Annex 7.1.B
Time-Deterministic Hybrid Transition Systems

Simin Nadjm-Tehrani

Dept. of Computer & Information Science, Linköping University,
S-581 83 Linköping, Sweden, e-mail: simin@ida.liu.se

1 Introduction

By a hybrid theory we mean a combination of the theory for continuous dynamic systems with the theory for discrete dynamic systems. This area, although highly underdeveloped, is an active area of research leading to candidate hybrid theories within different research disciplines. On the one hand, computer scientists are extending the classical theories of discrete systems, logics and automata, focusing on properties such as computational complexity, decidability and completeness. On the other hand, control engineers are expanding the classical theories of linear (and non-linear) systems, while focusing on properties such as time-optimality, stability and robustness. Both of these research directions are promising and have so far resulted in at least partially overlapping modelling and analysis paradigms.

While the theoretical work is carried out, safety-critical systems are being built and analyzed at great costs. Successful application of formal verification techniques in such systems needs to start from engineering models of hybrid systems and systematically transform them to analysable models in the theory. The work presented here draws from the experience with a number of case-studies, where the idea has been to start from the engineering models of physical systems "as they are".

Hybrid Transition Systems were proposed in 1993 as an attempt to capture both mode-switching physical systems and non-deterministically timed computer systems in a single formalism. The formalism was studied in the context of several realistic examples: a two car no-collision scenario [14], a 16th century siphon pump machine [21], and the landing gear system of an aircraft [15, 20, 16] among others. Some of these examples were verified by the application of deductive methods. Simpler models were augmented by addition of invariances after which they could be model-checked as linear hybrid automata (LHA) [18]. In this paper we propose a restriction of the formalism in which the engineering models can still be plugged in with no additional effort, and prove certain compositionality properties in the restricted model.

The motivation for the restrictions are twofold. First, a number of changes in the definition of the operational semantics and parallel composition operator facilitates proofs of compositionality. Second, restriction to time-determinism particularly suits analysis of systems controlled by programs from the so-called synchronous family of formal languages [8].

Safety properties of such embedded systems can of course be verified if a discrete model of the environment is derived [24]. As synchronous languages are becoming more widespread in industrial systems, and interfaces to verification tools are forthcoming (see the Lustre-PVS connection for example [3]), it is even interesting to study the timing behaviour of such systems in relation to their environments. This generally requires continuous analysis combined with deductive proofs. The restriction to time-determinism simplifies the development of representation templates in deductive proof tools (e.g. PVS [17]).

Hybrid transition systems in their full generality describe behaviours of systems as interleavings of continuous phases of activity (having positive durations) and discrete transitions (taking zero time). Each discrete transition, however, may be taken according to a (non-deterministic) time constraint. A transition is taken a least l time units and at most u time units after it has been enabled $(0 \leq l, u \leq \infty)$.

The work on case-studies suggests the use of time-deterministic control programs in many applications, and the need for deductive reasoning about cases where models of the physical environment is non-trivial (eliminating the possibility of model checking) [7]. We therefore propose to restrict the model to time-deterministic transitions while keeping the expressiveness with regard to dynamic continuous systems. Each discrete transition is thus required to be taken within a fixed period of time from the time at which it was enabled, but the fixed period may differ for different transitions.

We therefore expect to keep the best of both worlds in the following sense. Hybrid models are often built up from several modules each representing either a control unit or a physical apparatus (mechanics, hyraulics, electronics, etc.). Our experience shows that models derived using systematic physical modelling (e.g. by bond graphs) are naturally described by switching phases of continuous activity immediately on satisfaction of certain conditions (i.e. $l = u = 0$). Also, many realisations of control programs have a fixed period for each control function ($l = u = d$ for some period d). Having models and techniques for analysing the logical and causal behaviours of the synchronous programs [8] on their own, we can now extend the analysis to the testing of "the synchrony hypothesis" using a detailed model of the environment.

A different application of the time-deterministic model can be the composition of timed models of synchronous subsystems, each having their own computational period; the resulting system being a timed description of asynchronous networks of synchronous processes. Analysis of such applications is a subject for future work.

The rest of this paper describes certain features of hybrid transition systems (HTS) for natural modelling of engineering systems in section 2. Then follows the formal definition of time-deterministic HTS, and the parallel composition operator for this class of systems in section 3. Section 4 treats issues related to compositionality and progress properties, and section 5 discusses relation to other works.

2 Hybrid transition systems

Hybrid transition systems treat the discrete and continuous elements in each state on a par. That is, all elements of state are conceptually represented as piecewise continuous functions of time. Each system has a finite number of modes. In each mode, the continuous elements of the state are concretely represented by a set of differential and algebraic equations in state space form (see Fig. 1), where a real-valued variable may change either in accordance with a differential equation or according to an algebraic equation in each mode. This facilitates modelling the cases where the dimention of the system is changed from one mode to another (see the siphon pump in [22]). In any case, the representation of continuous change as above provides a natural interface to engineering models which have this form, and which can be plugged into the hybrid model without additional transformation (e.g. addition of invariances and exit conditions as in LHA).

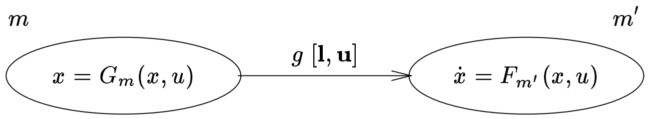

Fig. 1. A schematic illustration of hybrid transition systems as a graph.

The discrete elements (piecewise constant functions of time) are represented as variables taking constant values in each mode. Thus, changes in state within a mode are captured by differential and algebraic equations. Changes of state from one mode to another are conditional upon a *guard* becoming true, i.e. a boolean expression g over the state and input variables.

The uniform treatment of continuous and discrete elements as state variables provides a natural means of communication between subsystems. As opposed to shared labels or events we treat every state change as immediately visible by all other subsystems. Thus communication is by shared variables, where each variable is only allowed to be changed by one subsystem (the one which has this variable among its state variables). Other subsystems may (or may not) have this variable as an input variable – a model which supports non symmetric communication (broadcast as opposed to rendezvous).

The model is specially useful for modular developments. For each component the changes in state is defined in terms of current state and the current input. The input is, however, not modelled explicitly (unless it is the state of another subsystem). This allows the proper treatment of disturbances or inputs whose modelling is to be postponed (e.g. the driver behaviours in [14]).

Consider a mode change conditional upon the guard g and having a timing constraint $[l, u]$, as depicted above. Then a watchdog for the guard can result in setting a timer to zero whenever the value of g changes to true. From this point onwards the transition *may* be taken, but at the point when the timer hits u the transition *must* be taken. In what follows, we consider the systems where all the transitions *must* be taken at the point which is exactly t time units apart from the time when their guard becomes true.

3 Time-deterministic HTS

We now define time-deterministic hybrid transition systems (TD-HTS) formally.

Definition 1. A **time-deterministic hybrid transition system** is a tuple $\langle M, X, U, F, I, T\rangle$, where

M**:** is a non-empty set of *mode sets* $\{m_1, \ldots, m_q\}$. For a simple system m_i are singleton, for a composed system each m_i is a set of mode elements.

X**:** is a set of typed state variables with disjoint subsets X_c and X_d, $X = X_c \cup X_d$, where the domain of X_c variables is $\mathbb{R}$.

U**:** is a set of typed input variables $U = U_c \cup U_d$ where the domain of the variables in U_c is $\mathbb{R}$ and $X \cap U = \emptyset$.

$F : M \times X \to \mathcal{E}$ is a function associating an equation with each state variable in each mode, where $e = F(m, x)$ has the following forms:
- if $x \in X_d$ then $e = \ulcorner x = c \urcorner$ for some value c of the right type
- if $x \in X_c$ then
  - $F(m, x) = \ulcorner \dot{x} = f(\bar{x}, \bar{u}) \urcorner$ for some function f of appropriate type,
  - or $F(m, x) = \ulcorner x = g(\bar{x}, \bar{u}) \urcorner$ for some function g of appropriate type

  where $\bar{x}$ and $\bar{u}$ denote vectors of variables over X_c and U_c.

I**:** is an initial configuration consisting of $\langle m_0, s_0\rangle$, where $m_0 \in M$ and $s_0 \in S$.

T**:** is a set of *mode transitions* $\tau = \langle m, m', g, t\rangle$ where $m, m' \in M$, g is a boolean expression over terms of the form $z \# w$, with $z \in X \cup U$, $\# \in \{=, \leq, \geq, <, >\}$, and w is a type consistent term over X and U, elements from their value domains, and uninterpreted function symbols. $t \in \mathbb{N}$ corresponds to an exact delay on the transition τ.

□

The set of states of the system, S, is the set of type consistent interpretations of the variables in X. We denote by $s(x)$ the value of the variable x in the interpretation s.

The guard g is defined to be true in an interpretation $\mathcal{I}$ if it evaluates to true when all the variables z in g are substituted by the values given by $\mathcal{I}(z)$.

The operational semantics can be informally described as follows. A run for a system is defined from the initial configuration for an input time-function (a function from the reals to interpretations for the input variables). The run consists of a sequence of mode-state-input-time tuples where the changes in the input components are given by the input time-function at given time points.

The recorded time points are those at which a mode change takes place, or some guard to some transition changes its truth value. The state changes are compatible with the solutions to equations in each mode. The discrete mode changes are recorded by having the *same* time component and different mode-state components (though the value of continuous state variables are unchanged at mode changes). Such a mode change appears in a run if the guard g for some τ has been true for the duration t as dictated by τ. All enabled transitions take place at the end of their respective durations t (i.e. they *must* take place). If there are several such transitions, then one will appear non-deterministically in the current position in the run, and the others will also take place (with the same time component but at the next position in the run) provided they are still enabled in the current mode.

Next we give a formal account of the operational semantics. In definitions below we represent the interpretation of a set of variables Z as the set $\{\langle z, v\rangle \mid z \in Z, v \in \text{ domain of } z\}$.

Definition 2. Let $\langle M, X, U, F, I, T\rangle$ be a time-deterministic hybrid transition system, and γ denote a finitely variable function from $\mathbb{R}$ to the set of interpretations over U. Then the **run** of the system with input γ is an infinite sequence of situations $\sigma_0, \sigma_1, \ldots$ such that:

- $\sigma_j = \langle m_j, s_j, e_j, t_j\rangle$ where $e_j = \gamma(t_j)$
- $t_0, t_1, \ldots$ is a progressive time sequence with $t_j \in \mathbb{R}$
- $\sigma_0 = \langle m_0, s_0, \gamma(0), 0\rangle$
- each state component $s_j \in S$, and is compatible with the trajectory of the system in mode m_j – i.e. $s_j(x) = \gamma_x(t_j)$ where γ_x denotes the solution to the DAE in mode m_j as defined by F with respect to variable x, given the initial values corresponding to s_i and e_i from the last position i in the run when the mode changed, that is a point $i < j$, such that $m_i \neq m_j$, and $\forall k\ i < k \leq j\ m_k = m_j$
- at every t_j either the mode changes, or the guard to some transition changes truth value, i.e. $m_j = m_{j+1} \rightarrow \exists\langle m, m', g, t\rangle \in T$ such that g is true (false) in $s_j \cup e_j$ and false (true) in $s_{j+1} \cup e_{j+1}$
- mode changes take no time, i.e. $m_j \neq m_{j+1} \rightarrow t_j = t_{j+1}$
- a transition is taken only if it is enabled for long enough, i.e. $m_j \neq m_{j+1} \rightarrow \exists\tau = \langle m, m', g, t\rangle \in T\ \exists i \leq j$ such that $\forall i \leq k \leq j,\ g$ is true in $s_k \cup e_k,\ m \cap m_k \neq \emptyset$ and $m' \cap m_{j+1} \neq \emptyset$
- for every transition with gaurd g and delay t, if g becomes true at a time point t_i and stays true at all subsequent situations prior to the time point $t_j = t_i + t$, then the transition is taken at t_j provided that it can be taken , i.e. $\forall\tau = \langle m, m', g, t\rangle$, if $t_j = t_i + t,\ g$ true in $s_i \cup e_i,$ and $\forall i \leq k \leq j,\ g$ true in $s_k \cup e_k$ and $m \cap m_k \neq \emptyset$ then $\exists\sigma = \langle m'', s, e, t_j\rangle$ where $m' \cap m'' \neq \emptyset,\ s(x) = s_j(x)$ for all $x \in X_c$, and is otherwise defined by $F(m'', x)$ (for $x \in X_d$).

□

Note that we want to force progress as a rule. So any transitions which have been enabled “for long enough” *must* be taken. On the other hand, we do not want to force determinism — specially since parallel composition of several subsystems may lead to independent transitions get ready for being taken at the same time. The semantics thus allows for all such transitions to take place in an arbitrary order but at the same time point *provided* that they are not in causal conflict with each other. The latter is not explicit in the semantic definition but is implicit: a system which does not have a run according to the above definition does not have a well defined semantics. The detection of causal paradoxes , e.g. that a taken transition disables an already enabled transition, may be investigated using similar procedures to those for synchronous systems [5] – taking no regard to the timing delay.

Next we give the definition for composition of TD-HTS. Note that composition is defined if a condition on shared variables is respected. This might be slightly problematic with certain systems in which common outputs are desired, e.g. subsystems which both emit the same alarm signal i different situations. This situation can however be remedied by renaming of the state variable in one subsystem after composition.

Definition 3. Given time-deterministic HTS $H_1 = \langle M_1, X_1, U_1, F_1, I_1, T_1\rangle$, and $H_2 = \langle M_2, X_2, U_2, F_2, I_2, T_2\rangle$, such that $X_1 \cap X_2 = \emptyset$ and $M_1 \cap M_2 = \emptyset$, we define their **composition** denoted by $H_1 \| H_2$, as the time-deterministic HTS $H = \langle M, X, U, F, I, T\rangle$ where:

$M = \{m_1 \cup m_2 \mid m_1 \in M_1,\ m_2 \in M_2\}$,
$X = X_1 \cup X_2$,
$U = (U_1 \cup U_2) - X$,
$F : M \times X \to \mathcal{E}$ where $\mathcal{E}$ is the set of equations $\mathcal{E}_1 \cup \mathcal{E}_2$, such that
$F(m, x) = F_i(m_i, x)$, for $x \in X_i,\ m_i \subset m \in M,\ m_i \in M_i$ $(i \in \{1, 2\})$,
T is the smallest set of transitions such that:

if $\langle m_1, m_1', g_1, t_1\rangle \in T_1$, then for all $m \in M_2$, we have $\langle m_1 \cup m, m_1' \cup m, g_1, t_1\rangle \in T$, and
if $\langle m_2, m_2', g_2, t_2\rangle \in T_2$, then for all $m \in M_1$, we have $\langle m \cup m_2, m \cup m_2', g_2, t_2\rangle \in T$.

□

Proposition 4. *The parallel composition operator is commutative and associative.*

Proof. Follows directly from the definitions and the properties of set union.

□

4 Compositionality of TD-HTS

We are interested in compositionality of hybrid systems in two different ways.

- Proving assertions of the type "if H_1 and H_2 satisfy a particular property, so does $H_1||H_2$".
- Proving that "if $H_1||H_2$ satisfies a property P and H_1 is equivalent to H_3 in some sense, then $H3||H_2$ satisfies P.

The first one is needed for bottom up modelling and verification of systems, and the second for making simplifications prior to analysis, or for refining abstract designs to implementations. In this paper we treat the first aspect of compositionality. That is, we show that composition of systems preserves certain interesting properties. A central property for hybrid transition systems is having a well-defined semantics in terms of a set of runs. Thus, we would like to show that if H_1 and H_2 each have well-defined semantics in terms of a (non-empty) set of runs, then their composition also has a run. Note that the definition of an TD-HTS does not guarantee existance of a run for the system. In particular, systems with obscure behaviours (e.g. the so-called Zeno behaviours in which time progresses in small increments but never beyond a particular bound) do not have a run, but are not syntactically excluded. Unfortunately, we can not exactly characterize systems whose states are finitely variable (have finite number of discrete state state changes in a finite interval of time), or compositions of systems which have no runs.

Therefore, we proceed as follows. First, we distinguish systems in which an infinite number of discrete transitions are possible in a single point in time. The intuition for this rests on experience with modelling realistic systems. Models of physical systems which are derived with systematic modelling techniques do not exhibit Zeno behaviours, despite the fact that discrete transitions can naturally assumed to be instantanuous. Models of computer systems are definitely non-Zeno as long as there are non-zero delays associated with every transition. Therefore, we define the notion of *admissible* systems which at least do not exhibit the undesired behaviour of changing infinitely often in a single time point.

Definition 5. Let $H = \langle M, X, U, F, I, T\rangle$ be a TD-HTS. We associate a labelled graph $G_H = \langle V, E, L\rangle$ with H such $m \in M$ iff $v_m \in V$, and $\langle v, v'\rangle \in E$ iff $\exists \tau = \langle m, m', g, t\rangle \in T$, and the labelling function L associates a lable (g, t) with the edge $\langle v_m, v'_m\rangle$. We define the TD-HTS H as **admissible** iff for every cycle in G_H if all labels are of the form $(g, 0)$, then there are at least two edges in the cycle with guards g and g', where g and g' are mutually exclusive when evaluated in all interpretations of X and U.

□

Proposition 6. *If H_1 and H_2 are admissible then $H_1||H_2$ is admissible.* □

Obviously, this property does not guarantee existance of a run when two systems are composed, but it eliminates some cases which depend on inappropriate enabling of guards.

Next we show that the parallel composition of H_1 and H_2, produces a system H whose semantics is defined in terms of semantics of its constituents provided that it has a run. That is, a sequence belongs to the set of runs of H if it is grounded in runs of H_1 and H_2 in a sense that we make more precise as follows.

Proposition 7. *Let H_1 and H_2 be two TD-HTSs. Let γ_1, γ_2 and γ be three input time functions with the following properties:*

1. *H_i has a run with γ_i, $i \in \{1, 2\}$.*
2. *$\gamma(t)(u) = \gamma_1(t)(u)$ if $u \in U_1 - X_2$, and $\gamma(t)(u) = \gamma_2(t)(u)$ if $u \in U_2 - X_1$.*
3. *$\gamma_1(t_j)(u) = e_j(u)$ at every position j of the run for H_2 with γ_2 if $u \in U_1 \cap X_2$, and $\gamma_2(t_j)(u) = e_j(u)$ at every position j of the run for H_1 with γ_1 if $u \in U_2 \cap X_1$.*

If $H = H_1 || H_2$ has a run with γ, then for every such run of H, there is some run of H_1 with γ_1 (and H_2 with γ_2) such that for every element at position i in the run of H there is a corresponding element with the same time component t_i in the run of $H_1 (or H_2)$ such that the restriction of the element in H to (state, mode and input) variables of $H_1 (or H_2)$ gives the element in the run for $H_1 (or H_2)$.

□

Proof. By induction on the sequence of situations in the runs of H. Consider an arbitrary run of H with γ.
Base step: for the initial state the condition is trivially true.
Induction step: Consider a fragment of the run: $\ldots, \sigma_i, \sigma_{i+1}, \ldots$

We will show that for any element σ_{i+1} at position $i + 1$ there is an element related to it at a position $j < i + 1$ in the run for H, such that, if there is an element corresponding to σ_j in some run of H_1 with γ_1 (or some run of H_2 with γ_2), then there is an element corresponding to σ_{i+1} in that run of H_1 (or H_2). There are two cases:

1. $m_i \neq m_{i+1}$
   According to the operational semantics for H there exists a transition $\tau = \langle m, m', g, t \rangle$ in T such that $m_i \cap m \neq \emptyset, m_{i+1} = m'$, and g has been true in every $s_k \cup e_k$ since a position j in which g became true. Assume that σ_j restricted to variables of H_1 (or H_2) exists in a run for H_1 (or H_2). Then according to the operational semantics, a position corresponding to σ_{i+1} exists in the run of H_1 (or H_2).
2. $m_i = m_{i+1}$
   According to the operational semantics there is a transition with guard g in H such that g is false (true) in $s_i \cup e_i$ and true (false) in $s_{i+1} \cup e_{i+1}$. According to the definition of parallel composition this transition (guard) can be traced to one of the two systems H_1 and H_2. Without loss of generality assume that it comes from H_1. Consider a run for H_1 with γ_1. Assume that there is a situation σ_k with time component equal to t_i in this run for H_1, whose mode and state components are restrictions of m_i and s_i to variables of H_1. We have to show that σ_{k+1} in that run corresponds to a restriction of σ_{i+1} to

variables of H_1. First, we show that g is false (true) in $s_k \cup e_k$. This easily follows since s_k is a restriction of s_i to X_1, variables in g are a subset of $X_1 \cup U_1$, and due to conditions 2 and 3 in the antecedents of the proposition, the value of these variables is the same as that in $s_i \cup e_i$.

Next we show that g changes truth value at σ_{k+1} and that takes place at time point t_{i+1}. Again, since the variables in g are a subset of $X_1 \cup U_1$, based on definition of F in parallel composition, and restrictions 2 and 3 in the antecedent to the proposition, we can state the following: if t_{i+1} is the first time point after t_i in which g changed truth value in the run of H, then t_{i+1} is also the first time point after σ_k in the run of H_1 in which g changes truth value. Hence, according to the operational semantics, there exists a situation $\sigma_{k+1} = \langle m', s', e', t_{i+1} \rangle$ such that $m' \subset m_1,\ s' = s_{i+1}$ restricted to X_1, and $e' = \gamma_1(t_{i+1})$ in the run of H_1.

□

5 Related works

We share a common aspiration with data flow approaches like Signal [4]: that of treating the discrete and continuous elements on a par.

Hybrid transition systems can be seen as a modular version of phase transition systems [12], our main contribution being the treatment of compositionality and separation of input and state components. Other versions of phase transition systems [13] include the notion of important events which, as well as enabling conditions (corresponding to our guards), include assertions from the requirements specifcation for a system. Hybrid transition systems do not mix the model of a system and its requirements. Hybrid automata [18] differ mainly on the model for communication and parallel composition. There, communication is by shared synchronisation labels. Moreover, composition is defined on systems which have the same set of continuous variables, making modular models of different subsystems less natural. Hybrid automata also require invariants to force progress. This is often a significant additional modelling step not directly present in engineering models. In our model, progress is by default, and addition of invariances are seen as part of the verification process.

Modular modelling appears in certain other frameworks, e.g. in the work by Kaivola on an a discrete version of transition systems [9], and the work by Westhead and Hallam [23] which models a hybrid computation as a limit to discrete synchronous computational processes. Abadi and Lamport [1] discuss composition of, and decomposition into, specification modules. Their specifications are representations of safety and liveness properties in temporal logic. Separation of input and state arises naturally in models proposed within control theory, e.g. several models discussed in [6]. However, aspects such as stability are in the focus of discussions rather than compositionality.

A hybrid theory which treats aspects of compositionality on hybrid models can be found in the work by Lynch et.al. [11] wherein both communication by shared variables and shared labels is considered. This work also attacks proving non-Zenoness by switching to a game theoretic framework.

Going to the domain of timed systems, there are works which treat composition in presence of non-deterministic timing constraints in variants of timed automata [19, 10]. Sifakis and Yovine distinguish between transitions which *must* take place and those which *may* take place by using a notion of deadline in addition to the usual notion of invariance in timed automata. Kesten, Manna and Pnueli, on the other hand separate the progress conditions and enabling conditions by using different assertions for each. The former being global assertions and the latter as local predicates on each transition.

6 Summary and future works

The paper presents a model for hybrid systems where all elements of state (discrete and continuous) are represented as state variables. Modes of continuous activity are governed by differential and algebraic equations in state space form, and communication is by shared variables. The model is a modification of hybrid transition systems presented earlier — the modifications being restriction to time-determinism and composition by union of mode sets. The restrictions are based on practical experience with modelling realisic engineering systems, and an intention to link to verification efforts on the synchronous family of languages. A treatment of compositionality for open subsystems was discussed and a proof of preservation of semantics on composing subsystems was presented. Future works include application to industrial case studies controlled by a collection of synchronous programs, and investigation of closed system properties using tools such as PVS.

References

1. M. Abadi and L. Lamport. Conjoining Specifications. *ACM transactions on Programming Languages and Systems*, 1995.
2. R. Alur, T.A. Henzinger, and E. Sontag, editors. *Proc. of the DIMACS International Workshop on Verification and Control of Hybrid Systems, LNCS 1066*. Springer Verlag, 1996.
3. S. Bensalem, P. Caspi, and C. Parent-Vigouroux. Handling Data-Flow Programs in PVS. Technical report, VERIMAG, 1996. Draft report.
4. A. Benveniste. Compositional and Uniform Modelling of Hybrid Systems. In Alur et al. [2], pages 41–51.
5. G. Berry. The Constructive Semantics of Pure Esterel. Technical report, Centre de Mathematiques Appliquees, 1995. Draft report.
6. M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technoloy, June 1995. Dissertation no. LIDS-TH-2304.
7. B. Dutertre and V. Stavridou. Formal Requirements Analysis of an Avionics Control System. *IEEE Transactions on Software Engineering*, 23(5), 1997. To appear.

8. N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.
9. R. Kaivola. Compositional Model Checking for Linear-Time Temporal Logic. In *Proc. 4th int. workshop on Computer Aided Verification*, pages 248–259. Springer Verlag, 1992.
10. Y. Kesten, Z. Manna, and A. Pnueli. Verifying Clocked Transition Systems. In Alur et al. [2], pages 13–40.
11. N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O Automata. In Alur et al. [2], pages 496–510.
12. O. Maler, Z. Manna, and A. Pnueli. From Timed to Hybrid Systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G.Rozenberg, editors, *Real-Time: Theory in Practice, Proc. REX workshop 1991*, pages 447–484. Springer Verlag, LNCS 600, 1992.
13. Z. Manna and A. Pnueli. Models for Reactivity. *Acta Informatica*, 30:609–678, 1993.
14. S. Nadjm-Tehrani and J-E. Strömberg. From Physical modelling to Compositional models of Hybrid Systems. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proc. of the 3rd. International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 863*, pages 583–604. Springer Verlag, 1994.
15. S. Nadjm-Tehrani and J-E. Strömberg. Proving Dynamic Properties in an Aerospace Application. In *Proc. of the 16th International Symposium on Real-time Systems*, pages 2–10. IEEE Computer Society Press, December 1995.
16. S. Nadjm-Tehrani and J-E. Strömberg. Verification of Dynamic Properties in an Aerospace application. *Formal Methods in System Design*, 1998. To appear.
17. N. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Proc. 11th International Conference on Automated Deduction, LNCS 607.* Springer Verlag, 1992.
18. A. Pnueli and J. Sifakis, editors. *Theoretical Computer Science, Special issue on Hybrid Systems*, volume 138. Elsevier Publishers, February 1995.
19. J. Sifakis and S. Yovine. Compositional Specification of Timed Systems. In *Proc. of the 13th annual Symposium on Theoretical Aspects of Computer Science, STACS'96, LNCS 1046*, pages 347–359. Springer Verlag, 1996.
20. J.-E. Strömberg and S. Nadjm-Tehrani. Hybrid Systems Verification Combining Duration Calculus and Bond Graphs. In *Proc. IFAC-IFIP-IMACS Conference on Control of Industrial Systems.* IFAC, 1997. To appear.
21. J.-E. Strömberg, S. Nadjm-Tehrani, and J. Top. Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems. In Alur et al. [2], pages 282–293.
22. J.E. Strömberg and S. Nadjm-Tehrani. On Discrete and Hybrid Representation of Hybrid Systems. In *Proceedings of the SCS International Conference on Modeling and Simulation (ESM '94)*, pages 1085–1089, Barcelona, June 1994.
23. M. Westhead and J. Hallam. Modelling Hybrid Systems as the Limit of Discrete Computational Processes. In *Proceedings of the International Conference on Robotics and Automation.* IEEE, 1996.
24. M. Westhead and S. Nadjm-Tehrani. Verification of Embedded Systems using Synchronous Observers. In *Proceedings of the 4th International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 1135*, pages 405–419. Springer Verlag, September 1996.