

**OBJECT ORIENTED APPROACH FOR CANE SUGAR PRODUCTION:  
MODELLING AND ANALYSIS**

**E. Villani\*, J. C. Pascal<sup>+</sup>, P. E. Miyagi\*, R. Valette<sup>+</sup>**

*\* Escola Politécnica, University of São Paulo*

*Av. Prof. Mello Moraes, 2231 CEP 05508-900 São Paulo, BRAZIL*

*<sup>+</sup> Laboratoire d'Analyse et d'Architecture des Systèmes – LAAS / CNRS*

*7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 FRANCE*

*e-mail: [evillani@usp.br](mailto:evillani@usp.br), [jcp@laas.fr](mailto:jcp@laas.fr), [pemiyagi@usp.br](mailto:pemiyagi@usp.br), [robert@laas.fr](mailto:robert@laas.fr)*

**Abstract:** This paper introduces a new approach for the modelling and verification of behaviour properties in complex industrial plants that are hybrid in nature. It is based on Petri nets to represent the discrete view and differential equations to describe the continuous part. The object-oriented concepts are used to provide modularity and handle system complexity. With the respect to the system analysis, the object-oriented structure allows a global analysis problem with a number of objects to be divided, into a set of local problems involving one or a few objects only. The proposed approach is applied to a cane sugar factory. Copyright © 2003 IFAC.

**Keywords:** hybrid systems, Petri nets, object modelling techniques, analysis.

## 1. INTRODUCTION

With system integration and computer automation becoming increasingly popular in industrial plants, sophisticated hybrid systems, i.e., systems involving both discrete and continuous dynamic have to be handled (Antsaklis et al., 1998). As a result a major issue in control system design for industrial

applications is that of ensuring that the desired behaviour of the plant may be obtained throughout its operation. To guarantee reliability, control system and plant should both be modelled in a hybrid formalism, and then analysed using appropriate tools. However, given the growing complexity of these systems, modelling and analysis cannot be easily addressed by the techniques already developed, suitable for simple applications only.

In the field of hybrid system analysis, one of the major issue is the verification of the behavioural properties of the model, e.g., proving that a forbidden state will never be reached. However, most of the works already published can only be applied to special classes of hybrid systems. An example is the verification tool UPPAL (Amnell et al, 2000), for which the model must be reduced to a timed automaton. Other works are based on linear hybrid automata, such as (Gueguen & Zaytoon, 2001) and the verification tool HyTech (Henzinger et al, 1997). Only a few approaches, such as the verification tool Checkmate using non-linear hybrid automata, support non-linear models, but they cannot easily deal with large-scale systems (Silva et al, 2001). These restrictions result from :

- the non-decidability issue, as the property cannot be proven with a finite number of steps. Thus, as shown by (Alur et al, 1995), if continuous variables with different growing rates (different derivatives) are included in the model, reachability may become undecidable. Generally, this is the case of hybrid systems.
- the fact that even decidable problems may be computationally untractable due to the explosion of the number of states, related to hybrid nature of the model.

Therefore, the aim of this paper is to introduce a new approach for hybrid system modelling and analysis. Unlike the aforementioned references, Petri nets are used for modelling the discrete part, and differential equation systems address to the continuous part. No restrictions are imposed on continuous dynamics. The object-oriented concepts are designed to provide a structured modularity and deal with complexity at the modelling phase. Once the model is built, it can be analysed either by simulation or the formal verification of properties. Simulation is particularly attractive because it can be easily performed by adapting existing simulators (such as MatLab and Simulink), and can be used for observing the behaviour of the system under a particular scenario. However, as it shows just one of the possible scenarios for the system evolution, it cannot be used to guarantee the absence of errors in the system's behaviour. Thus,

with the respect to the analysis phase, this paper focuses the formal verification of model properties. The proposed approach uses linear logic to address the discrete state explosion problem. The main innovation is that it exploits the modular structure of the model (based on object-oriented concepts) to decompose the system analysis. Thus, an analysis problem, that would otherwise involve the overall system model, is divided into a set of simpler analysis problems regarding the model of one or a few objects. Another important point is that it is not entirely automated. A more balanced solution is proposed where the designer's knowledge of the system is used in order to reduce the solution space and avoid non-decidability (although no guarantee of a solution can be given).

The proposed approach is applied to a cane sugar factory, as described in Section 2. Section 3 introduces the object-oriented concepts to the Differential Predicate Transition Petri nets. In Section 4, the analysis approach is presented and, in Section 5, it is illustrated by the verification of a safety property for the system of Section 2. Finally, Section 6 draws a number of conclusions.

## 2. THE CANE SUGAR PRODUCTION PROCESS

The cane sugar factory considered in this paper is that of Usina Guarani, in Brazil. A cane sugar production consists of a series of mechanical and chemical processes grouped in stages. Some stages could be classified as batch processes while others are essentially continuous. A diagram of the process is given in Fig. 1.

When a cane truck drives in, a crane carries the cane to a conveyor (Reception Process). In the Preparation Process, the cane is chopped, shredded and sent to the milling train, composed of four to seven mills. At each mill, cane juice is collected and the cane fibre continues to the next mill. The juice of the last mill is used to soak the cane of the precedent mill and so on. Only the juice of the first and second mill is collected and routed to the next stage, where it is mixed with  $\text{SO}_2$  (to break sucrose molecules into glucose molecules). To restore the juice pH limewater is added. The juice is then pumped through a heater and sent to the clarifier. Here, the juice passes through a number of compartments where particles settle. The juice is sent to the five-stage evaporator to produce cane syrup. Evaporation is a continuous process where the sucrose concentration increases. The continuous flow of syrup is then stored in a tank

acting as capacity element before crystallisation in the vacuum pans, which is a batch process. In vacuum pans, the sugar syrup concentration is further increased, resulting in a sugar dough subsequently routed to the centrifuges, dried and packaged.

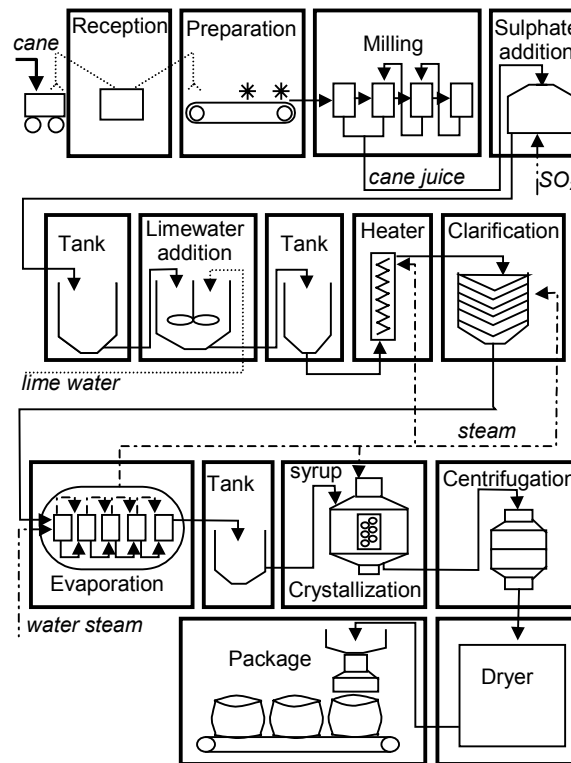


Figure 1. Cane sugar production process.

### 3. THE MODELLING APPROACH

#### 3.1. System decomposition and determination of the set of system classes

The first step in modelling is the decomposition of the system into a set of objects, grouped into classes. These objects should be highly correlated to closely match with the physical/real entities of the system, with a strong internal cohesion and only little relation to the other objects (Booch et al, 1998). Interactions among system entities are particularly important, resulting in communication among model objects and directly affecting the analysis decomposition, as shown in Section 4.

In hybrid models, communication can be either: discrete (represented by method calls) or continuous (sharing the values of object attributes). As a result, the following set of rules is defined to ensure object independence.

*1<sup>st</sup> Rule – Objects originating from decomposition can only have discrete interactions. Furthermore, method calls between them should be minimised.*

With the respect to the sugar production, the ‘Reception’ is the only process whose equipment behaviour can be modelled having discrete interactions only. All other processes involve a continuous flow of material, which results in a continuous communication among its objects. Examples of classes defined by applying the 1<sup>st</sup> Rule are *Truck* and *Crane*. They interact through methods such as ‘Load a bunch of cane’, which is modelled as a discrete event that decreases the amount of cane in the truck.

Since the first rule can result in complex classes, a trade off can be engineered between modularity and autonomy. Thus, a second rule is introduced and used for the decomposition of the remaining part of the system.

*2<sup>nd</sup> Rule - New classes resulting from decomposition can have continuous interactions but restricted to time intervals.*

This rule is closely related to batch processes, where interactions with other pieces of equipment are restricted to the phases of loading and unloading products. Example of classes defined by applying this rule to the cane sugar production are *Vaccum Pan* and *Centrifuge*. If this new rule is not enough to decompose the system into a set of simple classes, a third and last rule can be applied.

*3<sup>rd</sup> Rule - New classes resulting from decomposition can have continuous interactions not restricted to time intervals but closed loop of variable sharing are forbidden (e.g. when Object 1 uses an output variable of Object 2 that uses a variable of Object 1).*

This rule deals with pieces of equipment with a continuous process. Examples are *Heater*, *Milling Train*, *Evaporator*.

Closed loops of variable sharing are not always easily identified. Circles of continuous material flow usually results in closed loop of variable sharing. This is the case when a product continuously generated

by a certain step of the process is continually used by the previous step. If the decomposition proposed for the system does not fulfil the third Rule, the decomposition must be revised. The main reason for not admitting closed loop of variable sharing is that the equation systems of all the objects of the loop should be solved as a single one, resulting in a strong dependence among objects, which is contrary to the object-oriented paradigm. Furthermore, a closed loop of variable sharing may prevent any decomposition of the analysis process.

On the whole, cane sugar production is decomposed into 73 classes. Each has well-defined interfaces and is modelled separately from the rest of the system using the Differential Predicate Transition Petri nets and the object-oriented concepts, as described in the next section.

### *3.2. Differential Predicate Transition Petri Nets (DPT Petri Nets) and Object-Oriented Concepts*

Briefly, a DPT Petri net defines an interface between differential equation systems and Petri net elements.

Its main features are (Champagnat et al, 1998):

- A set of variables ( $x_i$ ) is associated with each token.
- A differential equation system ( $F_i$ ) is associated with each place ( $P_i$ ): it defines the dynamic of the  $x_i$  associated with the tokens in  $P_i$ , according to time ( $\theta$ ).
- An enabling function ( $e_i$ ) is associated with each transition ( $t_i$ ): it triggers the firing of the enabled transitions according to the value of the  $x_i$  associated with the tokens of the input places.
- A junction function ( $j_i$ ) is associated with each transition ( $t_i$ ): it defines the value  $x_i$  associated with the tokens of the output places after transition firing.

For the introduction of the object-oriented concepts to the DPT Petri nets, the following statements are defined, based on class and object concepts of (Booch et al, 1998):

- The behaviour of a class is modelled by a DPT Petri net.
- The methods provided by a class are associated with transitions of the DPT Petri net.
- The attributes of the class are modelled as the set of variables of the DPT Petri net.
- The first variable of the set of variables attached to a token in a class net is the identity of an object.
- An object is represented by a token (or a set of tokens with the same identity) in the class net.

Discrete method calls are represented by the merging of two transitions, the first one in the class that provides the method and the second in the class that calls the method. Continuous interactions are modelled by sharing continuous variables among objects. The value of the shared variables is determined by one object and can be used by the junction function, by the equation systems or by the enabling function of other objects. Regarding the graphic representation, in the PTD Petri net class model, the transitions represented by a white rectangle are methods provided by the class, those represented by a black rectangle are methods called by the class and those represented by a line are internal transitions.

In the object-oriented domain, another important issue is the relationship among classes. The main relationships are *composition* and *inheritance*. According to the proposed approach, the composition is achieved by simply merging the net of the classes, i.e., the two transitions of a method call between the classes (the transition that calls the method and the transition that provides the method) are considered as a single transition. Composition is particularly useful when, after the decomposition of the system into objects, the resulting classes have closed loop of variable sharing among them. In this case, a single class can be defined by the composition of the previous classes. Regarding the inheritance relationship, its main advantage is the definition of a new class from another without the need for copying all the information of the original class. It is particularly useful in object-oriented programming languages, because it allows code reuse. Its application to the DPT Petri net would mean to define a class just by specifying the new elements of the net. However, the graphical representation of such a class would be impacted because the new elements do not have a meaning themselves. The inheritance relationship is, therefore, not considered in this work.

As examples of class modelling, this paper presents the DPT net of four classes: *On/Off Valve*, *Tank*, *Evaporator* and *Clarifier*, which are used to illustrate the analysis approach described in Section 5. The flow of cane juice leaves the clarifier, passes through the evaporator and is stored in the tank. The on/off valve, controlled by the vacuum pan, sets the output flow from the tank.

#### ***Model of the On/Off Valve class***

This valve provides the sugar syrup to the vacuum pan, which is a batch reactor. Its model is presented in Fig. 2. When the vacuum pan is operating, the valve opens and closes with a fixed time interval, switching between “Opened” state ( $P_5$ ) and “Closed” state ( $P_4$ ). The time interval ( $\theta_{aux}$ ) between switching is associated with the time cycle of the vacuum pan and is represented by the constants  $K_c$  (for the "Closed" state) and  $K_o$  (for the "Opened" state). The flow of syrup ( $q_v$ ) is considered as constant ( $K_v$ ) when the valve is opened. The syrup supply can be interrupted by calling the “Block valve” method (associated with transition  $t_3$ ). However, even if the method call is immediately accepted (firing of  $t_3$ ), the valve is effectively blocked (firing of  $t_2$ ) only when it goes to the “Closed” mode. This is because supply to the vacuum pan cannot be interrupted when a cycle has already started, avoiding damage to the production.

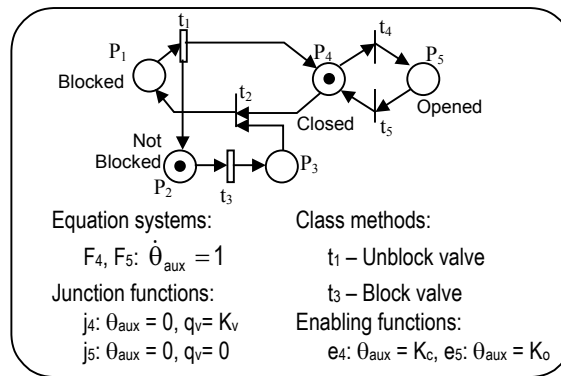


Figure 2. Model of the *On/Off Valve* class.

### **Model of the Tank class**

The tank model is presented in Fig. 3. Tank volume (variable  $V$ ) can vary between 0 and 100 (in relation to the percentage of its capacity). Its value depends on incoming ( $q_e$  - from the evaporator) and on the outgoing ( $q_v$  - to the vacuum pan). When in the “Normal” state ( $P_{10}$ ), if the volume reaches the upper limit of 96, the tank calls a method of the *Clarifier* class and goes to the “Alert” state ( $P_{13}$ ). If  $V$  continues to grow and reaches the 100 threshold, tank goes to the “Overflow” state ( $P_{14}$ ), which is a dead state (the system cannot return to the “Normal” state without external intervention). On the contrary, if  $V$  goes below the lower limit of 94 then another method of *Clarifier* class is called and the tank returns to the “Normal” state. A similar approach is used to avoid the “Empty” state ( $P_6$ ), another dead state. In this case, the methods are called from *On/Off Valve* class.

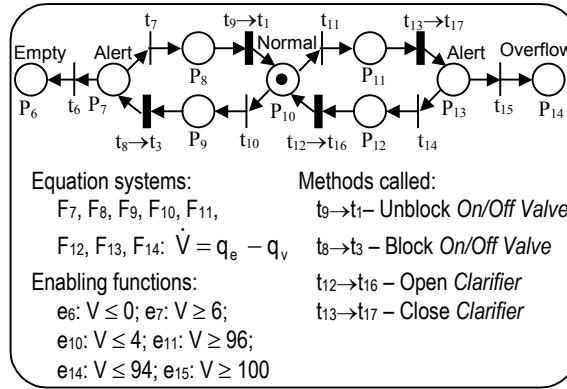


Figure 3. Model of the *Tank* class.

### Model of the *Evaporator* class

A simplified version of the *Evaporator* model is presented in Fig. 4. It considers only the relation between incoming flow ( $q_c$ ) and outgoing flow ( $q_e$ ), that are relevant variables for the property to be proven in Section 5. The model is composed by a single differential equation.  $K_e$  is the percentage of evaporated incoming flow and  $\tau_e$  is the time constant of the process. Roughly speaking, this time constant means that a variation of juice flow entering the *Evaporator* will result in a variation on the juice leaving the evaporator ( $q_e$ ) after a certain delay.

$$\text{Equation system: } \dot{q}_e = \frac{K_e * q_c - q_e}{\tau_e}$$

Figure 4. Model of the *Evaporator* Class.

### Model of the *Clarifier* Class

The model of the *Clarifier* is presented in Fig. 5. The juice flow entering the *Clarifier* is labelled  $q_{in}$  and is an external variable. The juice flow leaving the *Clarifier* ( $q_c$ ) is proportional to the incoming flow ( $q_{in}$ ) and to the quality factor of the juice ( $Q$ ), which is also an external variable.

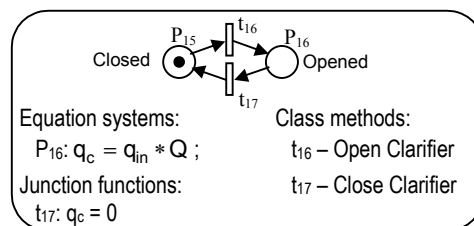


Figure 5. Model of the *Clarifier* Class.

## 4. THE ANALYSIS APPROACH

### 4.1. Decomposition of the Analysis Problem

As stated in the introduction, highlights of the proposed approach are the decomposition of analysis problems by dividing the model into objects and by decomposing them into a discrete part and a continuous one. Instead of analysing the overall model at once, each object (or a small set of objects) is analysed at each step. When it is necessary to reason globally on a set of objects, it will be possible to consider the discrete part or the continuous one only, thereby avoiding addressing them both at the same time. When verifying a property of an object, a set of hypotheses is made about the interaction with other objects. Thus, the property will be true in the analysed object if the set of hypotheses is also proven. Hypotheses are “proof obligations” therefore the initial proof is broken down into a sequence of local or simpler proofs.

Typically proof obligations are generated by possible interactions with other objects (method calls or shared variables). The *UML Collaboration Diagram* of the set of objects illustrates the possible sequences of proof obligation for an analysis problem. Considering the classes of Section 3.2, the *Collaboration Diagram* is presented in Fig. 6. The method call is modelled by a continuous arrow ( $\longrightarrow$ ), while the variable sharing is modelled by a hatched arrow ( $\dashrightarrow$ ).

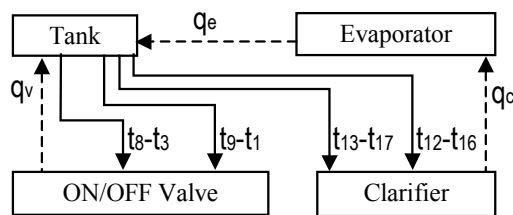


Figure 6. Collaboration Diagram of the example.

Each interaction of the *Collaboration Diagram* (each arrow) can result in a proof obligation. If a property is being verified for object *Tank*, proof obligations can be generated for object *Evaporator* (regarding  $q_e$ ), *ON/OFF Valve* (regarding  $q_v$  and the firing of  $t_8$  and of  $t_9$ ) and *Clarifier* (regarding the firing of  $t_{12}$  and of

$t_{13}$ ). When verifying the proof obligation for the *Evaporator* another proof obligation can be generated for the *Clarifier* (regarding  $q_c$ ).

#### 4.2. Analysis principle

Generally, the verification of behaviour properties can be classified into two main groups: safety property verification, i.e., to prove that a state or a set of states cannot be reached, and liveness property verification, i.e., to prove that a state, or a set of states, can always be reached. In both cases, for the proposed approach, if the property concerns the state and variables of a single object then this object is the first to be analysed, otherwise the set of objects concerned must be fused and analysed as a single one (under the form of a compound object).

In the analysis of a single object, linear logic is used as a formalism to explore the possibilities of object evolution. For the equivalence between Petri nets and linear logic, the reader is referred to the work of (Girault et al., 1997). Thus, a “sequent” in linear logic expresses reachability from an initial marking of the Petri net to a final marking by means of firing a set of transitions. The proof of the sequent is made by using a set of rules to verify that the sequent is correctly written (syntactic proof) and equivalent to the reachability proof for the Petri net. The interesting point here is that transition firings are not necessarily considered in sequence, and a partial order among them is derived from the proof. Regarding the methods for exploring possible scenarios of a Petri net, the approach of (Khalifaoui et al, 2002) is adopted. Forward reasoning and backward reasoning are defined for exploring the Petri net evolution. In forward reasoning the initial marking is completely or partially known. Transitions are fired (by applying the rules of linear logic) to determine the set of reachable states. In backward reasoning the starting point is the final marking and the aim is to explore possible scenarios that may lead to this final state. Backward reasoning is particularly useful for safety proofs and will be the focus of this paper, while forward reasoning is used for the proof of liveness properties.

#### 4.3. Backward reasoning

In this context, the methods proposed by (Khalifaoui et al, 2002) are considered with the modular approach presented in 3.1 for hybrid systems (modelled by DPT Petri nets). The main features of the association between Petri nets and linear logic for the backward reasoning are:

- An atomic proposition “p” is associated with each place “P” of the net and it represents the presence of one token in this place.
- Markings as well as pre and post conditions (Pre(t) and Post(t)) are represented by multiplicative formulas (monomials based on one of the two multiplicative connectives of linear logic that are not detailed here). Transitions are represented by implicative formulas (with the linear logic implication  $\multimap$ ) of the form: Pre(t)  $\multimap$  Post(t) [Girault et al, 1997].

Using this representation, the proof of the sequent:  $M_0, \sigma \vdash m_1, \Delta$  is used to derive a partially ordered list of events  $\sigma$  (transition firings) such that the partial marking  $m_1$  is reached. The partial marking  $\Delta$  corresponds to a set of tokens, necessarily produced when  $m_1$  is produced (side effect). The initial marking  $M_0$  is such that the transition firings in  $\sigma$  produce all the tokens in  $m_1$  and  $\Delta$ . As we deal with hybrid systems, the possible (from the discrete point of view) partially ordered list of events  $\sigma$  has to be consistent with some continuous initial state and continuous dynamics.

It is important to observe that linear logic does not distinguish propositions corresponding to different tokens in the same place. Therefore, objects cannot be identified. To avoid this difficulty, when a class has more than one object, the original net of the class (with the tokens of all the objects of that class) is “unfolded” in two or more nets, one for each object.

#### *4.4. Basic steps of the analysis*

The proposed approach is divided into the following steps:

*Step 1* – Build a scenario (or set of scenarios) in the object (using of a backward reasoning). By analysing causal relations, it shows how the forbidden state can be reached from a normal state. During this analysis, all state changes needed for the occurrence of the scenario (side-effects) are pointed out. When more than one scenario can lead to the forbidden state, the analysis approach is split into the number of

scenarios. Each scenario is analysed separately from the others and for each one, the next steps (Steps 2, 3 and 4) are performed. All scenarios must be false for the safety property to be true.

*Step 2* – Establish a list of hypotheses likely to support the proof. Each one is then considered as a proof obligation. They have to be proven (as lemmas). Hypotheses may involve the initial object or objects connected to the initial one either by means of discrete interactions (as pointed out in the preceding step) or continuous ones (shared continuous variables) as represented in the *UML Collaboration Diagram*.

*Step 3* – Prove that each scenario leading to the forbidden state is impossible when the continuous dynamics is taken into account. The possible behaviour of the continuous variables between the occurrences of the discrete events should be determined. Then the impossibility of the scenario may be proven:

- either by proving that a transition  $t_i$  that has to be fired is conflicting with another one  $t_j$  such that the enabling function of  $t_j$  will always be true before the enabling function of  $t_i$ ,
- or by proving that the continuous dynamics associated with one place is such that after firing of a transition producing a token in this place, it becomes impossible to fire the transition consuming this token in the defined scenario.

*Step 4* – Prove that each hypothesis made at Step 2 is true. This can be done by using the following approaches:

- by considering only the discrete aspects of one or a few objects – in this case, the traditional analysis tools developed for ordinary Petri nets can be used;
- by considering only the continuous dynamic of one or a few objects – in this case, any mathematical tool or theory developed for the analysis differential equation systems can also be used;
- or, if the hypothesis involves discrete and continuous aspects, by applying Steps 1, 2 and 3 recursively until the property is completely proven.

## 5. ANALYSIS OF THE EXAMPLE

### 5.1. General restrictions

For classes of Section 3.2, one of the safety properties to be verified is the non-reachability of state  $P_{14}$ , (tank overflow). In this case, the following restrictions are imposed:

- $R_1$ : The initial volume ( $V_0$ ) is equal to or less than 96 (in % of *Tank* capacity).
- $R_2$ : The initial value of  $q_e$  is within the interval  $[0, 0.35]$  (flow values are already divided by the *Tank* capacity and are in  $s^{-1}$ )
- $R_3$ : The initial states of the *Tank*, *On/Off Valve* and *Clarifier* are consistent: if  $P_{12}$ ,  $P_{13}$  or  $P_{14}$  is marked in *Tank* then the *Clarifier* is off, else the *Clarifier* is on. If  $P_6$ ,  $P_7$  or  $P_8$  is marked then the *On/Off Valve* is blocked.
- $R_4$ : The quality factor  $Q$  can vary between 0 and 1, while the incoming flow of the clarifier ( $q_m$ ) is within the interval  $[0, 0.7]$ .
- $R_5$ : The following values are chosen for the constants:  $K_v = 6.5$ ,  $K_o = 20$ ,  $K_c = 500$ ,  $K_e = 0.5$ ,  $\tau_e = 10s$ .

These restrictions define the property context and are assumed to be true. They are part of the analysis problem definition. Taking as an example restrictions  $R_1$  and  $R_2$ , there is no point in trying to prove the property if the *Tank* is almost full and the incoming flow is unlimited – the property will certainly be false. Similarly, in the case of  $R_4$ , there is no sense in considering that  $Q$  can assume any value, it will clearly not be higher than 1 (the juice leaving the *Clarifier*, after solid particles have settled down cannot exceed the juice entering the *Clarifier*). Or it cannot be zeros, as it is a fluid.

Restrictions are not part of the modelling activity because they can vary for different properties. For example: some restrictions may be imposed for the verification of a property related to the system performance under normal operation. In abnormal situations, the same performance may not be expected, meaning that in these cases the property does not need to be true. Restrictions are, therefore, used to overcome abnormal situations encountered during verification of the performance property. However, these restrictions are not valid for safety properties because the system should be safe even in abnormal situations. Incorporating the restrictions in the model would result in changing the model for each property.

### 5.2. Step1: building the scenarios

### *Analysis of Tank Object*

The property to be proven is that it is not possible to reach the overflow state (place  $P_{14}$ ) from a normal state (place  $P_{10}$ ). The sequent to be analysed is ( $M_0$  is such that there is a token in  $P_{10}$ , the remaining part is unknown):

$$M_0, \sigma \vdash p_{14}, \Delta$$

The only enabled transition is  $t_{15}$  (for backward firing). Firing of  $t_{15}$  results in  $\sigma = t_{15}, \sigma'$  and a new sequent to be proven:

$$M_0, \sigma' \vdash p_{13}, \Delta$$

Now, the only backward enabled transition is  $t_{13}$ , but this transition corresponds to a synchronization with the object *Clarifier* (transition  $t_{17}$  fired). In order to be able to backward fire the pair of transitions  $t_{13}$  and  $t_{17}$ , it is necessary to consider that  $\Delta = p_{15}, \Delta'$ . We must therefore prove that:

$$M_0, \sigma' \vdash p_{13}, p_{15}, \Delta'$$

After backward firing of  $t_{13//17}$ , and denoting  $\sigma'' = t_{13//17}, t_{15}, \sigma''$ , it is necessary to prove:

$$M_0, \sigma'' \vdash p_{11}, p_{16}, \Delta'$$

Finally, backward firing of  $t_{11}$  leads to:

$$M_0, \sigma''' \vdash p_{10}, p_{16}, \Delta'$$

with  $\sigma = t_{11}, t_{13//17}, t_{15}, \sigma'''$ .

It is now possible to prove this sequent if  $M_0$  contains just one token in  $P_{10}$  and one in  $P_{16}$ , and if  $\sigma'''$  and  $\Delta'$  are empty. Thus, the scenario from the normal state to the forbidden state “overflow” corresponds to the sequent:

$$M_0, t_{11}, t_{13//17}, t_{15} \vdash p_{14}, p_{15}$$

This scenario is the only possible way to reach  $P_{14}$ .

### *5.3. Step 2: Listing all hypotheses (generating proof obligations)*

The following hypotheses are made to prove the unfeasibility of the scenario:

- H<sub>1</sub>: *The proposition  $V \leq 96$  is always true when the Tank is on the Normal state ( $M(P_{10}) = 1$ ).*

The definition of the discrete states of *Tank* (Normal, Alert, Empty, Overflow) is connected to the value of its volume. Just like the condition that impose that the initial volume of the Tank must correspond to the normal state (restriction R<sub>1</sub>), it is reasonable to expect that during the system evolution when the Tank is in Normal state ( $M(P_{10})=1$ ) it will fulfil the condition  $V \leq 96$ . Furthermore, H<sub>1</sub> seems necessary because *Tank* overflow can be avoided if the *Tank* can accomodate the incoming flow ( $q_e$ ) between the firing of  $t_{11}$  and the effective end of flow  $q_e$ . If  $V=100$  when  $t_9$  or  $t_{12}$  are fired, the overflow place seems likely to be the next state. A straightforward consequence of this hypotheses is that  $V = 96$  when  $t_{11}$  fires.

- H<sub>2</sub>: *The Alert state of the Tank implies the Closed state of the Clarifier ( $M(P_{13}) = 1$  implies  $M(P_{15}) = 1$ ).*

H<sub>2</sub> is connected to R<sub>3</sub>. As the initial states of the objects must be consistent, it is reasonable to expect that they will maintain consistence throughout the system evolution. This hypothesis is necessary because the control action for avoiding the *Tank* overflow is closing the *Clarifier* (when a token is produced in P<sub>13</sub> of object *Tank*, a token is also produced in P<sub>15</sub> of object *Clarifier*), so it is important to know how long this control action will last, i.e., if the *Clarifier* remains closed while the *Tank* is on the state Alert (P<sub>13</sub>).

- H<sub>3</sub>: *The detection of an abnormal volume in the Tank ( $M(P_{11}) = 1$ ) implies that the Clarifier is opened ( $M(P_{16}) = 1$ ).*

H<sub>3</sub> is also connected to R<sub>3</sub>. Just like it is important to know how long the control action is needed for avoid the *Tank* overflow, it is also important to know if a delay will be needed between the detection of an abnormal volume ( $V > 96$  – firing of  $t_{11}$ ) and the control action (closing the *Clarifier* – firing of  $t_{13/17}$ ).

- H<sub>4</sub>: *The output flow of Evaporator is limited by  $0 \leq q_e \leq 0.35$*

H<sub>4</sub> is connected to R<sub>2</sub>. It is important to establish a bound for  $q_e$  because the derivate of  $V$  is  $q_e - q_v$  (equation system associated with P<sub>13</sub>). In order to avoid the *Tank* overflow,  $V$  must not increase too much (the second order peak must be smaller than the difference between the threshold attached to  $t_{15}$  (100) and the one attached to  $t_{14}$  (94)).

#### 5.4. Step 3: Taking the continuous dynamics into account

It is necessary to prove that the actual continuous behaviour is such that the scenario determined in Section 5.2 cannot occur. The first kind of situation to be analysed is that of conflicts. There is one conflict case in the scenario: when place  $P_{13}$  contains a token, transitions  $t_{14}$  and  $t_{15}$  can be fired. The threshold associated with transition  $t_{14}$  is  $V \leq 94$  and that of  $t_{15}$  is  $V \geq 100$ . When  $t_{13/17}$  is fired  $V = 96$  (from hypotheses  $H_1$  and  $H_3$ ), if for example  $V$  is increasing in place  $P_{13}$ , the presence of  $t_{14}$  does not prevent transition  $t_{15}$  from firing. The only way of proving that the scenario is impossible is by demonstrating that the continuous dynamics in place  $P_{13}$  is such that the firing of  $t_{15}$  is impossible. The continuous dynamics of  $V$  in place  $P_{13}$  involves the objects *Evaporator*, *ON/OFF valve* and *Clarifier* because the derivate of  $V$  depends on  $q_e$  and  $q_v$ , and  $q_e$  depends on  $q_c$  (see Fig. 6).

In order to reach the state of *Tank overflow* by firing of  $t_{15}$ , the enabling function of  $t_{15}$  ( $V \geq 100$ ) should be true:

$$V(\theta_{15}) = V(\theta_{13}) + \int_{\theta_{13}}^{\theta_{15}} (q_e - q_v) \cdot d\theta \geq 100 \quad (1)$$

From hypotheses  $H_1$  and  $H_3$ , it is stated that  $V = 96$  when the token appears in place  $P_{13}$ , therefore:

$$96 + \int_{\theta_{13}}^{\theta_{15}} (q_e - q_v) \cdot d\theta \geq 100 \quad \Rightarrow \quad \int_{\theta_{13}}^{\theta_{15}} (q_e - q_v) \cdot d\theta \geq 4 \quad (2)$$

This condition must be verified in order to reach the overflow state, i.e., in order for  $V$  to reach the value 100 when  $M(P_{13})=1$ . If there are no  $q_e(\theta)$  and  $q_v(\theta)$  that verifies the above condition then the overflow state will never be reached and the system will be safe. As  $q_e(\theta)$  and  $q_v(\theta)$  are variables of the objects *Evaporator* and *On/Off Valve*, the above condition implies in the analysis of these two objects. These proof obligations comply with the interactions indicated in Fig. 6.

#### ***Analysis of the ON/OFF Valve Object***

Since  $q_v(\theta)$  is always positive, the previous condition can be rewritten as:

$$\int_{\theta_{13}}^{\theta_{15}} q_e \cdot d\theta \geq 4 \quad (3)$$

The next object to be analysed is the *Evaporator*.

***Analysis of the Evaporator Object.***

From the equation system of this object, the condition (3) can be rewritten as:

$$\int_{\theta_{13}}^{\theta_{15}} q_e \cdot d\theta = \int_{\theta_{13}}^{\theta_{15}} [K_e \cdot q_c - \tau_e \cdot \dot{q}_e] d\theta = \tau_e \cdot [q_e(\theta_{13}) - q_e(\theta_{15})] + \int_{\theta_{13}}^{\theta_{15}} K_e \cdot q_c \cdot d\theta \geq 4 \quad (4)$$

According to H<sub>4</sub>, 0 ≤ q<sub>e</sub> ≤ 0.35 resulting in:

$$10 \cdot 0.35 - 10 \cdot 0 + \int_{\theta_{13}}^{\theta_{15}} K_e \cdot q_c \cdot d\theta \geq 4 \Rightarrow \int_{\theta_{13}}^{\theta_{15}} K_e \cdot q_c \cdot d\theta \geq 0.5 \quad (5)$$

Because q<sub>c</sub> is a variable from object *Clarifier*, this is the next object to be analysed.

***Analysis of the Clarifier Object.***

From H<sub>2</sub>, it is known that q<sub>c</sub> = 0 when M(P<sub>13</sub>) = 1. Therefore the condition (5) is not verified:

$$\int_{\theta_{13}}^{\theta_{15}} K_e \cdot q_c \cdot d\theta = 0 < 0.5 \quad (6)$$

If the condition is not true, V cannot reach the value 100 when M(P<sub>13</sub>) = 1 and the scenario determined in Section 5.2 will never happen according to the pre-defined hypothesis. Now, the hypotheses must be proven.

***5.5. Step 4 of the method: Proof of the hypotheses***

***H<sub>2</sub> and H<sub>3</sub> - Analysis of Tank + Clarifier Objects***

Proving H<sub>2</sub> and H<sub>3</sub> involve more than one object, but they uniquely involve the discrete view of the model. The Petri nets of objects *Tank* and *Clarifier* are fused by merging the transitions t<sub>13</sub> and t<sub>17</sub> on the one hand and t<sub>12</sub> and t<sub>16</sub> on the other hand. These hypotheses directly result from the two following p-invariants, which are highlighted by two different colours in the net shown in Fig. 7:

$$M(P_{12})+M(P_{13}) +M(P_{14}) = M(P_{15}) \text{ (in grey —)}$$

$$M(P_6)+M(P_7)+M(P_8)+M(P_9)+M(P_{10})+M(P_{11}) = M(P_{16}) \text{ (in black —)}$$

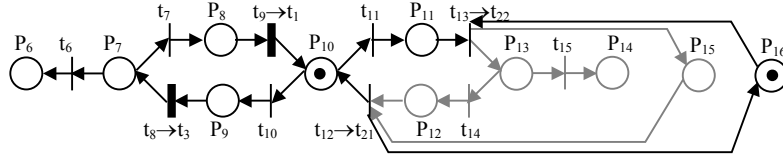


Figure 7. Resulting Petri net from the fusion of the objects *Tank* and *Clarifier*.

#### ***H<sub>4</sub> - Analysis of the Evaporator Object***

The proof of  $H_4$  is only based on considerations over the continuous dynamics. According to object *Evaporator*, the dynamic of  $q_e$  is determined by the following differential equation:

$$\dot{q}_e = \frac{K_e * q_c - q_e}{\tau_e}$$

When  $q_e > K_e * q_c$ ,  $\dot{q}_e < 0$ , therefore  $q_e$  decreases, approaching  $K_e * q_c$ . Similarly, when  $q_e < K_e * q_c$ ,  $\dot{q}_e > 0$  and  $q_e$  increases. The denominator of the differential equation is the time constant associated with the delay between  $K_e * q_c$  and  $q_e$ . Because  $q_e$  approaches  $K_e * q_c$ , the value of  $\dot{q}_e$  approaches 0, therefore  $q_e$  will never surpass the value of  $K_e * q_c$ . If the value of  $q_c$  changes,  $q_e$  instantaneously begins to follow the new value of  $K_e * q_c$ . As a consequence,  $q_e$  is limited by the values reached by  $K_e * q_c$ , i.e.,  $\max(q_e) \leq \max(K_e * q_c)$  and  $\min(q_e) \geq \min(K_e * q_c)$  (as long as  $q_e(0)$  is also within this interval). This statement is true independently of the implementation of the *Clarifier* object. If the *Clarifier* is changed then the analysis of the *Evaporator* does not need to be done again. This is one of the advantages of the proposed approach.

#### ***H<sub>4</sub> - Analysis of the Clarifier Object***

By considering the object *Clarifier*, the highest value of  $q_c$  is reached for  $Q=1$  when the *Clarifier* is ‘On’ and is  $\max(q_c)=0.7$ . The lowest value is reached when the *Clarifier* is ‘Off’ and is  $\min(q_c)=0$ . So the upper and lower bounds of  $K_e * q_c$  are 3.5 and 0 (these limits are assumed to be true at the initial state).

#### ***H<sub>1</sub> – Analysis of the Tank Object***

The proof of  $H_1$  is more complex. It requires proving that it is impossible to reach the place  $P_{10}$  in the object *Tank* with  $V > 96$ . The approach is similar to the preceding one, that is, it is first necessary to build all the scenarios leading to this place and then to prove that they are inconsistent with the proposition  $V > 96$  when the continuous dynamic is taken into account. It is assumed that the initial value of  $V$  is  $V_0 \leq 96$ . There are two scenarios leading to place  $P_{10}$ :

$$M_0, t_{14}, t_{12//16} \vdash p_{10}, p_{16}$$

with  $M_0$  consisting in one token in  $P_{13}$  and one token in  $P_{15}$ . As  $V \leq 94$  when  $t_{14}$  is fired and as there is no delay between the firing of  $t_{14}$  and that of  $t_{12//16}$ , then  $V > 96$  is inconsistent when the *Tank* reaches the normal state by the firing of  $t_{12//16}$ .

The second scenario is:

$$M_0, t_{10}, t_{8//3}, t_2, t_7, t_{9//1} \vdash p_{10}, p_4$$

with  $M_0$  consisting in one token in  $P_{10}$ , one in  $P_2$ , and one in  $P_4$ . When  $t_{10}$  is fired ( $V \leq 4$ ),  $t_{8//3}$  is fired without delay (the proof is similar to that of H2 and H3). Because this is the unique way of reaching  $P_7$ ,  $M(P_7) = 1$  implies  $0 \leq V \leq 6$ . When  $t_7$  is fired we have then  $V = 6$ . Transition  $t_{9//1}$  is fired without delay (same proof as above) and  $V > 96$  is inconsistent when the *Tank* reaches the normal state by the firing of  $t_{9//1}$ .

### 5.6. General comments about the above example

The aim of this example is to illustrate some important points of the proposed approach. The first one is that this approach does not deal with large models of the system where the state of all its components are considered at the same time. Instead of this, the proposed approach analyses each object (or small sets of objects) at a time and focuses on the causality constraints between them. Furthermore, if the state of one of the objects is not relevant for the proof, it is not taken into account (as in the case of proving hypotheses H<sub>2</sub>, H<sub>3</sub> and H<sub>4</sub>). In this manner, the state explosion problem can be avoided.

The use of DPT Petri nets, which represents the discrete dynamics and the continuous one with two distinct formalisms (Petri nets and differential equation systems) makes possible to prove some properties considering just one of the two aspects (discrete or continuous). By this way, analysis tools and theorem proving techniques developed for pure discrete or pure continuous systems, such as place invariants (H<sub>2</sub> and H<sub>3</sub>), can be used in the analysis problem, avoiding, whenever possible, the non-decidability problem.

Another important point is that the approach incorporates the user knowledge of the system within the analysis process by the definition of the hypotheses. These hypotheses are expected to be true based on the meaning of the model (its interpretation). Each hypothesis works as an intermediate lemma in a mathematical proof: it reduces a complicated proof into a concatenation of elementary proofs.

## 6. CONCLUSION

One of the problems of control system design is how to validate it, i.e., how to ensure that it behaves in accordance with a set of requirements previously defined. A possible way of dealing with this problem is to model both the control system and the plant using a chosen formalism, and to guarantee the requirements by proving behaviour properties of the model. Considering the particular case where the control system and the plant form a hybrid system, this paper proposes a solution to this problem by introducing a new approach for hybrid system modelling and analysis, based on the use of Petri nets and object-oriented concepts. Particularly, this paper considers the proof of safety properties.

By exploiting the object independence principle, a global analysis problem is decomposed into a set of local object proofs. By using the linear logic, the analysis is performed based on the partial order of the events, i.e. the causality among them. The internal evolution of each object can be analysed independently of the internal evolution of other objects.

Because this approach has been proposed for verifying properties of controlled plants (plant *and* control system are both modelled), it is expected that indeterminisms are solved “a priori”. If it is not the case, the analysis approach can still be applied, but it results in a larger number of scenarios to be analysed. An example of indeterminism is when the occurrence of fault is considered in the model, such as when a command to open a valve is given and the valve can open (normal behaviour) or not (abnormal or fault behaviour). According to the requirements of the system, it may be necessary to prove some safety properties under any circumstances, so the analysis must consider both behaviours.

It is important to highlight that the overall analysis approach cannot be automated because the user interaction is needed for the specification of hypotheses. A hypothesis is a statement about the system that is expected to be true, but it is not explicit in the models. It cannot be automatically deduced because it is not based on the modelling formalism but in the interpretation of the model. As a consequence, the analysis approach cannot be entirely performed by a computational tool. This restriction is not considered as an important limitation because the decidability problem of hybrid system cannot be solved, which

means that any automated approach will also be limited. Instead of that, as an attempt to avoid the non-decidability issue, this approach incorporates the user knowledge of the system through the definition of hypotheses and the use of Petri net properties. However, some steps of the approach can still be automated (such as the building of discrete scenarios by linear logic). As a result, user and computer aid can be incorporated in a synergetic way to solve complex problems.

Another consequence of the introduction of the user interaction is that the complexity of the analysis depends on the ability to specify hypotheses that, at the same time, are true and effectively help the analysis process. These hypotheses increase the analysis decomposition and make its application successful even for systems with a large number of objects. On the other hand, depending on the system, if the hypotheses are useless, the number of scenarios to be considered can increase so that it becomes untractable or, if the hypotheses are false, the analysis must be reconsidered with new hypotheses.

This approach has also been applied to other case studies, such as the air conditioning system of a building and the landing system of a military airplane.

#### ACKNOWLEDGMENTS

The authors would like to thank the partial financial support of the governmental agencies FAPESP, CNPq and CAPES.

#### REFERENCES

- Alur, R. et al. (1995). The algorithm analysis of hybrid systems. *Theoretical Computer Science* 138, 3-34.
- Amnell, T. et al (2000). UPPAAL – Now, Next and Future. *Proc. of MOVEP'2k: MOdelling and Verification of Parallel Processes*, Nantes, France.
- Antsaklis, P., et al., (1998). On Hybrid Control of Complex System: A survey. *European Journal of Automation* 32 (9-10), 1023-1045.
- Booch, G., et al, (1998). *The Unified Modeling Language User Guide*. Addison-Wesley Longman, Inc. Harlow, England.

- Champagnat, R. et al. (1998). Petri net based modelling of hybrid systems. *Computers in Industry* 36(1-2), 139-146.
- Delatour, J., & Paludetto, M. (1998). UML/PNO, A Way to Merge UML and Petri Net Objects for the Analysis of Real-Time Systems. *Lecture Notes in Computer Science* 1543, 511-514.
- Girault, F. et al. (1997). A logic for Petri nets. *JESA: Journal Européen des Systèmes Automatisés* 31 (3), 525-542, Editions Hermes.
- Gueguen, H., & Zaytoon, J. (2001). Principes de la vérification des systèmes hybrides, in: G. Juanole and R. Valette, *Modélisation des systèmes réactifs (Actes de MSR 2001)*, Hermès-Lavoisier (Paris), 427-444.
- Henzinger, T. A. et al (1997). HyTech: a model checker for hybrid systems. *Proc. of 9<sup>th</sup> International Conference on Computer-Aided Verification*, Haifa, Israel.
- Khalifaoui, S. et al (2002). An algorithm for deriving critical scenarios in mechatronic systems. *Proc. of IEEE International Conference on Systems Man and Cybernetics (SMC'02)*, Hammamet, Tunisie.
- Silva, B. I. et al (2001). An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems. *Proc. of 40th IEEE Conf. on Decision and Control*, Orlando, USA.