Chapter 1
Timed Controller Synthesis: An Industrial Case Study

Franck Cassez, Kim Larsen, Jean-François Raskin, and Pierre-Alain Reynier

1.1 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. The foundations of automatic synthesis of discrete and timed controllers have been presented in the preceding chapter 1. In this chapter, we illustrate the application of these approaches with an industrial case study provided by the HYDAC ELECTRONIC GMBH company in the context of the European research project Quasimodo [7]. We present in the sequel how to use (in a systematic way) the tool UPPAAL-TIGA [1] (see chapter 1 for an introduction to the tool) for the synthesis, together with tools for the verification and the simulation of a provable correct and near optimal controller for real industrial applications.

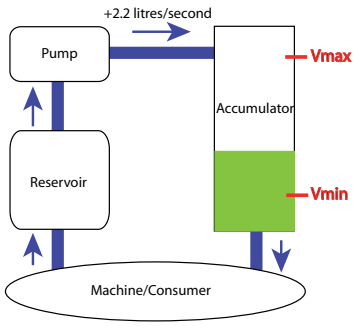


Fig. 1.1 Overview of the system.

The system to be controlled is depicted in Fig. 1.1 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) which is controlled using the pump must be maintained within 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 (a lower level of oil, and thus a lower gas pressure, reduces the wear on the pump and minimizes energy consumption). The maximum output rate of the pump is 2.2litre/sec, whereas the maximum rate of the consumer is 2.5litre/sec: proper operation of the pump should thus anticipate the maximum rate of the machine and pump oil in in advance to ensure that the pressure stays with the given bounds.

To solve the HYDAC ELECTRONIC GMBH control problem, we use three complementary tools for three different purposes: UPPAAL-TIGA for synthesis, the tool PHAVER [5, 4] for verification, and SIMULINK [8] for simulation. In this chapter, we are mainly interested in the synthesis step. For this phase, we show how to construct a (game) model of this 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, because it is robust.

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 computed on 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 [5, 4] 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 ELECTRONIC GMBH with the help of SIMULINK.

Structure of the chapter

In section 1.2, we present in more details the HYDAC ELECTRONIC GMBH control problem. In section 1.3, 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 1.4, 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 1.5, we analyze and compare the performances in term of mean volume of the three controllers using SIMULINK.

1.2 The Oil Pump Control Problem

In this section, we describe the components of the HYDAC ELECTRONIC GMBH case study and describe the different constraints they must satisfy. In addition, we provide a first model of these components, using hybrid automata notations. Though we will not use these models in the sequel to synthesize controllers (synthesis on real hybrid systems is very costly and beyond the scope of any tool), they offer a very precise specification of the system. Then we explain the control objectives for the system to design.

1.2.1 The Machine

The oil consumption of the machine is cyclic. One cycle of consumption, as given by HYDAC ELECTRONIC GMBH, is depicted in Fig. 1.5. Each period of consumption is characterized by a rate of consumption m_r (expressed as a number of litres per second), a time 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 $[c-\varepsilon, c+\varepsilon]$, where ε is fixed to $0.1\ l/s$. This property is noted F.

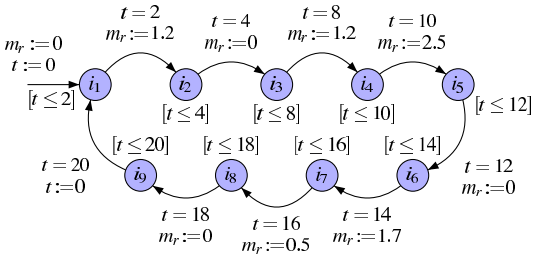


Fig. 1.2 The Machine

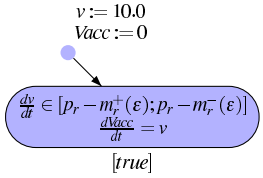


Fig. 1.3 The Accumulator

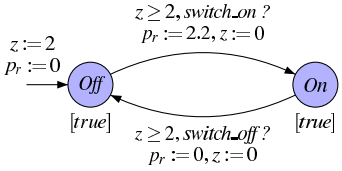


Fig. 1.4 Model of the pump

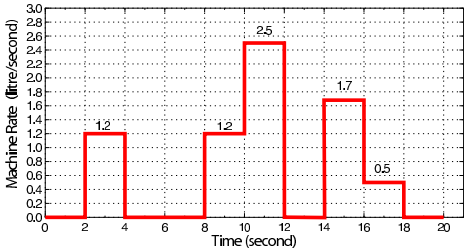


Fig. 1.5 Model of the scheduler

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 variable 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. 1.2. The noise on the rate of consumption is modeled in the model for the accumulator (see Fig. 1.3). We indicate for each location of the timed automaton its invariant in square brackets. For instance, one can stay in location i_1 as long as the value of the clock t is less than or equal to 2.

1.2.2 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-state timed automaton given in Fig. 1.4 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.

1.2.3 The Accumulator.

To model the behavior of the accumulator, we use a one-state hybrid automaton given in Fig. 1.3 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 $dv/dt \in [p_r - m_r^-(\varepsilon), p_r - m_r^+(\varepsilon)]$ 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 $dVacc/dt = v$.

1.2.4 The Control Problem

The controller must operate the pump (switch it *on* and *off*, respecting the latency constraint) to ensure the 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 $\int_{t=0}^{t=T} v(t)dt$, 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. 1.5 between 10 and 12), some extra amount of oil must be present in the accumulator before that period of consumption to ensure that the amount of oil stays above V_{min} (requirement R_1) is not violated[1].

1.2.5 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 of the machine may deviate slightly from the ideal values (this is called *noise*),

[1] It might be too late to switch the pump on when the volume reaches V_{min} and so the controller may have to anticipate.

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 the pump *on* (or *off*) at time t, it can do it a little before or after, that is at time $t \pm \Delta$ for a small Δ without impairing safety. This robustness requirement will be taken into account in the synthesis and verification phases described later.

1.2.6 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 ELECTRONIC 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 thus a simple two-point controller, 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 ELECTRONIC GMBH have designed a controller that exploit this timing. This second controller is called the *Smart controller* and works as follows [6]: in the first cycle the Bang-Bang controller is used and the pressure $p(t)$ is measured, the corresponding volume $v(t)$ is computed 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 next 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 $p(t)$ are again recorded every 10ms which is the basis for the computation of the start/stop commands for the next cycle and so on. If the pressure leaves a predefined safety interval, the Bang-Bang controller is launched again. Though simulations of SIMULINK models developed by HYDAC ELECTRONIC GMBH reveal no unsafe behaviour, the engineers have not been able to verify neither its correctness nor its 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.

1.3 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 automatically synthesize, 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 (see chapter 1 for a presentation of the tool).

1.3.1 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.

1.3.2 The UPPAAL-TIGA *Model*

We describe in the next paragraphs how we have obtained our game model for the hybrid automaton of the HYDAC ELECTRONIC GMBH case study. First, to keep the game model simple enough and to remain in a decidable framework[2], 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

[2] The existence of winning strategies for enriched timed games with extra cost variables in undecidable, see [2].

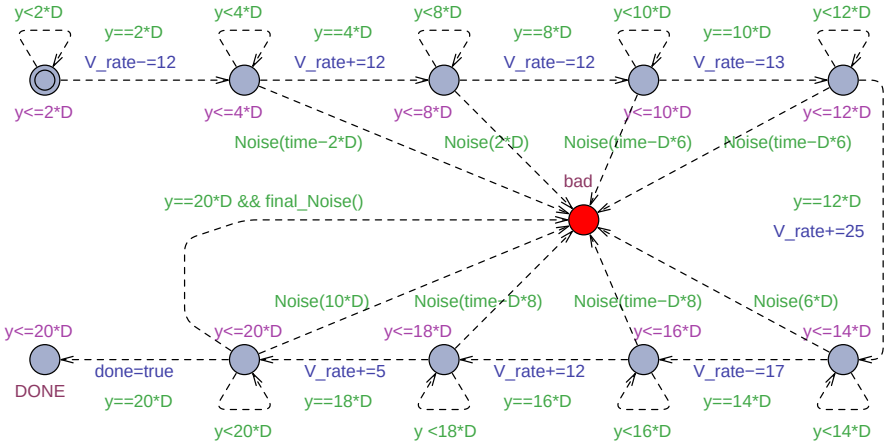


Fig. 1.6 Model of the cyclic consumption of the machine

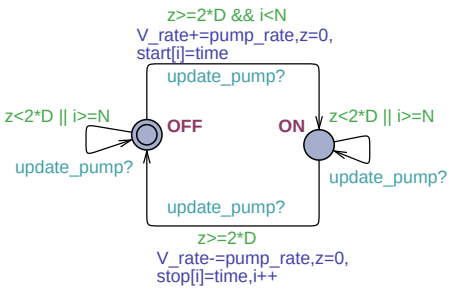


Fig. 1.7 Model of the pump

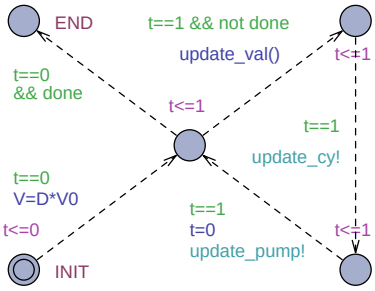


Fig. 1.8 Model of the scheduler

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 imprecision on measures of volume or time, we consider some margin parameter m which roughly represents how much the volume can deviate because of these imprecision in the measure. We will consider values in range $[0.1; 0.4] litre$ for measurement imprecision.

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[3]. 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. 1.6. 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 of the machine. This noise is not related with the parameter m introduced previously as it concerns some possible fluctuations in the consumption rate of the machine around its nominal rate. The function Noise (Fig. 1.9) checks whether the value of V, if modified by these fluctuations in the consumption rate of the machine, may be outside the safe interval $[V_{min}+0.1, V_{max}-0.1]$ [4]. The function final_Noise (Fig. 1.9) 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. 1.7 and is very similar to the timed automaton given on Fig. 1.4. 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

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

[4] For robustness, we restrain safety constraints to $m = 0.1\ l$.

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(){
// 20*D t.u. of consumption in 1 cycle
 return (V-20*D<V1F*D|V+20*D>V2F*D);}
void update_val(){
  int V_pred = V;
  time++;
  V+=V_rate;
  V_acc+=V+V_pred;
}
```

Fig. 1.9 Functions embedded in UppAal Tiga models

The Model of the Scheduler

We use a third automaton represented on Fig. 1.8 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[5] 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 $\mathscr{A}$ the automaton obtained by the composition of the three automata described above. 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 $\mathscr{A}(V_0, I_2)$ the composed system.

1.3.3 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 1.2. Indeed, R_1 is

[5] We did not represent this synchronization on Fig. 1.6 to ease the reading.

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 $m \in \mathbb{Q}_{>0}$) :

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

(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 imprecision, 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.

We now describe a procedure to compute an interval verifying Property $(*)$, and the associated strategies. We proceed as follows[6]:

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 $\mathsf{Score}(V_0, J)$ that can be guaranteed. This value $\mathsf{Score}(V_0, J)$ is

   $\min\{K \in \mathbb{N} \mid \mathscr{A}(V_0, J) \models$ `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. $\forall V_0 \in I_1$, $\mathscr{A}(V_0, I_2) \models$ `control: A<> Sched.END`
   b. the value $\mathsf{Score}(I_1) = \max\{\mathsf{Score}(V_0, I_2) \mid V_0 \in I_1\}$ is minimal.

3. For each $V_0 \in I_1$, compute a control strategy $\mathscr{S}(V_0)$ for the control objective `A<> Sched.END and V_acc<=K` with K set to $\mathsf{Score}(V_0, I_2)$. This strategy is defined by four dates of start/stop of the pump [7] and, by definition of $\mathsf{Score}(V_0, I_2)$, 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.

[6] 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.

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

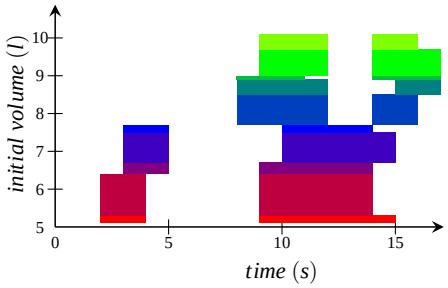


Fig. 1.10 For a margin parameter $m = 0.4l$ and a granularity of 1 (D=1 in the UPPAAL-TIGA model), we obtain as optimal stable interval the interval $I_1 = [5.1, 10]$. The corresponding optimal strategies are represented opposite.

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 corresponding optimal strategies are represented on Fig. 1.10. 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 1.1). The overall strategy we synthesize thus measures the volume just once at the beginning of each cycle and plays the corresponding "local strategy" until the beginning of next cycle.

Gran.	Margin	Stable interval	Nb 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 1.1 Main characteristics of the strategies synthesized with UPPAAL-TIGA.

Table 1.1 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 $\mathsf{Score}(I)/20$. These strategies are evaluated in sections 1.4 and 1.5.

1.4 Checking Correctness and Robustness of Controllers

In this section, we address the verification of the correctness and 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, 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 because we do not have a precise mathematical model of the algorithm designed by the HYDAC ELECTRONIC GMBH engineers; 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 imprecision inherent to any real implementation. The PHAVER models of the controllers are given below and the other models of the cycle and machine are the timed automata of Fig. 1.7 and Fig. 1.6. Our models take into account the fluctuations in the consumption rate of the machine as well the imprecision on the measure of the volume. We now review the results for the three controllers.

1.4.1 The Bang-Bang controller

The PHAVER code of the Bang-Bang controller is given in Fig. 1.11. 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 by at most $\varepsilon = 0.06\ l$ from the actual value (precision of the sensor). Tuning this controller amounts to choosing the tightest values for the floor and ceiling values at which the controller switches the pump (from *on* to *off* or the other way). 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[8], 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. 1.12; 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 thus possible to compute (with PHAVER) the interval of the accumulated volume over the two cycles: for this controller, the upper bound is 307 and the mean volume is $307/20 = 15.35$.

[8] And another suitable piece of PHAVER program to perform the needed computations.

```
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. 1.11 Bang-Bang controller in PHAVER

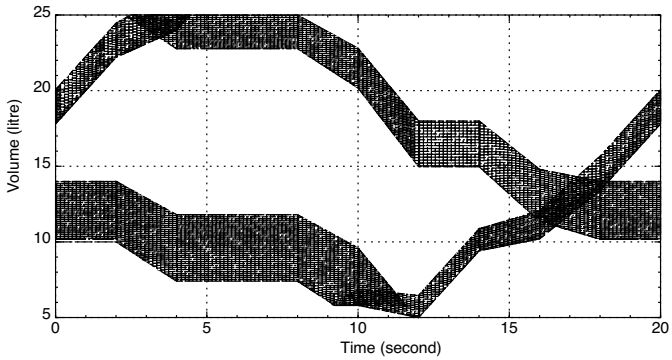


Fig. 1.12 Cyclic Behavior of the Bang-Bang controller with Noise

1.4.2 The Smart Controller

The Smart Controller designed by HYDAC ELECTRONIC GMBH is specified by a 400 line C program and computes the start/stop timestamps for the next cycle according to what was observed in the previous cycle (see end of section 1.2). 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.

```
1:  // ————————————————————————
    // HYDAC smart controller automaton
3:  // this controller starts in off and then
    // switch on or off at given time points
5:  // ————————————————————————

7:  // time between switch is at least 2

9:  automaton controller

11: contr_var: t; // this is the time reference of the controller
    synclabs: switch_on , switch_off, tau1 ; // synchronized with the cycle+tank
13:
    loc off1: while t<=2.16 wait {t'==1}
15:   when t==2.16 sync switch_on do {t'==t} goto on1;

17: loc on1: while t<=4.16 wait {t'==1}
      when t==4.16 sync switch_off do {t'==t} goto off2;
19:
    loc off2: while t<=9.05 wait {t'==1}
21:   when t==9.05 sync switch_on do {t'==t} goto on2;

23: loc on2: while t<=11.42 wait {t'==1}
      when t==11.42 sync switch_off do {t'==t} goto off3;
25:
    loc off3: while t<=13.96 wait {t'==1}
27:   when t==13.96 sync switch_on do {t'==t} goto on3;

29: loc on3: while t<=16.04 wait {t'==1}
      when t==16.04 sync switch_off do {t'==t} goto last;
31:
    loc last: while t<=20 wait {t'==1}
33:   when t==20 sync tau1 do {t'==0} goto off1;

35:  initially : off1 & t==0 ;

37: end
```

Fig. 1.13 HYDAC ELECTRONIC GMBH Smart Controller in PHAVER

Instead we have built the PHAVER controller (given in Fig. 1.13) 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 ELECTRONIC 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. 1.19 and by Fig. 1.14, obtained with PHAVER showing that the smart controller stabilizes with no fluctuations in the rate.

However, our simplified version of the Smart controller given in Fig. 1.13 (without imprecision on the timestamps 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. 1.15 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,

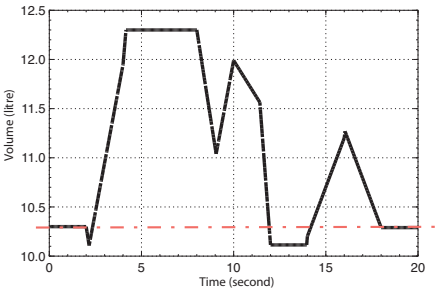


Fig. 1.14 Smart Controller / no fluctuations

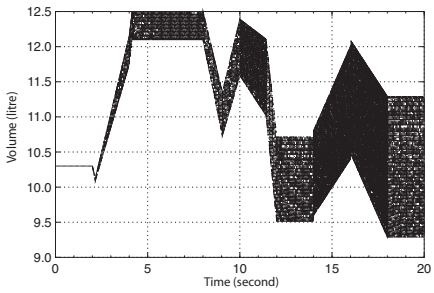


Fig. 1.15 Smart Controller / fluctuations

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.

1.4.3 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). Instead of proving by hand that the model and the objective yield to a correct and robust controller we perform a formal post-check of the controller in the presence of noise and imprecision. 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 imprecision: on the time-stamp 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. 1.16 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.

We have designed a generic PHAVER model for controllers with 2 starts and 2 stops during one cycle which is given in Fig. 1.17.

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

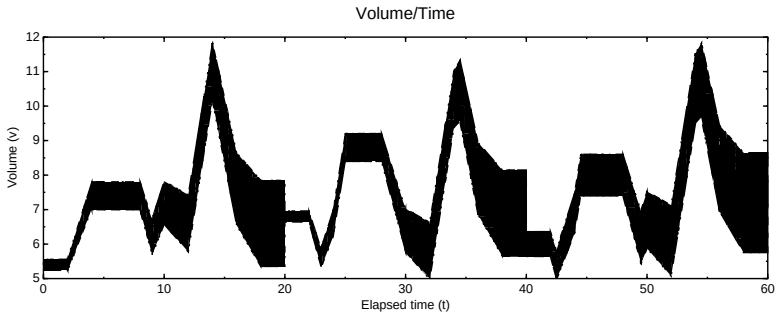


Fig. 1.16 The Pump Controlled over 3 Cycles

```
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. 1.17 Generic PHAVER Controller with two Start/Stop(s).

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. 1.4 and 1.5. The values v_1 and v_2 are set according to the controller we

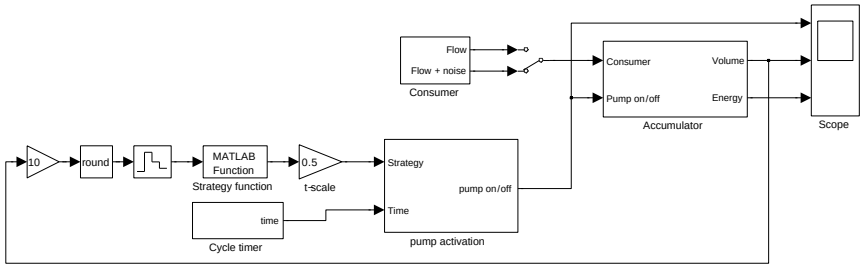


Fig. 1.18 The overall SIMULINK model.

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 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 a larger interval which is given by $[5.7-\varepsilon;6.3+\varepsilon]$. 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[9]. To validate the controller synthesized with UPPAAL-TIGA we check the following:

1. we set $\delta = 0.01$ second, $\varepsilon = 0.06$, 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 $I_\varepsilon = [5.1-\varepsilon;8.9+\varepsilon]$ 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).

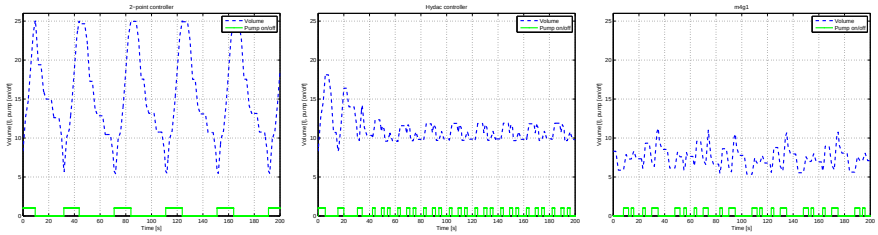
1.5 Simulation and Performances of the Controllers

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 ELECTRONIC 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.

Fig. 1.18 shows the SIMULINK block diagram for simulation of the strategies synthesized by UPPAAL-TIGA. The diagram consist of built-in functions and four

[9] 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 $[5.1;8.9]$.



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

Fig. 1.19 The three controller types with SIMULINK

subsystems: Consumer, Accumulator, Cycle and Pump (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 $[-\varepsilon,+\varepsilon]$ with $\varepsilon = 0.1l/s$ 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 by a factor 10, then rounded and fed 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 timestamps 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. 1.19 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 far from optimal.

This is highlighted in Table 1.2, 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 than 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 ELECTRONIC 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 1.2 Performance characteristics based on SIMULINK simulations.

1.6 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 ELECTRONIC 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 imprecision, e.g. with respect to the timing of consumer demands.
- Consideration of other optimization criteria. An interesting feature of the *Smart* controller of HYDAC ELECTRONIC GMBH 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. QUASIMODO. `http://www.quasimodo.aau.dk/`.
8. Simulink, 2008. `http://www.mathworks.com/products/simulink/`.