Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Proposition of completeness property to perform the plant modelling for manufacturing applications

Rohee B.*, Carre-Menetrier V.*, Riera B.*

*UFR Sciences Exactes et Naturelles,
CReSTIC EA 3804, Université de Reims Champagne Ardenne,
Moulin de la Housse, BP 1039 – 51687 Reims
(e-mail: benoit.rohee ; veronique.carre ; bernard.riera @univ-reims.fr)

Abstract: Research dealing with control behavior determination often needs plant modelling to use formal methods. However, plant models are difficult to obtain because a manufacturing application is composed of a huge set of components with numerous interactions each other. The goal of this paper is to propose and demonstrate a property to improve the designer's confidence in his plant model. The property is used a posteriori from plant model design. The completeness property tests whether a plant model is able to respond to specification solicitations. The application on several examples shows the usefulness of the verification in the context of independence of the plant model according to control behavior. Property can be used before control synthesis application design to determine if a plant model is able to respond to all control changes.

1. INTRODUCTION

Lots of research overhangs concern the control behavior of automated systems. The IEC 61 508 norm, entitled « functional safety of electrical / electronic / programmable electronic safety-related-systems » advocates formal method uses to develop most critical parts. Research thematic concerns more particularly design and realization of operational control systems (Rausch and Krogh, 1998), (Malik, 2002). Control synthesis in particular belongs to formal methods of safety control elaboration (Roussel and Giua, 2005). The principle is to restrict the plant model behavior with some specifications. However, according to application finalities, control synthesis can have various forms and, as a consequence, the modelling tools are adapted to the application context. In particular, the Ramadge and Wonham research (Ramadge and Woham, 1987) brought a mathematical framework from a theoretical point of view. Nevertheless, the applications that use this framework are often based on scheduling and execution optimization of elementary functions executed by the system (Liu and Darabi, 2002). (Balemi, 1992) proposed an interpretation of Ramadge and Wonham framework more adapted to industrial systems of control command. The plant evolutions are no longer perceived as elementary functions initiated by the controllable events. The uncontrollable events are not associated to the spontaneous evolutions of the plant. Each evolution of output signals of the control part is associated to one controllable event in the same way as each evolution of input signals is associated to an uncontrollable events. The control model to obtain corresponds to determine the output signals according to input signals of the control part (Zaytoon and Carre-Menetrier, 2001). Consequently, each event has to be associated to a signal change between the control part and the plant and unobservable events do not exist. The plant models that are used in this context are very different from those which are used in scheduling under constraints. The requirement that the plant model has to satisfy includes the relations between the signal changes (Philippot et al., 2005). So, this point of view requires taking into account these requirements and a preliminary verification of some plant model properties can be executed before using the plant model to generate the control part implementation model. In the context of this paper, the verification that a plant model is able to respond to control part solicitations in a particular context is the object of a more particular attention (Rohee et al., 2006). The proposed property – the core subject of this paper - is to test the plant model opposite a completeness specification and to apply it to verify if a plant model is able to assimilate all the controllable events that can occur.

The paper is organized as follows. In section 2, the context of plant modelling and the modelling tool is briefly reviewed. The mathematical compositions used for the property are described in section 3. Section 4 presents the property and the demonstration. A fictive example illustrates the demonstration. Several common manufacturing application examples of plant models are described and confronted to some completeness specifications in section 5. The result of the property is discussed and assumptions about the control part or the plant behavior can appear in the context of the property use. The perspectives of future works are discussed at the end of the paper.

2. CONTEXT

A discrete event system is a system that can take several situations during the functioning. The finite states machines provide a formal and mathematical representation tool. (Cassandra and Lafortune, 1999) sums up the general formalism and the operations on generic tools. For implementing, each situation is associated to a state. The system assumes that it can only occupy a finite number of states whether the moment. At each time, only one state of a

finite state machine is occupied by the system. Transitions represent the changes of states according to the events occurrences associated to the system evolutions. An event occurs instantaneously along the time. The process behavior is given by the event sequences that can occur and the states that the process occupies. A string represents such sequence of events.

A finite state machine G is described by a 4-tuple $(Q_G, \Sigma_G, Q_{0G}, E_G)$ where

Q_G : is the finite set of states, the state-space

Σ_G : is the finite set of events, the alphabet

$Q_{0G} \subseteq Q_G$: is the initial state

$E_G \subseteq Q_G \times \Sigma_G \times Q_G$: is the set of edges of G

An edge $(q_1, \sigma, q_2) \in E_P$ means there is an edge from state q_1 to state q_2 when event σ occurs. A deterministic machine involves there is no ambiguity of crossed edge when an event occurs from a state. By extension of the edges set for a deterministic state machine, the transition function determines the reached state after occurrence of a sequence of events also called a string:

$$\delta_G : Q_G \times \Sigma_G^* \to Q_G$$
$$\delta_G(q_1, \sigma) = q_2 \Leftrightarrow (q_1, \sigma, q_2) \in E_G$$
$$\delta_G(q, \varepsilon) = q \qquad q \in Q_G$$
$$\delta_G(q, s\sigma) = \delta_G(\delta_G(q, s), \sigma) \quad \sigma \in \Sigma_G, s \in \Sigma_G^*$$

where ε denotes the empty string.

From the edges set, the active event function determines the set of all events associated to a defined transition in a current state.

$$\Gamma_G : Q_G \to 2^{\Sigma_G}$$
$$\forall \sigma \in \Sigma_G, \forall q \in Q_G$$
$$\sigma \in \Gamma_G(q) \Leftrightarrow \exists q' \in Q_G \,|\, (q, \sigma, q') \in E_G$$

The completeness property proposes to test that a plant model is able to assimilate the completeness specification solicitations. The interest of the completeness property is to test a partial plant model usually used in decentralized or modular approaches like in (Gouyon et al., 2004) with a local completeness specification. The plant model represents the system functioning independently to the control behavior while the completeness specification describes the event sequences that the plant model has to assimilate.

The property formally tests if all the specification event sequences are included in the plant model restricted to the common events.

From a more concrete point of view, the completeness verification can be useful for the applications that use the Balemi interpretation. In order for a plant model to represent the complete functioning (including as well as the admissible and non-admissible behavior) of the system, the plant model has to accept all the possible control solicitations. A low level plant model used in the PLC applications can be tested to verify if it includes all the controllable events that can occur whatever are the uncontrollable event occurrences. The completeness specification can exclude the uncontrollable events corresponding to the input changes to test if the plant model authorizes the output changes whatever the input evolutions. In this context, a complete plant model contains all the output changes. Then, the control synthesis operation can extract from such a global plant model a model that respects security and the user's requirements.

In this paper, the property is formally demonstrated and then it is applied to test the completeness of a plant model according to a completeness specification of the controllable event acceptance.

3. OPERATIONS ON FINITE STATE MACHINE

Let be $P = (Q_P, \Sigma_P, q_{0P}, E_P)$ a finite state machine that represents empirically the behavior of a plant model. We propose to check that the plant model does not lack events: some applications need a modular plant model to accept an event whatever the current state. The completeness has to be described through a specification. The completeness specification is expressed by a finite state machine $C = (Q_C, \Sigma_C, q_{0C}, E_C)$. We assume that $\Sigma_C \cap \Sigma_P \neq \varnothing$ because the specification and plant models share events. The proposed property is quite tolerant in its use because it can check a partial plant model with a local specification provided that there are mutual events. The property uses two compositions. The definition of compositions can differ according to the authors following the habits or context.

3.1. The accessible part operation

The accessible part operation of a finite state machine, denoted *Ac(G)*, consists in deleting the states and the associated transitions that are not accessible or reachable from the initial state. The language generated by this finite state machine is still the same as the language of *G*. Formally,

$$Ac(G) = (Q_{Ac(G)}, \Sigma_G, Q_{0G}, E_{Ac(G)})$$

$Q_{Ac(G)}$ *is the set of accessible states in G.*

$$Q_{Ac(G)} = \{\forall q \in Q_{Ac(G)}, \exists s \in \Sigma_G^* \,|\, q = \delta_G(Q_{0G}, s)\}$$

The compositions are often used to express the result of the interactions between several finite state machines. Several interaction forms exist depending on the constraints of synchronization. Two compositions are defined to determine the interactions between two finite state machines. The finite state machine obtained is restricted to the reachable states from initial states.

To illustrate the use of these operators, an example is proposed with two finite states machines *P* and *C* on the Figure 1.

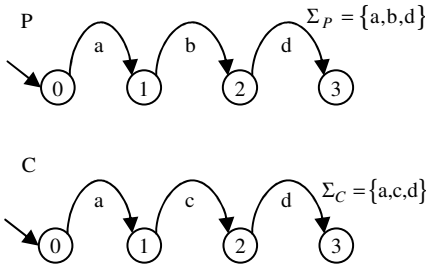


Fig. 1. Illustrated example

3.2. The parallel composition

The goal of the parallel composition is to synchronize two finite state machines by exploring the state space successively reached. The events that are common to the two finite state machine alphabets are authorized to occur only if the two finite state machines tolerate it from the current state. The events that are not common to the alphabets of the two finite state machines can freely occur.

In the proposed example, event *a* is authorized from the initial states of *P* and *C* because the two finite state machines simultaneously authorize it. Events *b* and *c* are then authorized because they are not included in the alphabets of the two finite state machines. Event *d* is common to the alphabets of the two finite state machines. The event *d* can only occur when the two machines authorize it. So, this can only occurs when the *states 2* of *P* and *C* are simultaneously reached. The finite state machine obtained is shown on Figure 2.

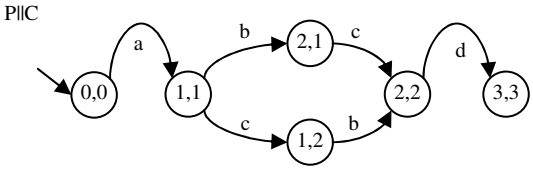


Fig. 2. The synchronous composition illustrated result

Formalization:

For two finite state automata $P=(Q_P,\Sigma_P,Q_{0P},E_P)$ and $C=(Q_C,\Sigma_C,Q_{0C},E_C)$, the parallel composition, denoted $P\|C$, is defined by:

$$P\|C = Ac\left(Q_P\times Q_C,\Sigma_P\cup\Sigma_C,Q_{0P}\times Q_{0C},E_{P\|C}\right)$$

where

$$\forall(p,\sigma,p')\in E_P,\ \forall(q,\rho,q')\in E_C \begin{cases} ((p,q),\sigma,(p',q'))\in E_{P\|C} \\ \qquad /\,\sigma=\rho\in\Gamma_P(p)\cap\Gamma_C(q) \\ ((p,q),\sigma,(p',q))\in E_{P\|C} \\ \qquad /\,\sigma\in\Gamma_P(p)\ \textit{and}\ \sigma\notin\Sigma_C \\ ((p,q),\rho,(p,q'))\in E_{P\|C} \\ \qquad /\,\rho\in\Gamma_C(q)\ \textit{and}\ \rho\notin\Sigma_P \end{cases}$$

The accessible part operation is applied to the obtained finite state machine to reduce the set of states that could potentially be reached.

The synchronous composition only synchronizes the mutual events from the alphabets of the two machines. The mutual events can only occur when both machines accept to execute it simultaneously. The non-mutual events can be executed when a machine authorizes its execution. The transition function of the synchronous composition is:

$$\delta_{P\|C}(\langle p,q\rangle,\sigma)=\begin{cases}\delta_P(p,\sigma)\times\delta_C(q,\sigma) & \sigma\in\Gamma_P(p)\cap\Gamma_C(q)\\ \delta_P(p,\sigma)\times q & \sigma\in\Gamma_P(p)-\Sigma_C\\ p\times\delta_C(q,\sigma) & \sigma\in\Gamma_C(q)-\Sigma_P\end{cases}$$

3.3. The broadcast composition

The broadcast composition is also based on the state space exploration. The composition synchronizes the mutual events from the alphabets of the two machines. Mutual events are also synchronized when both machines accept to execute it simultaneously. Otherwise, all the non mutual events can also occur when a machine authorizes their execution.

The broadcast composition differs from the parallel composition when an event common to the alphabets of the two finite state machines occur and only one machine authorizes this event. This composition is less used in the literature than the parallel composition because it is not used in the control synthesis operation of Ramadge and Wonham.

In the case of the previous example, event *a* is authorized because it is authorized from the initial state of *P* and *C*. Then, events *b* and *c* are authorized again in the example because they are not common to the alphabets of the two finite state machines. Event *d* is common to the alphabets of the two machines. Event *d* can occur even if only one machine authorizes it. The event can occur whatever the other machine current position. The obtained model for the broadcast composition is presented on Figure 3:

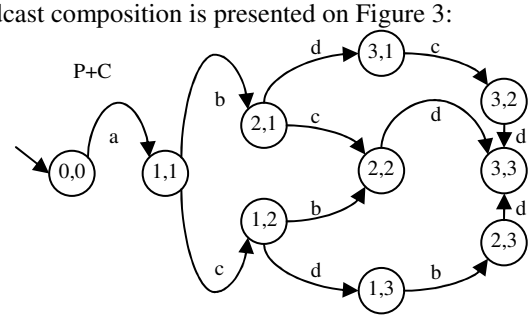


Fig. 3. The broadcast composition illustrated result

Formalization:

For two finite state automata $P=(Q_P,\Sigma_P,Q_{0P},E_P)$ and $C=(Q_C,\Sigma_C,Q_{0C},E_C)$, the broadcast composition, denoted $P+C$, is defined by:

$$P+C = Ac\left(Q_P\times Q_C,\Sigma_P\cup\Sigma_C,Q_{0P}\times Q_{0C},E_{P+C}\right)$$

where

$$\forall(p,\sigma,p')\in E_P,\ \forall(q,\rho,q')\in E_C \begin{cases} ((p,q),\sigma,(p',q'))\in E_{P+C} \\ \qquad /\,\sigma=\rho\in\Gamma_P(p)\cap\Gamma_C(q) \\ ((p,q),\sigma,(p',q))\in E_{P+C} \\ \qquad /\,\sigma\in\Gamma_P(p)\ \textit{and}\ \sigma\notin\Gamma_C(q) \\ ((p,q),\rho,(p,q'))\in E_{P+C} \\ \qquad /\,\rho\in\Gamma_C(q)\ \textit{and}\ \rho\notin\Gamma_P(p) \end{cases}$$

The accessible part operation is applied to the obtained finite state machine to reduce the set of states that could potentially be reached.

The transition function of the broadcast composition is:

$$\delta_{P+C}(\langle p,q\rangle,\sigma)=\begin{cases}\delta_P(p,\sigma)\times\delta_C(q,\sigma) & \sigma\in\Gamma_P(p)\cap\Gamma_C(q)\\ \delta_P(p,\sigma)\times q & \sigma\in\Gamma_P(p)-\Gamma_C(q)\\ p\times\delta_C(q,\sigma) & \sigma\in\Gamma_C(p)-\Gamma_P(p)\end{cases}$$

The Supremica tool includes these compositions (Akesson et al., 2003) and can be used to test the property.

4. PROPERTY TERMS AND DEMONSTRATING

4.1. The completeness property

Let a plant model represented by a finite state machine *P* and a specification that tests the completeness of the plant model represented by a finite state machine *C*. The completeness property can be represented as follows - the acceptation of an event authorized by the completeness specification implies this event is authorized by the plant model if this event is common to the alphabets of the two machines whatever the reached states of *P* and *C*.

$$\forall s\sigma \in (\Sigma_P \cup \Sigma_C)^*, p = \delta(Q_{0P}, pj(s,\Sigma_P)),$$
$$q = \delta(Q_{0C}, pj(s,\Sigma_C)),$$
$$\sigma \in \Gamma_C(q) \Rightarrow \sigma \in \Gamma_P(p)$$

$pj(s,\Sigma_P)$ *represents the natural projection of the string s over* Σ_P *alphabet.*

We demonstrate this condition is verified if and only if property *(1)* is verified:

$$((P+C)\|C) \equiv (P\|C) \qquad (1)$$

4.2. Property demonstration

Let the two finite state machines $P = (Q_P, \Sigma_P, Q_{0P}, E_P)$ and $C = (Q_C, \Sigma_C, Q_{0C}, E_C)$. We propose to demonstrate that the relation:

$\forall s\sigma \in (\Sigma_P \cup \Sigma_C)^*, p = \delta(Q_{0P}, s), q = \delta(Q_{0C}, s),$
$\sigma \in \Gamma_C(q) \Rightarrow \sigma \in \Gamma_P(p)$ is equivalent to the relation $((P+C)\|C) \equiv (P\|C)$.

The relation $((P\|C)\|C)$ is equivalent to the relation $(P\|C)$ because the parallel composition is associative $((A\|B)\|C) \equiv (A\|(B\|C))$ and idempotent $(A\|A) \equiv A$.

Calculus of the relation *(P+C)||C*:

$$\forall (p,\sigma,p') \in E_P, \forall (q,\rho,q') \in E_C \begin{cases} ((p,q,q),\sigma,(p',q',q')) \in E_{(P+C)\|C} \\ \qquad \sigma = \rho \in \Gamma_P(p) \text{ and } \Gamma_C(q) \\ ((p,q,q),\sigma,(p',q,q)) \in E_{(P+C)\|C} \\ \qquad \sigma \in \Gamma_P(p) \text{ and } \sigma \notin \Sigma_C \\ ((p,q,q),\rho,(p,q',q')) \in E_{(P+C)\|C} \\ \qquad \rho \in \Gamma_C(q) \text{ and } \rho \notin \Sigma_P \end{cases}$$

The broadcast composition *P+C* authorizes the common events of the *P* and *C* alphabets to occur even if only one machine authorizes it. The parallel composition between *P+C* and *C* allows restricting the occurrence of the common events of the *P* and *C* alphabets occurring only in *C*: indeed, the events that can simultaneously occur in *P* and *C* are not affected; the common events that can occur in *P* but not in *C* are not affected in the parallel composition; however, the common events that can occur in *C* but not in *P* are erased; the events that are not common to *P* and *C* are never affected.

Calculus of the second member of the relation *(P||C)||C*:

$$\forall (p,\sigma,p') \in E_P, \forall (q,\rho,q') \in E_C \begin{cases} ((p,q,q),\sigma,(p',q',q')) \in E_{(P\|C)\|C} \\ \qquad \sigma = \rho \\ ((p,q,q),\sigma,(p',q,q)) \in E_{(P\|C)\|C} \\ \qquad \sigma \in \Gamma_P(p) \text{ and } \sigma \notin \Sigma_C \\ ((p,q,q),\rho,(p,q',q')) \in E_{(P\|C)\|C} \\ \qquad \rho \in \Gamma_C(q) \text{ and } \rho \notin \Sigma_P \end{cases}$$

By comparing the two members of the relation, we can determine which the event occurrences that reject the relation are.

$$\forall (p,\sigma,p') \in E_P, \forall (q,\rho,q') \in E_C$$
$$\begin{cases} ((p,q,q),\sigma,(p',q',q')) \in E_{P\|C} \text{ if } \sigma = \rho \in \Gamma_P(p) \cap \Gamma_C(q) \\ \qquad \in E_{(P+C)\|C} \text{ if } \sigma = \rho \in \Gamma_P(p) \cap \Gamma_C(q) \\ ((p,q,q),\sigma,(p',q,q)) \in E_{P\|C} \text{ if } \sigma \in \Gamma_P(p) \text{ and } \sigma \notin \Sigma_C \\ \qquad \in E_{(P+C)\|C} \text{ if } \sigma \in \Gamma_P(p) \text{ and } \sigma \notin \Sigma_C \\ ((p,q,q),\rho,(p,q',q')) \in E_{P\|C} \text{ if } \rho \in \Gamma_C(q) \text{ and } \rho \notin \Sigma_P \\ \qquad \in E_{(P+C)\|C} \text{ if } \rho \in \Gamma_C(q) \text{ and } \rho \notin \Gamma_P(p) \end{cases}$$

The necessary and sufficient condition to verify the property is:

$$\text{if } \forall (p,\sigma,p') \in E_P, \forall (q,\rho,q') \in E_C \,|\, \exists s \in (\Sigma_C \cup \Sigma_P)^*$$
$$\text{with } \delta_P(Q_{0P}, s) = p,\ \delta_C(Q_{0C}, s) = q,$$
$$\nexists \rho \in \Gamma_C(q) \text{ and } \rho \notin \Gamma_P(p) \text{ and } \rho \in \Sigma_P$$

In other words, that signifies that, from a couple of states simultaneously reached by the plant model and the completeness specification, an event that the specification authorizes cannot exist if the plant model recognizes this event but does not authorize it.

It can be written more formally:

$\sigma \in \Sigma_C \cap \Sigma_P$ *and* $\sigma \in \Gamma_C(q) \Rightarrow \sigma \in \Gamma_P(p)$ QED

5. EXAMPLE OF APPLICATIONS

5.1. Example models used to define compositions

The result of the (P+C)||C finite state machine of the previous example is described on Figure 4.

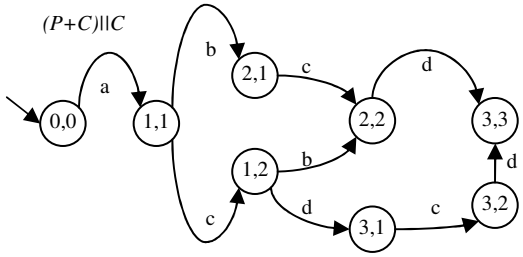


Fig. 4. Finite state machine result for the illustrated example

The transition from state 1,2 when event *d* occurs is deleted from *P+C* because of the parallel composition with *C*. If we compare this finite state machine with the finite state machine representing *(P||C)||C* equivalent to the finite state machine presented on Figure 2, we can conclude the property is not verified because sequence *acd* is accepted by *(P+C)||C* but not by *P||C*. That reveals event *d* cannot occur when the *P* current state is *1*. As a conclusion, we can assert that the confrontation between the completeness specification and the plant model is not approved.

5.2. Modeling of a cylinder with 1 sensor

A pneumatic cylinder, supplied by a single pilot valve and measured by an on/off sensor *a* constitutes the studied plant model that is presented on Figure 5.

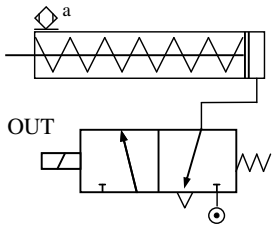
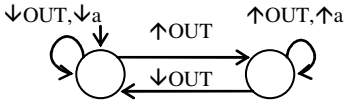


Fig. 5. Plant constituted of a cylinder a valve and a sensor

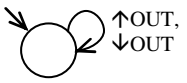
Figure 6 presents an empirical plant model. *OUT* output activation authorizes sensor *a* activation. Consequently, the ↑*OUT* controllable event allows the occurrence of uncontrollable event ↑*a*. In the same way, sensor a can be deactivated only if *OUT* output is deactivated. So, ↓*OUT* controllable event authorizes ↓*a* uncontrollable event occurrence.



with Σ_P={↑OUT,↓OUT,↑a,↓a}

Fig. 6. Empirical plant model

The controllability specification presented on Figure 7 check if the plant model accepts {↑*OUT*,↓*OUT*} events whatever active the plant model state. The specification is built of one state and two transitions associated to the controllable events. The sensor situations and the uncontrollable events do not intervene because the goal is to verify that the controllable events do not depend on the uncontrollable event occurrences.



with Σ_C={↑OUT,↓OUT}

Fig. 7. Controllability specification

In this example, the proposed property confronts the completeness specification to the plant model. The result property demonstrates that the plant model is complete according to the controllability specification.

5.3. Modelling of the arm manipulator

(Music et al., 2002) used a low level plant model to calculate the supremal controllable language of the arm manipulator. The goal of this research is to use the control synthesis from Ramadge and Wonham to determine the control model to implement in the Programmable Logical Controller that will control the plant. The studied part of the system is composed of a vacuum transducer, an electro-pneumatic valve and a pressure switch. The bistable valve is controlled by two Boolean signals: a signal *ag1* asks to suck in; the other one *ag0* asks the ejection. When the two signals are in the same status, there is memorization of the valve situation in the previous situation. The pressure switch *sg1* detects the product capture by the arm manipulator. This part of the plant can be represented by the diagram on Figure 8.

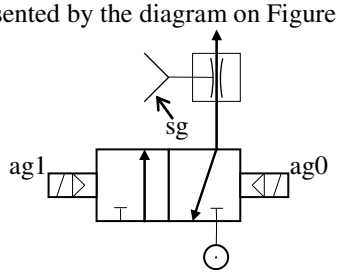


Fig. 8. Gripping device plant

The plant model that the authors propose for this part of the system is shown in Figure 9. The plant model contains ten states that correspond to the states reachable by the system during the functioning. The transitions associated to the changes of a signal from *0* to *1* or *1* to *0* are noticed with edges.

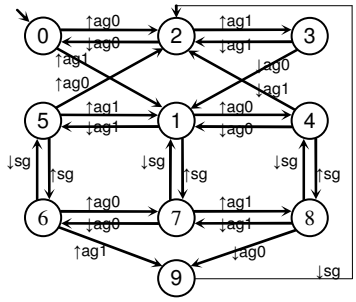


Fig. 9. Plant model of gripping device

We propose to put to the test this plant model with a completeness specification to verify if the plant model is independent from the control part that controls it. Each control signal change of the valve must be accepted in the proposed plant model.

The completeness specification that describes all the control changes that can occur must be described as a finite state machine. In this specification, we assume that the control changes of a signal alternate between rising edges and falling edges. This specification is initialized to be appropriate with the initial state of the plant model. This model is performed by indicating all the possible control changes. The pressure switch *sg* is not indicated to keep the independence of the completeness specification and the sensor behavior.

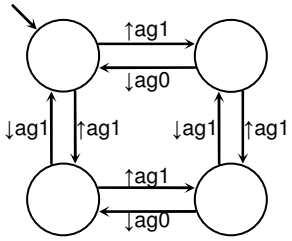


Fig. 10. Gripping device completeness specification

The result of the confrontation of the plant model and the completeness specification indicates the plant model is not complete according to the specification. Indeed, in state *9* of the plant model, the authors admit the product immediately falls down when the suck in control is inactive. No other event is tolerated between the stop of the suck in control and the deactivation of the pressure switch. As a consequence, this implies the delay between the falling edge of the control and the value of the sensor is nil. The specification was elaborated without this assumption. The plant model can also be challenged without making this assumption. In this case, control signals *ag0* and *ag1* can change in the state *9* of the plant model.

These examples show the use of the property on several cases. However, the use of the property on a partial plant model with a local specification was not presented because the models representation is too large. The result of the property allows to test if the plant model is adapted to some applications. When the property is not verified, the plant model or the specification can be challenged but the property facilitates the confrontation by the automatic and computerized treatment.

6. CONCLUSION AND PERSPECTIVES

Designing a plant model is a major problem in the discrete event system research. The enunciated property is a way to increase the designer confidence in the plant model he designs. The confrontation between the plant model and a completeness specification may indicate the omission the designer might have done. Using this property on a partial plant model allows a great flexibility. The applications that are based on decentralised or modular plant models can also use the property to increase confidence in the plant models and the application results. The use of the property on simple examples shows the interest of the verification of a plant model and can put forward the assumptions that the designer has implicitly made during plant modelling. The perspectives about this work are to integrate the property in a methodology to obtain the plant model. The use of such property on a plant model used for scheduling may be improved because decentralized or modular plant models can contain several hierarchical levels of modelling in which such a model can appear. Some pieces of information such as information about the product evolution can complete plant modelling. Consequently, this property may be taken into account and included in the plant modelling methodology.

7. REFERENCES

Akesson, K., M. Fabian, H. Flordal and A. Vahidi (2003). Supremica: a tool for verification and synthesis of discrete event supervisors. *Proc. of the 11th Mediterranean Conference on Control and Automation*.

Balemi S. (1992) Input/output discrete event processes and system modeling. In Balemi et al. [8], pages 15—27. *Proc. of the Joint Workshop on Discrete Event Systems (WODES'92)*, Prague, Czechoslovakia.

Cassandras C. G., Lafortune S. (1999) Introduction to Discrete Event Systems, *Kluwer Academic Publishers*, 1999

Gouyon, D., J.F. Petin and A. Gouin (2004). Pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research42(14)*,2839–2858.

Liu J, Darabi H., (2002) Ramadge-Wonham supervisory control of mobile robot in practice. *Proceeding of ICRA'02*

Malik P., (2002). Generating Controllers from Discrete-Event Models. *Proceedings of MOVEP'2002*. 17-21 June 2002 Nantes, France

Music G., Matko D., Zupancic B. (2002). Model based programmable control logic design. *Proceedings of 15th Triennial IFAC World Congress*, Barcelona, Spain, Jully 2002

Philippot A., Tajer A., Gellot F., Carré-Ménétrier V. (2005). Méthodologie de modélisation dans le cadre de la synthèse formelle des SED, *E-STA*, 2(2), mai 2005.

Ramadge, P.J. and W.M. Wonham (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization 25(1)*, 206–230.

M. Rausch, B. H. Krogh (1998) Formal verification of PLC programs *Proc. AAC'98* Philadelphia PA USA June 1998.

Rohée B., Riera B. Carré-Ménétrier V., Roussel JM. (2006). A methodology to design and check a plant model. *In proceedings of the 3rd IFAC Workshop, Discrete Event System Design 2006*, p245-250, Rydzyna, Poland

Roussel J.M. and A. Giua (2005). Designing dependable logic controllers using the supervisory control theory. *16th IFAC World Congress*, Prague, Czech Republic.

Zaytoon, J. and V. Carre-Menetrier (2001). Synthesis of control implementation for discrete manufacturing systems. *International Journal of Production Research 39(2)*, 329–345.