Integration of Analog and Discrete Synchronous Design

Simin Nadjm-Tehrani

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

Abstract. The synchronous family of languages (Lustre, Esterel, Signal, Statecharts) provide a great deal of support for verifying a control program at the design and compilation stage. However, a common aspect of embedded systems is that significant properties of the system can not be verified by formally analysing the controller (software) on its own. To analyse the system one requires to state and document assumptions on the environment. Furthermore, proving timeliness properties necessitates justifying a sampling interval and relating the synchronous step to metric time. Support for these activities is generally missing from current formal methods tools.

In this paper we exploit simulation models – based on physical modelling of the environment – together with theorem proving, to prove properties of a closed loop system. We report on the work in progress on a case study provided by Saab Aerospace where deductive tools such as NP-Tools and simulation environments such as MATRIXx-SystemBuild are jointly used for verifying designs in Statecharts or programs in Lustre. The case study treats temperature and flow control in a climatic chamber.

1 Introduction

Many applications of formal methods in system development are in the requirements specification phase – often formalising a subset of requirements corresponding to functional behaviour of the system [11, 7]. In embedded systems, these requirements commonly refer to the component which is under design – typically the controller for some physical devices (realised either as software or electronics).

One approach to verification of embedded systems concentrates on the *design*. This approach while providing a deep understanding of the system subject to design, can not be taken further when it comes to implemented code (except in few reported cases e.g. [13]). Nevertheless, several misconceptions can be identified while analysing early reqiurements (design) documents [4].

A different approach corresponds to verifying the very *control program* implemented in a language natural for this purpose. Here, the languages in the synchronous family for programming (Lustre, Esterel, Signal and Statecharts) play an important role [8, 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 reason for choosing such languages is the support provided in the development environments: the controller can be analysed to eliminate causal inconsistencies, and to detect nondeterminism in the reactive software. The clock calculii in Lustre and Signal, as well as constructive semantics in Esterel can be seen as verification support provided directly by the compiler (comparable to several properties verified by model checking in [4]). Most of the works reported within this community, however, apply verification techniques to check the controller on its own.

To attack the class of properties arising as a result of interaction between the controller and the controlled environment, we need to explicitly model those aspects of the environment which are relevant to the property in question. This approach is common in control engineering. However, the analysis tools within this field primarily provide support for continuous system simulation, and are less adequate for proving properties of programs with discete mode changes and (or) complex non-linear dynamics in the plant.

In this paper we present an approach whereby modelling tools used for analysis of analog systems can be used to substantiate the properties of the environment when formally verifying a synchronous controller. We use a case study provided by Saab Aerospace within the Esprit project SYRF on SYnchronous Reactive Formalisms, and present some verifications performed so far.

In the framework of the project we are able to use algorithmic (symbolic model checking techniques) as well as deductive (theorem proving) methods for verifcation. The former is possible with hybrid mathematical models where the variation of continuous variables is linear in time [9]. The latter approach uses the propositional theorem prover NP-Tools which is based on Stålmarck's method [3]. This tool has now been extended so that the models may include both boolean and integer variables, and the enhanced proving power with integer arithmetic is subject to test in the case studies. However, irrespective of the method used, we need constraints on the behaviour of the physical environment. This requires a modelling activity for which none of the above tools are suited.

With regard to the controller model, the idea is to combine this family of languages in the same environment so that appropriate parts of each application can be modelled according to the preferrable style (state-based, dataflow, textual or graphical). Thus, the verification techniques should be insensitive to the application language and operate on a common intermediate format (see [5]).

Our approach can therefore be summarised as follows. For safety properties of the closed loop system we decompose the property into conjuncts which can either be discharged based on the assumptions on the operating environment, verified on the control program by theorem proving, or verified in the physical environment model using analysis of a continuous mathematical model and (or) simulation. For timeliness properties our work is still explorative. We propose using the simulation models of the environment for estimating an adequate sampling interval, and combining this information with the bounded (step) response in the discrete setting. This can be achieved if discrete time models of the environment are translated to a logical representation via a two step translation process (via mode-automata and Lustre to NP-Tools format). Work on implementing these compilers is in progress in other parts of the project. For the time being we are working on NP-Tools models as if they were the result of automatic translation.

The rest of the paper is structured as follows. In section 2 we set out our framework for modelling and analysis. Section 3 is devoted to description of the case study. Section 4 presents the informal requirements document as presented to us by the industrial partners. Section 5 describes our understanding of the control goals of the system and a selected list of properties to prove. In section 6 we present a mathematical model of the environment based on specific assumptions. Section 7 uses this model to present some simulation examples and indicate a reasonable sampling interval. In section 8 we refine some of the selected properties and explain the verification steps performed so far, combining simulation and theorem proving. Section 9 covers our concluding remarks and related works.

2 Analog-digital synchronous design

Given a controller developed as a synchronous program, the following further activities are essential to verification of the embedded system.

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. Specifying physical properties of the 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.

Step 1 is certainly the most cognitive step. Due to lack of a common methodology for applying formal methods in industry, the requirements documents typically consist of prescriptive texts at the implementation level. Declarative goals for the global behaviour of the system are often implicit. We therefore began by eliciting from the requirements document the properties to check by a formal proof. This is the first difference between working with a real case study and an academic example aiming to test a particular theory or method. The case study also illustrates that the "step" is performed iteratively: first the properties to prove are identified and stated informally, then the detailed design and modelling in step 2 is used to formalise the requirements.

Even step 2 turned out to require more information than that presented to us. We have stated certain assumptions and formalised a model for the environment based on these assumptions.

Adopting the deductive method of proof in NP-Tools, step 4 is straightforward. First, a model of the program in Lustre (or Statecharts) can be readily translated to the format required by NP-Tools (translators for this purpose are being developed in other parts of the project). Secondly, the composition of the environment model and the Lustre (or Statechart) program simply amounts to a conjunction of the environment constraints (as logical formulas) and the translated controller formulas in NP-Tools. One goal of the named case study is to explore whether necessary properties of the environment can be abstracted and represented in this verification environment.

In this article we treat the first two steps and the insights gained for the third step through the case study. We also report on current work under step 5, identifying certain restrictions for step 3.

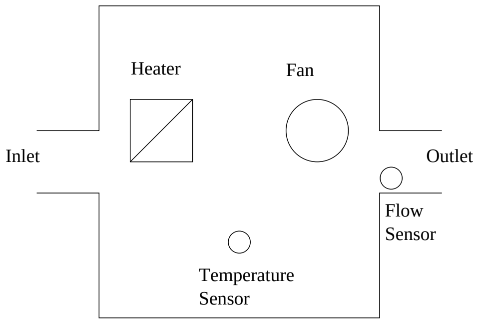

Fig. 1. The hardware components of the air control system.

3 The climatic chamber

The case study consists of a climatic chamber. A control system regulates and monitors the flow and the temperature of air which is circulating in the chamber. Originally, it was developed as a demo system which can be published freely, and demonstrates the kind of problems appearing in developing realistic subsystems such as the ventilation system in the JAS 39 Gripen aircraft. It was presented to the project partners in terms of a 4 page textual specification and an implemented code for a controller in the ASA development environment. ASA allows a top down breakdown of a functional description as a block diagram, where the lowest levels of the description are finite state machines[1].

[1] Currently, Saab is evaluating the tool Statemate for design specification as a replacement for ASA. The approach is equally applicable to any synchronous formalism with an interface to NP-Tools.

Figure 1 presents the component model of the chamber, while Figure 2 shows the interface between the system and the operator.

The chamber is to be ventilated through the inlet and outlet and has a given volume. It has two sensors for measuring the internal air temperature and the air flow. The external interface primarily consists of an on-off button, two analog knobs for setting the values for required temperature and flow, as well as warning signals in terms of a light and a sound. It also includes lights for showing some of its internal modes of operation (wait, work, or block). This may appear to be outside the realms of a component and interface specification (mixing the design related mode names) but has been kept for compatibility with the provided specification.

Unfortunately, the original architectural specification says nothing about the actuators. In the functional part of the document one can see that the heater and fan can be turned on and off. It is also implicit that their output effect can be adjusted in an analog manner, but there is no other information about the nature of control signals or the range of values. Section 6 attempts to fill this gap by making certain assumptions.

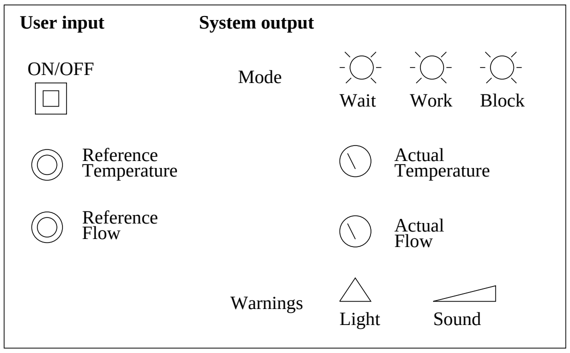

Fig. 2. The external interface to the system.

4 Functional specification

Here we present the original specification as presented to the project (only some variable names were changed to comply with the mathematical notation used in the physical model of section 6).

This informal (and incomplete) specification associates a number of operations with four discrete modes in the controller. The controller code or state machine design document is too large to be presented here. The description below should give an overall picture of the system and illustrate the large step needed before formal methods can be applied to a control system documented by informal text.

temperature control

- At start up, a requested temperature T_{Ref} is selected and the heater is turned on. It is possible to choose T_{Ref} in the interval $[T_{Refmin}, T_{Refmax}]$, where T_{Refmin} is greater than the incoming air temperature T_{in}. Any other chosen values outside the above interval are automatically adjusted up (down) to the lower (upper) limit respectively.
- Let $T_{min} = T_{Ref} - 2\Delta_{temp}$ and $T_{max} = T_{Ref} + 2\Delta_{temp}$. Then the system is considered to be in the **wait** mode while the actual chamber temperature (T_{chamb}) is not within those limits.
- The system will be in the **sol** mode (read solution mode) from the time the temperature hits the region $[T_{min}, T_{max}]$. The system makes a transition to the **work** mode when $|T_{chamb} - T_{Ref}| \leq \Delta_{temp}$ is fullfilled. The time taken for the system to enter and leave the **sol** mode is expected to be a given fixed time called *solution_time*.

ventilation control

- The ventilation or the flow rate is measured by the time it takes to replace the air in the chamber once.
- At start up, a requested flow time $tvent_{Ref}$ is selected and the fan is turned on. This chosen time for the rate of air change should be in the interval $[tventmin, tventmax]$. Other chosen values are automatically adjusted upwards (downwards) to these limits respectively.
- Ventilation is regulated so that the air in the chamber is changed at least once every $tvent_{Ref}$.
- The observed rate of change *flowtime* is based on a measure of the air flow q delivered by the sensor.

monitoring

- The continuously measured values of T_{chamb} and q are to be displayed on the control panel.
- Three lamps indicate being in the system modes **wait, work** and **block** respectively[2].
- A warning by light and sound shall be activated whenever $|T_{Ref} - T_{chamb}| > \Delta_{temp}$ after being in the **sol** mode for a duration of *solution_time*.
- The warning light is activated if $flowtime > X.tvent_{Ref}$ for some fixed ratio X.
- When in **work** mode, if $|T_{Ref} - T_{chamb}| > 2\Delta_{temp}$, or the derivative of the temperature, based on T_{chamb} exceeds a maximum value $Tgrad$, then the mode shall change to the **block** mode.

[2] Apparently no lights should be on when the system is in the solution mode.

- When in **block** mode, the heater is immediately turned off, the warning light and the light indicating the **block** mode are activated. The fan is active for another *tblock* seconds before it is turned off. It shall not be possible to influence the system by changing T_{Ref} or $tvent_{Ref}$. The system must be turned off before restart.
- If T_{Ref} is changed such that $T_{Ref} > T_{chamb} + \Delta_{temp}$ the system makes a transition to the **wait** mode, and the heater is turned on.
- If T_{Ref} is changed such that $T_{Ref} < T_{chamb} - \Delta_{temp}$, the system transits to **wait** mode, the heater is turned off and the fan will be on.

5 Selected Requirements

From the prescriptive requirements document we have identified the following overall goals for the controller.

- 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.

Note that these are not properties of the controller on its own. Note also that these global requirements are not dependent on the particular characteristics of the physical environment, the sensors or the actuators. Such detail will be added later when we formalise the conditions under which they will be proved.

Our approach is to apply different proof techniques where they suit best. Although "being stable in the work mode" can be seen as a safety propery (the conditions for leaving the mode will not be true), it is most expedient to use control theory methods for proving this property. This is due to the fact that not *all* inputs to the system are kept constant (see the result of the physical modelling step). Hence, it is formulated as a stability property.

Another aspect to point out is on the second (safety) property. Here we look beyond the functional demand on the system to monitor and warn when the temperature falls outside given intervals. We rather attempt to see what is the goal of devising such intervals and mode changes and envisage as a (mode-independent) goal of the system that the air is never heated to a hazardous level (even in the block mode and *after* warnings are issued).

The work on treatment of the third requirement builds on the first requirement and has not been carried out yet.

6 Physical environment

In this section we state some assumptions necessary in our initial model of the physical environment.

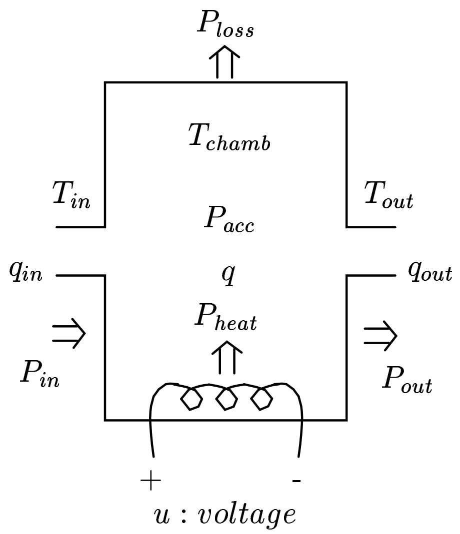

Fig. 3. The variables appearing in the heat model of the chamber.

6.1 Heat-flow inter-relationships

Heat and flow mutually affect each other in non-trivial ways [1, 2]. In a first coarse model of the climatic chamber we include the effects of flow on the heating process but ignore the effects in the opposite direction. The motivation being that the primary concern of the ventilation system is keeping the temperature within given bounds (this may be induced from the nature and extent of warnings).

Next, we make a simplification concerning how flow affects heat. We decide not to consider the air flow effects with respect to the heat transfer factor of the heater (i.e. how velocity of the moving air will affect heating of the air). We only consider the cooling effect as a result of the air exchange caused by the fan with respect to heat generation.

6.2 Assumptions on flow

The flow of mass and volume are related by air density, i.e. $q\ \rho = \dot{m}$, where $q\ [m^3/s]$ denotes volume flow, $\rho\ [kg/m^3]$ is a density factor (normally a function f of temperature T), and $\dot{m}\ [kg/s]$ is the mass flow. Since the design document mostly discusses "the time it takes for exchange of air", it is not clear whether one can measure volume flow or mass flow. No more information about the sensor is available.

The first assumption that we make is that the air pressure in the chamber remains constant relative to the external air pressure. This implies mass balance: that input and output mass flows are equal, $\dot{m}_{in} = \dot{m}_{out}$, which further leads to a similar assumption on volume flow. The latter can, however, be assumed provided that density as a function of temperature can be approximated to a constant factor. That is, assuming $f(T) = k_\rho$, we can further assume that $q_{in} = q_{out}$. Note that the fan can change the value of the chamber flow q, but the assumptions above constrain the in and out flows to stay in a constant relation: more specifically $q = q_{in} = q_{out}$. No further assumptions are made on the input to the flow actuator We simply assume that a given (reference) flow can be obtained placing the emphasis on the physical model of the heat process.

6.3 Model of the heating process

The heater produces heat which is accumulated in the chamber to an extent which depends on the flow. Figure 3 presents the variables appearing in the heat process model.

Based on the earlier assumptions the heat model will have three inputs: the air flow (q), the control signal to the heater (u), and the outside air temperature (T_{in}). The main state variable is the chamber temperature (T_{chamb}) which can be measured by the temperature sensor and delivered to the heat controller. However, to see how the final model is arrived at we start by stating the energy conservation principle:

$$P_{in} + P_{heat} - P_{out} - P_{loss} = P_{acc} \tag{1}$$

where the sum of heat power is stated to be constant, i.e. the incoming power (P_{in}), that resulting from the heater activation (P_{heat}), the outgoing power (P_{out}), and the power lost through the walls of the chamber (P_{loss}) are assumed to give the accumulated heat power in the chamber (P_{acc}).

Making a further assumption that $T_{chamb} = T_{out}$, using the earlier assumption $q = q_{in} = q_{out}$, and using equation 1, we can now simplify the above formula. The P-terms are replaced with their respective definitions, which contrain each element in terms of input/output variables, and some model coefficients (all the k-terms):

$$\begin{aligned}
P_{in} &= q_{in}\ T_{in}\ k_{in} \\
P_{heat} &= u^2\ k_{heat} \\
P_{out} &= q_{out}\ T_{out}\ k_{out} \\
P_{loss} &= T_{chamb}\ k_{loss} \\
P_{acc} &= \dot{T}_{chamb}\ k_{chamb}
\end{aligned}$$

Thus, we obtain the following equation defining the chamber temperature:

$$\begin{aligned}
\dot{T}_{chamb} = {} & 1/k_{chamb} \\
& (\ q\ T_{in}\ k_{in} + \\
& \ u^2\ k_{heat} - \\
& \ T_{chamb}(q\ k_{out} - k_{loss}))
\end{aligned} \tag{2}$$

Note that despite several simplifying assumptions the model is highly non-linear. So, formally proving the properties of the hybrid model consisting of the synchronous controller and this chamber model, is a challenge for hybrid system verification. For example, significant restrictions remain (to do step 4 in section 2) before a linear hybrid automata for this model is derived [6].

7 Environment constraints

So far we have developed an abstract mathematical model of the environment giving the relationship between the state variables and the inputs. The next step is to study its temporal behaviour. This necessitates a computational model of the physical environment based on equation 2 above.

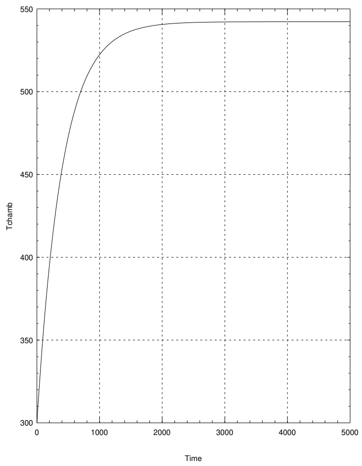

Fig. 4. The dynamics of the heat process with $u = 125V$, $q = 0.05\ m^3/s$, and $T_{in} = 300K$.

Here we have used the modelling and analysis environment provided by the tool MATRIXx with the accompanying toolbox SystemBuild [12]. The translation of the mathematical model into a block diagram is straight forward but there are a number of coefficients and constants which need to be fixed (estimated) Thus, the SystemBuild block diagram is anotated with actual values for the different k-terms in the model above.

Based on these estimates, upper and lower bounds provided for T_{in} and for $flowtime$, we can now simulate the chamber for different T_{in} signals and different input voltages. Figure 4 shows for example the dynamics of the system for a constant air flow, constant temperature control, and a constant environment temperature. A sinusuidal signal for T_{in} with a different value for the control signal gives the simulated trajectory presented in Figure 5.

7.1 Estimating the sampling interval

The first step for checking timing requirements must relate a synchronous step to metric time. Since one of the required properties involves timeliness, we can no longer "ignore" real time as is usual in synchronous programming.

Note that although rules of thumb exist for determining the sampling interval based on model coefficients, these are only rough guesses and reliable enough in the case of linear systems. For non-linear models and where the system is of higher dimension[3] performing selected simulations is the appropriate way to find an adequate step size.

Changing the input signal characteristics in the simulation model provides a feel for how fast the temperature inside the chamber varies in response to factors not governed by the heat controller. For example, the curves mentioned above show that heating up the chamber temperature to required levels takes up to 3000 seconds with the assumed heater.

The series of simulations performed have indicated that the control program provided by Saab has far too high sampling frequency. The program sampling rate is of course not explicit. This is due to the abstraction of time in the synchronous paradigm. However, since the program is supposed to warn for irregularities if the temperature does not stablise within given time limits, counters are used and incremented at each program cycle.

Deducing time from these counters, one can observe that the program is sampling at 10 Hz where 0.01 Hz is a more reasonable sampling rate. This information was in itself interesting for other work in the project: the test pattern generation team was discovering that to arrive at interesting results test data for 30,000 steps had to be generated.

8 Compositional verification

We now revisit the formulated requirements for the case study, and consider how the models presented 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.

8.1 Refinement of requirements

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 our knowledge of the system. These auxiliary properties may be of the following kinds:

[3] Recall that we have assumed the flow to be an input to begin with; adding the flow as a state variable makes the model more complex, and no longer monotonic in presence of a non-zero k_{loss}.

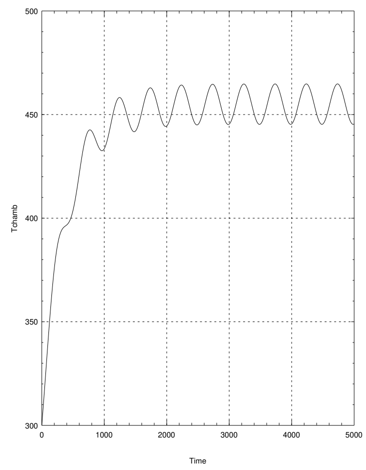

Fig. 5. The dynamics of the heat process with $u = 100$, $q = 0.05\ m^3/s$, and T_{in}: 50K amplitude at 0.002 Hz added to 300K bias.

– 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

Then the system satisfies the top requirement under the informally discharged assumptions.

8.2 Refinement of $\mathbf{R_2}$

Consider 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 *not* 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 $\mathbf{R_{2i}}$ such that $\bigwedge R_{2i}$ is sufficient for proving $\mathbf{R_2}$. We start with the 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 $T_{Ref_{max}} + 2\Delta < T_H$
$\mathbf{R_{22}}$: Whenever the system is in wait-, solution-, or work-mode, we have $T_{chamb} < T_H$
$\mathbf{R_{23}}$: The system is never in block-mode while $T_{chamb} > T_H$

These properties can be discharged informally or proved within the NP-Tools model except for $\mathbf{R_{23}}$ which we continue to refine:

$\mathbf{R_{231}}$: $T_{chamb} = T_{Ref} + 2\Delta < T_H$ 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 *tblock* seconds, and enters the off-mode
$\mathbf{R_{234}}$: The temperature T_{chamb} does not increase while the system is in the block mode

This is sufficient for proving the safety property provided that
$\mathbf{R_{231}} \wedge \mathbf{R_{232}} \wedge \mathbf{R_{233}} \wedge \mathbf{R_{234}} \rightarrow \mathbf{R_{23}}$.
Properties $\mathbf{R_{231}}$ to $\mathbf{R_{233}}$ are easily proved using the NP-Tools model of the controller. For the proof of $\mathbf{R_{234}}$ we use continuous reasoning based on the simulation models.

8.3 Refinement of $\mathbf{R_1}$

Consider now the first requirement. The stability component of this requirement can only be verified using control theory and exact knowledge of the control algorithm in the work mode. Here, we concentrate on the first component, denoting it 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:

$\mathbf{R_{11}}$: The system starts in the wait mode with the chamber temperature equal to T_{in}
$\mathbf{R_{12}}$: While T_{Ref} is constant, the only successor to the wait mode is the solution mode
Given input restrictions $\mathbf{R_{10}}$,
$\mathbf{R_{13}}$: The system leaves the wait mode within *wait_time* from the start of the system
$\mathbf{R_{14}}$: the system leaves the solution mode within *solution_time* from entering the mode
$\mathbf{R_{15}}$: While T_{Ref} is constant, the only successor to the solution mode is the work mode, and the work light is turned on whenever work mode is entered
$\mathbf{R_{16}}$: $\mathit{wait\_time} + \mathit{solution\_time} \leq t_1$

We initially claim that

$$\mathbf{R_{11}} \wedge \mathbf{R_{12}} \wedge \mathbf{R_{13}} \wedge \mathbf{R_{14}} \wedge \mathbf{R_{15}} \wedge \mathbf{R_{16}} \rightarrow \mathbf{R_1}$$

At a later stage we may drop $\mathbf{R_{11}}$ and replace it with the assumption that the initial chamber temperature is different from T_{in}. But to begin with, we make the restrictions in $\mathbf{R_{10}}$ more explicit, and show that

$$\mathbf{R_{10}} \rightarrow \mathbf{R_{12}} \wedge \mathbf{R_{13}}$$

Here, we have several paths to take, but the choice is guided by the verification techniques we intend to utilise. For example, the following restrictions justify the adoption of a discrete-time model of the environment in a mode-automaton [15] with n discrete modes. Each mode is then governed by a difference equation derived from the continuous model of section 6 in the standard manner (for details see [6]).

$\mathbf{R_{101}}$: q stays constant at Q $[m^3/s]$
$\mathbf{R_{102}}$: u may vary every t_{sample} seconds
$\mathbf{R_{103}}$: T_{in} is piecewise constant taking the values $\{v_1, \ldots, v_n\}$

Note that mode-automata [15] can be seen as a restriction of time-deterministic hybrid transition systems [17] in which changes in state variables in each mode are defined in terms of a Lustre program. Each state variable is thus defined by an equation relating the state variables at the previous (clock) step and the current input. Adopting these restrictions, the verification method would be as follows: using a scheme for compilation from mode-automata to Lustre we obtain a model of the environment in Lustre which can be composed with a controller in Lustre, and further compiled 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 $\mathbf{R_1}$ for a given t_{sample} provided that $t_1 \geq N t_{sample}$). This is a track we are currently exploring in the project. Note that this is one reason for not choosing a "too short" sampling interval. As well as other reasons which may be associated with oversampling, we would like to prove the bounded response property for as small N as feasible.

9 Concluding remarks

The work we have reported is at a too early stage for making definitive remarks about feasibility of combining "push-botton" theorem provers and simulation environments. More work is also needed to compare the method with "heavy duty" theorem proving in the spirit of [7]. However, some preliminary points for discussion have already emerged. Some of the shortcomings are reminiscent of those reported in [4]: the limitation to interger arithmetic, for example, means that the counter proofs presented by the system are more informative than the safety proofs holding over a limited range. This is, however, compensated in our approach by departing from fully formal proofs and combining with a simulation analysis when (local) reasoning over reals is crucial to the property in question.

There were several obstacles in achieving the goals set out, many of a practical nature, such as childhood problems in translators from e.g. Statecharts to NP-Tools. Since the translators to NP-Tools traditionally encapsulate the model as a circuit with only in- and out-pins visible, it is necessary to recompile the model every time a small error is detected in the original design description. The fact that only a subset of the Statemate languages was covered by the prototype translator also made the Statemate description of the continuous control parts less natural than it could be (c.f. an environment where state-based and data-flow descriptions can be combined). The work should however serve as an illustration of the many steps "in between" going from an informal document and a synchronous program to formal verification of "hybrid" properties.

Our model of the heat process intentionally made several simplifications to fit an early experimental set up. The interested reader may for example refer to a more complex model of heat exchangers in [16] where some of our restrictions are relaxed. The purpose of that paper is the illustration of a rich simulation language and only the plant part of the heat exchanger is subjected to validation by simulation.

To get an impression of the size of the system it should be indicated that the translated NP-Tools version of the full controller with a basic interface to the plant has 96 input variables and 88 output variables (seen as a circuit). The time taken for the proofs has been within fractions of a second for the requirements checked so far. These have included a number of sanity checks (such as determinism in the controller) and the subgoals for the safety property.

It is also interesting to note that the size of the real ventilation subsystem in the same format is 700 input variables and 500 output variables. Despite the seemingly large state space, the size of the reachable states set – as far as required for the types of properties mentioned – is small enough for practical purposes, even in the real system [14].

Acknowledgements

This work was supported by the Esprit LTR project SYRF, the Swedish board of Technical Research (TFR) and the Swedish board of technical development (Nutek). Also, cooperation of the Saab people, specially Ove Åkerlund, is gratefully acknowledged.

References

1. H. Alvarez. *Energiteknik, Del I.* Studentlitteratur, Lund, Sweden, 1990.
2. C.O. Bennett. *Momentum, Heat and Mass Transfer, 3rd Edition.* McGrawHill, 1982.
3. B. Carlson, M. Carlsson, and G. Stålmarck. NP(FD): A Proof System for Finite Domain Formulas. Technical report, Logikkonsult NP AB, Sweden, April 1997. Available from `http://www-verimag.imag.fr//SYNCHRONE/SYRF/HTML97/a321.html`.
4. W. Chan, R.J. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin, and J.D. Reese. Model Checking Large Software Specifications. *IEEE Transactions on Software Engineering*, 24:498–519, July 1998.
5. The SYRF Project Deliverables. Work package 2: Combination of Formalisms. Available from `http://www-verimag.imag.fr//SYNCHRONE/SYRF/HTML97/a21.html`, December 1997.
6. The SYRF Project Deliverables. Work package 7: Integration of Analog and Discrete Synchronous Design. Available from `http://www-verimag.imag.fr//SYNCHRONE/SYRF/HTML98/a72.html`, December 1998.

7. B. Dutertre and V. Stavridou. Formal Requirements Analysis of an Avionics Control System. *IEEE Transactions on Software Engineering*, 25(5):267–278, May 1997.
8. N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.
9. N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of Linear Hybrid Systems by means of Convex Approximations. In *In proceedings of the International Symposium on Static Analysis SAS'94, LNCS 864.* Springer Verlag, September 1993.
10. D. Harel. STATECHARTS: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.
11. M. Heimdahl and N. Leveson. Completeness and Consistency in Heirarchical State-based Requirements. *IEEE transactions on Software Engineering*, 22(6):363–377, June 1996.
12. Integrated Systems Inc. *SystemBuild v 5.0 User's Guide.* Santa Clara, CA, USA, 1997.
13. H. Jifeng, C.A.R. Hoare, M. Fränzle, M. Müller-Olm, E-R. Olderog, M. Schenke, M.R. Hansen, A.P. Ravn, and H. Rischel. Provably Correct 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 288–335. Springer Verlag, 1994.
14. O. Åkerlund. Application of Formal Methods for Analysis of the Demo System and parts of the Ventilation System of JAS 39 (in swedish). Technical report, Saab Aerospace AB, Linköping, Sweden, January 1997.
15. 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.
16. S.E. Mattsson. On modelling of heat exchangers in modelica. In *Proc. 9th European Simulation Symposium*, Passau, Germany, October 1997. Available through `http://www.modelica.org/papers/papers.shtml`.
17. S. Nadjm-Tehrani. Time-Deterministic Hybrid Transition Systems. In *Hybrid Systems V, Proceedings of the fifth international workshop on hybrid systems, 1997, LNCS, To appear.* Springer Verlag.