automata that model system components such as sensors, actuators, engine models and road segments. The combination facilities of the language are then used to build generalized models of vehicles and roadside devices as aggregations of the library elements. The abstraction facilities are used to specialize the generalized models to a particular scenario which are then instantiated to populate a simulation. The basic component libraries consist of road-side elements and vehicle components as illustrated in Figure 6.
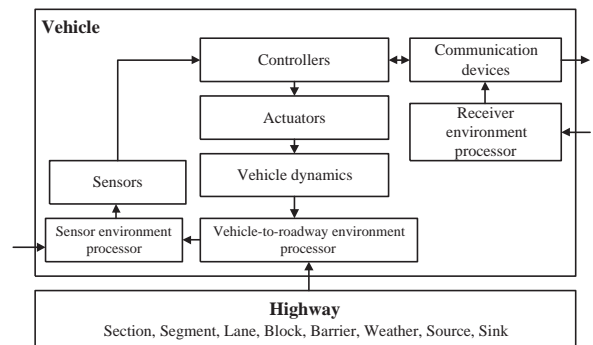

Figure 6: SmartAHS components. Figure from [69].

Road-side elements. The road-side library provides basic elements to model a highway. A highway is modeled as an aggregation of these elements as illustrated in Figure 6. Lanes model the right of ways for driving, segments model the geometric characteristics and sections model contiguous highway stretches. The abstract Shift interface for a Segment is given by:

```
type Segment {
 output
 number length;         // length of left edge
 number xOffset;        // length ...
 number gxa,gya,gza;    // global coordinates
 number orientation;    // angle between ...
 ...
 Weather weather;       // weather condition
 Section section;       // segment's section
 Segment upSegment;     // upstream segments
 Segment downSegment;} // downstream segments
```

Note that a segment itself aggregates other library primitives such as `Weather` objects and even `Segment`.

To illustrate the extension mechanism we may wish to associate a Segment with a Traffic Management Center. Assuming that we have defined such an abstract type the specialization follows as:

```
type custom_segment : segment {
  TMC tmc; } // Traffic management center
```

All outputs defined for the `segment` type are inherited.

Complementary tools such as Carmma [69] have beed developed that allow the graphical specification of complex highway scenarios. Such tools generate the Shift code that instantiates lanes, segments and sections that model a complex highway.

Vehicle components. The highest level vehicle components presented to the user are illustrated in Figure 6. A vehicle is modeled as an aggregation of these basic components. Each component is itself an aggregation of more basic counterparts. For example, the `VehicleDynamics` interface defines an interface for component that outputs the vehicles linear and rotational speeds. The library itself is further populated with specializations of this interface to kinematic and dynamic vehicles. The former are used for large simulations (*eg. ¿* 1000 vehicles) and the latter for smaller simulations (*eg.* 10-100 vehicles). For example, a typical dynamical vehicle adds a lumped engine, powertrain and steering model to the basic `VehicleDynamics` interface.

For a detailed description of SmartAHS components see [69]. We conclude our presentation with a Shift code segment that illustrates the use of hybrid automata to model the Vehicle-to-roadway environment processor (VREP). The purpose of this component is to provide a localized coordinate frame for other vehicle components such as the dynamics and controllers. The following code segment illustrates the automatic road-side management performed by a VREP components. The VREP is associated to a vehicle and as the vehicle moves to a next section, its lane, segment and section are updated. These updates are modeled as a discrete transition of the hybrid automaton that models the VREP:

```
 cruise -> cruise {updateSection}
 when rxp > length(section)
      and size(laneDown(lane)) > followLane
 define { ... }  do { ... };
```

The VREP is assumed to be in a discrete operation mode called `cruise`. When the relative position `rxp`

satisfies the given condition a self-loop to this discrete mode is executed. The transition is closed on the event `updateSection` which may be used to synchronize with other transitions of other components. The updates (not shown) are performed in the `define` and `do` clauses.

5 Conclusions

From the above it is clear that traditional questions in control have been reopened and investigated from the perspective of hybrid systems. Some of the main difficulties are still there. One example is the problem of solving Hamilton-Jacobi-Bellman equations, that is still seen as the Holy Grail in control. Another example is reach set computation for hybrid systems. In most simulation tools a transition is taken when the corresponding guard becomes true – the *as soon as possible* semantics. This may not be the case in reality, where the transition may take place at a later time. This behavior is known as laziness – the notion of laziness explicitly distinguishes between the enabling and the firing of an event in a transition system. Reach set computation for lazy systems is an open problem.

New questions, that could not have been formulated a few years ago, are being posed, mainly due to the technological advancements in computing and communication. Again, these technological developments made possible to envision the implementation of systems which could have not been imagined before. Consider the following example:

Example 1 *Let us look at the problem of real-time oceanography and adaptive sampling strategies [24][6]. Adaptive sampling with multiple vehicles can be used as the motivation for formulating a more abstract problem that has the potential for being applied in other domains.*

Consider a group of vehicles that, in turn, is organized into two sub-groups. The first sub-group acts as a navigation device for the second sub-group. It does this by broadcasting accurate positions of its members, along with timing information. In fact, this group acts as a Local Positioning System (LPS) – quite useful for underwater applications. The second group – hereafter designated as the sensor sub-group – acts as a distributed sensor by collecting data in spatially distributed

---

[6]The next generation of oceanographic field programs requires economic access to the ocean. Imperatives include abilities to: 1) obtain spatially distributed, temporally correlated measurements; 2) respond in a timely fashion to episodic events; 3) obtain time series of spatially distributed phenomena, and 4) to interact with measurement platforms in the course of observations. We refer to these collective requirements as real-time oceanography.

locations, and by integrating that data in order to determine the sampling strategy – the motions of this group with respect to the phenomena under observation. Obviously the two sub-groups exchange information. The same happens among the vehicles in each sub-group. Communications are constrained in terms of bandwidth and range.

The two sub-groups have to coordinate their motions and activities: 1) the relative positions of the vehicles for each sub-group should satisfy the local (sub-group) communication constraints; 2) the relative positions of the two sub-groups should satisfy global (inter-group) communication constraints; 3) the spatial arrangement of the sensor sub-group is dictated by measurements; 4) the spatial arrangement of the LPS sub-group is dictated by optimal positioning constraints; 5) the LPS sub-group follows the distributed sensor sub-group in order to provide positioning information to this sub-group; 6) the sensor sub-group has to incorporate the motion constraints imposed by the operation of the LPS in the sampling strategy .

Several issues arise from this discussion: 1) the role and relative position of each vehicle may change with time; 2) the role and relative position of each vehicle are determined by logical links among the vehicles of the group; 3) each sub-group, when properly positioned, exhibits a functionality, for example LPS; 4) the whole group, exhibits another functionality, whose performance is a function of the other two functionalities; 5) the configuration of the links is controlled and determines first the performance of each sub-group and, then, the performance of the ensemble.

This rather conceptual example serves to motivate and illustrate some important points [31]:

1. The need for an integrated perspective.

2. The need for methods to model and design the structure of interacting systems, in short, their architecture, and to capture properties not to be found in constituent modules. For example, the concept of controlling links to achieve functionalities that, in themselves, have a meaning and a value attached. This seems to require new semantic concepts and, possibly, a different perspective on control.

3. The problems of the expressiveness of specification languages and mapping of requirements onto designs.

The integrated perspective is essential to articulate the modeling and design process, and to incrementally adjust and adapt requirements and designs. This issue has been informally discussed by Sloman [73]. According to him behaving systems inhabited two linked spaces. One is the space of designs – a design is an abstract specification which can be instantiated in working systems that fit the design. The other is the niche space – the space of requirements. Both spaces are layered in that regions within them can be described at different levels of abstraction. Relationships between a design and a niche space can have a complex form. Satisfaction can be interpreted as a function from a design and a niche to a partially ordered set of descriptions of the type and kind of satisfaction.

Hoare [42] and Lamport et al. [52] have discussed set-theoretic models as a basis to develop new and simple theories. The elegant simplicity and expressiveness of untyped specification languages based on set-theoretical models is discussed in [52]. Research on nonsmooth analysis has established important connections between the functional and the set theoretic representations. The later constitute a privileged environment to reconcile those representations, as well as operations among them.

The representation and control of structures and links requires the consideration of mappings from some adequate state-space to discrete configuration spaces. The problem of expressing logically constrained changes of configurations can be addressed in the context of a specific instance of switched controls with graph constrained strategies.

As conclusions we contend that there is great potential for hybrid systems research, and identify possible avenues of research. New developments should be triggered by the interactions between engineering practice and its theoretical background.

If it is true that this paper reflects the opinions - and hence the biases - of the authors, we hope that this discussion motivates further discussions, thus contributing to the advancement of hybrid systems research.

References

[1]   Discrete abstractions of hybrid systems. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):971–84, 2000.

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

[3]   R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in charon. In *Proceedings of the HSCC'00, 3rd International Workshop on Hybrid Systems: Computation and Control*, 2000.

[4]   R. Alur and T.A. Henzinger. A really temporal logic. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 164–169. IEEE Computer Society Press, 1989.

[5]   R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 390–401. IEEE Computer Society Press, 1990.

[6]   R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 100–109. IEEE Computer Society Press, 1997.

[7]   A. Antoniotti, A. Desphande, and A. Girault. Microsimulation analysis of multiple merge junctions under autonomous ahs operation. In *Proceedings of Conference on Intelligent Transportation Systems ITSC '97*, pages 147–52. IEEE, 1997.

[8]   M. Antoniotti. A first cut micro simulation of circadian rhythms as a hybrid system. 2001.

[9]   Jean-Pierre Aubin. *Viability theory*. Birkhauser, 1991.

[10]   Jean-Pierre Aubin. Impulse differential equations and hybrid systems: A viability approach. Lecture Notes, University of California at Berkeley, 2000.

[11]   Jean-Pierre Aubin and Helene Frankowska. *Set-valued analysis*. Birkhauser, 1990.

[12]   A. Bensoussan and J. L. Lions. *Impulse control and quasi-variational inequalities*. Gauthier-Villars, 1984.

[13]   C. Bizingre, P. Oliveira, A. Pascoal, F. Lobo Pereira, J. P. Pignon, E. Silva, C. Silvestre, and J. Borges de Sousa. Design of a mission management system for the autonomous underwater vehicle marius. In *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology*, pages 112–121. IEEE, 1994.

[14]   O. Botchkarev. Ellipsoidal techniques for verification of hybrid systems. January 2000.

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

[16]   Michael Branicky. *Studies in Hybrid Systems: Modeling, Analysis and Control*. PhD thesis, MIT, 1995.

[17]   M.S. Branicky and S.E. Mattsson. Simulation of hybrid systems in omola/omsim. In *Computer Aided Control Systems Design*, pages 15–20. Permagon, 1997.

[18]   R. W. Brockett. Dynamical systems and their associated automata. In R. Menicken U. Helmke and J. Saurer, editors, *Mathematical Theory and Applications*, pages 29–57. Akademie Verlag, 1994.

[19]   M. Broucke. A geometric approach to bisimulation and verification of hybrid systems. In *Proceedings of the 37th IEEE Conference on Decision and Control Conference*, pages 4277–82. IEEE, 1998.

[20]   I. Capuzzo-Dolcetta and L. C. Evans. Optimal switching for ordinary differential equations. *SIAM J. Control and Opt.*, 22(1):143–161, 1984.

[21]   C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1999.

[22]   F. H. Clarke, Y.S. Ledyaev, E.D. Sontag, and A.I. Subbotin. Asymptotic controllability implies feedback stabilization. *IEEE Transactions on Automatic Control*, vol.42,(no.10):1394–407, 1997.

[23]   F. H. Clarke, Y.S. Ledyaev, R. J. Stern, and P.R. Wolenski. Qualitative properties of trajectories of control systems: A survey. *Journal of Dynamical Control*, (1):1–48, 1995.

[24]   T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous ocean sampling networks. *Oceanography*, 6(3):86–94, 1993.

[25]   C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool kronos. In *In Hybrid Systems III, Verification and Control*, Lecture Notes in Computer Science, pages 1066–1080. Springer Verlag, 1996.

[26]   L. de Alfaro and T.A. Henzinger. Concurrent omega-regular games. In *Proceedings of the 15th Annual Symposium on Logic in Computer Science*, pages 141–154. IEEE Computer Society Press, 2000.

[27]   J. Borges de Sousa and A. Deshpande. Real-time multi-agent coordination using diadem: applications to automobile and submarine control. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1769–74. IEEE, 1997.

[28]   J. Borges de Sousa, A. Girard, and K. Hedrick. Real-time hybrid control of mobile offshore base scaled models. In *Proceedings of the American Control Conference*, 2000.

[29]   J. Borges de Sousa, A. Girard, and N. Kourjanskaia. The mob shift simulation framework. In *Proceedings of Third International Workshop on Very Large Floating Structures*, pages 474–482, 1999.

[30]   J. Borges de Sousa and A. Gollu. A simulation environment for the coordinated operation of multiple autonomous underwater vehicles. In *Proceedings of the 1997 Winter Simulation Conference*, pages 1169–75, 1997.

[31]   J. Borges de Sousa and F. Lobo Pereira. Some questions about hybrid systems. In *submitted for an invited session at the European Control Conference 2001*, 2001.

[32]   R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–82, 2000.

[33]   A. Deshpande, A. Gollu, and L. Semenzato. The shift programming language and run-time system for dynamic networks of hybrid automata. Technical Report UCB-ITS-PRR-97-7, California PATH, 1997.

[34]   A. Deshpande and P. Varaiya. Viable control of hybrid systems. In *Hybrid Systems II*, pages 128–147. Springer, 1995.

[35]   F. H. Clarke et. al. *Nonsmooth Analysis and Control Theory*. Springer, 1998.

[36]   D. Godbole, N. Kourjanskaia, R. Sengupta, and M. Zandonadi. Methodology for an adaptive cruise control study using the shift/smart-ahs framework. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3217–22. IEEE, 1998.

[37]   D.N. Godbole, J. Lygeros, E. Singh, A. Deshpande, and A.E. Lindsey. Communication protocols for a fault-tolerant automated highway system. *IEEE Transactions on Control Systems Technology*, 8(5):787–800, September 2000.

[38]   A. Gollu and P. Varaiya. Smartahs: a simulation framework for automated vehicles and highway systems. *Mathematical and Computer Modelling*, 27(9-11):103–28, 1998.

[39]   R.H. Hardin, Z. Har'El, and R.P. Kurshan. Cospan [coordination specification analyzer]. In *Proceedings of CAV'96*, pages 423–7. Springer-Verlag, 1996.

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

[41]   T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In J.C.M. Baeten and S. Mauw, editors, *CONCUR 99: Concurrency Theory*, Lecture Notes in Computer Science 1664, pages 320–335. Springer-Verlag, 1999.

[42]   C. A. R. Hoare. Algebra and models. In Ian Wand and Robin Milner, editors, *Computing tomorrow : future research directions in computer science*, pages 158–187. Cambridge University Press, 1996.

[43]   G.J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):279–95, May 1997.

[44]   A. N. Krasovskii. *Control under lack of information*. Birkhauser, 1995.

[45] N.N. Krasovskii and A.I. Subbotin. *Game-theoretical control problems*. Springer-Verlag, 1988.

[46] R. Kumar and V. Garg. *Modeling and control of logical discrete event systems*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1995.

[47] R. P. Kurshan. *Computer-aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton University Press, 1994.

[48] A. B. Kurzhanskii. *Advances in nonlinear dynamics and control : a report from Russia*. Birkhauser, 1993.

[49] A. B. Kurzhanskii. *Ellipsoidal calculus for estimation and control*. Birkhauser, 1997.

[50] A. B. Kurzhanskii and P. Varaiya. Ellipsoidal techniques for reachability analysis. In N. Lynch and B. Krogh, editors, *Computation and control*, Lecture Notes in Computer Science, pages 202–214. Springer-Verlag, 2000.

[51] A. B. Kurzhanskii and P. Varaiya. On reachability under uncertainty. *Siam Journal of Control and Optimization*, to appear, 2001.

[52] L. Lamport and L. C. Paulson. Should your specification language be typed? *ACM Transactions on Programming Languages and Systems*, 21(3):502–526, 1999.

[53] E. Lee and A. Sangiovanni-Vincentelli. Comparing models of computation, http://ptolemy.eecs.berkeley.edu/ eal/talks/index.html. 1996.

[54] P. Li, L. Alvarez, and R. Horowitz. Ahs safe control laws for platoon leaders. *IEEE Transactions on Control Systems*, 5(6):614–28, November 1997.

[55] D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, 1999.

[56] J. Liu, B. Wu, X. Liu, and E.A. Lee. Interoperation of heterogenous cad tools in ptolemy ii. In *Proceedings of the SPIE – The International Society for Optical Engineering*, pages 249–58, 1999.

[57] J. Lygeros, Datta N. Godbole, and Shankar Sastry. A game theoretic approach to hybrid system design. Technical Report UCB/ERL M95/77, University of California, Berkeley. Electronics Research Laboratory, 1995.

[58] J. Lygeros, D.N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43(4):522–39, April 1998.

[59] A. Matveev and A. Savkin. *Qualitative Theory of Hybrid Dynamical Systems*. Control Engineering. Birkhäuser, 2000.

[60] Robin Milner. Semantic ideas in computing. In Ian Wand and Robin Milner, editors, *Computing tomorrow : future research directions in computer science*, pages 246–283. Cambridge University Press, 1996.

[61] Robin Milner. *Communicating and mobile systems : the Π-calculus*. Cambridge University Press, 1999.

[62] A.A. Mir, S. Balakrishnan, and S. Tahar. Modeling and verification of embedded systems using cadence smv. In *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings*, pages 179–83. IEEE, 2000.

[63] Z. Manna O. D. Maler and A. Pnueli. From timed to hybrid systems. In W. Roever J. Bakker, K. Huizing and G. Rozenberg, editors, *Real-Time: Theory in Practice*, LNCS 600, pages 447–484. Springer-Verlag, 1992.

[64] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. Pvs: Combining specification, proof checking and model checking. *Computer-Aided Verification, CAV'96*, 1102:411–4, July/August 1996.

[65] G. Pappas. *Hybrid Systems: Computation and Abstraction*. PhD thesis, University of California at Berkeley, 1998.

[66] Anuj Puri. *Theory of hybrid systems and discrete event systems*. PhD thesis, University of California at Berkeley, 1995.

[67] A. Ravn R. Grossman, A. Nerode and H. Rischel, editors. *Hybrid Systems*. Lecture notes in computer science ; 736. Springer, 1993.

[68] Mary Shaw and David Garlan. Formulations and formalisms in software architecture. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, pages 307–323. Springer-Verlag, 1996.

[69] shift home page. http://www.path.berkeley.edu/ shift. 1996.

[70] S. Simic, K. Johansson, S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In *Proceedings of the Hybrid Systems 2000 Workshop*. IEEE, 2000.

[71] T. Simsek. The λ-shift specification language for dynamic networks of hybrid automata, master of science research project, university of california, berkeley. 2000.

[72] T. Simsek, R. Sengupta, S. Yovine, and F. Eskafi. Fault diagnosis for intra-platoon communications. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 3520–5. IEEE, 1999.

[73] A. Sloman. The "semantics" of evolution: trajectories and trade-offs in design space and niche space. In *Progress in Artificial Intelligence - IBERAMIA 98*, pages 27–38. Springer-Verlag, 1998.

[74] E.D. Sontag and Y.S. Ledyaev. A lyapunov characterization of robust stabilization. *Journal of Nonlinear Analysis*, 37:813–840, 1999.

[75] E.D. Sontag and H.J. Sussmann. Nonsmooth control-lyapunov functions. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2799–805. IEEE, 1995.

[76] P. H. Ho T. Henzinger and H. Wong-Toi. Hytech: The next generation. In *Proceedings of IEEE Real-Time Systems Symposium, PTSS'95*, pages 169–183. IEEE Publications, 1995.

[77] C. J. Tomlin, J. Lygeros, and S. Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–70, 2000.

[78] Jan van Leeuwen, editor. *Handbook of theoretical computer science*. Elsevier, 1990.

[79] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(3):195–207, February 1993.

[80] P. Varaiya. Reach set computation using optimal control. In *Proceedings of the KIT Workshop on Verification of Hybrid Systems*. Verimag, Grenoble, France, 1998.

[81] P. Varaiya. Control design of an automated highway system. *Proceedings of the IEEE*, 88(7):913–25, July 2000.

[82] G. Walsh, Y. Hong, and L. Bushnell. Stability analysis of networked control systems. In *Proceedings of the 1999 American Control Conference*, pages 2876–80. IEEE, 1999.

[83] H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, 1966.

[84] Jiogmin Yong. A zero-sum differential game in a finite duration with switching strategies. *SIAM J. Control and Opt.*, 28(5):1234–1250, 1990.

[85] S. Yovine. *Methods et Outils pour la Verification Symbolique de Systemes Temporises*. PhD thesis, Institut National Polytechnique de Grenoble – France, 1993.

[86] S. Yovine and A. Olivero. Kronos: a tool for verifying real-time systems. user's guide and reference manual. Technical report, Verimag, Grenoble, France, 1992.