Hybrid Systems

Frits Vaandrager

1 Introduction

Hybrid systems are systems that intermix discrete and continuous components, typically computers that interact with the physical world. Due to the rapid development of processor and circuit technology, such systems are becoming more and more common in all application domains, ranging from avionics and process control to robotics and consumer electronics. The specification, design and analysis of hybrid systems require a synthesis of ideas, concepts, mathematical theories and tools that are currently spread over distinct disciplines, most notably Computer Science and Control Theory. The interest in the formal treatment of digital systems that interact with an analog environment is certainly not new, but received a new impetus by the extension of formal models from Computer Science with real-time around 1991 and by the explosion of embedded applications of computer technology within our society. Today, the study of hybrid systems has grown into an area of research with rapidly increasing popularity.

In this contribution, we will briefly sketch the development of this field. Also, we will discuss recent work on hybrid systems in the group Concurrency and Real-time Systems at CWI.

2 Motivation

With the decrease in the size and price of computing elements, more and more computers are used within real-world technical applications such as in avionics, process control, robotics, telecommunications and consumer products. In all these "embedded" applications it is software that determines to a large extent the functionality of the products and that offers the required dynamics and flexibility. This makes the construction of large real-time embedded computer systems one of the most challenging tasks facing the computer science community, if not the engineering community as a whole.

Characteristic of embedded computers systems is that they are *reactive*: they accept stimuli from the outside world and react to those stimuli. This means that one can *only* design and reason about the correctness of these systems if one takes the behavior of the outside world into account. As an example, consider a computer controlling a chemical plant. Regularly, the control program reads sensor data, such as temperature and pressure, of the plant. Based on these data, the computer may decide to turn on a heating system, switch off a pump, etc. When a dangerous situation arises, for instance the pressure in a tank gets too high, the computer has to initiate appropriate action, like opening a valve. In order to formally prove correctness properties like "No explosion will ever occur", the underlying mathematical model needs to contain information about the way in which the chemical reactions take place, the pressure in the tank evolves, etc.

Traditionally, computer scientists take a discrete view of the world: they assume that both a computer system and its environment can be modelled as a discrete event system. This assumption is justified and has proved to be very successful within application areas such as operating systems and computer networks. However, we see more and more applications, in particular in the area of embedded computer systems, in which modelling the environment as a discrete process greatly distorts reality and may lead to unreliable conclusions. For these applications, the continuous models for the real-world developed by physicists and control engineers are typically more compact, more tractable and more accurate than the discrete approximations computer scientists tend to come up with. Since designing and reasoning about software for reactive systems is only possible if one has a model for both the software and the environment in which it operates, this provides a real practical motivation for the development of a comprehensive theory of *hybrid systems*, i.e., systems consisting of a non-trivial mixture of discrete and continuous components.

The need for such a theory has also been identified by the control community. Inherently unstable applications such as flight by wire of an unstable aircraft cannot be controlled by a classical control system employing a single control rule. The only solution, which is the one being implemented today for such systems is a hybrid system, in which a digital controller keeps switching between different continuous control laws or control modes very rapidly.

The real-world is usually modeled by control engineers as a continuous dynamical system, described in the language of functional analysis. Computer scientists investigate the dynamics of discrete systems, described in the language of mathematical logic. Since the construction of large embedded computer systems will always involve representatives of both cultures, a comprehensive theory of hybrid systems is required to avoid a Tower of Babel. Such a theory must comprise a synthesis of ideas, concepts, mathematical theories and tools that are currently spread over several disciplines, most notably Computer Science and Control Theory, but also Electrical Engineering, Mechanical Engineering, Physics and others.

Even though hybrid applications have been built in practice since the beginning of the fifties, it is only since a few years that the mathematical study of hybrid systems is starting to receive the attention that it deserves. Until the end of the seventies, most research on program verification was devoted to the analysis of programs for "autistic" (i.e. nonreactive) sequential batch computers. The eighties have witnessed a revolution in the formal methodology for the specification, verification, and development of reactive programs (and more general reactive systems). Once the concept of a reactive system was well understood, it turned out not to be so difficult to bring quantitative, real-time aspects into the picture. Even though the development of reliable real-time systems will remain an important research topic for many years to come, several important theoretical results concerning real-time systems were obtained by 1991. Around that time, the first two important workshops on hybrid systems were being organized in Ithaca, NY, USA and Lyngby, Denmark, and triggered a rapidly increasing interest in this new area, both in computer science and in applied mathematics.

Research on hybrid systems is part of the general effort to apply formal methods in software development. Formal methods have been under development since the mid-1960s, but it is in the last decade that significant developments have evolved, and over the last few years interest in formal methods has grown phenomenally. Highly publicized accounts of the application of formal methods to a number of well-known systems, such as the Darlington Nuclear Facility and Airbus, have helped to bring the industrial application of formal methods to a wider audience. Today use of formal methods is required in the construction of software and digital hardware for certain critical systems. A recent paper by Gibbs [2] in Scientific American presents plenty of well-laid-out arguments and experiences of formal methods. For an overview of the field we refer to the World Wide Web page

`http://www.comlab.ox.ac.uk/archive/formal-methods.html`,

which contains numerous pointers to other electronic archives throughout the world.

3 Research on Hybrid Systems

3.1 Semantic Models and Logics

Several semantic models for hybrid systems have been proposed in the literature. Good starting point for reading are the Springer LNCS volume [3] and the recent Special Issue of Theoretical Computer Science [4]. Even though there are many differences between the approaches, there is a surprising consensus that the behavior of a hybrid system is best represented as an alternating sequence of instantaneous "discrete" actions and "continuous" phases during which time advances. Within the continuous phases the system's variables vary continuously according to a control law, fixed for the interval. Via the discrete actions the system can move to a new state, where a new control law becomes valid.

This view of hybrid systems is highly compatible with the earlier computer science semantic models for real-time systems: roughly speaking, the only difference is that where in real-time models the only available information about the continuous phases is the amount of time that passes, the hybrid models allow one to use differential equations, etc., to specify how precisely the system variables evolve in a continuous phase. Due to this similarity, many semantic models for timed systems generalize smoothly to hybrid systems. In several cases it is even possible to use verification techniques for timed systems directly for hybrid systems. Also in the area of specification logics many ideas generalize from timed to hybrid systems and logics such as TCTL, ICTL, TLA and the calculus of durations are being used in both settings.

3.2 Verification

However, there is a problem. Because the expressivity of the hybrid system models is enormous, the verification problem for these models is intrinsically difficult, even under severe restrictions. Typically, verification problems that are decidable in polynomial time for untimed systems (represented as finite transition systems), become exponentially hard for timed systems, and undecidable even for simple classes of hybrid systems. Much work remains to be done to identify restricted classes of hybrid systems that on the one hand are sufficiently expressive to model realistic applications in the area of embedded systems, but on the other hand can be analyzed by algorithmic means. A very promising line of research here is the work on (semi) decision procedures and tools for (subclasses of) the so-called *linear hybrid automata* of Alur et al. Two prototype tools have been developed, the KRONOS tool in Grenoble and the HYTECH tool at Cornell. Using these tools, their implementors have been able to verify automatically dozens of toy examples proposed in the literature on hybrid systems, as well as some practical examples from industry. The key idea behind these tools is to represent infinite sets of states as convex polyhedra in multi-dimensional space and to use standard data structures and geometric algorithms for polyhedral manipulation to do reachability analysis and model checking.

Despite the recent successes, computer aided verification techniques for linear hybrid automata will never become the solution to all industrial verification problems: they perform well in the case of small tricky circuits like the Philips audio control protocol analyzed at CWI, but are not designed to face the immense timing problems that arise in larger applications, such as control systems for sonar applications, air-traffic control, etc. Nevertheless, one can envisage two important uses of this work in a large-scale application. First, the knowledge about (linear) hybrid automata will be useful in a mathematical formulation of the problem. Second, there will be isolated situations where the tools themselves can be applied.

Clearly, the work on semantic models, logics and verification methods for hybrid systems is just starting. In the case of untimed discrete event systems, a rich body of closely related theories has been created during the last twenty years involving temporal and modal logics, assertional verification methods, process algebras, and tools for computer aided verification. Almost all of this work still needs to be lifted to (or at least related to) the setting of hybrid systems.

3.3 Perspective from Control Theory

Besides Computer Science also Control Theory plays an important role in the theory of hybrid systems, and thus we see for instance at CWI that both the Computer Science group Concurrency and Real-time Systems, and the Mathematics group System and Control Theory have become interested in hybrid systems, and benefit from each others expertise. It is interesting to note that the questions asked in Control Theory are quite different from those asked in Computer Science. Whereas in Computer Science there is much emphasis on verification, Control Theory concentrates more on synthesis: it aims at finding synthesis procedures for a supervisor that forces a discrete event system such that it satisfies prespecified control objectives. A well-known theory of "supervisory control" for untimed discrete event systems has been developed by Ramadge and Wonham. There are a few applications of this theory, but more experience must be gained and the model needs to be refined before it will become really useful in practice. Interesting approaches to the control of timed systems are proposed by Maler, Pnueli and Sifakis, and by Wong-Toi and Hoffman.

In traditional control system theories stability is an important performance criterion. Here stability means that for the controlled system, small changes in input and relevant parameters yields small changes in output. All theories of stability of continuous systems are topologically based. In the design of discrete event systems it is very difficult to come up with a similar notion of stability: in software engineering it is well-known that replacing a "," by a "." can have a dramatic impact on the behavior of almost any program. Kohn's theory of Declarative Control is an attempt to define a notion of stability for hybrid systems, using non-Hausdorff subtopologies of the usual topologies for continuous systems.

Networks of continuous devices have been studied by control engineers for a long time and some hybrid system models build directly on this long tradition. It is important to relate these models to the automaton based models proposed by computer scientists.

3.4 Specification and Implementation of Embedded Systems

Virtually all of the specification languages that are currently used by computer scientists to formally specify software systems have a discrete event semantics and are not directly suitable for the formal specification and analysis of hybrid systems. (Examples of such languages are VDM, Z, COLD-1 and LOTOS.) A possible exception is the language Funmath, a declarative formalism for describing systems with both analog and digital components, that has been developed in Nijmegen by Boute and his team.

If one leafes through a document with a state-of-the-art specification of an embedded software system, one encounters a mixture of architecture diagrams, programming text, flow charts, transition tables, diagrams by electrical engineers with IC's, transistors, resistors, etc., and diagrams made by mechanical engineers in which the mechanical parts of the system are described. The relations between the various parts of the specification are only described by informal text, there is no such thing as a common semantic framework for all the design notations that are used. This is undesirable, since in the design of embedded systems hardware and software are more and more viewed as interchangeable: often mechanical and electronic implementations of new and improved functions are replaced by software solutions (anyone can think of dozens of examples in the area of consumer electronics). The theory of hybrid systems aims at providing a semantic basis for a new generation of wide-spectrum formal specification languages in which *all* relevant elements in a design of an embedded software system can be described formally in an integrated way. In order to be accepted by software and control engineers in industry, these new languages should contain (close variants) of design notations that are currently used as sublanguages. The role of hybrid system theory will mainly be one of glue by which different design notations can be formally related.

It is well-know that the formalization step is a major source of errors in the design of critical software; therefore an important consideration in the design of a specification language is the readability of the expressions written in it. From this perspective, the work of on graphical specification languages for hybrid systems is quite important.

4 Work at CWI

Central to the approach of the Concurrency and Real-time Systems group at CWI is the I/O automata model of Lynch and Tuttle. Lynch and Vaandrager extended this model for reactive systems to the setting of real-time and hybrid systems. Below we will first briefly outline the main features of this extension and then discuss a practical application.

4.1 Timed Transition Systems

In the theory of reactive systems, a central role is played by the notion of a *transition system (TS)*. A TS consists of a set of *states*, a subset of *initial* states, a set of *(discrete) actions*, and a set of *(discrete) transitions*, which are triples

$$s \xrightarrow{a} s'$$

specifying that from state s the system can evolve to state s' by the instantaneous occurrence of the action a. A run of a TS starts in an initial state. The system jumps from state to state via instantaneous transitions, and in between these transitions, it can remain arbitrarily long in any state.

At the lowest level, timed and hybrid systems can be described by TS's with as an additional component a collection of *time transitions*, which are triples

$$s \xrightarrow{d} s'$$

specifying that from state s the system can evolve in a positive, real-valued amount of time d to state s'. In the model of *timed transition systems (TTS)* of Lynch and Vaandrager, two axioms are imposed on time transitions. The first axiom states that if there are time transitions $s \xrightarrow{d} s'$ and $s' \xrightarrow{d'} s''$, there exists a time transition $s \xrightarrow{d+d'} s''$. The second axiom, which is a bit more involved to state, postulates that for each time transition $s \xrightarrow{d} s'$ there exists a *trajectory*, a function w that specifies an intermediate state for each intermediate point in time, such that $w(0) = s$, $w(d) = s'$, and for all $t, t' \in [0, d]$ with $t < t'$,

$$w(t) \xrightarrow{t'-t} w(t').$$

Thus a trajectory describes *how* the system evolves from s to s'. A run of a TTS consists of a sequence of two-phase steps. The first phase of a step corresponds to a continuous state transformation described by a trajectory. In the second phase the state is submitted to a discrete change taking zero time described by a discrete transition.

We can add more structure to timed transition systems by defining states to be pairs $(\vec{x}, \vec{y})$ of a vector $\vec{x}$ of *discrete variables* and a vecor $\vec{y}$ of *continuous variables*. In a discrete transition both the discrete and continuous variables can be changed. However, in a time transition only the continuous variables may change. In the timed automata model of Alur and Dill, which is widely used for the description and analysis of real-time systems, all time transitions are of the form

$$(\vec{x}, \vec{y}) \xrightarrow{d} (\vec{x}, \vec{y} + d),$$

where $\vec{y}+d$ is the vector obtained by adding d to each of the variables in vector $\vec{y}$. In this model, the continuous variables behave as perfect logical *clocks*, whose values increase with the same rate as time. In the timed transition systems associated to the more general models of hybrid automata (see the paper of Alur et al. in [4]), the way in which the continuous variables change can be specified via differential equations. In Figure 1, an example is presented of a hybrid automaton modelling the temperature control system of a nuclear reactor. In [1], Bosscher, Polak and Vaandrager develop a language for the specification of linear hybrid automata, and define the semantics of this language via a translation to timed transition systems.

4.2 An Application

In [1], the theoretical work on linear hybrid systems has also been applied to solve a problem from Philips. This application will be briefly discussed below.

Fully fletched computer networks are standard features in today's consumer electronics, like the Philips 900 audio system. These networks make it possible for the different devices to talk to each other, and to offer a series of new, useful services to the consumer. A consumer can for instance wake up the whole system by touching a single button: there is no need to switch on the tuner first, then the CD player, then the amplifier, etc. Instead the system will do this job by broadcasting a "wake up" message over the network. The main technical difficulty in building the network for the Philips 900 audio system was that it had to be cheap: consumers are only willing to pay a tiny bit more for the additional services provided by the network. In fact, the only additional hardware that Philips needs to implement the network consists of a few transistors, resistors, etc., for the bus interface. The software runs on microprocessors that have to be present anyway. Because the clocks of these microprocessors drift, and because sometimes the programs dealing with the network have to wait for other programs that run on the same microprocessors, the network protocol has to deal with a significant uncertainty in the timing of events. In fact, Philips allows for a tolerance of 1/20 on all the timing.

At CWI, correctness was proved of part of the Easylink real-time protocol used by Philips to achieve reliable communication between the devices despite this very large timing uncertainty. The protocol is modeled as a linear hybrid automaton, with continuous variables to represent the drifting logical clocks of the sender and receiver in the protocol. Formally, the drifting is expressed by the requirement that the first derivative of the clock variables is in the interval $[1-\mathsf{T}, 1+\mathsf{T}]$, where T is the tolerance on the timing. Correctness of the protocol has been proved if the tolerance is less than $1/17$. This value is larger than the tolerance of $1/20$ that is allowed by Philips. A counterexample shows that the protocol fails for tolerances greater or equal to $1/17$.

In order to manage the complexity of real world applications, mechanical support is absolutely essential. Therefore, much of the research effort on hybrid systems is currently devoted to the development of mechanical tools that support specification and verification. At CWI, Griffioen has succeeded in mechanically checking the complete verification of the audio control protocol using the general purpose theorem proving tool LP. An impressive complementary result has recently been obtained by Ho and Wong-Toi from Cornell University. Based on the CWI modelling of the Philips protocol, they verified an instance fully automatically using the HyTech symbolic model checker, and also synthesized automatically the maximum clock drift of $1/17$. Independently, Daws and Yovine from Grenoble have also verified the protocol fully automatically using the Kronos tool.

References

1. D.J.B. Bosscher, I. Polak, and F.W. Vaandrager. Verification of an audio control protocol. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proceedings of the Third International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems,* Lübeck, Germany, September 1994, LNCS 863, pages 170–192. Springer-Verlag, 1994. Full version available as Report CS-R9445, CWI, Amsterdam, July 1994.
2. W.W. Gibbs. Software's chronic crisis. *Scientific American*, pages 72–81, September 1994.
3. R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems*, LNCS 736. Springer-Verlag, 1993.
4. A. Pnueli and J. Sifakis, editors. *Special Issue on Hybrid Systems of Theoretical Computer Science,* 138(1). Elsevier Science Publishers, 1995.

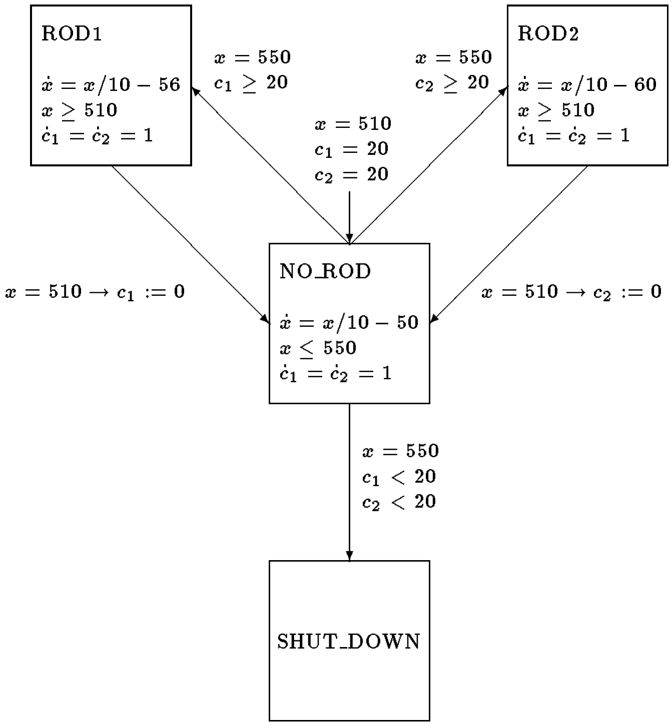

FIGURE 1. A hybrid automaton for a simple temperature control system. The system controls the coolant temperature in a reactor tank by moving two independent control rods. The temperature of the coolant is represented by the variable x. Initially x is 510 degrees, both rods are outside the reactor core and the system is in state NO_ROD. In this state the temperature rises according to the differential equation $\dot{x} = x/10 - 50$. If the temperature reaches 550 degrees, three things can happen: either rod 1 is put into the reactor core and the automaton moves to state ROD1, or rod 2 is put into the reactor core and the automaton moves to state ROD2, or a complete shutdown occurs and the automaton moves to state SHUT_DOWN. For mechanical reasons, a rod can only only be placed in the core if it has not been there for at least 20 seconds. Only if there is no rod available a shutdown will occur. In the automaton, variables c_1 and c_2 are used to measure the times elapsed since the last use of rod 1 and rod 2, respectively. The initial values of both c_1 and c_2 are set to 20 seconds, so that initially both rods are available. Since they represent perfect logical clocks, the first derivatives of c_1 and c_2 with respect to time are always equal to 1. Control rod 1 refrigerates the coolant according to the differential equation $\dot{x} = x/10 - 56$; control rod 2 has a stronger effect and refrigerates the coolant according to the differential equation $\dot{x} = x/10 - 60$. If the temperature has decreased to 510 degrees, the system moves back from state ROD1 or ROD2 to state NO_ROD, and variable c_1 or c_2, respectively, is reset to 0. The correctness property for this system, which is not difficult to prove, says that the SHUT_DOWN state cannot be reached. (This example is from Leveson et al.)