Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study

Franck Cassez[1], Jan J. Jessen[2], Kim G. Larsen[2],
Jean-François Raskin[3], Pierre-Alain Reynier[4]

[1] National ICT Australia & CNRS, Sydney, Australia
[2] CISS, CS, Aalborg University, Denmark
[3] CS Department, Université Libre de Bruxelles, Belgium
[4] LIF, University of Marseilles & CNRS, UMR 6166, France

Abstract. In this paper, we show how to apply recent tools for the automatic synthesis of robust and near-optimal controllers for a real industrial case study. We show how to use three different classes of models and their supporting existing tools, UPPAAL-TIGA for synthesis, PHAVER for verification, and SIMULINK for simulation, in a complementary way. We believe that this case study shows that our tools have reached a level of maturity that allows us to tackle interesting and relevant industrial control problems.

1 Introduction

The design of controllers for embedded systems is a difficult engineering task. Controllers have to enforce properties like safety properties (e.g. "nothing bad will happen"), or reachability properties (e.g. "something good will happen"), and ideally they should do that in an efficient way, e.g. consume the least possible amount of energy. In this paper, we show how to use (in a systematic way) models and a chain of automatic tools for the synthesis, verification and simulation of a provably correct and near optimal controller for a real industrial equipment. This case study was provided to us by the HYDAC ELECTRONICS GMBH company in the context of a European research project[5].

The system to be controlled is depicted in Fig. 1 and is composed of: (1) a machine which consumes oil, (2) a reservoir containing oil, (3) an accumulator containing oil and a fixed amount of gas in order to put the oil under pressure, and (4) a pump. When the system is operating, the machine consumes oil under pressure out of the accumulator. The level of the oil, and so the pressure within the accumulator (the amount of gas being constant), can be controlled using the pump to introduce additional oil in the accumulator (increasing the gas pressure). The control objective is twofold: first the level of oil into the accumulator (and so the gas pressure) can be controlled using the pump and must be maintained into a safe interval; second the controller should try to minimize the level of oil such that the accumulated energy in the system is kept minimal.

[5] Quasimodo: "Quantitative System Properties in Model-Driven-Design of Embedded", see `http://www.quasimodo.aau.dk/` for details.

In a recent work [5], we have presented an approach for the synthesis of a correct controller for a timed system. It was based on the tool UPPAAL-TIGA [1] applied on a very abstract untimed game model for synthesis and on SIMULINK [6] for simulation. To solve the HYDAC ELECTRONICS GMBH control problem, we use three complementary tools for three different purposes: UPPAAL-TIGA for synthesis, PHAVER [4, 3] for verification, and SIMULINK for simulation. For the synthesis phase, we show how to construct a (game) model of the case study which has the following properties:

- it is simple enough to be solved automatically using algorithmic methods implemented into UPPAAL-TIGA;
- it ensures that the synthesized controllers can be easily implemented.

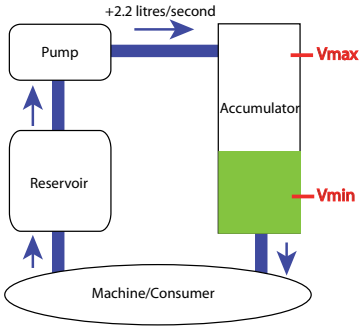


Fig. 1. Overview of the System.

To meet those two requirements, we consider an idealized version of the environment in which the controller is embedded, but we put additional constraints into the winning objective of the controller that ensure the robustness of winning strategies. As the winning strategies are obtained in an abstract model of the system, we show how to embed automatically the synthesized strategies into a more detailed model of the environment, and how to automatically prove their correctness using the tool PHAVER for analyzing hybrid systems. While the verification model allows us to establish correctness of the controller that is obtained automatically using UPPAAL-TIGA, it does not allow us to learn its expected performance in an environment where noise is not completely antagonist but follows some probabilistic rules. For this kind of analysis, we consider a third model of the environment and we analyze the performance of our synthesized controller using SIMULINK.

To show the advantages of our approach, we compare the performances of the controller we have automatically synthesized with two other control strategies. The first control strategy is a simple two-point control strategy where the pump is turned on when the volume of oil is reaching a floor value and turn off when the volume of oil reaches a ceiling value. The second control strategy is a strategy designed by the engineers at HYDAC ELECTRONICS GMBH with the help of SIMULINK.

Structure of the paper. In section 2, we present the HYDAC ELECTRONICS GMBH control problem. In section 3, we present our construction of a suitable abstract model of the system, and the strategy we have obtained using the synthesis algorithm of UPPAAL-TIGA. In section 4, we embed the controllers into a continuous hybrid model of the environment and use the tool PHAVER to verify their correctness and robustness: we prove that strategies obtained using UPPAAL-TIGA are indeed correct and robust. In section 5, we analyze and compare the performances in term of mean volume of the three controllers using SIMULINK.

2 The Oil Pump Control Problem

We give now more details about the system, the constraints to respect, and the control objectives. The controller must operate the pump (switch it on and off) to ensure the following two main requirements:

- (R_1): the level of oil $v(t)$ at time t (measured in litres) into the accumulator must always stay within two *safety* bounds $[V_{min}; V_{max}]$, in the sequel $V_{min} = 4.9l$ and $V_{max} = 25.1l$;
- (R_2): a large amount of oil in the accumulator implies a high pressure of gas in the accumulator. This requires more energy from the pump to fill in the accumulator and also speeds up the wear of the machine. This is why the level of oil should be kept minimal during operation, in the sense that $\int_{t=0}^{t=T} v(t)$ is minimal for a given operation period T.

While requirement (R_1) is a *safety requirement* and so must never be violated by any controller, (R_2) is an *optimality* requirement and will be used to compare different controllers.

The Machine. The oil consumption of the machine is cyclic. The cycle of consumptions, as given by the HYDAC ELECTRONICS GMBH company, is depicted in Fig. 2.

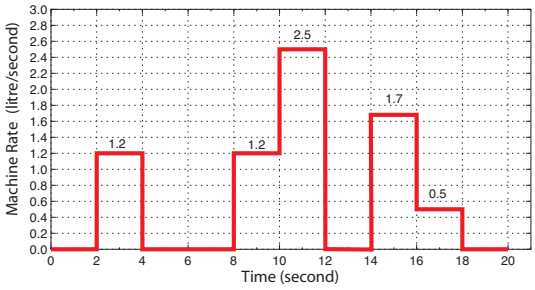


Fig. 2. Cycle of the Machine.

Each period of consumption is characterized by a rate of consumption (expressed as a number of litres per second), a date of beginning and a duration. We assume that the cycle is known *a priori*: we do not consider the problem of identifying the cycle (which can be performed as a pre-processing step). The control strategy must allow the machine to operate for an arbitrarily large number of cycles. At time 2, the rate of the machine goes to $1.2l/s$ for two seconds. From 8 to 10 it is 1.2 again and from 10 to 12 it goes up to 2.5 (which is more than the maximal output of the pump). From 14 to 16 it is 1.7 and from 16 to 18 it is 0.5. Even if the consumption is cyclic and known in advance, the rate is subject to *noise*: if the mean consumption for a period is c l/s, in reality it always lies within that period in the inverval $[c - \epsilon, c + \epsilon]$, where ϵ is fixed to 0.1 l/s. This property is noted F.

The Accumulator and the Pump. The volume of oil within the accumulator is initially equal to 10 l. The pump is either *on* or *off*, and we assume it is initially *off*. The operation of the pump must respect the following *latency* constraint: there must always be two seconds between any change of state of the pump, i.e.

if it is turned *on* (respectively *off*) at time t, it must stay on (respectively *off*) at least until time $t + 2$, we note P_1 this property. When it is *on*, its *output* is equal to $2.2l/s$. Note that as the power of the pump is not always larger than the demand of the machine during one period of consumption (see Fig. 2 between 10 and 12), some extra amount of oil must be present in the accumulator before that period of consumption to ensure that the minimal amount of oil constraint (requirement R_1) is not violated[6].

Additional Requirements on the Controller. To summarize, the controller to design must turn on and off the pump at the appropriate points in time while respecting the *latency* of the pump (property P_1) to ensure that requirements R_1 is satisfied, even under the fluctuations F within the cyclic consumption phase, for an arbitrarily long period of time. Moreover we should try to minimize the accumulated oil during each cycle (requirement R_2).

Because the consumptions are subject to noise, it is necessary to allow the controller to check periodically the level of oil in the accumulator (as it is not predictable in the long run). Nevertheless, the controller should exploit the cyclic nature of the consumption to optimize the level of oil. So, we will also allow our controllers to take control decisions at predefined instant in time during the cycle using timers.

The Bang-Bang Controller and the Smart Controller. In the next sections, we will show how to use synthesis algorithms implemented in Uppaal-TiGA to obtain a simple but still efficient controller for the oil pump. This controller will be compared to two other solutions that have been previously considered by the Hydac Electronics Gmbh company.

The first one is called the *Bang-Bang controller*. Using the sensor for oil volume in the accumulator, the Bang-Bang controller turns *on* the pump when a *floor* volume value V_1 is reached and turns *off* the pump when a *ceiling* volume value V_2 is reached. The Bang-Bang controller is simple but it does not exploit the timing information about the consumption periods within a cycle. To obtain better performances in term of energy consumption, engineers at Hydac Electronics Gmbh have designed a controller that exploit this timing.

This second controller is called the *Smart controller*. With this controller, after an initial phase for learning the consumer demands, the pump is activated at points in time precomputed in advance for each cycle according to a rather complex, adaptive strategy. Though simulations of Simulink models developed by Hydac Electronics Gmbh reveal no unsafe behaviour, the engineers have not been able to verify its correctness and robustness. As we will see later, in a simplified form the strategy is not safe in the long run in presence of noice.

3 The Uppaal-TiGA Model for Controller Synthesis

In this section, we show how to synthesize automatically, from a game model of the system and using Uppaal-TiGA, an efficient controller for the Hydac case

[6] It might be too late to switch the pump on when the volume reaches V_{min}.

study. UPPAAL-TIGA is a recent extension of the tool UPPAAL which is able to solve timed games.

Game Models of Control Problems. While modeling control problems with games is very *natural* and *appealing*, we must keep in mind several important aspects. First, solving games is computationally hard, so we should aim at game models that are sufficiently abstract. Second, when modeling a system with a game model, we must also be careful about the information that is available to each player in the model. The current version of UPPAAL-TIGA offers *games of perfect information* (see [2] for steps towards games for imperfect information into UPPAAL-TIGA.) In games of perfect information, the two players have access to the full description of the state of the system. For simple objectives like safety or reachability, the strategies of the players are functions from states to actions. To follow such strategies, the implementation of the controller must have access to the information contained in the states of the model. In practice, this information is acquired using sensors, timers, etc.

The UPPAAL-TIGA *Model.* We describe in the next paragraphs how we have obtained our game model for the Hydac case study. First, to keep the game model simple enough, we have designed a model which: (a) considers one cycle of consumption; (b) uses an abstract model of the fluctuations of the rate; (c) uses a discretization of the dynamics within the system. Second, to make sure that the winning strategies that will be computed by UPPAAL-TIGA are implementable, the states of our game model only contain the following information, which can be made available to an implementation:

- the volume of oil at the beginning of the cycle;
- the ideal volume as predicted by the consumption period in the cycle;
- the current time within the cycle;
- the state of the pump (*on* or *off*).

Third, to ensure robustness of our strategies, *i.e.* that their implementations are correct under imprecisions on measures of volume or time, we consider some margin parameter m which represents how much the volume can deviate because of these imprecisions. We will consider values in range $[0.1; 0.4]l$.

Global Variables. First, we discretize the time w.r.t. ratio stored in variable D, such that D time units represent one second. Second, we represent the current volume of oil by the variable V. We consider a precision of $0.1l$ and thus multiply the value of the volume by 10 to use integers. This volume evolves according to a rate stored in variable $\mathsf{V\_rate}$ and the accumulated volume is stored in the variable $\mathsf{V\_acc}$[7]. Finally, we also use an integer variable time which measures the global time since the beginning of the cycle.

The Model of the Machine. The model for the behaviour of the machine is represented on Fig. 3(a). Note that all the transitions are uncontrollable (represented by dashed arrows). The construction of the nodes (except the mid-

[7] To avoid integers divisions, we multiply all these values by D.

dle one labelled bad) follows easily from the cyclic definition of the consumption of the machine. When a time at which the rate of consumption changes is reached, we simply update the value of the variable V_rate. The additionnal central node called bad is used to model the uncertainty on the value of V due to the fluctuations of the consumption. The function Noise (Fig. 4) checks whether the value of V, if modified by these fluctuations, may be outside the interval $[V_{min} + 0.1, V_{max} - 0.1]$ [8]. The function final_Noise (Fig. 4) checks the same but for the volume obtained at the end of cycle and against the interval represented by V1F and V2F. Note that this modelling allows in some sense to perform partial observation using a tool for games of perfect information. Indeed, the natural modelling would modify at each step the actual value of the variable V and the strategies would then be aware of the amount of fluctuations. In our model the ideal value of V is predictable because it directly depends on the current time and from the point of view of the controller (i.e. the state of the system) it does not give any information about the volume.

The Model of the Pump. The model for the pump is represented on Fig. 3(b). The transitions are all controllable (plain arrows). The pump simply consists of two locations representing whether the pump is ON or OFF. Moreover, the latency constraint[9] P_1 is expressed using the clock z. An additional integer variable i is used to count how many times the pump has been started on. We use parameter N to bound this number of activations, which is set to 2 in the following. Note also that the time points of activation/desactivation of the pump are stored in two vectors start and store.

The Model of the Scheduler. We use a third automaton represented on Fig. 3(c) to schedule the composition. Initially it sets the value of the volume to V0 and then it repeats the following actions: it first updates the global variables V, V_acc and time through function update_val. Then the scheduling is performed using the two channels update_cy[10] and update_pump. When the end of the cycle of the machine is reached, the corresponding automaton sets the boolean variable done to true, whic forces the scheduler to go to location END.

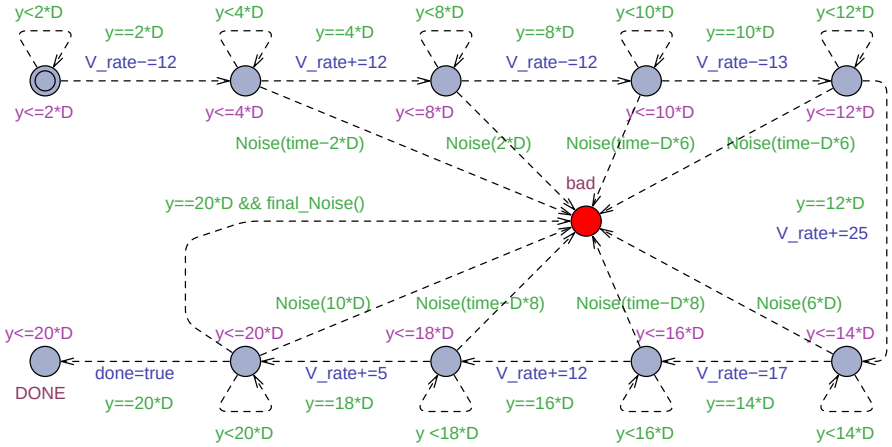
Composition. We denote by $\mathcal{A}$ the automaton obtained by the composition of the three automata described before. We consider as parameters the initial value of the volume, say V_0, and the target interval I_2, corresponding to V1F and V2F, and write $\mathcal{A}(V_0, I_2)$ the composed system.

Global Approach for Synthesis. Even if the game model that we consider is abstract and concentrates on one cycle, note that our modelling enforces the constraints expressed in section 2. Indeed, R_1 is enforced through function Noise, F is handled through the two functions Noise and final_Noise, and P_1 is expressed explicitely in the model of the pump. To extend our analysis from one cycle to any number of cycles, and to optimize objective R_2, we formulate the following

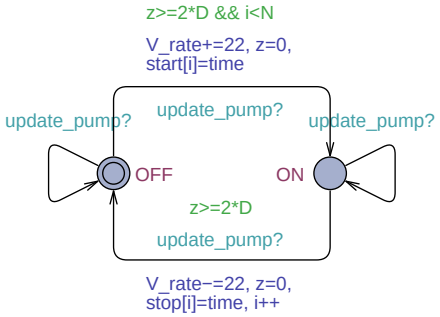
[8] For robustness, we restrain safety constraints of 0.1 l.

[9] Notice that we impose a bit more than P_1 as we require that 2 seconds have elapsed at the beginning of the cycle before switching on the pump.

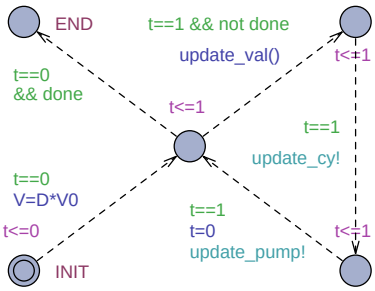
[10] We did not represent this synchronization on Fig. 3(a) to ease the reading.



(a) Model of the cyclic consumption of the machine



(b) Model of the pump



(c) Model of the scheduler

Fig. 3. UPPAAL-TIGA models.

control objective (for some fixed margin $m \in \mathbb{Q}$) CO:

Find some interval $I_1 = [V_1, V_2] \subseteq [4.9; 25.1]$ such that (Property $(*)$):

(i) I_1 is *m-stable*: from all initial volume $V_0 \in I_1$, there exists a strategy for the controller to ensure that, whatever the fluctuations on the consumption, the value of the volume is always between 5 l and 25 l and the volume at the end of the cycle is within interval $I_2 = [V_1 + m, V_2 - m]$,

(ii) I_1 is *optimal* among m-stable intervals: the worst accumulated volume of the strategies of I_1 is minimal.

The strategies that fulfill that control objective have a nice *inductive property*: as the value of the volume of oil at the end of the cycle is ensured to be within I_2, and $I_2 \subset I_1$ if $m > 0$, the strategies computed on our one cycle model can be safely repeated as many times as desired. Moreover, the choice of the margin parameter m will be done so as to ensure robustness. We will verify in PHAVER

```
bool Noise(int s){
// s is the duration of consumption (in t.u.)
  return (V-s<(Vmin+1)*D | V+s>(Vmax-1)*D);}

bool final_Noise(){
// 10*D t.u. of consumption in 1 cycle
  return (V-10*D<V1F*D | V+10*D>V2F*D);}
```

```
void update_val(){
  int V_pred = V;
  time++;
  V+=V_rate;
  V_acc+=V+V_pred;
}
```

Fig. 4. Functions embedded in UppAal Tiga models

that even in presence of imprecisions, the final volume, if it does not belong to I_2, belongs to I_1, because of the definition of I_2 as strict subset of I_1.

We now describe a procedure to compute an interval verifying Property $(*)$, and the associated strategies. We proceed as follows [11]:

1. For each $V_0 \in I_1$, and target final interval J, compute (by a binary search) the minimal accumulated volume $Score(V_0, J)$ that can be guaranteed. This value $Score(V_0, J)$ is

$$\min\{K \in \mathbb{N} \mid \mathcal{A}(V_0, J) \models \texttt{control: A<> Scheduler.END and V\_acc<=K}\}$$

2. Compute an interval I_1 such that, for $I_2 = [V_1 + m, V_2 - m]$:
   (a) $\forall V_0 \in I_1$, $\mathcal{A}(V_0, I_2) \models$ `control: A<> Scheduler.END`
   (b) the value $Score(I_1) = \max\{Score(V_0, I_2) \mid V_0 \in I_1\}$ is minimal.

3. For each $V_0 \in I_1$, compute a control strategy $\mathcal{S}(V_0)$ for the control objective `A<> Scheduler.END and V_acc<=K` with `K` set to $Score(V_0, I_2)$. This strategy is defined by four dates of start/stop of the pump [12] and, by definition of $Score(V_0, I_2)$, minimizes the accumulated volume.

It is worth noticing that the value *Score* is computed using the variable V_acc which is deduced from intermediary values of variable V. Since V corresponds to the value of the volume with no noise, V_acc represents the *mean value* of the accumulated volume for a given execution.

Results. For a margin $m = 0.4l$ and a granularity of 1 (D=1 in the UPPAAL-TIGA model), we obtain as optimal stable interval the interval $I_1 = [5.1, 10]$. The set of corresponding optimal strategies are represented on Fig. 5.

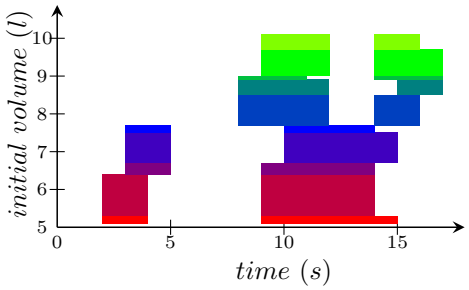


Fig. 5. Strategy for $D = 1$ and $m = 0.4$ l.

For each value of the initial volume in the interval I_1, the corresponding period of activation of the pump is represented. We have represented volumes which share the same strategy in the same color. For the 50 initial possible values of volume, we obtain 10 different strategies (first row of Table 1).

[11] Control objectives are formulated as "control: P" following UPPAAL-TIGA syntax, where P is a TCTL formula specifying either a safety property `A[]`ϕ or a liveness property `A<>`ϕ.

[12] This is easy to obtain these times using the vectors start anf stop of the pump.

The overall strategy we synthesize thus measures the volume just once at the beginning of each cycle and play the corresponding "local strategy" until the beginning of next cycle.

Table 1 represents the results obtained for different granularities and margins. It gives the optimal stable interval I that is computed, (note that it is smaller if we allow a smaller margin or a finer granularity), the number of different local strategies, and the value of worst case mean volume which is obtained as $Score(I)/20$. These strategies are evaluated in sections 4 and 5.

Granularity	Margin	Stable interval	Number of strategies	Mean volume
1	4	$[5.1, 10]$	10	8.45
1	3	$[5.1, 9.8]$	10	8.35
1	2	$[5.1, 9.6]$	9	8.25
1	1	$[5.1, 9.4]$	9	8.2
2	4	$[5.1, 8.9]$	14	8.05
2	3	$[5.1, 8.7]$	14	7.95
2	2	$[5.1, 8.5]$	11	7.95
2	1	$[5.1, 8.3]$	11	7.95

Table 1. Main Characteristics of the Strategies Synthesized with UPPAAL-TIGA.

4 Checking Correctness and Robustness of Controllers

In this section, we report on the results concerning the verification of the correctness robustness of the three solutions mentioned in the previous sections. To analyze the correctness and the robustness of the three controllers, we use the tool PHAVER [4, 3] for analysing hybrid systems. PHAVER allows us to consider a rich continuous time model of the system where we can take into account the fluctuations of consumption of the machine as well as adequate models of imprecisions inherent to any real implementation. The PHAVER model of the cycle together with the pump is given in Appendix A.1. This model takes into account the fluctuations in the consumption rate of the machine as well the imprecision on the measure of the volume. We now summarize the results for the three controllers.

The Bang-Bang controller. The PHAVER automaton of the Bang-Bang controller is given in Appendix A.2. This automaton turns *on* the pump when a floor volume value is reached and turns *off* the pump when a ceiling value is reached. To ensure robustness and implementability of this control strategy, we introduce imprecision in the measure of the oil volume: when the volume is read it may differ at most by $\epsilon = 0.06\ l$ from the actual value (precision of the sensor). Tuning this controller amounts to choose the tightest values for this floor and

ceiling. In our experiment we found that 5.76 and 25.04 are the best margins we can expect.

With this PHAVER model and the previous margins[13], we are able to show that: (1) this control strategy enforces the safety requirement R_1, i.e. the volume of oil stays within the bounds $[4.9; 25.1]$; (2) the set of reachable states for initial volume equal to 10 l can be computed and it is depicted in Fig. 6; this means that this controlled system is "cyclic" from the end of the first cycle on and the same interval $[10.16; 14]$ (for the volume) repeats every other cycle. It is also possible to compute an interval for the cumulated volume over two cycles and for this controller it is 307 and the mean volume is 15.35.

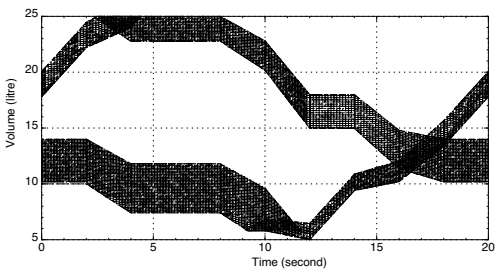


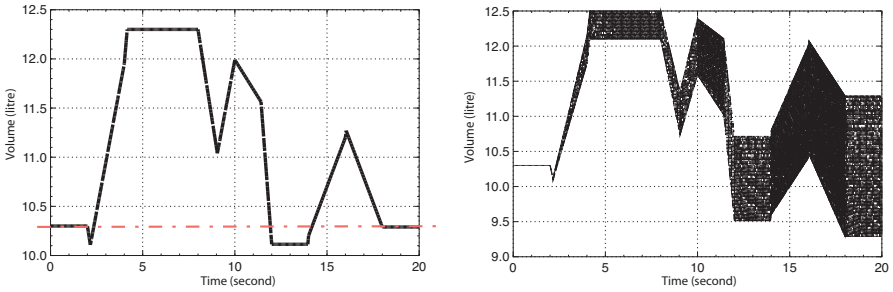
Fig. 6. Cyclic Behavior of the Bang-Bang controller with Noise

The Smart Controller. The Smart Controller designed by HYDAC ELECTRONICS GMBH is specified by a 400 line C program and computes the start/stop dates for the next cycle according to what was observed in the previous cycle: it is not possible to translate it precisely with PHAVER. Instead the PHAVER code of a simplified version of the controller, obtained from observing its behaviour in the absence of noise, is given in Appendix A.3: it turns *on* and *off* so that the pump is active exactly during the three intervals $[2.16; 4.16]$, $[9.05; 11.42]$ and $[13.96; 16.04]$ during each cycle. Indeed using simulation, the engineers of HYDAC ELECTRONICS GMBH had discovered that the behavior of their controllers in the absence of noise was cyclic (stable on several cycles) if they started with an amount of oil equal to 10.3 l. This is confirmed by the simulations we report on at the end in Fig. 9 and by Fig. 7(a), obtained with PHAVER showing that the smart controller stabilizes with no fluctuations in the rate. However, our simplified version of the Smart controller (without imprecision on the dates of start and stop of the pump), is not robust against the fluctuations of the rate: the behavior of the system in the presence of noise is depicted in Fig. 7(b) and it can be shown with our PHAVER models that after four cycles, the safety requirement R_1 can be violated. Unfortunaltey, there is no way of proving the correctness of the *full* Smart controller with PHAVER and SIMULINK only gives an average case. In this sense we cannot trust the Smart controller for ensuring the safety property.

The ideal Smart Controller (no noise on the rate) produces an average cumulated volume of around 221 per cycle i.e. an average volume of 11.05.

Controller Computed with UPPAAL-TIGA. We now study the correctness and robustness of the controller synthesized with UPPAAL-TIGA. This verification

[13] And another suitable piece of PHAVER program for the computations.



(a) Without fluctuations (b) With fluctuations

Fig. 7. Behavior of the HYDAC ELECTRONICS GMBH Smart Controller

phase is necessary because during the synthesis phase we have used a very abstract model of the system and also discrete time. To force robustness and correctness, we have imposed additional requirements on the winning strategies (our inductive property together with the margin). But instead of proving by hand that the model and the objective are giving by construction robust and correct controller, it is more adequate to formally verify this. We summarize here the results of this verification phase.

In the sequel we use the controller for granularity 2 and margin 4: this controller can be seen as 14 different local controllers, each one managing one of the 14 intervals in which the initial volume can be at the beginning of a cycle. We will focus on those strategies here but we have automated the process and the others may be treated along the same lines.

The PHAVER models for those local strategies are obtained automatically from the generic controller depicted on Figure 13 in Appendix A.4. To make sure that our strategies are implementable, we have verified them in presence of fluctuations of the rate consumption and two types of imprecisions: on the date of start and stop of the pump (we use $\delta = 0.01$ second), and on the measure of the initial volume $\epsilon = 0.06$ l. A summary of the verification results is given in Table 2. From this table, we formally prove that our controller is correct and inductive i.e. robust against the three types of imprecisions mentioned above.

5 Simulation and Performances of the Controllers

In this section, we report on results obtained by simulating the three controller types in SIMULINK, with the purpose of evaluating their performance in terms of the accumulated volume of oil.

SIMULINK models of the *Bang-Bang* controller as well as of the *Smart* controller of HYDAC ELECTRONICS GMBH have been generously provided by the company. As for the eight controllers – differing in granularity and margin – synthesized by UPPAAL-TIGA, we have made a RUBY script which takes UPPAAL-TIGA strategies as input and transforms them into SIMULINK's *m*-format.

Controller for:	Range of volume	Correctness	Range of final volume	Robustness
[5.1; 5.2]	[5.0270; 11.8449]	OK	[6.2519; 8.6931]	OK
[5.3; 5.5]	[5.2270; 11.8398]	OK	[5.3519; 7.8651]	OK
[5.6; 5.6]	[4.9740; 11.1320]	OK	[5.6519; 7.9481]	OK
[5.7; 6.3]	[4.9770; 11.8320]	OK	[5.7519; 8.6481]	OK
[6.4; 6.6]	[5.0270; 11.8320]	OK	[5.3519; 7.8485]	OK
[6.7; 6.9]	[4.9733; 11.3320]	OK	[5.6513; 8.1492]	OK
[7; 7.4]	[4.9770; 11.8320]	OK	[5.9479; 8.6480]	OK
[7.5; 7.6]	[5.3480; 11.7320]	OK	[5.3480; 7.7480]	OK
[7.7; 7.7]	[5.0270; 8.4440]	OK	[5.5480; 7.8480]	OK
[7.8; 8.2]	[4.9714; 8.9480]	OK	[5.6453; 8.3520]	OK
[8.3; 8.5]	[4.9770; 9.2440]	OK	[6.1244; 8.6480]	OK
[8.6; 8.6]	[5.2770; 8.6600]	OK	[5.3480; 7.6887]	OK
[8.7; 8.8]	[4.9740; 8.8600]	OK	[5.4480; 7.8480]	OK
[8.9; 8.9]	[4.9700; 8.9600]	OK	[5.6440; 7.9495]	OK

Table 2. Correctness and Robustness of a UPPAAL-TIGA Controller (Granularity 2, margin 4).

Fig. 8 shows the SIMULINK block diagram for simulation of the strategies synthesized by UPPAAL-TIGA. The diagram consist of built-in functions and four subsystems: Consumer, Accumulator, Cycle timer and Pump activation (we ommit the details of the subsystems). The Consumer subsystem defines the flow rates used by the machine with the addition of noise: here the choice of a uniform distribution on the interval $[-\epsilon, +\epsilon]$ with $\epsilon = 0.1l/s$ has been made. The Accumulator subsystem implements the continous dynamics of the accumulator with a specified initial volume ($8.3l$ for the simulations). In order to use the synthesized strategies the volume is scaled with a factor 10, then rounded and feed into a zero-order hold function with a sample time of 20s. This ensures that the volume is kept constant during each cycle, wich is feed into the strategy function. The Pump activation subsystem takes as input the on/off dates from the strategy (for the given input volume of the current cycle) and a Cycle timer, that holds the current time for each cycle.

Now, the plots in Fig. 9 are the result of SIMULINK simulations of the ten controllers, illustrating the volume of the accummulator as well as the state of the pump (on or off) for a duration of 200 s, i.e. 10 cycles. Though the simulations do not reveal the known violation of the safety requirement R_1 in the HYDAC Smart controller case, the simulations yield useful information concerning the performance of the controllers. In particular, the simulations indicate that the accumulated oil volume for all controllers grow linearly with time. Also, there is clear evidence that the strategies synthesized by UPPAAL-TIGA outperform the Smart controller of HYDAC – which is not robust – and also the Bang-Bang controller – which is robust but very non-optimal.

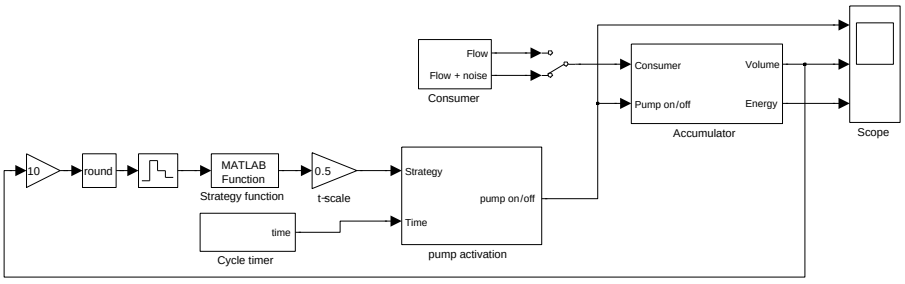


Fig. 8. The overall SIMULINK model.

This is highlighted in Table 3, giving – for each of the ten strategies – the simulation results for the accumulated volume of oil , the corresponding mean volume as well as the worst case mean volume according to synthesis of UPPAAL-TIGA. The table shows – as could be expected – that UPPAAL-TIGA's worst case mean volumes consistently are slightly more pessimistic that their simulation counter-parts. More interestingly, the simulation reveals that the performances of the synthesized controllers (e.g. G2M1) provide a vast improvement both of the Smart Controller of HYDAC ELECTRONICS GMBH (33%) and of the Bang-Bang Controller (45%).

Controller	Acc. volume	Mean volume	Mean volume (TIGA)
Bang-Bang	2689	13.45	-
HYDAC	2232	11.16	-
G1M4	1511	7.56	8.45
G1M3	1511	7.56	8.35
G1M2	1518	7.59	8.25
G1M1	1518	7.59	8, 2
G2M4	1527	7.64	8.05
G2M3	1513	7.57	7.95
G2M2	1500	7.5	7.95
G2M1	1489	7.44	7.95

Table 3. Performance characteristics based on SIMULINK simulations.

6 Conclusion

In this paper we have presented a model-based methodology for the systematic development of robust and near-optimal controllers. The methodology applies a chain of tools for automatic synthesis (UPPAAL-TIGA), verification (PHAVER)

and simulation (SIMULINK). Initially, sufficiently simple and abstract game models are used for synthesis. The correctness and robustness of the strategies are then verified using continuous hybrid models and – finally – the performance of the strategies are evaluated using simulation models.

Applied to the industrial case study provided by HYDAC ELECTRONICS GMBH, our method provides control strategies which outperforms the *Smart* controller as well as the simple *Bang-Bang* controller considered by the company. More important – whereas correctness and robustness of the Smart controller is unsettled – the strategies synthesized by our method are provably correct and robust. We believe that the case study demonstrates the maturity and industrial relevance of our tools.

Directions for further work include:

- Improve the performance of our controller further by optimizing over several cycles, and/or
- Improve the performance of our controller further by adding some predefined points when we can measure the volume (even with imprecision).
- Consideration of other imprecisions, e.g. with respect to the timing of consumer demands.
- Consideration of other optimization criteria. An interesting feature of the *Smart* controller of HYDAC ELECTRONICS GMBH seems to be that the oil volume is kept in a rather narrow interval, a feature which could possibly be benificial for increasing the life-time of the Accumulator.
- Use the emerging version of UPPAAL-TIGA supporting synthesis under partial observability in order to allow more accurate intial game models.

References

1. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *19th Int. Conf. CAV 2007*, volume 4590 of *LNCS*, pages 121–125. Springer, 2007.
2. F. Cassez, A. David, K. G. Larsen, D. Lime, and J.-F. Raskin. Timed control with observation based and stuttering invariant strategies. In *5th Int. Symp. ATVA 2007*, volume 4762 of *LNCS*, pages 192–206. Springer, 2007.
3. G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. *Int. Journal on Software Tools for Technology Transfer (STTT)*, 1(1–2), December 1997. Extended and Revised version of [4].
4. G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *8th Int. Work. HSCC 2005*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.
5. J. J. Jessen, J. I. Rasmussen, K. G. Larsen, and A. David. Guided controller synthesis for climate controller using Uppaal Tiga. In *5th Int. Conf. FORMATS 2007*, volume 4763 of *LNCS*, pages 227–240. Springer, 2007.
6. Simulink, 2008. `http://www.mathworks.com/products/simulink/`.

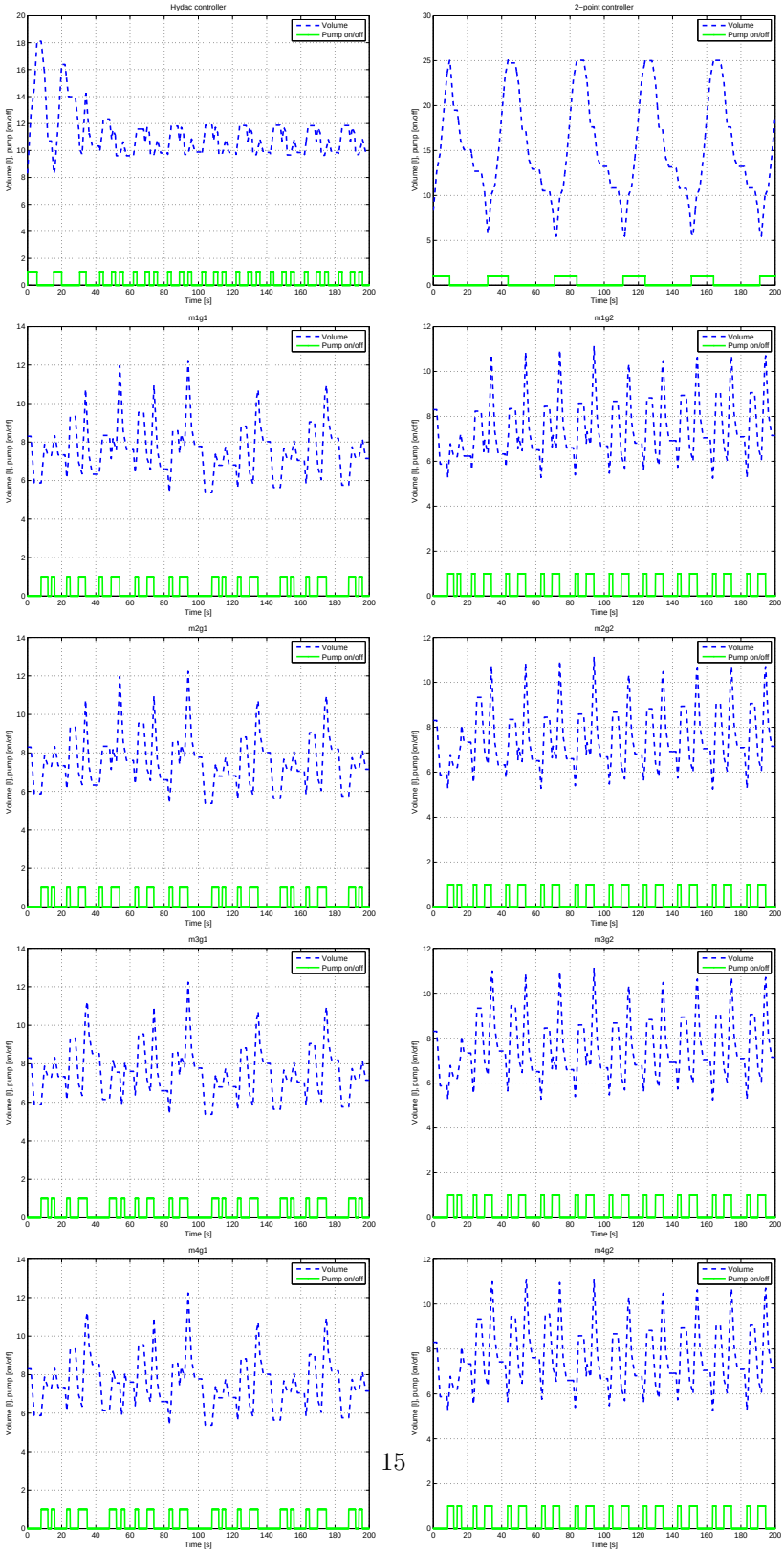


Fig. 9. SIMULINK simulations.

A PHAVer Models

A.1 Model for the cycle and the pump

The PHAVer hybrid automaton `cycle` for the cycle and the pump[14] is given in Fig. 10. This automaton has two synchronisation labels: `switch_off` and `switch_on`. They model the actions of switching on and off the pump and are issued by the controller. The `cycle` is *complete* w.r.t. these actions and so does not constraint their occurrences. This model is cyclic (`t` is reset in locations `l8*`). It also takes into account the fluctuations (variable `f`) of the rate of consumption of the machine when it is consuming (otherwise no fluctuation). Another uncertainty we take into account is on the measure of the initial volume: the initial state is given by an interval $[v_1 - \epsilon; v_2 + \epsilon]$ where ϵ (`eps`) is the uncertainty on the volume measure. If `eps` is 0 we assume the cycle starts with exactly an amount between `v1` and `v2` litres in the accumulator. The states of the automaton `cycle` indicates the current point in the cycle: the phases are numbered from 0 to 8 according to the consumption rate of the machine. When the consumption rate of the machine is 0 there is no fluctuation of the rate: this is why in locations `loc2*` the error `f` is not used in the derivative `v'` of v.

```
1:  rate0:=0; // rates of the machine; rate(i) is the rate of phase i in the schedule
    rate1:=-1.2;
3:  rate2:=0;
    rate3:=-1.2;
5:  rate4:= -2.5;
    rate5:= 0;
7:  rate6:= -1.7;
    rate7:= -0.5;
9:  rate8:= 0;
    pump:=2.2 ; // rate of the pump when on
11: f:=0.1; // fluctuation of uncertainty of the pump+machine rate
    // be careful: do not use when pump+machine are off
13: eps:=0.06 ; // this is the size of the interval around the init volume

15: automaton cycle
    contr_var: t, v; // elapsed time in a cycle and current volume in the pump
17: synclabs: tau, // needed because each trans must a label
              switch_on, switch_off;
19:
    //rate0=0 no noise f
21: loc l0off: while t<=2 wait {t'==1 & v'==rate0}
        when true sync switch_on do {t'==t & v'==v} goto l0on ;
23:     when t==2 sync tau do {t'==t & v'==v} goto l1off ;

25: loc l0on: while t<=2 wait {t'==1 & v'<=rate0+pump & v'>=rate0+pump}
        when true sync switch_off do {t'==t & v'==v} goto l0off ;
27:     when t==2 sync tau do {t'==t & v'==v} goto l1on ;
```

[14] For technical reasons (derivative value `v'`) it is not easy to make two models one for the cycle and one for the pump and synchronize them.

```
30:  loc l1off: while t<=4 wait {t'==1 & v'<=rate1+f & v'>=rate1-f}
         when true sync switch_on do {t'==t & v'==v} goto l1on ;
32:      when t==4 sync tau do {t'==t & v'==v} goto l2off ;

34:  loc l1on: while t<=4 wait {t'==1 & v'<=rate1+pump+f & v'>=rate1+pump-f}
         when true sync switch_off do {t'==t & v'==v} goto l1off ;
36:      when t==4 sync tau do {t'==t & v'==v} goto l2on ;

38:  // rate2=0 no noise f
     loc l2off: while t<=8 wait {t'==1 & v'==rate2}
40:      when true sync switch_on do {t'==t & v'==v} goto l2on ;
         when t==8 sync tau do {t'==t & v'==v} goto l3off ;
42:
     loc l2on: while t<=8 wait {t'==1 & v'<=rate2 + pump & v'>=rate2 + pump }
44:      when true sync switch_off do {t'==t & v'==v} goto l2off ;
         when t==8 sync tau do {t'==t & v'==v} goto l3on ;
46:
     loc l3off: while t<=10 wait {t'==1 & v'<=rate3+f & v'>=rate3-f}
48:      when true sync switch_on do {t'==t & v'==v} goto l3on ;
         when t==10 sync tau do {t'==t & v'==v} goto l4off ;
50:
     loc l3on: while t<=10 wait {t'==1 & v'<=rate3+pump+f & v'>=rate3+pump-f}
52:      when true sync switch_off do {t'==t & v'==v} goto l3off ;
         when t==10 sync tau do {t'==t & v'==v} goto l4on ;
54:
     loc l4off: while t<=12 wait {t'==1 & v'<=rate4+f & v'>=rate4-f}
56:      when true sync switch_on do {t'==t & v'==v} goto l4on ;
         when t==12 sync tau do {t'==t & v'==v} goto l5off ;
58:
     loc l4on: while t<=12 wait {t'==1 & v'<=rate4 + pump +f & v'>=rate4 + pump -f}
60:      when true sync switch_off do {t'==t & v'==v} goto l4off ;
         when t==12 sync tau do {t'==t & v'==v} goto l5on ;
62:
     //rate5=0 no noise f
64:  loc l5off: while t<=14 wait {t'==1 & v'==rate5}
         when true sync switch_on do {t'==t & v'==v} goto l5on ;
66:      when t==14 sync tau do {t'==t & v'==v} goto l6off ;

68:  loc l5on: while t<=14 wait {t'==1 & v'<=rate5 + pump & v'>=rate5 + pump }
         when true sync switch_off do {t'==t & v'==v} goto l5off ;
70:      when t==14 sync tau do {t'==t & v'==v} goto l6on ;

72:  loc l6off: while t<=16 wait {t'==1 & v'<=rate6+f & v'>=rate6-f}
         when true sync switch_on do {t'==t & v'==v} goto l6on ;
74:      when t==16 sync tau do {t'==t & v'==v} goto l7off ;

76:  loc l6on: while t<=16 wait {t'==1 & v'<=rate6 + pump +f & v'>=rate6 + pump -f}
         when true sync switch_off do {t'==t & v'==v} goto l6off ;
78:      when t==16 sync tau do {t'==t & v'==v} goto l7on ;

80:  loc l7off: while t<=18 wait {t'==1 & v'<=rate7 + f & v'>=rate7 - f}
         when true sync switch_on do {t'==t & v'==v} goto l7on ;
82:      when t==18 sync tau do {t'==t & v'==v} goto l8off ;

84:  loc l7on: while t<=18 wait {t'==1  & v'<=rate7 + pump +f & v'>=rate7 + pump-f}
         when true sync switch_off do {t'==t & v'==v} goto l7off ;
86:      when t==18 sync tau do {t'==t & v'==v} goto l8on ;

88:  // rate8=0 no noise
     loc l8off: while t<=20 wait {t'==1 & v'==rate8};
90:      when true sync switch_on do {t'==t & v'==v} goto l8on ;
         when t==20 sync tau do {t'==0 & v'==v} goto l0off;
92:
     loc l8on: while t<=20 wait {t'==1 & v'==rate8+pump};
94:      when t==20 sync tau do {t'==0 & v'==v} goto l0on;
         when true sync switch_off do {t'==t & v'==v} goto l8on ;
96:
      initially : l0off & t==0 & v>=v1-eps & v<=v2+eps ;
98:  end
```

Fig. 10. Cycle and Pump in PHAVer

A.2 Model for the Bang-Bang Controller

The PHAVer code of the Bang-Bang controller is given in Figure 11.

```
1:  // ------------------------------
    // bang bang Controller automaton
3:  // this controller starts in on or off and then
    // switch on or off when a bound is reached
5:  // ------------------------------

7:  eps1:=0.06; // imprecision on the volume measure
    margin_min:=0.86; // best we can do
9:  margin_max:=0.06; // best we can do

11: automaton controller
    input_var: v; // v is given by the cycle+tank automaton
13:
    synclabs: switch_on , switch_off ; // synchronized with the cycle+tank
15:
    loc on: while v <= VMAX - margin_max + eps1 wait {true}
17:     when v>=VMAX-margin_max-eps1 sync switch_off do {true} goto off;

19: loc off: while v>=VMIN+margin_min - eps1 wait {true}
       when v<=VMIN+margin_min+eps1 sync switch_on do {true} goto on;
21:
     initially : off & true ; // values for no noise
23: end
```

Fig. 11. Bang-Bang controller in PHAVer

A.3 Model for the Hydac Electronics Gmbh Smart Controller

The PHAVer model for th Hydac Electronics Gmbh controller is represented on Figure 12.

A.4 Model for the controllers computed with Uppaal-TiGA

We have designed a generic PHAVer model for controllers with 2 starts and 2 stops during one cycle which is given in Figure 13.

For example, the controller for initial volume within $[5.7; 6.3]$ is obtained by setting `starti` and `stopi` with the correct values for this initial volume. In this automaton, there is a variable δ (`delta`) which models the interval in which we issue the start/stop commands: we cannot measure time with infinite accurracy and thus we will only be able to issue the start/stop actions in an interval around the precise time points given by the controller: if the ideal synthesized controller has to issue `switch_on` at 2.5, the implementation of the controller can only ensure it will be issued in $[2.5 - \delta; 2.5 + \delta]$. We use the model for the cycle and pump automaton given Fig. 10 in Appendix A.1. The values v_1 and v_2 are set according to the controller we want to check (e.g. $v_1 = 5.7$ and $v_2 = 6.3$ for the controller which has to be used for the volume within $[5.7; 6.3]$). As our controllers

```
// ----------------------------
// HYDAC smart controller automaton
// this controller starts in off and then
// switch on or off at given time points
// ----------------------------

// time between switch is at least 2

automaton controller

contr_var: t; // this is the time reference of the controller
synclabs: switch_on , switch_off, tau1 ; // synchronized with the cycle+tank

loc off1: while t<=2.16 wait {t'==1}
   when t==2.16 sync switch_on do {t'==t} goto on1;

loc on1: while t<=4.16 wait {t'==1}
    when t==4.16 sync switch_off do {t'==t} goto off2;

loc off2: while t<=9.05 wait {t'==1}
   when t==9.05 sync switch_on do {t'==t} goto on2;

loc on2: while t<=11.42 wait {t'==1}
    when t==11.42 sync switch_off do {t'==t} goto off3;

loc off3: while t<=13.96 wait {t'==1}
   when t==13.96 sync switch_on do {t'==t} goto on3;

loc on3: while t<=16.04 wait {t'==1}
    when t==16.04 sync switch_off do {t'==t} goto last;

loc last: while t<=20 wait {t'==1}
    when t==20 sync tau1 do {t'==0} goto off1;

 initially : off1 & t==0 ;

end
```

Fig. 12. HYDAC ELECTRONICS GMBH Smart Controller in PHAVER

```
1:  // ————————————————————————
    // TIGA controller automaton
3:  // this controller starts in off and then
    // switch on or off at time points defined in another file
5:  // ————————————————————————

7:  // time between switch is at least 2
    delta:=0.01; // this defines the maximum error on
9:               // the date at which start/stop are performed

11: automaton controller

13: contr_var: t; // this is the time reference of the controller
    synclabs: switch_on , switch_off ; // synchronized with the cycle+tank
15:
    loc off1: while t<=start1+delta wait {t'==1}
17:    when t>=start1-delta sync switch_on do {t'==t} goto on1;

19: loc on1: while t<=stop1+delta wait {t'==1}
        when t>=stop1-delta sync switch_off do {t'==t} goto off2;
21:
    loc off2: while t<=start2+delta wait {t'==1}
23:    when t>=start2-delta sync switch_on do {t'==t} goto on2;

25: loc on2: while t<=stop2+delta wait {t'==1}
        when t>=stop2-delta sync switch_off do {t'==t} goto last;
27:
    loc last: while true wait {true} ;
29:
     initially : off1 & t==0 ;
31:
    end
```

Fig. 13. Generic PHAVer Controller with two Start/Stop(s).

should handle all the possible values of the volume at the beginning of a cycle, as well as to be robust w.r.t. errors in the volume measurement, we add a variable (ϵ) `eps` which models this error: it means we use the controller for $[5.7; 6.3]$ on an larger interval which is given by $[5.7 - \epsilon; 6.3 + \epsilon]$. Still our controller should ensure that the final volume is within $[5.1; 8.9]$. To ensure overlapping and full coverage of the initial volume range, we need to set ϵ larger than 0.05: we choose 0.06 for the following experiments[15]. To validate the controller synthesized with UPPAAL-TIGA we check the following:

1. we set $\delta = 0.01$ second, $\epsilon = 0.06$, and the maximum rate fluctuation is $f = 0.1$;
2. we check that the set of reachable states of each of the 14 controllers is within $[V_{min}; V_{max}]$ which is the safety requirement of the accumulator;
3. we check that, starting from $I_\epsilon = [5.1 - \epsilon; 8.9 + \epsilon]$ the final values of the volume are within the interval $[5.1; 8.9]$. Thus we have an inductive proof that our controller is safe and robust w.r.t. triple (δ, ϵ, f).

These computations can be done using the following PHAVER program (Fig. 14):

[15] If the real volume is 5.65, we may obtain a measure of 5.7 or 5.6: what we check is that both the controllers for 5.7 and 5.6 will ensure the final volume is the interval $[5.1; 8.9]$.

```
REACH_CONSTRAINT_LIMIT = 48;
REACH_CONSTRAINT_TRIGGER = 96;
CONSTRAINT_BITSIZE = 24;
REACH_BITSIZE_TRIGGER = 200;

// Other constants
REFINE_DERIVATIVE_METHOD = 0; // say 'constrained based' in the manual
PARTITION_PRIORITIZE_ANGLE = false; // when partitioning use the spread angle
PARTITION_DERIV_MINANGLE = 0.99 ; // spread angle almost 1 means almost derivative
REACH_USE_CONVEX_HULL=true;
REACH_USE_BBOX=true;

// the system is
sys = cycle & controller ;

// parameters for refinement
m1:=0.05;
m2:=0.1;

sys.add_label(tau_sys); // new label for refinement
sys.set_refine_constraints((t,m1,m2),(v,m1,m2),tau_sys);

bad=sys.{$ & v<VMIN, $ & v>VMAX};

// now compute the reach
echo "Now computing the state space ... ";
reach=sys.reachable;
r=reach;
r1=reach;

reach.intersection_assign(bad);
echo "checking wether the safety bounds are met";
reach.is_empty;

vmintarget:=5.1;
vmaxtarget:=8.9;
r2=sys.{$ & t==20 & v<vmintarget, $ & t==20 & v>vmaxtarget};
r1.intersection_assign(r2);
echo "checking wether the final interval is within the initial";
r1.is_empty ;
```

Fig. 14. PHAVer Program to Check Correctness and Robustness.