Grafcet: Behavioural Issues and Control Synthesis

Véronique Carré-Ménétrier, J. Zaytoon

European Journal of Control (2002)8:375–401

European Journal of Control

Grafcet: Behavioural Issues and Control Synthesis

V. Carré-Ménétrier and J. Zaytoon*

LAM, Faculté des Sciences, France

This paper discusses the difficulties encountered in identifying the correct behaviour of Grafcet specifications in view of synthesising a correct control execution model. Formal algorithms for the synthesis of supervisory controllers related to Grafcet are also proposed. These algorithms consider the constraints induced by the behaviour of the controlled plant so as to identify the required correct behaviour of Grafcet during control execution. The resulting control execution is deadlock-free, and represents the minimal possible restriction of the behaviour of a given Grafcet, subject to a number of specified constraints.

Keywords: Grafcet; Automata; Control synthesis; Supervisory control theory; Intersection; Behaviour

1. Introduction

Grafcet or function charts for control systems is an international standard used for the specification and the implementation of logic controllers in manufacturing systems [10,14]. Throughout the twenty years that have passed since it was defined, Grafcet is becoming widely used in the industry [3] and in education [16]. The main contribution of Grafcet, which uses a Petri-net like formalism, is that it allows a clear modelling of inputs and outputs and of their relations. It also allows modelling of concurrency and synchronisation. This simplifies the specification and the simulation of the control logic of the system. Many PLC builders today use the Grafcet as a specification and/or as a programming language. Among the large companies using it widely or recognising it as an internal standard are: Siemens, Renault, Peugeot, Michelin, and others. However, and in spite of its advantages, Grafcet has long been criticised because it is not supported by a formal foundation that guarantees correctness and safety requirements for the target controller implementation. Many approaches have, therefore, recently emerged [28,31] to provide formal verification possibilities to Grafcet. A more challenging problem is that of providing a formal framework for the automatic synthesis of an optimal control implementation for a given Grafcet.

The supervisory control theory has been introduced [24] to provide algorithms for the synthesis of supervisory controllers from their specifications. Despite its theoretical appeal, there are very few control logic synthesis applications based on this theory. This is due to the fact that the proposed logical model assumes a plant that generates events spontaneously and the only control mechanism available to the supervisor is the ability to enable or to prevent the occurrence of some events called controllable events. In contrast, real time systems usually react to commands as inputs with responses as outputs.

Despite the numerous extensions of the supervisory control theory and intensive research efforts on the theoretical modelling, only a few applications of

Correspondence and offprint requests to: V. Carré-Ménétrier, LAM, Faculté des Sciences, B.P. 1039, 51687 Reims Cedex 2, France. Tel.: +33 3 26 91 32 26; Fax: +33 3 26 91 31 06. E-mail: veronique.carre@univ-reims.fr
*E-mail: janan.zaytoon@univ-reims.fr

Received 14 June 2001; Accepted in revised form 18 February 2002. Recommended by A. Giua and A. Benveniste.

supervisory controllers were reported [1,2,4,7,19]. In [2], controlled automata are employed and a computer in which the supervisory-control strategies resided was directly wired to the devices to be controlled. The control strategy in [1,4] was developed based on controlled automata manually translated into a ladder logic code, and subsequently programmed manually into the PLC. In [19], an implementation method that utilises results of the supervisory control theory in conjunction with PLC technology is used for online generation of limited-size control strategies. A host PC generates the online control strategies and downloads them to a PLC that supervises a manufacturing workcell, reacting to events and enforcing device behaviour based on the current strategy. In [7], implementation of the supervisory control theory for control of a miniature assembly line from LEGO blocks is presented. The controller is extracted from the maximally permissive supervisor for the purpose of implementing the control by selecting, when possible, only one controllable event among the ones allowed by the supervisor. The controlled automata used in these approaches provide a general framework for establishing fundamental properties of discrete-event controllers problems. However, they do not always represent convenient or intuitive models for practical systems because of the large number of states they have to introduce to represent several interacting subsystems, or because of the lack of structure.

These limitations have motivated the use of Petri nets to provide compact descriptions in the context of supervisory control [12,13,18,21,27,33] because the structure of the net may be maintained small in size even if the marking grows. Most of these existing Petri net based approaches are concerned with the supervisory control level, which refers to maintaining overall control of a number of machines or subsystems including PLCs to prevent conflicts and enable co-operative or shared tasks. They don't address the control of the operating cycle of individual machines. This paper addresses this local control level by using Grafcet, which is derived from Petri nets and widely spread in industrial applications for the specification of PLC controllers.

The objective of the paper is to discuss the aspects related to the generation of a correct behavioural model for a given Grafcet, and then to give formal algorithms for the synthesis of the corresponding optimal control implementation. The description of the behaviour of Grafcet in terms of an automaton of reachable situations will be presented in Section 2. The first contribution of this paper (Section 3) is to provide an extensive discussion, illustrated through an example of a parts-handling system, about the behavioural and semantic issues to be considered when using Grafcet as a specification model in view of synthesising a control execution model. This presentation emphasises the necessity of modelling the controlled plant in view of identifying the correct behaviour of Grafcet. The concepts to consider for the synthesis of an optimal control implementation for a given Grafcet has been intuitively presented in a previous paper by the authors [30]. The second contribution of this paper (Section 4) is to provide an algorithm formalising these concepts, and to illustrate the results of the application of the consecutive steps of the algorithm using the same example of the parts-handling system. This synthesis algorithm generates an execution control model representing the most permissible subset of the behaviour of Grafcet that is deadlock-free, and satisfies the imposed constraints.

2. Grafcet and the Automaton of Stable Situations

Grafcet (or sequential function charts) is an international standard used for the specification of logic controllers in manufacturing systems [9,15]. This model consists in describing parallel and synchronised sequences of elementary operations (Fig. 1) applied to the plant with due consideration to plant's response. The basic concepts of Grafcet are quite clear and simple. The step, drawn as a square, represents a partial state of the controller to which orders can be associated. A step can be active or idle; associated orders are performed when the step is active and

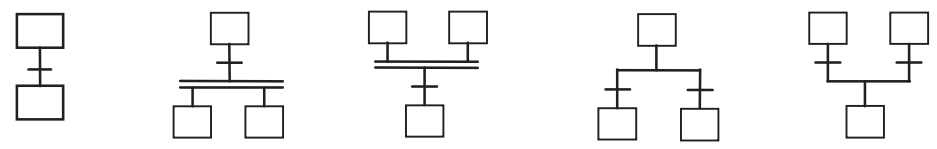


Fig. 1. Different configurations of Grafcet.

remains asleep when it is idle. A situation is given by the set of active steps. The transition, represented as a bar, links one (or several) previous step(s) to one (or several) following step(s). A logical expression, called receptivity, is associated to each transition. This expression manipulates Boolean variables, representing controller inputs or the activation state of individual steps, and events corresponding to rising edges, $\uparrow$, or falling edges, $\downarrow$, of Boolean inputs.

To achieve both reactivity and determinism, it is desirable to extract a behavioural model for Grafcet that evolves instantaneously from one stable situation to another, upon the occurrence of input events. A stable situation is a state in which the Grafcet cannot evolve without acquiring a new input. To ensure that deterministic semantics are conferred to Grafcet, orders are only performed during stable situations; orders related to the intermediate unstable situations remain asleep [11,20,26].

The Automaton of Stable Situations, ASS [25,29] is traditionally used to express the behaviour of Grafcet model. The extraction of this Automaton will be illustrated using the didactic Grafcet depicted in Fig. 2(a). Despite its simplicity, this example illustrates many of the convenient features of Grafcet, including parallel sequences, simultaneous transitions, search for stability, logical expressions involving input variables, events and/or Boolean-step variables. Capital letters are used for Grafcet orders, and small letters for the input variables. Note that a variable "*Xyy*" represents the activation state (true if active and false if inactive) of step *yy* and that the expression */a* stands for the logical negation of the variable *a*.

Each state of the ASS (Fig. 2(b)) represents a stable situation of Grafcet, and is therefore defined in terms of a set of simultaneously active steps, as well as the orders to be performed during the situation. A transition of ASS, which represents an evolution from one stable situation of Grafcet to another, is labelled with an event that can be associated with a logical expression. Each transition is identified with the corresponding transition(s) of Grafcet.

The initial state of ASS corresponds to the initial situation of Grafcet, in which only step 0 is active and order ACT6 is performed. In this state, transition 1 is taken when $\uparrow f$ occurs to activate the situation $\{1\}$, which performs the orders ACT1, ACT2, ACT3 and ACT4. Then, when $\uparrow a$ occurs, and provided that *d* is *true*, transition 2 activates steps 10 and 20. In this situation, where no orders are performed, only transition 3 can occur (because transition 6 is conditioned with the activation of step 11) to activate the situation $\{11, 20\}$. The next evolution depends on the values of the inputs *a*, *b* and *e*. If *a* is *true* when $\uparrow b$ occurs, transition 6 activates the situation $\{11, 21\}$. On the other hand, if *b* becomes *true* before *a*, the next evolution will be conditioned by *e*. If *e* is *false* when $\uparrow a$ occurs, transition 4 is taken and the resulting situation $\{12, 20\}$ is stable, because transitions 5 and 6 are not enabled. Otherwise, a *true* value for *e* when $\uparrow a$ occurs implies that $\{12, 20\}$ is unstable, because transition 5 can be taken immediately. Therefore, order ACT5 will not be performed in this case and a second evolution step will take place instantaneously to attain the stable situation $\{13, 20\}$, which causes ACT6 to be performed. This two-step evolution is represented on ASS of Fig. 2(b) by the transition labelled with "(4 then 5)". The resulting situation $\{13, 20\}$, which can also be reached when $\uparrow e$ occurs in situation $\{12, 20\}$, is a deadlock because it has no output transition; deadlock should be properly removed by the synthesis process. All the evolutions starting from situation $\{11, 21\}$ will take a few transitions to return to the initial state.

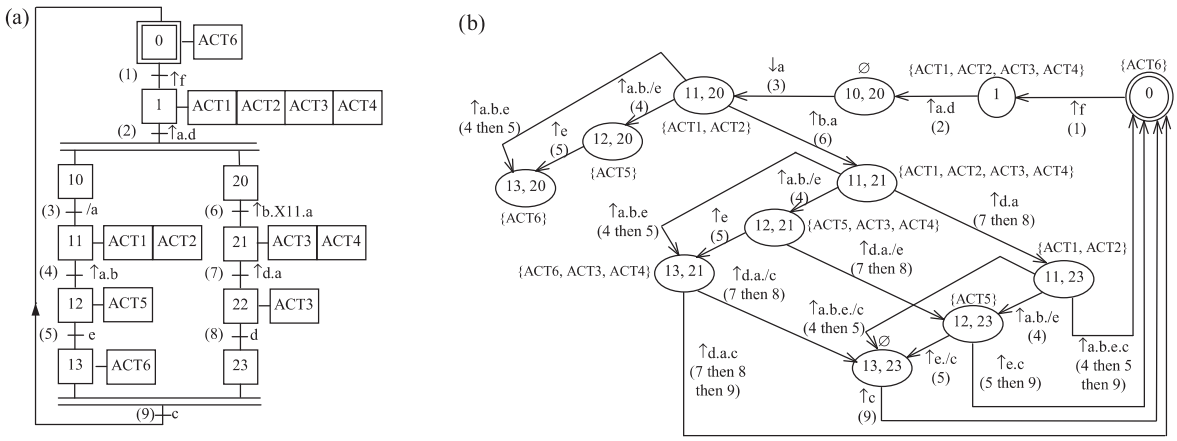


Fig. 2. A didactic Grafcet (a) and the corresponding automaton ASS (b).

From a mathematical point of view, ASS is defined by a 5-tuple (E, Z, Y, T, y_0), where

- E is the set of Grafcet inputs.
- Z is the set of Grafcet outputs.
- Y is the set of states, each of which represents a distinct reachable situation of Grafcet. To each state y of Y, is associated the following two sets:
  - $Z_y \subseteq Z$, the set of active outputs during the state y.
  - Step_y, the set of Grafcet steps that are active during the state y.
- T: $Y \times f(E) \rightarrow Y$ is a partial function representing the transitions (evolutions) between Grafcet states. A transition is represented by the 3-tuple $(y, f(E), y') \in T$, where y is the input state, y' the output state, and $f(E)$ is a logical expression combining input (binary) variables and their edges.
- y_0 is the initial state.

The role of the automaton ASS in view of the synthesis of a correct behavioural execution model for Grafcet will be discussed in the following sections. Section 3 also shows that an explicit model of the controlled plant must be used together with the automaton ASS to obtain a correct behaviour of Grafcet. The synthesis procedure given in Section 4 is based on the use of these two models (ASS and a model of the plant) as well as a model of the constraints to be satisfied by the synthesised controller.

3. Behavioural Aspects: Illustrative Example

The parts-handling example depicted in Fig. 3 will be used to illustrate the necessity of using an explicit model of the plant to identify the correct behaviour of Grafcet in view of the synthesis of a correct control implementation. The system controls the arrival, the storage, and the retrieval of parts having a similar physical structure, but belonging either to type 1 or type 2.

The mechanical structure is composed of a conveyor transporting pieces in two possible movement directions, a platform with clockwise or anti-clockwise rotation, and a moving bridge comprising a carriage for horizontal movement and an electromagnet for vertical movement. The pieces are stored on the platform, which is divided into four subsections, each including an inward and an outward storage position. The moving bridge is used to transfer the parts from the conveyor to the platform.

The electromagnet moves downward and then upward to seize the parts. These movements are performed when the orders DOWN and UP, respectively, are issued by the controller, and the sensors *high* and *low* are used to indicate the upper and lower positions of the electromagnet, respectively. The order EM is used to activate the electromagnet.

The carriage, supporting the electromagnet, can move to the left and to the right when it receives the orders LEFT and RIGHT, respectively. The loading position above the conveyor, *p_conv*, the intermediate position between the conveyor and the platform, *p_inter*, and the storage positions above the platform (outward position of the platform: *p_ext* and inward position of the platform: *p_int*) inform the controller about the location of the carriage.

The platform is divided into four sections and it performs clockwise or anti-clockwise rotations when it receives the orders TURN_*c* or TURN_*ac*, respectively. The $c_{\frac{1}{4}}$ sensor signals a $\frac{1}{4}$ revolution of the platform.

The conveyor advances (or moves back) upon the reception of the order MOVE (MOVE_*back*), and the sensor *p_ part* signals the presence of a part in the loading area. If a part crosses the loading area without being seized by the electromagnet, it will be evacuated by an auxiliary system when it reaches the conveyor end-point.

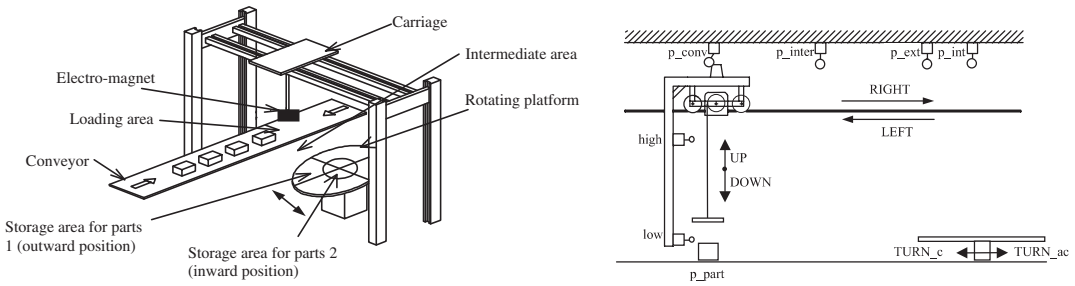


Fig. 3. Parts-handling system.

3.1. Control Specifications

For reasons of simplicity, and size limitation, only the storage mode will be treated here. The input parts are supplied to the conveyor alternatively (in the order: type 1, type 2, type 1, ...) by means of an auxiliary system. In the storage mode, the system is required to place the type 1 parts on the outward position of the platform, and type 2 parts on the inward positions. An intuitive controller specification is given by the 4 partial Grafcets (*G*1, *G*2, *G*3, and *G*4) depicted in Fig. 4. Grafcet *G*1 is used to translate the carriage, *G*2 controls the displacement and the activation of the electromagnet, *G*3 is used to advance the conveyor, whereas *G*4 controls the rotation of the platform. These partial Grafcets are cyclic and hence each of them returns to its initial step upon the accomplishment of the corresponding cycle. They are synchronised by means of the internal variables X102 and X106 corresponding to the seizure of a piece, and the variables X108 and X205 indicating the end of placement of the type 2 part.

Description of Grafcet G1: This Grafcet is used to store a part of type 1 then a part of type 2, before returning to the initial situation. The arrival of a part activates step 101 to translate the carriage to the left until the position above the conveyor, where the sensor *p_conv* activates step 102 to wait for the seizure of the type 1 part. At this point, the carriage is translated to the outward position of the platform, *p_ext*, where step 104 is activated to wait for the deposit of the part. When the end of the deposit is signalled (variable X205), transition 5 is fired to activate step 105 which translates the carriage again to the position above the conveyor. The seizing of a type 2

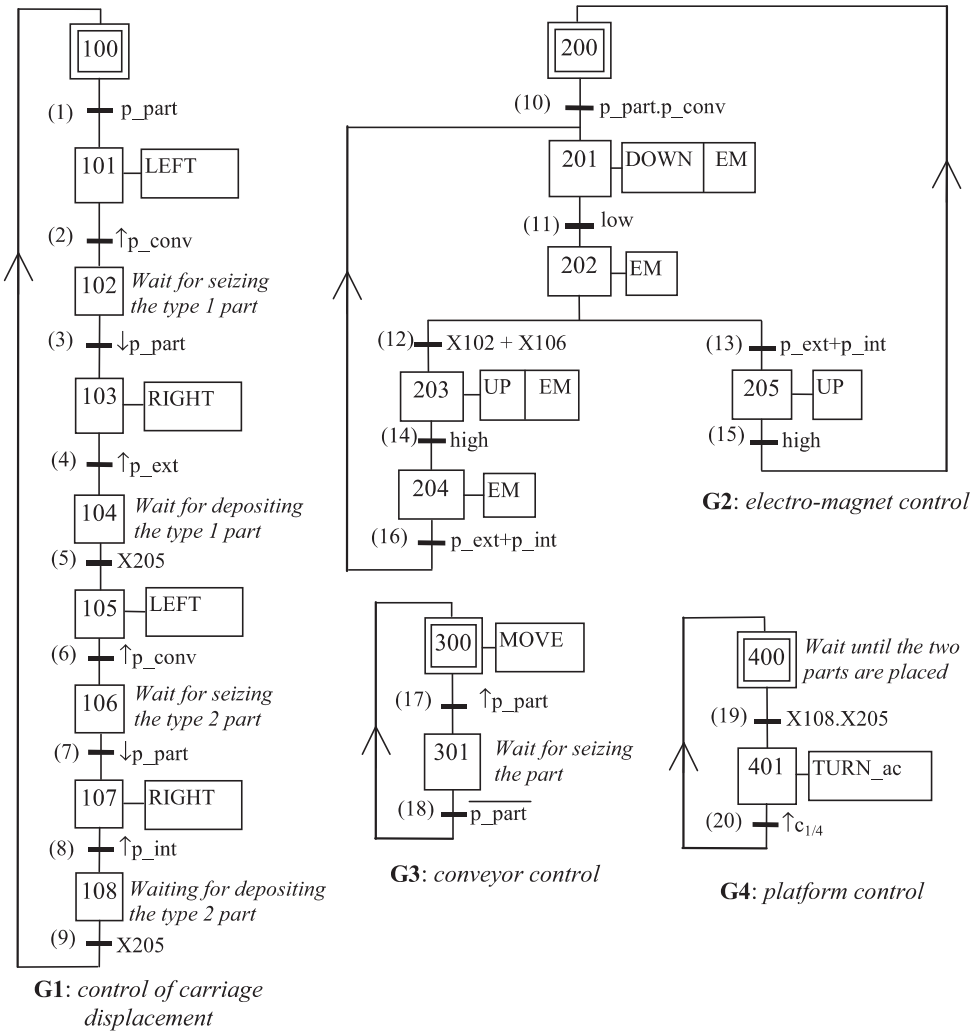


Fig. 4. Grafcet for the example.

part, $\downarrow p\_part$, activates step 107 to order the movement of the carriage to the inward position, *p_int*, of the platform. After placing the part, the carriage starts a new cycle.

Description of Grafcet G2: When a part arrives in the loading area and provided that the carriage is above the conveyor, step 201 is activated to order the activation and the descent of the electromagnet. When the lower position, *low*, is reached, the descent is stopped while the activation of the electromagnet is maintained (step 202). Two evolutions are possible from step 202:

– If the system is waiting for the seizure of a part (variables X102 or X106), the electromagnet is ordered to go up, and its activation is maintained. Then, the arrival of the carriage above a storage area of the platform, *p_ext* or *p_int*, activates a new descent (step 201 then 202) to depose the part.
– If the carriage is above a storage area of the platform (*p_ext* or *p_int*), the electromagnet is deactivated and ordered to move up (step 205) until it reaches its upper position, corresponding to the initialisation of *G*2.

Description of Grafcet G3: This Grafcet is used to advance the conveyor continuously in the absence of parts (step 300). This movement is stopped when a part arrives in the loading area (step 301), and then resumes when the part is seized.

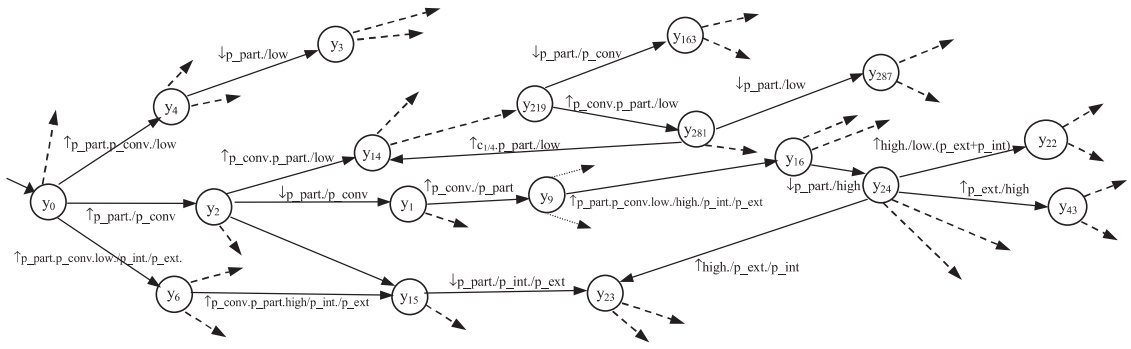
Description of Grafcet G4: Starting from the initial situation, when the two parts are placed (X108 and X205 set to true), the platform performs a clockwise rotation of 90°. The end of this rotation brings G4 back to the initial step to wait for the next rotation.

3.2. Behavioural Analysis of the Controller

The automaton of stable situations, ASS, corresponding to the Grafcet of Fig. 4 has 840 states and 8134 transitions. The first evolutions of this automaton are depicted in Fig. 5, and the formal definition of ASS has been introduced in Section 2.

Note that the automaton ASS corresponds to a super-set of the possible behaviour of Grafcet during control execution because it is calculated without taking into account the constraints related to the reactions of the controlled plant. In fact, the controlled plant induces a number of constraints on the evolutions of the inputs and restricts, by consequence, the behaviour of the controller when it is executed to control the process in real time. These constraints may be static or dynamic. Static constraints are related to the structure of the plant. For example, the inputs corresponding to the two end-of-course sensors of a valve cannot be "true" at the same time. On the other hand, dynamic constraints characterise the evolution scenarios of Grafcet inputs, which depend on the dynamics of the plant and its interactions with Grafcet. The ASS evolutions not complying with these scenarios must, therefore, be eliminated to obtain the correct behaviour of Grafcet.

Through the presentation of a number of trajectories of the ASS of the parts-handling example, the following sections illustrate the necessity of taking the behaviour of the controlled plant into account to be able to identify the correct behaviour of Grafcet, and to correctly analyse deadlock situations as well as other desired specific safety and liveness properties.



$Step_{y0}$ = {100,200,300,400}; Z_{y0} = {MOVE}
$Step_{y1}$ = {101,200,300,400}; Z_{y1} = {MOVE, LEFT}
$Step_{y2}$ = {101,200,301,400}; Z_{y2} = {LEFT}
$Step_{y3}$ = {101,201,300,400}; Z_{y3} = {MOVE, DOWN, EM, LEFT}
$Step_{y4}$ = {101,201,301,400}; Z_{y4} = {DOWN, EM, LEFT}
$Step_{y6}$ = {101,202,301,400}; Z_{y6} = {EM, LEFT}
$Step_{y9}$ = {102,200,300,400}; Z_{y9} = {MOVE}
$Step_{y14}$ = {102,201,301,400}; Z_{y14} = {EM, DOWN}
$Step_{y15}$ = {102,204,301,400}; Z_{y15} = {EM}
$Step_{y16}$ = {102,203,301,400}; Z_{y16} = {EM, UP}
$Step_{y43}$ = {104,203,300,400}; Z_{y43} = {MOVE, EM, UP}
$Step_{y219}$ = {101,200,301,401}; Z_{y219} = {LEFT, TURN_ac}
$Step_{y287}$ = {103,201,300,401}; Z_{y287} = {RIGHT, EM, DOWN, MOVE, TURN_ac}
$Step_{y22}$ = {103,201,300,400}; Z_{y22} = {MOVE, RIGHT, EM, DOWN}
$Step_{y23}$ = {103,204,300,400}; Z_{y23} = {MOVE, RIGHT, EM}
$Step_{y24}$ = {103,203,300,400}; Z_{y24} = {MOVE, RIGHT, EM, UP}
$Step_{y163}$ = {101,200,300,401}; Z_{y163} = {LEFT, MOVE, TURN_ac}
$Step_{y281}$ = {102,201,301,401}; Z_{y281} = {DOWN, EM, TURN_ac}

Fig. 5. First elements of the ASS of Grafcet.

3.2.1. Example of Unrealistic Evolutions of ASS

The situation $\{102, 200, 300, 400\}$, in which the controller waits for the first part to be seized, will be used as an example to illustrate the impossibility of occurrence of many of the evolutions of ASS during control execution. Among the 12 possible ASS transitions originating from this situation (Fig. 6), only two (represented with solid lines) correspond to a possible behaviour of the controller in real execution. The ten other transitions cannot occur in reality because they correspond to unrealistic evolutions such as the simultaneous setting of the two variables *high* and *low* to 1, the activation of *p_conv* while the carriage is above the conveyor, or the occurrence of a falling edge of *p_ part* whereas a rising edge is expected because step 300 waits for the arrival of a part.

3.2.2. Deadlocks

The control execution may lead to deadlocks that do not correspond to blocking situations of ASS. For example, the evolutions depicted in Fig. 7 illustrate the possibility of occurrence of a deadlock in real-execution although it is not identified as such in ASS. At situation $\{107, 201, 300, 400\}$, the occurrence of the event $\uparrow$*low* before $\uparrow$*p_int* results in activating the situation $\{107, 205, 300, 400\}$, in which the carriage is ordered to move right in view of depositing the part on the inward position of the platform. Then, the successive firing of transitions (2), (3) and (4) leads ASS to the situation $\{108, 200, 301, 400\}$, which is apparently non-blocking because it has three output transitions. However, this situation represents a deadlock in real execution because it does not perform any Grafcet order, whereas its output transitions are conditioned by the return of the carriage to the loading position, or by the retrieval of the part.

3.2.3. Safety and Liveness Constraints

Different properties (or constraints) can be specified to verify and validate the control specifications. In the parts-handling example, two safety constraints ($c1$ and $c2$) and a liveness constraint ($c3$) are to be satisfied:

- $c1$: the rotation of the platform should be prohibited during the positioning of a part on it.
- $c2$: to avoid the interactions between the parts on the platform and the parts to deposit, the diagonal movements toward the platform should be prohibited beyond the intermediate position, *p_inter*, when parts are held by the electromagnet.
- $c3$: all the pieces introduced to the conveyor must be eventually retrieved.

In the following, we show that the validation (respectively, invalidation) of a given constraint with respect to the theoretical model of the controller given by Grafcet does not imply that the constraint is system

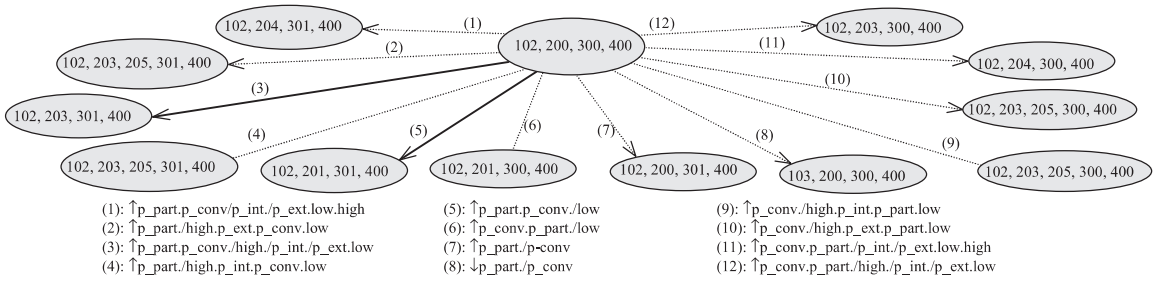


Fig. 6. Output transitions of the situation $\{102, 200, 300, 400\}$ of ASS.

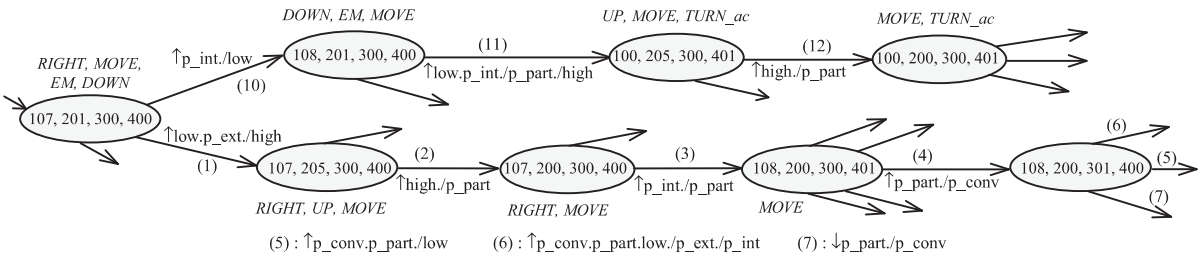


Fig. 7. A trajectory leading to a deadlock in control execution.

valid (respectively, invalid). For example, the execution trajectories of ASS depicted in Fig. 8 highlight a behavioural difference between the control model (given by ASS) and its real execution for c1 and c3.

Figure 8(a) depicts a trajectory of ASS which seems to invalidate $c1$, because it leads to situation $\{104, 201, 300, 401\}$ where the orders DOWN, TURN_*ac*, EM, and MOVE are simultaneously active. However, the logical condition associated with the transition (2) can never be valid in real execution since it implies that the electromagnet is at positions *high* and *low* at the same time. Hence, this trajectory cannot be executed by the control system, and, consequently, it will not invalidate the property in real execution. The second trajectory shown in Fig. 8(b) seems to satisfy the constraint $c3$, because the piece introduced to the conveyor (condition p_part associated with transition (1)) is retrieved later on ($\downarrow p\_part$ associated with transition (3)). However, this retrieval can occur in real execution only if the orders UP and EM are simultaneously active, which is not the case in situation $\{102, 201, 301, 400\}$. Hence, the transition (3) cannot occur in real execution, and the constraint $c3$ is system invalid although it is satisfied with respect to ASS.

3.3. Modelling of the Plant

The above discussion has shown that the distance between the theoretical behaviour of the control model (given by the automaton ASS calculated without taking the reactions of the controlled plant into account) and its real behaviour entails some direct consequences on the properties of the control system. For example, certain deadlocks identified in the ASS are not reachable because the reactions of the plant imply that the executions (evolutions) of the ASS leading to the corresponding situations will never occur in reality. Hence, these "theoretical deadlocks" cannot occur throughout the execution of the controller. On the other hand, even though an ASS is deadlock-free, the corresponding control execution can lead to a deadlock if it brings the plant to a state where the logical expression of the enabled transitions can never become *true*. In the same way, the safety and liveness constraints that are proved to be satisfied with respect to the theoretical model may be invalidated by the real execution of the controller and vice versa.

This implies that some knowledge of plant behaviour is required to be able to identify the correct behaviour of Grafcet and, consequently, to be able to perform model analysis and control synthesis accurately. Indeed, during the specification phase, the control designer tends to implicitly integrate an image of the plant within the developed Grafcet, in order to express the existing causality relations between Grafcet orders and the consequent plant reactions. The underlying hypothesis is that all the reactions of the plant and their interleaving are known precisely and in advance, for each control action. Unfortunately, this hypothesis is restrictive and unrealistic for the case of complex systems involving partial Grafcets and parallel evolutions related to the different elements of the plant. The theoretical behaviour of the resulting model may, therefore, be different from its behaviour when it is executed to control the plant.

It, therefore, becomes necessary to make the image of the plant explicit, by connecting an appropriate abstract model for each of its elements. However, the precise description of the behaviour of the plant is not trivial and the difficulty lies in the choice of the aspects to be modelled and the degree of granularity of the required model [26]. This requires the use of an adequate methodology for the modelling of the plant

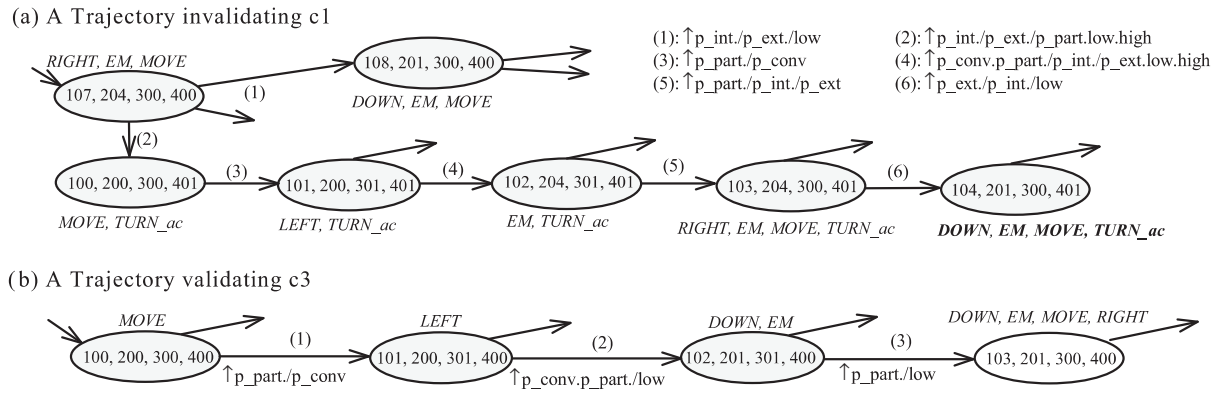


Fig. 8. Illustrative executions of ASS.

[6,32], but this issue goes beyond the scope of this paper. Meanwhile, a modular approach is required to extract a simple model for each of the elements of the plant. Such a model can be derived using automata that accept the control actions, and react by changing the logical values of Grafcet inputs. To be consistent with the synthesis framework proposed in Section 4, the behaviour of the plant will be modelled by means of the spontaneous event generators of the supervisory control theory [24]. Generally speaking, since the control objective does not necessarily concern all the possible evolutions of the plant, a simplified plant model can be used in view of the synthesis procedure. In the following sections, both a detailed and a simplified plant model will be given for the parts-handling example.

3.3.1. Detailed Model of the Plant

In the first place, we consider a detailed model of the plant, which takes all its evolutions into account. This model is given by the five automata representing the translation of the carriage (Fig. 9(a)), the displacement of the electromagnet (Fig. 9(b)), the activation of the electromagnet (Fig. 9(c)), the rotation of the platform (Fig. 9(d)), and the displacement of the conveyor (Fig. 9(e)). The initial state of the plant is fixed as follows: the carriage is situated between the intermediate position, *p_inter*, and the outward position of the platform, *p_ext*, the electromagnet is in its upper position, *high*, and the platform is in its initial position, pos_0. A rising edge, $\uparrow$, and a falling edge, $\downarrow$, correspond, respectively, to the activation and deactivation of the corresponding variable. Capital letters are used for Grafcet orders, and small letters for the variables representing the plant reactions.

Starting from its initial position at state 0, the carriage (Fig. 9(a)) can move to the right (state 1) or to the left (state 2). The activation of the order LEFT entails the movement of the carriage to the left until it reaches the intermediate position (activation of state 4). By continuing the movement to the left, it leaves the intermediate position (activation of state 8) until reaching the position above the conveyor (state 14). From each of these positions, it is possible to stop the movement to the left, $\downarrow$LEFT, then receive the order RIGHT to move right, all the way back. The activation of the order RIGHT in the initial state results in a symmetrical behaviour of the automaton, starting from state 1.

The electromagnet (Fig. 9(b)) can receive the orders UP or DOWN at its upper initial position (state 0). The order UP activates state 1 in which only the deactivation of this order can be accepted. On the other hand, the reception of the order DOWN at the initial state activates state 2 and starts the descent of the electromagnet to leave its upper position (activation of state 4), and then to reach its lower position (activation of state 7) unless the descent is interrupted. The interruption of the descent activates state 5 or 8 where it is possible to intercept the order UP so that the carriage can move to its upper position, and returns therefore to its initial state when the ascent is terminated. Throughout all of these movements, the electromagnet can be activated and deactivated at any instant (Fig. 9(c)).

The platform (Fig. 9(d)) can turn clockwise or anticlockwise. In any of these directions, the platform leaves its initial state (state 0) to reach the four successive storage sections if the movement is not interrupted. If the movement is interrupted, the rotation may be ordered in the opposite direction.

Finally, the conveyor (Fig. 9(e)) can move both forward (states 1, 2 and 4) and backward (states 5, 6 and 8). At the initial state, the order MOVE activates state 1 to advance the conveyor until the arrival of a part in the loading area, which activates state 2. At this state, the movement of the conveyor can either be stopped (activation of state 3), or continued until the part leaves the loading position. In this case, step 4 is activated and it becomes possible to deactivate the order MOVE to bring the conveyor back to its initial state. At state 3, the conveyor can again receive the order MOVE to activate state 2, MOVE *_back* to activate state 6, or $\downarrow$*p_part* (corresponding to retrieval of the part by the electromagnet) to return to the initial state. Starting from the initial state, the behaviour of the conveyor consequent to the MOVE_*back* order is symmetrical to its behaviour induced by the order MOVE.

The global detailed model of the plant, obtained by composing the five automata has 105462 states and 509132 transitions.

3.3.2. Simplified Model of the Plant

The conveyor can move in two directions. However, for our control purpose, only the forward movement is required, and the evolutions related to the backward movement in Fig. 9(e) will not be activated by the specified controller. Figure 10(a), which only considers the forward movement, can therefore be used instead of Fig. 9(e). In the same way, even though the platform can turn in two directions, the control objective is only concerned with the clockwise rotation in view of advancing the section to deposit the next part.

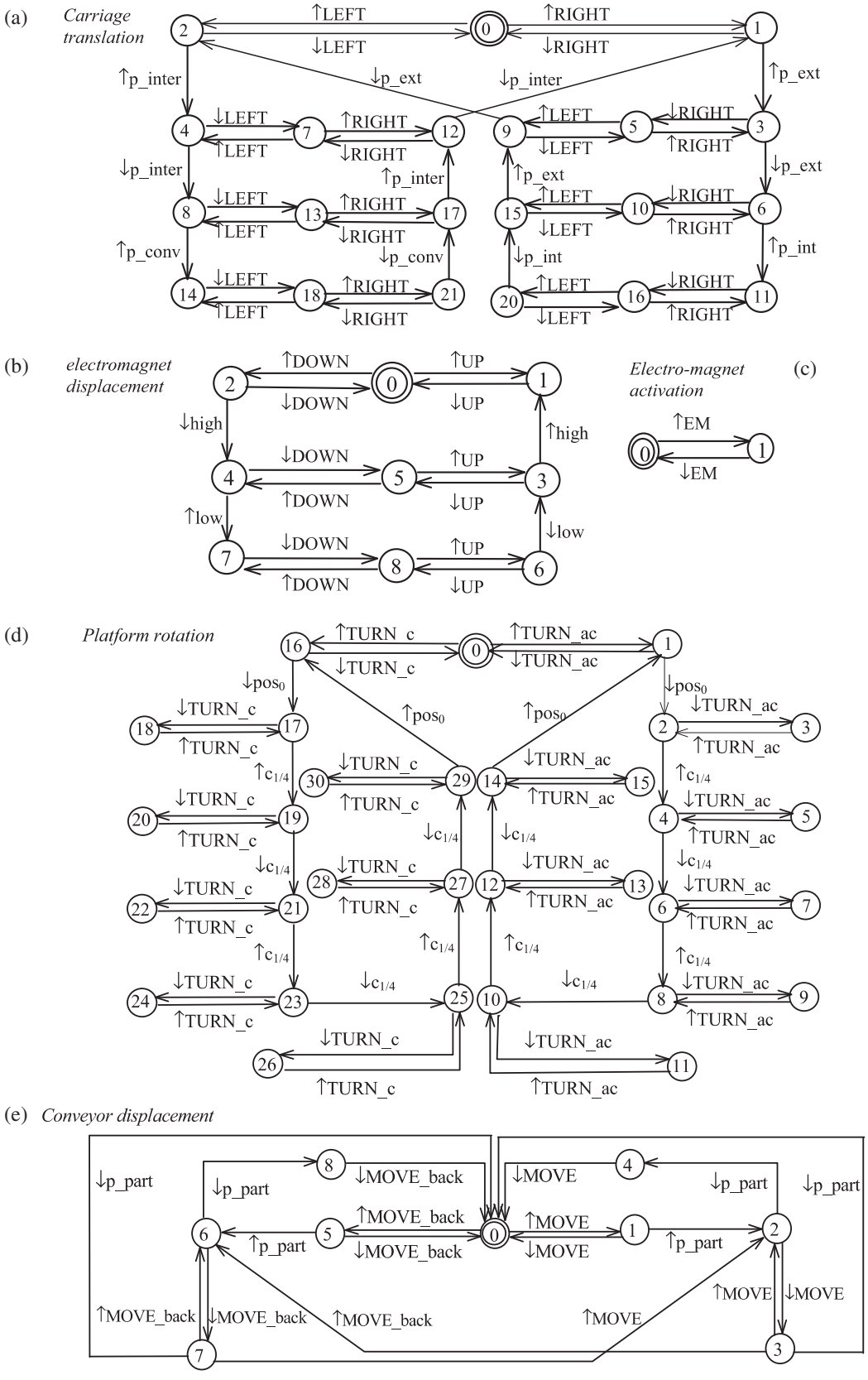


Fig. 9. Detailed model of the plant.

Furthermore, the exact knowledge of the current sector is not necessary. Hence, the model of the platform given in Fig. 9(d) can be aggregated to obtain the automaton of Fig. 10(b).

Now, the simplified global model of the plant can be obtained by composing the two simplified automata of Fig. 10 with the three automata of Fig. 9(a)–(c). The resulting global automaton has 7560 states and 58836 transitions, which represents a significant reduction relative to the global automaton of the detailed model.

3.4. Modelling of the Constraints

The synthesis procedure presented in the subsequent sections requires the use of an explicit model of the desired constraints. Therefore, each of the constraints, $c1$, $c2$ and $c3$, will be expressed here in terms of a corresponding automaton.

The first constraint to satisfy is related to the prohibition of the platform's rotation during the deposit of a part. This constraint is specified using the automaton depicted in Fig. 11(a), which indicates that the orders TURN_*ac* and DOWN cannot be performed simultaneously. State 0 corresponds to the set of states of the process in which all the orders are authorised. The self-looping transition can be taken by all the possible events of the system (given by the set Σ), apart from the activation of the sensor *p_ext*, which corresponds to the arrival of the carriage above the platform, and results in the activation of state 1. Here, the order ↑DOWN activates state 2, in which the automaton cannot receive the event ↑TURN_*c* and ↑TURN_*ac*. The deactivation of the order DOWN brings the automaton back to state 1. In the same way, a rotation order results in activating state 3 in which the order ↑DOWN is prohibited. The initial state can only be attained from state 1, when the carriage reaches the intermediate position, *p_ inter*.

The constraint given by the second automaton of Fig. 11(b) corresponds to the prohibition of diagonal movements toward the platform, beyond the intermediate position, *p_int*, when a part is held by the electromagnet. Seizing a part, ↓*p_ part*, at the initial state (state 0) leads to state 1, from which two evolutions are possible:

- Activation of state 2 if the carriage reaches the intermediate position, ↑*p_inter*, before the electromagnet reaches its upper position. In this case, the order to move right is prohibited until the electromagnet reaches its upper position (activation of state 3).
- Activation of state 3 if the electromagnet reaches its upper position, *high*, before the carriage reaches the

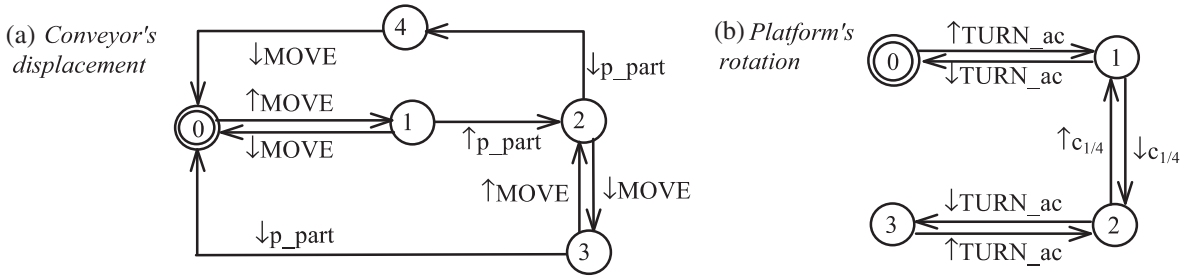


Fig. 10. Simplified models for the conveyor and the platform.

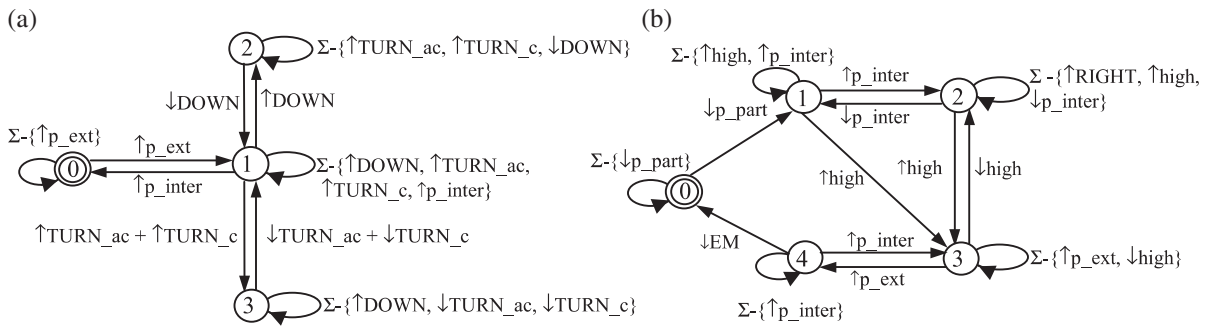


Fig. 11. Models of the safety constraints.

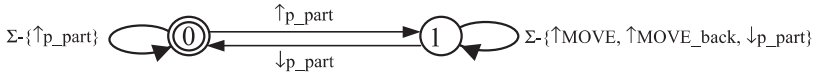


Fig. 12. Model of the liveness constraint.

intermediate position. At state 3, if the electromagnet leaves its upper position, $\downarrow high$, before the arrival of the carriage above the platform, $\uparrow p\_ext$, then state 2 is reactivated and the movement to the right is again prohibited. On the other hand, if the carriage arrives first, state 4 will be activated and, then, the deposit of the part, $\downarrow EM$, brings the automaton back to its initial state.

The third automaton (Fig. 12) represents the liveness constraint where the detection of a part on the conveyor, $\uparrow p\_part$, activates the state 1, in which the order MOVE or MOVE_BACK cannot be accepted. These orders can only be received when the initial state is reactivated again after the removal of the part, $\downarrow p\_part$.

It is desired to control the system so as to satisfy the three constraints simultaneously. The overall constraint is therefore represented by a global automaton whose underlying language is specified by the intersection of the languages corresponding to the automata of Figs 11 and 12. If Σ is given by the set of events of the complete model of the plant (Fig. 9), then this global automaton has 38 states and 1159 transitions. On the other hand, if one only takes into account the events related to the simplified plant (Fig. 10), the resulting global model of the constraints will have 38 states and 1093 transitions.

4. From Grafcet Specification to Supervisory Control Implementation

Control implementation can be synthesised in the frame of the supervisory control theory by computing the supremal controllable sublanguage of the plant language that satisfies the required safety, liveness and deadlock freeness constraints. Such an approach is presented in [7], for example, where the control tasks to be performed are formulated in terms of a set of desired progress specifications. These progress specifications together with the plant model and the required safety constraints are used to derive a supervisor that enforces the specifications while offering a maximum behavioural flexibility. Subsequently, a controller is extracted from the maximally permissive supervisor for the purpose of implementing the control by selecting, whenever possible, only one controllable event from among the ones allowed by the supervisor. The advantage of such an approach resides in the relative simplicity of the formal calculations involved because it uses the supervisory control theory as a sole formal framework.

However, the description of each control task in terms of automata-based separate progress specification restricting the plant model is not a very common practice in industry when specifying the control of the operating cycles for individual machines. To develop such controllers, industrial practice is primarily concerned with control-based specifications rather than plant-based specifications. These control specifications are commonly given by means of high-level specification models, such as Grafcet, that provide a straightforward means to describe the required control tasks and capture the concurrency (of actions and transitions), synchronisation and the possibility of using events, conditions and complex logic operators. Progress-type automata specifications, on the other hand, are inconvenient for capturing these aspects because they are only concerned with the sequential aspects related to individual task. Furthermore, a systematic way of choosing the possible controllable events from among the ones allowed by the maximally permissive supervisor corresponding to a number of progress specifications cannot be guaranteed to correspond to the best choice available for each particular case. For these reasons, it seems to be more interesting to use Grafcet to specify the "unconstrained" control specifications and to propose a synthesis procedure that generates an execution model corresponding to the minimally restrictive behaviour of Grafcet that reinforces the specified constraints.

A control synthesis approach for Grafcet in the frame of the supervisory control theory has been proposed in [8]. Despite its successful application to an industrial system (with 77 partial Grafcets, and 200 inputs and outputs), the application of this approach is restricted to a sub-class of Grafcet implying a number of structural restrictions, and limited to the use of impulse-type orders and receptivities given in terms of simple events. The underlying semantics is based on the hypothesis that a Grafcet order and its corresponding plant reaction are considered as an atomic event, implying that the reaction can only be intercepted by the event associated to the downstream transition of the step associated with the order.

This hypothesis is rather restrictive because the controller in real systems usually performs a number of parallel orders and an order may be maintained through a number of consecutive Grafcet steps or issued by a number of parallel steps. Furthermore, the plant may induce complex interleaving causality relationships between control actions and their consequent reactions. The discussion in Section 3 shows that these relationships can only be captured using an explicit model of the plant.

A general synthesis framework will, therefore, require the use of: (i) a behavioural model that considers all Grafcet features without any particular restrictions, (ii) an explicit model of the controlled plant, and (iii) a model expressing the constraints to be satisfied by the synthesised control implementation. Such a framework must use a dedicated intersection algorithm, which takes into account the semantic differences between these three models as well as the behavioural difference between the control model and resulting controller implementation. In [30], the authors have provided a gradual and informal presentation of the concepts and issues that should be considered to implement such an intersection procedure. This previous paper has also highlighted the necessity of combining an analysis and a synthesis phase to provide user-guidance for analysis and correction of design errors for sizeable real-world applications. A direct dedicated intersection algorithm between the three models taking into account their semantic difference and the specificity of their interaction is likely to provide the most efficient solution for implementing such an analysis–synthesis framework in terms of algorithmic complexity. However, the design of such an algorithm is not a trivial task due to the semantic difference between the Grafcet model (which is based on conditions, events, logic operators, double time-scale interpretation, synchronism, reactivity, possibility of simultaneous transition firings [20]) and the formal aspects related to the required models of the plant and the constraints. To conquer this algorithmic difficulty, this paper proposes a multi-step synthesis framework that builds on well-established works related to Grafcet [23] and to the supervisory control theory. The aim of this framework is to synthesise an optimal and deadlock-free control implementation; optimality is to be understood here in the sense that the resulting control implementation represents the minimum possible restriction of the behaviour of a given Grafcet, and satisfies the required safety and liveness properties. This synthesis framework (Fig. 13) is based on the use of a supervisor and a controller; the supervisor enforces the given safety and liveness specifications, whereas the controller directs the system toward the desired goal, to accomplish a specific set of tasks. The controller is given in terms of the automaton of stable situations, ASS, corresponding to the specified Grafcet [23]. For the parts-handling example, the first elements of ASS are depicted in Fig. 5.

The supervisor is synthesised on the basis of a plant model given by automata corresponding to spontaneous event generators [24]. The activation, $\uparrow Z$, and deactivation, $\downarrow Z$, of Grafcet orders correspond to controllable events, Σ_c, because it is possible to prevent their occurrence by an appropriate conditioning

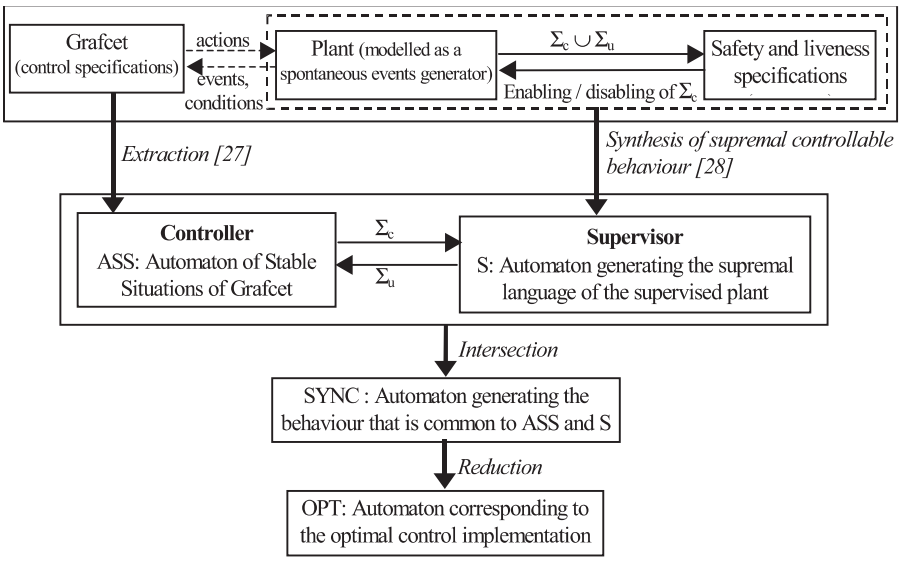


Fig. 13. Synthesis framework.

of Grafcet orders. Uncontrollable events, Σ_u, are initiated by the plant. These events, which cannot be disabled by control action, are associated with the rising edges, $\uparrow E$, and the falling edges, $\downarrow E$, of Grafcet inputs. This interpretation can be considered as an adaptation of Balemi's scheme [2] to the case of Grafcet.

Based on the automata representing the plant as well as automata specifying the required safety and liveness constraints, the supervisor is obtained by applying the synthesis algorithm proposed in [17], which is given in the annex. The resulting supervisor realisation is that of a discrete event system $S = (\Sigma, Q, \Delta, q_0)$, where Σ is a set of events, Q is the set of states, $\Delta : \Sigma \times Q \rightarrow Q$ is a partial function called the transition function, and q_0 is the initial state. The transition structure of S corresponds to the maximum non-blocking allowable behaviour of the controlled plant with respect to the given safety and liveness specifications. This means that the part of Grafcet that will be allowed to execute should be confined within the language that can be generated by S.

For the given example, the automaton S, generated on the basis of the simplified plant model (given in Section 3.3.2), has 68840 states and 513696 transitions. Some of these elements are depicted in Fig. 14. This automaton gives the way in which the different evolutions of the plant automata (Figs 9(a)–(c) and 10) are ordered to comply with the specified constraints. For example, the first constraint to satisfy is related to the prohibition of the platform's rotation during the deposit of a part, implying that the order DOWN should not be activated unless the order TURN_*ac* is deactivated, and vice versa. Figure 14 shows that the states q_{3154} and q_{3105}, by virtue of their input and output transitions, ensure the satisfaction of this constraint. The liveness constraint, c3, which requires the deactivation of the order MOVE as long as the detected part is not retrieved from the conveyor, $\downarrow p\_part$, is also satisfied in the automaton S. Take the situation q_{193}, which is reached from the initial situation further to the occurrence of the sequence of events: activation of the order MOVE, arrival of a part, and then the activation of the order LEFT (either before or after) the deactivation of the order MOVE. This state, q_{193}, does not admit the reactivation of the order MOVE; i.e., it has no output transition labelled with ↑MOVE. This is also the case for state q_{88} and all the other states in S where the order MOVE has been deactivated previously, and the part is not yet retrieved. Therefore, $c3$ is satisfied with respect to S.

The sequence of events that can be generated both by the controller, ASS, and by the supervisor, S, are extracted by a dedicated intersection algorithm presented in Section 4.1. This algorithm results in an event-based automaton, SYNC, which is then treated by a reduction algorithm to remove the deadlocks of SYNC and to generate the control-execution model. This reduction algorithm, given in Section 4.2, is used to generate the automaton OPT representing the most permissible subset of Grafcet behaviour that is deadlock-free, and satisfies the imposed constraints.

4.1. Intersection

The aim of this step is to generate the automaton SYNC representing the behaviour common to the automata S and ASS, by retaining the evolutions of the control model, ASS, that are authorised by the supervisor, S. The intersection principle is rather specific due to the semantic differences between the automata ASS and S. Each state in ASS is associated with the orders of the corresponding situation of Grafcet, and the transitions of ASS are given in terms of logical expressions combining input variables and their edges. On the other hand, S is an elementary automaton whose transitions correspond to simple events. The proposed intersection algorithm goes

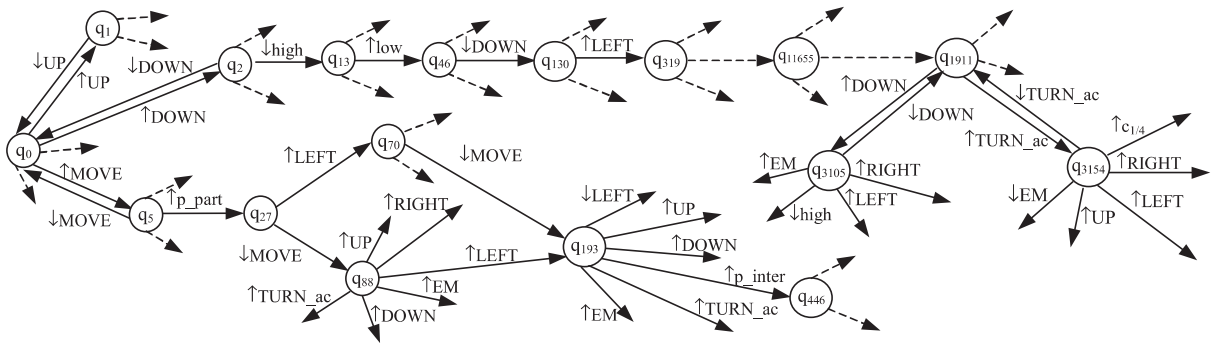


Fig. 14. An extract of the supervisor automaton, S.

through the events that can be accepted by ASS and S to construct the automaton SYNC. A transition labelled with a controllable event is included in SYNC when S accepts this event, provided that the event is not in contradiction with the current output state of ASS. In this way, only the control evolutions that are acceptable by the supervisor are retained in SYNC. On the other hand, since uncontrollable events cannot be inhibited, transitions labelled with uncontrollable events are added to SYNC whenever S can generate these events, irrespective of whether ASS can accept them.

The transitions of the automaton SYNC correspond to simple events. The orders to be simultaneously performed in a given Grafcet situation are represented in SYNC by means of sequences of interleaved transitions corresponding, each, to the activation or the deactivation of one of these parallel orders. The states relating these transitions are grouped in a region which corresponds to the given Grafcet situation (Fig. 15). The intra-region transitions correspond to controllable events and the inter-region transitions represent uncontrollable events. The interleaving of parallel orders (inside a region) is used here to provide an event-based semantics to SYNC in view of simplifying the formal development of the intersection procedure. The reduction step, which is presented in Section 4.2, will be used to reconstruct the execution model by aggregating each region into a single state associated with the set of orders to perform in parallel.

Formally, the automaton SYNC is defined by the 6-tuple $(\Sigma, \mathrm{ST}, \mathrm{TR}, st_0, r_0, \mathrm{BLOC})$, where,

- $\Sigma = \Sigma_c \cup \Sigma_u$; $\Sigma_c = \uparrow Z \cup \downarrow Z$ and $\Sigma_u = \uparrow E \cup \downarrow E$. In the following, $\uparrow z$ corresponds to an element of $\uparrow Z$, $\downarrow z$ to an element of $\downarrow Z$, $\uparrow e$ to an element of $\uparrow E$, and $\downarrow e$ to an element of $\downarrow E$.
- ST is the set of states of SYNC, which are partitioned into sub-sets that are called regions. A state $st \in \mathrm{ST}$ corresponds to a unique configuration of the 3-tuple (q, y, E), where q is a state of S, y a state of ASS, and E is the set of logic valuations (0 or 1) of Grafcet inputs. All the states corresponding to a situation of Grafcet and to a unique valuation of its inputs, belong to one region.
- $\mathrm{TR} : \mathrm{ST} \times \Sigma \rightarrow \mathrm{ST}$ is a partial function representing the transitions of SYNC. A transition is defined by a 3-tuple $(st, \sigma, st') \in \mathrm{TR}$.
- $st_0 \in \mathrm{ST}$ is the initial state, given by (q_0, y_0, E_0), where q_0 is the initial state of S, y_0 the initial state of ASS, and E_0 corresponds to the set of initial valuations of Grafcet inputs.
- $r_0 \subset \mathrm{ST}$ is the initial region corresponding to the initial situation, y_0, of Grafcet and to the initial valuation, E_0, of its input variables.
- $\mathrm{BLOC} \subset \mathrm{ST}$ is the set of blocking states, i.e. the states having no output transitions.

4.1.1. Intersection Algorithm

The automaton SYNC is generated by means of the following iterative algorithm that terminates when no new region can be created anymore:

Step 1: Develop the initial region r_0 from the state st_0.
Step 2: For each developed region (by step 1 or step 2-b):
  2-a: create the first state of each of its output regions,
  2-b: develop these output regions.

This algorithm receives as an input the automaton $S = (\Sigma, Q, \Delta, q_0)$, corresponding to the supervisor generated by the synthesis step, the automaton $\mathrm{ASS} = (E, Z, Y, T, y_0)$, equivalent to Grafcet, and the initial structure of the automaton SYNC, given by $(\Sigma, \{st_0\}, \emptyset, st_0, \{st_0\}, \emptyset)$. In the following, the current state of SYNC is characterised by the current state of S, the current situation of ASS, as well as the current valuation of the input variables.

The iterative procedure corresponding to step 1 is given in Fig. 16. This procedure creates the states and the transitions of the region r_0. The created transitions correspond to the controllable events, which are enabled by the supervisor and non-contradictory with the outputs associated with the situation of Grafcet.

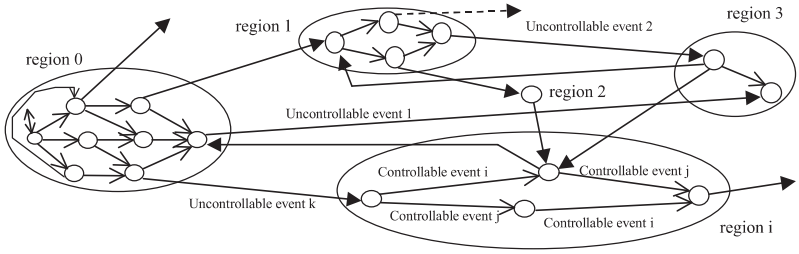


Fig. 15. Semantic model of the automaton SYNC.

1) $\forall(q, y_0, E_0) \in r_0, \forall(q, \Sigma, q') \in \Delta$ such that $(\Sigma = \uparrow z \wedge z \in Z_{y0})$:
   i) $ST = ST \cup \{ (q', y_0, E_0) \}$;
   ii) $TR = TR \cup \{ ((q, y_0, E_0), \Sigma, (q', y_0, E_0)) \}$;
   iii) $r_0 = r_0 \cup \{ (q', y_0, E_0) \}$;
2) $\forall(q, y_0, E_0) \in r_0, \forall(q, \Sigma, q') \in \Delta$ such that $(\Sigma = \downarrow z \wedge z \notin Z_{y0})$:
   i) $ST = ST \cup \{ (q', y_0, E_0) \}$;
   ii) $TR = TR \cup \{ ((q, y_0, E_0), \Sigma, (q', y_0, E_0)) \}$;
   iii) $r_0 = r_0 \cup \{ (q', y_0, E_0) \}$;

Fig. 16. Development of the initial region r_0.

1) $\forall(q, y_c, E_c) \in r_c, \forall(q, \Sigma, q') \in \Delta$ such that $(\Sigma = \uparrow z \wedge z \in Z_{yc})$:
   - If $((q', y_c, E_c) \in ST \wedge (q', y_c, E_c) \in r_e \wedge r_e \neq r_c)$
     Then i) $r_c = r_e \cup r_c$; ii) eliminate(r_e)
   - If $((q', y_c, E_c) \notin ST)$ Then i) $ST = ST \cup \{ (q', y_c, E_c) \}$; ii) $r_c = r_c \cup \{ (q', y_c, E_c) \}$;
   - $TR = TR \cup \{ ((q, y_c, E_c), \Sigma, (q', y_c, E_c)) \}$;
2) $\forall(q, y_c, E_c) \in r_c, \forall(q, \Sigma, q') \in \Delta$ such that $(\Sigma = \downarrow z \wedge z \notin Z_{yc})$:
   - If $((q', y_c, E_c) \in ST \wedge (q', y_c, E_c) \in r_e \wedge r_e \neq r_c)$
     Then i) $r_c = r_e \cup r_c$; ii) eliminate(r_e)
   - If $((q', y_c, E_c) \notin ST)$ Then i) $ST = ST \cup \{ (q', y_c, E_c) \}$; ii) $r_c = r_c \cup \{ (q', y_c, E_c) \}$;
   - $TR = TR \cup \{ ((q, y_c, E_c), \Sigma, (q', y_c, E_c)) \}$;

Fig. 17. Development of a region r_c.

The first iteration of the procedure creates the controllable output transitions of st_0 as well as their downstream states, which are added to the initial region. Each of the following iterations continues the development of the region r_0, starting from one of the states created by a previous iteration, until all the states of the region are treated. An iteration consists of testing the controllable output transitions of the current state, q, of S. If such a possible transition of S corresponds to the activation of a Grafcet order, $\uparrow z$, that is associated to the initial situation of Grafcet ($z \in Zy_0$), sub-step 1, creates a new state in ST and adds the transition leading to this state to TR. The created state, (q', y_0, E_0), is distinguished from the input state of the added transition by the first element of the 3-tuple, q', corresponding to the state of S after executing the transition. The state (q', y_0, E_0) is added to r_0 if it was not already included in this region. A similar treatment is applied in sub-step 2 for the controllable events corresponding to the deactivation of Grafcet orders, $\downarrow z$: if the corresponding order, z, is not included in the set of orders associated to the initial state of ASS, then a transition corresponding to $\downarrow z$ is created in r_0 to indicate that this order is inhibited during the initial situation of the controller.

Step 2-b of the intersection algorithm aims at developing a newly created region, r_c, starting from its first state, generated by step 2-a. The corresponding procedure (Fig. 17) is similar to the one used to develop the initial region in step 1, i.e. the controllable transitions of r_c are created together with their output states if these states were not already created by a previous iteration of this step. However, the output states of the created controllable transitions may belong to an already existing region, r_e, that is different from r_c. In this case, the two regions, r_e and r_c, are merged in r_c because they correspond to the same situation of Grafcet with a unique valuation of its inputs, and the region r_e is eliminated from the partition of the subsets of ST by using the operator *eliminate*(r_e).

Step 2-a uses the procedure given in Fig. 18 to create the inter-region transitions, which represent the uncontrollable output events of a recently created region, r_r, i.e. the possible reactions of the plant. Each iteration of this procedure is composed of two substeps to create the uncontrollable transitions leaving one of the states st of the region r_r. The first step deals with the transitions representing the uncontrollable events that are accepted in the current state, q, of the

$\forall st = (q, y, E) \in r_r$ Do:
1- $\forall$ (q, $\uparrow$e, q') $\in \Delta$ such that (e$\in$E $\wedge$ e=0) Do:
- If ($\exists$ ((y, f(E), y') $\in$ T) such that f(E)=true) Then st'=(q',y'',E') $\wedge$ (y''=y') Else st'=(q',y'',E') $\wedge$ (y''=y); where the valuation E' differs from E only w.r.t the value of e which becomes 1 instead of 0;
- TR = TR $\cup$ { (st, $\uparrow$e, st') };
- If st'$\notin$ ST Then
i) ST = ST $\cup$ {st'},
ii) create the partition r = {st'} $\subset$ ST,

2- $\forall$ (q, $\downarrow$e, q') $\in \Delta$ such that (e$\in$E $\wedge$ e=1)) Do:
- If ($\exists$ ((y, f(E), y') $\in$ T) such that f(E)=true) Then st'=(q',y'',E') $\wedge$ (y''=y') Else st'=(q',y'',E') $\wedge$ (y''=y); where the valuation E' differs from E only w.r.t the value of e which becomes 0 instead of 1;
- TR = TR $\cup$ { (st, $\downarrow$e, st') };
- If st'$\notin$ ST Then
i) ST = ST $\cup$ {st'},
ii) create the partition r = {st'} $\subset$ ST,

Fig. 18. Creation of a new region.

supervisor (i.e. a possible reaction of the plant) and correspond to the rising edge of an input variable of Grafcet $\uparrow e$. Each of these events is admitted in the common behaviour, SYNC, if the current value of the related variable $e \in E$ in the ASS is 0, i.e. if the rising edge of this variable can effectively take place. The corresponding transition created in SYNC leads to a state st', where the value of e becomes 1. This state corresponds to the state q' of S following the occurrence of $\uparrow e$, and to the situation y of ASS. If the event $\uparrow e$ entails the validation of a logical expression, $f(E)$, associated with an output transition of the current situation, y, of ASS, then y corresponds to the downstream state of this transition. Otherwise, $y = y$. If the state st' does not already exist in ST, then it will be created and included in a new region. The second sub-step of this procedure is based on a similar principle to deal with the uncontrollable transitions corresponding to the falling edges $\downarrow e$ of the input variables of Grafcet. These transitions are added to ST if the current value of the variable e is 1. In this case, the value of this variable becomes 0.

4.1.2. Intersection Results for the Example

The automata S and ASS of the parts-handling example will be used now to illustrate the application of the intersection algorithm (see Fig. 19). The set of Grafcet inputs is given by $E = \{p\_part, high, low, p\_conv, p\_inter, p\_ext, p\_int, c_{1/4}\}$. Step 1 of the intersection algorithm is related to the development of the initial region, r_0, starting from the initial state (q_0, y_0, E_0), where $E_0 = \{0, 1, 0, 0, 0, 0, 0, 1\}$ and $Z_{y0} = \{\text{MOVE}\}$. This value of Z_{y0} implies that $\uparrow$MOVE is the only controllable event occurring in the current situation, y_0, of ASS. Since this event is enabled by the current state, q_0, of the supervisor, r_0 will include one transition corresponding to the occurrence of $\uparrow$MOVE and leading to a newly created state, (q_5, y_0, E_0). Next, step 2-a is applied to create the inter-region uncontrollable output transitions of states (q_0, y_0, E_0) and (q_5, y_0, E_0). All the output transitions of q_0 correspond to controllable events, and hence there is no uncontrollable output transition originating from state (q_0, y_0, E_0). On the other hand, one of the eight output transitions of q_5 is associated with an uncontrollable event, $\uparrow p\_part$. Since the value of the variable p_part is 0 at y_0, the rising edge, $\uparrow p\_part$, can occur at (q_5, y_0, E_0). This event validates the logical expression of the transition relating y_0 to y_2 in ASS and sets the value of p_part to 1. Hence, the initial state of the newly created region, r_1, is given by (q_{27}, y_2, E_1) with $Z_{y2} = \{\text{LEFT}\}$ and $E_1 = \{1, 1, 0, 0, 0, 0, 0, 1\}$. Since the events $\downarrow$MOVE and $\uparrow$LEFT are enabled by the current state, q_{27}, of S, and since MOVE $\notin Z_{y2}$ while LEFT $\in Z_{y2}$, step 2-b of the intersection algorithm will generate the interleaving sequence of controllable transitions ($\downarrow$MOVE then $\uparrow$LEFT, or $\uparrow$LEFT then $\downarrow$MOVE) as well as the destination states – (q_{88}, y_2, E_1), (q_{70}, y_2, E_1) and (q_{193}, y_2, E_1) – which are included in r_1. Step 2-a is applied next to determine the uncontrollable output transitions of each of the states of r_1. These transitions, representing the enabled transitions of the corresponding states of S, are identified in Fig. 19 by (2: $\downarrow p\_part$), (3: $\downarrow p\_part$), (4: $\uparrow p\_inter$) and (5: $\uparrow p\_inter$). Transition 2 (respectively, 3) entails the activation of state q_{87} (respectively, q_{192}) in S, of situation y_1 in ASS, and the preset of the value of $p\_part \in E$ to 0. Since the transitions 2 and 3 lead to two states in SYNC corresponding to the same

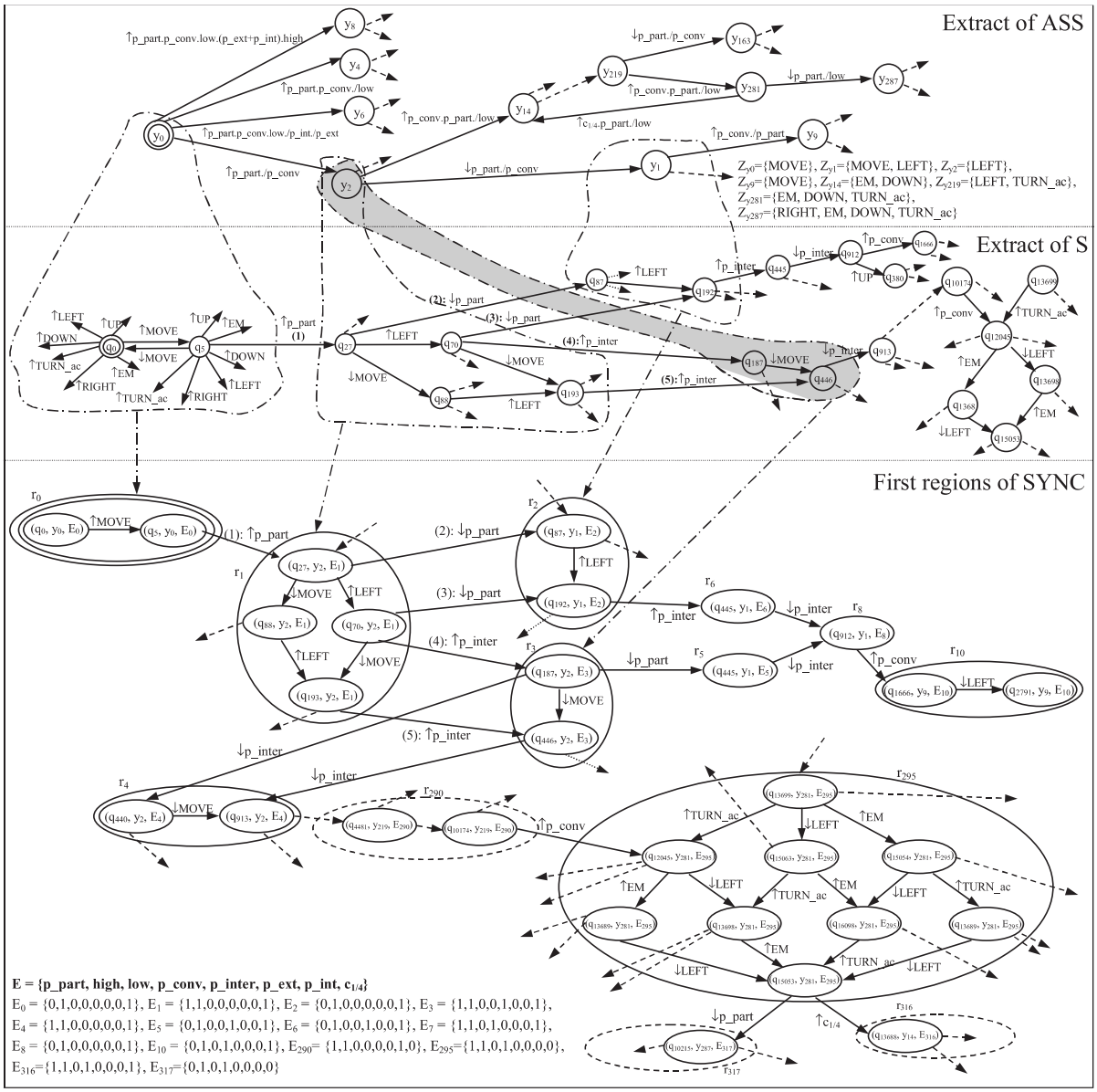


Fig. 19. Illustration of the intersection algorithm.

situation, y_1, of ASS with the same valuation for the input vector, E_2, the two output states will be grouped into one region, r_2. These states are related by the controllable transition, ↑LEFT, because this event is enabled in q_{87} and $\text{LEFT} \in Z_{y1}$ (step 2-b). The occurrence of ↑*p_inter* in r_1 corresponds to an evolution of the automaton S and of the input vector, but does not entail an evolution of the ASS situation. The corresponding transitions, 4 and 5, lead, respectively, to states (q_{187}, y_2, E_3) and (q_{446}, y_2, E_3), which are grouped in the region r_3 because they correspond to the same situation of ASS and the same valuation of E. The other elements of the automaton SYNC are constructed in the same way.

The resulting automaton, SYNC, has 9016 transitions and 3812 states that are grouped in 524 regions. The first elements of this automaton are depicted in Fig. 19. Some regions, such as r_6 and r_8, only include a single state because they correspond to a situation where no new order should be activated or deactivated. For the regions comprising two states, such as r_2 and r_3, only one order is activated or deactivated. The regions with many states, such as r_{295}, correspond to situations where many orders are to be activated and/or deactivated. Note that the state (q_{2791}, y_9, E_{10}) of region r_{10} represents a deadlock situation, because it has no output transitions.

After deposing a type-2 part on the inward position of the platform, the carriage must return to the loading area to wait for a new part of type 1. The region r_{295} illustrates the incidence of the intersection procedure when the carriage reaches the loading zone, $\uparrow p\_conv$. This region corresponds to Grafcet situation y_{281}, whose orders, $Z_{281} = \{\text{EM}, \text{DOWN}, \text{TURN\_ac}\}$, imply the descent of the electromagnet to pick up a new part of type 1 while the platform is positioned to receive the next part. To satisfy the constraint c1, the order DOWN should be prohibited in this case, and consequently, the region does not include any controllable transition related to the activation of the descent, $\uparrow$DOWN.

4.2. Reduction

The reduction step, which involves two phases (Treatment of the blocking states and Aggregation), is used to generate the maximum, reactive and non-blocking execution controller. The role of the first phase is to reduce SYNC by removing its deadlock states as well as the evolutions leading to these deadlocks in real execution. This phase also removes the transitions of SYNC representing the occurrence of uncontrollable events at the non-terminating states of all the regions. In fact, since the interleaving orders of a region are to be performed simultaneously in real execution, the execution of an interleaving sequence of orders can be regarded as occurring at the same time instant, and hence the plant reaction can only occur after the execution of all the transitions of the interleaving sequence. Therefore, only the uncontrollable transitions leaving the terminal states of a region need to be maintained. The second phase is applied next to aggregate each region into a single state, in view of obtaining the automaton OPT, representing the execution control model. Each situation of OPT is associated with the orders to be activated and deactivated when the situation is active. The states of OPT are connected by uncontrollable transitions relating the corresponding regions of SYNC. Hence, starting from a situation, the occurrence of a plant event drives the automaton immediately to a unique situation, and results in an instantaneous activation and deactivation of the orders associated to the situation. The resulting automaton, OPT, gives the largest admissible non-blocking behaviour of the controller with respect to the specified constraints.

4.2.1. Treatment of the Blocking States

The first phase consists in removing the blocking states (states with no output transitions) from SYNC and trimming their upstream transitions. It is also used to guarantee the non-reachability of the trimmed uncontrollable transitions in real execution. The corresponding algorithm is given in Fig. 20.

1) $\forall$ st $\in$ ST, If ($\nexists$ (st, σ, st') $\in$ TR) Then BLOC = BLOC $\cup$ {st}
2) $\forall$ st $\in$ BLOC Do:
  2.1)$\forall$ t = (st', σ, st) $\in$ TR Do:
    a) TR = TR - {t}
    b) If $\sigma \in \Sigma_u$ Then:
      - clean(st')
      - If ($\forall$ state' $\in$ reg_{st} : $\nexists$ (state, τ, state') $\in$ TR such that $\tau \in \Sigma_u$) Then remove(reg_{st})
    c) If $\sigma = \uparrow z \in \uparrow Z$ Then $\forall$ state $\in$ reg_{st} Do:
      - If $\exists$ (state, σ, state') $\in$ TR Then:
        - TR=TR-{ (state, σ, state') }
        - If ($\nexists$ (st'', τ, state') $\in$ TR) Then trim(state')
        - If ($\nexists$ (state, τ, state'') $\in$ TR) Then BLOC = BLOC $\cup$ {state}
    d) If $\sigma \in \downarrow Z$ Then:
      - $\forall$ (state, τ, state') such that state' $\in$ reg_{st}: TR = TR-{ (state, τ, state') }; clean(state)
      - remove(reg_{st})
  2.2) ST = ST - {st} , BLOC = BLOC - {st}
3) $\forall$ (st, α, st'') $\in$ TR such that $\alpha \in \Sigma_u$,
  If ($\exists$ (st, σ, st') $\in$ TR such that $\sigma \in \Sigma_c$) Then:
    - TR = TR - { (st, α, st'') };
    - If ($\forall$ state $\in$ $reg_{st''}$: $\nexists$ (state', τ, state) $\in$ TR such that $\tau \in \Sigma_u$) Then remove($reg_{st''}$)
      Else If ($\nexists$ (state'', τ, st'') $\in$ TR) Then trim(st'')

Fig. 20. Deadlock-treatment algorithm.

The first step of the algorithm starts by adding the states of ST with no output transitions to the list of blocking states of SYNC, BLOC. In step 2.1, the input transitions of each blocking state, $st \in$ BLOC, are removed to render these states unreachable. The inconsistencies, deadlocks and unreachable states produced by these removals are next examined and the corresponding treatment depends on the type of event related to the suppressed transition: uncontrollable event ($\sigma \in \Sigma_u$), activation of a Grafcet order ($\sigma \in \uparrow Z$) or inhibition of an order ($\sigma \in \downarrow Z$).

If the removed transition corresponds to an uncontrollable event, $\sigma \in \Sigma_u$, then step 2.1b is used to render the input state of the transition unreachable because such a transition represents a reaction of the plant that cannot be prohibited by the controller. The procedure *clean(state)*, which is described below, achieves this treatment and takes into consideration the deadlocks that may, in turn, be introduced by the removal of the transition. If, following the suppression of the transition, the region including the blocking state *st* becomes unreachable, i.e. if none of its states has uncontrollable input transitions anymore, then the procedure *remove(r)* is used to remove this region. The operator reg_{st} returns the region that includes the state *st*.

If the removed transition represents the activation of an order *z*, i.e. $\sigma = \uparrow z \in \uparrow Z$, then step 2.1c is used to ensure the prohibition of this order in the current situation of the controller. Consequently, all the transitions corresponding to the activation of this order should be removed from the region. If the removal of one of these transitions renders its output state unreachable, the procedure *trim(state)* is invoked to eliminate this state and to treat its output transitions. On the other hand, if an input state of a removed transition has no other output transitions, this state is added to the list BLOC.

If the removed transition corresponds to the deactivation of an order, $\sigma \in \downarrow Z$, then this implies that this order should not be deactivated as specified in the corresponding Grafcet situation. On the other hand, the synthesis procedure cannot impose performing such an order because this will be in contradiction with the Grafcet specifications. To alleviate this contradiction, and since all the interleaving sequence of events in a region are replaced with the parallel orders of the corresponding situation in the execution model, step 2.1d proceeds by removing the whole region under study. Before removing the region, the set of input transitions originating from other regions are eliminated, and the procedure *clean(state)* is invoked to treat the input states of these transitions so as to guarantee the non-reachability of the removed region in real execution.

After these different treatments, step 2.2 removes the blocking state *st* from the list ST and from the list of blocking states BLOC. Then, step 3 is used to remove the uncontrollable transitions leaving the non-terminating states of a region. These transitions, $\alpha \in \Sigma_u$, are removed if their input states have other output controllable transitions. The output regions and states that have become unreachable after these suppressions, are then removed by calling the procedures *remove(r)* and *trim(state)*, respectively.

The deadlock-treatment algorithm also uses the three auxiliary procedures: *remove(r)*, *trim(state)*, and *clean(state)*, which are given in Fig. 21.

The procedure *remove(r)* (Fig. 21(a)) is used to remove a region, *r*, having become unattainable from the automaton SYNC. First, all the states of this region are removed from the lists ST and BLOC. Then, the output transitions of these states are removed. If the removal of an uncontrollable transition renders its output region (or state) unreachable, this output region (or state) is also removed. Afterwards, the region *r* is eliminated from the partition of the sub-sets of ST by using the operator *eliminate(r)*.

The procedure *trim(state)*, given in Fig. 21(b), removes a state that has either become unreachable, or should be rendered unreachable. It proceeds by removing this state from the lists ST and BLOC, and then removing its output transitions. If the output regions and states of these transitions become unreachable, then they are also removed.

The procedure *clean(state)* (Fig. 21(c)) is used to treat the input state of an uncontrollable transition to be removed. In the first place, the state and its output transitions are removed by calling the procedure *trim(state)*. Then, the input transitions of this state are removed. If one of these transitions is uncontrollable, the procedure is applied recursively to its input state. In this way, all the sequences of uncontrolled transitions leading to the removed state will be eliminated to guarantee the non-reachability of the removed state in real execution. Finally, the states having become blocked by the suppression of these transitions are added to the list BLOC so that it can be treated in the next iteration of the deadlock treatment algorithm (Fig. 20).

The application of the intersection algorithm to the automaton SYNC of Fig. 19 results in a reduced automaton whose first evolutions are depicted in Fig. 22. This figure shows that the blocking region, r_{10}, of Fig. 19 has been removed together with its input regions r_8, r_6, r_5 and r_2. Note also that the region r_2 has been removed by the deadlock-treatment algorithm

a) *remove(r)*

$\forall$ st $\in$ r Do:
- ST = ST - {st}
- BLOC = BLOC - {st}
- $\forall$ t = (st, σ, st') $\in$ TR: Do
  - TR = TR- {t},
  - If $\sigma \in \Sigma_u$ Then
    - If ($\forall$ state $\in$ $reg_{st'}$: $\nexists$ (state', τ, state) $\in$ TR such that $\tau \in \Sigma_u$) Then remove($reg_{st'}$)
    - Else If ($\nexists$ (state'', τ, st') $\in$ TR) Then trim(st')

eliminate(r)

b) *trim(state)*

- ST = ST - {state}
- BLOC = BLOC - {state}
- $\forall$ t = (state, σ, state') $\in$ TR Do:
  - TR = TR - {t},
  - If $\sigma \in \Sigma_u \wedge$ ($\forall$ st $\in$ $reg_{state'}$: $\nexists$ (st', τ, st) $\in$ TR such that $\tau \in \Sigma_u$) Then remove($reg_{state'}$)
  - If ($\nexists$ (state'', τ, state') $\in$ TR) Then trim(state')

c) *clean(state)*

- trim(state)
- $\forall$ t = (state'', σ, state) $\in$ TR Do:
  - TR = TR - {t}
  - If $\sigma \in \Sigma_u$ Then clean(state'') Else If $\nexists$ (state'', τ, st) $\in$ TR Then BLOC = BLOC $\cup$ {state''}

Fig. 21. Auxiliary procedures.

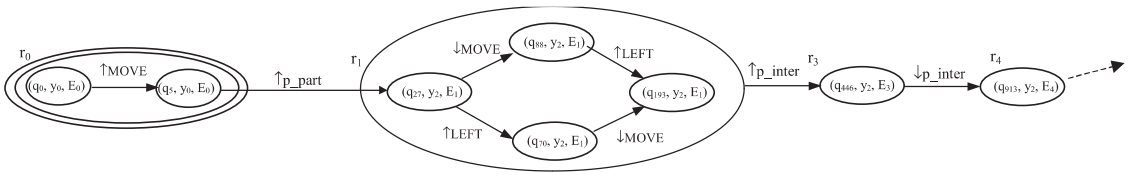


Fig. 22. Automaton SYNC generated by the deadlock-treatment algorithm.

and that the regions r_3 and r_4 only include a single state after the reduction.

4.2.2. Aggregation

The previous phase results in a reduced automaton having the same structure as that of SYNC, but without deadlocks or unreachable situations. The aggregation phase consists of generating the automaton OPT, whose states correspond to the regions of the reduced SYNC. The automaton OPT is given by the 4-tuplet (Σ_u, ET, ϕ, $state_0$), where,

- Σ_u is the set of uncontrollable events;
- ET is the set of states corresponding, each, to a region of SYNC. The 4-tuple (ACT_c, NAC_c, Z_c, Sit_c) is associated to each state $c \in ET$, where,
  - ACT_c is the set of orders to activate in state c,
  - NAC_c is the set of orders to deactivate in state c,
  - Z_c is the set of Grafcet outputs corresponding to state c, i.e. $Z_c = Z_{yc}$, where y_c is the Grafcet situation corresponding to region r_c of the automaton SYNC;
  - Sit_c is the set of Grafcet steps corresponding to *state* c, i.e. $Sit_c = Step_{y_c}$, where y_c is the Grafcet situation corresponding to region r_c of the automaton SYNC;
- $\phi : \Sigma_u \times ET \rightarrow ET$ is a partial function representing the transitions of OPT.
- $state_0$ is the initial state.

The aggregation procedure is given in Fig. 23. Each region, r_c, is aggregated to a state, c, representing the corresponding situation of the controller. The states

- $ET = \emptyset$, $\phi = \emptyset$, $state_0 = 0$;
- $\forall r_c \subset ST$ Do:
  - i) $ET = ET \cup \{c\}$;
  - ii) $Sit_c = Step_{yc}$;
  - iii) $Z_c = Z_{rc}$;
  - iv) $\forall$ state $\in r_c$, $\forall$ (state, $\uparrow$z, state ') $\in$ TR : $ACT_c = ACT_c \cup z$;
  - v) $\forall$ state $\in r_c$, $\forall$ (state, $\downarrow$z, state '') $\in$ TR : $NAC_c = NAC_c \cup z$;
- $\forall r_c, r_{c'} \subset ST^2$, If $\exists$ (state, α, state ') $\in$ TR such that (state $\in r_c \wedge$ state ' $\in r_{c'}$) Then $\phi = \phi \cup \{ (c, \alpha, c') \}$

Fig. 23. Generation of the automaton OPT.

of OPT are related by uncontrollable transitions relating the regions of SYNC. Hence, starting from a given state, the occurrence of an event of the plant implies an evolution of the automaton to a unique state, and results in an instantaneous activation and/or deactivation of the orders associated to this state. The automaton OPT therefore represents the deterministic and reactive behaviour corresponding to the largest admissible behaviour of the controller, with respect to the specified constraints.

The resulting supervisory controller OPT can be implemented as an underlying optimal execution motor for the specified Grafcet, subject to the supervisory specifications.

During control execution, the orders to output to the plant, in a given state of OPT, are determined by the sets ACT and NAC, together with the orders performed in the previous state. Indeed, and by construction, the sets ACT and NAC of the automaton OPT are associated, respectively, with the orders to activate and deactivate relatively to the previous orders of the controller. Consequently, the orders to perform by OPT are given by the set ORD which is updated during execution. This set is initialised to ACT_0. When an event occurs in the plant, the automaton OPT advances to a new state, c, where ORD is calculated by the expression: $ORD = (ORD \cup ACT_c) - NAC_c$. In the same way, the orders that are prohibited by the synthesis procedure are those associated with the current situation of Grafcet but which are not performed in real execution. These orders, which are also calculated dynamically, are given by the set $PROH = Z_c - ORD$. The sets, ORD and PROH, are also used to allow the designer to distinguish those outputs which are maintained or restricted by the synthesis process.

4.2.3. Results

For the parts-handling example, the automaton OPT, has 2287 states and 6311 transitions. Some illustrative elements of this automaton are depicted in Fig. 24, where the first evolutions (states: 0, 1, 3 and 4) correspond to the four regions of the reduced SYNC automaton given in Fig. 22. The sets (ACT_c, NAC_c, Z_c, Sit_c) are associated with each state of OPT. The two states of the region r_0, corresponding to the initial situation of Grafcet, are aggregated into a unique state, $state_0$, with $ACT_0 = \{MOVE\}$, $NAC_0 = \emptyset$, $Z_0 = \{MOVE\}$ and $Sit_0 = \{100, 200, 300, 400\}$. In execution, the set of orders to perform in $state_0$ is given by $ORD = ACT_0 = \{MOVE\}$, and $PROH = Z_0 - ORD = \emptyset$; i.e. no order is prohibited in this state. In the same way, the four states of r_1 are aggregated into $state_1$ whose associated sets are: $ACT_1 = \{LEFT\}$, $NAC_1 = \{MOVE\}$, $Z_1 = \{LEFT\}$, and $Sit_1 = \{101, 200, 301, 400\}$. The orders to perform in this state are given by $ORD = (ORD \cup ACT_1) - NAC_1 = \{LEFT\}$, and no order is prohibited in this state because $PROH = Z_1 - ORD = \emptyset$. For the other states, the sets ORD and PROH are calculated in the same way.

Figure 24 also illustrates two corrections induced by the synthesis procedure. The first restriction imposed by the synthesis procedure is related to the constraint $c1$, which implies that the rotation of the platform should be prohibited during the positioning of a part on it (see the discussion related to region 295 of Fig. 19 in Section 4.1.2). Consider the situation $\{101, 200, 301, 401\}$ of Grafcet (corresponds to state 290 of OPT) in which the orders LEFT and TURN_*ac* are activated by virtue of the set ORD. As soon as the carriage attains the loading position above the conveyor, $\uparrow p\_conv$, Grafcet's situation $\{102, 201, 301, 401\}$ is activated ($state_{295}$), and the corresponding orders are given by the set, $Z_{295} = \{EM, DOWN, TURN\_ac\}$. However, the satisfaction of $c1$ implies that the orders TURN_*ac* and DOWN should not be activated simultaneously. The resulting correction induced by the synthesis procedure consists therefore of prohibiting the order DOWN in OPT during the rotation of the platform. Consequently, the orders to perform and to prohibit in state 295 are given, respectively, by

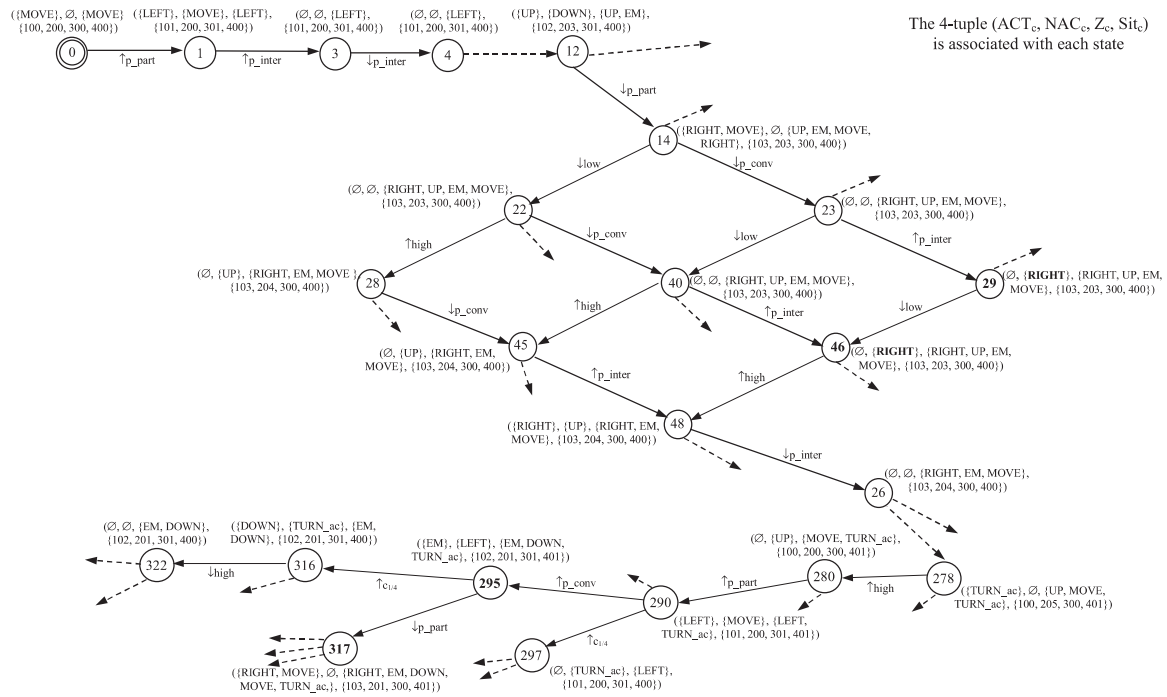


Fig. 24. Illustration of the restrictions induced by the synthesis procedure in execution.

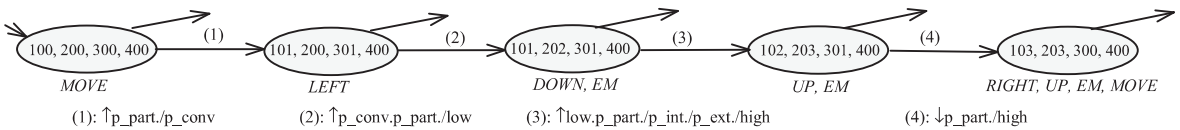


Fig. 25. Illustrative execution of ASS.

the sets: ORD = {EM, TURN_*ac*} and PROH = {DOWN}. This correction can also be observed in state 317 of Fig. 24, where the order DOWN is included in the set PROH.

A second correction is highlighted through Fig. 25, which shows a trajectory of the automaton of stable situations, ASS, that cannot be used to determine whether *c*2 is valid or not. In fact, in situation {102, 203, 301, 400}, the retrieval of a part from the loading area results in activating the situation {103, 203, 300,1400}. In this situation, the orders RIGHT, UP, EM and MOVE are simultaneously active, and this contradicts the safety constraint *c*2 if the carriage reaches the intermediate position, *p_inter*, before the upper position, *high*. However, if the carriage reaches the *high* position first, the ascent will be stopped beyond the intermediate position, *p_inter*, and the trajectory satisfies *c*2. Therefore, in view of the specified Grafcet, the automaton ASS cannot be used to conclude about the validity or invalidity of the property. The synthesis procedure solves this conflict by generating the necessary prohibitions to restrict the control executions to the runs that satisfy the constraint. In the resulting automaton, OPT (Fig. 24), the retrieval of a part of type 1 from the loading position activates the state 14, representing the situation {103, 203, 300, 400} of Grafcet which outputs the orders RIGHT, UP, EM and MOVE in view of placing the part on the outward position of the platform, *p_ext*. However, to satisfy the constraint *c*2 during this situation, the order RIGHT should be prohibited if the carriage attains the intermediate position, ↑*p_inter* before the upper position, ↑*high*. Hence, situation {103, 203, 300, 400} of Grafcet is split among the states 14, 23, 29, 22, 40 and 46 in the automaton OPT, with the order RIGHT being prohibited in states 29 and 46. Therefore, during the execution of these states, the set ORD is given by {UP, EM, MOVE} and PROH = {RIGHT}. On the other hand, if the high position, ↑*high*, is attained

before the intermediate position, $\uparrow p\_inter$, no correction would be necessary and the orders associated with the related steps (28, 45 and 48) will be consistent with the orders of the corresponding situation ($\{103, 204, 300, 400\}$) of Grafcet, i.e. $ORD = \{RIGHT, EM, MOVE\}$ and $PROH = \emptyset$.

In the same way, the specified constraints are respected throughout all the evolutions and the states of OPT.

The two cases above illustrate the corrections induced by the occurrence order of uncontrollable plant events. Although the intuitive explanation of the corrections seems to be rather simple, a straightforward application of common sense analysis methods cannot be easily used to highlight and correct the corresponding errors. In the first case, due to plant reactions, only one among the 16 situations of Grafcet which activates the actions DOWN and TURN_*ac* can occur in real execution, requiring the interdiction of the simultaneous execution of these two actions and the rescheduling of their execution. This observation cannot be captured by intuitive analysis of the Grafcet model. Such an analysis is far more complicated in the second case which requires the identification of all the situations in which the orders RIGHT and UP are performed simultaneously. In fact, common sense analysis in this case requires the formulation of some hypothesis about the occurrence order of the plant events *high* and *p_inter* because *p_inter* is not involved in any of the receptivities of the original Grafcet. Finally, intuitive analysis cannot guarantee the optimality of the corrections for complex examples, whereas the approach presented above provides an optimal rescheduling and correction of Grafcet actions to satisfy the imposed constraints which can be applied in a gradual manner.

4.2.4. Implementation and Complexity Issues

Formal synthesis approaches are not common practice to industrial users because they are difficult to understand. To alleviate this problem, the synthesis approach presented in this paper is based on the use of Grafcet, which is widely used in industry for the specification of logic controllers. Furthermore, the corrections induced by the proposed synthesis framework can be traced back to the Grafcet specification model in real execution so as to highlight and explain the corrections and the prohibitions induced by the synthesis framework. This use of Grafcet both for control specification and for highlighting the synthesis results aims at reinforcing the confidence of industrial users in formal synthesis approaches.

An implementation architecture and a feasibility prototype, built using C++, have been developed [35] to execute OPT and update the state of Grafcet correspondingly so as to visualise the execution of the control model and to highlight the prohibitions induced by the synthesis framework. This implementation (Fig. 26) is based on the use of a state/transition/action table, an input/output driver, a sequencer, and a module for update and animation of Grafcet. Each entry of the table corresponds to a state of the correct controller, OPT. It includes the following elements related to the state: the state index, a list of active Grafcet steps, the actions to perform and to prohibit; and the output transitions given in terms of the firing event, the associated logical expression, and the destination state. The input/output driver transmits the actions of the current situation to the plant and receives the plant inputs. An interrupt-driven architecture is used to guarantee reactivity; when a plant-originated event occurs, the sequencer goes through the events corresponding to the current state of the table to determine the interrupt source. The next state associated with this event becomes the new current state, and the actions to be performed are sent to the input/output driver. The update module provides a synthetic image of control execution, and allows the designer to easily recognise the control tasks to be maintained, restricted or inhibited by the synthesis approach.

This approach has been applied to a drilling machine [30], a robotic transfer system [22] as well as a number of educational test-beds and prototypes for which the size of the synthesised automaton, OPT, did not exceed 10000 states and 20000 transitions.

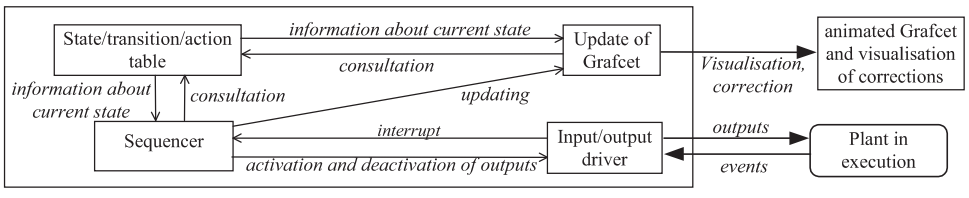


Fig. 26. Architecture of control implementation.

However, more complex systems require a very large memory, and, therefore, an alternative online synthesis approach has been proposed [22] to reduce the size of such an exhaustive model. The online synthesis proceeds by successive calculations of a partial control implementation model for a temporal window involving a number of anticipated future evolutions**.** This approach is based on the use of a partial intersection to generate the behaviour that is common to ASS and S for a number, N, of future evolutions. The blocking situations and evolutions of this behaviour are next removed by applying the reduction step which results in a partial automaton representing the anticipated control implementation model. As soon as an event occurs in the plant, partial intersection and reduction are executed again for one more step to update the partial control automaton.

The size of the automaton OPT is very small in this case compared to an exhaustive control implementation model. In fact, the worst-case-size of the automaton generated by the offline synthesis approach is proportional to $|\mathrm{ASS}| \times |\mathrm{S}| \times 2^{\Sigma_u/2}$, where $|\mathrm{ASS}|$ and $|\mathrm{S}|$ represent the number of states of the automaton ASS and S, respectively, whereas the worst-case-size of the automaton generated in the online case is proportional to $(|\Sigma_u|/2)^N$. Hence, for complex systems involving a large number of states and of inputs and outputs, the economy in the size of memory is significant. However, the partial intersection of the online synthesis approach only considers a limited temporal evolution window for the system. As a consequence the blocking situations which may arise beyond the current window cannot be avoided when the next partial intersections are calculated. If the size N of the window is increased, these blocking situations may be avoided, or at least be detected early enough to anticipate the required corrective actions. The choice of N should, therefore, be a compromise between the risk of deadlock and the size of the memory required to implement the partial optimal control automaton. Our current work, therefore, aims at implementing heuristics to choose the pertinent value for the number of required evolutions in the online case in such a way as to avoid deadlocks and to guarantee a maximum "acceptable" size of the partial state/transitions/actions table.

5. Conclusion

This paper has discussed the behavioural issues related to Grafcet, to emphasise the necessity of modelling the control plant in view of identifying the correct behaviour of Grafcet. Formal algorithms are also proposed for the synthesis of a correct controller implementation starting from high-level specifications given in Grafcet. The resulting controller represents the execution motor for Grafcet specifications, subject to a number of safety and liveness constraints. The proposed synthesis algorithm is based on the use of the supervisory control theory and automata to model the plant, whereas the behaviour of Grafcet is given in terms of the automaton of its reachable situations.

Current research is directed at enhancing the practicability of the proposed formal framework by proposing a dedicated assistance methodology for the modelling of the plant, and improving the performance of the developed software prototype. Another study is currently being undertaken to identify the complementary aspects between the fully automatic control synthesis framework proposed in this paper, and the semi-automatic Grafcet validation framework developed by the authors [28] in terms of the algorithmic complexity, and the type of properties and constraints to be considered. The aim of this study is to develop an assistance tool providing the control designer with pertinent information about the way of combining the validation and the synthesis framework on the basis of the structure of the models used for a given control system as well as the type of specified properties and constraints.

References

1. Arinez JF, Benhabib B, Smith KC, Brandin BA. Design of a PLC-based supervisory-control system for a manufacturing workcell. In: Proceedings of the Canadian High Technology Show and Conference. Toronto, Canada 1993
2. Balemi S, Hoffmann GJ, Gyugyi P, Wong-Toi H, Franklin GF. Supervisory control of a rapid thermal multiprocessor. IEEE Trans Autom Control 1993; 38: 1040–1059
3. Baracos P. Automation engineering in north America. In: Proceedings of the 2nd International Conference on Industrial Automation. Nancy, France 1995; pp 3–8
4. Brandin BA, Wonham WM, Benhabib B. Discrete event system supervisory control applied to the management of manufacturing workcells. VC Venkatesh and JA Mgeough (eds). Comp-Aided Prod Eng 1991; Elsevier, pp. 527–536
5. Carré-Ménétrier V, Ndjab C, Gellot F, Zaytoon J. A CASE tool for the synthesis of optimal control implementation of Grafcet, In: Proceedings 1999 IEEE International Conference on Systems, Man, and Cybernetics. Tokyo, Japan 1999; pp 796–801
6. Chandra V, Kumar R. A new modeling formalism and automata model generator for a class of discrete event

systems. In: Proceedings of the American Control Conference, Arlington 2001
7. Chandra V, Mohanty SR, Kumar R. Discrete event modeling and control of an assembly line built using LEGO Blocks. IEEE Trans Syst, Man Cybernet Part C 2000 (submitted)
8. Charbonnier F, Alla H, David R. The supervised control of discrete-event dynamic systems. IEEE Trans Control Syst Tech 1999; 7: 175–187
9. David R, Alla H. Petri nets and Grafcet: tools for modelling of discrete-event systems. Englewood Cliffs, NJ: Prentice-Hall 1992
10. David R. Grafcet: A powerful tool for specification of logic controllers. IEEE Trans Control Systems Technology 1995; 3: 253–268
11. Frachet JP, Colombari G. Elements for a semantics of the time in Grafcet and dynamic systems using non-standard analysis. Autom Control Prod Syst A.P.I.I. 1993; 27: 107–125
12. Giua A. Petri net techniques for supervisory control of discrete event systems. In: Proceedings of the 1st International Workshop on Manufacturing and Petri Nets, Osaka 1996
13. Holloway LE, Krogh BH. Synthesis of feedback control logic for a class of controlled Petri nets. IEEE Trans Autom Control 1990; 35: 514–523
14. International Electrotechnical Commission (IEC). Preparation of function charts for control systems. Publication 848, 1988
15. International Electrotechnical Commission (IEC). Preparation of function charts for control systems. Publication 848, 1991 (revised version)
16. Kheir NA, Astrom KJ, Auslander D, Checok KC, Franklin GF, Masten M, Rabins M. Control systems engineering education. Automatica 1996; 32: 147–166
17. Kumar R. Supervisory synthesis techniques for discrete event dynamical systems. Thesis for Ph.D. degree, University of Texas, 1991
18. Kumar R, Holloway LE. Supervisory control of Petri nets with regular specification languages. IEEE Trans Autom Control 1996, 41: 245–249
19. Lauzon SC, Ma AK, Mills JK, Benhabib B. An implementation methodology for the supervisory control of flexible manufacturing workcells. J Manuf Syst 1997; 16: 91–101
20. Lhoste P, Faure JM, Lesage JJ, Zaytoon J. Comportement temporel du Grafcet. Eur J Autom J.E.S.A. 1997; 31: 695–711 (in French)
21. Li Y, Wonham WM. Control of vector discrete-event systems I – the base model. IEEE Trans Autom Control 1993; 38: 1214–1227
22. Ndjab Hagbebell C. Synthèse de la commande des systèmes à événements discrets par Grafcet. Thesis for Ph.D. degree, Université de Reims Champagne Ardenne (France), 1999 (in French)
23. Ramadge PJ, Wonham WM. The control of discrete-event systems. Proceedings of the IEEE 1989; 77: 81–97
24. Roussel JM. Analyse de Grafcets par génération logique de l'automate équivalent. Thesis for Ph.D. degree, Ecole Normale supérieure de Cachan (France), 1994 (in French)
25. Roussel JM, Lesage JJ. Validation and verification of Grafcet using finite state machine. In: Proceedings CESA'96 IMACS/IEEE Symposium on Discrete Events & Manufacturing Systems. Lille, France 1996; pp 765–770
26. Union Technique d'Electricité (UTE). Function charts GRAFCET – extension of basic principles. Publication UTE C03-191, 1993
27. Yamalidou K, Moody JO, Lemmon MD, Antsaklis PJ. Feedback control of Petri Nets based on place invariants. Automatica 1996; 32
28. Zaytoon J. A contribution to the validation of Grafcet controlled systems. European Journal of Control 2000; 6: 488–506
29. Zaytoon J, Carré-Ménétrier V. Grafcet et graphes d'état: comportement, raffinement, vérification et validation. Eur J Autom J.E.S.A. 1999; 33: 751–782 (in French)
30. Zaytoon J, Carré-Ménétrier V. Synthesis of a correct control implementation for manufacturing systems. Int J Prod Res 2001; 39: 329–345
31. Zaytoon J, Lesage JJ, Marce L, Faure JM, Lhoste P. Vérification et validation du Grafcet. European Journal of Automation J.E.S.A. 1997; 31: 713–740 (in French)
32. Zaytoon J, Villermain Lecolier G. Two methods for the engineering of manufacturing systems. Control Eng Pract 1997; 5: 185–198
33. Zhou M, Dicesare F. Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Kluwer Academic Publishers, Boston 1993

ANNEX: Supervisory Synthesis Algorithm of Kumar

Among the different algorithms proposed for the synthesis procedure, the algorithm of Kumar [28] has been used to generate the supervisor. This algorithm, which is given in Fig. A, receives as inputs the automaton $G = (\Sigma, P, \delta, p_0)$ representing the global model of the plant, the automaton $T = (\Sigma, X, \xi, x_0)$ representing the global model of the constraints, and the initial structure of the resulting supervisor automaton, given by $(\Sigma, \{q_0\}, \Delta, q_0)$. A state q of S is characterised by the couple (state of G, state of T); hence, $q_0 = (p_0, x_0)$.

The algorithm is based on four steps. The first step gives the synchronous product of the automaton of the process and the automaton representing the constraints. The second step aims at identifying the forbidden states of the synchronous product. A forbidden state is a state for which there exists a non-controllable event, σ, accepted in the corresponding state of the plant, q, and unauthorised in the corresponding state of the constraints, x. Following the definition of the controllability, if the synchronous product does not include a forbidden state then it is controllable and can, thereby, be used as a supervisor. The third step is therefore used to identify the weakly forbidden states, i.e.

1- $\forall (p, x) \in Q$: If $\exists (p, \sigma, p') \in \delta$ and $\exists (x, \sigma, x') \in \xi$ Then $\Delta = \Delta \cup \{ ((p, x), \sigma, (p', x')) \}$;

2- $\forall q = (p, x) \in Q$: q is forbidden if $(\exists (p, \sigma, p') \in \delta / \sigma \in \Sigma_u, \wedge \nexists (x, \sigma, x') \in \xi)$

3- If $(\nexists q' \in Q)$ such that q' is forbidden,
Then End
Else $\forall q \in Q$: q is weakly forbidden if (q is not forbidden and $\exists s \in \Sigma^* / (\forall \sigma \in s, \sigma \in \Sigma_u) \wedge (q' = \Delta(q,s)$ is forbidden)),

4- $\forall q \in Q$ such that q is forbidden or weakly forbidden Do:
- $\forall (q', \sigma, q) \in \Delta, \Delta = \Delta - \{ (q', \sigma, q') \}$
- $\forall (q, \sigma, q') \in \Delta, \Delta = \Delta - \{ (q, \sigma, q') \}$
- $Q = Q - \{q\}$

$\forall q' \in Q$ such that $\nexists (s \in \Sigma^* / q' = \Delta(q_0, s))$: $Q = Q - \{q'\}$

Fig. A. Supervisory synthesis algorithm according to [31].

the states from which there exists a sequence of non-controllable events leading to a forbidden state; Σ^* denotes the set of all finite strings of elements of the set Σ. The last step removes the forbidden and weakly forbidden states as well as their input and output transitions. Then the states, which are unreachable from the initial state, are trimmed from the synchronous product.