Project 22703: SYRF

Deliverable 7.2

Hybrid Systems

---

1 Introduction

This work package aims to extend the applicability of formal verification techniques to properties *arising from the interaction of a control program and a physical system.* In deliverable 7.1 for the project we described a framework within which we address this issue. That is, given a controller in some synchronous language, we set out the following steps as necessary for formal verification of the closed loop properties.

1. Identifying and formalising the properties to prove – i.e. those critical to correct functioning of the controlled system based on informal requirements specifications.
2. Modelling the physical environment.
3. Applying abstraction techniques to obtain "expressive enough" mathematical models.
4. Composing the controller and environment models to obtain the model of the closed loop system.
5. Applying algorithmic or deductive methods for verifying the properties specified under 1.

During the first year of the project we have also reported on a case study elicited from Saab Aerospace. This example was then used to start step 1, and perform step 2 in the above list. The physical model in terms of differential equations and the underlying assumptions can be found in Annex 7.1.A to the year 1 deliverables [9].

This report concentrates on abstraction techniques for step 3. Making som assumptions on the system explicit allows further formalisation of the requirements. That is, for two of the original requirements we provide sufficient conditions for (or refinements of) the requirements to hold.

Once a mathematical model of the physical environment is available, two different strategies can be adopted based on applying steps 3 and 4 *in different orders.* Since the area of hybrid systems verification is highly under-developed, we are open to exploration of both strategies in the project.

In the first year of the project we attacked an approach whereby step 4 precedes step 3. That is, a general model of hybrid systems adapted for synchronous programs was presented (Time- deterministic Hybrid Transition Systems, TD-HTS) and some compositionality properties were proved [6]. Also, some abstraction techniques for approximating linear hybrid automata in order to obtain decidability were presented [2].

During the second year of the project, we have studied the modelling implications for performing step 3 before step 4. This means that a model of a physical system is first "simplified" and then composed with the synchronous controller to obtain a model of the closed loop system for the purpose of analysis. Various techniques for deriving these simplified models in presence of various assumptions, is thus the main topic of this report.

The report is organised as follows. In section 2 we present the main problem with transforming engineering models for a physical environment to a model analysable in conjunction with a controller program. Three possible restrictions on unmodelled inputs are then proposed, contributing to three different abstractions. These are presented in section 3. In particular, in section 3.1 we take one such model and relate it to mode-automata. This facilitates obtaining a closed loop system as a composition of mode-automata under step 4 of our framework. In section 4 we present further formalisation of the requirements specifications for the air control case study. Finally, sections 5 and 6 present related and on-going works.

2 From HTS to simpler models

A hybrid transition system (HTS) describes the continuous change in a system by associating a differential or algebraic equation with each *state* variable. The change in state is thus given in terms of a function over *inputs* and other system variables. Therefore, as long as the set of input variables is not empty, a HTS describes an open system. This conforms with the engineering practice and allows modular development of models for parts of systems.

The inputs to a physical system normally consist of one or more control signals to be specified by a controller, and possibly some unmodelled inputs from the physical world (operator signals or disturbances). Also, it should be noted that the equations may specify state in a non-linear relation to inputs (see e.g. the air control case study presented in section 3).

To get a mathematical model of a closed loop system it is straight forward to apply the parallel composition operator on HTSs. However, even in presence of restriction to (timed) synchronous controllers (giving a TD-HTS), the formal analysis of the closed loop model will be non-trivial in presence of non-linearity – both in time and in other state variables. Algorithmic approaches to analysis of hybrid models are restricted to evolutions which are linear in time, in so-called *linear* hybrid automata (HA) [1], and deductive proofs require stating continuous analysis results in hybrid logics [7].

It is therefore desirable to "simplify" an HTS model of the environment before combining with a controller model. One way of simplification is to put some restrictions on inputs to the model while not completely specifying it. Note that we do not want to fully describe the inputs to an environment model. In that case, we would have fully specified the controller! We want the possibility of composing the environment model with different controller models (during the development process). At the same time, we want to restrict the inputs to the environment to specific classes of signals.

2.1 Classification of input restrictions

Let $H = \langle M, X, U, F, I, T \rangle$ be a model of a physical environment[1]. For the mode $m \in M$, let $F(m, x) = \ulcorner \dot{x} = f(\bar{x}, \bar{u}) \urcorner$ define the evolution of the continuous state variable $x \in X_c \subseteq X$.

For every input variable $u \in U$ we consider the following restrictions on the time function γ_u associated with u:

(1) γ_u is piecewise constant with a finite range of values

(2) γ_u has a piecewise constant slope

The motivation for the restrictions is as follows. For any u which is the state variable of another physical system, after an application of the parallel composition operator, it is eliminated as an input and associated with a differential equation. Those u which are control signals can be assumed to be piece-wise constant. Others, represent unmodelled parts of the environment, and can take any of these two forms.

We can now assume further (orthogonal) restrictions on inputs under class (1) as follows:

(1.a) changes in γ_u may appear at equidistance points of time

(1.b) γ_u is constant throughout a run

A fourth class of restrictions is obtained by placing a third restriction on class (1) of inputs. To introduce this restriction we first present the notion of *input acceptance restriction.*

Definition 1 *An HTS H is defined to have the* input acceptance restriction *property iff its set of runs are restricted to those in which the input variables take finitely many values staying constant between any two time points t_i and t_{i+1} appearing in positions σ_i and σ_{i+1} of the run, and for none of the transitions of the system the guard condition includes the input variables.*

This is an adaptation of the type of models treated by Lemmon and Antsaklis [4], recast in the context of HTS with the operational semantics defined in terms of a set of runs [6]. In systems with input acceptance restriction, the system accepts new input only at time points when some border in a (finitely) partitioned state space is crossed. At the start of the system, any one of the finitely many constant inputs may be chosen. The state of the system then varies in accordance with the given (instantiated) differential equations until any of the "significant" regions in the state space are hit, at which point a change in input can be accepted. For each possible new input the system has a different possible run from this point.

This restriction is thus both a restriction on the system itself (in terms of restriction of guards, and a finite number of significant regions), and the type of inputs accepted by the system (piecewise constant, changing at points determined by the system). We specify this restriction as a third category under class (1):

[1] For syntax and operational semantics of HTS refer to [6] or Annex 7.1.B in year 1 deliverables of SYRF [9].

(1.c) changes in γ_u are compatible with system H having the input acceptance restriction property

In the original plan to the project the idea was to use this restriction to arrive at an abstraction of the physical model which preserves all the reachability properties.

The next section discusses the applicability of the above restrictions in our case study. When applicable, these can be used to systematically obtain a "simpler" HTS; in the sense that each mode is replaced by a number of modes with simpler dynamics. Thus, the transformed system is easier to analyse locally, e.g. by stating invariances determined through extreme-case simulations. These transformations will be illustrated on the air control case study.

3 The air control case study

Consider the physical model developed for the climatic chamber case study in the first year of the project [9]. The hybrid system representing this model, as derived from engineering models, has one mode and one set of differential equations under those initial assumptions. The differential equation, in which u_i are inputs, x is the only state variable, and a, b and c are constants, has the following form:

$$\dot{x} = au_1x + bu_2 + cu_1u_3$$

In general, there may be more than one mode in a physical system ($|M| > 1$) due to a change in the dimension of the system. Here, with no loss in generality, we assume an HTS with one mode, since the forthcoming "simplification" steps will map every mode to a number of new modes with a simpler dynamics in each mode.

3.1 The discrete-time model

Here we assume that inputs are restricted to be of type (1), but to restrict further, we need to exploit our knowledge about the application characteristics.

Our knowledge about the design of the controller indicates that the voltage output by the controller may change even during intervals in which the chamber temperature remains within significant regions. The requirements/design document refers to "within Δ of the reference temperature" or "within 2Δ of the reference temperature", but the voltage might be adjusted upwards or downwards while the temperature remains within these intervals. Hence, restriction 1(c) does not apply to voltage (the square of which is represented by u_2 here).

As far as other inputs are concerned, as a first approximation it is reasonable to assume that the flow is constant at all times: u_1 is restricted to type 1(b). We further assume that the incoming air temperature u_3 is piecewise constant with a finite range of values, but no other assumptions (so, it is of type 1). Then the model of the system can be transformed to an HTS with one mode for every possible value of u_3. This analysis gives us Figure 1 as the first approximation.

Next, we assume that the heater is controlled by a synchronous program placing the restriction 1(a) on the incoming voltage u. This makes the input u_2 to the chamber model

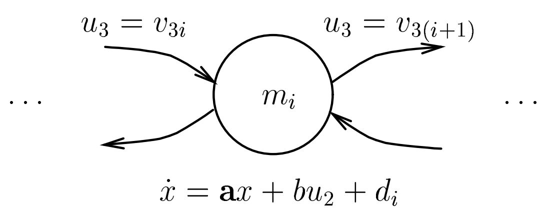

Figure 1: Assuming that domain of $u_3 = \{v_{31}, \ldots, v_{3n}\}, \mathbf{a} = au_1$, and $d_i = cu_1 v_{3i}$

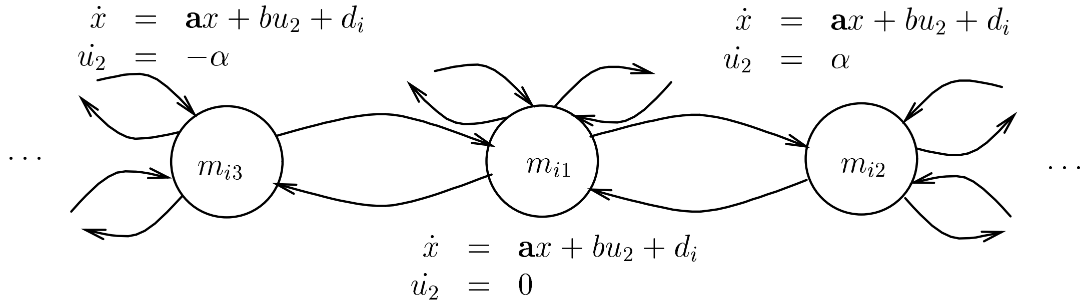

Figure 2: An HTS model with u_2 as a state variable: assuming that it may stay constant, or increase/decrease at a constant rate.

also of type 1(a). After this assumption, one can rewrite the differential equations into a discrete-time form by making the sampling interval T a parameter of the model. Thus, every differential equation in Figure 1 may be replaced by the difference equation:

$$x((k+1)T) = x(kT)e^{\mathbf{a}T} + b/\mathbf{a}(e^{\mathbf{a}T} - 1)\; u_2(kT) + d_i/\mathbf{a}(e^{\mathbf{a}T} - 1)$$

That is, the $(k+1)$th value of x is defined in terms of the kth value of x and the kth value of u_2 (which is assumed constant during the interval $[kT, (k+1)T]$). This reduces the chamber model to a mode-automaton which is a hierachical model compilable to a Lustre program [5]. The syntax and semantics of mode-automata can also be found in the year 1 deliverable 2.1 of SYRF [8].

3.2 The HA model

A totally different class of models is obtained if some inputs to a physical model are assumed to be restricted to type 2. In this section we take the model of Figure 1 and restrict u_2 to be of type 2. The motivation being as follows: we do not want to make exact assumptions on controller actions, but it is reasonable to assume that the controller has three types of behaviours. It may keep the voltage constant, it may increase it, or it may decrease it. This assumption leads to a model of the chamber whereby every mode in Figure 1 will be replaced by three modes as displayed in Figure 2. The figure shows the obtained hybrid automaton fragment, where the incoming and outgoing transitions from the fragment are left out for clarity.

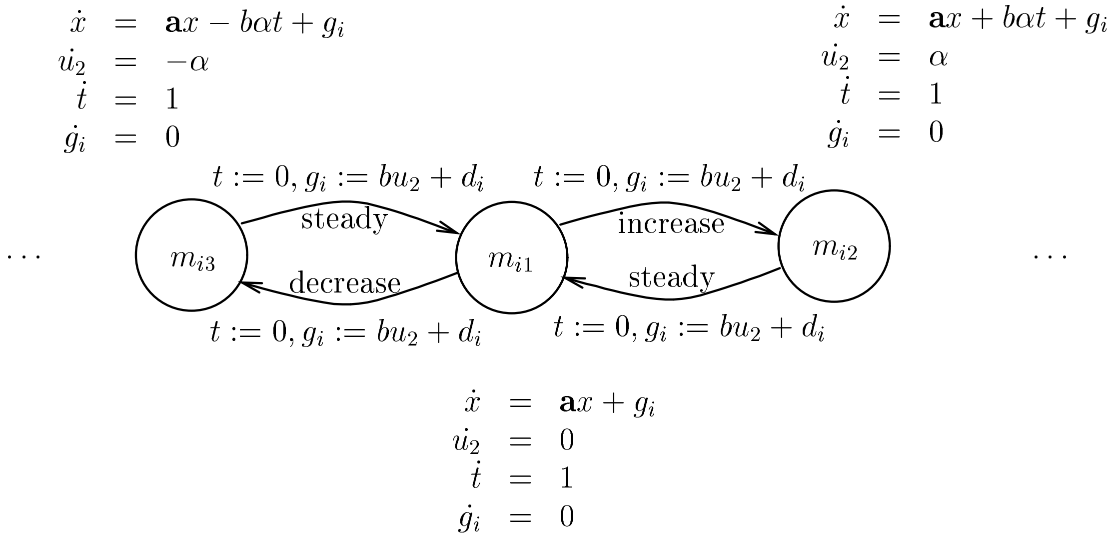

Figure 3: Fragment of a hybrid automaton model with the same assumptions as in Figure 2 – the clock t and the piece-wise constant function $g_i = bu_2 + d_i$ have been added to relate the changes in x to u_2.

It should be clear that by specifically stating the rate of change for u_2 this variable can no longer be considered as an input variable in the HTS. In order to utilise the added knowledge for simplifying the equation for x, we need to relate rate of change of u_2 with the changes in x. Thus, we need to explicitly represent a clock which measures how long the system has resided in each mode since the last time it was entered. We use the clock t for this purpose. This variable has to be reset to zero every time a mode is entered. In HTS it is possible to do this by adding a mode which is instantanuously entered and left for each mode of the system. In hybrid automata this is captured in a more compact way by allowing assignments on the transitions. Furthermore, we need to establish the condition for leaving a mode and entering a new one, which obviously arises due to actions of the controller. In HTS we would reflect this by adding a discrete (boolean) variable. In HA, this is achieved by adding synchronisation labels corresponding to controller actions. The HA in Figure 3 allows arbitrary changes of slope within the range $\{-\alpha, 0, \alpha\}$ by the controller. Note that the value of x is now dependent on the (apriori unknown) value of u_2 on entry to the mode. This value on entry is captured by adding a piece-wise constant variable (g_i) and an assignment at each mode change.

It can be observed that the obtained hybrid automaton still is not analysable algorithmically. That is, it is not yet a *linear* hybrid automaton (x does not vary with constant slope). To make this model analysable using the Polka tool, we need to add bounds on the evolution of x by performing local simulations in each mode. So far, simulation models of the differential equations have been built in the MatrixX-SystemBuild environment. Deriving the limiting constant slopes is work in progress.

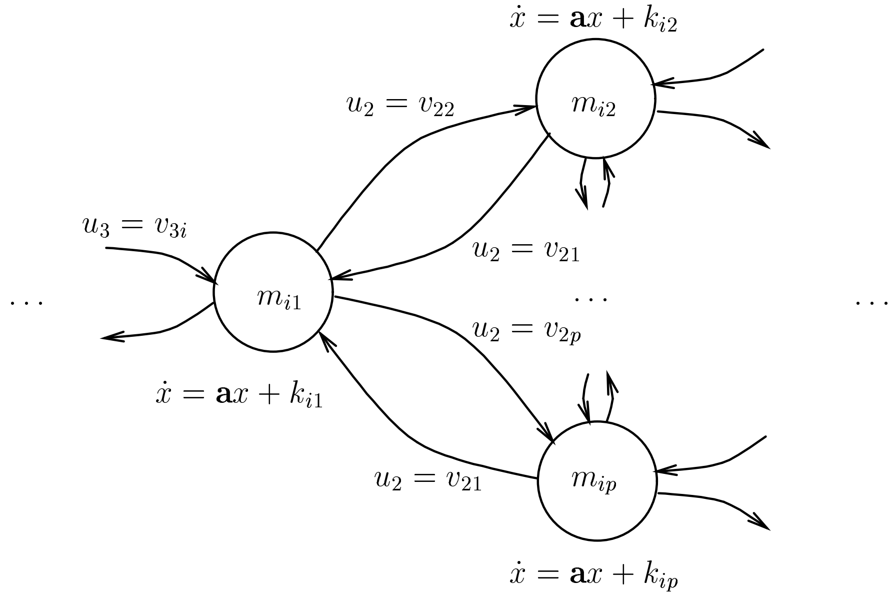

Figure 4: The HTS obtained with type 1 restriction used for the control variable u_2 – where $k_{ij} = bv_{2j} + d_i$.

3.3 Physical variables vs. control variables

In this section we illustrate why adopting any of the previous two models may be preferable to a model in which u_2 is restricted to type 1.

Recall that section 3.1 restricted the control signal u_2 to type 1(a), and section 3.2 assumed it to have type 2. We now assume that u_2 has type (1) with no further assumptions. This leads to a new model, starting from the HTS in Figure 1 and repeating the same step earlier performed for u_3. That is, the voltage signal u is assumed to have a finite range of values leading to the finite range $\{v_{21}, \dots, v_{2p}\}$ for u_2. Replacing every mode of the HTS in Figure 1 with p modes, we get a totally connected HTS of the form shown in Figure 4.

Note that "simplifying" with the same treatment for two different types of input variables gives different results. In the case of a physical variable (the incoming temperature), it is reasonable to assume that values $v_{31}, \dots, v_{3n}$ can be taken by u_3 in that order. In the case of the control signal u_2 we should assume that the variable may be set to any of the values $v_{21}, \dots, v_{2p}$ in any order. We simply have no continuity assumptions on a discrete signal. Here, simplification of the continuous dynamics in a mode comes at a much higher price in terms of the increase in the number of (discrete) modes.

How many inputs are control or physical signals, and how large the respective n and p numbers are in each application, are factors which affect the choice of this third simplification.

4 Refinement of requirements specifications

We now revisit the formulated requirements for the case study, and consider how the transformations made so far can be used to decompose the requirements into "simpler" requirements. The idea is to break down the overall goals of verification so that suitable methods can be selected for proving different subgoals. In particular, different formal verification techniques or analysis tools from control theory can be used for different subgoals.

Given the design specification presented in [9], we consider satisfaction of the following three requirements as overall goals of verification.

- Keeping the reference values constant,
  - the work light will be lit within a time bound from the start of the system, and
  - the system will be stable in the work mode.
- Chamber temperature never exceeds a given limit.
- Whenever the reference values are (re)set, the system will (re)stablise within a time bound or warnings are issued.

4.1 Decompositional verification

The above properties are non-trivial to prove in a hybrid context as exemplified by our model. The approach we adopt for performing step 5 (verification of closed loop model) combines formal and informal reasoning as well as continuous analysis. First, we attempt to find sufficient conditions which facilitate proving a property using domain knowledge. These auxiliary properties may be of the following kinds:

- an assumption which we discharge informally
- a property of the controller or the environment which we formally prove locally
- another property arising as an interaction of the two, which we further refine by finding further sufficient conditions

To prove that the conjunction of the sufficient conditions implies a given requirement may initially be verified informally. The process is continued by further refinements, and formal or informal discharge of derived requirements, until no conditions remain to be proved. Then the system satisfies the top requirement under the informally discharged assumptions.

4.2 Refinement of $\mathbf{R_1}$

Consider the first requirement. The last component of this requirement is within the realms of control theory. We will therefore attempt to refine the other component. We denote the first conjunct in the requirement by $\mathbf{R_1}$.

$\mathbf{R_1}$: Keeping the reference values constant, the work light will be lit within T_1 from the start of the system

First, we provide sufficient conditions for $\mathbf{R_1}$ to hold in the design model:

$\mathbf{R_{10}}$: The system starts in the wait mode with the chamber temperature equal to T_{in}

$\mathbf{R_{11}}$: For certain input restrictions the system will leave the wait mode within T_1 from the start of the system

$\mathbf{R_{12}}$: The only successor to the wait mode is the work mode, and the work light is turned on whenever work mode is entered

We initially claim that

$$\mathbf{R_{10}} \wedge \mathbf{R_{11}} \wedge \mathbf{R_{12}} \rightarrow \mathbf{R_1}$$

At a later stage the very proof of this relation might also be subjected to formal proofs. Also, we may drop R_{10} and replace it with the assumption that the initial chamber temperature is different from T_{in}. But to begin with, we need to make $\mathbf{R_{11}}$ more explicit. Here, we have several paths to take, but the choice can be guided by the verification techniques we intend to utilise. For example, the model in section 3.1 justifies the following refinement:

$\mathbf{R_{111}}$: q stays constant at Q $[m^3/s]$

$\mathbf{R_{112}}$: u may vary every T_{sample} seconds

$\mathbf{R_{113}}$: T_{in} is piecewise constant taking the values $\{v_1, \ldots, v_n\}$, with $v_i \in [T_{min}, T_{max}]$

These requirements (assumptions) allow the use of the discrete time model of the environment as mode-automata. Then, the verification method would be as follows: a compilation from mode-automata to Lustre for the environment model, composition with a controller in Lustre, and further compilation to NP-Tools. In NP-Tools it is possible (but tedious) to show that the number of steps leading to the work light coming on is $\leq N$ for some N (This proves R_1 for a given T_{sample} provided that $T_1 \geq NT_{sample}$). This is a track we continue to explore in the future.
Thus, under these assumptions it is indirectly proven that:
$\mathbf{R_{111}} \wedge \mathbf{R_{112}} \wedge \mathbf{R_{113}} \rightarrow \mathbf{R_{11}}$.

4.3 Refinement of $\mathbf{R_2}$

Consider now the second property which is a safety property. The only actuator in the system causing hazards is the heater which must be shown to heat the air to desired levels but *n*ot to hazardous levels. Let $\mathbf{R_2}$ express this property.

$\mathbf{R_2}$: The chamber temperature T_{chamb} never exceeds a limit T_H

The aim is to find (strong enough) properties $\mathbf{R_{2i}}$ which are sufficient for proving $\mathbf{R_2}$. We start by following conditions:

$\mathbf{R_{20}}$: The chamber temperature is equal to T_{in} at start time

$\mathbf{R_{21}}$: The reference temperature T_{Ref} can never exceed $T_{Ref_{max}}$, and $T_{Ref_{max}} + 2\Delta < T_H$

$\mathbf{R_{22}}$: When the system is in wait-, solution-, or work-mode, $T_{chamb} < T_{Ref_{max}} + 2\Delta$

$\mathbf{R_{23}}$: The system is never in block-mode while $T_{chamb} > T_H$

Where we claim that $\mathbf{R_{20}} \wedge \mathbf{R_{21}} \wedge \mathbf{R_{22}} \wedge \mathbf{R_{23}} \rightarrow \mathbf{R_2}$. We then continue to refine $\mathbf{R_{23}}$:

$\mathbf{R_{231}}$: $T_{chamb} = T_{Ref} + 2\Delta < T_H$ when entering the block-mode

$\mathbf{R_{232}}$: u = 0 throughout the stay in block-mode

$\mathbf{R_{233}}$: The system leaves the block-mode after T_{block} seconds, and enters the off-mode

Again claiming that $\mathbf{R_{231}} \wedge \mathbf{R_{232}} \wedge \mathbf{R_{233}} \rightarrow \mathbf{R_{23}}$.
All of these properties are easily proved using the NP-Tools model of the controller.

5 Related works

Several authors use the notions of abstraction or simulation as a way of modelling a system which allows more behaviours than the system under study, e.g. [3, 2, 10]. The main ingredient is that a requirement is proved to hold for a more general model than the actual system – implying that the system satisfies the requirement.

Puri and Varaiya [10] reason about safety properties only. They propose two abstractions: with the first method a differential equation is replaced by a simpler differential inclusion. In the second approach a differential equation is replaced by a timed automaton. That is, the bounds on the evolution of trajectories from a differential equation are used to map the states of a timed automaton to regions in the state space of the hybrid automaton. To show that H satisfies a specification S, it is shown that reachable states of H (denoted by $R[H]$) are a subset of S. The actual proof is performed on a bound on reachable sets of an abstraction of H (denoted $h(R[H_A])$, which is possible since $R[H] \subset h(R[H_A]) \subset S$. Both of these approaches are illustrated on the train and gate example but are very difficult to apply to more complex examples (e.g. our chamber).

The work we present goes in the opposite direction, although it has some similarities with the work of Puri and Varaiya. Abstraction in our sense is "losing some information" by ignoring some physical realities. Although we know that an input signal may vary in several ways we assume that it has a piecewise constant slope for example. A system that uses this signal as an input has therefore less behaviours than if inputs were not restricted. Hence, it is *more specific* than the original model, not more general.

We reason over runs of a system, not just the set of reachable states. In our work M_1 "simpler than" M_2 means that the runs of M_1 are a subset of runs of M_2. If we use M_1 to prove a property P, then the proof is valid provided that the underlying assumptions are valid, since M_1 is more specific than M_2.

6 Work in progress

Compilers developed in other work packages of the project will be used for translations via mode-automata and Lustre to NP-Tools for models falling under section 3.1. Furthermore, our HA models are still not "simple enough" to be algorithmically analysed or conveniently checked by (primitive) theroem provers. The ongoing work concentrates on abstractions in the style of [10] for obtaining bounds by linear evolutions on the model from section 3.2.

Much of the actual effort taken up by the case study so far has been due to the interest in the assessment of the commercial tools by the industrial partner (Saab). Their interest is to use Statemate in a synchronous fashion. We have modelled the climatic chamber in a restricted subset of Statemate (corresponding to Argos), and attempted to analyse the model translated to NP-Tools using a prototype by Prover Technology. The process has shown certain shortcomings in the approach and the prototype translator. So, models developed at Saab directly in NP-Tools are currently used as a basis for verification, while different ways for getting analysable models are explored in this project and elsewhere.

Another track of activity is the systematic decomposition of requirements according to different assumptions/abstractions. This work is so far exploratory. A future goal would be to formally relate the specialisations/abstractions in the system model with refinements/sufficient conditions in the requirements specifications.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Journal of Theoretical Computer Science*, 138:3–34, 1995.

[2] N. Halbwachs, Y.E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis . *Formal Methods in System Design*, 11(2):157–185, August 1997.

[3] T.A. Henzinger and P-H. Ho. A note on abstract Interpretation Strategies for Hybrid Automata. In *proceedings of Hybrid Systems II, LNCS 999*, pages 252–264. Springer Verlag, 1994.

[4] M. Lemmon and P. J. Antsaklis. Inductively Inferring Valid Logical Models of Continuous-state Dynamical Systems. *Journal of Theoretical Computer Science*, 138:201–210, 1995.

[5] F. Maraninchi and Y. Rémond. Mode-automata: About modes and states for reactive systems. In *European Symposium On Programming*, Lisbon (Portugal), March 1998. Springer verlag.

[6] S. Nadjm-Tehrani. Time-Deterministic Hybrid Transition Systems. In *Hybrid Systems V, Proceedings of the fifth international workshop on hybrid systems, LNCS , To appear*. Springer Verlag.

[7] S. Nadjm-Tehrani and J-E. Strömberg. Verification of Dynamic Properties in an Aerospace application. *Formal Methods in System Design*. To appear.

[8] The SYRF Project. Work package 2 Deliverable: Combination of Formalisms. Technical report, Available from `http://www-verimag.imag.fr//SYNCHRONE/SYRF/HTML97/a21.html`, December 1997.

[9] The SYRF Project. Work package 7 Deliverable: Analog/discrete synchronous design – Annex A.7.1a. Available from `http://www-verimag.imag.fr//SYNCHRONE/SYRF/HTML97/a71.html`, December 1997.

[10] A. Puri and P. Varaiya. Verificaion of Hybrid Systems Using Abstrations. In *proceedings of Hybrid Systems II, LNCS 999*, pages 359–369. Springer Verlag, 1994.