Performance verification of discrete event systems using hybrid model-checking

Bruno Denis, Jean-Jacques Lesage, Zulema Juarez Orozco

PERFORMANCE VERIFICATION OF DISCRETE EVENT SYSTEMS USING HYBRID MODEL-CHECKING

Bruno Denis [(1)], **Jean-Jaques Lesage** [(1)], **Zulema Juárez-Orozco** [(1), (2)]

[(1)] *LURPA-ENS de Cachan, France, {denis, lesage, juarez}@lurpa.ens-cachan.fr*
[(2)] *CIATEQ - Querétaro, Mexico, PhD grant financed by CONACYT (Mexico)*

Abstract: The results generated over the past few years on the formal verification of both Discrete Event Systems (DES) and Hybrid Dynamic Systems (HDS) are quite substantial, especially as regards the controller's properties of liveness and safety. In this paper, we will study the range of possibilities offered using the model-checking techniques in order to evaluate DES performances (in terms of quality of service provided by the automated system). This task calls for proceeding with a model-based approach that couples a hybrid model of the plant with a timed discrete model of the controller. We will also show, using a basic example, that by parameterizing the hybrid process model, the model-checker may then be employed to evaluate the robustness of the discrete control to perturbations encountered by the plant.

Keywords: DES controller, hybrid plant model, model-based verification, model-checking, linear hybrid automaton, HYTECH.

1. INTRODUCTION

A physical system is not, in most cases, intrinsically either purely discrete or purely continuous; instead, it's the abstraction the control engineer undertakes to ensure automation specification is being met that lends one distinction or the other. In the area of manufacturing systems for example, many processes have mobility axes that enable products to circulate, establish their position, etc.; consequently, some of the major physical variables controlled are either displacements or speeds. Depending not only on the level of quality to be guaranteed for these controlled variables, but also on the aggressiveness or variability in the external environment (e.g. type and importance of perturbations, parameter variation interval for the law of movement) and on the relative weight of economic constraints, the control engineer is required to choose between a servo control or a discrete control for each of these axes. Such automation-related choices will yield the physical variables to be observed and controlled by the controller, with either continuous or discrete control abstraction. Once the requisite displacement axis positioning quality has been achieved and provided that the perturbations encountered remain tolerable, the control engineer will then select a discrete control for obvious cost reduction reasons. This discrete displacement control, despite often being able to accommodate mobile positioning quality requirements, still constitutes an abstraction, and as such necessarily a simplification of the associated physical variables. The logic control of a linear displacement between two extreme positions, observed by means of two limit switch sensors, clearly does not enable ascertaining the precise position of the mobile, nor the time elapsed to complete this displacement.

Satisfying industrial system dependability requirements often necessitates conducting offline analyses, such as formal verification, before placing the automated system into operation (for further information on this topic, see the standard IEC 61508 entitled " Functional safety of electrical / electronic / programmable electronic safety-related systems "). These verifications, which are now frequently performed by means of model-checking (MacMillan, 1993), may be practiced by electing to incorporate or not a plant model.

In this paper, our efforts have focused on checking systems composed of a discrete controller coupled with a continuous (or partially-continuous) physical process whose entire set of observed and controlled variables constitute discrete abstractions of physical variables. To proceed, we will make use of model-checking techniques by coupling the discrete controller model with a hybrid plant model; this set-up will demonstrate that beyond the liveness and safety properties, it is indeed possible to check whether automated system performance is compatible with that stipulated in the specifications. We will also show that by parameterizing the hybrid process model, it becomes possible to use the model-checker to evaluate the robustness of discrete control to perturbations encountered by the plant.
This paper has been organized as follows. After having recalled the possibilities and limitations of both DES and HDS model-checking, we will introduce the expectations derived from a hybrid plant model for verifying a discrete controller. In order to illustrate our approach, the paper's second part will present the example of a positioning axis, which represents a component of a more complex assembly system. We will thus be able to show that use of a model-checker (such as HYTECH), by implementation of a hybrid process model, makes it possible to verify the expected performance of this DES. Furthermore, a sensitivity study conducted on model parameters will allow evaluating the robustness of discrete control when confronted with perturbations.

2. ACQUIRED KNOWLEGE AND LIMITATIONS FROM AUTOMATED SYSTEM VERIFICATION

2.1 DES verification.

Formal verification techniques stem from the field of computer science. Only recently have they been adapted and applied to DES verification and, more specifically, to model-checking (Clarke E. M., *et al*., 1986). The general principle behind model-checking may be expressed as follows (see Figure 1).

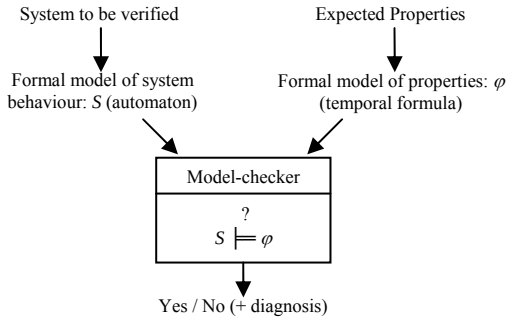

Fig. 1. Model-checking scheme.

Let's start with a system that has been designed to verify an entire array of properties (logical correctness, dependability, liveness, etc.). The first task of model-checking consists of formalizing system behavior in the form of a finite state automaton: *S*, plus the properties to be verified within a temporal algebra such as CTL (Emerson and Halpern, 1986): φ. The model-checker then conducts a thorough analysis of the state space reachable by *S*, which serves either to prove that $S \models \varphi$ (this algebraic statement denotes that "the system model satisfies the set of properties φ") or, when such is not the case, to propose a counterexample that revokes those properties not verified by S.
Moreover, a DES may be represented in a generic manner, as shown in Figure 2: a discrete controller acting in a closed loop on a plant. As part of a dependable controller design approach, the system being targeted for verification can thus be (according to Frey. and Litz, 2000) either the controller on its own, presumed to be operating within an open loop on the plant (a non "model-based" verification), or the {controller + plant} assembly set interacting within a closed loop ("model-based" verification).

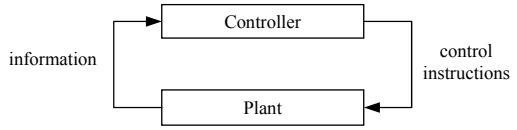

Fig. 2. A generic closed-loop DES.

The research work focusing on DES verification initially favored a non model-based approach (Moon I., 1994). The reachable state space of the controller model is thus to be built in the most permissive manner possible, i.e. such that the evolution of its inputs are in no way constrained by plant behavior. In this case, the safety properties capable of being demonstrated provide the basis for strong proof given that they can be demonstrated regardless of the evolution in controller inputs. On the other hand, a good number of liveness or accessibility properties cannot be demonstrated via a non model-based model-checking approach given the often fast-paced combinatory explosion of the reachable state space.
One means for reducing this combinatory explosion, using realistic constraints that depict the interaction of plant behavior with controller behavior, is to conduct a model-based verification. (Rausch and Krogh, 1998), (Machado, *et al.*, 2003).
Through reliance upon these results, we are now in a position to study the possibilities offered by the model-checking procedure in evaluating DES performance (in terms of quality of service provided by the automated system). To accomplish this task, it is necessary to undertake a model-based approach by selecting a hybrid plant model. The physical variables of the plant, and not their discrete abstraction by the controller, are what give the actual performance measures to be evaluated. The coupling of a hybrid plant model with a discrete controller model thus leads to a Hybrid Dynamic System (HDS) verification problem. We will now proceed by recalling the basic knowledge acquired and current limitations of HDS verification.

2.2 Hybrid systems verification.

While model design using hybrid automata is not exactly straightforward, their semantics is well-adapted to the analysis of HDS behavior. We will then assume that HDS verification is tantamount to

exploring the reachable state space of a hybrid automaton. When framed as such, the fundamental problem of HDS verification becomes one of computing the reachable state space of a hybrid automaton from an initial region. Not only is this computation prone to encountering a rapid combinatory explosion, but in the general case the problem is not-decidable, i.e. impossible to identify a computation algorithm with guaranteed convergence (Henzinger, *et al.*, 1995a). The linear hybrid automata (Alur, et al. 1995) constitute a category of hybrid automata that places verification within the domain of a decidable problem and for this reason, such automata have been selected for the purposes of our work. From an other hand, modular approaches are essential for modeling and verifying of industrial systems. This is why we felt that the HYTECH tool (Henzinger, *et al.*, 1995b) was adapted to our needs, by virtue of allowing for the verification of models obtained by composition of linear hybrid automata.
With the context and objectives of our research now established and both the category of hybrid automata and model-checker selected, we will turn our attention next to presenting our approach for verifying discrete control performance via the assessment of a pertinent example.

3. A CASE STUDY

3.1 Presentation of the production system.

The system we are targeting is one of assembling / disassembling bearings within gears; it comprises four workstations (see Figure 3a). The four workstations pertain respectively to the following functions: loading at the beginning of the line (Station 1); identification of the gear material and eventual presence of a bearing (Station 2); insertion or removal of a bearing in the gear (Station 3); and gear sorting, taking into account the material, in anticipation of their unloading (Station 4). This entire assembly line is controlled by a Programmable Logic Controller (PLC).

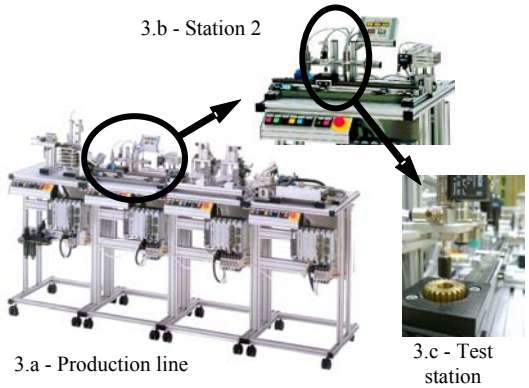

Fig. 3. Assembly/disassembly line.

Control of this assembly system is purely discrete; since this proves to be the most typical case for such systems, seeking more competitive costs leads to avoiding servo controls as much as possible in favor of logic controls. All continuous physical variables of the process being controlled (which for the most part consist of positions) are thus observed and controlled via their discrete abstractions. In all, 82 logical inputs and 50 logical outputs are managed by the PLC. The control model, written using Sequential Function Chart (SFC) syntax according to IEC 61131-3 standard has been verified with a timed model-checking technique (Bel Mokadem, *et al.*, 2005).
The objective of our study herein consists of verifying whether the quality of the positions obtained using such a discrete control is compatible with the precision required by the process. This concept of precision only takes on meaning with respect to the positioning of objects manipulated by either a displacement axis or a group of combined axes. It is thereby unnecessary to include the entire control model and the plant model in order to verify position quality, but rather just the subset that pertains to control of the axis (or group of axes) involved in the studied displacement. Our investigation will focus more heavily on the displacement axis of station 2 (Fig. 3b), with the only relevant information available for displacement control being given in Figure 4.

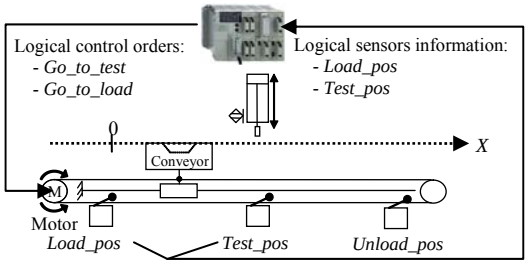

Fig. 4. Displacement axis of station 2.

The gear is initially located on the conveyor at the loading station (*Load_pos*). A motor rotation order in a clockwise direction (*Go_to_test*) implies the conveyor in displacement to the test position (*Test_pos*), a detailed view is provided in Figure 3c. For the test to be successfully conducted, stopping precision must be ≤ 1.5 mm. The conveyor position between these two discrete positions is denoted *X*. When the presence test of a bearing is completed, a subsequent motor rotation order in the clockwise direction positions the gear, where it gets unloaded (*Unload_pos*). A motor rotation in the counterclockwise direction then brings the conveyor into the loading position, where a new cycle can be launched. We are seeking to quantify the precision of gear positioning at the test station, enabled by this discrete control along the path *Load_pos* → *Test_pos*, so as to verify whether this precision is compatible with specification. Execution of this task will entail application of hybrid model-checking using the HYTECH tool (Henzinger, *et al.*, 1995b).

3.2 Modelling.

The model of the entire {controller + displacement axis} set, which allows this positioning quality verification, has been laid out in Figure 5.
The hybrid automaton is composed of two synchronized automata.

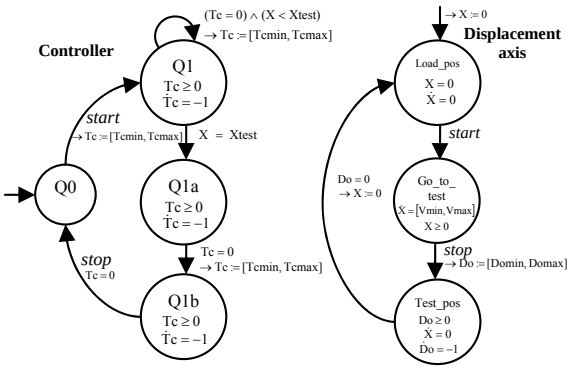

Fig. 5. Hybrid automata of the axis control.

The automaton that model electromechanical axis contains three discrete locations: two correspond with conveyor stopping positions at the extremities (*Load_pos* and *Test_pos*), and the other location corresponds with the movement phase between these extreme positions (*Go_to_test*). The return to loading position, which does not enter into the scope of the present study, is modeled by means of a timer with variable duration D_0. The two continuous state variables associated with this automaton are conveyor position X and timer duration D_0. Upon initialization, the conveyor is placed into the loading position and hence $X = 0$. Both of the stopping situations are associated with the function $dX/dt = 0$ (denoted $\dot{X} = 0$); the movement location is associated with a continuous dynamics of the form $\dot{X} = a$, with a representing a rational constant arbitrarily chosen within an interval: $[V_{min}, V_{max}]$. This arbitrary assignment of constant a enables abstracting the variability in conveyor displacement speed due to perturbations undergone by the displacement axis (mechanical friction, resistant torque on the motor axis, kinematic non-linearities, etc.). Two synchronization events with the automaton used for modeling controller behavior, called *start* and *stop* respectively, are associated with the transitions that correspond to the conveyor movement phase.

The automaton that models controller behavior is composed of four discrete locations and is initialized at location Q_0. The *start* synchronization event with the axis automaton shows that the motor starts up during a PLC cycle change. The monitor of this PLC is of the "periodic" type, indicating that execution of the control program is to take place according to a constant-duration cycle (see Figure 6), which the user is required to setup in PLC. In practice, this cycle duration undergoes slight variations (on the order of 1%), called "jitters". To translate these operations into the hybrid controller automaton, the duration of each PLC processing cycle (Tc) is assigned an arbitrarily-chosen value within an interval: $[Tc_{min}, Tc_{max}]$. Once location Q_1 has been entered, the automaton remains there until the axis has traveled the distance required to position it in alignment with the test station (at which point $X = Xtest$). Locations $Q1_a$ and $Q1_b$ serve to translate that this information is taken into account by the PLC during the subsequent input reading phase, followed by processing and then execution of the motor stop order (Figure 6). The *stop* synchronization event thereby triggers controller automaton re-initialization at the same time it activates the *Test_pos* situation of the displacement axis automaton.

Figure 6 shows the synchronization of the two automata presented in Figure 5 by the *stop* event as well as the evolution in continuous variables Tc and X. The axis positioning error at the level of the test station thus results from the time lag between the moment the axis reaches the *Xtest* position and the moment the PLC is able to react at this event in the input reading phase, with the consequence being that the axis stop order may be issued either within the same PLC cycle or one cycle later. The experimental campaign conducted using HYTECH has revealed this finding and the following discussion is intended to provide greater detail.

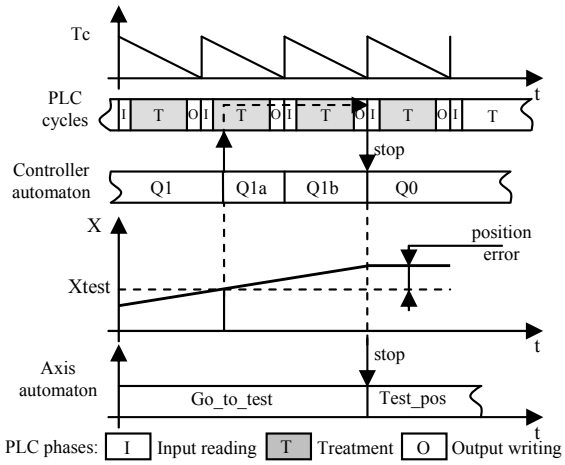

Fig. 6. Synchronous evolutions of automata.

3.3 Implementation and the experimental protocol.

All of our experiments have been carried out by constraining the variation of duration the conveyor's return to the loading position (D_0) over the interval [*3000ms, 3500ms*] (which thus yields *Domin = 3000ms* and *Domax = 3500ms*). Moreover, the abscissa (*Xtest*) of the conveyor at the test station is 80 mm.

Given that these data remain fixed, the HYTECH code obtained from the hybrid automaton shown in Figure 5 is a parametric code, with intervals [*Tcmin, Tcmax*] (for the cycle time to be adjusted on the PLC) and [*Vmin, Vmax*] (for the translation speed of the conveyor subjected to perturbations) constituting the two parameters. For each experiment, numerical values are ascribed to these two intervals. The model-checker then computes the region the automaton is capable of reaching from its initial region. We will initially derive the intersection of this reachable region with the discrete location *Test_pos* so as to build the accessibility domain of the axis position, which is then obtained by projection of this sub-region with respect to the variable X. The set of positions reached by the conveyor thus assumes the form of an interval on X. As an example, for *Vmin = Vmax = 200mm/s*, *Tcmin = 11ms* $^{-0.5\%}$ and *Tcmax = 11ms* $^{+0.5\%}$, the conveyor stopping position in front of the test station

is contained within the interval [*83.182mm, 84.018mm*], which represents a resultant position deviation of 0.836mm.

3.4 Position-stopping quality without perturbations.

For this initial study, the hypothesis has been adopted that the conveyor is not submitted to any perturbation. Its translation speed is thus presumed to be constant and yields: *Vmin = Vmax = 200 mm/s*. The changes in conveyor stopping position may then be observed as a function of the PLC cycle time value when such value varies by more or less than 0.5% around the period adjusted by the user (the so-called jitter phenomenon described in Section 3.2).
Figure 7 reveals that the positioning error evolves in accordance with two phenomena that we will now differentiate. The first corresponds to a systematic error (positive slope line passing through the average error interval value), which can be corrected by an adapted adjustment of sensor position *Test_pos*. The second phenomenon is homogeneous to a random error (the fixed interval deviation on the positions reached for a given *Tc*), whose general amplitude shape increases with PLC cycle time. It can nonetheless be remarked that for certain *Tc* values (e.g. *Tc = 9ms* or *Tc = 13.1ms*), the range of these "random" errors is narrower than that for proximate *Tc* values. The existence of these particular zones of smaller positioning error, while PLC cycle time is increasing, highlights a non-trivial temporal correlation phenomenon between the PLC cycle and occurrence of the test position arrival event depicted in Figure 8.

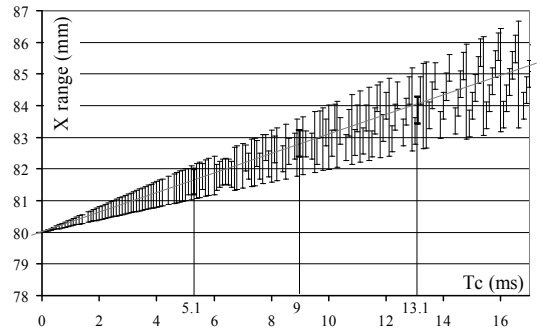

Fig. 7. Conveyor positioning error.

When the temporal detection region of the *X=Xtest* event has been included within a single PLC cycle, conveyor stopping will systematically be controlled during the subsequent cycle output assignment phase (see Figure 8a). In contrast, while the temporal detection region of the *X=Xtest* event straddles two PLC cycles, stopping will intervene during the output assignment phase of one of the two subsequent cycles (Figure 8b). The range of the stopping zone will thus increase.
This temporal dispersion in *stop* order execution is heavily correlated with the PLC cycle time jitter (Figure 8c). Though the ratio between the distance to be traversed and *Tc* is such that, despite the jitter on *Tc*, the number *N* of PLC cycles elapsed between the conveyor start order and detection of the test position (*X=Xtest*) is the same with each new conveyor displacement from *X=0* to *X=Xtest*, the conveyor stop order will always be issued at the end of the same PLC cycle and the error range will always solely depend on the jitter and cycle number *N* (this configuration will be referred to as "Scenario 1" in the following discussion). On the other hand, should the ratio between the distance traveled and *Tc* be such that a variable number of cycles is necessary for conveyor displacement, the *stop* order may then be issued at the end of the various PLC cycles (Figure 8c). The error range is thus merely a function of *Tc* (this configuration will be called "Scenario 2").

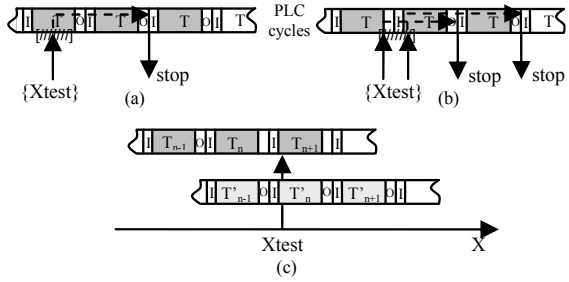

Fig. 8. Correlation between *Tc*, *Xtest* event occurrences and *stop* order.

Figure 9 clearly highlights this phenomenon. For Scenario 2, the stopping range is proportional to the cycle time value. Whereas for Scenario 1 positioning error is independent of cycle time (at least up to the limit of values selected for our experiment).
If specifications call for a maximum positioning error in *Test_pos* of 1.5mm, the PLC cycle period has to be set at a value of below 7.4ms. If the full PLC load (for execution of the overall assembly line program with its 82 inputs and 50 outputs) does not enable parameterizing such a low cycle time, it would then be necessary to target a Scenario 1 parameterization, e.g. 9.0ms or 13.1ms.

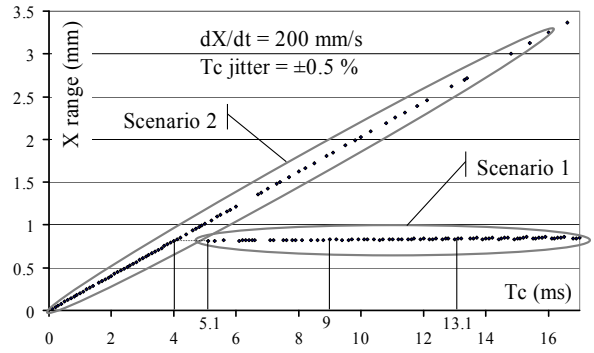

Fig. 9. Conveyor positioning error range.

3.5 Positioning robustness.

It has already been demonstrated that the PLC cycle time may be judiciously chosen in order to guarantee the quality of conveyor positioning under a hypothesis of no translation speed perturbations. Figure 10 presents the results from experiments conducted in order to evaluate the influence of such perturbations on positioning quality for a given cycle time *Tc=13.1ms* (which remains contaminated with a jitter of ± 0.5%). This curve shape is the same for *Tc = 9ms* or *Tc = 5.1ms*.

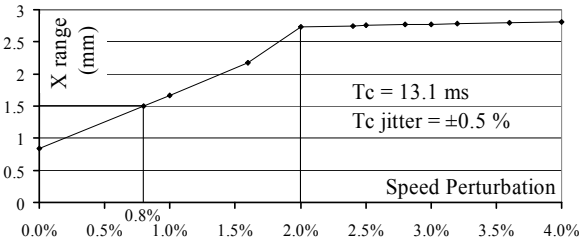

Fig. 10. Effect of conveyor speed perturbation.

Two distinct zones appear on this curve. Above 2% speed perturbation, the positioning error is basically constant; this corresponds to Scenario 2, for which the error range is directly correlated with *Tc* without any major influence on the speed-related perturbation. Below this 2% level, the error increases linearly with perturbations since the system here is behaving according to Scenario 1, which is sensitive to both the jitter on *Tc* and perturbations on the conveyor translation speed.
For a requirement that limits the positioning error to 1.5mm, it thus becomes necessary to ensure that speed perturbations remain less than 0.8% (in the case of a PLC adjustment to *Tc=13.1ms*). Should such not be the case, adopting a servo control in the conveyer position would need to be examined.

3.5 Assessment of the use of HYTECH.

The results provided above have necessitated several hundreds of model-checking experiments on the hybrid automaton shown in Figure 5 using the HYTECH tool, with different parameter sets being introduced for each experiment. The search for the reachable region of a model never took more than 5 minutes and, in most instances, 30 seconds were sufficient. Computation time did not therefore pose any problem for this study. The primary difficulty encountered had to do with the interruption of reachability computations due to exceeding the integer limit value (2^{64}) during the region computation step. Our case study did necessitate manipulating numerical values with quite varied scales. As an example, the conveyor position progressed over a total path length of 80mm, whereas the calculated stopping error at times amounted to just a few hundredths of a millimeter. For all of the distances manipulated, numerical values out to 4 significant digits thus had to be used. The same difficulty gets magnified when manipulating time since the duration of the conveyor position return lasts approximately *3s*, while a $\pm 0.5\%$ jitter on $Tc = 1ms$ equals $\pm 5\mu s$; seven significant digits therefore proved necessary. This numerical value coding is very detrimental when running HYTECH, which interrupts the computation of regions once the integer coding capacity has been exeeded, thereby making it impossible to draw any conclusion on the current verification. In order to avoid this predicament, the units for each model parameter set have been adjusted. For example, depending on the value of *Tc*, the time unit has been fixed at $1\mu s$, $10\mu s$ or $100\mu s$, which makes only *5%* of the verifications inconclusive.

4. CONCLUSION AND OUTLOOK

In this paper, we have focused on the verification of systems composed of a discrete controller coupled with a continuous (or partially-continuous) physical process, whose full extent of observed and controlled variables are logical abstractions of physical variables. For that, we employed a model-checking technique that couples the timed discrete model of the controller with a hybrid plant model. By relying upon a case study, we showed the possibility of verifying whether or not system performance is compatible with the specifications. By parameterizing the hybrid system model, it becomes possible for the model-checker to evaluate discrete control robustness to plant perturbations.
According to this approach, HYTECH has proven to be an efficient tool. It goes without saying that the size of the model being verified must remain compatible with the difficulties inherent in the well-known combinatory explosion phenomenon typical of model-checking. For this reason, it is necessary to limit performance verification to just the {control + plant} subsystem involved in deriving the evaluated variables. Determining this subsystem being targeted by performance verification along with building the hybrid model used as the abstraction are the current focus of a complementary research project.

REFERENCES

Alur R. et al. (1995). The algorithmic analysis of hybrid systems. Theoritical computer science, **Vol. 138**, p. 3-34.

Bel Mokadem H., Bérard B., Gourcuff V., Roussel J.-M., De Smet O. (2005). Verification of a timed multitask system with UPPAAL. *Proc. of 10th IEEE ETFA*, CDRom paper, 8 pages, September, Catania-Italy.

Clarke, E. M., Emerson, E. A. and A. P. Sistle, (1986). Automatic verification of finite state concurrent system using temporal logic. *ACM Trans. on Programming Languages and Systems* **Vol. 8**, n° 2, p. 244-263.

Emerson E.A. and Halpern J.Y. (1986). Sometimes and Not Never revisited : on branching versus linear time temporal logic. *Journal of the ACM*. **Vol. 33**, n° 1, p. 151-178.

Frey G. and Litz L. (2000). Formal method in PLC programming. *Proc. of IEEE SMC'2000*, CDRom paper, 6 pages, October 8-11, Nashville, USA.

Henzinger T.A., Kopke P.W., Puri A., Varaiya P. (1995a). Waht's decidable about hybrid automata ?. *Proc. of the 27th annual ACM Symposium of Theory of Computing*, p. 373-382.

Henzinger T.A., Ho P.-H., Wong-Toi H. (1995b). A user guide to HyTech. *Proc. of TACAS*, Lecture Notes in Computer Science 1019, Springer-Verlag, p. 41–71.

Mac Millan K.L. (1993). *Symbolic model checking*, Kluwer Academic.

Moon I. (1994). Modeling programmable logic controllers for logic verification. *IEEE Control Systems*, **Vol. 14**, n° 2, 1994, p. 53-59.

Rausch M. and Krogh H. (1998). Formal verification of PLC programs. *Proc. of ACC'98*, Philadelphia, USA.