Modelling and formal verification

By Simin Nadjm-Tehrani and Jan-Erik Strmberg[1]

Fundamental research in formal methods and its deployment in several real world large scale applications has seen a major progress over the past decade. A recent report by the formal methods working group of the ACM workshop on strategic directions in computing research lists an impressive number of cases. Among the notable examples are the application of formal methods and tools to the London air traffic management system [12], avionic software for the Lockheed C130J [7], the Traffic Collision avoidance system (TCAS) which is compulsory for all large passenger airplanes operating in the US [14], the Motorola 68020 microcode processor [3], and the proof of correctness for an SRT division algorithm (similar to the one used by Pentium the error in which costed the company an estimated $500m) [29].

It is however clear that despite the impressive results, the road to application of formal techniques as a routine step within the system development process is quite a long and bumpy one. To attack the problems along this road research work in two directions is needed. There is a need for basic research in formal languages, data structures and algorithms for specification and reasoning about complex embedded systems. There is also a requirement to study the needs of application domain engineers, the suitable models and abstractions in different domains (e.g. telecommunications, aerospace, electronic design, etc.) and to obtain suitable interfaces accessible to non formal method experts. The key to success is to get a good synergy between both of these tracks of research.

The first phase of the Cohsy project was started along the second track. Some needs were identified [8] and a number of modelling abstractions were investigated [23]. One major activity was modelling, carried out in the context of the landing gear and hydraulic subsystems of JAS 39 Gripen, and performed in cooperation with Arne Jansson from the mechanical engineering department (see also chapter **??** in this volume). A further activity has centered around two issues: the application of state of the art formal verification techniques and the integration of formal methods in the system development process using

[1]Currently at the Dept. of Signals sensors and systems, Royal Institute of Technology, S-100 44 Stockholm, Sweden.

industrial-strength development environments.

This chapter provides an overview of these three tracks of activity. A longer report on the work performed is available as a technical report and through the www [23]. Also the work has been documented and presented at international conferences [21, 22, 32, 34, 31] the proceedings of which the interested reader is referred to for further reading. In what follows, we present a brief overview.

0.1 Hybrid modelling and verification

0.1.1 Architectural models

Our main thesis is that verifying properties of computer based systems benefits from modelling both the embedded controller and its surrounding physical environment. The models of the hardware and the software are typically fundamentally different in their character. Hence, different competences are required in development and documentation of these models. However, it is the composition of the models for these two parts, also referred to as the *closed loop* model, which is needed for proving the properties we are interested in. We employ architectural models to make the interface between models of different character explicit. For example, in the landing gear system, we start by defining the interfaces between the mechanical gear and door actuation systems, the hydraulic power supply system, the hydraulic diagnosis system, and the gear control system. The architectural model for top-level decomposition of hybrid systems [18, 20] provides a modular framework allowing to plug in a refined version of an arbitrary module without having to consider (or redo) the rest of the model.

0.1.2 From engineering documents to mathematical models

Another observation is that unless care is taken in developing the mathematical models which form the basis of formal verification, the results of verification can not be relied upon. Therefore, rather than verifying properties of ad hoc models we take a number of steps for systematically deriving the models from engineering design documents.

Deriving mathematical models from engineering documents, is an inherently difficult task. We have approached this problem by adopting an iterative process in which models are successively refined as a result of earlier analysis. The level of detail in the model is dependent on the requirement specification, assumptions made about the physical components, and the choice of verification technique. For this kind of iterative process to be realistic, the formal models derived and the assumptions made must be easy to convey to the domain specialists.

The above aspects necessitate the employment of a range of formalisms and methodologies in our application. At the physical level of abstraction we have ideal physical model diagrams and switched bond graphs [33]. At the mathematical level of abstraction, we have employed hybrid transition systems (HTS) [19], and to perform various verification techniques depending on the characteristics of the models, we have used specific mathematical languages such as Linear Hybrid Auatomata (LHA) [2] with the tool Hytech, and Extended Duration Calculus (EDC) [5] with its proof system.

To begin with, models for the hydro-mechanical subsystems (in terms of schematic engineering diagrams) were developed. These models have been transformed to the energy-based graphical language of switched bond graphs [32, 31]. The latter step significantly aids the derivation of mathematical models for mode-switching physical systems. These are physical systems which can not be described by one set of differential and algebraic equations (DAEs); rather different sets of DAEs are needed depending on the discrete mode in the system. Next, models for diagnosis and controller modules (to be implemented in hardware or software) were developed.

The modelling phase was followed by proofs of the following three properties for several versions of landing gear models.

- The door and the gear do not collide under movement
- Whenever the pilot issues the landing (airborne) command, the gear will be out (in) and the door closed within T seconds.

0.1.3 Symbolic model checking of hybrid models

Within algorithmic techniques for hybrid models a specific subclass of models dominate the scene. These are hybrid systems in which the continuous variables change at a linear rate of time. A common technique for verification is symbolic model checking, originally developed for purely discrete models combining model checking [6] with representations in binary decision diagrams (BDD) [4, 17]. Nicollin et al. and Alur et al. extend this method for proving properties of timed and hybrid stystems [24, 2]. Linear hybrid automata is an example formalism [1] in which symbolic model checking can be used as a verification technique.

This method works on partitions of the infinite state space of the system. Symbolic manipulation is performed on finite descriptions (polyhedra) of infinite sets of points in the n-dimensional state space. More precisely, the set of reachable states is computed symbolically by iterative computations of the fixed point of a function, whereby the unreachability of unsafe states can be proved if the fixed point computation terminates. The problem is in general undecidable for hybrid systems (even with the restriction to LHA), but there are suggestions that computations terminate for most practical cases.

The symbolic model checking tool that we studied was the 1994 version of HyTech; a prototype implemented by Henzinger et. al. The tool accepts a model of the hybrid system and its surrounding environment (the pilot) in a restricted form of LHA. It also requires a set of "bad" states characterized as a boolean expression over the continuous state variables. In our example bad states are those in which the the door is not open and the gear is moving.

We applied this verification technique to a hybrid model of the landing gear where the model of the hydraulics and mechanics was based on an assumption of constant pressure supply (leading to time linear differential equations).

However, we used a dynamic controller which incorporated deterministic timing constraints on discrete transitions, corresponding to communication and computation delays [22].

The hybrid transition model of the landing gear was then transformed to the syntax accepted by HyTech. This involved several straight forward but tedious transformations, mainly due to not accepting non-convex guards and invariances. The alternative pilot models were provided as different LHA – for example, a pilot which changes its landing command every 5 seconds.

Our experiments show the viability of the technique for applications with subcomponents which can be assumed to have time-linear evolutions. The bulk of the work was, however, in justifying this assumption by the physical modelling activity, and presenting the exact hardware assumptions under which this verification result makes sense.

0.1.4 Logical proofs for non-linear hybrid models

In the next iteration over the physical modelling activity we obtained a model which is a non-linear system of DAE. This corresponds to a mathematical model for a refined version of the hydraulic supply system in which the hydralic pressure is not assumed to be constant as in the first case. Rather, it is assumed to be regulated by a hydro-mechanic pump [32, 31].

In this more detailed model of the power supply system we make an explicit model of the hydro-mechanic regulator in the bond graph model; this regulator attempts to make the pressure 'as constant as possible' given a particular engine velocity ω. In this case the overall bond graph converts to a non-linear dynamic system

$$\begin{aligned}
\dot{x}_d &= \alpha'_d \, p \, u_d, \quad x_d \in [0,1] \\
\dot{x}_g &= \alpha'_g \, p \, u_g, \quad x_g \in [0,1] \\
\dot{p} &= -(\gamma + \gamma_d \, |u_d| + \gamma_g \, |u_g|) \, p + \beta
\end{aligned}$$

where $\alpha'_d, \alpha'_g, \gamma_d, \gamma_g > 0$ are constants and p is the hydraulic supply pressure. The quotient β/γ is a monotonic function of the aircraft engine velocity ω.

Obviuously, symbolic model checking can no longer be applied to the closed loop model incorporating the new set of DAEs. To prove the closed loop properties of a system based on this model we have adopted a deductive approach to verification of hybrid models. The approach builds on representation of the closed loop model and the required properties in a variant of interval temporal logic called *Extended Duration Calculus* (EDC). Then verifying that the requirements are met by the given model are achieved through proving theorems in the proof system of the logic.

We use the version of EDC in accordance with [28] in which a mathematical theory about state functions is assumed as given. The syntactic constituents of the language are *state names* ($f, g, \dot{f}, \dot{g}$, etc) denoting a real valued function on time, *boolean state names* denoting a boolean valued function on time, operators and relations on real numbers, and propositional logic operators, resulting in state expressions and state assertions.

For example, in the landing gear example, state names $x_d, x_g, \dot{x}_d, \dot{x}_g$, and boolean state names *open* and *closed*, as well as the other boolean state names are constituents in the mathematical theory over which assertions can be made. Furthermore, EDC uses the notions of 'length of an interval' and 'duration'. The length of an interval is equal to the duration of the constant boolean state *true* (which has a value 1 at any time), expressed as $\ell = \int true$. Furthermore, for Q being an assertion, the notion of Q holding over an interval (represented by $\lceil Q \rceil$) is related to durations by the following definition:

$$\lceil Q \rceil \widehat{=} (\textstyle\int Q = \ell) \wedge (\ell > 0)$$

Informally speaking, state assertions are evaluated to 0 or 1 whereby their durations can be reasoned about. In particular, $true$ and $false$ as state assertions evaluate to 1 and 0 respectively.

Atomic duration formulas have the form $\lceil Q \rceil$ or $\triangleleft(dt_1, \ldots, dt_n)$ where $\triangleleft$ is an n-ary relation on reals and dt_i are duration terms. Duration terms have essentially the form $\int se$ (for state expression se), $\mathbf{b}.se$ and $\mathbf{e}.se$ denoting the value of the state expression se at the beginning and end of an interval respectively, and terms built with operators on reals. Compound duration formulas are build from atomic formulas using the usual logical connectives as well as the chop connective of interval tempotal logic (denoted by ;) which chops an interval in two parts. Formulas are interpreted over an interval [b,e] given an interpretation for the state names and operators in the language.

The semantics of the chop operator can be informally stated as follows. Let D_1 and D_2 be duration formulas. For a given interpretation, we say that $D_1 \,;\, D_2$ is true over an interval $[b, e]$ if and only if there exists an m such that D_1 is true over $[b, m]$ and D_2 is true over $[m, e]$. The interpretation of ℓ in an interval $[b, e]$ corresponds to the length of the interval and equals $b - e$.

The requirements specifications presented earlier can then be presented in the standard form for invariance properties within EDC. That is, the property "never P" is presented by $\Box \neg \lceil P \rceil$ which means that for every sub-interval it is not the case that P holds during the interval[2]. Let's consider the safety property that the door and gear do not collide in operation. The property is proved by proving the stronger property that the gear must never move when the door is not fully open. This requirement R_1 may be formalized as

$$R_1 \quad \equiv \quad \Box \neg \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil$$

where x_d and x_g represent the position of the door and the gear respectively, and x_d varies between 0 (fully closed) and 1 (fully open).

The second performance (or timeliness) property requires that the extension (or retraction) to be completed within T time units, is formalized by the two formulae R_2 and R_3. Let $r > 0$ (resp. $r \leq 0$) denote that extention (resp. retraction) command has been given by the pilot. Then

[2] $\Box$ corresponds to the modal logic operator Always which in DC is defined using chop and negation. It means "during every subinterval of this interval".

$$R_2 \equiv \Box\neg((\lceil r > 0 \rceil \wedge \ell = T) \; ; \lceil \neg(x_g \geq 1 \wedge x_d \leq 0)\rceil)$$
$$R_3 \equiv \Box\neg((\lceil r \leq 0 \rceil \wedge \ell = T) \; ; \lceil \neg(x_g \leq 0 \wedge x_d \leq 0)\rceil)$$

In other words, it is never the case that the extention (retraction) command is in operation for an interval of length T and the desired effect has not been achieved in the immediately following interval. In order to verify these properties we thus have to prove that the model of the closed loop system logically implies the properties, i.e. in any interval that S holds a conjunction of R_1, R_2 and R_3 holds.

$$\lceil S \rceil \Rightarrow R_1 \wedge R_2 \wedge R_3$$

Two alternative closed loop models were verified: both consisting of a static controller but differing on plant models (one time-linear and the other non-linear). The proofs of the properties of the closed loop model were obtained using a few of the 107 laws for DC as presented in [28]. The proof strategy was usually based on contradiction (e.g. showing that $\lceil S \rceil \wedge \neg R_1 \Rightarrow false$).

The following observations sum up the experience from the application of the method in the case study [23]. The EDC formulation of the closed loop model, using the architectural decomposition and the mathematical models for each component, were easy to obtain. Moreover, switching from plant model to another plant model simply amounts to changing one conjunct in the formula representing S (the closed loop system) with another (representing the refined plant model).

The EDC proofs of course require a general familiarity with constructing logical proofs. However, having this familiarity, the use of EDC proof system was not particularly difficult. Nevertheless, since similar proof tactics (contradiction, case analysis, etc) tend to appear in different applications, the use of mechanized proof assistants is recommendable. Thus, the next step for making this type of analysis worthwhile in a larger example is the use of theorem provers. A mechanical prover for Duration Calulus implemented on top of PVS [25] exists [30] and attempts in mechanizing the proofs of the landing gear properties are in progress.

The overall proof strategy was based on a derived region-based transition system with invariances at each discrete mode. This resembles a hybrid automaton in which taking an enabled discrete transition is not optional but progress is enforced. A reflection over the work performed is the close connection between the region based transition system and the structure of the proof. In contrast with much of the literature on formal verification we do not assume that the structures over which proofs are performed exist apriori[3]. In the EDC

[3]HyTech could for example analyze the same transition system in presence of the coarse (time-linear) model of the plant thereby eliminating the need for the hand proof in this case. However, our study illustrates that when combining the time-linear plant with the static controller, the very derivation of the hybrid automaton from the architectural model (corresponding to the proof structure in EDC) is the main issue – once a satisfactory model is found the proof work is more or less routine.

approach, however, the insights gained while verifying the coarse model could later be reused for verifying the non-linear refinement. This confirms that modelling remains a cornerstone of the verification activities in the hybrid domain.

0.2 Verification of discrete models: Esterel

A common approach to modelling and verification of discrete control systems is the discrete events approach, whereby the system is represented as parallel composition of a set of finite automata. The requirements may be represented according to different preferences (e.g. as formulas in a temporal logic or as finite automata). Although the theory for synthesis of discrete controllers has existed for a while [27], its application to verification requires obtaining natural discrete models of the plant, and practical automatic verification tools.

In this section we report on an application of these ideas within the development environment of Esterel/Mauto – a language falling in the framework of synchronous languages[10]. This family of languages have the benefit of a formal semantics (as synchronous I/O machines or Mealy automata) and an intuitive appeal within the engineering community. One may question why is formal verification in such a setting worth some attention when recent advances in verification techniques are within the timed and hybrid domains? One argument is the attractiveness of code generation capabilities in the development environments for this class of languages. Once the model of a controller in interaction with a plant has been formally verified for certain properties, then the high level controller representation can be automatically translated to C, Ada or a circuit design language. Furthermore, models in Esterel and statecharts with the above underlying semantics can also be bridged to models in the data flow languages of Lustre and Signal (with the associated commercial tools, e.g. SAO+).

Our application of these techniques to the landing gear example rests on two basic ideas. First, that realistic discrete models of the mechanics and hydraulics can be developed once the physically driven models are available and well understood. Second, that the safety requirements can easily be encoded as synchronous observers [11] of the plant and the controller. An observer is a reactive state machine which "watches" the emitted signals of another machine. If a particular safety property is violated the observer emits an alarm signal. Formal verification amounts to checking that a parallel composition of the observer and the system does not emit alarms.

This approach suits verifying safety properties (i.e. "something bad will not happen"). For bounded response properties (i.e. "something good will happen within x steps") formulas of temporal logic are easier to specify. Though there exists a system which gives an automatic translation of such a property in temporal logic to a synchronous observer [15].

In [34] we report on whether the proof of a property of the system should be proved on a model of the composition of its subsystems, or that it should be split into different properties of the subsystems and proved separately for each subsystem. The study of the compositional versus decompositional proof techniques in the case of tightly interacting systems is exemplified by providing the numbers of states and transitions (in our case related to the size of proofs and intermediate proof steps) in both compositional and decompositional methods.

In the application of the observer based verification technique we have modelled the door, the gear, and the controller each as separate Esterel programs [34]. Communication between the subsystems is through synchronous broadcast of shared events (signals). The supporting environment can be used to translate the textual code to systems of parallel machines, and if required, to a graphical representation via Mauto.

The property that we are interested in verifying is that the door should always be open and stationary when the landing gear is in motion. To represent this property in a discrete setting, we treat it as a composition of two properties, that the door and gear are never moving at the same time, and that the door is always open when the gear is in motion. In our discrete models of the door and the gear we therefore include states from which these subsystems are able to report whether they are moving or stationary. Notice that both properties, are properties of the composed system, and do not hold over neither the plant nor the controller.

These two properties can be checked by a pair of observer machines combined in parallel. The first simply emits an alarm (signal) and halts whenever an event occurs containing reports of the door and gear moving at the same time. The second machine watches the sensor states from the plant, in aparticular, whether the door is open or closed. It emits an alarm if the gear-moving signal ever occurs when the sensors have indicated that the door is not open.

To check that the system satisfies the safety property in the compositional style, the complete observer was combined in parallel with the system, and the resulting state machine checked for the emission of alarm signals. In [34] we also present the comparative search spaces and the methodological issues if the decompositional style to verification is adopted.

0.3 Semi-formal industrial tools

So far, the success stories of formal methods have mostly been in the areas of design specification and verification. The report of the formal methods ACM working group on strategic directions in computing research puts forward several future directions in which formal methods can complement non-formal methods used in the development process. The report specifically mentions requirements analysis (nailing down unclear customer requirements), refinement (synthesizing a lower level specification from a high level one), and testing (use of formal specifications for generating test suites). An existing example of such an integration is the case of Display Information System of the London air traffic control system mentioned earlier. Another visible trend is the attempt to integrate the informal structured design languages (SA/RT, Ward & Mellor diagrams) with formal languages like Z and the operational semantics for graphical languages with formal sematics , e.g. fragments of statecharts (for example [26]).

In this project we were primarily interested in investigating formal verification techniques in realistic settings and therefore commercial tool support was a point for initial study [8]. A primitive technique for detecting violation of safety properties uses systematic and exhaustive search of the state space. This method is obviously not optimal for large systems but it is unfortunately the only verification technique available in commercial environments for high level design descriptions of complex systems[4].

One such specification tool is Statemate which supports development of high level hierarchical specifications and code generation from statecharts. The "dynamic check" facility in the tool Statemate falls into the above category and should be seen as an organized simulation of a particular design description until the proof of violation of a certain property is obtained. The exploration of the state space follows all possible execution branches according to the version of statecharts semantics implemented in Statemate [13].

Note that considering Statemate as a semi-formal software engineering tool is due to the fact that the formal semantics exists for a fragment of the features supported by the tool. For example, the different model views, i.e. the dataflow like language of activity charts and the architectural view provided by module charts, have not been mathematically connected to the statecharts language. Also, formal semantics has only been provided for the pure event-based fraction of statecharts although state variables with value domains ranging over reals or integers are also supported by the tool. The latter feature was exploited in representing the closed loop model for the landing gear in order to stay as close to the hybrid mathematical model as possible[5]. We found that modelling controllers with complex control structures is not a problem but continuous plant models can have to be represented as discrete time (difference) equations.

To model the requirements specifications we adopted the observer based methodology described earlier. Using the discrete time steps underlying the semantics of Statemate we have a means of representing the timeliness property in terms of the clock cycles of the system. To check each requirement we devise a watchdog which observes the variables of all the system components and takes the transition from a safe to an alarm state whenever the required property is violated.

0.4 Concluding remarks

We have presented a short overview of state of the art physical and mathematical modelling and verification techniques in the first phase of the Cohsy project. More details of the landing gear models consisting of mechanical, hydraulical, electronic and software components can be found in [23].

We initiated the study at the informal engineering documents describing the physical components, derived bond graph and mathematical models for the

[4]We classify system specifications in propositional logic, as supported by NP Tools (Logikkonsult AB) for example, as low level descriptions into which higher level system specifications may well be transformed eventually.

[5]Though full formal semantics of the language is a topic for active research. So is the translation of statechart like languages to boolean equations or BDDs.

environment systematically, and then combined them with the mathematical model of the discrete controller(s).

We used a generic architecture to derive alternative closed loop models based on alternative plant and controller models. The plant models ranged from simple time-linear models to non-linear models allowing variable hydraulic pressure.

We have illustrated different formal verification techniques, ranging from state space search in Statemate to hand proofs in Extended Duration Calculus to arrive at proofs of safety and timeliness properties of the different closed loop models. While symbolic model checking techniques could be used for verifying complex timed dynamic controllers with a simple model of the environment, the proof with the next (refined) model of the environment necessitated deductive reasoning in a real-time temporal logic (EDC).

The experience with the different modelling and verification techniques confirms that a principled derivation of the models remains a cornerstone of the verification activities. For smaller systems it is possible to manually translate between model types while preserving their essential properties; but applications in an industrial setting require more support tools for derivation and management of models.

Bibliography

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P-H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In Grossman et al. [9], pages 209–229.

[2] R. Alur, T.A. Henzinger, and P-H. Ho. Automatic Symbolic Verification of Embedded Systems. In *Proc. of 14th Annual Real Time Systems Symposium*, pages 2–11. Computer Society Press, 1993.

[3] R. Boyer and Y. Yu. Automated Proofs of Object Code for a Widely used Microprocessor. *Journal of the ACM*, 43(1):166–192, 1996.

[4] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proc. IEEE Conf. on Logic in Computer Science*, pages 428–439. Computer Society Press, 1990.

[5] Z. Chaochen, A. P. Ravn, and M. R. Hansen. An Extended Duration Calculus for Hybrid Real-Time Systems. In Grossman et al. [9], pages 36–59.

[6] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986. An early version of the paper appeared at the 10th ACM symposium on principles of programming languages, 1983.

[7] M. Croxford and J. Sutton. Breaking through the V and V bottleneck. In *In proceedings of Ada in Europe*. Springer Verlag, 1995.

[8] R. Germundsson, J. Gunnarsson, A. Jansson, P. Krus, M. Morin, S. Nadjm-Tehrani, J. Plantin, M. Sethson, and J-E. Strömberg. Complex Hybrid Systems I, A Study of Available tools and Specification of Planned Work. Technical Report LITH-IDA-R-94-29, Dept. of Computer and Information Science, Linköping University, September 1994.

[9] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Proc. Workshop on Theory of Hybrid Systems, October 1992, LNCS 736*, Lyngby, Denmark, 1993. Springer Verlag.

[10] N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.

[11] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Third Int. Conf. on Algebraic Methodology and Software Technology, AMAST'93*, Twente, June 1993. Workshops in Computing, Springer Verlag.

[12] A. Hall. Using Formal Methods to develop an ATC Information System. *IEEE Software*, 12(6):66–76, 1996.

[13] D. Harel and A. Naamad. The Statemate Semantics of Statecharts. *ACM Transactions on Software Engineering Methods*, 1996.

[14] M. Heimdahl and N. Leveson. Completeness and Consistency in Heirarchical State-based Requirements. *IEEE transactions on Software Engineering*, 22(6):363–377, June 1996.

[15] L.J. Jagadeesan, C. Puchol, and J.E. Von Olnhausen. A Formal Approach to Reactive Systems Software: A Telecommunications Application in Esterel. *Formal Methods in System Design*, 8:123–151, 1996.

[16] 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.* Springer Verlag, 1994.

[17] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[18] S. Nadjm-Tehrani. *Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification.* PhD thesis, Dept. of Computer and Information Science, Linköping University, March 1994. Dissertation No. 338.

[19] S. Nadjm-Tehrani. Reactive Systems in Physical Environments. In *Hybrid Problems, Hybrid Solutions – proc. of the 10th. Biennial Conference on AI and Cognitive Science*, pages 169–180. IOS Press, April 1995.

[20] S. Nadjm-Tehrani and J-E. Strömberg. From Physical modelling to Compositional models of Hybrid Systems. In Langmaack et al. [16], pages 583–604.

[21] S. Nadjm-Tehrani and J-E. Strömberg. Modelling and Formal Analysis of an Aircraft Landing Gear System. In *Proc. of the 2nd. European workshop on Real-time and Hybrid Systems*, pages 239–246. VERIMAG Laboratory, Grenoble, May 1995.

[22] 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.

[23] S. Nadjm-Tehrani and J-E. Strömberg. JAS-95 Lite: Modeling and Formal Analysis of Dynamic Properties. Technical Report LITH-IDA-R-96-X, Dept. of Computer and Information Science, Linköping University, 1996. To appear, currently available from `http://www.ida.liu.se/~snt/activities.html`.

[24] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An Approach to Description and Analysis of Hybrid Systems. In Grossman et al. [9], pages 149–178.

[25] N. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *In Proc. 11th International Conference on Automated Deduction, LNCS 607*. Springer Verlag, 1992.

[26] C. Petersohn, C. Huizing, J. Peleska, and W.-P. de Roever. Formal Semantics for Ward and Mellor's TRANSFORMATION SCHEMA and the Specification of Fault-Tolerant Systems. In *Proc. of the first European conference on Dependable Computing, LNCS 852*, pages 59–76. Springer Verlag, 1994.

[27] P.J.G. Ramadge and W.M. Wonham. The Control of Discrete Event Systems. *Proceedings of the IEEE*, 77:81–97, 1989.

[28] A.P. Ravn. Design of Embedded Real-time Computing Systems. Technical Report ID-TR:1995-170, Dept. of Computer Science, Technical University of Denmark, October 1995.

[29] H. Ruess, N. Shankar, and M. Srivas. Modular Verification of SRT division. In *In proceedings of the International Conference on Computer Aided Verification, CAV'96, LNCS 1102*, pages 123–134. Springer Verlag, 1996.

[30] J. U. Skakkebaek and N. Shankar. Towards a Duration Calculus Proof Assistant in PVS. In Langmaack et al. [16], pages 660–679.

[31] 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.

[32] J.-E. Strömberg, S. Nadjm-Tehrani, and J. Top. Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems. In *Proc. of the DIMACS International Workshop on Verification and Control of Hybrid Systems*. Springer Verlag, 1996. LNCS 1066.

[33] J.E. Strömberg. *A mode switching modelling philosophy*. PhD thesis, Linköping University, Linköping, 1994. Dissertation no. 353.

[34] 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*, pages 405–419. Springer Verlag, September 1996. LNCS 1135.