Available online at www.sciencedirect.com
SCIENCE @ DIRECT®

ELSEVIER

Reliability Engineering and System Safety 81 (2003) 239–257

RELIABILITY ENGINEERING & SYSTEM SAFETY

www.elsevier.com/locate/ress

Fluid Petri Nets and hybrid model-checking: a comparative case study[☆]

M. Gribaudo[a,*], A. Horváth[a], A. Bobbio[b], E. Tronci[c], E. Ciancamerla[d], M. Minichino[d]

[a]*Dip. di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy*
[b]*Dip. di Informatica, Università del Piemonte Orientale, Corso Borsalino 54, 15100 Alessandria, Italy*
[c]*Dip. di Informatica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy*
[d]*ENEA, CR Casaccia, 00060 Roma, Italy*

Abstract

The modeling and analysis of hybrid systems is a recent and challenging research area which is actually dominated by two main lines: a functional analysis based on the description of the system in terms of discrete state (hybrid) automata (whose goal is to ascertain conformity and reachability properties), and a stochastic analysis (whose aim is to provide performance and dependability measures).

This paper investigates a unifying view between formal methods and stochastic methods by proposing an analysis methodology of hybrid systems based on Fluid Petri Nets (FPNs). FPNs can be analyzed directly using appropriate tools. Our paper shows that the same FPN model can be fed to different functional analyzers for model checking. In order to extensively explore the capability of the technique, we have converted the original FPN into languages for discrete as well as hybrid as well as stochastic model checkers. In this way, a first comparison among the modeling power of well known tools can be carried out.

Our approach is illustrated by means of a 'real world' hybrid system: the temperature control system of a co-generative plant.

Keywords: Fluid petri nets; Hybrid model; Hybrid automata

1. Introduction

This paper investigates a unifying view between formal methods and stochastic methods for hybrid systems by proposing an analysis methodology based on Fluid Petri Nets (FPNs) [1,16,14]. The case study, that is used to illustrate the methodology, is a discrete state controller that operates according to the variation of suitable continuous quantities (temperature, heat consumption).

Model parameters are usually affected by uncertainty. A common way to account for parameter uncertainty is to assign to the parameters a range of variation (between a minimum and a maximum value), without any specification on the actual value in a specific realization (nondeterminism). Hybrid automata [3] and model checking tools [8] operate along this line. If a weight can be assigned to the parameter uncertainty through a probability distribution, the nondeterministic model is replaced by a stochastic model: the FPN formalism [16,11] has been proposed to include stochastic specifications. Recently, probabilistic model checkers of Markovian models have also been implemented [4,20].

This paper intends to show that a FPN model of a hybrid system can be used as an input model both for functional analysis as well as for stochastic analysis. In particular, this paper shows that the a FPN model can be translated into a hybrid automaton model [2,24], a discrete model [5], or, finally, a discrete/probabilistic model [20].

FPN's are an extension of Petri nets able to model systems with the coexistence of discrete and continuous state variables [1,16,14]. The main characteristics of FPN is that the primitives (places, transitions and arcs) are partitioned in two groups: discrete primitives that handle discrete tokens (as in standard Petri nets) and continuous (or fluid) primitives that handle continuous quantities (referred to as fluid). Hence, in a single formalism, both discrete and continuous variables can be accommodated and their mutual interaction represented.

Even if Petri nets and model checking rely on very different conceptual and methodological bases (one coming from the world of performance analysis and the other from the world of formal methods), we attempt to gain cross fertilizations from the two areas. The main goal of this paper

[☆] This research has been partially supported by the Italian Ministry of Education under Grants FIRB-RBNE019N8N and MEFISTO.

[*] Corresponding author.

E-mail addresses: marcog@di.unito.it (M. Gribaudo), horvath@di.unito.it (A. Horváth), bobbio@mfn.unipmn.it (A. Bobbio), tronci@dsi.uniroma1.it (E. Tronci), ciancamerl@casaccia.enea.it (E. Ciancamerla), minichino@casaccia.enea.it (M. Minichino).

0951-8320/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0951-8320(03)00089-9

is to investigate the possibility of defining a methodology which allows us to apply a common FPN model both for formal specification and verification via model checking and for performance analysis.

We describe our approach and show its usefulness by using a meaningful 'real world' application. We take the control system for the temperature of the primary and secondary circuits of the heat exchange section of the ICARO co-generative plant [4] in operation at ENEA CR Casaccia as our example. The plant is composed of two sections: the gas turbine that produces electrical power and the heat exchange section that extracts heat from the turbine gases.

The comparative analysis documented in this paper using the case study, allows us to draw some preliminary conclusions about the modeling power of the tools and techniques we utilize and their analytical tractability.

The paper is organized as follow. Section 2 describes the case study. Section 3 introduces the main elements of the FPN formalism, provides a FPN model for the case study and show some results obtained with the simulator described in Ref. [10]. Section 4 introduces the concept of hybrid automata, provides the conversion of the FPN model into a hybrid automaton and analyzes the resulting hybrid automaton by means of the tool HyTech [18]. Section 5 shows how the same FPN model can be translated into a discrete model suitable for the model checker NuSMV [23] and provides some experimental results. Section 6 translates the FPN model into a DTMC model suitable for the probabilistic model checker PRISM [22]. Section 7 concludes the paper with some preliminary considerations about the modeling and analysis features of the different tools.

2. Temperature control system

The ICARO co-generative plant comprises two sections: the electrical power generation and the heat extraction from the turbine exhaust gases. The exhaust gases can be conveyed to a re-heating chamber to heat the water of a primary circuit and then, through a heat exchanger, to heat the water of a secondary circuit that, actually, is the heating circuit of the ENEA Research Center. If the thermal energy required by the end user is higher than the thermal energy of the exhaust gases, fresh methane gas can be fired in the re-heating chamber where the combustion occurs. The block diagram of the temperature control of the primary and secondary circuits is depicted in Fig. 1. The control of the thermal energy used to heat the primary circuit is performed by regulating both the flow rate of the exhaust gases through the diverter D and the flow rate of the fresh methane gas through the valve V. T_1 is the temperature of the primary circuit, T_2 is the temperature of the secondary circuit, and u is the thermal request by the end user.

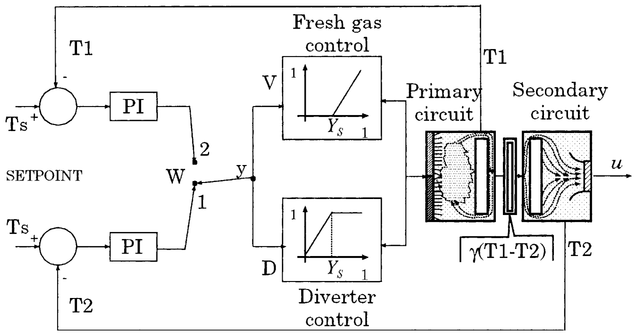


Fig. 1. Temperature control of the primary and secondary circuits of the ICARO plant.

The controller has two distinct regimes (two discrete states) represented by the position 1 or 2 of the switch W in Fig. 1. Position 1 is the normal operational condition, position 2 is the safety condition. In position 1, the control is based on a proportional-integrative measure (performed by block PI$_1$) of the error of temperature T_2 with respect to a (constant) set point temperature T_s. Conversely, in position 2, the control is based on a proportional-integrative measure (performed by block PI$_2$) of the error of temperature T_1 with respect to a (constant) set point temperature T_s. Normally, the switch W is in position 1 and the control is performed on T_2 to maintain constant the temperature to the end user. Switching from position 1 to position 2 occurs for safety reasons, when the value of T_2 is higher than a critical value defined as the set point T_s augmented by a hysteresis value T_h and the control is locked to the temperature of the primary circuit T_1, until T_1 becomes lower than the set point T_s.

The exit of the proportional-integrative block (either PI$_1$ or PI$_2$, depending on the position of the switch W) is the variable y which represents the request of thermal energy. When y is lower than a split point value Ys the control just acts on the diverter D (flow of the exhaust gases), when the diverter is completely open, and the request for thermal energy y is greater than Ys, the control also acts on the flow rate of the fresh methane gas by opening the valve V.

The heating request is computed by the function $f(y)$ represented in Fig. 2. Since the temperature T_2 is used when W is in position 1, and the temperature T_1 is used in state 2, the function $f(y)$ depends on y_2 when $W = 1$ and on y_1 when $W = 2$. The function $f(y)$ is defined as the sum of two nondeterministic components, namely: $g_1(y)$ which

$$\frac{g_1(y)}{\alpha_{max}} + \frac{g_2(y)}{\beta_{max}} \le f(y) \le \frac{g_1(y)}{\alpha_{min}} + \frac{g_2(y)}{\beta_{min}}$$

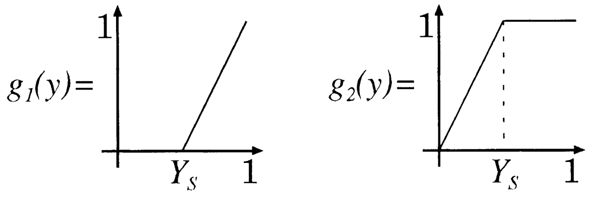


Fig. 2. The heating request function $f(x)$.

represents the state of the valve V, and $g_2(y)$ which represents the state of the diverter D. The nondeterminism is introduced by the parameters $\alpha_{\min}, \alpha_{\max}$ that give the minimal and maximal heat induced by the fresh methane gas, and $\beta_{\min}, \beta_{\max}$ that define the minimal and maximal heat induced by the exhaust gases.

Finally, the heat exchange between the primary and the secondary circuit is approximated by the linear function $\gamma(T_1 - T_2)$, proportional (through a constant γ) to the temperature difference.

3. Fluid petri nets

FPN are an extension of standard Petri Nets [21], where, beyond the places that contain a discrete number of tokens, a new kind of place is added that contains a continuous quantity (fluid). Hence, this extension is suitable for modeling and analyzing hybrid systems. Two main formalisms have been developed in the area of FPN: the Continuous or Hybrid Petri net (HPN) formalism [1], and the Fluid Stochastic Petri net (FSPN) formalism [16,14]. A complete presentation of FPN is beyond the scope of the present paper. An extensive discussion of FPN in performance analysis can be found in Ref. [11].

Discrete places are drawn according to the standard notation and contain a discrete amount of tokens that are moved along discrete arcs. Fluid places are drawn by two concentric circles and contain a real variable (the fluid level). The fluid flows along fluid arcs (drawn by a double line to suggest a pipe) according to an instantaneous flow rate. The discrete part of the FPN regulates the flow of the fluid through the continuous part, and the enabling conditions of a transition depend only on the discrete part.

3.1. S-FPN analysis techniques

Stochastic FPN (S-FPN) are FPN where transitions have associated a random firing time [10]. The dynamic evolution of a S-FPN in time is described by a set of partial differential equations, with a partial differential equation for each fluid place considered. Two kinds of solution approaches are available to solve the equations that characterize the stochastic process of a S-FPN: numerical techniques (see [16,14,11]) and simulation (see [7,13]). Numerical techniques achieve very detailed results at the expense of very high computational cost. The equations become almost impossible to solve for models with more than two or three fluid places. Simulation techniques compute average measures instead. They do not suffer from the limitations imposed by the numerical techniques, but accuracy is often very hard to control.

3.2. A FPN description of the system

The case study of Fig. 1 is represented as an FPN model in Fig. 3. The FPN contains two discrete places: $P1$ which is marked when the switch W is in state 1, and $P2$ which is marked when the switch W is in state 2. Fluid place *Primary* (whose fluid level is denoted by T_1) represents the temperature of the primary circuit, and fluid place *Secondary* (whose fluid level is denoted by T_2) represents the temperature of the secondary.

The fluid arcs labeled with $\gamma(T_1 - T_2)$ represent the heat exchange between the primary and the secondary circuit. The system jumps from state 1 to state 2 due to the firing of immediate transition $Sw12$. This transition has associated a guard $T_2 > T_s + T_h$ that makes the transition fire (inducing a change of state) as soon as the temperature T_2 exceeds the *setpoint* T_s augmented by an hysteresis value T_h. The jump from state 2 to state 1 is modeled by the immediate transition $Sw21$, whose firing is controlled by the guard $T_1 < T_s$ that makes the transition fire when the temperature T_1 goes below the *setpoint* T_s. In order to simplify the figure, we have connected the fluid arcs directly to the immediate transitions. The meaning of this unusual feature is that fluid flows across the arcs as long as the immediate transitions are enabled regardless of the value of the guards.

The outward fluid arc from place *Secondary*, represents the end user demand whose range of variation is denoted by $[u_1, u_2]$. Fluid place $CTR2$, whose marking is denoted by y_1, models the exit of the proportional-integrator PI_1. This is achieved by connecting to place $CTR2$ an input fluid arc with a constant fluid rate equal to the *setpoint* T_s and an

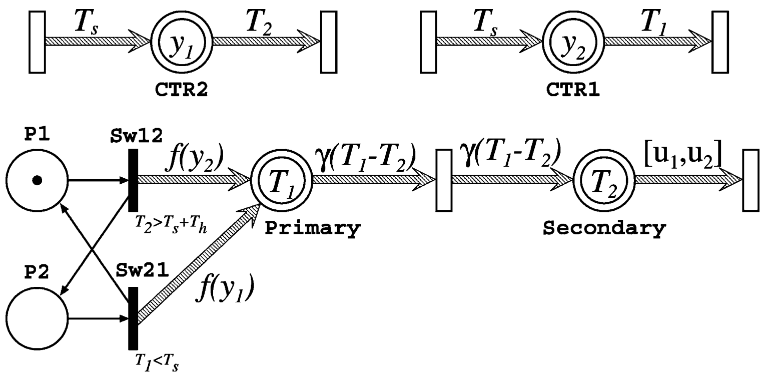


Fig. 3. FPN model of the temperature controller.

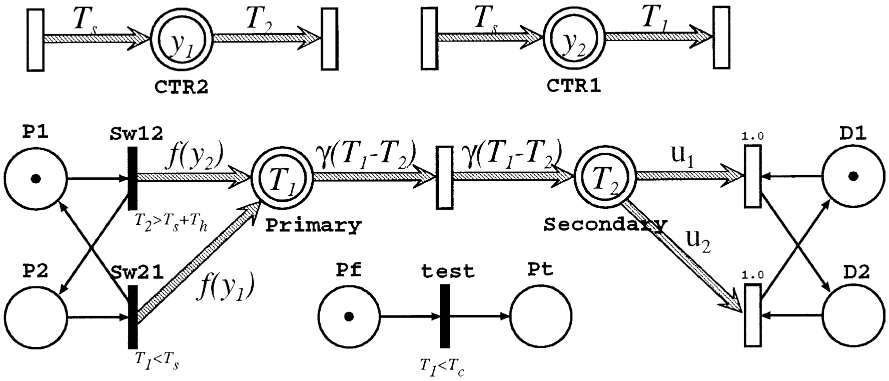


Fig. 4. S-FPN model of the temperature controller.

output fluid arc characterized by a variable flow rate equal to T_2. In a similar way, the exit of the proportional-integrator PI_2 is modeled by fluid place *CTR*1 (whose marking is denoted by y_2). The fluid arcs that connect transition *Sw*12 and *Sw*21 to fluid place *primary* represent the heating up of the primary circuit.

3.3. Results from S-FPN simulator

In nonstochastic settings, the firing times for the transitions in the net can be given either by a constant value or by an interval with nondeterministic choice. Instead, in the stochastic setting, the firing times are random variables and are defined by assigning a firing intensity to the timed transitions. The firing intensity may be constant (firing times exponentially distributed) or may depend on the marking.

Nondeterministic flow rates require a more sophisticated approach. We represent a stochastically varying flow rate with a continuous time Markov chain that modulates the flow rate. In the considered example, in order to model a stochastically varying user demand (between a minimum value u_1 and a maximum value u_2) we use the simple two state Markov chain of Fig. 4. Places $D1$ and $D2$ model the two states of the Markov chain. When place $D1$ is marked, the user demand is u_1, when place $D2$ is marked, the user demand is u_2. The mean sojourn time of the Markov chain in states $D1$ and $D2$ is equal to one time unit. Tuning the transition probabilities between states $D1$ and $D2$ or adding more states, allows us to accommodate any stochastic behavior of the user demand.

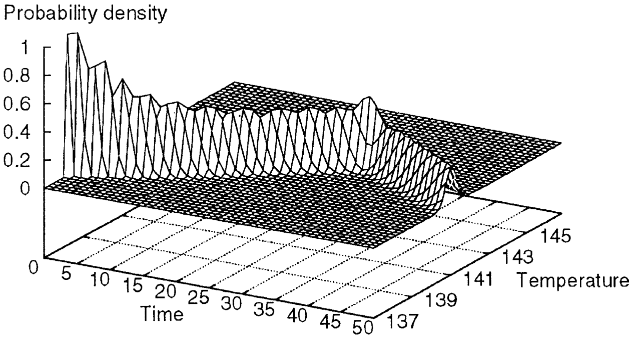


Fig. 5. Distribution of temperature T_1 as function of time.

The system dynamics has been evaluated using a S-FPN simulator [13]. Figs. 5 and 6 show the distribution of the temperatures T_1 and T_2 of the primary and secondary circuit, respectively, as the function of time. The distributions at some particular time instants are presented in Fig. 7. In particular time $t = 5$ shows the distributions during the heating up stage, time $t = 33$ shows the distributions near the maximum mean temperature, and $t = s.s.$ represents the distributions in steady state. The other parameter values are set according to the same values that will be presented in Section 5.

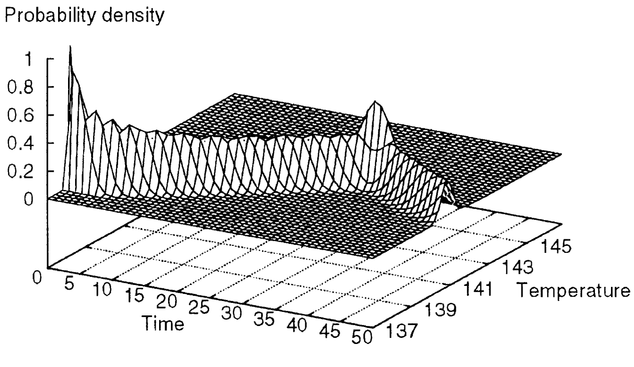


Fig. 6. Distribution of temperature T_2 as function of time.

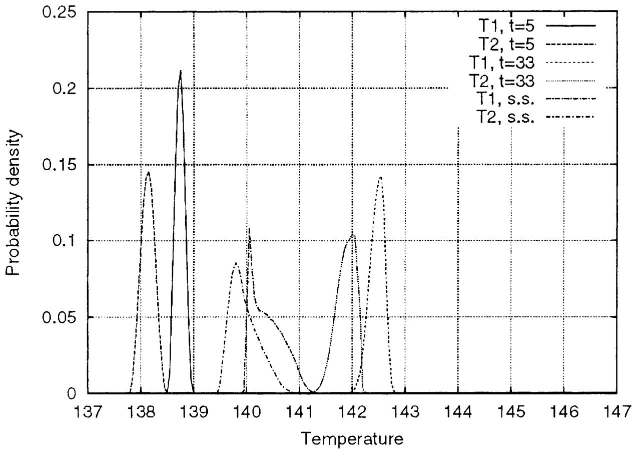


Fig. 7. Temperature distribution at some time instants and in steady state.

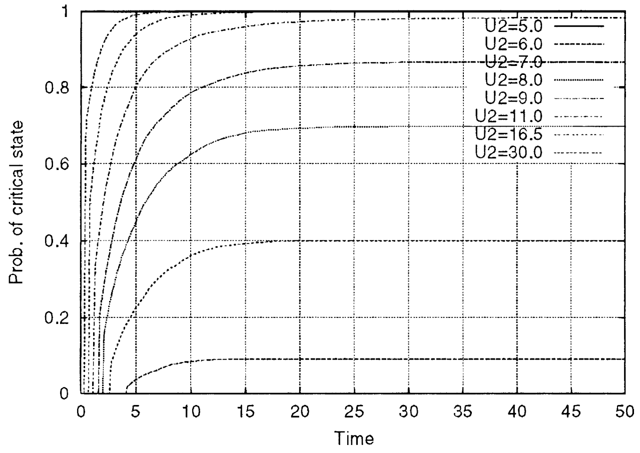


Fig. 8. Distribution of time to critical state for various user maximum demand.

Other interesting measures can be computed using S-FPN by adding some elements to the model. For example, the probability of reaching a bound which may bring the system to an unsafe condition can be evaluated by adding two places and a transition with a guard. When the guard level is reached a token is transferred from the input to the output place.

With reference to the present case study, we can define a critical temperature threshold value T_c (for example $T_c = 137$ °C). The attainment of the critical condition is modeled in Fig. 4 by adding two places P_f and P_t and one immediate transition *test* controlled by a guard ($T_1 < T_c$ or $T_2 < T_c$). As soon as the critical condition becomes true, the transition *test* is enabled and moves one token from place P_f to place P_t. The probability of the critical condition versus time can be computed, using S-FPN transient analysis, as the probability of having place P_t marked. Fig. 8 shows various distributions of the probability of reaching the critical state at a given time instant, by varying the maximum user demand u_2. As we can see, the system never reaches the critical state if the maximum user demand is $u_2 \leq 5$. For $5.5 \leq u_2 \leq 13$ there is a nonnull probability of reaching the critical state and for $u_2 \geq 16$ the probability of reaching the critical state tends to 1 as time increases. The higher the maximum user demand, the sooner the critical state is reached.

Fig. 9 represents the steady state probability of reaching the critical state, as a function of the maximum user demand.

4. Hybrid automata

A hybrid automaton [3] is a finite state machine whose nodes (called control modes or *locations*) contain real valued variables with a definition of their first derivatives and possible bounds on their values. The edges represent discrete events and are labeled with guarded assignments on the real variables.

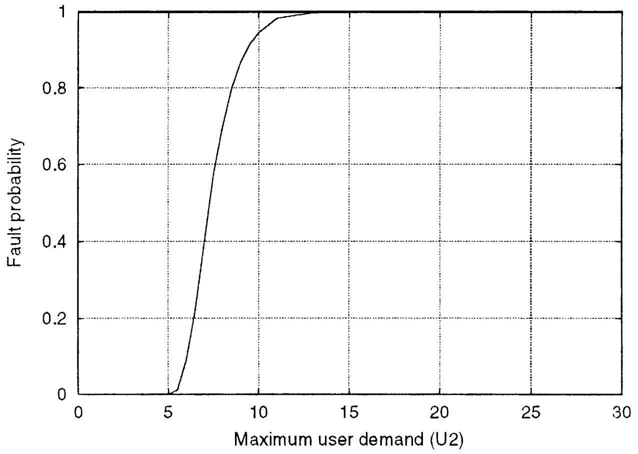


Fig. 9. Steady state fault probability as function of the maximum user demand.

Given a hybrid automaton and a legal formula on its variables (essentially a legal formula is a Boolean combination of linear constraints), the *model checking problem* asks to compute a region that satisfies the formula, or to find at least one counterexamples that contradicts the formula.

4.1. From FPN to hybrid automata

In order to use a FPN model in a model checking environment, the FPN formalism should be converted into a hybrid automaton. A general conversion algorithm could be envisaged following Refs. [2,24], and is based on the conversion rules summarized in Table 1.

The application of the conversion algorithm to the case study FPN of Fig. 3 provides the hybrid automaton [3] of Fig. 10.

According to Table 1, the hybrid automaton has two locations $P1$ and $P2$ (corresponding to the two discrete markings of the FPN) and the following set of real variables T_1, T_2, y_1 and y_2 (corresponding to the fluid variables of the FPN). Each continuous variable has a derivative equal to the flow rate of the corresponding fluid place in that state. Transitions from location $P1$ to $P2$ and from $P2$ to $P1$ are labeled with the guards of the immediate transitions that cause the state change. State $P1$ has also associated the bound (invariant condition) $T_2 \leq T_s + T_h$ and $P2$ the bound $T_1 \geq T_s$ to reflect the same bounds posed on those fluid places.

Table 1
Conversion rules from FPN to HA

Fluid Petri Net		Hybrid automaton
Discrete states	⇒	Locations
Fluid variables	⇒	Continuous variables
Fluid flow rates	⇒	Derivatives versus time
Guards	⇒	Bounds

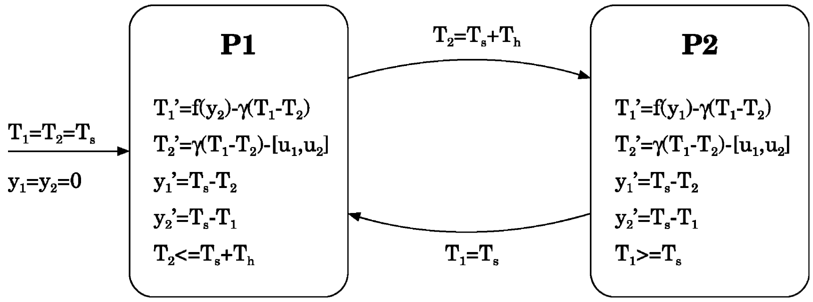


Fig. 10. Hybrid Automata obtained from the FPN of Fig. 3.

4.2. From hybrid automata to HyTech

HyTech [18,3,15] is a model checker for Linear Hybrid Automata (LHA), i.e. for hybrid automata which only use a restricted form of linear differential equations to define the dynamics of the continuous state variables. It is, thus, quite natural to attempt to analyze the hybrid automaton of Fig. 10 using HyTech. Unfortunately, the automaton in Fig. 10

```
define(Ts, 2)              -- set point (offset w.r.t. 138)
define(Th, 2)              -- hysteresis 2C
define(Tsh, 4)             -- (Ts + Th)
-- T = 0 stands for T = 138C
var
T1, T2, y1, y2: analog;

automaton FG103
synclabs : ;


initially p_s1_1_1_1;

loc p_s1_0_0_1 :
while T2 <= Tsh & -1 <= T1 & T1 <= 1 & -1 <= T2 & T2 <= 1 &
      6 <= y1 & y1 <= 18
wait {dT1 = 1, dT2 in [-4, -3], dy1 = 40,  dy2 = 40 }
when T2 <= Tsh goto p_s1_0_0_2 ;
when T2 <= Tsh goto p_s1_0_1_1 ;
when T2 <= Tsh goto p_s1_0_1_2 ;
when T2 <= Tsh goto p_s1_0_2_1 ;
...
when T2 >= Tsh goto p_s2_5_4_2 ;
when T2 >= Tsh goto p_s2_5_5_1 ;
when T2 >= Tsh goto p_s2_5_5_2 ;

loc p_s1_0_0_2 :
while T2 <= Tsh & -1 <= T1 & T1 <= 1 & -1 <= T2 & T2 <= 1 &
      18 <= y1 & y1 <= 30
wait {dT1 = 6, dT2 in [-4, -3], dy1 = 40,  dy2 = 40}
when T2 <= Tsh goto p_s1_0_0_1 ;
when T2 <= Tsh goto p_s1_0_1_1 ;
when T2 <= Tsh goto p_s1_0_1_2 ;
when T2 <= Tsh goto p_s1_0_2_1 ;
when T2 <= Tsh goto p_s1_0_2_2 ;
...
when T2 >= Tsh goto p_s2_5_4_1 ;
when T2 >= Tsh goto p_s2_5_4_2 ;
when T2 >= Tsh goto p_s2_5_5_1 ;
when T2 >= Tsh goto p_s2_5_5_2 ;
```

Fig. 11. A glimpse of the beginning of our HyTech model.

```
var init_reg,
    final_reg,
    reachable,
    reached_viol,
    reached    :  region;

-- initially: (T1 = 140) & (T2 <= 140)
   init_reg:=  (T1 = 2) & (T2 <= 2) &
               (y1 = 12) & (y2 = 12) &
               loc[FG103] = p_s1_1_1_1;

-- spec violation:
-- (T1 <= 138) | (T1 >= 143) | (T2 <= 138) | (T2 >= 143)
   final_reg:= (T1 <= 0) | (T1 >= 5) | (T2 <= 0) | (T2 >= 5);

-- forward analysis
 reachable:= reach forward from init_reg endreach;
 reached_viol:= reachable & final_reg;
 if empty ( reached_viol )
              then prints "System verified";
             else prints "System not verified";
                  print trace to final_reg using reachable;
     endif;
```

Fig. 12. A glimpse of the final part (analysis commands) for our HyTech model.

cannot be directly translated into a HyTech model. In fact, the specification of the derivatives of the continuous variables in HyTech is restricted to be in the form of a constant nondeterministic range (a constant lower and upper bound). In the automaton of Fig. 10 the time derivatives for T_1 and T_2 are not constant, but are expressed as functions of the continuous variables themselves. Thus, some manipulation is required.

The above problem is common when modeling with HyTech. The standard solution is to split *each* continuous

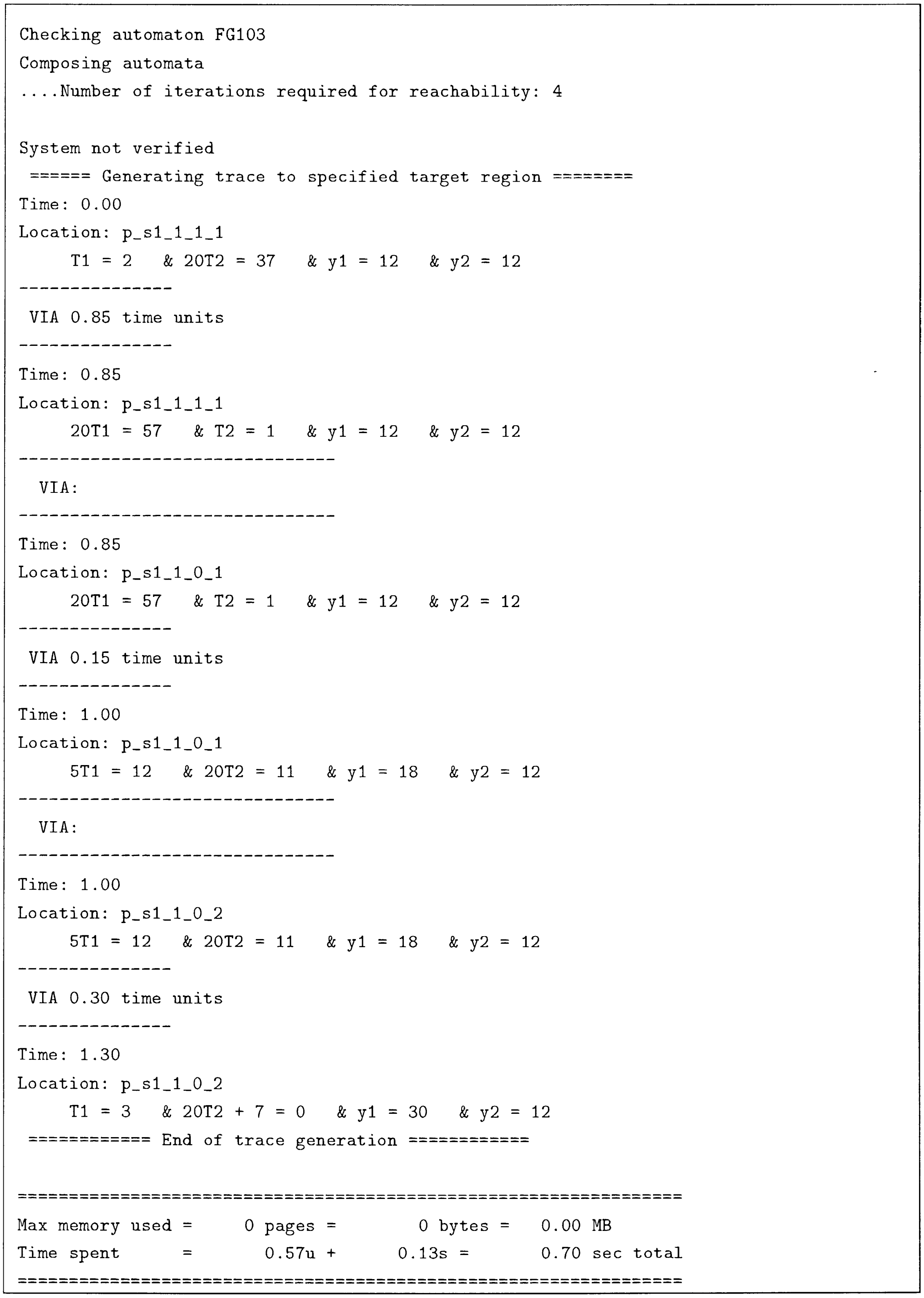
```
=================================================================
HyTech: symbolic model checker for embedded systems
Version 1.04 10/15/96
For more info:
    email: hytech@eecs.berkeley.edu
    http://www.eecs.berkeley.edu/~tah/HyTech
Warning: Input has changed from version 1.00(a). Use -i for more info
=================================================================


Checking automaton FG103
Composing automata
....Number of iterations required for reachability: 4

System verified

=================================================================
Max memory used =     0 pages =        0 bytes =    0.00 MB
Time spent      =       0.46u +      0.11s =       0.57 sec total
=================================================================
```

Fig. 13. HyTech answer when the user demand is between $u_1 = 2$ and $u_2 = 3$.



```
Checking automaton FG103
Composing automata
....Number of iterations required for reachability: 4

System not verified
 ====== Generating trace to specified target region ========
Time: 0.00
Location: p_s1_1_1_1
     T1 = 2   & 20T2 = 37   & y1 = 12   & y2 = 12
---------------
 VIA 0.85 time units
---------------
Time: 0.85
Location: p_s1_1_1_1
     20T1 = 57   & T2 = 1   & y1 = 12   & y2 = 12
-------------------------------
  VIA:
-------------------------------
Time: 0.85
Location: p_s1_1_0_1
     20T1 = 57   & T2 = 1   & y1 = 12   & y2 = 12
---------------
 VIA 0.15 time units
---------------
Time: 1.00
Location: p_s1_1_0_1
     5T1 = 12   & 20T2 = 11   & y1 = 18   & y2 = 12
-------------------------------
  VIA:
-------------------------------
Time: 1.00
Location: p_s1_1_0_2
     5T1 = 12   & 20T2 = 11   & y1 = 18   & y2 = 12
---------------
 VIA 0.30 time units
---------------
Time: 1.30
Location: p_s1_1_0_2
     T1 = 3   & 20T2 + 7 = 0   & y1 = 30   & y2 = 12
 ============ End of trace generation ============

=================================================================
Max memory used =     0 pages =        0 bytes =   0.00 MB
Time spent      =       0.57u +      0.13s =       0.70 sec total
=================================================================
```

Fig. 14. HyTech answer when the load is between $u_1 = 2$ and $u_2 = 4$.

variable into intervals in which the time derivatives can be bound using constants. Of course, this approach leads to many locations, since the smaller the intervals the better the approximation: increasing the number of intervals entails an increase in the number of locations. Thus a reasonable compromise must be found.

Because of its size, writing a HyTech model by hand using the above technique is out of question for our case study.

Thus, we wrote a shell script that generates our HyTech model using the approach outlined above together with a *rescaling* of the continuous values in order to have small numbers thereby avoiding HyTech arithmetical overflows.

The resulting HyTech program has about 150 locations. In Fig. 11 we show the initial part of the HyTech specifications for our model and in Fig. 12 the final part containing the analysis commands (i.e. the properties that we want to check by means of HyTech). Our specification requires that temperatures T_1 and T_2 be always within assigned bounds for an assigned range of user demand $[u_1, u_2]$. In Fig. 12 we ask HyTech to check if it is possible to reach a state that violates this specification.

Depending on the range of variation of the end user demand $[u_1, u_2]$, the required specification may or may not be satisfied. Fig. 13 shows the HyTech answer when the specification is satisfied (user demand $[u_1, u_2]$ in the range [2,3]). Fig. 14 shows the error trace returned by HyTech when our specification is not satisfied (user demand $[u_1, u_2]$ in the range [2,4]).

The above discussion shows that generating the HyTech model for a nontrivial hybrid automaton requires a considerable effort because of the restricted class of hybrid automata accepted by HyTech. Moreover the HyTech model can be automatically analyzed by HyTech only if it is of moderate size. This suggests that for hybrid systems similar to the one we studied HyTech can only be used to analyze small (abstracted away) models of the system at hand.

5. Analysis of the FPN model via NuSMV

Discrete model checking is based on a finite state machine model in which the variables and their derivatives are discretized and in which the time increases with a predefined time step. The parameters and their derivatives can be assigned uncertainty ranges (e.g. a min and a max value) with nondeterministic logic. The predicates to be checked are specified using a Computational Tree Logic (CTL) or a Real Time CTL (RTCTL) [8,9].

In order to show the generality of our approach and to give an insight on the class of models that can be automatically derived from the FPN description, we sketch, in brief, how the FPN can be converted into a discrete model to be checked using discrete model checking techniques and we present some typical analyses and results that can be obtained from the converted model. For the purpose of the present paper, we have chosen the language NuSMV [23].

5.1. Converting a FPN into a NuSMV model

The algorithm to convert a FPN model into the NuSMV language requires the following steps:

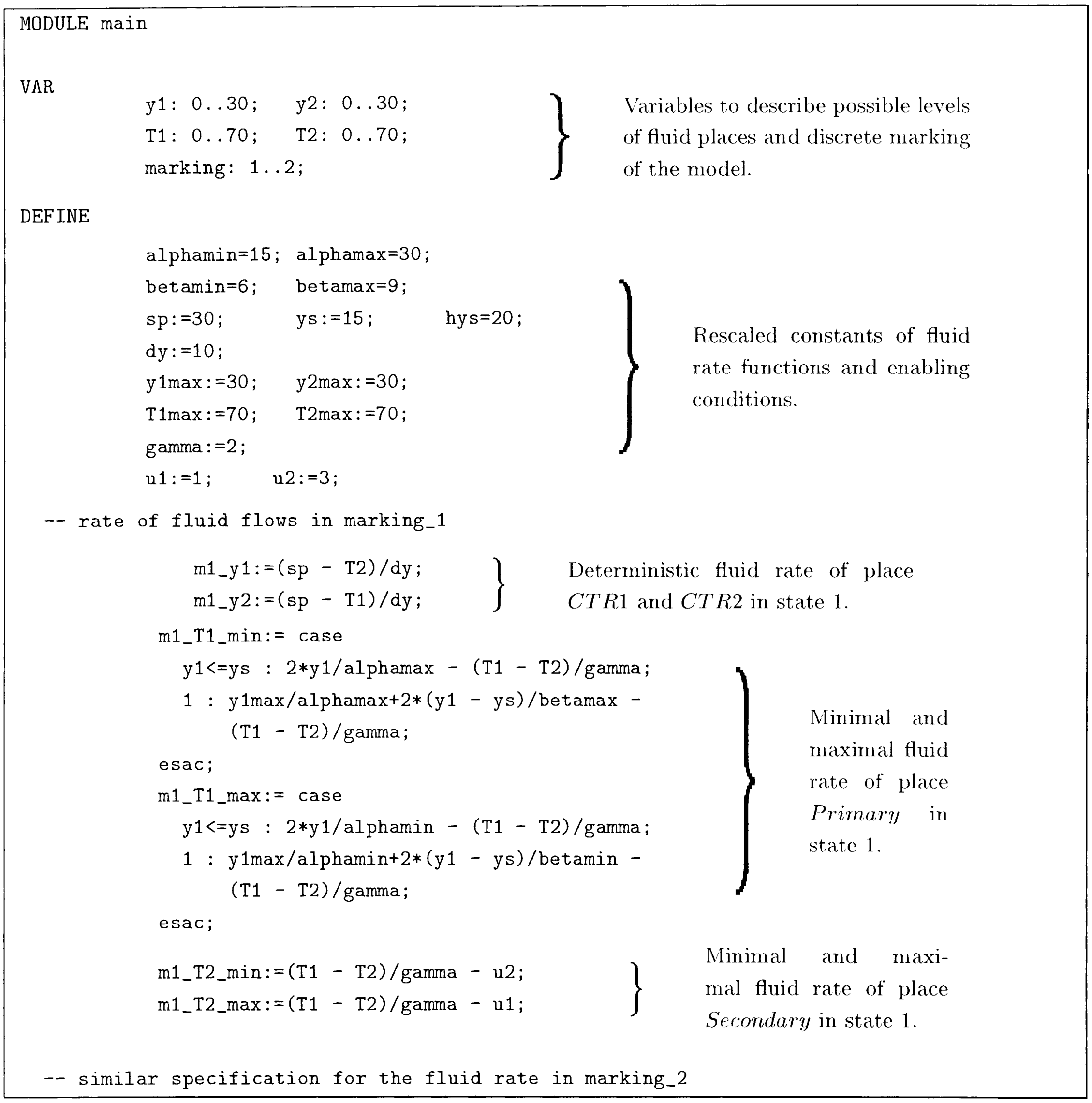
1. Definition of the variables. The discrete markings of the FPN model are directly translated into a discrete variable in NuSMV. All the continuous variables (fluid levels of the FPN) and their rates of variation must be, instead, suitably discretized (choosing a suitable discretization interval). Variables are introduced in the NuSMV specification using the keywords `VAR`.
2. The second step defines the FPN constants that are used in the fluid rate functions or in the enabling conditions. Moreover, the ranges of variation (min and max values) for the continuous variables and for their rates must be set. All these quantities (constants and bounds) must be suitably discretized and rescaled according to the discretization intervals chosen in step 1), above. The constants and the bounds are listed under the keyword `DEFINE`.
3. In order to analyze the behavior of the control system versus time, a time step (in arbitrary units) is assumed and the dynamic evolution of the system at the integer multiples of the time step must be described. The evolution of the model is stated under the keyword `TRANS` and must be described marking by marking.
4. Finally, the fourth step consists in defining an initial state from which the dynamic evolution of the model starts. The initial state is described under the keyword `INIT`.

We now particularize the above general steps to the present case study. The discrete part of the FPN model is reflected in the variable *marking*, whose value is either 1 or 2. Furthermore, all the continuous variables (fluid levels of the FPN) and their range of variation must be discretized.

Four fluid variables are defined in the FPN of Fig. 3, namely: y_1, y_2 and T_1, T_2. In the NuSMV description, the variables representing the fluid levels y_1, y_2, T_1 and T_2 are denoted by `y1`, `y2`, `T1`, and `T2`. The fluid levels y_1 and y_2, of fluid places *CTR*1 and *CTR*2, respectively, are discretized with a step interval 1/30, so that `y1` and `y2` range in the interval `0:30` (Fig. 15). The normalization constant for y_1 and y_2 is denoted by dy and represents how fast the system reacts to the temperature difference with respect to the setpoint. The fluid levels T_1 and T_2 of fluid places Primary and Secondary, respectively, are bounded between $T_\ell = 138$ and $T_u = 145$ and the discretization step chosen for these variables is 0.1. With these assumptions, the temperature variables in NuSMV are scaled with respect to the lower bound T_ℓ so that `T1` and `T2` range in the interval `0:70`. A value $\mathtt{T1}(2) = i, 0 \le i \le 70$ implies that $\mathtt{T1}(2) = T_\ell + i * 0.1$.

The second part under the keyword `DEFINE` gives the possible fluid changes in the different states of the model. Both minimal and maximal fluid changes have to be calculated, this is done by summing ingoing and outgoing fluid rates and considering minimal and maximal values of the appearing variables. For `marking` = `1`, these are the following (similar definitions hold for `marking` = `2`):

- $m1\_y1$ gives the (deterministic) fluid rate of place *CTR*1 in state 1;



```
MODULE main

VAR
          y1: 0..30;    y2: 0..30;
          T1: 0..70;    T2: 0..70;
          marking: 1..2;
```

Variables to describe possible levels of fluid places and discrete marking of the model.

```
DEFINE
          alphamin=15; alphamax=30;
          betamin=6;   betamax=9;
          sp:=30;      ys:=15;      hys=20;
          dy:=10;
          y1max:=30;   y2max:=30;
          T1max:=70;   T2max:=70;
          gamma:=2;
          u1:=1;     u2:=3;
```

Rescaled constants of fluid rate functions and enabling conditions.

```
  -- rate of fluid flows in marking_1

              m1_y1:=(sp - T2)/dy;
              m1_y2:=(sp - T1)/dy;
```

Deterministic fluid rate of place *CTR*1 and *CTR*2 in state 1.

```
           m1_T1_min:= case
             y1<=ys : 2*y1/alphamax - (T1 - T2)/gamma;
             1 : y1max/alphamax+2*(y1 - ys)/betamax -
                 (T1 - T2)/gamma;
           esac;
           m1_T1_max:= case
             y1<=ys : 2*y1/alphamin - (T1 - T2)/gamma;
             1 : y1max/alphamin+2*(y1 - ys)/betamin -
                 (T1 - T2)/gamma;
           esac;
```

Minimal and maximal fluid rate of place *Primary* in state 1.

```
           m1_T2_min:=(T1 - T2)/gamma - u2;
           m1_T2_max:=(T1 - T2)/gamma - u1;
```

Minimal and maximal fluid rate of place *Secondary* in state 1.

```
  -- similar specification for the fluid rate in marking_2
```

Fig. 15. A glimpse of our NuSMV model.

- $m1\_y2$ gives the (deterministic) fluid rate of place *CTR*2 in state 1;
- *m*1_*T*1_*min* and *m*1_*T*1_*max* give the minimal and maximal flow rate of fluid place *Primary* in state 1;
- *m*1_*T*2_*min* and *m*1_*T*2_*max* give the minimal and maximal flow rate of fluid place *Secondary* in state 1;

In order to analyze the behavior of the control system versus time, we assume a time step (in arbitrary unit) and describe the dynamic evolution of the system at the integer multiples of the time step. The evolution of the model is stated under the keyword TRANS and must be described marking by marking. Since in the present model we have two markings (states), the evolution description is restricted to four expressions:

- possible changes of the variables inside state marking = 1;
- possible changes of the variables inside state marking = 2;
- jump from marking = 1 to marking = 2;
- jump from marking = 2 to marking = 1.

Finally, the initial state of the model is described under the Keywords INIT. An excerpt of the NuSMV specifications for the case study at hand is provided in Figs. 15 and 16. The complete specification is in Ref. [17].

5.2. NuSMV results

NuSMV is a model checking tool that also contains a simulation engine that allows us to explore the dynamics of the system. To increase the readability of our results, we report the variables in their true units (and not in the rescaled units used by NuSMV). Figs. 17 and 18 depict the evolution of the temperatures (T1 and T2) and of y1 and y2,

```
INIT
           marking=1 & y1=0 & y2=0 & T1=20 & T2=20     }     Initial situation
TRANS
            (
            marking=1 & T2<sp+hys &
            next(marking)=1 &
            ( (y1+m1_y1<0 & next(y1)=0) |
              (y1+m1_y1>y1max & next(y1)=y1max) |
              (y1+m1_y1>=0 & y1+m1_y1<=y1max &
               next(y1)=y1+m1_y1) ) &
            ( (y2+m1_y2<0 & next(y2)=0) |                  Possible
              (y2+m1_y2>y2max & next(y2)=y2max) |          change of the
              (y2+m1_y2>=0 & y2+m1_y2<=y2max &             variables in
               next(y2)=y2+m1_y2) ) &                      one step in-
            ( (T1+m1_T1_max<0 & next(T1)=0) |              side marking
              (T1+m1_T1_min>T1max & next(T1)=T1max) |      1 described
              (T1+m1_T1_max>=0 & T1+m1_T1_min<=T1max &     using the
               next(T1)>=T1+m1_T1_min &                    rates defined
               next(T1)<=T1+m1_T1_max) ) &                 above.
            ( (T2+m1_T2_max<0 & next(T2)=0) |
              (T2+m1_T2_min>T2max & next(T2)=T2max) |
              (T2+m1_T2_max>=0 & T2+m1_T2_min<=T2max &
               next(T2)>=T2+m1_T2_min &
               next(T2)<=T2+m1_T2_max) )
            ) |
    -- Possible change of the variables in one step inside
    -- marking 2 described using the rates defined above.
    -- Possible jump in one step from state 1 to state 2
    -- Possible jump in one step from state 2 to state 1
```

Fig. 16. A glimpse of our NuSMV model (continued from Fig. 15).

respectively, for the same simulation trace, starting from the initial state [y1 = 0, y2 = 0, T1 = 138, T2 = 138].

The design specification for the temperatures of the system (given as invariant condition or bounds) are: ($139 \le T1 \le 144$ and $139 \le T2 \le 141$). If the invariant does not hold, i.e. the temperatures exceed the bounds, NuSMV produces a counterexample as in Table 2. Table 2 shows a case with [gamma = 2, dy = 10, Ts = 140]. Both temperatures start initially from 141 and decrease because of the heat consumption of the user. As T2 (T1) reaches Ts, y1 (y2) starts to increase. However, the reaction is not fast enough to avoid the undesirable condition and the secondary temperature T2 crosses the lower bound. Modifying the design parameters may avoid to incur in this situation

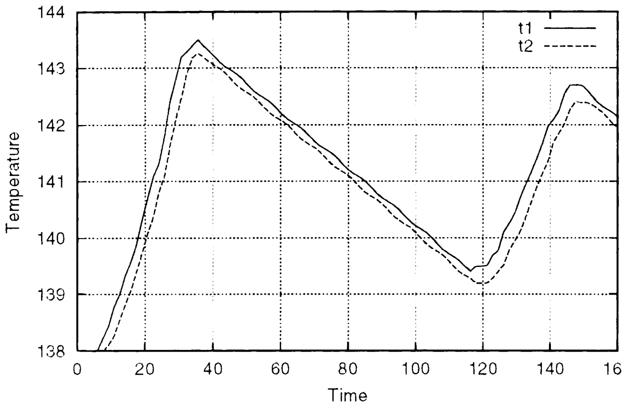


Fig. 17. Change of temperature given by a simulation trace.

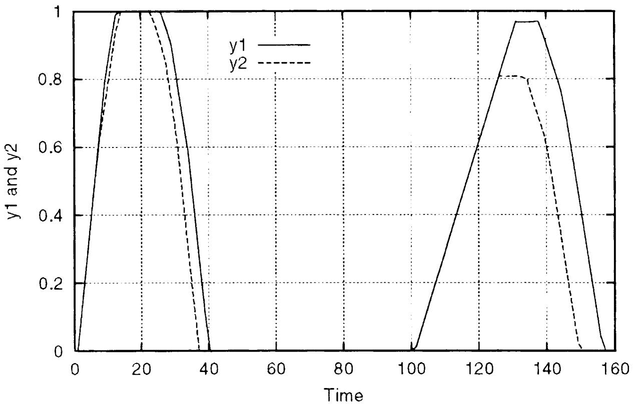


Fig. 18. Change of y_1 and y_2 given by a simulation trace.

Table 2
Counter example

Step	1	2	…	6	7	8	9	10	11	12	13	14
State	1	1		1	1	1	1	1	1	1	1	1
$T1$	141	141		140.4	140.3	140.1	140	139.8	139.7	139.5	139.4	139.2
$T2$	141	140.7		140.1	139.9	139.8	139.6	139.5	139.3	139.2	139	138.9
$y1$	0	0		0	0	1/30	2/30	3/30	4/30	5/30	6/30	8/30
$y2$	0	0		0	0	0	0	1/30	2/30	3/30	4/30	5/30

(f.i. setting [gamma = 2, dy = 1/10, Ts = 139.8], i.e. speeding up the reaction of the system, and reducing the setpoint temperature).

Using RTCTL [9] expression, one can check the trajectory on which the system proceeds. For example, starting from the lowest possible temperatures (T1 = T2 = 138) the formula

```
AF (AG (T1 > = 139 & T1 < = 144 &
T2 > = 139 & T2 < = 141)
```

is true if the system gets back to stable state for sure and remains there forever. Setting [gamma = 2, dy = 1/10, sp = 18], the formula evaluates to true. The same formula, with the same settings evaluates to true as well, if the system starts from the upper bound of the temperatures.

Knowing the timing behavior of the system, one can use NuSMV to compute the minimal or maximal time needed to get a given set of states from an initial condition. For example, the commands

```
COMPUTE MIN [y1 = 0 & y2 = 0 & T1 = 145 &
T2 = 145, AG (T1 > = 139 & T1 < = 144
T2 > = 139 & T2 < = 141)]
COMPUTE MAX [y1 = 0 & y2 = 0 & T1 = 145 &
T2 = 145, AG (T1 > = 139 & T1 < = 144
T2 > = 139 & T2 < = 141)]
```

give the length of the minimal and maximal paths that lead from the initial condition [y1 = 0, y2 = 0, T1 = 145, T2 = 145] of high temperatures (out of the required range) to temperatures inside the required range in such a way that the system does not leave this range in the future. The above command with parameters [gamma = 2, dy = 1/10, sp = 18] results in $min - path = 21$ and $Max - path = 64$.

6. Probabilistic model checking

In this section, we move from a nondeterministic setting to a probabilistic one and show some typical results that can be obtained from a probabilistic model checker. In particular, rather than using a nondeterministic model for the user demand u (as we did in Sections 4.2 and 5) here we use a probabilistic model (as we did in Section 3.3. This allows us to exploit the statistical knowledge about the user demand dynamics. As a result the response of the model checker is no more in Boolean form (true or false), but rather the possible evolution paths are weighted with probabilities. Requirements can thus be weighted with probabilities too. For example instead of requiring that the secondary temperature T_2 be always below, say, $141C$, we may require that $T_2 \leq 141C$ with an assigned (high) probability.

6.1. Model checking using PRISM

PRISM [20,19,22] is a probabilistic model checker. It allows modeling and analysis of systems exhibiting probabilistic behavior. Essentially PRISM requires two input files: a description of the system to be analyzed and a set of properties to be checked.

The system to be analyzed is defined using a simple state-based language. From this description, PRISM constructs a probabilistic model, namely: a DTMC, a Markov Decision Process (MDP) or a Continuous-Time Markov Chain (CTMC). Here our attention is focused on DTMC's.

For example, given a DTMC $\mathscr{M}$ and an initial state I for $\mathscr{M}$ PRISM can compute, for each state x of $\mathscr{M}$, the probability that an infinite sequence of transitions (a *path*) reaches x from I. To carry out its computation PRISM uses *numerical* methods (based on sparse matrices), Ordered Binary Decision Diagram (OBDD) [6] based methods as well as a *hybrid* approach based on a combination of numerical methods and OBDDs. Numerical methods tend to be faster than OBDD based methods, however numerical methods consume more memory. Thus, OBDD based methods become necessary for large systems. For middle size systems a hybrid approach usually works better.

One may think that the model used for NuSMV can be translated easily to a PRISM model. However the arithmetic support of PRISM is weaker than that of NuSMV, e.g. the division operation is not available within PRISM. This forced us to change considerably our NuSMV model before feeding it to PRISM.

As a matter of fact we followed the same approach used in Section 4.2 to build a HyTech model. We discretize each state variable and precompute the next state value for that variable using the equations of the hybrid automata in Fig. 10. This prevents us using the division operator which is not available in PRISM.

Of course the smaller the discretization step the better the approximation. For the temperature we used a discretization step of 0.5 C. This is quite rough, however making the discretization step smaller leads to a model file too big for PRISM. As for HyTech, we wrote a shell script generating a PRISM model along the above lines.

Figs. 19–21 give a glimpse of the PRISM model derived from the NuSMV model by allowing probabilistic changes in the user demand. Fig. 19 shows constants and variable declarations. The state variables are the same used for the HyTech model in Section 4.2 (or for the NuSMV model in Section 5 plus some book-keeping variables (namely: `delay`, `sck` and `counter`). Fig. 19 also shows the modeling in PRISM of the switching rules (*marking rules*) between the two locations (`P1` and `P2`) of the hybrid automaton of Fig. 10. Variable `sck` in Fig. 19 is used to schedule our model activities. When `sck` = 0, state variables `T1`, `T2`, `y1`, `y2` are updated (system dynamics). When `sck` = 10, we compute the next marking (i.e. variable `marking` is updated). When `sck` = 15, the user demand is updated.

The rather coarse grain of the PRISM model forced us to adjust the constants used in the NuSMV model. The verification strategy must also be changed. In fact in the PRISM model we must stick to values of the user demand u in the range [1,3]. Going to the range [1,4] would lead to a too big model file.

Thus rather than analyzing the system behavior with respect to variations in the user demand range, here we study the system behavior with respect to variations of the user demand dynamics. Intuitively, if the user demands changes *too quickly* (with respect to the model parameters) system specifications will not be satisfied since the controller will not be able to restore the standard operating conditions and errors will add up.

Fig. 20 shows our PRISM modeling for the user demand dynamics. Indeed, apart from the necessary changes due to a different language semantics, this is the original part in the probabilistic modeling w.r.t. the nondeterministic modeling used with HyTech (Section 4.2) and NuSMV (Section 5).

We model the user demand with a variable u that can take values in the set $\{1, 2, 3\}$. Moreover we constrain the dynamics of u by requiring that u must stay at the reference value ($u = 2$) *long enough* (such time is measured by the variable `delay` in Fig. 20). Moreover we require that, within our time horizon, the number of changes of u values be bounded. The number of changes of the user demand u are counted by the variable `counter` in Fig. 20.

When u is allowed to change its value (rule `uchange` in Fig. 20) u does so with the following probabilities: $\text{Prob}(u = 1) = \text{Prob}(u = 3) = 0.3$ and $\text{Prob}(u = 2) = 0.4$. That is the user demand u is a discrete random variable with finite support between $u = 1$ and $u = 3$ and with mass given

```
// Temperature Control Model Using Discrete Time Markov Chains
probabilistic
// Parameters used during PRISM model generation (same meaning as in NuSMV model)
// T = 138C (T = 145C) is represented by T = 0 (T = 14) in PRISM model
// dy = 3, gamma = 10/9, alpha = 1, beta = 1
// Ys = 2, sp = 4, hys = 4, Umin = 1, Umax = 3
// Temperature discretization step: 0.5C.
// PRISM constants
const sp = 4; const hys = 4; const DELAY = 20; const MAXCOUNTER = 5;

module fg103
T1 : [0..14] init 7; T2 : [0..14] init 4; y1 : [1..3] init 2;
y2 : [1..3] init 2; marking : [1..2] init 1; u : [1..3] init 2;
sck : [0..20] init 0; delay : [0..20] init 20; counter: [0..5] init 5;
// marking rules
[] (sck = 10) & (marking = 1) & (T2 <= sp + hys)  ->
   1.0: (marking' = 1) & (sck' = 15);
[] (sck = 10) & (marking = 1) & (T2 > sp + hys) ->  1.0: (marking' = 2) & (sck' = 15);
[] (sck = 10) & (marking = 2) & (T1 >= sp)  ->  1.0: (marking' = 2) & (sck' = 15);
[] (sck = 10) & (marking = 2) & (T1 < sp) ->  1.0: (marking' = 1) & (sck' = 15);
```

Fig. 19. A glimpse of our PRISM model. Declarations and marking rules.

```
// Rule uchange. When delay is 0 user demand is allowed to change its value
[] (sck = 15) & (delay = 0) ->
   0.3 : u’=1 & (delay’ = DELAY) & sck’= 0 +
   0.4 : u’=2 & sck’= 0 +
   0.3 : u’=3 & (delay’ = DELAY) & sck’= 0 ;  // go to dynamics
// Rule rho. With probability rho = 0.1 we may set delay to 0 thus allowing
// u to change its value before standard conditions are restored by the controller.
[] (sck = 15) & (delay > 1) & (counter > 0) ->
   0.9: (u’ = 2) & (delay’ = delay - 1) & (sck’ = 0) + // go to dynamics
   0.1: (u’ = 2) & (delay’ = 0) & (counter’ = counter - 1) & (sck’ = 0);
        // go to dynamics (no resetting)
// When counter is 0 no more u changes are allowed
[] (sck = 15) & (delay > 1) & (counter = 0) ->
   1.0: (u’ = 2) & (delay’ = delay - 1) & (sck’ = 0);  // go to dynamics
// After at most DELAY time units controller succeeds in restoring
// setpoint conditions. (DELAY >= 9).
[] (sck = 15) & (delay = 1) ->
   1.0 : u’=2  & (delay’ = 0) & (T1’ = 7)  & (T2’ = 4) & (y1’ = 2)
   & (y2’ = 2)  & (sck’= 10);   // go to marking
```

Fig. 20. A glimpse of our PRISM model (continued from Fig. 19). Probabilistic selection of user demand.

by the relation:

$$[P(u = 1) = 0.3;\ \ P(u = 2) = 0.4;\ \ P(u = 3) = 0.3].$$

Under normal operation the user demand stays at its reference value *long enough* (namely, DELAY time units in our model). DELAY is the time needed by the controller to restore standard operating conditions after a user demand change. We verified that the controller needs at most 9 time units to restore standard operating conditions after a user demand change. Thus it is enough to choose a value of at least 9 for the constant DELAY. We relax the above requirement by allowing that the user demand u may change

```
[] (sck = 0) & (marking = 1) & (T1 = 0) & (T2 = 0) &
   (y1 = 1) & (y2 = 1) & (u = 1) ->
   1.0 : (T1’ = 1) & (T2’ = 0) & (y1’ = 2) & (y2’ = 2) & (sck’ = 10) ;
[] (sck = 0) & (marking = 1) & (T1 = 0) & (T2 = 0) &
   (y1 = 1) & (y2 = 1) & (u = 2) ->
   1.0 : (T1’ = 1) & (T2’ = 0) & (y1’ = 2) & (y2’ = 2) & (sck’ = 10) ;
[] (sck = 0) & (marking = 1) & (T1 = 0) & (T2 = 0) &
   (y1 = 1) & (y2 = 1) & (u = 3) ->
   1.0 : (T1’ = 1) & (T2’ = 0) & (y1’ = 2) & (y2’ = 2) & (sck’ = 10) ;
[] (sck = 0) & (marking = 1) & (T1 = 0) & (T2 = 0) &
   (y1 = 1) & (y2 = 2) & (u = 1) ->
   1.0 : (T1’ = 1) & (T2’ = 0) & (y1’ = 2) & (y2’ = 3) & (sck’ = 10) ;
...
[] (sck = 0) & (marking = 2) & (T1 = 14) & (T2 = 14) &
   (y1 = 3) & (y2 = 3) & (u = 2) ->
   1.0 : (T1’ = 14) & (T2’ = 12) & (y1’ = 1) & (y2’ = 1) & (sck’ = 10) ;
[] (sck = 0) & (marking = 2) & (T1 = 14) & (T2 = 14) &
   (y1 = 3) & (y2 = 3) & (u = 3) ->
   1.0 : (T1’ = 14) & (T2’ = 11) & (y1’ = 1) & (y2’ = 1) & (sck’ = 10) ;


endmodule
```

Fig. 21. A glimpse of our PRISM model (continued from Fig. 20). System dynamics.

```
// probability of reaching a state s.t. (T1 < 139) is at most 0.3.
P<=0.3 [ true U (T1 < 2) ]

// probability of reaching a state s.t. (T1 > 144) is at most 0.3.
P<=0.3 [ true U (T1 > 12) ]

// probability of reaching a state s.t. (T2 < 139) is at most 0.3.
P<=0.3 [ true U (T2 < 2) ]

// probability of reaching a state s.t. (T2 > 141) is at most 0.3.
P<=0.3 [ true U (T2 > 6) ]
```

Fig. 22. The properties we passed to PRISM for verification (i.e. PRISM.pctl file).

its value *before* `DELAY` time units are elapsed, with probability ρ. This is done in the rule `rho` in Fig. 20.

Fig. 21 gives a glimpse of the PRISM model for the system dynamics. This is obtained by discretizing all the differential equations in the hybrid automata of Fig. 10.

Fig. 22 gives the specifications we passed to PRISM. Again we require that temperatures be within given bounds, only with high probability in this case. In Fig. 22 this is done by asking that the probability of reaching a state violating a requirement is below a given probability threshold.

For each of the requirements φ in Fig. 22 we used the PRISM option `-verbose` to compute the probability of φ holding in the initial state.

The results are summarized in Table 3. The first row of Table 3 gives the probability ρ that the user demand u changes its value (rule `rho` in Fig. 20) before the controller restores the standard operating conditions. The rows from the second to the fifth give the probability of reaching a state satisfying the error condition shown in the first column (from the initial state) as a function of ρ.

Moreover PRISM output certifies that the specifications in Fig. 22 hold in *any* state. This means that from any state (not just from the initial state) the probability of reaching a state violating a requirement is less than 0.3. As to be expected, from Table 3, it is clear how an increase in the probability ρ of a change in the user demand (u) entails an increase in the probability of violating some of the requirements for T_2. The requirements for T_1 instead do not appear to depend from ρ.

7. Discussion and conclusion

Using a real world hybrid system as a case study we presented an approach to integrate FPNs and model checking via hybrid automata and discrete as well as probabilistic models. Such integration turned out to be conceptually useful but raised a series of questions and problems that we want to elucidate and discuss in the present section. The goal of this discussion is not to compare the real power of the different tools used in this study, but is limited to the experience gained within our specific application. Moreover, this discussion can be the base for future research work in the area of the integration of models and tools for performance, reliability and safety analysis.

7.1. Modeling language

Central to the present study is the search for a unifying modeling language. We think that FPN offers a convenient tool from this point of view, since they have the flexibility and generality to cope with hybrid, deterministic and stochastic systems. This statement is supported by the consideration that all the models used in this paper could be derived rather directly from the FPN model.

FPN can be solved analytically if some restrictions are met [11]. In the present study, a simulative approach has been adopted. Automatic conversion of FPN into hybrid automata has been the object of recent research [24], and more general structures can be envisaged. However, as

Table 3
Probabilities of requirement violation as a function of the probability ρ that the user demand u changes its value. The first row gives ρ. Rows from second to fifth give the probability of reaching a state satisfying the error condition shown in the first column

ρ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$(T_1 \leq 139)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$(T_1 \geq 141)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$(T_2 \leq 139)$	0.0	0.171	0.195	0.201	0.205	0.21	0.216	0.223	0.232	0.244
$(T_2 \geq 141)$	0.0	0.171	0.195	0.201	0.205	0.21	0.216	0.223	0.232	0.244

the present case study has shown, the resulting hybrid automaton may be not in linear form and hence may not be directly analyzed using the HyTech tool. To use HyTech on such models some manipulation may be necessary. This, in turn, may lead to a *too big* state space (*state explosion*).

The derivation of discrete models (NuSMV-like) from FPN seems the most natural passage. However, when discretizing continuous variables, care should be given in the adopted grid points. An additional problem in NuSMV is the need of rescaling the parameters. Combination of Petri nets with formal methods has been the object of previous research [5].

The PRISM model checkers allows analysis of Discrete Markov Chains. However its modeling language has little support for arithmetic (e.g. the division operator is not present). This is not a big problem when analyzing protocols (the usual PRISM *target* systems), however this may force heavy manipulation of the model when working with hybrid systems.

7.2. Probabilistic vs nondeterministic

This dilemma was the main reason of this research work. In probabilistic models the timing of events, and the uncertainty in the knowledge of the model parameters is represented by means of random variables. Typical fields of application are performance evaluation and dependability analysis, and the obtainable measures are mean values and distributions.

In nondeterministic models the timing of events is represented by constant values or nondeterministic ranges. Typical fields of application are real time and time critical systems and safety critical systems. The obtainable measures are reachability properties via CTL and model checking.

Probabilistic model checking mixes the two paradigms allowing to define reachability properties in terms of probability of occurrence.

A complete characterization of a hybrid system for the control of safety critical application requires both performance (probabilistic) and reachability properties to be analyzed. Indeed, it is exactly this need that motivates our work here.

7.3. Simulation vs exhaustive exploration

A reachability analysis via Model Checking computes *exactly* all reachable states of a finite state system (*Exhaustive Exploration*). This is equivalent to run *all* possible simulations for that system. Of course, because of the huge state space, running all possible simulations is impossible. Model checkers achieve this result by using suitable algorithms and data structures.

For example when a model checker like NuSMV in Section 5 says that when the user demand u is in the range [1,3] requirements are never violated it means that there is no system evolution that can possibly lead to a requirement violation. On the other hand if in a simulation run requirements are not violated we cannot conclude that there is no system evolution that can lead to a requirement violation.

However, one must always keep in mind that the computational complexity of model checking is far greater than that of simulation. Thus model checking is to be used only when simulation does not suffice.

In summary, if we are interested in *typical* behaviors or to errors occurring with a *high enough probability*, than we may want to use simulation. On the other hand to look for *low probability* errors or to gain a very high confidence in the system correctness w.r.t. given requirements (as needed for safety critical system) we may want to use model checking. Usually one uses simulation first. When no more errors are found by simulation then model checking may be used to get rid of the remaining *hard to find* errors.

7.4. Automatic analysis of probabilistic models

Automatic analysis of probabilistic models can be carried out either using a S-FPN solution package, or using a probabilistic model checker like PRISM (Section 6).

S-FPN solution is based on the computation of the transient or steady state probability distribution of the underlying stochastic process. From that solution, various performance indexes can be derived. These indexes may represent some performance measure, like a throughput, or some other characteristic of the system, like a mean temperature.

Very detailed information can be derived from the solution of the underlying stochastic process. However, as outlined in Section 3.1, in order to compute this solution, a very complex system of partial differential equations must be solved. For this reason automatic analysis of S-FPN is only possible for very small models.

Given a discrete Markov chain $\mathscr{M}$ and an initial state I for $\mathscr{M}$, PRISM can compute, for each state x of M, the probability that an infinite walk from I reaches x. PRISM uses numerical as well as OBDD based techniques to carry out its analysis. In some cases such techniques can also be used for the numerical solution of S-FPN, like in [12].

Usually PRISM can handle larger models that those tractable with S-FPN. Note however that with S-FPN we are considering hybrid models whereas PRISM only handles finite state models (namely, DTMCs).

7.5. Continuous vs discrete

In discrete models the state space is discrete and can be exhaustively searched for. The dynamic evolution of the system in time can be represented as a sequence of transitions among discrete states. The dynamics of the system under analysis can be arbitrary, however the applicability of the method is limited by the possibly huge amount of memory needed to store the whole state space (*state explosion*).

Hybrid models contain discrete as well as continuous variables in the same model, and have a greater modeling power. Typical examples are discrete controllers that control continuous variables (as the case study in this paper). The available analysis tools for hybrid systems (e.g. HyTech) are limited to linear hybrid automata, and this may pose severe restrictions when dealing with real cases. In fact, the possibility of modeling continuous variables may be lost since we may need to resort to a stepwise (discrete) linearization to handle nonlinear dynamics.

7.6. Scalability and complexity

The scalability and the complexity of the proposed approaches mainly depend on the hybrid automata solution component. The process of translating a FPN to Hybrid System is exponential in the dimension of the FPN: it requires the creation of its reachability graph which is done through a depth-first visit of its state space. This step is clearly exponential in the dimension of the model (see for example [11]). After the model has been translated, the complexity of the analysis depends on the complexity of the algorithms used by the Model Checker to analyze the obtained hybrid systems.

Although model checking worst case complexity is exponential in the size of the input, model checkers performs reasonably well in many practical cases. However, depending on the system at hand, one model checker may perform better than another. This is because each model checker is optimized for a certain class of systems.

In our case HyTech and PRISM had a much harder time than NuSMV to complete our verification task. This forced us to considerably scale down the models we used for HyTech and PRISM. We should note however that HyTech handles continuous time models (whereas NuSMV only handles discrete time models) and PRISM handles probabilistic models (whereas NuSMV only handles nondeterministic models).

The scalability of our technique is thus limited by two different factors: the exponential complexity of the translation process, and the (possibly exponential) complexity of the Hybrid Automata analysis technique. At

Table 4
Comparison of the various analysis techniques

Tool	Main features	Best suited for
Simulation of S-FPN	Can be used on any kind of system. Computationally very efficient both in space and in time. On the other hand it only explores a *randomly chosen subset* of the state space. Simulation is very useful to validate our model and to find the most evident (i.e. *statistically easy to find*) errors. Errors that occurs only in a small number of system evolutions may easily go undetected in a simulation analysis.	Validation and testing of any model.
NuSMV	Can only be used for finite state systems, thus if our system model is not finite state we must discretize it. Space and time complexity can be exponential in the size of the description of the system to be analyzed (*state explosion*). However the *all* state space is explored. Thus all errors are found. For each error E the tool returns a possible system evolution (counter-example) leading to error E.	Verification (i.e. exhaustive exploration) of moderate size finite state models.
Hy Tech	Can only be used on hybrid systems where the continuous dynamics is defined using constant bounds on time derivatives (*linear hybrid automata*). When our model has a more complicated continuous dynamics, we must approximate our model by splitting it into many submodels having constant bounds on time derivatives. Space and time complexity can be exponential in the size of the description of the system to be analyzed. The *all* state space is explored, thus all errors are found. For each error E the tool returns a possible system evolution (counter-example) leading to error E.	Verification (i.e. exhaustive exploration) of small size linear hybrid automata.
PRISM	Can be used on *Discrete Time Markov Chain*. Space and time complexity can be exponential in the size of the description of the system to be analyzed. PRISM performs an exact analysis of the input Markov chain. E.g. given an initial state I, for each reachable state x PRISM Compute exactly the probability of reaching state x from I with a walk of infinite length.	Exact probabilistic analysis of moderate size discrete time Markov chains.
Numerical solution of S-FPN	Can be used on any kind of system. Space and time complexity is very high since a system of partial differential equations is to be solved. For this reason it can only be used on small models. On the other hand the results produced are very detailed.	Exact analysis of small models, e.g. detailed models of subsystems.

the present time, these constraints limit the applicability of the proposed technique only to very small (in term of FPN description elements) models.

7.7. When and what

As in Section 7.3 the approach to be used depends on our goals. Stochastic simulation of the performance model is to be used if we are mainly interested in average behaviors. Probabilistic model checking is to be used if we are also interested in *low probability* events. For example, this is the case for safety critical systems. Numerical solution of the performance model is to be used when we are interested in some complex performance index like the distribution of the time after which a low probability event occurs.

For example we used simulation in Section 3.3 to obtain distributions and average values, whereas we used PRISM to compute the *exact* probability of given events (namely, violation of requirements). We could have not applied numerical solution of the FPN, because it would have been too complex for any existing machine.

Essentially the various tools we used trade precision of the analysis with model complexity. Thus depending on the level of details to which we are interested in we will use a tool or another. Basically we may go from an approximate analysis (e.g. by simulation) of very complex models to a highly detailed analysis of low complexity models. Model complexity can be limited in several ways, each of which generates a class of analysis tools. For example, we may limit model complexity by restricting ourselves to finite state systems (as in NuSMV and PRISM). Model complexity can also be limited by restricting the continuous dynamics of our system to have, essentially, constant values for time derivatives (e.g. as in HyTech).

Table 4 summarizes the main features and the main intended application (column *best suited for*) of the tools we used in our case study. From Table 4 it is quite clear that all of the tools we examined are needed to cover the full range of models occurring in the design and analysis of a complex systems like the one (Section 2) considered in this paper.

7.8. Future trends: heterogeneous models

We have compared several modeling paradigms on the same case-study. The usability of a model can be classified according to its *modeling power* (the ability of the technique to allow an accurate and faithful representation of the system) and its *decision power* (the ability of the technique to be analytically tractable and to provide results with reasonable space and time complexity). These two features are in competition and a single modeling paradigm may not be sufficient in any practical situation. Our research effort in heterogeneous modeling is to explore the possibility of combining stochastic and deterministic timing and discrete and continuous (hybrid) variables in the same framework. A modeling and analysis tool supporting these ideas is under development [10,25].

References

[1] Alla H, David R. Continuous and hybrid Petri nets. J Syst Circuits Comput 1998.

[2] Allam M. Sur l'analyse quantitative des réseaux de Petri hybrides: une approche baseée sur les automates hybrides. Technical report, PhD Thesis. Institut National Polytechnique de Grenoble (in French); 1998.

[3] Alur R, Henzinger TA, Ho PH. Automatic symbolic verification of embedded systems. IEEE Trans Software Engng 1996;22.

[4] Baier C, Haverkort BR, Hermanns H, Katoen JP. Model checking continuous-time Markov chains by transient analysis. In: Emerson EA, Sistla AP, editors. Computer aided verification. LNCS, Springer Verlag, vol. 1855.; 2000. p. 358–72.

[5] Bobbio A, Horváth A. Petri nets with discrete phase timing: a bridge between stochastic and functional analysis. In: Corradini F, Vogler W, editors. Second International Workshop on Models for Time-Critical Systems (MTCS 2001) Electronic Notes in Theoretical Computer Science, vol. ENTCS-52. Amsterdam: Elsevier Science Publishers; 2002. URL: http://www.elsevier.nl/locate/entcs/volume52.html.

[6] Bryant R. Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput 1986;C-35(8).

[7] Ciardo G, Nicol DM, Trivedi KS. Discrete-event simulation of fluid stochastic Petri nets. IEEE Trans Software Engng 1999;2(25): 207–17.

[8] Clarke EM, Emerson EA, Sistla AP. Automatic verification of finite state concurrent systems using temporal logic specifications: a practical approach. ACM Trans Program Languages Syst 1986;8(2): 244–63.

[9] Emerson EA, Mok AK, Sistla AP, Srinivasan J. Quantitative temporal reasoning. J Real Time Syst 1992;4:331–52.

[10] Gribaudo M. FSPNEdit: a fluid stochastic Petri net modeling and analysis tool. Technical report, Tools of Aachen—International Multiconfernce on Measurements Modelling and Evaluation of computer Communication Systems—University of Dortmund, Bericht No. 760/2001; 2001.

[11] Gribaudo M. Hybrid formalism for performance evaluation: theory and applications. Technical report, PhD Thesis, Dipartimento di Informatica, Università di Torino; 2001.

[12] Gribaudo M, Horváth A. Fluid stochastic petri nets augmented with flush-out arcs: a transient analysis technique. IEEE Trans Software Engng 2002;28(10):944–55.

[13] Gribaudo M, Sereno M. Simulation of fluid stochastic petri nets. In Proceedings MASCOTS'2000, San Francisco, CA; 2000. p. 231–9.

[14] Gribaudo M, Sereno M, Horváth A, Bobbio A. Fluid stochastic Petri nets augmented with flush-out arcs: modelling and analysis. Discrete Event Dynamic Syst 2001;January.

[15] Henzinger TA, Ho PH, Wong-Toi H. Hytech: a model checker for hybrid systems. Software Tools Technol Transfer 1997;1.

[16] Horton G, Kulkarni V, Nicol D, Trivedi K. Fluid stochastic Petri nets: theory, application and solution techniques. Eur J Operat Res 1998; 105(1):184–201.

[17] Horváth A, Gribaudo M, Bobbio A. From FPN to NuSMV: The temperature control system of the ICARO cogenerative plant. Technical report, Università del Piemonte Orientale; Feb 2002: http://www.di.unipmn.it/Tecnical-R/TR-INF-2001/tr-2002-02-02.pdf

[18] HyTech; http://www.eecs.berkeley.edu/~tah/HyTech.

[19] Kwiatkowska M, Norman G, Parker D. Prism: probabilistic symbolic model checker. In Proceedings TOOLS 2002, vol. 2324; LNCS, Springer Verlag; April 2002.

[20] Kwiatkowska M, Norman G, Parker D. Probabilistic symbolic model checking with prism: a hybrid approach. In: Proceedings TACAS' 02. vol. 2280, LNCS, Springer Verlag; 2002.

[21] Murata T. Petri nets: properties, analysis and applications. Proc IEEE 1989;77:541–80.

[22] PRISM; http://www.cs.bham.ac.uk/~dxp/prism/

[23] NuSMV: http://nusmv.irst.itc.it/index.html

[24] Tuffin B, Chen DS, Trivedi K. Comparison of hybrid systems and fluid stochastic Petri nets. Discrete Event Dynamic Syst 2001;11(1/2): 77–95.

[25] Vittorini V, Franceschinis G, Gribaudo M, Iacono M, Mazzocca N. DrawNet + + model objects to support performance analysis and simulation of complex systems. In Proceedings 12th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (Tools 2002); 2002.