SYNTHESIS OF SUPERVISED CONTROLLER BASED ON BOOLEAN CONSTRAINTS AND BOOLEAN AUTOMATA

Pascale Marangé, Abdelouahed Tajer, François Gellot, Véronique Carré-Ménétrier

SYNTHESIS OF SUPERVISED CONTROLLER BASED ON BOOLEAN CONSTRAINTS AND BOOLEAN AUTOMATA

P. Marangé, A. Tajer, F. Gellot, V. Carré-Ménétrier

Centre de Recherche en STIC, Moulin de la Housse B.P. 1039, 51687 REIMS Cedex 2, FRANCE. Tel: +33.3.26.91.32.26, Fax: +33.3.26.91.31.06
{ abdelouahed.tajer, pascale marange, francois.gellot, veronique.carre}@univ-reims.fr

Abstract: In order to establish a method to synthesis controllers, an essential step is the modelling of both plant and constraints. However, this step remains a very complex task. To mitigate this difficulty and facilitate modelling, we present a methodology for plant modelling based on rules; as well as a user friendly methodology for constraints modelling based on logical equations in the traditional Boolean algebra. Then, we present an adaptation of the Kumar algorithm synthesis adequated to these new modelling. To conclude, we show that our synthesis approach can constitute a help in controller development and to be diverted from its first function to be used in controller validation.

Key word: Boolean Automata, plant model, constraint model, supervisory control theory, Boolean algebra, validation, controller development.

1. INTRODUCTION

The automation process complexity increases the requirements for designers in terms of operational safety as well as program design reliability. Two approaches are possible: the validation approach and the synthesis approach. We focus on the synthesis approach which consists in building a controller in which the properties required for the system are taken into account from the beginning of the designing process. This work make it possible to characterize the necessary steps to move to a Grafcet specification (IEC, 2002) to the implementation of the corresponding controller. To formalize these steps, we chose to base our approach (Carré and Zaytoon, 2002) on tools/methods with solid formal bases, such as the automaton and the supervisory control theory (SCT) (Wonham and Ramadge, 1987). This formal context was meant to prove the feasibility of this approach while initially not taking into consideration the difficulties involved in its use. Our work thus led to the development of a global synthesis and of an implementation solution of: reactivate controllers as close as possible to the requirements of Grafcet, of the process model to be controlled and of the constraints to be respected on the process evolutions. We showed the approach applicability on several examples in (Carré and Zaytoon, 2002), and (Tajer, 2005). However, the plant and the constraints modelling proved to be a complex task because the SCT recommends event and rudimentary automata models. In addition, the combinative explosion inherent in this type of models limited the use of our synthesis approach to simple systems in terms of number of inputs/outputs.

To mitigate these problems, this approach is based on structured and advanced models while using for the plant the Boolean models containing rules, and for the constraints the logical equations in the traditional Boolean algebra. In order to help the designers develop the controller and refine the starting models they are given the possibility to visualize ant analyse the deadlocking sequences as well as the suggested corrections (Tajer, 2005). The methodology employed for the plant can be considered as an adaptation of Chandra's approach (Chandra and Kumar, 2001) as is the structure implemented here. For the constraints, a modelling by logical equations in the traditional Boolean algebra between the inputs/outputs of the controller is adopted. It is the model user-friendliness and ability to reduce the combinative explosion which is sought here. The scheme takes into account these remarks and presents the approach into as a whole.

After the modelling phase, the controller described in the Grafcet is converted into an automaton of stable situations GSS. To take into account the new models

of plant and constraints, an adapted synthesis algorithm is proposed according to the principles stated by SCT and Kumar, thus generating the supervisor representing the acceptable maximum behaviour of the process. Then, an intersection operation makes it possible to obtain an automaton corresponding to the common behaviour between automaton of the stable situations and the supervisor Boolean automaton. To integrate the user into the loop of controller development and thus to make the approach less sensitive to modelling errors, it is suggested to carry out the construction in either automatic, or semi-automatic mode. In the latter mode, the user can visualize either the evolution traces of the controller leading to a deadlock, or a list of constraints not respected at the level of the controller.

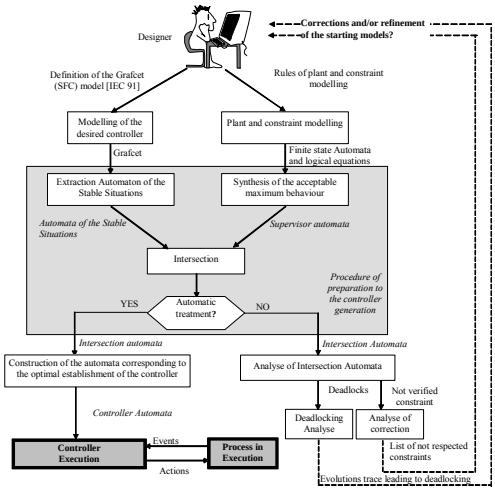


Fig.1. Formal approach synthesis starting from controller specifications expressed in Grafcet

In section 2, we present the methodology of plant modelling, in section 3 the constraints modelling by Boolean equations are presented. In section 4, we briefly give the steps of the synthesis algorithm of the supervisor. In section 5 we develop a help for the designer in developing a controller.

As an illustration, this work will be based on an application designed to automatically sort out cases of two different sizes. The system is composed of a conveyor (conveyor 1) bringing the cases, two double rod cylinders (V1, V2), two simple rod cylinders (V3, V4), and two conveyors (conveyor 2 and 3) for allowing the cases evacuation. The first conveyor takes the cases along one after the other. According to the size of the cases, the push rod made up of cylinder V1 and V2, places the cases in front of cylinder V3 or V4.

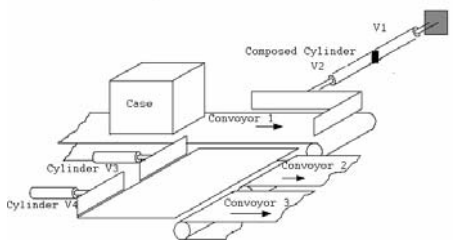


Fig.2. Case sorting system

In the subsequent sections, we will use the following variables: v_{i0} for the input positions of cylinder v_i, v_{i1} for the output positions of cylinder v_i, cp_{11} for the sensor of small cases and cp_{12} for the sensor of large cases. The system outputs are defined by: *GO_OUT1(2)*: order to release the cylinder V1 (V2), *GO_IN1(2)*: order to retreat the cylinder V1 (V2), *GO3(4)*: order to release simple rod cylinder V3 (V4). From there, we define input and output vectors as: $E = \{v_{10}, v_{11}, v_{20}, v_{21}, v_{30}, v_{31}, v_{40}, v_{41}, cp_{11}, cp_{12}\}$, Z = {GO_OUT1,GO_OUT2,GO_IN1,GO_IN2,GO3,GO4}.

2. PLANT MODELLING

The precise description of the plant behaviour is a complex operation because the evolutions of a physical system are asynchronous and nondeterministic. To compensate for the difficulties of a global modelling, we chose a modular approach allowing us to express simple causalities between the plant elements under normal functioning. Consequently, such a model can be derived from using an automaton that accepts the control actions, and can react by changing the logical values of the Grafcet inputs. Considering the systems complexity an adequate methodology is required for modelling the plant. This methodology is based on occurrence rules and precedence relations. These rules define the interactions between controllable or uncontrollable events. The precedence relation specifies the links between uncontrollable events. The user defines the rules and relations which are translated into automata compatible with our synthesis approach. We looked at a normal functioning without considering the possible defects and risks of operation

2.1. Formal framework

To preserve the benefit of the formal framework proposed by the supervisory control theory (Wonham and Ramadge, 1987), plant modelling is carried out under the form of automata describing the physically possible evolutions caused by simple events under normal process operation. To reflect the interactions between the Grafcet controller and the plant, we chose the interpretation of Balemi (Balemi et al., 1993), where the controllable events Σ_c represents the inputs process and the uncontrollable events Σ_u their outputs. The controller can consequently force the input process at any moment and the generation of events is initiated jointly by the process and/or the controller. This interpretation is specific so as to reconcile the nature of continuous actions and input variables of Grafcet with the event related feature of the model theory. Thus, we retained the following correspondence: a controllable event corresponds either to activation $\uparrow Z$ or to deactivation $\downarrow Z$ of a controller Grafcet, while an uncontrollable event is associated with the rising edge $\uparrow E$ or with the falling edge $\downarrow E$ with an input variable Grafcet. The Σ_c and Σ_u sets are written then $\Sigma_c = \uparrow Z \cup \downarrow Z$ and $\Sigma_u = \uparrow E \cup \downarrow E$.

2.2. Occurrence rules and precedence relations

To determine the occurrence rules, it is necessary (i) to fix the initial conditions of the system, (ii) to determine all the events related to the plant element, (iii) to define with rules the influence of the controllable events on the uncontrollable events.

Each rule expresses a "cause/consequence"; the cause relates the controllable event to the consequence (uncontrollable event). For each occurrence rule with the same cause, it is then necessary to establish with preceding relations, the chronology between the consequent uncontrollable events. The occurrence rules and the precedence relations obtained will make it possible thereafter to determine, starting from the initial state, the controllable and the uncontrollable evolutions in the element automaton plant to model.

Table 1: V1 movement parameters

Initial conditions	Occurrence rules	Precedence rules
$\overline{GO\_IN1}$	$\uparrow GO\_IN1 \rightarrow \downarrow v_{10}$ $\uparrow GO\_IN1 \rightarrow \uparrow v_{11}$	$\downarrow v_{10}$ precede $\uparrow v_{11}$
$\overline{GO\_OUT1}$	$\uparrow GO\_OUT1 \rightarrow \downarrow v_{11}$ $\uparrow GO\_OUT1 \rightarrow \uparrow v_{10}$	$\downarrow v_{11}$ precede $\uparrow v_{10}$
$\overline{v_{11}}$	$\downarrow GO\_IN1 \rightarrow \uparrow v_{11}$	nothing
v_{10}	$\downarrow GO\_OUT1 \rightarrow \uparrow v_{10}$	nothing

For example, the complete model consists of four automata describing respectively the movement of each cylinder (example cylinder V1: figure 3.c). These models take into account the cylinder technology. For the movement of the double rod cylinder V1, there are two controllable events (*GO_OUT1; GO_IN1)* and two uncontrollable events (v_{10} ; v_{11}.). In the initial situation, the cylinder is on the input position and no order is sent to the plant. To define the occurrence rules, it is necessary to observe the consequences of the orders sent. The *GO_OUT1* order makes it possible the cylinder to move towards the line and to leave its output position. The chronology of the uncontrollable events is the following: a reset of v_{10} then an activation of v_{11}. The parameters are presented in table 1.

2.3. Plant automaton construction

The automaton structure adopted to Discrete Events Systems (DES) introduced in (Ramadge, 1989), is less explicit because the states do not bring all the information about the system. It is interesting to enrich the model in terms of input and output vectors associated with the automaton in each state. Indeed, the input and output vectors give information both on the plant state and the controller state. For this reason we propose to define an automaton model with Boolean state (Wang, 2000). The automaton can be defined by a 4-tuple G = (P, $\sum$, δ, p_0), where $\sum = \sum_u \cup \sum_c$ is the set of events describing the transition. These transitions are characterized by controlled events $\sum_c$ or uncontrolled events $\sum_u$, δ is the transition function that is defined by δ: P×$\sum$→P, the initial state: p_0 of the automaton G and eventually P the set of state of G that is defined by:

$P = \{[p_1,\ldots, p_n] / p_j \in \{0,1\}, j = 1,\ldots, n\} \subseteq IB^n$

P defines a set of n-dimensional Boolean vectors states. Each component of the vector state can have a value 0 or 1. The number of maximum states of the model automaton is decided by the number of state variables, i.e. for n variable there are 2^n states. In Boole algebra, the set IB is defined by $(\{0.\ 1\}, \wedge, \vee, \neg)$.

In accordance with the semantics recommended by our approach, E and Z are two vectors associated in a state p with the automaton G such as $E = [e_1..., e_m]$ and $Z = [z_1..., z_s]$ when m is the number of the system inputs and s the number of the system outputs. E represents the inputs vector associated with a given state; Z represents the outputs vector associated with this state. The set of the states of G is defined by:

$P = \{[[z_1,\ldots, z_s], [e_1,\ldots, em]] /$
$z_j, e_i \in \{0,1\} * \{0,1\}, j=1,...,s; i=1,\ldots,m, n=s+m\} \subseteq IB^n$.

2.4. The proposed construction model

The plant automaton construction is based on rules and relations which are carried out according to the four following steps:

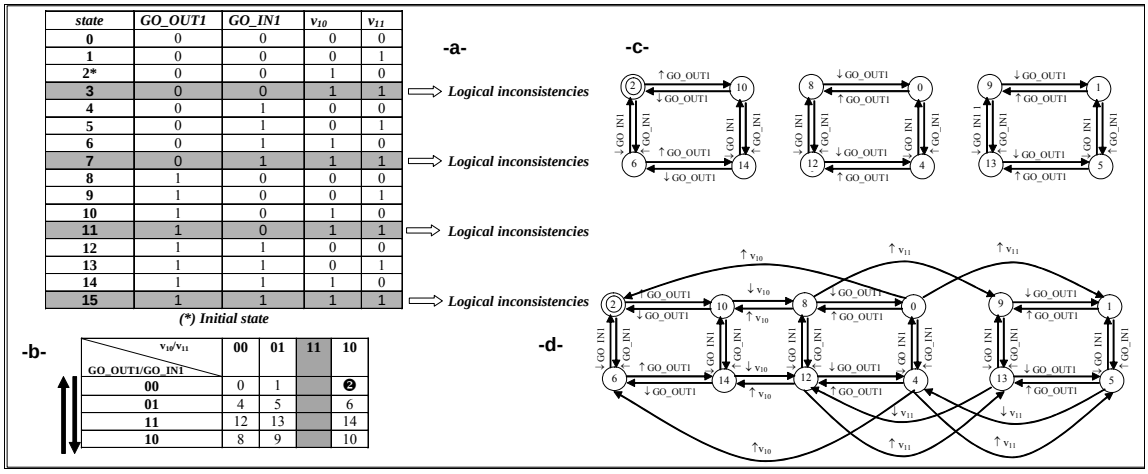
1. Building the table with 2^n states (n is the number of inputs/outputs of the systems) describing all the possible combination between inputs/outputs of the systems (figure 3.a).
2. Deleting the logical inconsistencies between the inputs (sensors). Here, we are not interested in the logical inconsistencies related to the controllable events but those related to the reactions (uncontrollable events) of the plant compared to these events (figure 3.b). For the cylinder example, the position sensors cannot be true at the same time.
3. Building the automaton equivalent to the diagram of the controllable evolutions. The diagram is given starting from the truth table representing the remaining states after the suppression of the logical inconsistencies. For each combination, the occurrence of a controllable event allows to change the output variable which leads to a new state (figure 3.c).
4. Completing the automaton with the uncontrollable evolutions. This operation consists in using the occurrence rules, the precedence relations and initial conditions (figure 3.d).

This method of modelling is carried out for each part of the system. The complete model of plant is then obtained by an asynchronous composition of all the elementary models. The different steps of the approach are illustrated by the example in figures 3.a, 3.b, 3.c, and 3.d.

For the V1 movement, the first step consists in establishing 2^4 combinations between GO_OUT1, GO_IN1, v_{11} and v_{10}. The second step consists in removing 4 logical inconsistencies. It is the case when $v_{10} = v_{11} = 1$ (figure 3.a).

The automaton of the controllable evolutions is determined starting from the truth table thus expressing all the possible evolutions between the states. These evolutions are only defined by controllable events $\uparrow GO\_OUT$, $\downarrow GO\_OUT1$, $\uparrow GO\_IN1$, $\downarrow GO\_IN1$. We note that GO_OUT1 or GO_IN1 . can take value 1 in activation and 0 in deactivation (figure 3.b).

With the precedence relations, the last step gives the final automaton presented in figure 3.c. For example, starting from the initial state (2) the outgoing possible controllable events are $\uparrow GO\_OUT1$ and $\uparrow GO\_IN1$. Occurrence of the controllable event $\uparrow GO\_IN1$ led to the state 6 and that of $\uparrow GO\_OUT1$ to state 10. The relation of precedence corresponding to this event is defined by the uncontrollable event $\downarrow v_{10}$ making it possible to reach state 8. The appearance of event $\uparrow v_{11}$ then leads the plant to state 9. In the same way, the other uncontrollable



state	GO_OUT1	GO_IN1	v_{10}	v_{11}
0	0	0	0	0
1	0	0	0	1
2*	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

v_{10}/v_{11} GO_OUT1/GO_IN1	00	01	11	10
00	0	1		❷
01	4	5		6
11	12	13		14
10	8	9		10

Figure 3. V1 movement automaton

evolutions will be added to the basic automaton.

The automata models of the other cylinders are established in an identical way. The complete plant automaton obtained by the asynchronous product of the elementary models is composed then of 15552 states and 163295 transitions.

3. CONSTRAINTS MODELLING

The constraints to be modelled are of two types: safety constraints (what should not be done) and liveness constraints (what it is necessary to do).To integrate constraints consists in inhibiting actions and/or arranging and sequencing the orders sent to the process. We recommend for these constraints modelling, the use of logical equations which are functions of Grafcet inputs/outputs (Tajer, 2005). The use of the logical equations instead of automaton reduces the combinative explosion. Works also use this formal framework (Roussel et al., 2003) by considering an algebraic approach on the binary digits and taking into account the temporal properties that we do not consider.

The use of the Boolean logical equations has the advantage of expressing the constraints in an explicit and flexible way, in a language well-known to the users. These writing constraints are determined simply by using the implication connector of "*If... Then...*" between the events (inputs/outputs). This implication can express the two types of constraints of safety and liveness but with a static characteristic. For example, in our approach, the cylinder dynamics is not considered.

For example, for the V1 movement, the cylinder cannot go in and go out at the same time i.e., the simultaneous sending of the two orders *GO_OUT1* and *GO_IN1* is prohibited. If we use the automaton to express this specification, the latter is expressed by three states (see figure 4.a). It is to be compared with the logical equation of figure 4.b. obtained through the following implication: "*If* GO_OUT1=1 *Then* GO_IN1 = 0" or "*If* GO_IN1=1 *Then* GO_OUT1=0". This can be also written under the form of a logical expression which is the complement of AND logic between GO_OUT1 and GO_IN1.

Thereafter, we define two other constraints. As in the constraint (1), the cylinder V2 cannot go in and go out at the same time. The second one refers to prohibition for the cylinder V2 cannot go out if a small case is present. The 3 constraints are defined by: GO_OUT1 ∧ GO_IN1 = 0 (1), GO_OUT2 ∧ GO_IN2 = 0 (2), GO_OUT2 ∧ cp11 = 0 (3).

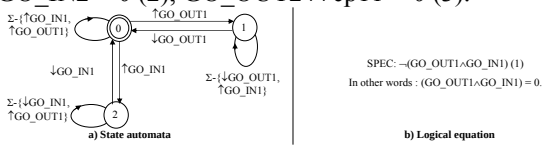


Fig. 4. Constraint: No to GO_OUT1 and to GO_IN1 at the same time

4. SUPERVISOR SYNTHESIS

4.1. Principe

In this part, we seek to synthesize a supervisor using the Boolean automaton models (G = (P, ∑, δ, p_0)) and logical equations for the constraints *K*. The theory of supervision according to RW consists in prohibiting a controllable event in certain states to prevent the system from going towards states that do not meet the specifications.

One of the most important concepts in the supervision theory is the controllability (Ramadge and Wonham, 1987). Within our framework the principle of controllability is:

(∀p ∈ P, such as p verifies K)
∀σ ∈ Σ_u such as δ (p,σ) verifies K
Then K is controllable

The specification K is controllable if and only if at any state that satisfies K, the occurrence of any uncontrollable event σ will not lead to a state that does not satisfy K.

In the case of an automaton modelling, one of the most used algorithms for this objective is the Kumar algorithm (Kumar, 1991). Taking into account modelling based on automata in Boolean state. Existing publishes allow for a synthesis known as symbolic system to obtain a supervisor based on Boolean automata (Gunnarsson, 1997). However, in our step, we show how to adapt the algorithm of Kumar to our Boolean models, allowing to preverse the initial logic of the synthesis approach.

4.2. Synthesis Algorithm

This algorithm receives as inputs a Boolean automaton G representing the total model of the process, a set of logical specifications representing

the constraints of safety and liveness as well as the initial structure of the resulting automaton given by $(\Sigma, \{q_0\}, \delta, q_0)$. The reader will find the formalism of this algorithm in (Tajer, 2005).We point out here briefly the three steps of the algorithm:

The **first step** processes the controllable events related to the activation or the deactivation of a controller output z ($\uparrow z$, $\downarrow z$). If, as a set K_s, the logical equations associated *s* are true, then the corresponding evolution is not forbidden and the corresponding transition is added to the unit from the transition Δ.

The **second step**, starting from a state running p, consists in building if they do not exist yet, the evolutions of SUP characterized by uncontrollable events. These uncontrollable events correspond to the reactions of the plant compared to the control commands. In this step, there are two evolutions:

- The first evolution consists in identifying the forbidden states leading towards prohibited states where the constraints K are not true.
- The second evolution consists in identifying the weakly forbidden states. It presents all the states from which there are a sequence of uncontrollable events w making it possible to reach a defended state.

The **last step** which consists, first in deleting all the forbidden states, the weakly forbidden ones as well as the transitions upstream and downstream that are associated to them. Then the states, not accessible starting from the initial state, are withdrawn from the automaton obtained.

5. HELP TO CONTROLLER DEVELOPMENT

5.1. Introduction

Until now, the work that we undertook, aimed at implementing a controller while following an automatic logic synthesis. Hence, this masked for the designer the deadlocks processed and the corrections carried out before the implementation. The idea was to show the corrections on original Grafcet at the time of the controller execution (figure 1).

It is at the intersection level between the controller automaton and the supervisor automaton, which is deadlocking information in real execution and the corrections (inhibition of actions) made to the controller taking into account the constraints.

It is by exploiting the region of the intersection automaton, that the following stage guarantees that the automaton to be established is not only deterministic and reactive, but represents also the minimal restriction of the behaviour of Grafcet, guaranteeing born conformity to the specified constraints and not to block the system.

For simpler systems, the designer request is different and is directed rather towards a request for assistance to the controller design. The required objectives do not really relate any more to the concepts of controller optimality and sedentary approach, but taking the designer into account in the loop of the controller development. Consequently, new services can be proposed to the designer as for example the visualization of the deadlocking sequences which are completely masked in the automatic approach. This makes it possible for the designer to act on the reasons for deadlocking for example modifying the controller, relating constraints or refining the plant model. The new synthesis iterations can be then carried out starting from the corrected models to finally generate a correct controller model (Tajer, 2005). We will speak then about the step of semi-automatic synthesis of the controller.

Consequently, it is a question of exploiting the information of the intersection automaton to detect (i) deadlocks and (ii) the corrections introduced taking into account the constraints. Two cases which can occur at the end of the intersection operation:

- The controller respects the specifications imposed on the system, then the designer can establish the controller synthesized in a P.L.C. in full safety, it is thus certain that there is no risk for the operative material, the operators or the products.
- The intersection procedure detects:
  - Deadlocking situations, the designer analyzes the situation blocking thanks to elaborate information starting from the region of the intersection automaton and acts consequently to modify for example the controller, to relax constraints or to refine the plant mode.
  - The non respect of the designer intentions. Indeed, there exists, in the automaton intersection, some corrections generated to take into account the constraints imposed on the system.

The designer will visualize a constraints list that is not respected at the level of his original controller and will infer some possible modifications. A new iteration of the procedure will then be carried out starting from some corrected models.

5.2. Application

Figure 5.a presents an evolution trace leading to a deadlock in real execution. Indeed, no event makes it possible to leave the region r_{18} built by the procedure of intersection between the controller automaton and the supervisor automaton. The deadlocking situation is thus characterized by the controller situation: steps X102, X300, and X400 activate: figure 5.b and the situation of the supervisor: state q_4: figure 5.c. At the level of the controller steps, the only order sent to the plant is GO_OUT2. At the supervisor level, order GO_OUT2 is prohibited, orders $\uparrow GO\_OUT1$, $\uparrow GO\_IN1$, $\uparrow GO\_IN2$, $\uparrow GO3$ are authorized. The deadlock identified is thus logical because the intersection between the orders sent to the plant and the orders authorized by the supervisor is empty.

The event that led to deadlocks is the arrival of the small case ($\uparrow cp11$). Because the constraints (3) specified by the designer, prohibited to leave cylinder P2 (order GO_OUT2) in the event of the presence of a small case, is not true. It is thus impossible that the receptivity associated the transition (3) from Grafcet G1 be passable because order GO_OUT2 is prohibited thus v_{21} cannot occur, thus the system is blocked. It should be noted that this deadlocking occurs only in the event of the presence of small case. We are in the typical situation where the plant is in an incompatible state with the sensitized transition. In conclusion, the designer can slacken the constraint because he considers it too restrictive or

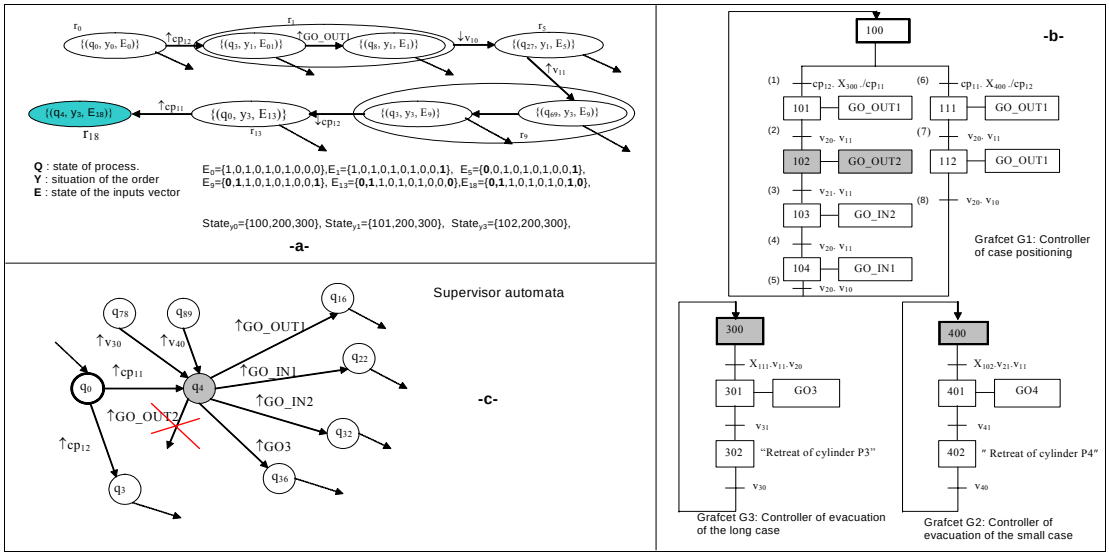


Fig. 5. Example of case storing

modify its Grafcet G1 to take into account this particular condition.

6. CONCLUSION

Using the supervisory control theory in our synthesis step made it necessary for us to use the automata to model the plant and the constraints. This modelling led, to both a design difficulty, and a combinative explosion problem. Indeed when using the algorithm of Kumar (Kumar, 1991), the complexity to generate the supervisor automaton is the worst case proportional to N^nM^k, (N being the states number of each process automaton, n is the automata number, M is the states number in each constraint and k the constraints number).

To compensate for these difficulties, we presented an advanced model of plant and constraints. We used a structured model containing rules for the plant, and a model based on logical equations in the Boolean algebra for the constraints. To take into account these new models, we used a new adequate synthesis algorithm. The algorithm complexity is expressed by $k2^n$ (where K is the number of constraints and n the number of variables), that makes it possible to reduce the combinative explosion in terms of states and transition numbers. The next table illustrate this subject.

Table 2: Algorithm comparison

	Algorithm using Boolean automata	Kumar algorithm using rudimentary automata
States	1571	16524
Transition	21456	153432

The number of states generated by this approach is proportional to $|GSS|*2^nk*2^{\Sigma u/2}$ where GSS represent the number of states of the Automaton of the Stable Situations resulting from the controller specification expressed in Grafcet (figure 1).

We showed that this approach could be diverted from its first function which is known as controller synthesis and thus can be used as a procedure for controller validation. As well as, it's possible to use it at the teaching problems of industrial automatisms where the students are confronted with the use of real plant allowing them to test control laws while using industrial material.

REFERENCES

Balemi S., G.J. Hoffmann, P. Gyugyi, H. Wong-Toi, G.F. Franklin, "Supervisory control of a rapid thermal multiprocessor", *IEEE Transactions on Automatic Control*, vol. 38, n°7, p. 1040-1059, 1993

Carré-Ménétrier V., Zaytoon J., "Grafcet: behavioural resulting and control synthesis", *European Journal of Control*, vol. 8, n°4, p.375-401, 2002.

Chandra V., Kumar R., "A Discrete Event Systems Modelling Formalism Based one Event Occurrences Rules and Precedence", *IEEE Transactions one Robotics and Automation*, vol. 17, n°6, 2001.

Gunnarsson J. Symbolic Methods and Tools for Discrete Event Dynamic Systems, *PhD Thesis*, Linkoping University, 1997

International Electrotechnical Commission, "Preparation of function charts for control systems". Publication 848, 2002

Kumar R., "Supervisory Synthesis Techniques for Discrete Event Dynamical Systems", *Thesis for Ph. D. Degree*, University of Texas, 1991.

Ramadge P.J., Wonham W.M., "Supervisory control of discrete events system", *Proceeding IEEE, Special issues on discrete events systems*, 77(1): pp81-98, January 1989

Roussel J.M., Medina A., Faure J.M., "Synthèse d'un programme de commande d'un système logique à partir de l'expression algébrique de ses spécifications", *Actes MSR'03*, pages 77-93, 6-8 octobre 2003, Metz

Tajer A., "Contribution aux approches formelles de synthèse de commande spécifiée par Grafcet", *Thèse de doctorat de l'université de Reims Champagne Ardenne*, novembre 2005

Wang Y., "Supervisory control of Boolean discrete-Event Systems", M.A.Sc. *Thesis, Dept. of Electl. & Cmptr Engrg.*, Univ. of Toronto, June 2000

Wonham W. M., Ramadge P.J., "On the supremal controllable sublanguage of has given language", *SIAM J Control Optimization*, flight 25, n°3, p.637-659, 1987