Validation and verification of grafcets using state machine

Jean-Marc Roussel, Jean-Jacques Lesage

Validation and Verification of grafcets using finite state machine

Jean-Marc ROUSSEL & Jean-Jacques LESAGE
Laboratoire Universitaire de Recherche en Production Automatisée

LURPA - ENS Cachan
61 Avenue du Président Wilson
94235 CACHAN Cedex

Email: roussel@lurpa.ens-cachan.fr
URL: http://www.lurpa.ens-cachan.fr/~Roussel.html

ABSTRACT

This paper presents a method to *verify* (the internal consistency) and to *validate* (with respect to the purpose of the builders) *Sequential Function Charts* [8] (grafcets in French). The method is based upon the translation of any grafcet into its equivalent finite automaton. The proofs of consistency of the models are then established on this automaton. The main difficulty of this approach is the control of the combinatorial explosion implied by the parallel and the synchronous nature of Grafcet. A specific grammar has been developed in order to express the expected properties to prove. An example is given to illustrate the presented approach.

Key-Words: Sequential Function Chart (S.F.C.), Grafcet, validation, verification, reachable situations graph, finite state machine.

1. INTRODUCTION

During the development of a control system design project, the Sequential Function Charts (SFC: Grafcet in French) are often used in two essential steps: the specification of the expected behavior of the control system and the Programmable Logical Controllers (PLC) programming. It must be emphasized that although we are talking about Grafcet in these two cases, the underlying formalisms are different ; the specification of the behavior of control systems is written according to the IEC 848 standard («Preparation of function charts for control systems» [9]) while the PLC programs are written according to the IEC 1131-3 standard («Programmable controllers - Part 3: Programming languages» [10]). This paper only concerns the specification of the dynamic behavior of logical sequential systems using the Grafcet IEC 848 standard and take into account the theoretical hypothesis of this model [5].
The increasing complexity of automated systems and the consequent requirements in the field of timeliness and safety require the use of formal methods to validate the control system models from the specification step. In this paper we will adopt, for the grafcet models, the definition given in [14] concerning the verification and the validation of models (Fig. 1).
The *verification* is the proof that the internal semantics of a model is correct, independently from the modeled system. The searched properties of the models are stability, deadlock existence, ... The *validation* determines if the model agrees with the designer's purpose. The searched properties of the models are then safety properties, temporal properties and liveness properties. Our approach takes into account these two aspects and aims at proving the global consistency of a grafcet: its consistency with respect to the hypothesis and the syntax of IEC standard, as well as with respect to the expected properties of the modeled system.
In order to illustrate the objectives of our approach, we are going to present an example of the expectations of the Grafcet builder in the field of verification and validation.

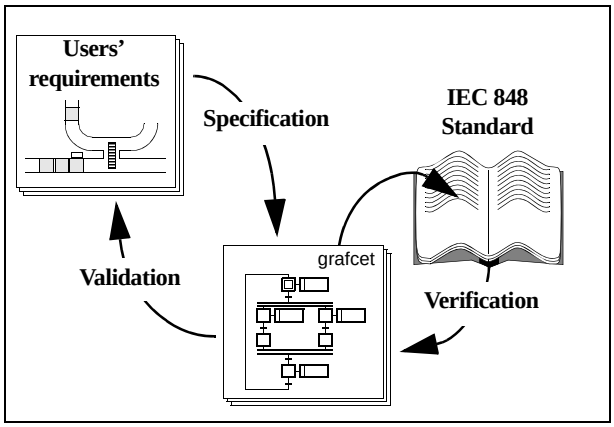

Fig.1 : Verification and validation of grafcets

2. EXAMPLE

This example concerns the control of a transfer module for a convoy system (Fig. 2).
To reduce waiting delay, the functions «pallet in» and «pallet out» are cut into two elementary parts. The control of the transfer module must enable to simultaneously handle the maximum of pallets without collision. The running rules are the following:

- R1: In the transfer module, there is not more than one pallet in or out.
- R2: A pallet cannot be cleared if another one is coming in or out.
- R3: priority to out.

A corresponding specification by Grafcet is given (Fig. 3).
The designer has built 4 connex grafcets:

- the grafcet (steps 1 to 10) concerns the control of actuators of the main line,

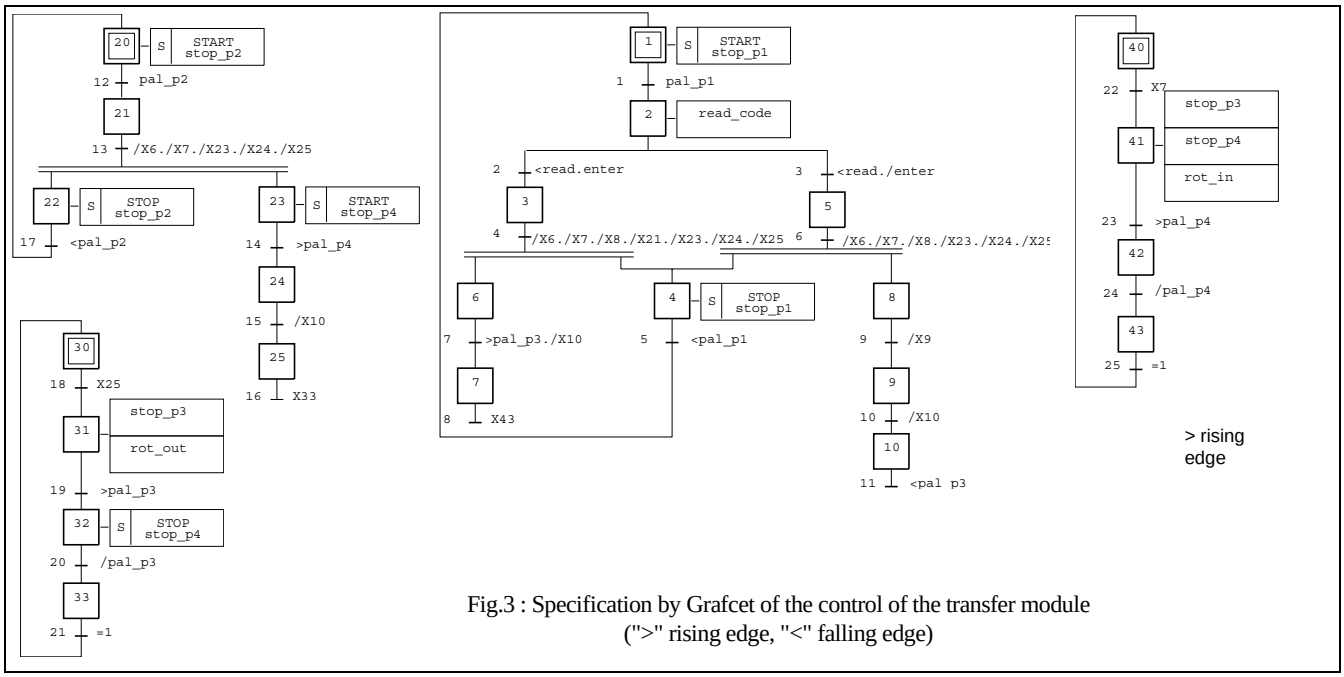

Fig.3 : Specification by Grafcet of the control of the transfer module
(">" rising edge, "<" falling edge)

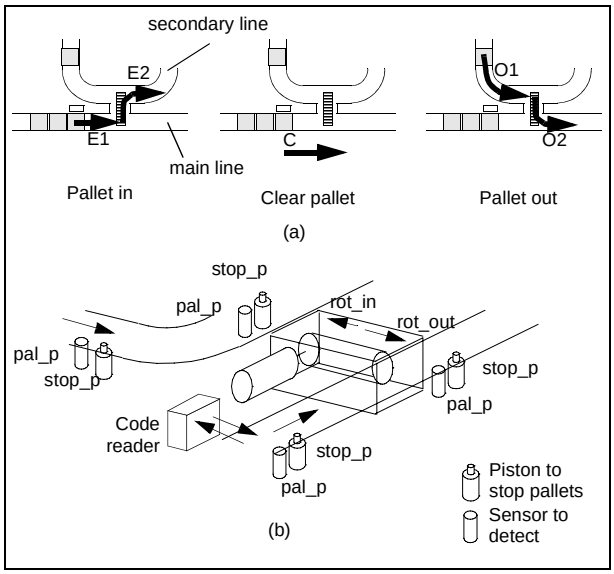

Fig.2 : Tranfer module for a convoy system: (a) Elementary movements, (b) Position of sensors and actuators

- the grafcet (steps 20 to 25) concerns the control of actuators of the secondary line,
- the 2 grafcets (steps 30 to 33 and steps 40 to 43) concern the control of actuators of transfer module.

The synchronizations between graphs are made by using step activity variables in transition condition (transition 4, 6, 13 for example) and the control of output variables is made by continuous or stored actions.

To verify this specification, the designer must prove:
- the stability property of the model,
- the lack of dead-lock,
- the good use of stored actions,
- ...

To validate this specification, he must also prove:
- the lack of collisions of pallets,
- the correct control of the actuators of the transfer (for example, the motor is not controlled simultaneously in both directions)
- the respect of priority rules between pallets,
- ...

3. THE ISSUE

One of the most interesting capacities of Grafcet is to allow the designer to simply describe parallelisms [3] (by using multi-graph descriptions, and convergence symbol, stored actions, ...). For this reason, a *state* of the control system modelized is represented by a *situation* of the grafcet (i.e. a set of active steps at a given moment). For example, on the grafcet Fig. 3, the situation corresponding to the initial state of the control system is S_0 = {1, 20, 30, 40}. Similarly, the *change from a state* of the modeled system to another is represented by a *set of transitions* which have to be simultaneously cleared. For instance, on the grafcet Fig. 3 the evolution of the situation S_0 to the situation S_1 (S1 = {2, 21, 30, 40}) is achieved by the simultaneous clearing of the transitions t_1 and t_2 when $pal\_p1 \cdot pal\_p2 = 1$. These concepts of *situation* and *evolution between situations* then contribute to increase the capacity of the Grafcet to represent significant parallelisms with compact models.

In fact, the number of steps and transitions of a grafcet are to the must equal to the number of states and changes between states of the modeled system; in the general case the number of steps and transitions is much lower. In our example, the grafcet has only 24 steps and describes a control system of 238 stable states.

Contrarily, this power of modeling makes the validation of the models very hard to establish. In fact, once it has been constructed in terms of *steps* and *transitions*, a model must be validated in terms of *situations* and *evolutions between situations*.

4. OUR APPROACH

The set of situations which can be reached and the set of the possible evolutions between these situations are described in the Reachable Situations Graph (R.S.G.) [4]. The R.S.G. is then the bearer of all the semantics of a grafcet in relation to the modeled system. That is why our validation method [15] is based upon the automatic generation of the R.S.G., which is a finite automaton by which we prove the expected properties of the control system (Fig. 4), contrary to other approaches of validation which translate grafcets into other languages (automata [1] [6] [12], temporal algebra [11], synchronous languages [1] [12], (Max,+) algebra [13]).

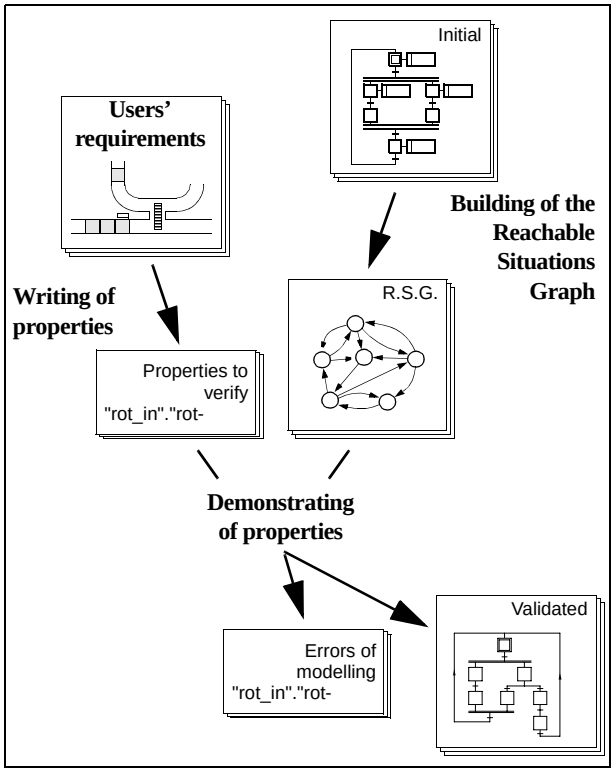

Fig.4 : Our approach

This strategy brings numerous performances and more flexibility during the proof of properties:

- When the R.S.G. is built, it is easier to prove a property because it is simpler to verify if a situation exists than to know if it can be obtained.
- A property of accessibility between specific situations is easier to prove because it is only a search of paths in a graph.
- By separating the building of R.S.G. from the demonstration of properties, it is possible to take Grafcet specificities such as reachability of stable situation or immediate reactivity of input variations into account.
- When a property is not proved, the knowledge of all faulty situations or evolutions facilitates the correction of the initial grafcet.

Building of the R.S.G.

To validate a grafcet by using its R.S.G., the equivalence between the two models must be ensured. This equivalence is not only a behavior equivalent (the same input variations give the same output variations for the two models) but it means that the R.S.G. must be composed of all and only all the reached situations of the grafcet and of all and only all the evolutions between these situations. The forgetting or the adding of situations may altered the results of validation.

It is the reason why we have developed a method to construct the R.S.G. in two stages.

In the first stage, a primary R.S.G. is built. All situations and evolutions of the grafcet are iteratively found. From each reachable situation, all the sets of validated transitions which are simultaneously cleared are built. The condition of each set of clearing transitions is obtained by computer algebra from the receptivities [7]. All the evolutions of inputs are considered in the initial construction of the R.S.G. However, a fundamental hypothesis of Grafcet specifies that two independent inputs cannot change of value simultaneously.

In the second stage, the evolution of inputs which do not verify this hypothesis are suppressed in the R.S.G. (evolution where two inputs simultaneously change (Fig. 5a), and evolution where an input changes twice in the same direction (Fig. 5b)).

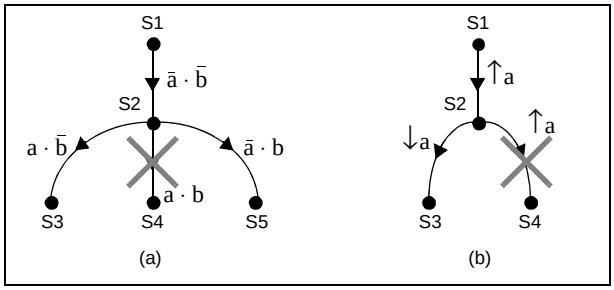

Fig.5 : Examples of incoherent evolutions

This method of construction of the R.S.G includes the following Grafcet specificities:

- the possibilities of interpreted parallelism,
- the use of edges and step variables in the transition conditions [7],
- the reachability of a stable situation [5],
- the immediate reactivity to input variations.

From a mathematical point of view, the generated R.S.G. is a completely specified Mealy Machine with a uniform sequential machine [18]. We describe this machine by a 6-uple [X, Y, Z, Y_0, T, A] where:

- X is the set of grafcet inputs,
- Y is the set of states (each state represents a different reached situation of the grafcet),
- Z is the set of grafcet outputs,
- Y_0 is the initial state,
- T is the set of transitions (each transition represents an evolution between two reached situations of the grafcet),
- A is the set of actions.

Each transition is defined by a 3-uple (Yup , Cond, Ydo) where « Yup » and « Ydo » are two states (respectively upstream state and downstream state) and «Cond» a boolean expression of inputs.

This definition allows us to combine the «state machine» aspect (notion of inputs, outputs, states and actions) with the «graph» aspect (notion of succession of states) into a set description which is adapted to the proof of properties.

Writing of properties

To validate a grafcet by using its R.S.G., it is necessary to express the properties to verify by the mean of a mathematical expression, so as to prove them. We have thus developed a specific grammar with the following aspects:

- the properties concern the Grafcet: operators of this grammar must refer to the Grafcet vocabulary (step, situation, inputs, action, ...),
- some properties concern accessibility between situations: operators of this grammar must refer to graph theory,
- to validate a grafcet, it is necessary to write the properties concerning several criterions: this grammar must use the notion of set in order to compose complex properties.

Presently, this grammar is composed of 13 operators:

- the 3 operators of the set theory:
  Union ($\cup$), Intersection ($\cap$) and Subtraction (-) of sets,

- 6 operators[1] to analyze graph aspects:
  $Int(Y_i)/Ext(Y_i)$ are respectively the sets of transitions where the upstream / downstream state is an element of Y_i ,
  $Amont(T_i)/Aval(T_i)$ are respectively the sets of states which are the upstream / downstream state of a element of T_i ,
  $Succ(Y_i, T_i)$ is the set of states which are reached from an element of Y_i by only following an element of T_i ,
  $Pred(Y_i, T_i)$ is the set of states from which it is possible to reach an element of Y_i by only following an element of T_i ,

- 4 operators related to Grafcet specificities:
  $Act(E)/Des(E)$ are respectively the sets of states which represent a situation where all elements of E is / is not active,
  $Nec(Exp\_in)$ is the set of transitions which are cleared only if the combinatorial expression of *inputs* «Exp_in» is true,
  $Emis(Exp\_out)$ is the set of states from which the combinatorial expression of *outputs* «Exp_out» is sent.

Demonstrating of properties

The result of the 13 grammar operators are sets of transitions or of states. To prove the properties given by this grammar, a specific module for the manipulation of sets has been built.
In this module, we have reproduced the algorithms used in MEC [2] for operators related to graph aspects.
For the operators which work from combinatorial expression, we use the computer algebra module developed for the building of R.S.G.

5. VALIDATION POSSIBILITIES

Example of properties

In this section, we present several properties that can be verified with our approach. Each one is expressed in our grammar.

Lack of dead-lock

In Grafcet, the possibility to obtain a dead-lock is increased by the use of step activity variables in condition transitions. There are two types of dead-lock: the dead-lock is said *global* if the whole grafcet is locked or said *local* if only a part of the grafcet is locked.

To know the global dead-locked situation, it is necessary to evaluate the following set:

$$Y - Amont(T)$$

that contains the states without downstream transition.
The local dead-lock is more usual. To know all the steps which are included in a local dead-lock, it is necessary to evaluate the following set:

$$\{e \mid ((Act(\{e\}) - Pred(Des(\{e\}), T)) \cup (Des(\{e\}) - Pred(Act(\{e\}), T)) \neq \varnothing)\}$$

that contains the steps with the state cannot change since a given situation.

Come back to the initial situation

The initial situation is often a strategic situation for the grafcet: it corresponds to a falling-back state for the system. To know the situations from which the initial situation cannot be reached, it is necessary to evaluate the following set:

$$Y - Pred(Y_0, T)$$

that contains the states from which no path exists to initial state.

Output sending

With the presented grammar, it is possible to express and verify properties related to the simultaneous or sequential sending out of outputs.

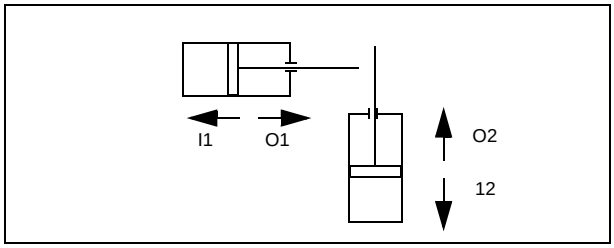

Fig.6 : Collision between pistons

For example, to validate the control without collision of two pistons disposed as Fig. 6, it is necessary to verify if the two pistons do not move simultaneously:

$$Emis((I1 + O1) \cdot (I2 + O2)) = \varnothing$$

and verify if the going out of a piston always occurs after the going in of the other:

$$\begin{cases} Emis(O1) \cap Pred(Emis(O2), T - Ext(Emis(I1))) = \varnothing \\ Emis(O1) \cap Pred(Emis(O2), T - Ext(Emis(I1))) = \varnothing \end{cases}$$

Use of stored actions

The use of stored actions is often forbidden in industry as it introduces an interpreted parallelism in the control model (a part of the control system is not represented in the model).
Let «o» be an output of the grafcet which is used in stored actions and Y_{start_o} (Y_{stop_o}) be the set of states from which the start (stop) stored action «o» is send.
To verify the good use of stored actions, we must know if:

- the output variable «o» is not simultaneously start and stop stored,

$$Y_{start\_o} \cap Y_{stop\_o} = \varnothing$$

1. This operators are taken from the MEC tool developed for the validation of transitions systems [2]

- for each state, the output variable «o» has always the same value,

$$(\mathrm{Succ}(Y_{start\_o}, (T - \mathrm{Int}(Y_{stop\_o}))) \cup \mathrm{Emis}(o)) \cap$$
$$\mathrm{Succ}((Y_{stop\_o} \cup (Y_0 - Y_{start\_o})), (T - \mathrm{Int}(Y_{start\_o}))) = \varnothing$$

- the output variable «o» is not twice stored,

$$(\mathrm{Succ}(Y_{start\_o}, (T - \mathrm{Int}(Y_{stop\_o}))) - Y_{start\_o}) \cap$$
$$\mathrm{Pred}((Y_{stop\_o} \cup (Y_0 - Y_{start\_o})), (T - \mathrm{Int}(Y_{start\_o}))) = \varnothing$$

Application of our example

The validation of the specification by Grafcet for our example was made with the software «AGGLAÉ» (Fig. 7) [17], developed in our laboratory.

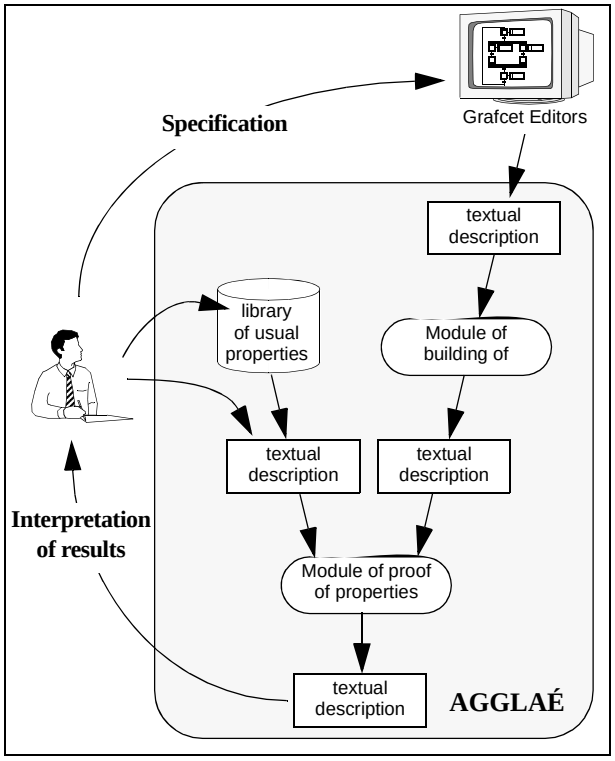

Fig.7 : Structure of «AGGLAÉ»

Firstly, the R.S.G. was calculated (2 seconds for a PC compatible computer equipped with Pentium75 processor). During this operation, it was necessary to simplify 12 496 combinatorial expressions formally. This R.S.G. contains 298 states and 822 transitions.

Secondly, we verified the good use of Grafcet Standard. We proved:

- the stability property of the Grafcet model,
- the lack of dead-lock,
- the good use of stored actions,
- ...

Thirdly, we validated the specification related to the user's requirements. We proved:

- the control without conflict of the transfer motor,
- the possibility to simultaneously have 4 pallets in the transfer module,
- ...

The collision risk of pallets in the transfer module is very important in this case. We have built the grafcet (Fig. 3) such as:

- the sequence (23, 24, 25) controls the exit of pallets,
- the sequence (6, 7) controls the entry of pallets,
- the step 10 represents a pallet which is running on the main line.

A collision thus occurs when the following combinatorial expression is true:

$$X10 \cdot (X7 + X25) + (X6 + X7) \cdot (X23 + X24 + X25)$$

The grafcet (Fig. 3) includes an error[1] that induces a collision between pallets. The R.S.G. (an excerpt is given Fig. 8) contains the faulty[2] situation {1 10 20 25 31 40}.

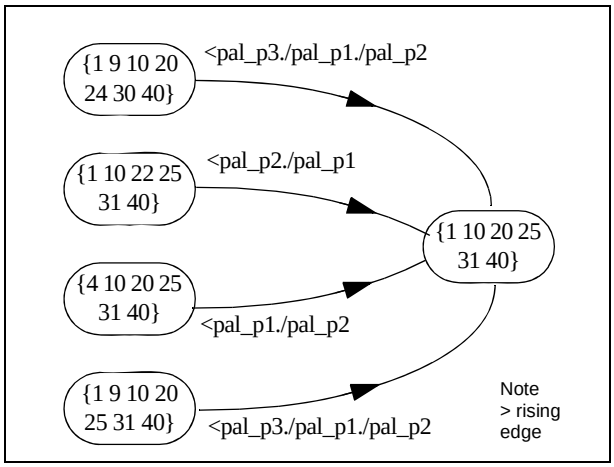

Fig.8 : possibilities to reach the situation {1 10 20 25 31 40} in the given specification

To uncover this mistake, the evolutions that end in a faulty situation have to be discovered, in order to suppress them. These evolutions are obtained by calculating the following set of transitions:

$$\mathrm{Ext}((Y - \mathrm{Act}(\{10, 25\}))) \cap \mathrm{Int}(\mathrm{Act}(\{10, 25\}))$$

In our example, to obtain a specification without collision, it is enough to complete the condition associated to the transition «15» by «/X9» (Fig. 9).

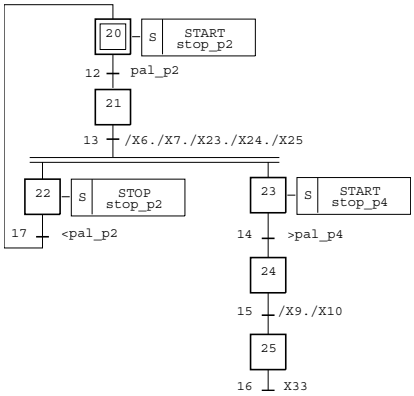

Fig.9 : Correction of the grafcet (Fig. 3)

1. this mistake could be done by every designer !
2. for this situation, $X10 \cdot X25 = 1$

For the final grafcet, its R.S.G. contains 238 states and 646 transitions.

In [16], a whole example is given. All properties to verify are fully expressed.

During our experimentations of different applications, we observed the frequent use of similar properties. In order to simplify validation activity, we are currently developing the use of a library of specific properties. Each of these properties is written in our grammar. Analysts can use each property by precising parameters and personalize this library by introducing their own properties (Fig. 7).

6. CONCLUSION

This paper presents a method to verify and to validate Sequential Function Charts. Our method is based upon the translation of any grafcet into its equivalent finite automaton. We can thus prove numerous safety and liveness properties of the modeled system. The experimental results show that we control the combinatorial explosion implied by the parallel and the synchronous nature of Grafcet. Our present works aim at including temporal aspects of the models into our approach (delayed or time limited action, time dependent transition condition, ...) and at improving the ergonomics of our software.

7. REFERENCES

[1] C. André, D. Gaffé, «Proving properties of grafcet», in this issue

[2] A. Arnold, «Mec: a system for constructing and analysing transition system», in J. Sfakis, editor, «Automatic verification of finite state systems» pp 117-132, 1989

[3] P. Baracos, «Grafcet step by step», Famic Inc. editor, 1992 - 156 p.

[4] M. Blanchard, «Comprendre, maitriser et appliquer le Grafcet», Toulouse: Cepaduès Editions, 1979 - 174 p.

[5] N. Bouteille, P. Brard, G. Colombari, N. Cotaina, D. Richet, «Le Grafcet», Toulouse: Cepaduès Editions, 1992 - 144p.

[6] V. Chapurlat, M. Larnac, G. Dray, «Analysis and formal verification of grafcet using interpreted sequential machine», in this issue

[7] B. Denis, J.J. Lesage, J.M. Roussel, «A Boolean algebra for a formal expression of events in logical systems», Proceedings of Congres «1.Mathmod Vienna», pp. 859-862, Vienna, 2-4 Februar 1994

[8] Work Group Grafcet of Afcet association, «Grafcet WWW Server», http://www.lurpa.ens-cachan.fr/grafcet.html

[9] International Electronical Commission, «IEC 848: Preparation of function charts for control systems», Geneve: IEC Editions, First Edition 1988

[10] International Electronical Commission, «IEC 1131-3: Programmable controllers - Part 3: Programming languages», Geneve: IEC Editions, 1992

[11] P. de Loor, J. Zaytoon, «Validation of real-time properties of grafcet controlled systems», in this issue

[12] L. Marcé, D. L'Her, P. Le Parc, «Modelling and verification of temporized grafcet», in this issue

[13] H. Panetto, S. Gaubert, P. Lhoste, «Stability property of the Grafcet Model with (Max,+) algebra», in this issue

[14] A.K. Patankar, S. Adiga, «Entreprise integration modelling: a review of theory and practice», Computer Integrated Manufacturing Systems Vol. 8-1, pp 21-34, 1995

[15] J.M. Roussel, «Analyse de grafcets par génération logique de l'automate équivalent», PhD thesis, ENS de Cachan France, 16 december 1994

[16] J.M. Roussel & J.J. Lesage, «Définition d'un cadre formel pour l'expression et la vérification de propriétés d'un modèle Grafcet», Proceedings of AFCET Congress «Modélisation des Systèmes Réactifs» (MSR'96), pp 229-237, Brest-France, 28-29 Mars 1996

[17] J.M. Roussel & J.J. Lesage, «AGGLAÉ: a tool for validate grafcets», http://www.lurpa.ens-cachan.fr/csap/agglae.html

[18] J. Zahnd, «machines séquentielles», Paris: Dunod Editions, 265 pages, 1987