A generic approach to build plant models for DES verification purposes

José J.B. Machado, Bruno Denis, Jean-Jacques Lesage

8th International Workshop on Discrete Event Systems, WODES'06, Ann Arbor, Michigan, (USA), pp 407-412, July 2006

A generic approach to build plant models for DES verification purposes

J. Machado, B. Denis, and J.-J. Lesage

***Abstract*—The modeling of plant behavior is often essential in the design, performance analysis or diagnosis of Discrete Event Systems (DES). Yet this task remains a difficult one for which little research has been devoted. In this paper, we propose a technique for building behavioral models specific to large-scale plants, in order to perform a formal verification of the controller by means of "model-checking". In this aim, we have opted to use a modular approach with an appropriate class of automata. To obtain the overall plant model, parallel evolutions of the elementary automata are to be coordinated by a sequencer that ensures consistency of these evolutions.**

I. Introduction

The behavior of a reactive Discrete Event System (DES) stems from the interaction between a controller and a plant (Fig. 1). Just the given expected overall DES behavior (i.e. behavior of the {controller + plant} closed-loop system, which in most cases is only informally described in the functional specifications) is insufficient therefore for the control engineer to design controller behavior if the event plant behavior is not known (or postulated). The automation paradigm set forth in [1] expresses this very aspect:

$$\textit{Process dynamics (Known)} \wedge \textit{Control rules (Unknown)} \supset \textit{Goals (Known)} \quad (1)$$

Although this paradigm had been proposed in a context other than DES, it still serves as a reference for a considerable body of methodological work in this field (e.g. [2]). In particular, it provides an entirely appropriate formulation for expressing the basis of controller synthesis (as regards the "supervisory control theory" introduced by Ramadge and Whonam [3]). This approach is in effect aimed at determining the most permissive control laws (the so-called *Unknown Control rules*), which satisfy the imposed specifications (*Known Goals*), with plant behavior being either known or postulated (*Known Process dynamics*). According to this approach, the relevance of the plant model is determinant for the synthesis of an optimal controller. The difficulties encountered in deriving a plant model adapted to this theory have been discussed in [4].

Equation (1) provides an expression for not only the synthesis, but also the set of interactions between plant and controllers during other phases of the DES life cycle:

- during the analysis phase, which entails ensuring (by either simulation or a formal "model-based" verification [5]) that the interaction between a controller model and a plant model encompasses the expected overall DES behavior;
- during the operation phase, in which model-based diagnosis techniques consist of identifying whether the observed DES behavior, resulting from the actual interaction between plant and controller, corresponds to a faulty behavior anticipated by the diagnoser, given that the diagnoser has itself been synthesized from a plant model / controller model couple [6].

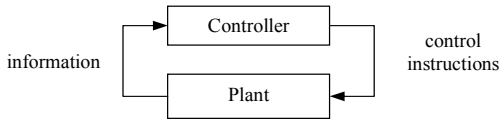

Fig. 1. A generic closed-loop DES

The construction of a good plant behavior model is thus vital to DES design, analysis and operation; yet paradoxically, only very little research has been devoted to this specific topic. In this paper, we propose a modular and systematic approach for building dependable and realistic plant models. Our scope incorporates the objective of introducing verification by means of a model-based model-checking, i.e. through exploring the state space reachable by a controller model coupled to a plant model. This work is structured as follows: Section II will review the contributions and limitations offered by the research we feel the most pertinent in terms of the behavioral modeling of plants. We will establish the need for an approach dedicated to plant model-building, to be presented in Section III and then applied to treat a significant example.

II. The plant model in DES automation

A. DES synthesis

The automated design of DES control laws offers a problem that has been challenging the scientific community for quite some time. Following a number of valuable contributions in this field (references [7] and [8] merit recognition), Ramadge and Wonham's supervisory control theory [3]

Manuscript received January 23, 2006.

J. Machado is with the Department of Mechanical Engineering, University of Minho, Azurém Campus, 4800-058 Guimarães PORTUGAL.

B. Denis is with the University Research Laboratory in Automated Production, École Normale Supérieure de Cachan, 61 av. du Président Wilson, 94235 Cachan FRANCE (corresponding author, phone: +33 1 47 40 24 13; fax: +33 1 47 40 22 20; e-mail: bruno.denis@lurpa.ens-cachan.fr).

J.-J. Lesage is with the University Research Laboratory in Automated Production, École Normale Supérieure de Cachan, 61 av. du Président Wilson, 94235 Cachan FRANCE (e-mail: jean-jacques.lesage@lurpa.ens-cachan.fr).

sparked keen interest in DES synthesis. Fifteen years after this fundamental work, the gap between the large number of theoretical contributions and the few successful industrial implementations still stands out [9]. Application of the supervisory control theory remains a wide open problem that has mobilized the scientific community of DES over the past decade ([10], [11], [12]). Several reasons underlie the difficulties encountered in applying this theory. From a more general standpoint, the size of models handled constitutes a key difficulty. Using decentralized, modular or hierarchical control structures or performing modeling with formalisms other than finite-state machines, such as Petri nets, provides for an attractive alternative. We feel however, in conjunction with [4], that another primary difficulty lies in the plant modeling set-up. Two reasons have led us to this assertion. First of all, the synthesis of a controller requires building a highly-detailed plant model whose events translate state changes in both the information delivered by sensors and the orders given to preactuators. The R&W synthesis method is based on language theory and then implies the use of a class of automaton poorly adapted to plant behavior modeling at this level of abstraction [13].

Moreover, the complexity of industrial systems necessitates the use of modular approaches for constructing the overall plant model [14]. To carry this step out once the elementary component automata have been established, the model of the entire plant is most often derived by means of composition, which leads to generating a model that contains many surplus states and transitions devoid of physical meaning.

B. DES verification

Formal verification techniques using model-checking [15] have been employed in the area of DES automation for the past twelve years. As part of a dependable controller design approach, the target verification system may comprise [16] either the controller on its own, assumed to be placed in an open-loop on the plant (non model-based verification) or the {controller + plant} set interacting within a closed-loop (model-based verification).

The formalisms and abstraction levels used to build a plant model for model-based verification purposes are quite diverse. They may consist of simple algebraic constraints for translating partial or local behavior, which are primarily intended to reduce the combinatorial explosion of the state space reachable by the controller [17]. On the other hand, they may consist of highly-detailed state models designed to improve the quality of proofs relative to safety properties [18]. The modular construction techniques applied to this plant model, as well as their inherent disadvantages, often turn out to be the same as those for the synthesis (see section II.A).

C. DES diagnosis

In our opinion, the topic of DES diagnosis has yielded the most significant recent contributions to plant modeling. Setting up a diagnoser actually presumes the preexistence of a plant model (that includes faulty behavior) with sufficient enough detail for the diagnosis to be pertinent. Nearly all techniques for plant behavior modeling therefore entail building a knowledge model in the form of digraphs, Binary Decision Diagrams or, more frequently, in the form of finite-state machines ([19], [20], [21]). For this purpose, behavioral models of elementary system components and their interactions have to be designed. Behavior of the full system is then most often obtained by composing the various component automata. As is the case with controller synthesis, this solution is difficult to put into practice for large-sized systems. Furthermore, an expert approach is necessary in order to remove the surplus states and transitions from the composition-generated automaton [6], which as opposed to the synthesis must not remain within the plant model.

D. Assessment

A plant model is mandatory in order to accomplish the majority of DES life cycle phases: synthesis, verification, diagnosis. Nonetheless, this model must be built using a formalism that relies upon a point of view and a level of abstraction compatible with its ultimate use. Under all circumstances, only modular approaches are compatible with the complexity of industrial DES. Alternatives to automata synchronous or asynchronous products must however be identified so as to avoid generating surplus states and transitions devoid of physical meaning.

We will now present our approach for the modular construction of plant models in the aim of performing a model-based controller verification.

III. Construction of a Plant Model in the Aim of Controller Verification

A. The controller verification procedure

In most model-based model-checking approaches [5], [16], [17], the verified model results from the composition of three models: the user program model, the control unit execution model, and the plant model.

For complex systems, the plant model must be derived from a library of generic models instanced and composed. Major contributions for modular plant modeling using standard “blocks” can be founded in [22], that defined Condition/Event Systems, or in [23] that defined Net Condition/Event Systems. However, our purpose is verification using model-checking, so we have chosen to use a “ready-to-check” and powerful class of automaton.

B. Selected class of automaton

We have selected an automaton class stemming from the one described in [21] for the UPPAAL model-checker and have elected to retain the same notations.

Automaton $\mathcal{A}$ is thus a triplet $\langle N, n_0, E\rangle$, where:

- N is a finite set of states;
- $n_0 \in N$ is the initial state;

- $E \subseteq N \times \tau \times \Sigma \times N$ is the set of transitions, with τ being the set of Boolean expressions defined on the set of logic variables $\mathcal{V}$, and Σ a partition of the set of assignments on $\mathcal{V}$.

An automaton network $\{\mathcal{A}_1,\ldots,\mathcal{A}_n\}$ is a set of automata whose Boolean expressions τ, associated with the transitions, are defined on the same set of logic variables $\mathcal{V}$. Within an automaton network, the evolutions are asynchronous and based on variable sharing. This implies that at each point in time, a single transition of a single automaton may be fired. Throughout the remainder of this paper, the syntax of Boolean expressions τ will replicate the syntax from C programming language ("&&" for the AND operator, "||" for the OR operator and "!" for the NO operator). In the discussion of the ensuing example, clocks and hand-shake synchronizations will not be employed.

C. Case study

The system chosen for this case study lies in the well-known category of "pick-and-place" systems (Fig. 2); its function is to take parts, fed by gravity into three feed chutes, for placement in a single unloading chute. Sensors pp1, pp2 and pp3 indicate the presence of a part in one of the feed chutes, while sensor pp0 signals the presence of a part in the unloading chute. The device that enables picking and placing a part is composed of a group of three pneumatic cylinders plus a vacuum suction cup system. The vertical cylinder (VC) places the suction cup in contact with a part. Longitudinal cylinders L1C and L2C are arranged in series to allow positioning the vertical cylinder VC in front of the four chutes (L2C stroke is twice as long as than L1C stroke). The four positions reached are thereby detected by position sensors s0, s1, s2 and s3. The depression in the suction cup is obtained by virtue of a venturi and detected by a vacuum sensor.

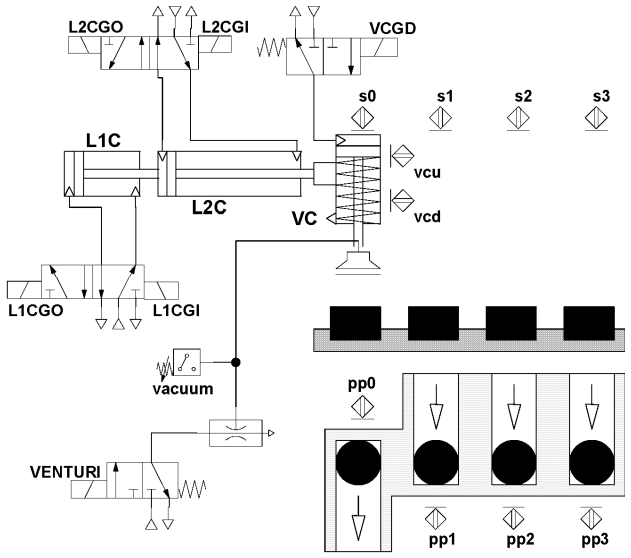

Fig. 2. Schematic view of the studied system

D. Construction of the plant-model

The plant model is built using generic models extracted from a library and then supplemented, if necessary, by adding in specific behavioral components. $\mathcal{A}_{Pi}$ will denote herein those automata that describe the behavior of the elementary plant components. Figure 3 illustrates the modularity of this approach and displays the result of generic model instantiation for the L1C, L2C two-cylinder subset and sensor s2.

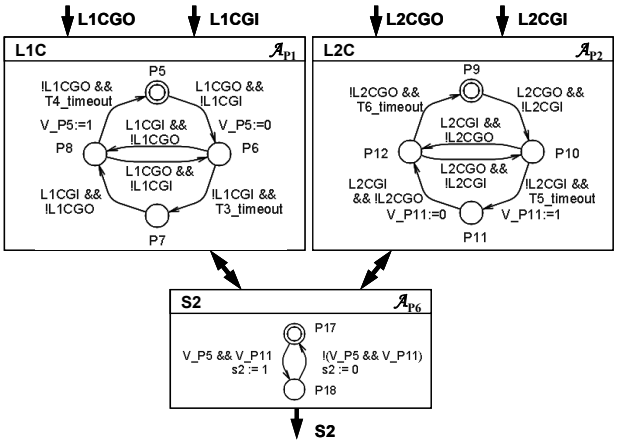

Fig. 3. Pant model obtained after the instantiation of generic automata from the library.

The behavior of each of the double-acting cylinders L1C and L2C with its 5/2-way double solenoid valve is built by instancing a single generic automaton. Automaton network $\{\mathcal{A}_{P1}, \mathcal{A}_{P2}, \mathcal{A}_{P6}\}$ models the behavior of this plant subset. Let us examine L1C automaton. It is composed of four states: P5 models the inward rod position, P6 models that the rod moves outwards, P7 models the outward rod position, P8 models that the rod moves inwards. In the states P6 and P8 the rod moves, and its movement will stop without order from controller. For that, a timer is associated to the transitions P6-P7 and P8-P5 to model the duration of the stroke deployment.

The transition $\langle P5, g, a, P6\rangle$, which will also be denoted $P5 \xrightarrow{g,a} P6$, features a guard "g" equal to "L1CGO && !L1CGI" and an action "a" equal to "V_P5:= 0". This transition may be fired when the guard is true, i.e. when both solenoid L1CGO (L1C Go Out) is activated and solenoid L1CGI (LIC Go In) is not activated. Its completion causes the variable V_P5 to be reset. This transition thus allows modeling the start of the cylinder deployment once the L1CGO order has been issued.

The transition $P6 \xrightarrow{!L1CGI \&\& T3\_timeout, -} P7$ models the end of the cylinder deployment movement by testing for the completion of the timer (T3_timeout), which represents the deployment duration. $\mathcal{A}_{Tj}$ is the notation used for automata describing the logic abstraction of timers. Following the instancing phase, an expert needs to intervene to coordinate the automata from a functional standpoint. As an illustration,

the sensor s2 automaton must be coordinated with the automata of cylinders L1C and L2C in order to determine whether or not cylinder VC lies above the second feed chute. The transition guards of automaton $\mathcal{A}_{P6}$ must therefore be able to test the activation of states $\mathcal{A}_{P1}$ and $\mathcal{A}_{P2}$. This feature is obtained using shared logic variables added to network automaton states. To avoid encumbering Fig. 3, only those variables necessary for modeling the activity of states P5 and P11 have been represented (i.e. V_P5 and V_P11). Automaton $\mathcal{A}_{P6}$ moves from state P17 to P18 as cylinder L1C is retracted (V_P5 equals 1) and as cylinder L2C is deployed (V_P11 equals 1). The action associated with this transition (s2:=1) indicates that the logic value emitted by the sensor is thereby set at 1. The shared logic variables included in order to mark each state are generated automatically according to the procedure described by algorithm 1.

Algorithm 1. Addition of a logical variable for each state of each $\mathcal{A}_{Pi}$ automaton and action update. These variables become state marking.

for each $\mathcal{A}$ **in** $\{\mathcal{A}_{P1},\ldots,\mathcal{A}_{Pn}\}$ **do**
  for each n **in** N **of** $\mathcal{A}$ **do**
    $\mathcal{V} := \mathcal{V} \cup \{ V\_n \}$
    for each $\langle n_s, g, a, n_d \rangle$ **in** E **of** $\mathcal{A}$ **do**
      if $n = n_s$ **then**
        $a := a \cup \{ V\_n := 0 \}$
      end if
      if $n = n_d$ **then**
        $a := a \cup \{ V\_n := 1 \}$
      end if
    end for
  end for
end for

E. Plant model coordination

Such a modular approach, that is necessary for complex systems modeling, requires examining closely the behavior of the assembled modules. Indeed, a plant evolution, following a controller order emission, results in a succession of evolutions of the state of the modules whose propagation must be coordinated (it is the role of the Plant Sequencer which is described in the following section). Only the *stable* states of an automata network, reached at the end of the controller order propagation, represent a relevant state of the plant. An example of such a succession of evolutions is given bellow.

Let us starting from the situation {*P8*, *P11*, *P17*} Figure 3 (cylinder L1C retracted and cylinder L2C deployed). The arrival of cylinder L1C at the end of its stroke (T4_timeout true) leads the system into situation {*P5*, *P11*, *P17*}. In this case, cylinder VC is positioned opposite chute 2, while sensor s2 is still emitting opposite information (s2 = 0). This unstable state will be leaved since the variable V_P5 becomes true. The final stable situation is {*P5*, *P11*, *P18*}. It is the relevant situation that corresponds to a real plant configuration.

In short, in each module a stable state can be leaved: either because a new controller order appears or because a no null duration action of the plant is completed (such as for example the end of the L1C rod inward move which is modeled using the timeout of timer T4). The succession of unstable states before reaching a stable one is due to the firing of transitions (as P17→P18) sensitive to the change of state of other modules.

In order to manage this search of stability we have added to the plant model an automatically-generated automaton, called "plant sequencer" (denoted $\mathcal{A}_{PS}$), which also permits to identify that all consequences of a variation in controller orders have been fully propagated. In a same manner, it is necessary to manage interactions between the controller model and the plant model evolutions. To do that, we have added a "general sequencer" (denoted $\mathcal{A}_{GS}$), no more detailed in this paper.

Figure 4 presents the whole model for this case study and is structured as follows:

$$\{\mathcal{A}_{GS},\ \underbrace{\mathcal{A}_{CS}, \mathcal{A}_{CP},}_{\text{Controller model}}\ \underbrace{\mathcal{A}_{PS}, \mathcal{A}_{P1},\ldots, \mathcal{A}_{P12}, \mathcal{A}_{T1},\ldots, \mathcal{A}_{T13}}_{\text{Plant model}}\}$$

↑ General sequencer

The plant sequencer automaton is composed of four states:

- PIR (Plant Inputs Reading) during the activity of which the orders emitted by the controller are taken into account and held;
- PTR (Plant TReatment) that allows the plant model evolution until reaching a stable state;
- POU (Plant Outputs Updating) during the activity of which the change of sensor values are transmitted to the controller;
- when PTE (Plant TEst) is active, the state of the plant model corresponds to the real configuration of the plant. The verification can be performed.

To ensure coordination with $\mathcal{A}_{PS}$, automata $\mathcal{A}_{Pi}$ must be enhanced such that:

- when $\mathcal{A}_{PS}$ fires transition PIR→PTR, all automata $\mathcal{A}_{Pi}$ must asynchronously fire one and only one transition;
- once all automata $\mathcal{A}_{Pi}$ have fired a transition, $\mathcal{A}_{PS}$ fires either PTR→PTR if at least one $\mathcal{A}_{Pi}$ has change of state or PTR→POU otherwise (Fig. 5).

This automaton enhancement is entirely automated in compliance with the procedure described in algorithm 2.

The overall DES model also includes a general sequencer A_{GS} that ensures the evolution alternatively of the controller model and plant model. The properties to be verified are assessed solely at the end of each completed evolution of one of the two models.

F. Discussion

The framework of this paper does not provide ample space to present the entire model-checking process of the case study. The key general features of the verification process adopted, condensed into just a few words, are as

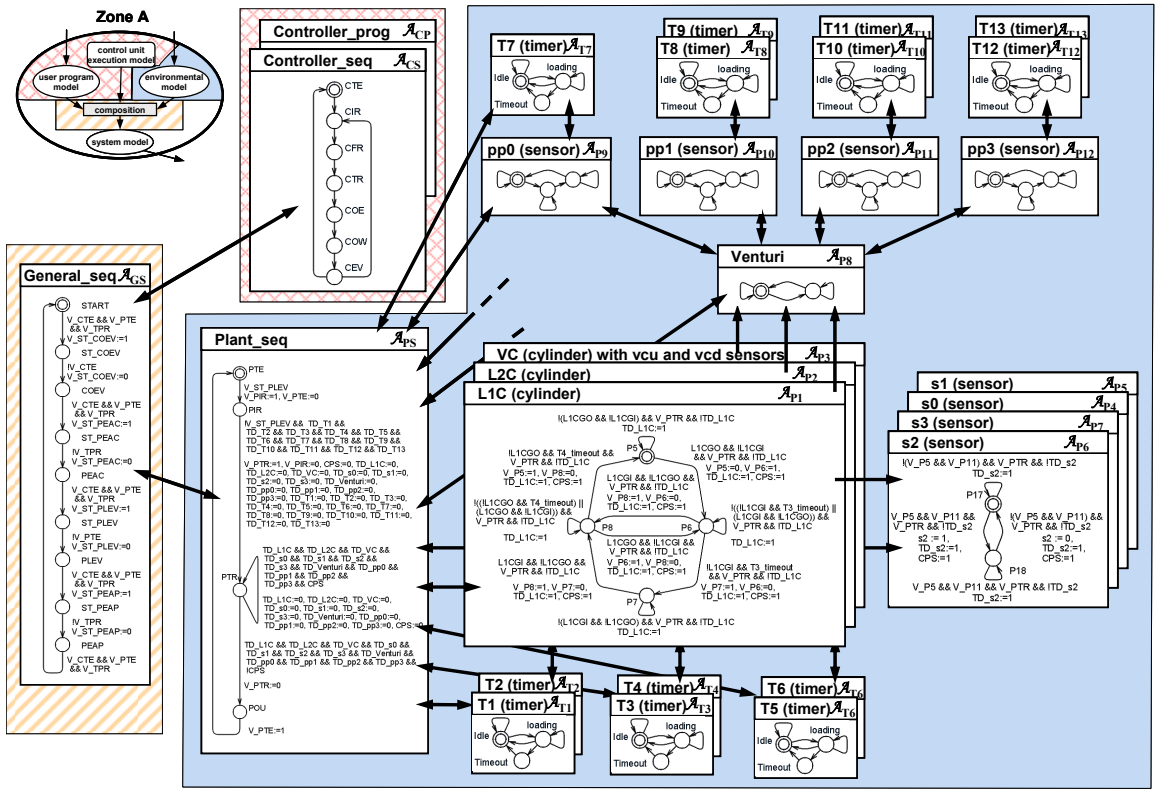

Fig. 4. Whole model ready for performing the verification.

follow:

- NuSMV has been chosen as model-checker due to its performance for timeless verification;
- the computer which perform the verification hosts a CPU P4 at 2.54GHz and 2GB RAM ;
- a set of 42 liveness and safety properties are to be proved.

Obtained from two experiments focusing on the same properties but checking two different automaton networks, the results underline the advantage of our approach.

- *experiment 1* (pure automata product approach, without plant sequencer neither general sequencer). The checked model is the network $\{\mathcal{A}_{CS}, \mathcal{A}_{CP}, \mathcal{A}_{P1},\ldots, \mathcal{A}_{P12}, \mathcal{A}_{T1},\ldots, \mathcal{A}_{T13}\}$. The $\mathcal{A}_{Pi}$ automata do not have been enhanced as shown Fig 3. The NuSMV input code is 788 lines long (without comments neither empty lines). After 3h30min of computation, model-checker use 3.2GB memory and make the computer swap heavily. No answer can be obtained. The too great number of surplus states, due to automaton product, avoid the checker to complete proves.
- *experiment 2* (our automata coordination approach with plant sequencer and general sequencer). The checked model is the network $\{\mathcal{A}_{GS}, \mathcal{A}_{CS}, \mathcal{A}_{CP}, \mathcal{A}_{PS}, \mathcal{A}_{P1},\ldots, \mathcal{A}_{P12}, \mathcal{A}_{T1},\ldots, \mathcal{A}_{T13}\}$. The $\mathcal{A}_{Pi}$ automata have been enhanced by applying the algorithm 2. The NuSMV input code is 1372 lines long. All properties have been successfully proved within 8h10min.

To evaluate consequences of surplus states generated by the automata product on the verification process we check the following property "when VC is in front of chute 2 then sensor s2 always detect it".

- *experiment 1* with automata product approach the property becomes in Computation Tree Logic $\square((V\_P5 \wedge V\_P11) \rightarrow s2)$ and it is checked as false;
- *experiment 2* with automata coordination approach, to prevent that the checking occurs for unstable states, the property becomes $\square((V\_PTE \wedge V\_P5 \wedge V\_P11) \rightarrow s2)$ where V_PTE is true when the plant model has reached a stable state. This property is checked as true.

Contrary results for the proof of the same property in these two experiments were expected. Indeed, in experiment 1 the model-checker declares the property false because it

Algorithm 2. Addition of further information into guards and addition of new transitions to take into account the coordination with plant sequencer automaton ($\mathcal{A}_{PS}$).

$\mathcal{V} := \mathcal{V} \cup \{ CPS \}$
for each $\mathcal{A}$ **in** $\{\mathcal{A}_{P1},\ldots,\mathcal{A}_{Pn}\}$ **do**
  $\mathcal{V} := \mathcal{V} \cup \{ TD\_\mathcal{A} \}$
  for each $\langle n_s, g, a, n_d\rangle$ **in** E of $\mathcal{A}$ **do**
    if $n_s \neq n_d$ **then**
      $g := g$ && V_PTR && !$TD\_\mathcal{A}$
      $a := a \cup \{ TD\_\mathcal{A} := 1 \} \cup \{ CPS := 1 \}$
    end if
  end for
  for each n **in** N of $\mathcal{A}$ **do**
    $a := \{ TD\_\mathcal{A} := 1 \}$, $g := 0$
    for each $\langle n_{s1}, g_1, a_1, n_{d1}\rangle$ **in** E of $\mathcal{A}$ **do**
      if $n_{s1} = n$ **then**
        $g := g \,||\, g_1$
      end if
    end for
    $g :=$!g && V_PTR && !$TD\_\mathcal{A}$
    $E := E \cup \{ \langle n, g, a, n\rangle \}$
  end for
end for

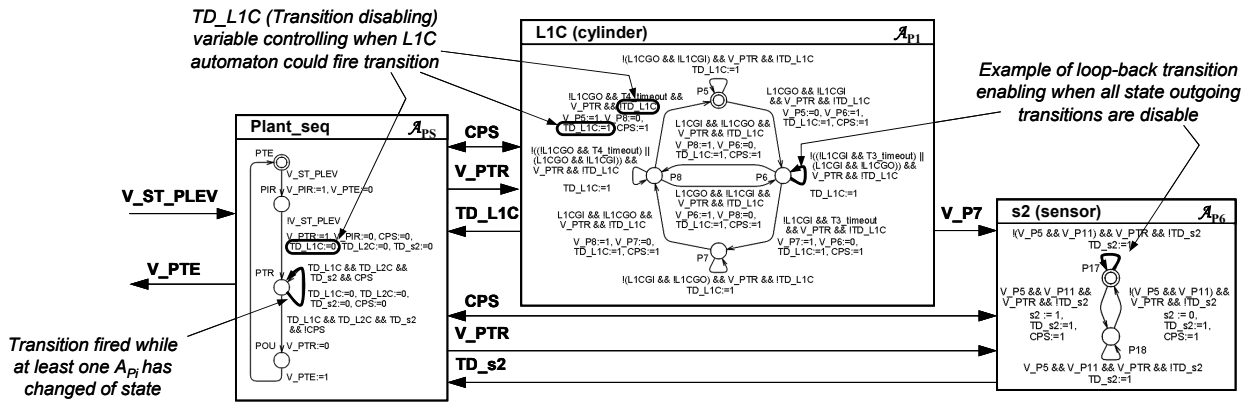

Fig. 5. Coordination between plant sequencer automaton and two plant component automata. $\mathcal{A}$p1 and $\mathcal{A}$p6 automata have been enhanced applying algorithm 2.

identifies an unstable situation in which L1C is in inward position, L2C is in outward position and the s2 sensor is not yet set to 1. This no significant situation is avoided using our coordination approach in experiment2.

IV. Conclusion

In this paper, we have considered the problem of building plant models for large complex DES with the aim of controller verification. For these systems, a modular approach of plant model building by assembly of generic component models is necessary. We showed that this assembly generates difficulties of controlling the propagation of the behaviors between modules. In our opinion, all modular approaches which aim to preserve a real encapsulation of the component models encounter this problem independently of the modeling formalism or of the granularity of the models. We thus proposed a plant sequencer devoted to coordinate the asynchronous behaviors of the modules until obtaining a stable situation of the whole plant model.

References

[1] A. Fusaoka, H. Seki, and K. Takahashi, "A description and reasoning of plant controllers in temporal logic", in *Proc. 8th Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany, 1983, pp. 405-408.

[2] G. Morel, J.-F. Petin, P. Lamboley, "Formal specification for manufacturing systems automation", in *CDROM Proc. IFAC Int. Conf. on Information Control Problems in Manufacturing*, Vienna, Austria, September 2001, 6 pages.

[3] P.J. Ramadge, and W.M. Wonham, "The control of discrete event systems", IEEE special issue on Discrete Event Dynamic Systems, vol. 77, n° 1, January 1989, pp. 81-98.

[4] J.-M. Roussel, and A. Giua, "Designing dependable logic controllers using the supervisory control theory", in *CDROM Preprints 16th IFAC World Congress,* Praha, Czech Republic, July 2005, paper n° 04427, 6 pages.

[5] T. Merkte and T. Menzel, "Methods and tools to the verification of safety-related control software", in *Proc. IEEE int. conf. on Systems Man and Cybernetics*, Nashville, USA, October 2000, pp. 2455-2457.

[6] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models", *IEEE Trans. C S T*, vol. 4, n° 2, pp. 105-124, March 1996.

[7] D. Crockett D, A. Desrochers, F. Dicesare, and T. Ward, "Implementation of a Petri net controller for a machining workstation", in *Proc. IEEE conf. Robotics and Automation*, Raleigh, North Carolina, USA, April 1987, pp. 1861-1867.

[8] M.C. Zhou, F. Dicesare, and D. Rudolph, "Design and Implementation of a Petri net based supervisor for a flexible manufacturing System", *Automatica*, vol. 28, n° 6, pp. 1199-1208, 1992.

[9] M. Fabian and A. Hellgren, "PLC-based implementation of supervisory control for discrete event systems", in *Proc. 37th IEEE Conf. on Decision and Control*, Tampa, USA, December 1998, pp. 3305-3310.

[10] S. Balemi, G.J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G.F. Franklin, "Supervisory control of a rapid thermal multiprocessor", *IEEE Trans. Automatic Control*, vol. 38, n° 7, pp.1040-1059, 1993.

[11] L.E. Holloway, A. Callahan, J. O'Rear, and X. Guan, "Spectool: automated synthesis of control code for discrete event controllers", in *Proc. 5th Int. Workshop on D E S*, Ghent, Aug. 2000, pp. 383-389.

[12] M.H. De Queiroz and J.E.R. Cury, "Synthesis and implementation of local modular supervisory control for a manufacturing cell", in *Proc. 6th Int. Workshop on D E S*, Zaragoza, Spain, Oct. 2002, pp. 377-382.

[13] A. Philippot, A. Tajer, F. Gellot, and V. Carré-Ménétrier, "On line synthesis approach based on a structured plant modelling", in *Proc. 7th Int. Workshop on D E S*, Reims, France, Sep. 2004, pp. 397-402.

[14] D. Gouyon, J.-F. Petin, and G. Morel, "Control synthesis for product-driven automation", in *Proc. 7th Int. Workshop on Discrete Event Systems*, Reims, France, September 2004, pp. 19-24.

[15] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen, *"Systems and software verification: model-Checking techniques and tools"*, Springer, 1999, 190 pages.

[16] G. Frey and L. Litz, "Formal method in PLC programming", in *CDROM Proc. IEEE int. conf. on Systems Man and Cybernetics*, Nashville, Tennessee, USA, October 2000, pp. 2431-2436.

[17] G. Canet, S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen, "Towards the automatic verification of PLC programs written in instruction list", in *Proc. IEEE int. conf. on Systems Man and Cybernetics*, Nashville, Tennessee, USA, Oct. 2000, pp. 2449-2454.

[18] J. Machado, B. Denis, J.-J. Lesage, J.-M. Faure, J.C. Ferreira da Silva, "Increasing the efficiency of PLC program verification using a plant model", in *CDROM Proc.6th Int. Conf. on Industrial Engineering and Production Management*, Porto, Portugal, May 2003, 10 pages.

[19] J. Lunze, J. Schröder, and P. Supavatanakul, "Diagnosis of Discrete-Event Systems: the method and example", in *Proc. 12th int. workshop on principles of diagnosis*, Sansicario, Italy, Mar. 2001, pp. 111-118.

[20] J. Sztipaovits and A. Mistra, "Diagnosis of Dicrete Event Systems using ordered binary decision diagram", in *Proc. 7th International workshop on principles of diagnosis*, Val Morin, Canada, 1996.

[21] J. Bengtsson and W. Yi, "Timed Automata: Semantics, Algorithms and Tools," *LNCS 3098*, Springer-Verlag, 2004.

[22] R.S.Sreenivas and B.H.Krogh On condition/event systems with discrete state realizations Discrete Event Dynamic Systems: Theory and Applications, 2(1):209-236,1991

[23] M. Rausch and H.-M. Hanisch. Net condition/event systems with multiple condition outputs. Symposium on Emerging Technologies and Factory Automation, Paris, France, pp 592-600, Oct. 1995.