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, Aix-Marseille Universités & 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 company in the context of a European research project Quasimodo*.

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

* Work supported by the projects: (i) Quasimodo: "Quantitative System Properties in Model-Driven-Design of Embedded", `http://www.quasimodo.aau.dk/`, (ii) Gasics: "Games for Analysis and Synthesis of Interactive Computational Systems", `http://www.ulb.ac.be/di/gasics/`, and (iii) Moves: "Fundamental Issues in Modelling, Verification and Evolution of Software", `http://moves.vub.ac.be`.

** This author is supported by a Marie Curie International Outgoing Fellowship within the 7th European Community Framework Programme.

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.

In a recent work [7], 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 [8] for simulation. To solve the HYDAC control problem, we use three complementary tools for three different purposes: UPPAAL-TIGA for synthesis, PHAVER [5, 4] 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.

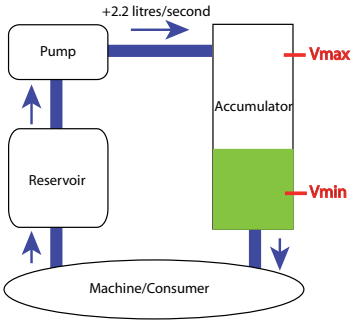


Fig. 1. Overview of the System.

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 a simplified 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 reaches a floor value and turned off when the volume of oil reaches a ceiling value. The second control strategy is a strategy designed by the engineers at HYDAC with the help of SIMULINK.

Structure of the paper. In section 2, we present the HYDAC 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

In this section, we describe the components of the HYDAC case study using hybrid automata notations. Then we explain the control objectives for the system to design.

The Machine. The oil consumption of the machine is cyclic. One cycle of consumptions, as given by HYDAC, is depicted in Fig. 2(d). Each period of consumption is characterized by a rate of consumption m_r (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). 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 $c\,l/s$ (with $c > 0$), in reality it always lies within that period in the interval $[c - \epsilon, c + \epsilon]$, where ϵ is fixed to $0.1\ l/s$. This property is noted F.

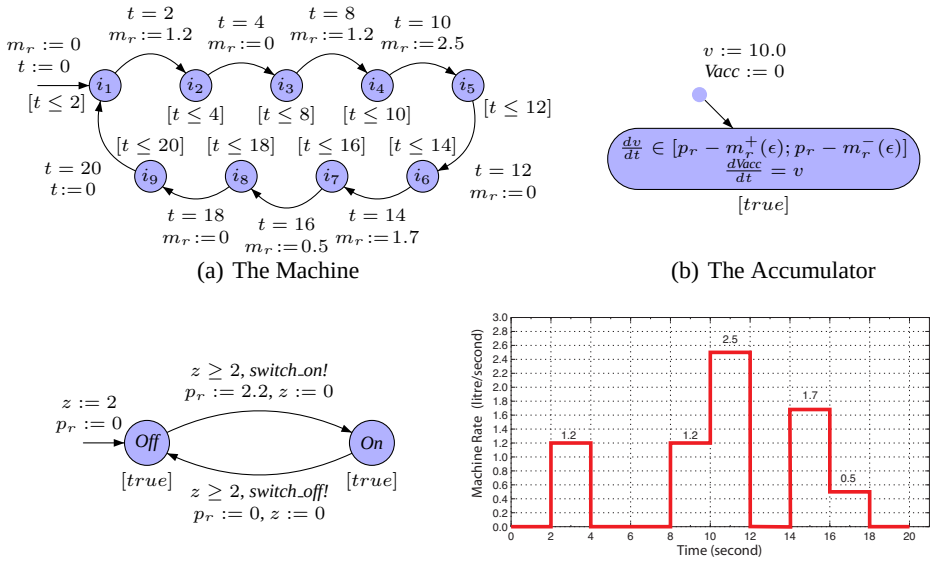
To model the machine, we use a timed automaton with 2 variables. The discrete variable m_r models the consumption rate of the machine, and the clock t is used to measure time within a cycle. The variable m_r is shared with the model of the accumulator. The timed automaton is given in Fig. 2(a). The noise on the rate of consumption is modeled in the model for the accumulator.

The Pump. 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 $2.2l/s$. We model the pump with a two states timed automaton given in Fig. 2(c) with two variables. The discrete variable p_r models the pumping rate of oil of the pump, and is shared with the accumulator. The clock z ensures that 2 t.u. have elapsed between two switches.

The Accumulator. To model the behavior of the accumulator, we use a one state hybrid automaton given in Fig. 2(b) that uses four variables. The variable v models the volume of oil within the accumulator, its evolution depends on the value of the variables m_r (the rate of consumption depending of the machine) and p_r (the rate of incoming oil from the pump). To model the imprecision on the rate of the consumption of the machine, the dynamics of the volume also depends on the parameter ϵ and is naturally given by the differential inclusion $dv/dt \in [p_r - m_r^-(\epsilon), p_r - m_r^+(\epsilon)]$ with $m_r^{\bowtie}(x) = m_r \bowtie x$ if $m_r > 0$ and m_r otherwise. The variable *Vacc* models the accumulated volume of oil along time in the accumulator. It is initially equal to 0 and its dynamic is naturally defined by the equation $d\mathit{Vacc}/dt = v$.

The Control Problem. The controller must operate the pump (switch it *on* and *off*, respecting the latency constraint) 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 $V_{max} = 25.1l$;



(a) The Machine (b) The Accumulator

(c) The Pump (d) Cycle of the Machine

Fig. 2. Hybrid Automaton Model of the System.

– (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 $\int_{t=0}^{t=T} v(t)dt$, that is $V_{acc}(T)$, is minimal for a given operation period T.

While (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.

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(d) 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[5].

Additional Requirements on the Controller. When designing a controller, we must decide what are the possible actions that the controller can take. Here are some considerations about that. First, as 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). Second, as the consumption of the machine has a cyclic behavior, the controller should use this information to optimize the level of oil. So, it is natural to allow the controller to take control decisions at predefined instants during the cycle. Finally, we want a robust solution in the sense that if the controller has to turn *on* (or *off*) the pump at time t, it can do it a little before or after, that is at time $t \pm \Delta$

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

for a small Δ without impairing safety. This robustness requirement will be taken into account in the synthesis and verification phases described later.

Two existing solutions. 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 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 thus a simple two-point controller, but it does not exploits the timing information about the consumption periods within a cycle.

To obtain better performances in term of energy consumption, engineers at HYDAC have designed a controller that exploit this timing. This second controller is called the *Smart controller*. This controller was designed by HYDAC, and works as follows [6]: in the first cycle the Bang-Bang controller is used and the volume $v(t)$ is measured and recorded every 10ms. According to the sampled values $v(t)$ computed in the initial cycle, an optimization procedure computes the points at which to start/stop the pump on the new cycle (this optimization procedure was given to us in the form of a C code executable into SIMULINK; unfortunately we do not have a mathematical specification of it). On this next cycle the values $v(t)$ are again recorded every 10ms which is the basis for the computation of the start/stop commands for the next cycle. If the volume leaves a predefined safety interval, the Bang-Bang controller is launched again. Though simulations of SIMULINK models developed by HYDAC reveal no unsafe behaviour, the engineers have not been able to verify its correctness and robustness. As we will see later, this strategy (we use the switching points in time obtained with SIMULINK when the C code is run) is not safe in the long run in presence of noise.

3 The UPPAAL-TIGA Model for Controller Synthesis

The hybrid automaton model presented in the previous section can be interpreted as a game in which the controller only supervises the pump. 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 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 timed 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 [3] 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 hybrid automaton of the HYDAC case study. First, to keep the game model simple enough and to remain in a decidable framework[6], 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. Note that since the discretization impacts both the controller and the environment, it is neither an over- nor an under-approximation of the hybrid game model and thus we can not deduce directly the correctness of our controllers. However, our methodology includes a verification step based on PHAVER which allows us to prove this correctness. 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; we thus only measure the oil once per cycle, leading to more simple controllers.
- 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 roughly represents how much the volume can deviate because of these imprecisions. We will consider values in range $[0.1; 0.4]l$.

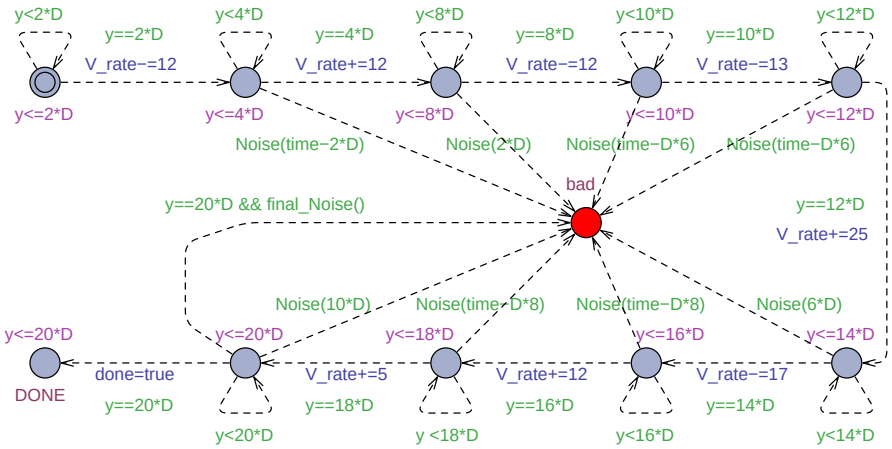
Global Variables. First, we discretize the time w.r.t. ratio stored in variable D, such that D time units represent one second. Second, we represent the current volume of oil by the variable V. We consider a precision of $0.1l$ and thus multiply the value of the volume by 10 to use integers. This volume evolves according to a rate stored in variable V_rate and the accumulated volume is stored in the variable 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 middle 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 additional 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)

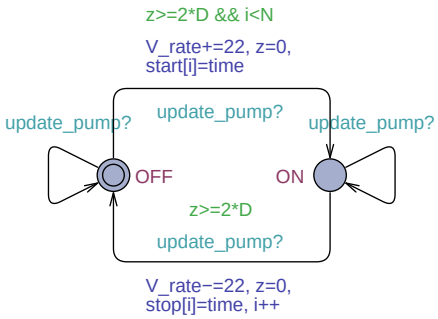
[6] The existence of winning strategies for timed games with costs in undecidable, see [2].

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

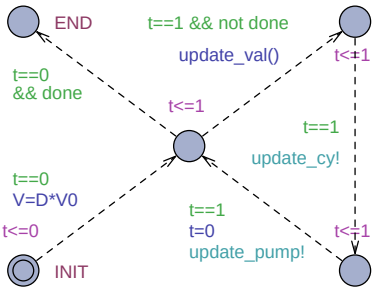
[8] For robustness, we restrain safety constraints of $0.1\ l$.



(a) Model of the cyclic consumption of the machine



(b) Model of the pump



(c) Model of the scheduler

Fig. 3. UPPAAL-TIGA models.

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 it does not give any information about the fluctuation.

The Model of the Pump. The model for the pump is represented on Fig. 3(b) and is very similar to the timed automaton given on Fig. 2(c). Note that the transitions are all controllable (plain arrows) and 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. Moreover, 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/deactivation of the pump are stored in two vectors start and stop.

```
bool Noise(int s){
// s is the number of t.u. of consumption
 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

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[9] and update_pump. When the end of the cycle of the machine is reached, the corresponding automaton sets the boolean variable done to true, which 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 restricted to 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 explicitly 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 control objective (for some fixed margin $m \in \mathbb{Q}_{>0}$) :

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 supremum, over $V_0 \in I_1$ and over the strategies satisfying (i), of the accumulated volume 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 that even in presence of imprecisions, the final volume, if it does not belong to I_2, belongs to I_1: this is the reason why we fix a strict-subinterval of I_1 as a target in the synthesis phase.

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

We now describe a procedure to compute an interval verifying Property $(*)$, and the associated strategies. We proceed as follows[10]:

1. For each $V_0 \in [4.9; 25.1]$, and target final interval $J \subseteq [4.9; 25.1]$, compute (by a binary search) the minimal accumulated volume $Score(V_0, J)$ that can be guaranteed. This value $Score(V_0, J)$ is

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

2. Compute an interval $I_1 \subseteq [4.9; 25.1]$ 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<> Sched.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<> Sched.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 [11] 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. 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). 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.

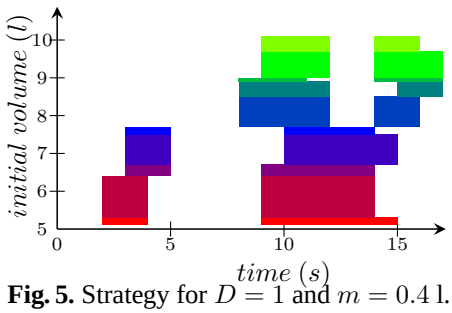


Fig. 5. Strategy for $D = 1$ and $m = 0.4$ l.

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.

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

[10] 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.

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

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.

correctness and the robustness of the three controllers, we use the tool PHAVER [4, 5] for analysing hybrid systems. Robustness is checked according to the type of controller we use: for the Bang-Bang controller, it amounts to saying that the volume cannot be measured accurately and also that the rate fluctuates ($\pm 0.1 l/s$); for the Smart controller, robustness against rate fluctuation cannot be checked; for our synthesized controller, we take into account the rate fluctuation, the imprecision on the measure of the volume and the imprecision on the measure of time.

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 models used in this paper are available in the extended version of this paper on the authors' webpages. 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.

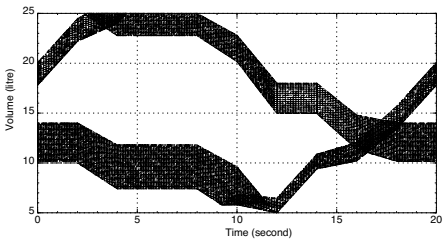


Fig. 6. Cyclic Behavior of the Bang-Bang controller with Noise

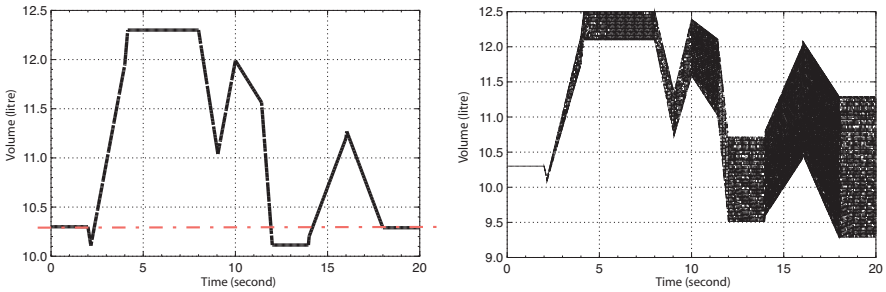
The Bang-Bang controller. To ensure robustness and implementability of this control strategy, we introduce imprecision in the measure of the oil volume: when the volume is read it may differ at most by $\epsilon = 0.06\ l$ from the actual value (precision of the sensor). Tuning this controller amounts 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[12], 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

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

volume) repeats every other cycle. It is thus possible to compute (with PHAVER) the interval of the accumulated volume over the two cycles: for this controller, the upper bound (worst case) is 307 and the mean volume is $307/20 = 15.35$.

The Smart Controller. The Smart Controller designed by HYDAC 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 (see end of section 2). This controller requires to sample the plant every 10ms in order to compute the strategy to apply in the next cycle: although it is theoretically possible to specify this controller in PHAVER, this would require at least 100×20 discrete locations to store the sampled data in the previous cycle. It is thus not realistic to do this as PHAVER would not be able to complete an analysis of this model in a reasonable amount of time. Instead we have built the PHAVER controller that corresponds to the behaviour of the smart controller in a stationary regime, and in the absence of noise. 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 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. 10 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. Unfortunately, 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 accumulated volume of around 221 per cycle i.e. an average volume of 11.05.



(a) Without fluctuations (b) With fluctuations

Fig. 7. Behavior of the HYDAC Smart Controller

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

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/stop of the pump (we use $\Delta = 0.01$ second), and on the measure of the initial volume, the imprecision being $0.06\ l$. Fig. 8 shows how the volume is controlled over 3 cycles: after the first one at $t = 20$, we measure the real volume with uncertainty ($0.06\ l$) and use the corresponding controller from 20 to 40 and for 40 we again switch to another one.

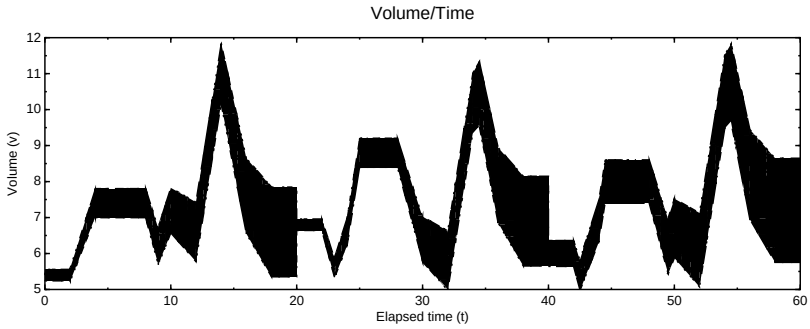


Fig. 8. The Pump Controlled over 3 Cycles with the UPPAAL-TIGA Controller

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 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.

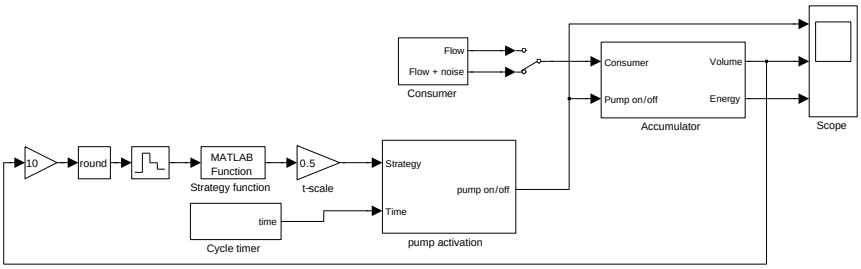
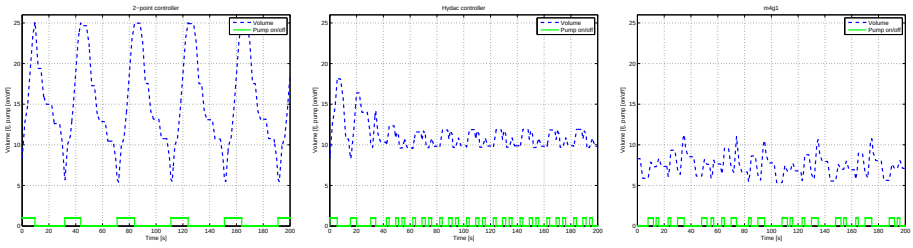


Fig. 9. The overall SIMULINK model.



(a) Bang-Bang controller (b) Smart Controller (c) Synthesized Controller

Fig. 10. The three controller types with SIMULINK

Fig. 9 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 omit 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 continuous 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, which 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. 10 are the result of SIMULINK simulations of the controllers, illustrating the volume of the accumulator 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.

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 2. Performance characteristics based on SIMULINK simulations.

This is highlighted in Table 2, 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 (33%) and of the Bang-Bang Controller (45%).

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, 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 seems to be that the oil volume is kept in a rather narrow interval, a feature which could possibly be beneficial 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 initial 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. Th. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *3rd Int. Conf. FORMATS'05*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.
3. 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.
4. 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 [5].
5. 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.
6. Holger Hermanns, Kai Sven Mittermüller, Teun van Kuppeveld, Jan Storbank Pedersen, and Poul Hougaard. Preliminary descriptions of case studies. Quasimodo Deliverable D5.2, v1.0, July 2008. Confidential Document.
7. 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.
8. Simulink, 2008. `http://www.mathworks.com/products/simulink/`.