Chapter 1

Hybrid Systems

1.1 Introduction

The research on hybrid systems at Verimag has as a major objective to export some ideas and insights originating from computer science toward other domains of applied science and engineering that do not enjoy these insights and for which the computer is often viewed merely as a tool, rather than a source of inspiration for a new kind of mathematical models. This research direction is motivated by the conviction that concepts developed within various areas of computer science and engineering have a large potential contribution to the emergence of a modern *system design discipline* for complex and heterogeneous systems, embedded or not. The concepts that we have in mind include the distinction between syntax and semantics (in logics, programming languages and system description formalisms), relationships between models at different levels of abstraction, verification and controller synthesis using graph-theoretic techniques, algorithms and complexity, hierarchical and modular modeling and so forth. To be able to communicate these ideas to diverse communities, one must invest in understanding their languages, models (systems over continuous state variables and metric time), and performance criteria which, more often than not, are quantitative rather than qualitative ("non functional" in contemporary parlance).

The hybrid systems group at Verimag is one of the founders and world-wide leaders in hybrid systems, as witnessed by membership in the steering and program committee of the major hybrid systems conference [MP03], the foundation of new workshops on timed systems [AMY02] and on verification of analog circuits [DM05], the coordination of and participating in several top Europeans projects on the topic (VHS, CC, AMETIST), writing some introductory articles

[Mal01, Mal02, Mal04] and, of course, some important contributions to the domain in concepts, theory and tools. All these achievements are due to readiness to invest in understanding topics that cannot be solved by a small modification of one's own favorite concepts and tools, an investment that does not always pay in terms of immediate recognition by a close community (which is a major reason why so few computer scientists work on the topic).

1.2 Verification and Synthesis of Continuous and Hybrid Systems

The major innovative contribution of computer science to hybrid systems is the definition of the *hybrid automaton* model and the adaptation of verification techniques to such automata, where the hard part is, of course, the treatment of the continuous dynamics. After a short historical survey we explain the contribution of Verimag to this area during the reporting period.

1.2.1 Historical Survey

The hybrid automaton model was proposed in the early nineties in order extend verification toward systems that include both discrete and continuous components. The first papers on the topic advocated a deductive approach for verification but soon, influenced by decidability results for timed automata, the emphasis moved to algorithmic verification which means essentially to compute all the reachable states of a system, starting from an initial set and subject to an admissible set of inputs (disturbances). The early days of the domain focused on the so-called "linear" hybrid automata, in which the continuous variables have a *constant derivative* in every discrete state (mode). Like in timed automata, the set of time successors of a polyhedral set of states can be computed using linear algebra without resorting to the solution of real differential equations.[1]

However it turned out soon that except for some very special cases, the reachability problem for such automata is undecidable, a fact that, together with the expressive limitations on the continuous dynamics, led to a decay in the interest in this domain, except for some interesting theoretical results. From a tool development point of view, the outcome of this period include Kronos, by Sifakis and Yovine for timed automata and HyTech by Henzinger et al for "linear" hybrid automata. These tools and related papers have also established the standard form of reachability computation for timed and hybrid systems by alternating continuous

[1] This is part of the attractiveness of such automata for computer scientists.

dynamics and discrete transitions. Starting from a set P of states the procedure works as follows:

1. Compute the set of time successors of P according to continuous dynamics of the current mode
2. Intersect the result with the staying conditions of the mode to obtain all states reachable from P via transition-free trajectories
3. For each transition, intersect the set of time successors with the transition guard to obtain all "source" states of the transition
4. Compute the "target" set of the transition by applying the reset map to the set of source states
5. Restart the computation from the obtained set according to the continuous dynamics of the target mode.

It is important to mention that interest in hybrid systems was (and still is) very high in the control community, relatively much more than within computer science. This is partly due to the fact that many of the analytical methods used in control design are tailored for purely continuous systems and hence are restricted to idealized subsystems (for example, ignoring saturation or assuming linearity) and do not extend easily to the hybrid case. However, much of the early work from the academic control community was either on the fringe ("intelligent control") or more orthodox mathematical work devoid of significant computational content. Toward the end of the 90s, together with the reorganization of the hybrid systems community around the HSCC workshop, the focus moved toward reachability computation for "real" continuous systems, those that their continuous dynamics are defined by non-trivial differential equations. Of course, given all the undecidability results for simpler models, there is no hope for *exact* computation which, when you come to think of it, is a too strong requirement for physical continuous systems. In fact, the process of accompanying ideas from finite-state systems to systems with numeric variables involves an ongoing lowering of expectations, starting from exact computations which are guaranteed to terminate (timed automata), via exact computations that may not terminate ("linear" hybrid automata) to approximate computations (arbitrary differential equations). As we shall see later, the process does not stop there.

In the next section we describe briefly the techniques underlying **d/dt**, the major contribution of Verimag to reachability computation for hybrid systems. Although this is not intended to be a scholarly survey, we mention some preceding

relevant work such as that of Kurshan and McMillan on using reachability to construct finite-state abstractions of circuits from transistor models, related work by Greenstreet, various works in numerical analysis using *interval arithmetics* and the work of Kurzhanski et al on *ellipsoids*. Other tools that were developed in parallel with (and independently of) **d/dt** are CheckMate by Chutinan and Krogh at CMU, which used similar ideas, and Hysdel by Bemporad and Morari at ETH, which employs hybrid optimization to solve verification and synthesis problems for bounded horizon. Later tool development contributions to the domain include the work of Mitchell and Tomlin at Stanford on using PDE solvers for reachability computations and the Verishift tool of Botchkarev and Tripakis (then a post-doc at Berkeley) which uses ellipsoids.

1.2.2 Linear Systems and the d/dt Tool

The tool **d/dt** [ADM01a, ADM01b, ADM02] has been developed within the thesis of Thao Dang, 1996-2000 under the supervision of E. Asarin and O. Maler. It treats systems that are piecewise-linear or, more generally, piecewise-affine, that is, hybrid automata where the continuous dynamics in each mode q is defined by an equation of the form $\dot{x} = A_q x + B_q u$ where u is an external disturbance bounded inside some convex set. The basic problem is to compute an over-approximation of the time successors of a convex polyhedron P under the continuous dynamics. For constant derivative systems this can be achieved for all times by linear algebra, but in the general case we need to imitate numerical integration by fixing a time step r and computing a sequence of sets $\{P_i\}$ such that each P_i contains states reachable from P in the time interval $[(i-1)r, ir]$. Note that for discrete-time systems, P_i consists of points reachable from P *exactly* at time ir. This computation is done by numerical integration from the vertices of P, followed by a convex hull and additional modification to guarantee over-approximation. The approximation error does not accumulate and can be controlled by making r smaller. The sequence $\{P_i\}$ is stored either as a union of convex polyhedra or over-approximated by a non-convex orthogonal polyhedron for which we have developed a canonical representation. It is typically the intersection with transition guards that makes the shape of the reachable set more complex and requires additional vertices.

In addition to verification, **d/dt** also offers a procedure for synthesizing switching controllers for safety specifications and handles under-specified disturbances by performing an optimization procedure at every step for finding the input u that pushes each face of the polyhedron in "outermost" way. The tool is available publicly and its reachability algorithm has been integrated as the computational engine behind the Charon modeling and verification tool [ADE+01] developed at the uni-

versity of Pennsylvania. Due to the need to manipulate polyhedra whose dimensionality is that of the state space, the tool is currently limited to system of a modest dimension (around 5). In the sequel we describe extensions to **d/dt**, mostly due to the work of Thao Dang and collaborators, as well as some other investigations in hybrid reachability.

1.2.3 Extension to Non-Linear Systems

The extension of **d/dt** to deal with non-linear systems [ADG03], was one of the major developments during this period. This work was to a large extent a result of a successful collaboration between Verimag (E. Asarin., S. Yovine, T. Dang) with the applied mathematics MOSAIC team (J. Della Dora) within the IMAG Mash project [DMMRY01, DY01], culminating in the thesis of Antoine Girard [Gir04] . Underlying this extension is a new method of "hybridization", that is, transforming any reasonable non-linear system $\dot{x} = f(x)$ into a *piecewise-affine* system that approximates its behavior. This transformation is achieved by first triangulating the continuous state space into simplices, and then for each simplex, computing by interpolation the linear operator A that agrees with f on the vertices of that simplex. The maximal error between f and A is estimated and represented in the piecewise-affine dynamics as an additive disturbance. This way the system is modeled as a hybrid automaton where each state corresponds to a simplex and the transitions correspond to boundary crossings. Other good properties of the transformation include its continuity along the boundaries and its good error convergence. These results, which demonstrate also how hybrid system ideas can sometime breathe life into old disciplines like numerical analysis, have been integrated into **d/dt** and tested on several examples.

1.2.4 Treating Parameters

In the analysis of a continuous system $\dot{x} = f(x, u)$ we make a distinction between state variables x and input variables u. The former need to be stored at each step because they determine the future evolution of the system. Hence each of them increases the dimensionality of the stored polyhedra. Inputs, on the other hand, are not part of the state space and need not be stored because at every step they can "choose" a value taken from a bounded set, independently of their previous value. Parameters allow to define systems of the form $\dot{x} = f(x, u, p)$ where p is chosen at time zero and does not change during the lifetime of a trajectory. Hence parameters are like state variables in the sense that they need to be stored, but different from them in the sense that they do not undergo any dynamics. A specialized procedure

that treats parameters properly and hence reduces the dimensionality of matrix computation has been developed and integrated.

1.2.5 Differential Algebraic Equations

The dynamics of electrical circuits, a new application domain for hybrid systems, is typically described using *differential algebraic equations* whose numerical integration is a crucial part in circuit simulation. When those equations are of index one, their numerical solution can be performed as a combination of standard integration and projection on the constraint surface. This technique has been implemented and tested on small analog circuits [DDM04].

1.2.6 Experience with d/dt

Most of the few dozens of **d/dt** users are academic (we have some downloads from industry but without feedback). These users come from all over the world including France (INRIA, IRISA, Supelec), USA (Notre Dame, Vanderbilt, Penn, SRI, Honeywell), Germany (Oldenburg, Aachen, Karlsruhe, Hannover), as well as Netherland, Italy, Russia, China and Brazil. We currently have no resources to maintain an interactive user group, but we are aware that some use it for educational purposes. At Verimag we have applied **d/dt** to various case studies originating from diverse domains, including:

Automotive: Two case studies were treated by the predicate abstration tool build on top of **d/dt** within the DARPA-funded MoBies project. The first was an automative coordination system, with 4 continous variable and a dozen of modes which was proved to be collision free. The second was an electronic throttle control system, with 7 continous variables and 8 modes. Within the CC project **d/dt** has been used to synthesize an idle-speed controller on a model with 6 real variables and 3 modes [Dan05].

Computational biology: The non-linear extension of **d/dt** was used to analyze a model of the genetic regulatory network of Vibrio fisheri with 3 variables, non-linear dynamics and several modes [AD04].

Analog circuits: A model of a biquad low-pass filter with 3 continuous variables, and dynamics defined by differential algebraic equations with some non-linearity has been verified in [DDM04]. On the other hand, to illustrate the current limitations, we have not succeeded so far in verifying a sigma-delta A-to-D convertor with 3 state variables and two modes due to the frequent mode switching that may happen at every moment, and had to resort to bounded horizon techniques.

1.2.7 Abstraction of Hybrid Systems

Abstraction techniques fight state explosion by replacing a complex system S by a simpler system S' which over-approximates S in the sense that every behavior exhibited by S can be exhibited by S'. We have focused on two commonly-used approaches to abstraction, one is the reduction to a finite-state system by *partitioning* the state-space ("predicate abstraction"), and the other is by *projection*.

Predicate abstraction works by partitioning the continuous state space into regions, each of which is viewed as a discrete state of S'. Then the transition relation of S' consists of all pairs (R, R') of regions such that there exists a direct trajectory (a trajectory that does not visit any intermediate region) between some $x \in R$ to some $x' \in R'$. The computation of this relation using reachability techniques is much simpler than standard reachability because the complexity of the sets does not accumulate. In fact, one can avoid explicit reachability computation altogether by analyzing the vector field along the boundary between R and R' and establishing the existence of a boundary point in which it has a positive normal component.

However the artificial "transitivity" in the transition relation of S' may create many false counter examples, that is, abstract behaviors that do not correspond to any concrete trajectory. When such a counter example is encountered, one can refine the partition hoping to create eventually an abstraction fine enough, either for proving the property or for finding a counter example that can indeed be concretized. This methodology, popular for verification of software and infinite-state systems in general, has a lot of heuristic ingredients such as the choice of the initial partition (based sometimes on the properties to be proved) and its refinement procedure. In a series of works [ADI02, ADI03a, ADI03b, ADI04, ADI05] the applicability of counter-examples based abstraction refinement for hybrid systems has been explored.

The other natural abstraction technique is to reduce the state-space of the system by projecting away certain variables. A variable thus projected moves from the category of state variables to that of under-specified inputs, resulting in a more nondeterministic system. In other words, we over-approximate a system of (nonlinear) differential equations by a hybrid system with differential *inclusions* in lower dimension. The technique developed in [AD04] is based on this principle, augmented with ideas coming from qualitative physics and with a method for error control. While this abstraction approach can be used for dimension reduction, its effective application requires the ability to deal with nonlinear differential inclusions. We developed a reachability analysis method for uncertain bilinear systems which, in combination with the abstraction by projection, allows us to treat multi-affine systems. An experimental implementation of this technique enabled us to study

a bacteria model in bioregulator networks, but more experimentation is needed to assess its scope and limits.

1.2.8 Analysis of Polygonal Hybrid Systems

In parallel with approximate computations, the limits of exact reachability have been explored. Polygonal hybrid systems are a subclass of planar hybrid automata which can be represented by piecewise-constant differential inclusions. We have studied qualitative properties of such systems and established several decidability and undecidability results within the thesis of Gerardo Schneider (supervised by S. Yovine and E. Asarin).

In [ASY01], we have developed an algorithm for solving the reachability problem. Our procedure is not based on the computation of the reach-set but rather on the computation of the limit of individual trajectories. A key idea is the use of one-dimensional affine Poincaré maps for which we can easily compute the fixpoints. We have shown that between any two points linked by an arbitrary trajectory there always exists a trajectory without self-crossings. Thus, solving the reachability problem requires considering only those. We proved that, indeed, there are only finitely many qualitative types of those trajectories. For each type, we constructed a decision procedure based on the analysis of the limits of extreme trajectories.

In [ASY02] we have studied phase portraits. We analyzed the qualitative behavior of sets of trajectories having the same cyclic pattern. Using the classification of cyclic behaviors of [ASY01], we have given a classification of cyclic behaviors, we have shown how to compute the *viability* kernel of a cycle, that is, the set of points which can keep rotating in the cycle forever. We have shown that this kernel is a non-convex polygon and give a non-iterative algorithm for computing the coordinates of its vertices and edges. Furthermore, have defined and computed the *controllability* kernel, a cyclic polygonal stripe within which a trajectory can reach any point from any point, analog of the notion of limit cycle. Indeed, we have proven that the distance between any infinite trajectory performing forever the same cyclic pattern and the controllability kernel always converges to zero. These results are the analog of Poincaré-Bendixson for polygonal differential inclusions. A tool for qualitative analysis of such systems and based on these results has been developed by Gordon Pace [APSY02].

In [AS02] we have revisited the decidability of the reachability problem for hybrid systems of a low dimension. We have characterized several classes of two-dimensional hybrid systems for which the problem is undecidable, and found other classes for which reachability is as hard as reachability for piece-wise affine maps, which is a very well known open problem.

1.2.9 Symbolic Reachability Computation

In [LPY01] we have presented the first known families of linear differential equations with a decidable reachability problem. This is achieved by posing the reachability computation as a quantifier elimination problem in the decidable theory of the reals. We have characterized families of linear control systems of the form $\dot{x} = Ax + Bu$, where u belongs to a set U of possible inputs, for which the set of reachable states is definable in the theory of real numbers. Indeed, we have identified fragments of the real field extended with exponential and trigonometric functions that admit quantifier elimination by applying an appropriate change of variables. A quantifier-free representation of the reachable sets can then be obtained by quantifier elimination using tools such as REDLOG and QEPCAD. This result is the basis of the REQUIEM tool later developed at the University of Pennsylvania.

1.2.10 Perspectives

Computing trajectory tubes for continuous and hybrid systems subject to disturbances and parameter variations is a novel contribution to control and to the analysis of dynamical systems. It can replace analytic methods when those fail, and complement simulation for small critical subsystems. These techniques suffer, however, from severe *state explosion* that limits their applicability. Like discrete formal verification, they are not easy to explain to engineers who are accustomed to reasoning about *individual trajectories*. Our current thinking is that progress should be made along two complementary directions, algorithmic improvements and further relaxations of completeness criteria.

The first direction is to improve the performance and usability aspects of **d/dt** and change its nature from a proof-of-concept to a more mature tool. A lot can be done in improving the "main loop" of reachability analysis by using alternative geometric objects and data structures (we are currently exploring the applicability of *zonotopes*, a special class of polyhedra, whose usefulness for reachability computation has been recently demonstrated by Antoine Girard), employing variable-step time discretization which can be coarser in uninteresting parts of the state space, using more sophisticated froms of abstraction or by exploiting analytic information extracted from the system dynamics.

Such an effort should aim at increasing the capabilities of the tool by one order of magnitude, say, few dozens of modes and few dozens of state variables. Such an improvement together with a more flexible user interface will increase the user base of **d/dt** and provide the necessary feedback for future developments and eventual

industrial transfer. To achieve this ambitious goal an *additional researcher* and an *engineer* are badly needed.

The second direction consists in seeking alternatives to the "classical" approach for analyzing hybrid (and timed) systems by exhaustive computation of reachable sets of states, and develop methods based on adaptive search in the space of (simulated) trajectories. Such methods can be more intuitive for the engineer and will offer different tradeoffs between computational complexity and coverage, much in the spirit of testing for discrete system. These ideas will be discussed in the next Section.

1.3 A Unified Approach to Controller Synthesis

Working with discrete and continuous systems, being exposed to conrol, verification, scheduling and other domains, one cannot but observe that many problems treated under different names within different disciplines, have more resemblence if we look at them through an appropriate abstraction that filters their domain-specific details. Among these problems and techniques we mention the algorithmic approach to discrete systems verification by forward or backward fixpoint computation, the derived reachability algorithms for continuous and hybrid systems, bounded model checking (using satisfiability solvers to verify correctness for a bounded horizon), computational techniques for optimal control such as dynamic programming and model-predictive control, simulation, search methods in AI and Markov decision processes. Much of our effort during the reporting period was concerned with building a general unifying game-theoretic scheme, for which various system design and validation problems are concrete instances. We have also invested a significant effort in attacking several concrete instances of this scheme, most notably scheduling under uncertainty and hybrid optimal control. We have also investigated controller synthesis for distributed systems and some practical aspects related to

1.3.1 The Game-Theoretic Approach to Design

The paper [Mal04] lays down a *unified* and *domain-independent* model for control in the presence of adversaries. The model uses the metaphor of a two-player game between the controller to be synthesized and the environment it is supposed to control. Control synthesis is viewed as finding an optimal (or satisfactory) strategy for the controller, where optimality is parameterized by two factors:

1. The cost function associated with *individual* trajectories

2. The way the costs of all possible adversary-induced trajectories under a given strategy are combined to compute the overall value of the strategy (worst case, average case).

Verification, and open-loop control ("ballistic" control, planning) are obtained as special cases where either one of the players is suppressed, i.e. is assumed to be deterministic with no choice. On this model we identify three generic approaches for solving controller synthesis problems:

- For *bounded timed horizon* the problem can be posed as a constrained optimization problem as is done in model-predictive control and bounded model-checking (in the latter, it is often the case that *feasibility* of the constraints is emphasized rather than cost optimality).

- The other class of methods, roughly characterized as *dynamic programming* (DP), is based on iterative computation of a value function (cost-to-go), which determines the optimal cost and action at every state of the system. For discrete systems such techniques are used in backward verification and synthesis for automata and for Markov decision processes. In continuous systems the value function is often computed as a solutions of some partial differential equations due to Hamilton, Jacobi, Bellman and Isaacs (HJBI).

- The third approach is to perform a forward search in the space of strategies and trajectories, an approach used by game-playing programs. In verification, when the search is not exhaustive over all adversarial behaviors, this activity can be viewed as testing. For control this approach is not mainstream but it seems to be popular in some related domains such as reinforcement learning and robot motion planning. The advantage of this approach compared to DP is that the value function needs to be computed only for the reachable part of the state space, a fact that may moderate the curse of dimensionality.

The separation between the domain-independent abstract scheme and the specific computational aspects of each domain, may facilitate the development of a "universal" controller synthesis tool based on this model. The specifity of each domain will be manifested by the type of computational engine used, for example a SAT solver for discrete systems and an LP solver when the dynamics is linear. The study of solvers for hybrid constrained optimization problems, the basic computational tool for all verification and synthesis problems, is becoming an important research issue by itself and our contribution to the domain is described in Section 1.6.1. We

started exploring the applicability of the game-theoretic model and its variants to several domains as described in the sequel.

1.3.2 Scheduling under Uncertainty

One important instance of the game-theoretic scheme is the problem of scheduling under uncertainty. Here the controller is a scheduler which decides at certain points in time whether to allocate resources to enabled tasks, and the adversary is used to model various types of uncontrolled uncertainty such as in task arrivals, duartions, outcomes or faults. Thanks to the state-space modeling framework for scheduling using timed automata, developed within the the thesis of Yasmina Abdeddaïm (1999-2002) [Abd02], we can pose such problems very naturally as finding a strategy for a timed game automaton and solve them rather efficiently (recall that all the problems we deal with are NP-hard at best). All this work had been done within the European projects VHS (Verification of Hybrid Systems) and Ametist (Advanced Methods for Timed Systems).

In [AAM05] the problem of scheduling under temporal uncertainty, where task durations are bounded within an interval, has been addressed. Since in this problem worst-case optimality is somewhat trivialized because the worst-case optimum can be obtained by a *static* scheduler that assumes a maximal duartion for each task, a more flexible criterion called d*-future optimality* has been defined, and strategies optimal with respect to this criterion have been computed using dynamic programming on timed automata. On randomly-generated instances the average performance of schedules thus obtained was within 1.5% from the clairvoyant solution while static schedules deviated from it by more than 12.5%. This result can serve as a basis for a new approach for handling "soft" real-time systems.

Note that this problem is a special case of the more general controller synthesis problem for timed automata, for which an algorithm which uses zones but works on-the-fly in a forward manner, has been developed and implemented in [AT02].

In [BKM04] we tackled the problem of optimal scheduling under discrete uncertainty. Tasks are related by precedence constraints as in standard scheduling problems as well as by *conditional dependencies*, where the outcome of one task may determine whether another task should be executed. This is a natural model for scheduling programs with *if-then-else* branches on parallel machines, and can be very useful for high-level synthesis. The whole situation is modeled as a timed game automaton where the adversary chooses the task outcomes, that is, values for certain Boolean variable that may appear in the activation condition of further tasks. The problem is then reduced to a shortest path problem in discrete weighted *game graphs* and solutions are found using a variant of depth-first min-max search.

We are currently comparing various heuristics for more efficient exploration of this game graph. For the moment we can easily find optimal solutions for problems with 20 tasks and few conditions, and sub-optimal (but good) solutions for problems with hundreds of tasks and up to 10 conditions. This work is done within the thesis of Abdelkarim Kerbaa, supervised by O. Maler and M. Bozga.

Note that in this work, due to the discreteness of the adversary, we can easily apply forward search algorithms, while for the previously-mentioned problem of duration uncertainty, the adversary is dense and the solution is found by the much more costly dynamic programming procedure. We are currently exploring a discretization approach for the duration uncertainty problem by assuming only finitely many possible durations for each task. Although using this approach the system may reach states for which a strategy has not been computed, we believe that using approximate strategies, a significant increase in performance can be acheieved. Not covering all adversary choices while computing strategies is also an interesting option for more general continuous and hybrid systems as discussed in the next section.

The behavior of real-time systems with preemptive schedulers can be modelled by *stopwatch automata*. Nevertheles, the expressive power of stopwatch automata discouraged for a long time their use for verification purposes. Indeed , the reachability problem (even for a single stopwatch) has been proven to be undecidable. There are, however, some decidable sub-classes such as the so-called *integration graphs*, *suspension automata* and *timed automata with tasks*. In [Zan04] we have studied sytems with preemptive scheduling (with static, e.g. RMA, or dynamic, e.g. EDF, priorities), uncertainty (lower and upper bounded execution times), and precedence dependencies. The behavior of these systems cannot be straightforwardly translated into a decidable extension of timed automata. We showed that the reach-set can be represented by formulas involving difference constraints on clocks, and time-invariant equalities capturing the values of stopped clocks. This result implies decidability and leads to an efficient implementation. Moreover, it gives a precise symbolic characterization of the state space for the considered class of systems.

In [MKM02] we developed an abstract model of the problem of control under limited computational resources, where the controller had to keep the performance measures of a multi-plant system bounded by mode switching, where each mode represents a different mix of attention and resources, and hence a different derivative for the performance vector. We show how controllability for safety can be reduced to some reachability problem on “linear” hybrid automata whose decidability status is still open.

1.3.3 Search-based Verification and Control of Continuous Systems

For a system $\dot{x} = f(x, u)$, when $u \in U$ is interpreted as an external disturbance, exahustive verification requires computation with subsets of $\mathbb{R}^n$ with all the associated difficulties. An alternative approach would be to use discrete time (which is equivalent to restricting the input space to piecewise-constant signals) and discretize U into a finite set $\bar{U}$. By doing so we reduce the input space into the familiar and tree-like object $\bar{U}^*$ that we all love. If the discretization is sufficiently dense, correctness with respect to $\bar{U}^*$ is a good enough approximation for exhaustive correctness. Our first work in this direction was [KMSK03] and can be viewed as an alternative to "classical" reachability where instead of storing all points reachable within a time interval in a single geometric object, neighborhoods of sample trajectories are stored and merged. This approach allows one to export many graph-searching ideas from discrete to continuous systems.

When u is interpreted as a control input, this approach becomes even more interesting because in synthesis we are sometimes happy to find *one* control sequences which is good enough and do not always care about being exhaustive or optimal. We started exploring this idea in the control context in [Ali03] and demonstrated its applicability on a toy problem of guiding a missile inside a tunnel. Is is clear, however, that search alone, no matter how intelligent, will cause an explosion which, in this approach, is more related to the number of input sequence ($|\bar{U}|^k$ where k is the decision horizon) rather than to the dimensionality of the state space. Further progress requires some exploitation of knowledge of the system dynamics and combination with more traditional techniques for optimal control.

As a test case for search-based controller synthesis we looked at the ABB case study of the CC project. This is a small-scale model of a power ditribution network where the goal is to restabilize the system after a failure at time zero. The controller has 3 different types of actions (tap ratio, compensating capacitance and load shedding) to influence the system, each with its cost and effectiveness. A naive application of this idea would involve searching over all sequences ranging over all the 60 combination of the control variables values, however the monotone effect of each of them on the dynamics, together with the ordering of their costs, allowed us to use a segregated search that tries the cheap controls first. The resulting algorithm, reported in [SD05], is a variant of model-predictive control where the optimization for the bounded horizon is performed as a search over a finite set. The algorithm was tested on various scenarios in which the impedance of the system jumps from 0.25 p.u. to a higher value. When this value was 0.60 or smaller we could quickly find stabilizing sequences. The quality of the solutions found by segregated search was as good as exhaustive search for the decision horizons for

which the latter could be applied.

The current effort in this direction, conducted within the thesis of Alexandre Donzé (supervised by O. Maler and T. Dang), is to develop more general-purpose optimal control algorithms that are based on search, without sacrifising rigorous analytic properties. Recently a new variant of the temporal-difference algorithm has been developed [Don05] and performed well on a toy inverted pendulum problem. The next problems that we intend to tackle are those of finding optimal paths for mobile robots amidst obstacles, and stabilizing a double pendulum.

1.3.4 Synthesizing Low-Complexity Schedulers

In order to synthesize a scheduler, the basic idea consists in constructing the set of reachable states and, thus, identify the deadlocks. These are the states where the application threads are deadlocked, or the states where some thread has missed its deadline or period. Having obtained the deadlocked states, we do a backwards traversal of the whole state space starting from the deadlocked states, and eliminating transitions leading to them, until there are no more deadlocks. Even though the basic idea is simple, it is evident that in practice it suffers from the state explosion problem. Therefore, it is imperative that we use techniques to minimize the size of the state space.

In [KY03] we have developed a method for systems composed of cooperating threads scheduled on a preemptive real-time operating system running on a monoprocessor hardware architecture. Our method consists of synthesizing schedulers for successively more detailed models, adding thus complexity to a model only when we have already calculated how we can constrain the more abstract one. The scheduler synthesis is performed in five major steps:

- Compositional Synthesis: first, we decompose the system and synthesize constraints independently for each of the components.We then apply the synthesis algorithm again on the parallel composition of the already constrained models.

- Abstraction of Time: second, we consider the issue of time. We examine the untimed model of the system. Searching for deadlocks in the untimed model allows us to examine a much smaller search space. In order to make the problem more tractable, we then reduce the timed model modulo the branching bisimulation equivalence, which eliminates unobservable actions (in our case time-passing transitions) but only when doing so preserves the branching structure of processes.

- Execution Model: third, we analyze the behavior of the system for two different execution models, namely preemptive and non-preemptive. We first consider that the application threads cannot be preempted while they are computing. The non-preemptive execution model hypothesis reduces the state space, since it removes all the cases where the execution of a thread is suspended so as to handle an interrupt. Once we can indeed safely schedule the system under the hypothesis that threads are never preempted, then we can use the constraints obtained during this step to reduce even further the state space that we have to construct and analyze when we do allow threads to be preempted.

- Observability of Clocks: the constraints we produce during the synthesis use the state of the system to decide what are the safe choices at each point during the execution and, therefore, also make reference to the values of the local clocks of the threads. However, these clocks do not really exist in the application but are introduced as a way to model computations. As using timers may substantially increase the execution time of the scheduler, we investigate the possibility of synthesizing a clock-free one. Not observing clock values, actually defines an equivalence relation that further reduces the state-space.

- Policies: once we have synthesized a scheduler, which is indeed non-deterministic, we can compose it with policies to reduce non-determinism (or even to make the scheduler deterministic).

This methodology has been successfully used in [KNY03] for generating schedulers for real-time Java applications.

In [Klo04] we have shown how one can adapt data-mining techniques to decrease the size of a synthesised scheduler to allow optimizing its implementation.

1.4 Decentralized Observation and Control

This research action aims at studying problems of observation and control in a distributed setting: instead of a single observer or controller, there are many such agents which monitor or control a plant at the same time. Each of them has partial information about the plant, and this is what makes the problem difficult. The problem is worth studying, however, since it embodies many interesting applications, including the long-pursued goal of protocol synthesis [PTV01].

The two basic properties of most centralized observation and control settings[2], namely, (1) that existence of an observer-controller implies existence of a finite-state observer-controller, and (2) that checking existence and synthesizing an observer-controller is undecidable, break down in many decentralized settings. The precise features of decentralization which result in loss of such properties as decidability are still only partially understood.

Our work has contributed in the understanding of the computability limits in decentralized observation and control. In [Tri01, **?**] we have shown that undecidability occurs even in very basic decentralized observation and control problem on regular star-languages. This work also revealed how a decentralized observation problem where decisions are made off-line at a central decision point (whereas observations are gathered on-line by the decentralized observers) can be reduced to a decentralized control problem with no direct communication among controllers: the reduction is possible because communication can be modeled indirectly using an appropriate plant. In [Tri02, Tri04a] we have studied the problems where communication among observers-controllers is built in the setting: we have shown that bounded-delay communication makes the problem decidable, although unbounded-delay communication does not help in this direction. Finally, in [Tri05] we have established links of the above decentralized observation problem to the theories of traces and rational relations. These links have permitted to identify a number of special cases where the problem is decidable.

1.5 Specification Formalisms for Hybrid Behaviors

The added value of "formal methods" to software and hardware engineering is not restricted to verification procedures. A large part of its contribution to system design is in putting *proprties* at the center stage of the validation process. Properties are syntactic objects that specify in a rigorous manner which traces of I/O behaviors the system may exhibit while interacting with its external environment. Properties are typically expressed in some "declarative" formalism such as predicate logic, temporal logic or regular expressions over the observable alphabet.

The validation of a system with respect to a given property is based on transforming the property into a *property monitor* (observer, tester), a mechanism that checks whether a given behavior (sequence of I/O events) satisfies the property. This monitor can be viewed either as an automaton accepting exactly the set of satisfying behaviors or as a procedure working recursively both on the length of

[2]Including partial-observation settings

the sequence and on the syntactic structure of the property.[3]

The theoretical aspects of such transformation are related to the theories of formal languages and logic that deal with the expressive power of various formalisms for specifying discrete behaviors. One of the remarkable of this classic theory is that regular (finite-state) languages admit a variety of logical, algebraic and operational description formalisms which are all equivalent. We want to extend such results for timed and hybrid systems, partly for the purpose of exporting the language-theoretic aspects of formal methods to the continuous domain, and partly due to interest in fundamental theoretical problems related to timed and hybrid formal languages. Much of this work is carried out within the European project PROSYD (Property-based System Design) and some of its aspects are related to the work on testing timed automata mentioned in Section **??**.

1.5.1 Timed Regular Expressions

The result in [ACM02] is one of the strongest theoretical results concerning timed formal languages. It starts by defining decent semantic domains for timed behaviors (the monoid of *time-event sequences* and the monoid of discrete-valued *signals*) and then goes on to define timed regular expressions as means for specifying sets of such timed behavior. This class of expressions extends untimed expression by adding a time restriction operator, and by using conjunction and renaming (both are provably necessary to match the power of Alur-Dill timed automata). The translation from these expressions to timed automata is simple and the hard part is the proof of the other direction from automata to expressions, which involves the solution of a new type of language equations. This Kleene theorem for timed automata is part of our effort to elevate the theory of timed languages and automata to the level of the classical untimed theory. In another recent work in this direction [MP04] we define a notion of recognizability for timed languages and show that it coincides with deterministic timed automata. Mininization and determinization of timed automata have been studied also in [Tri04b] and other algebraic aspects of the theory of timed languages and expressions were explored in [Dim01b, Dim01a, Dim02, AD02].

1.5.2 Temporal Logic for Continuous Signals

The PROSYD project is concerned with the *property specification Language* (PSL), accepted as a standard by the semi-conductor industry. This is essentially Pnueli's

[3] It is worth mentioning that some researchers and tool vendors advocate the direct use of monitoring procedure without using an external specification formalism.

temporal logic augmented with syntactic sugar, regular expressions and additional features which are needed in order to move from a clean theory to industrial-strength tools. Our role in this project is to develop an extension of PSL to describe properties of analog and mixed signals (AMS). This work is related to a new promising application domain, verification of analog circuits, to be discussed in Section **??**, which examplifies the inter-cultural gap between computer science and more "physical" domains.

As a starting point [MN04] we took a bounded version of the real-time logic MITL, and augmented it with static predicates on the continuous domain that transfom analog signals into Boolean ones. Using the obtained logic STL (Signal Temporal Logic) one can express "sequential" properties that specify the order of occurrence of certain events (threshold crossings of continuous signals) as well as the temporal distance between them. These properties are quite different from the "properties" currently and implicitly employed in analog design which are more of a macroscopic nature (frequency, signal-to-noise ratio, energy). So far we have demonstrated how control-oriented properties such as stabilization and tracking can be expressed in STL and are currently working on properties related to flash memories, as communicated to us by the STM partners in PROSYD.

The development of future versions of STL depends on an ongoing interaction with the potential users of the technology including partners in PROSYD and other collaborators including the group at CMU with which we have recently collaborated on defining and checking oscillation properties [FKRM05]. Among the extension to STL that we coinsider we mention frequency-domain properties as well as property that allow continuous values at different times to be related directly and not only through their Boolean abstractions, for example something in the spirit of $t' - t < a$ implies $\xi[t] - \xi[t'] < b$. Other "properties" to be considered are those that output numerical values that reflect some quantitative characteristics of the signal. Other properties such as following a reference signal, require further investigations in new metrics for signals. This work on STL and its monitoring is part of the thesis of Dejan Nickovic supervised by O. Maler.

1.5.3 Application: Property Monitors

Large systems with some hundreds of thoushands of transistors will have to wait for quite some time until exhaustive hybrid verification techniques will be able to handle them. The semi-formal alternative is to use montiors in conjunction with numerical simulators so that the monitors observes simulation traces, detects property violations and liberates the user from the need to construct monitors by hand or to inspect long waveforms or large files manually. We have built an offline monitor-

ing procedure for STL [MN04] which takes a sampled-time simulation trace, and marks the truth values of sub-formulae backwards until the truth value of the formula itself is determined for time zero. The monitor has been applied to simulation traces generated by a Simulink model of a CC project case study, the water-level controller by EDF.

The disadvantage of offline monitoring is that it can start only after the simulation terminates, while it is often the case that a property is already satisfied or violated by a short prefix of a long trace, a fact that an online monitor can detect. This can be very meaningful when simulation is costly as is the case with certain analog circuits: it may take few weeks to simulate one behavior of an RF circuit. Another advantage of online monitors is that they can be used to monitor *real* systems during execution and not only simulated systems. Online monitoring requires, however, deterministic (or determinizable) monitor, which is a problem because MITL can express properties that are not acceptable by a deterministic timed automaton. We mention several possible remedies to this problem:

1. Use the observation of Tripakis that on-the-fly subset construction with respect to a *given* trace is possible, an observation used for testing timed automata [KT04].

2. Use specification formalisms that admit deterministic timed automata, for example the *past* fragment of MITL [MNP05].

3. Develop (as we did) a piecewsie-backward procedure that cuts a signal ξ into finitely many pieces $\xi_1, \ldots \xi_n$ and monitors them, one after the other, in a backward fashion.

1.6 Fighting the Clock Explosion

Timed automata can be viewed either as a special class of hybrid automata with a very simple dynamics in each mode (all clocks variables have the same derivative) and a restricted reset map, or as an extension of automata where the dynmaics takes into account not only the qualitative input history but also the time elapsed since certain events have occurred. This requires as many clock variable as there are events whose occurrence times need to be simultaneously memorized.

The basic verification questions for timed automata are decidable and are solved by symbolic reachability algorithms like the one sketched in Section **??**, specialized to the dynamics of clocks. The decidability comes from the fact that there are finitely many "timed polyhedra" encounterd during the reachability computation,

but from a complexity point of view this does not help much. The analysis of a system consisting of n components each with m states and one clock may generate as much as $m^n k^n n!$ zones, where k is the largest constant appearing in transition guards. The representation size of each zone can be n^2 which does not make life easier.

Although a lot of effort has been invested during the last decade in finding more efficient ways to represent and manipulate the reachable states of timed automata, to the best of our knowledge, this problem has not been solved and very few systems with more than a dozen of clocks has ever been verified using timed automata, which is a pity given their extremely useful expressive power. By analogy, the current status of timed verification is like that of discrete verification before the BDD revolution. One of the major problems seems to be the lack of an effective symbolic representation for both the discrete and continuous parts. Since we are interested in large-scale application like circuit timing analysis and scheduling, we have invested quite a lot in various attempts to solve the problem. In the past we have tried BDD representation (binary-encoded discrete clocks) and a canonical representation of non-convex timed polyhedra, while more recently we have focused on the methods described below.

1.6.1 SAT Solvers for Difference Logic

Bounded model-checking (BMC) for discrete systems profited from the impressive progress in the performance of propositional SAT solvers. Systems that made reachability computation explode, can now be treated in a satisfactory manner using such solvers. In a series of works (in collaboration with P. Niebert and E. Asarin) we have tried to apply these ideas to timed automata. We have first identified *difference logic*, propositional logic augmented with difference constraints of the form $x - y < d$, as the appropriate logic for expressing the transition relation of timed automata as well as other timing-related problems such as scheduling.

Our first experience with the art (or more appropriately, the black magic) of SAT solvers was through the MX-Solver developed within the thesis of Moez Mahfoudh [NMA+02], during which we have studied different approaches for enhancing SAT to treat enriched logics. Our approach was based on a dynamic interaction between the Boolean and numerical parts. Each time a difference constraint was implied by the current assignment (i.e. became a unit clauses) it was put in a large difference-bounds matrix (DBM) which was checked for consistency using algorithms for negative cycle detection. Note that the DBMs used for SAT are several orders of magnitude larger that those used in TA verification. The performance of this solver on problems coming from BMC for timed automata was still much

inferior than that of standard TA verification tools. It was much superior, however, to several other solvers developed around the same time (the topic was in the air) on problems dominated by difference constraints such as job-shop scheduling.

The second round of work on the topic started with the masters thesis of Scott Cotton.[4] The first version of his DLSAT solver [CAMN04] employed conflict analysis and learning as well as a variety of techniques that improved the performance results significantly. The second (and still unreported) version of DLSAT has already obtained some impressive results. To begin with, for purely-propostional formulae it beats a world-class solver such as zChaff on some instances. Secondly, for some notorious job-shop scheduling problems it found the optimum (!) while most solvers would explode much before that. Finally the results on bounded model-checking for TA are much more encouraging than the previous results (for example is is able to check a formula obtained from a 25-unfolding of a 64-gate circuit (2^{64} states and 65 clocks) but more work still needs to be done concerning more efficient translations of BMC to SAT. It might be the case that like other asynchronous systems, TA are not amenable for efficient BMC. Like in untimed systems, the use of the solver is not restricted to BMC and, it can serve as an alternative computation engine inside (unbounded) reachability algorithms.

1.6.2 Abstraction Techniques for Timed Systems

Another attack on the clock explosion problem is conducted within the project *Combining Formal Verification and Timing Analysis* sponsored by Intel. The goal is to use the expressive power of timed automata to verify circuits at the level of abstraction of Boolean functions plus bi-bounded delays. Although we have managed in the past to verify some non-trivial asynchronous circuits in this level of abstraction [BJMY02], the size of most circuits is orders of magnitude larger than what TA tools can handle. To tackle this problem we resort to divide-and-conquer methods that cut circuits into small enough pieces, analyze each piece separately to construct a small abstract model of its I/O behavior for further use.

In [SBM03] we have applied this idea to combinational circuits whose inputs change only at time zero and hence their automata are acyclic and every run reaches a stable state within a finite amount of time. We add an additional clock which is never reset to zero and hence it measures absolute time. We analyze the circuit using reachability analysis (which eliminates qualitative behaviors which are impossible due to timing constraints) and then, after having used the clocks we throw them away by projecting guards on the absolute clock, hide transitions in internal

[4] In Max-Planck Institute after not having been admitted scandalously to the mediocre doctoral school of UJF.

gates and minimize the automaton. This way we obtain an over-approximation of the circuit which is faithful to its qualitative behavior and to the absolute occurrence times of events but is more liberal with respect to their temporal distances. Using this compositinal approach we could analyze circuits with almost 100 gates, a great step for timed automata, but admittedly, still a far cry from the size of circuits analyzed by traditional methods.

Targetting more sophisticated asynchronous circuits, we have extended this technique to deal with open reactive systems whose inputs may change *any* time. This is a necessary step toward "hierarchical" compositional reasoning. Here there is no absolute clock to projet on because we need to memorize input events that happen at different times. Our solution is to use dynamic clocks generated at every input event. Since we deal with well-formed circuits without auto-generated oscillations, every input event is propagated within finite time to the output, its occurrence time can be forgotten and its clock can be reused, a fact that guarantees a finite number of clocks. Applying the abstraction technique, this time projecting on the input clocks, we obtain an abstract model that relates the occuurence times of input and output events. This work, performed within the thesis of Ramzi Ben Salah (supervised by O. Maler and M. Bozga), involves a lot of implementation effort and is currently in its final phase [SBM05]. It could serve as a method for creating timed abstraction of components in general, not necessarily those made of hardware.

1.6.3 Getting Rid of Zones

As mentioned in Section 1.2.10, we believe that further progress in hybrid verification will require some backtracking from the orthodox approach to reachbility analysis which is too "breadth-first" and less single-trajectory oriented. Our work on scheduling illustrates how this can be done. The modeling of acyclic optimal scheduling problems with timed automata is straightforward, as well as its solvability by standard reachability analysis. However trying to solve it using zone-based reachability we have quickly reached the limits. Upon closer inspection we realized that the dense temporal uncertainty is, in fact, a result of a choice of a scheduler which may start a task at any time since it is enabled. It turned out that the optimum is achievable by a special class of "non-lazy" strategies that either grant the resource upon arrival to a state where the task in question is enabled, or wait until something else happens. This observation allowed us to analyze a finite number of runs and store states as clock vectors rather than polyhedra. Using this technique we could solve scheduling problems much larger than possible by standard TA tools [AKM03, AAM05]. Moreover, a similar observation let us solve preem-

tive scheduling problems [AM02] without the trouble associated with the different type of zones that correspond to stopwatch automata. Although the optimality of non-lazy solutions does not extend to all types of scheduling problems, we believe that restricting the exploration to a discrete subset of the scheduler actions is a promising idea and its applicability to the actions of the adversary is worth being investigated.

1.7 From Control Loops to Real-Time Programs

Hybrid systems research, being situated is one of the intersections of computation and control, helps also in understanding topics related to the computer realization of control loops. Issues such as sampling, numerical integration, code generation, schedluing, robustness or power consumption are better understood when equipped with the conceptual tools of both disciplines. In this sense, the hybrid research at Verimag has been privilleged to enjoy the presence of Paul Caspi as a never-ending source of practical and theoretical knowledge. His collaboration with Stavros Tripakis on transforming control loops written in Simulink/Stateflow into scheduled Lustre programs, described elsewhere in Section **??** is a prime example of this synergy. It is part of the theses of Adrian Curic and Christos Sofronis, with a contribution of Norman Scaife. On the pedagogical side, the introductory survey [CM05] attempts to classify and explain the various implementation strategies for control-oriented embedded systems.

1.8 Analog Verification

Digital circuits have been one of the driving forces behind discrete verification, due to their mass pruduction, which justifies large investments in exhaustive validation prior to irreversible fabrication, as well as their complexity which demanded performant verification techniques. There are some reasons to believe that analog and mixed-signals circuits will play a similar role in the development and acceptance of validation techniques based on hybrid models. It is commonly-accepted that the analog parts of systems-on-a-chip risk becoming a bottleneck in the design flow due to the lack of system-level EDA tools in general and validation tools in particular. We are actively involved in perliminary exploration in this domain which includes the verification of some small circuits [DDM04, FKRM05], the generation of monitors for signal temporal logic [MN04], the organization of a workshop on analog verification [DM05] and interaction with academic collaborator and industrial partners of the PROSYD project. Hopefully these activities will bear fruits

in the future.

1.9 Concluding Remarks

Inter-disciplinary research is a double-edged sword. On one hand it can serve as a fertile ground for new ideas, sometimes much more interesting that those encountered in "standard" disciplines which very often degenerate into a boring repetition. On the other hand, inter-disciplinarity can be used to hide mediocre and even bad work that could not be accepted within any established serious community. It could also be an opportunity to re-cycle old results and find imaginary nails for one's own hammer. We hope that the work described in this chapter does not fall completely withn the latter category. We believe to have exported some useful computer science ideas to hybrid systems research and learned few things in the process. A large part of the research described here represents a conscious choice concerning the niche in the theory-to-application spectrum that we want explore, which is closer to the theoretical side than other approaches to embedded systems. This choice is based on the belief that when developing new models for new phenomena it is important to focus on a few new aspects as possible rather than go directly and try to solve practical problems with all their dirty details, which often take more than 80% of the effort. We believe that the possibility to opt for long-term goals and not produce complete solutions to industrial problems (or pretend to produce such) is a privilege of the researcher, compared to the engineer, and when not abused, it can eventually contribute to technological progress, sooner or later.

We close this chapter with a final remark on *timed systems*. While hybrid systems have their natural niche in the intersection of control and computer science, and are easy to explain to the relative layman, this is unfortunately not the case for timed systems whose understanding requires some sensitivity to subtle issues concerning levels of abstraction. The application of timed automata are spread all over the place (scheduling at various scales, timing analysis of hardware and software, time management and planning) and their treatment is often domain-specific and community-specific. Timed automata may give all these phenomena a state-space based model as do automata for the discrete case and differential equations for the continuous case. However, much of the theoretical work on timed automata only pays a lip service to applications, and then goes on to prove theorems, some of which are interesting. We believe that this unhealthy situation will change when the scalability barrier will be broken, and we keep on directing our efforts toward this goal.

Bibliography

[AAM05] Y. Abdeddaïm, E. Asarin, and O. Maler. Scheduling with timed automata. *Theoretical Computer Science*, 2005.

[Abd02] Yasmina Abdeddaïm. *Scheduling with Timed Automata*. PhD thesis, INPG Grenoble, November 2002.

[ACM02] E. Asarin, P. Caspi, and O. Maler. Timed regualr expressions. *Journal of the ACM*, 49:172–205, 2002.

[AD02] E. Asarin and C. Dima. Balanced timed regular expressions. In *MTCS'2002*, volume 68 of *ENTCS*, Brno, Czech Republic, August 2002. issue 5.

[AD04] E. Asarin and T. Dang. Abstraction by projection. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control*, LNCS 2993, pages 32–47. Springer-Verlag, 2004.

[ADE+01] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierarchical hybrid modeling of embedded systems. In T.A. Henzinger and C.M. Kirsch, editors, *Embedded Software, LNCS 2211*, pages 14–31. Springer, 2001.

[ADG03] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Computation and Control*, LNCS 2623, pages 20–35. Springer-Verlag, 2003.

[ADI02] R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems via predicate abstraction. In M. Greenstreet and C. Tomlin, editors, *Hybrid Systems: Computation and Control*. Springer, 2002. to appear.

[ADI03a] R. Alur, T. Dang, and F. Ivancic. Counter-example guided predicate abstraction of hybrid systems. In *Tools and Algorithms for the Construction and Analysis of System*. Springer-Verlag, 2003.

[ADI03b] R. Alur, T. Dang, and F. Ivancic. Progress on reachability analysis of hybrid systems using predicate abstraction. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, LNCS 2623. Springer-Verlag, 2003.

[ADI04] R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems via predicate abstraction. *ACM transactions on embedded computing systems (TECS)*, 2004.

[ADI05] R. Alur, T. Dang, and F. Ivancic. Counter-example guided predicate abstraction of hybrid systems. *Theoretical Computer Science*, 2005. to appear.

[ADM01a] E. Asarin, T. Dang, and O. Maler. d/dt: A tool for reachability analysis of continuous and hybrid systems. In *Proc. IFAC Symposium on Non-linear Control*. IFAC, 2001.

[ADM01b] E. Asarin, T. Dang, and O. Maler. d/dt: A verification tool for hybrid systems. In *Proc. of the 40th IEEE CDC, Orlando, Florida*, 2001.

[ADM02] E. Asarin, T. Dang, and O. Maler. The d/dt tool for verification of hybrid systems. In *Proc. CAV'02*, number 2404 in LNCS, pages 365–370. Springer, 2002.

[AKM03] Y. Abdeddaïm, A. Kerbaa, and O. Maler. Task graph scheduling using timed automata. In *FMPPTA*, 2003.

[Ali03] Olfa Ben Sik Ali. Simulation des systèmes continus ouverts. Master's thesis, DEA Informatique: Système et Communication, Université Joseph Fourier, Grenoble, June 2003.

[AM02] Y. Abdeddaïm and O. Maler. Preemptive job-shop scheduling using stopwatch automata. In J.-P. Katoen and P. Stevens, editors, *Proc. TACAS'02*, number 2280 in LNCS, pages 113–126. Springer, 2002.

[AMY02] E. Asarin, O. Maler, and S. Yovine, editors. *Theory and Practice of Timed Systems, TPTS'02*, April 2002.

[APSY02] E. Asarin, G. Pace, G. Schneider, and S. Yovine. Speedi – a verification tool for polygonal hybrid systems. In *Proceedings of "Computer Aided Verification, CAV'02"*, Lecture Notes in Computer Science, Denmark, July 2002. Springer.

[AS02] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In *CONCUR'2002*, number 2421 in LNCS, pages 193–208, Brno, Czech Republic, August 2002. Springer.

[ASY01] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In A. Sangiovani-Vincentelli and M. di Bendetto, editors, *Hybrid Systems: Computation and Control*, number 2034 in LNCS, pages 89–104. Springer, 2001.

[ASY02] E. Asarin, G. Schneider, and S. Yovine. Towards computing phase portraits of polygonal differential inclusions. In M. Greenstreet and C. Tomlin, editors, *Hybrid Systems: Computation and Control*. Springer, 2002. to appear.

[AT02] K. Altisen and S. Tripakis. Tools for controller synthesis of timed systems. In *RT-TOOLS*, 2002.

[BJMY02] M. Bozga, H. Jianmin, O. Maler, and S. Yovine. Verification of asynchronous circuits using timed automata. In *Proc. TPTS 2002*, number 65 in ENTCS, 2002.

[BKM04] M. Bozga, A. Kerbaa, and O. Maler. Scheduling acyclic branching programs on parallel machines. In *Proc. RTSS'04*. IEEE Press, 2004.

[CAMN04] S. Cotton, E. Asarin, O. Maler, and P. Niebert. Some progress in satisfiability checking for difference logic. In *Proc. FORMATS/FTRTFT'04*. Springer, 2004.

[CM05] P. Caspi and O. Maler. From control loops to real-time programs. In *Handbook of Networked and Embedded Control Systems*. CRC Press, 2005.

[Dan05] T. Dang. Application of reachability analysis to idle speed control synthesis. *International Journal of Software Engineering &*

Knowledge Engineering IJSEKE, Special issue of selected papers from the International Embedded and Hybrid Systems Conference IEHSC'05:to appear, 2005.

[DDM04] T. Dang, A. Donze, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *FMCAD'04*. Springer, 2004.

[Dim01a] C. Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6:3–23, 2001.

[Dim01b] C. Dima. *Théorie algébrique des langages formels temps réel*. PhD thesis, Université Joseph Fourier, Grenoble, 2001.

[Dim02] Catalin Dima. Computing reachability relations in timed automata. In *LICS*, 2002.

[DM05] T. Dang and O. Maler, editors. *Workshop on Formal Verification of Analog Circuits, FAC'05*, Edinburgh, apr 2005. LFCS, University of Edinburgh.

[DMMRY01] J. Della Dora, A. Maignan, M. Mirica-Ruse, and S. Yovine. Hybrid computation. In *ISSAC'01*, July 2001.

[Don05] Alexandre Donzé. On temporal differences algorithms for continuous systems. Technical Report TR-2005-8, Verimag Technical Report, 2005.

[DY01] J. Della Dora and S. Yovine. A methodology for analyzing the dynamics of hybrid systems. In *European Control Conference, ECC'01*, September 2001.

[FKRM05] G. Frehse, B. Krogh, R. Rutenbar, and O. Maler. Time domain verification of oscillator circuit properties. In T. Dang and O. Maler, editors, *Workshop on Formal Verification of Analog Circuits, FAC'05*, pages 1–13, 2005.

[Gir04] A. Girard. *Analyse algorithmique des systèmes hybrides*. PhD thesis, Institut National Polytechnique de Grenoble, 2004.

[Klo04] Ch. Kloukinas. Data-mining synthesised schedulers for hard real-time systems. In *Proceedings of the 19th IEEE Conference on Automated Software Engineering (ASE 2004)*, Linz, Austria, September 2004. IEEE Computer Society Press.

[KMSK03] J. Kapinski, O. Maler, O. Stursberg, and B. Krogh. On systematic simulation of open continuous systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*. Springer, 2003.

[KNY03] Ch. Kloukinas, Ch. Nakhli, and S. Yovine. A methodology and tool support for generating scheduled native code for real-time Java applications. In *EMSOFT 2003*, volume 2855 of *LNCS*, 2003.

[KT04] M. Krichen and S. Tripakis. Black-box conformance testing for real-time systems. In *11th International SPIN Workshop on Model Checking of Software (SPIN' 04)*, volume 2989 of *LNCS*. Springer, 2004.

[KY03] Ch. Kloukinas and S. Yovine. Synthesis of safe, QoS extendible, application specific schedulers for heterogeneous real-time systems. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, ISBN 0-7695-1936-9, 2003.

[LPY01] G. Lafferriere, G. Pappas, and S. Yovine. Symbolic reachability computation of families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, September 2001.

[Mal01] O. Maler. Guest editorial: Verification of hybrid systems. *European Journal of Control*, 7:357–365, 2001.

[Mal02] O. Maler. Control from computer science. *Annual Reviews in Control*, 2002.

[Mal04] O. Maler. On optimal and sub-optimal control in the presence of adversaries. In *Proc. Wodes'04*, 2004.

[MKM02] O. Maler, B. Krogh, and M. Mahfoudh. On control with bounded computational resources. In *FTRTFT'02*, number 2469 in LNCS, pages 147–164, 2002.

[MN04] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proc. FORMATS/FTRTFT'04*. Springer, 2004.

[MNP05] O. Maler, D. Nickovic, and A. Pnueli. Real-time temporal logic: past, present, future. submitted for publication, 2005.

[MP03] O. Maler and A. Pnueli, editors. *Hybrid Systems: Computation and control, HSCC'03*, volume 2326 of *LNCS*, April 2003.

[MP04] O. Maler and A. Pnueli. On recognizable timed languages. In I. Walukiewich, editor, *FOSSACS*. Springer, 2004.

[NMA+02] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, N. Jain, and O. Maler. Verification of timed automata via satisfiability checking. In W. Damm and E-R Olderog, editors, *FTRTFT*, volume 2469 of *LNCS*, pages 225–244. Springer, 2002.

[PTV01] A. Puri, S. Tripakis, and P. Varaiya. Problems and examples of decentralized observation and control for discrete event systems. In *Workshop on Supervisory Control for Discrete Event Systems, SCODES*, 2001.

[SBM03] R. Ben Salah, M. Bozga, and O. Maler. On timing analysis of combinational circuits. In *FORMATS'03*, 2003.

[SBM05] R. Ben Salah, M. Bozga, and O. Maler. Automatic abstraction of real-time components. submitted for publication, 2005.

[SD05] S. Shapero and A. Donzé. Search-based control of the simplified model of the abb case study. Technical Report 2005-6, Verimag, February 2005.

[Tri01] S. Tripakis. Undecidable problems of decentralized observation and control. In *IEEE Conference on Decision and Control*, 2001.

[Tri02] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. In *6th International Workshop on Discrete Event Systems (WODES'02)*. IEEE CS Press, 2002.

[Tri04a] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. *IEEE Transactions on Automatic Control*, 49(9), September 2004.

[Tri04b] S. Tripakis. Folk theorems on the determinization and minimization of timed automata. In *Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*. Springer, 2004.

[Tri05] S. Tripakis. Two-phase distributed observation problems. In *5th Intl. Conf. on Application of Concurrency to System Design (ACSD'05)*. IEEE CS Press, 2005. to appear.

[Zan04] Marcelo Zanconi. *Modélisation et analyse de systèmes temps-réel avec préemption, incertitude et dépendences*. PhD thesis, Université Joseph Fourier, Juin 2004.