A formal semantics for Grafcet specifications

Julien Provost, Jean-Marc Roussel, Jean-Marc Faure

A formal semantics for Grafcet specifications

Julien Provost Jean-Marc Roussel Jean-Marc Faure

Abstract— **This paper shows how the behavior of a model described in the specification language proposed by the IEC 60848 standard can be represented, without semantics loss, in a formal manner, by a finite state machine (FSM) with logic inputs and outputs. This contribution is illustrated on a non-trivial example; this case study points out that the duration of the construction of the equivalent FSM complies with the requirements of designers of automation systems.**

I. Introduction

Standardized specification languages have been developed to ease design of industrial systems. To meet this objective, they propose powerful, often graphical, constructs that allow the designers express easily their needs and facilitate communication during the life-cycle. In the case of control systems, the IEC 60848 standard presents a specification language, called Grafcet, which describes graphical constructs to express parallelism, concurrency, rendez-vous, outputs assignment, and other mechanisms frequently met in this class of systems. Unfortunately, no formal semantics of this language is provided, as this is usually the case in standards. This drawback prevents from using formal analysis methods, like model-checking [1], to verify whether a Grafcet behaves as expected once it is built.

The aim of this paper is to tackle out this issue by providing a formal semantics of the IEC 60848 Grafcet language. This will be achieved by representing the full behavior of a Grafcet model in the form of a finite state machine, named Stable Location Automaton (SLA). For room reasons, only non-timed models will be considered in this paper; this limitation is not too strong because the first concern of engineers, during development, is functional correctness, time correctness becoming a concern once functional correctness is ensured.

It shall be noted that similar works have been achieved for the IEC 61131 programming languages [2], [3], [4], [5], [6]. Whatever the value of these works, it must be highlighted that it is more significant, for costs reasons, to analyze specification models, e.g. Grafcet models, before these models are implemented, in the form of PLC programs for instance, because the earlier in the life-cycle flaws are detected, the less expensive flaws removal is; verification of a specification does not prevent from checking the implementation of this specification but facilitates strongly this latter analysis. Moreover, the formal semantics used in the referred works is often that of a specific tool, e.g. a non-timed or timed model-checker, which limits the scope of the contributions, while the method proposed in this paper is based on a generic model which can be translated into several well-known formalisms, like Mealy machines or transition systems.

Several formalization methods of Grafcet specifications have been proposed earlier [7], [8], [9], [10]. [7] and [8] present a formalization of the Grafcet behavior using a static meta-model. Then, interpretation rules need to be introduce in order to take into account the dynamic aspect of the Grafcet behavior. [9] proposes a formalization of reactive Grafcet (Grafcet with event-based inputs), unfortunately this formalization does not take into account any logical output. [10] proposes a formalization method based on iterative computation of the reachable sets of active steps from the initial set of active steps defined in the Grafcet specification.

More recently, [11] proposed a formalization of Grafcet for DES control synthesis. This formalization, based on the current version of the Grafcet standard [12], does not take into account the different types of actions associated to the Grafcet steps, e.g. stored actions and conditional actions.

The formalization method proposed in this paper permits to take into account the main features of the Grafcet semantics (conditional and stored actions, macro-steps, ...). Besides, since the knowledge of relations between inputs and outputs is necessary to many verification and validation methods, this formalization extends the one proposed by [10] and puts the stress on the definition of the emitted outputs.

The outline of this paper is the following. The Grafcet syntax and semantics are reminded in Section II. The formal definitions of Grafcet and SLA are given respectively in Sections III and IV while Section V deals with the construction of the equivalent SLA of a Grafcet. This proposal is exemplified in section VI and some perspectives for further work are sketched in section VII.

II. Grafcet specification language

Grafcet is a standardized graphical specification language [12] to describe the behavior of logic sequential systems. This language is widely used in several industrial domains, like railway transport, electrical power production, manufacturing industry, environment, to specify the expected behavior of a logic control system which is connected to a physical system (plant) that sends logic signals to the control system and receives the logic signals which are generated in response. Grafcet was first standardized in France at the beginning of the 1980s, and at the international level in 1988. Since this date, several extensions have been proposed

This work is funded by the French Research Agency (TESTEC project, Ref. TLOG 07-022)

The authors are with LURPA, École Normale Supérieure de Cachan, Cachan, 94230, France

E-mail: `firstname.name@lurpa.ens-cachan.fr`

to enhance the modeling possibilities; they are included in the last version of the standard [12]. A good scientific presentation of the main features of the previous and current versions of the Grafcet standard can be found respectively in [13] and [14]. Last, the reader is warned that the specification language described in the IEC 60848 standard differs from the SFC (Sequential Function Chart) proposed by the IEC 61131-3 standard [15], even if both are often named SFC in English and if models in these two languages may look similar; the differences stand both in syntax and semantics. The main differences between those two languages will be discussed in subsection 'Differences between Grafcet and SFC'. To avoid misunderstandings, only the term Grafcet will be kept in the sequel of this paper for the specification language.

Grafcet has been developed from the results of the Petri nets community and in particular from those on Interpreted Petri Nets. A specific syntax and semantics have been defined, in a textual way, however in the standard, to take into account the specific needs of engineers when specifying complex sequential systems. The key features of these definitions are briefly recalled in what follows.

A. Grafcet syntax

A Grafcet model describes the expected behavior of a logic controller which receives logic input signals and generates logic output signals; then, the input and output variables of a Grafcet are both logic variables. A Grafcet (Figure 1) comprises steps, graphically represented by squares, and transitions, represented by horizontal lines; a step can only be linked to transitions and a transition only linked to steps. The links from steps to transitions and from transitions to steps are oriented links. The default orientation is from top to bottom and it is not necessary in this case to put an arrow on the link. An arrow must be put on a link if this link goes from bottom to top or may be put on any link to ease understanding. A step defines a partial state of the system and can be active or inactive; hence, a Boolean variable, named step activity variable can be defined for each step. Actions may be associated to a step; an action associated to a step is performed only when this step is active and then acts upon an output variable. A transition condition must be associated to each transition; this condition is a Boolean expression which may include input variables, steps activity variables and conditions on time. As only non-timed systems are considered in this work, only the Grafcets whose transition conditions are built from input variables and steps activity variables are dealt with.

Moreover, macro-steps may be introduced in a Grafcet, to ease modeling. A macro-step, represented graphically by a square with double-lines on top and at the bottom, is a synthetic view of a part of the specification. The detailed description of this part is termed macro-step expansion chart and is a set of connected steps and transitions that starts and ends by only one step, called macro-step expansion input and output steps. Then, a Grafcet model may be composed of several charts: classical charts, that include (normal or macro) steps and transitions, and macro-step expansion charts.

Figure 1 depicts a Grafcet that comprises a part of an example coming from industry. This model is composed of two classical charts (on the left side) and two macro-step expansions (on the right side); these latter charts are the expansions of macro-steps 'M10' and 'M20'.

B. Evolution rules

The detailed behavior of any Grafcet model can be obtained by applying five evolution rules that can be stated as follows:

R1 At the initial time, all the initial steps, defined by the model designer and double-squared, are active; all the other steps are inactive.

R2 A transition is enabled when all the steps that immediately precede this transition (upstream steps of the transition) are active. A transition is fireable when it is enabled and when the associated transition condition is true. A fireable transition must be immediately fired.

R3 Firing a transition provokes simultaneously the activation of all the immediately succeeding steps and the deactivation of all the immediately preceding steps.

R4 When several transitions are simultaneously fireable, they are simultaneously fired.

R5 When a step shall be both activated and deactivated, by applying the previous evolution rules, it is activated if it was inactive, or remains active if it was previously active.

These textual definitions of the evolution rules come from the standard and are obviously not formal; formal definitions of these rules will be given in Section V. However, these rules show that the global state of a Grafcet, called situation, is defined by the set of all the simultaneously active steps; the initial situation of the model of figure 1, for instance, is $\{A1,30,32,34,36,38\}$. An evolution from the current situation to a new one corresponds to the firing of simultaneously fireable transitions, according to rules 2 (fireable transitions are compulsory fired) and 4 (simultaneously fireable transitions are simultaneously fired). This new situation may be *transient* or *stable*; a situation is transient if at least one transition of the Grafcet can be fired from this situation without change of the inputs values, and stable if no enabled transition is fireable from this situation for the current values of inputs. Then, the state of a Grafcet evolves from stable situation to stable situation, possibly by crossing transient situations. An evolution between two stable situations corresponds, in a Grafcet model, to a sequence (may be reduced to one) of firings of sets of fireable transitions and is instantaneous. Examples of evolutions will be given in section VI, once the formal semantics of Grafcet defined.

C. Actions

Two kinds of actions can be used in a Grafcet to specify the values of outputs: continuous actions and stored actions. An action is graphically represented by a rectangle linked to the step symbol.

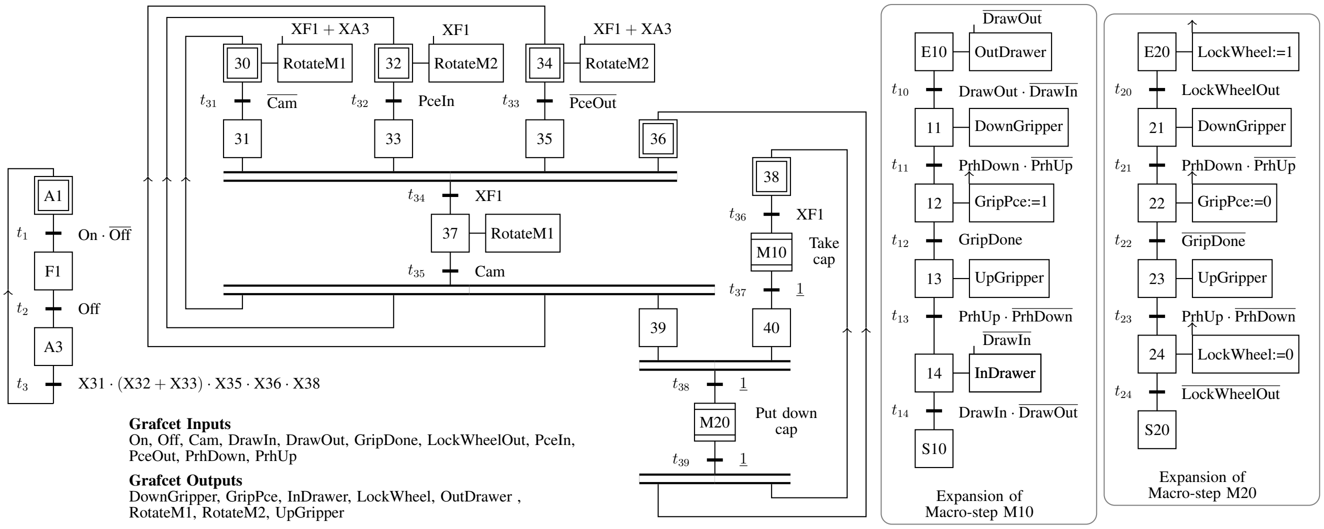
XF1 + XA3
30
RotateM1
XF1
32
RotateM2
XF1 + XA3
34
RotateM2
t_{31}
$\overline{\text{Cam}}$
t_{32}
PceIn
t_{33}
$\overline{\text{PceOut}}$
31
33
35
36
A1
t_1
On · $\overline{\text{Off}}$
F1
t_2
Off
A3
t_3
X31 · (X32 + X33) · X35 · X36 · X38
t_{34}
XF1
37
RotateM1
t_{35}
Cam
38
t_{36}
XF1
M10
Take cap
t_{37}
$\underline{1}$
39
40
t_{38}
$\underline{1}$
M20
Put down cap
t_{39}
$\underline{1}$
Grafcet Inputs
On, Off, Cam, DrawIn, DrawOut, GripDone, LockWheelOut, PceIn, PceOut, PrhDown, PrhUp
Grafcet Outputs
DownGripper, GripPce, InDrawer, LockWheel, OutDrawer , RotateM1, RotateM2, UpGripper
$\overline{\text{DrawOut}}$
E10
OutDrawer
t_{10}
DrawOut · $\overline{\text{DrawIn}}$
11
DownGripper
t_{11}
PrhDown · $\overline{\text{PrhUp}}$
12
GripPce:=1
t_{12}
GripDone
13
UpGripper
t_{13}
PrhUp · $\overline{\text{PrhDown}}$
$\overline{\text{DrawIn}}$
14
InDrawer
t_{14}
DrawIn · $\overline{\text{DrawOut}}$
S10
Expansion of Macro-step M10
E20
LockWheel:=1
t_{20}
LockWheelOut
21
DownGripper
t_{21}
PrhDown · $\overline{\text{PrhUp}}$
22
GripPce:=0
t_{22}
$\overline{\text{GripDone}}$
23
UpGripper
t_{23}
PrhUp · $\overline{\text{PrhDown}}$
24
LockWheel:=0
t_{24}
$\overline{\text{LockWheelOut}}$
S20
Expansion of Macro-step M20

Fig. 1. Grafcet specification used to illustrate the proposed approach

- A continuous action specifies the current value of an output according to the current values of steps activity variables and inputs. For figure 1, for instance, output ‘DownGripper’ is true if and only if step ‘11’ or step ‘21’ is active; output ‘OutDrawer’ is true if and only if step ‘E10’ is active and the associated condition ‘$\overline{\text{DrawOut}}$’ is true (i.e. ‘DrawOut’ is false).
- A stored action describes how a Boolean value is allocated to an output variable, according to an allocation rule. A rising arrow associated to the action symbol means that the output variable is allocated when the step becomes active. On the other hand, a falling arrow means that the output variable is allocated when the step is deactivated. For the example, output ‘LockWheel’ is allocated to true (Set) when step ‘E20’ becomes active and allocated to false (Reset) when step ‘24’ becomes active.

It matters to underline that a continuous action is executed if and only if the current situation is stable whereas a stored action is executed whatever the situation.

This set of evolution rules and actions definitions ensures that Grafcet models are deterministic, what is mandatory for controllers’ specifications.

D. Differences between Grafcet and SFC

This comparison is based on the two normative texts that define Grafcet and SFC, respectively [12] and [15]. In what follows, the extracts from these documents will be written in italics.

Grafcet is a *specification language for the functional description of the behavior of the sequential part of a control system*. On the opposite, SFC is an element to structure the internal organization of a program organization unit executed by a PLC (Programmable Logic Controller) and written with one of the four IEC 61131-3 languages (LD, ST, IL, FBD). Then, a Grafcet model is used to specify a behavior regardless of the controller that will be used to implement this specification, while an SFC model describes (part of) the structure of a software running on a PLC.

The semantics of Grafcet and SFC are different. For instance, in case of selection of sequence, *exclusive activation of a selected sequence is not guaranteed from the structure*, in Grafcet; *the designer should ensure that [...] the transition conditions are mutually exclusive*. On the contrary in SFC, *any sequence selection is exclusive, [...] it cannot have crossing simultaneous transitions in a sequence selection; to do this, the user can define priorities between branches at the divergence of sequence selection*.

But the main semantic difference is that the evolutions of a Grafcet model are caused by the changes of its inputs values (event-driven approach) whereas the evolutions of an SFC model are controlled by the input scanning cycle of the controller that executes this model (time-driven approach). When the value of an input of a Grafcet model changes, this model evolves to a new stable situation; this evolution is instantaneous to avoid inputs values changes be missed, even if transient situations are crossed. On the opposite, in an SFC model, *the clearing time*[1] *of a transition may theoretically be considered as short as one may wish, but it can never be zero*; in practice, this time is equal to the input scanning cycle time of the controller on which the SFC runs. The consequence is that only one set of simultaneous fireable transitions (may be reduced to one transition) is fired at every scanning cycle. Hence, the concepts of transient and stable situations do not make sense in SFC; a situation lasts at least one cycle time in this case.

III. Formal definition of a Grafcet model

Formally, a Grafcet G is a 4-tuple $(I_G, O_G, C_G, S_{InitG})$ where:

- I_G is the non-empty set of logic inputs.
- O_G is the non-empty set of logic outputs.

[1] clearing time = firing time

- C_G is the set of Grafcet charts.
- S_{InitG} is the set of initial steps.

The charts set C_G is partitioned into the set C_C of classical charts and the set C_E of macro-steps expansions charts.

- A classical chart $c \in C_C$ is defined by a 3-tuple (S, T, A) where:
  - S: non-empty set of steps s of c.
  - T: set of transitions t of c.
  - A: set of actions a of c.
- A macro-step expansion chart $c \in C_E$ is defined by a 5-tuple $(m, s_I, s_O, S_{oth}, T, A)$ where:
  - m: macro-step name.
  - s_I: input step of the expansion.
  - s_O: output step of the expansion.
  - S_{oth}: set of the other steps s of the expansion.
  - T: set of transitions t of the expansion.
  - A: set of actions a associated to the steps of c.

Let $S(c)$ be the set of all steps of the macro-step expansion chart c ($S(c) = \{s_I(c), s_O(c)\} \cup S_{oth}(c)$). Let S_G be the set of all steps s of the Grafcet.

A transition $t \in T$ of a given chart c is defined by a 3-tuple $(S_U, S_D, E_{Cond(I_G,S_G)})$ where:

- S_U: set of the immediately upstream steps of t.
- S_D: set of the immediately downstream steps of t.
- $E_{Cond(I_G,S_G)}$: transition condition. It is a Boolean expression on inputs and steps activity variables[2].

Let T_G be the set of all transitions t of the Grafcet.

The set of actions A is partitioned into the set A_S of stored actions and the set A_C of continuous actions. Similarly, the outputs set O_G is partitioned into the set O_S of stored outputs (controlled by stored actions) and the set O_C of continuous outputs (controlled by continuous actions).

- A continuous action $a_c \in A_C$ of a given chart c is defined by a 3-tuple $(s, o, E_{Cond(I_G,S_G)})$ where:
  - s: step which the action is associated with.
  - o: output which is assigned by the action.
  - $E_{Cond(I_G,S_G)}$: continuous action condition. It is a Boolean expression on inputs and steps activity variables.
- A stored action $a_s \in A_S$ of a given chart c is defined by a 4-tuple (s,o,op,$inst$) where:
  - s: step which the action is associated with.
  - o: output which is allocated by the action.
  - op: kind of allocation, $op \in \{Set, Reset\}$.
  - $inst$: instant when the allocation is done, $inst \in \{Act, Deact\}$, where Act is the step activation instant and $Deact$ the step deactivation instant.

For each continuous output, an *output condition emission* is then defined by a Boolean expression on inputs and steps activity variables, as follows:

$$E_{Emit(I_G,S_G)}(o) = \sum_{\substack{a \in A_C \\ o(a)=o}} \big(X(s(a)) \cdot E_{Cond(I_G,S_G)}(a)\big)$$

This expression means merely that a continuous output is emitted if at least one step to which is associated an action that assigns this output is active and the condition of this action true.

On the opposite, the values of stored outputs do not depend simply on the values of inputs and steps activity variables but must be computed dynamically during the construction of the SLA, as it will be shown in Section V.

IV. Formal definition of the Stable Location Automaton (SLA) of a given Grafcet

Formally, a stable location automaton SLA is a 5-tuple $(I_{SLA}, O_{SLA}, L, l_{Init}, Evol)$ where:

- I_{SLA} is the set of inputs of Grafcet G: $I_{SLA} = I_G(G)$.
- O_{SLA} is the set of outputs of G: $O_{SLA} = O_G(G)$.
- L is a set of stable locations l.
- l_{Init} is the initial location, $l_{Init} \in L$.
- $Evol$ is a set of evolutions e.

A stable location is characterized by a set of simultaneously active steps, a set of emitted outputs and a stability condition, Boolean expression on the inputs values. Then, a stable location is defined by the 3-tuple: $(S_{Act}, O_{Emit}, E_{Stab(I_G)})$ where:

- S_{Act} is a subset of steps G, ($S_{Act} \subset S_G(G)$).
- O_{Emit} is a subset of outputs of G, ($O_{Emit} \subset O_G(G)$).
- $E_{Stab(I_G)}$ is a Boolean expression on inputs of G. This *stability condition* is true only for the combinations of inputs values for which the location l is stable.

An evolution e of $Evol$ is defined to represent an evolution between stable locations. Each one of these evolutions can be formally defined by the 3-tuple: $(l_U, l_D, E_{Evol(I_G)})$, where:

- l_U is the upstream location, $l_U \in L$.
- l_D is the downstream location, $l_D \in L$.
- $E_{Evol(I_G)}$ is a Boolean expression on inputs of G. This *evolution condition*, is true only for the combinations of inputs values for which the SLA evolves from location l_U to location l_D.

An SLA is well-defined if the following seven properties are satisfied:

1) Uniqueness of locations: $\forall (l_1, l_2) \in L^2$,
$(S_{Act}(l_1), O_{Emit}(l_1)) \neq (S_{Act}(l_2), O_{Emit}(l_2))$
2) Uniqueness of evolutions: $\forall (e_1, e_2) \in Evol^2$,
$(l_U(e_1), l_D(e_1)) \neq (l_U(e_2), l_D(e_2))$
3) Absence of self-loop: $\forall e \in Evol,\ l_U(e) \neq l_D(e)$
4) Determinism of evolution[3]: $\forall (e_1, e_2) \in Evol^2$,
$l_U(e_1) = l_U(e_2) \Rightarrow E_{Evol(I_G)}(e_1) \cdot E_{Evol(I_G)}(e_2) = 0$
5) Distinguishability between stability and evolution:
$\forall e \in Evol,\ E_{Evol(I_G)}(e) \cdot E_{Stab(I_G)}(l_U(e)) = 0$
6) Completeness[4]: $\forall l \in L$,
$$E_{Stab(I_G)}(l) + \sum_{e \in Evol,\ l_U(e)=l} E_{Evol(I_G)}(e) = 1$$
7) Absence of transient location: $\forall (e_1, e_2) \in Evol^2$,
$l_D(e_1) = l_U(e_2) \Rightarrow E_{Evol(I_G)}(e_1) \cdot E_{Evol(I_G)}(e_2) = 0$

[2] $X(s)$: activity variable of step s

[3] In Boolean equations, 0 means False, 1 means True

[4] For each location and for each combination of inputs values, the behavior is completely defined (either the location is stable, or there exists an evolution from this location).

V. Construction of the SLA of a given Grafcet model

As previously pinpointed, every evolution of an SLA comes from a change of inputs values. However, two kinds of evolutions may occur in an SLA:

- Evolutions that correspond to the firing of a sequence of sets of simultaneously fired transitions in the Grafcet.
- Evolutions that do not correspond to the firing of Grafcet transitions but only to the change of the emitted outputs.

In the first case, the set of active steps S_{Act} is changed; this is not true in the second one where only O_{Emit} and $E_{Stab(I_G)}$ are modified. The SLA of a given Grafcet model G is then built from the initial stable location by determining all these evolutions.

From a stable location $l = (S_{Act}, O_{Emit}, E_{Stab(I_G)})$, computation of evolutions conditions is performed by symbolic calculus on Boolean expressions on inputs and steps activity variables; if two sets S_1 and S_2 of inputs values combinations are defined by expressions Exp_1 and Exp_2, then sets $S_1 \cap S_2$, $S_1 \cup S_2$, and $S_1 \setminus S_2$ are respectively represented by $(Exp_1 \cdot Exp_2)$, $(Exp_1 + Exp_2)$ and $(Exp_1 \cdot \overline{Exp_2})$. Locations stability and outputs emission conditions will be computed in a similar manner. This solution avoids combinatorial explosion and is well suited to Grafcet models where transitions and actions conditions are Boolean expressions.

A. SLA evolutions due to the firing of a sequence of sets of simultaneously fired Grafcet transitions

These evolutions are determined from a stable location $l = (S_{Act}, O_{Emit}, E_{Stab(I_G)})$, by:

1) Looking for the set of fireable Grafcet transitions and all subsets of simultaneously fireable transitions, from the situation S_{Act}.
2) Determining the situation that is reached when each subset of simultaneously fireable transitions is fired; if the new situation is transient, then 1 and 2 are to be reiterated until a stable situation is found[5].
3) Finding the set of emitted outputs for the stable situation reached by the sequence of firings.

In the remainder of this section, these three computations are detailed and illustrated from the stable location

$$l_s = (\{\text{A}1, 30, 33, 35, 36, 38\}, \emptyset, ((\overline{\text{On}} + \text{Off}) \cdot \text{Cam}))$$

of the Grafcet model presented Figure 1. This location is reachable from the initial location and has been selected because it permits to well illustrate the three computations.

1) Phase 1: A Grafcet transition is fireable when it is enabled and its associated condition true. Hence, the set of enabled transitions t from location l is first computed:

$$T_{Enab}(l) = \{t \in T(G) \mid S_U(t) \subset S_{Act}(l)\}$$

Thus, it comes for l_s: $T_{Enab}(l_s) = \{t_1, t_{31}, t_{36}\}$

Then, $T_{Fire}(l)$ the set of fireable transitions from location l is calculated. This subset of $T_{Enab}(l)$ contains only transitions t for which the transition condition is compatible with the current location:

$$T_{Fire}(l) = \{t \in T_{Enab}(l) \mid E_{I_G}^{Fire(l)}(t) \neq 0\}$$

Where $E_{I_G}^{Fire(l)}(t)$ is the simplification of the transition condition of t according to l. It represents the combinations of inputs for which transition t can be fired from location l. This Boolean expression is easily obtained from the transition condition of t by substituting all steps activity variables by their corresponding value (1 or 0) according to $S_{Act}(l)$.

Thus, it comes for l_s: $T_{Fire}(l_s) = \{t_1, t_{31}\}$ since:

$$E_{I_G}^{Fire(l)}(t_1) = \text{On} \cdot \overline{\text{Off}} \qquad E_{I_G}^{Fire(l)}(t_{36}) = 0$$
$$E_{I_G}^{Fire(l)}(t_{31}) = \overline{\text{Cam}}$$

Let $SSFT(l)$ be the set of couples $(SFT, E_{SFT}^{Fire(l)})$ of subsets SFT of simultaneously fireable transitions from l and the corresponding firing condition $E_{SFT}^{Fire(l)}$. This set is defined by:

$$SSFT(l) = \{(SFT, E_{SFT}^{Fire(l)}) \mid \\ [SFT \subset T_{Fire}(l)] \wedge [E_{SFT}^{Fire(l)} \neq 0]\}$$

where

$$E_{SFT}^{Fire(l)} = \prod_{t \in SFT} E_{I_G}^{Fire(l)}(t) \cdot \prod_{t \in T_{Fire}(l) \setminus SFT} \overline{E_{I_G}^{Fire(l)}(t)}$$

For l_s, $SSFT(l_s)$ comprises three couples:

$$SSFT(l_s) = \{(\{t_1\}, \text{On} \cdot \overline{\text{Off}} \cdot \overline{\overline{\text{Cam}}}), \\ (\{t_{31}\}, \text{Cam} \cdot \overline{(\text{On} \cdot \overline{\text{Off}})}), (\{t_1, t_{31}\}, \text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}})\}$$

2) Phase 2: The situation that is reached when a given set SFT of simultaneously fireable transitions are fired from location l is defined by:

$$S_{SFT}^{l} = \left(S_{Act}(l) \setminus \bigcup_{t \in SFT} S_U(t) \right) \cup \bigcup_{t \in SFT} S_D(t)$$

Let SS_{SFT}^{l} the set of couples $(S_{SFT}^{l}, E_{SFT}^{Fire(l)})$ of situations S_{SFT}^{l} and the corresponding accessing conditions $E_{SFT}^{Fire(l)}$.

For l_s, this gives:

$$SS_{SFT}^{l_s} = \{(\{\text{F}1, 30, 33, 35, 36, 38\}, \text{On} \cdot \overline{\text{Off}} \cdot \text{Cam}), \\ (\{\text{A}1, 31, 33, 35, 36, 38\}, (\overline{\text{On}} + \text{Off}) \cdot \text{Cam}), \\ (\{\text{F}1, 31, 33, 35, 36, 38\}, \text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}})\}$$

The first term of each couple of $(S_{SFT}^{l}, E_{SFT}^{Fire(l)})$, may be a fully stable situation, a totally transient situation, or a

[5] When the sequence of firings is infinite, e.g. a cycle of evolutions from transient situation to transient situation, the Grafcet is said 'not sound' and is to be redesigned.

partially transient situation. A partially transient situation is a situation such that the combinations of inputs values can be partitioned into two subsets: a subset that lets the situation stable and a subset that drives the situation transient.

Hence, for each situation sit reachable from location l by firing the set SFT of simultaneously fireable transitions, the stability condition E_{Stab} and the transient condition E_{Tran} are to be calculated as follows:

$$E_{Stab}(sit) = E_{SFT}^{Fire(l)} \cdot \overline{\sum_{t \in T_{Fire}(sit)} E_{I_G}^{Fire(sit)}(t)}$$

$$E_{Tran}(sit) = E_{SFT}^{Fire(l)} \cdot \sum_{t \in T_{Fire}(sit)} E_{I_G}^{Fire(sit)}(t)$$

If one of these expressions is always false the situation is fully stable or totally transient; otherwise, the situation must be split into two situations, one stable and one transient, according to the subsets of inputs combinations.

From a transient situation sit, computation of the next sets of simultaneously fireable Grafcet transitions and reachable situations from sit is to be reiterated until a stable situation is reached.

For l_s, these conditions are:
$E_{Stab}(\{\mathrm{F1}, 30, 33, 35, 36, 38\}) = 0$
$E_{Tran}(\{\mathrm{F1}, 30, 33, 35, 36, 38\}) = \mathrm{On} \cdot \overline{\mathrm{Off}} \cdot \mathrm{Cam}$
$E_{Stab}(\{\mathrm{A1}, 31, 33, 35, 36, 38\}) = (\overline{\mathrm{On}} + \mathrm{Off}) \cdot \mathrm{Cam}$
$E_{Tran}(\{\mathrm{A1}, 31, 33, 35, 36, 38\}) = 0$
$E_{Stab}(\{\mathrm{F1}, 31, 33, 35, 36, 38\}) = 0$
$E_{Tran}(\{\mathrm{F1}, 31, 33, 35, 36, 38\}) = \mathrm{On} \cdot \overline{\mathrm{Off}} \cdot \overline{\mathrm{Cam}}$

3) Phase 3: For each stable situation, it is necessary to determine the emitted outputs.

A continuous action a_c is executed from a situation sit if the continuous condition of this action satisfies:

$$E_{I_G}^{Exe(sit)}(a_c) \neq 0$$

Where $E_{I_G}^{Exe(sit)}(a_c)$ is the simplification of the continuous condition of a_c according to sit. It represents the combinations of inputs for which action a_c can be executed in situation sit. This Boolean expression is easily obtained from the continuous condition of a_c by substituting all steps activity variables by their corresponding value (1 or 0) according to sit.

A continuous output o_c is emitted for a stable situation $(sit, E_{Stab}(sit))$ for the following condition:

$$E_{I_G}^{Emit(sit)}(o_c) = E_{Stab}(sit) \cdot \sum_{a_c \in A(o_c, sit)} E_{I_G}^{Exe(sit)}(a_c)$$

where $A(o_c, sit) = \{a_c \mid [s(a_c) \in sit] \wedge [o(a_c) = o_c]\}$

Let $SSEO(sit, E_{Stab})$ be the set of couples $(SEO_c, E_{SEO_c}^{Emit(sit)})$ of subsets SEO_c of simultaneously emitted continuous outputs from sit and the corresponding emission condition $E_{SEO_c}^{Emit(sit)}$. This set is defined by:

$$SSEO(sit, E_{Stab}) = \{(SEO_c, E_{SEO_c}^{Emit(sit)}) \mid \\ [SEO_c \subset O_C] \wedge [E_{SEO_c}^{Emit(sit)} \neq 0]\}$$

where

$$E_{SEO_c}^{Emit(sit)} = \\ \prod_{o \in SEO_c} E_{I_G}^{Emit(sit)}(o) \cdot \prod_{o \in O_C \setminus SEO_c} \overline{E_{I_G}^{Emit(sit)}(o)}$$

The set of emitted outputs that are controlled by stored actions is determined by analyzing the sequence of firings between two successive stable situations. When a stored action is associated to a step that belongs to a (stable or transient) situation crossed by this sequence, the corresponding output is set or reset, according to the action type. When several stored actions on the same output are sequentially executed during a sequence, only the consequence of the latter one remains.

B. SLA evolutions without firing transitions

These evolutions correspond to changes of the set of emitted outputs while staying in the same situation. They are determined by finding the sets of simultaneously emitted continuous outputs obtained from the combinations of inputs that satisfy this equation (no transition enabled for $S_{Act}(l)$ can be fired).

$$E_{Stab} = \prod_{t \in T_{Fire}(l)} \overline{E_{I_G}^{Fire(l)}(t)}$$

VI. Illustration on the example

A software tool has been developed during this work to automate the construction of the SLA of a given Grafcet by using the above-presented definitions. This tool is available at http://www.lurpa.ens-cachan.fr/isa/teloco.

It has been used for the example presented on figure 1; the SLA of this Grafcet contains 343 locations and 7813 evolutions. The construction of this SLA lasts approximately 30 s, what shows that the use of such a tool in an industrial context is quite realistic. For the studied location
$l_s = (\{\mathrm{A1}, 30, 33, 35, 36, 38\}, \emptyset, ((\overline{\mathrm{On}} + \mathrm{Off}) \cdot \mathrm{Cam}))$
only 11 evolutions to other stable locations are possible. Table I presents these evolutions, line by line. The first column gives the location that is reached when the evolution occurs, the second column contains the evolution condition (when the source location is active and this condition true, the evolution occurs), the third column the sequence of sets of simultaneously fired transitions from the source to the target location and the continuous outputs calculation. For the fourth and fifth evolutions, for instance, two sets of transitions are successively fired; from the source location, transitions t_1 and t_{31} are first simultaneously fired, leading to a transient situation, then transitions t_{34} and t_{36} are fired simultaneously and the stable locations ({F1,37,E10}, {RotateM1}, $\overline{\mathrm{Off}} \cdot \overline{\mathrm{Cam}} \cdot \mathrm{DrawIn} \cdot \mathrm{DrawOut}$) and ({F1,37,E10}, {RotateM1,OutDrawer}, $\overline{\mathrm{Off}} \cdot \overline{\mathrm{Cam}} \cdot \overline{\mathrm{DrawOut}}$) are finally reached. The difference between these two evolutions is that the evolution condition of the first one does not satisfy the continuous action condition associated to output ‘OutDrawer’, whereas the second one does.

Reachable location	Evolution condition	Sequence of SFT, Outputs changes
{A1,31,33,35,36,38}, {}, $\overline{\text{On}} + \text{Off}$	$(\overline{\text{On}} + \text{Off}) \cdot \overline{\text{Cam}}$	$\langle\{t_{31}\}\rangle$, {}
{F1,30,33,35,36,E10}, {RotateM1}, $\overline{\text{Off}} \cdot \text{Cam} \cdot \text{DrawIn} \cdot \text{DrawOut}$	$\text{On} \cdot \overline{\text{Off}} \cdot \text{Cam} \cdot \text{DrawIn} \cdot \text{DrawOut}$	$\langle\{t_1\}, \{t_{36}\}\rangle$, {RotateM1}
{F1,30,33,35,36,E10}, {RotateM1,OutDrawer}, $\overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{DrawOut}}$	$\text{On} \cdot \overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{DrawOut}}$	$\langle\{t_1\}, \{t_{36}\}\rangle$, {RotateM1,OutDrawer}
{F1,37,E10}, {RotateM1}, $\overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \text{DrawIn} \cdot \text{DrawOut}$	$\text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \text{DrawIn} \cdot \text{DrawOut}$	$\langle\{t_1, t_{31}\}, \{t_{34}, t_{36}\}\rangle$, {RotateM1}
{F1,37,E10}, {RotateM1,OutDrawer}, $\overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{DrawOut}}$	$\text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{DrawOut}}$	$\langle\{t_1, t_{31}\}, \{t_{34}, t_{36}\}\rangle$, {RotateM1,OutDrawer}
{F1,30,33,35,36,11}, {RotateM1,DownGripper}, $\overline{\text{Off}} \cdot \text{Cam} \cdot (\overline{\text{PrhDown}} + \text{PrhUp}))$	$\text{On} \cdot \overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot (\overline{\text{PrhDown}} + \text{PrhUp})$	$\langle\{t_1\}, \{t_{36}\}, \{t_{10}\}\rangle$, {RotateM1,DownGripper}
{F1,37,11}, {RotateM1,DownGripper}, $\overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot (\overline{\text{PrhDown}} + \text{PrhUp})$	$\text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot (\overline{\text{PrhDown}} + \text{PrhUp})$	$\langle\{t_1, t_{31}\}, \{t_{34}, t_{36}\}, \{t_{10}\}\rangle$, {RotateM1,DownGripper}
{F1,30,33,35,36,12}, {RotateM1,GripPce}, $\overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{GripDone}}$	$\text{On} \cdot \overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot \overline{\text{GripDone}} \cdot \text{PrhDown} \cdot \overline{\text{PrhUp}}$	$\langle\{t_1\}, \{t_{36}\}, \{t_{10}\}, \{t_{11}\}\rangle$, {RotateM1}
{F1,37,12}, {RotateM1,GripPce}, $\overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{GripDone}})$	$\text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot \overline{\text{GripDone}} \cdot \text{PrhDown} \cdot \overline{\text{PrhUp}}$	$\langle\{t_1, t_{31}\}, \{t_{34}, t_{36}\}, \{t_{10}\}, \{t_{11}\}\rangle$, {RotateM1}
{F1,30,33,35,36,13}, {RotateM1,UpGripper,GripPce}, $\overline{\text{Off}} \cdot \text{Cam} \cdot (\text{PrhDown} + \overline{\text{PrhUp}}))$	$\text{On} \cdot \overline{\text{Off}} \cdot \text{Cam} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot \text{GripDone} \cdot \text{PrhDown} \cdot \overline{\text{PrhUp}}$	$\langle\{t_1\}, \{t_{36}\}, \{t_{10}\}, \{t_{11}\}, \{t_{12}\}\rangle$, {RotateM1,UpGripper}
{F1,37,13}, {RotateM1,UpGripper,GripPce}, $\overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot (\text{PrhDown} + \overline{\text{PrhUp}}))$	$\text{On} \cdot \overline{\text{Off}} \cdot \overline{\text{Cam}} \cdot \overline{\text{DrawIn}} \cdot \text{DrawOut} \cdot \text{GripDone} \cdot \text{PrhDown} \cdot \overline{\text{PrhUp}}$	$\langle\{t_1, t_{31}\}, \{t_{34}, t_{36}\}, \{t_{10}\}, \{t_{11}\}, \{t_{12}\}\rangle$, {RotateM1,UpGripper}

TABLE I

SET OF EVOLUTIONS FROM LOCATION ({A1,30,33,35,36,38},$\emptyset$,$(\overline{\text{ON}} + \text{OFF}) \cdot \text{CAM}$) FOR THE GRAFCET MODEL PRESENTED FIGURE 1

VII. CONCLUSIONS AND PROSPECTS

A proposal to endow the Grafcet model with a formal semantics is the main contribution of this paper. On this basis, any non-timed IEC 60848 model can be translated into an equivalent formal model, a FSM with inputs and outputs, which can be used for verification purposes or to build conformance test sequences [16]. The knowledge of the SLA permits for instance to check absence of deadlock, reachability of particular locations, possibility of firing sequences or absence of conflicting outputs at every moment. Moreover, experiments have shown that this translation is fast enough to be integrated into industrial environments for automation systems development.

Further work is aiming at extending this contribution by focusing on timed systems. Timed constructs of Grafcet, timed actions and transition conditions with timed constraints on time, will be then to consider and the semantics shall be expressed by a timed extension of the SLA.

REFERENCES

[1] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen., *Systems and software verification : model-checking techniques and tools*. Springer-Verlag New York, Inc., 2001.

[2] N. Bauer, S. Engell, R. Huuck, S. Lohmann, B. Lukoschus, M. Remelhe, and O. Stursberg, "Verification of plc programs given as sequential function charts," in *Integration of Software Specification Techniques for Applications in Engineering*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3147, ch. Part V: Verification, pp. 517–540.

[3] H. Bel Mokadem, B. Bérard, V. Gourcuff, O. De Smet, and J.-M. Roussel, "Verification of a timed multitask system with UPPAAL," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 4, pp. 921 – 932, 10 2010.

[4] V. Gourcuff, O. de Smet, and J.-M. Faure, "Efficient representation for formal verification of PLC programs," in *Proceedings of 8th International Workshop On Discrete Event Systems (WODES'06)*, 2006, pp. 182–187.

[5] R. Huuck, "Semantics and analysis of instruction list programs," in *Proceedings of the Second Workshop on Semantic Foundations of Engineering Design Languages (SFEDL 2004)*, vol. 115, 2005, pp. 3 – 18.

[6] M. Rausch and B. Krogh, "Formal verification of PLC programs," in *Proceedings of the American Control Conference*, vol. 1, 1998, pp. 234 – 238.

[7] P. Lhoste, H. Panetto, and M. Roesch, "Grafcet: from syntax to semantics," *Automatique Productique Informatique Industrielle*, vol. 27, no. 1, pp. 127–141, 1993.

[8] E. Bierel, O. Douchin, and P. Lhoste, "Grafcet: from theory to implementation," *Journal Européen des Systèmes Automatisés*, vol. 31, no. 3, pp. 543–559, 1997.

[9] F. Cassez, "Formal semantics for reactive GRAFCET," *European Journal of Automation*, vol. 31, no. 3, pp. 581–603, 1997.

[10] J.-M. Roussel and J.-J. Lesage, "Validation and verification of Grafcet using state machine," in *Proceedings of IMACS-IEEE "CESA'96"*, 1996, pp. 758–764. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00353188

[11] A. Philippot and A. Tajer, "From Grafcet to equivalent graph for synthesis control of discrete events systems," in *18th Mediterranean Conference on Control and Automation*. Marrakech, Morocco: IEEE, 2010, pp. 683–688.

[12] IEC 60848, *GRAFCET specification language for sequential function charts*, 2nd ed. International Electrotechnical Commission, 2002.

[13] R. David, "Grafcet: a powerful tool for specification of logic controllers," *IEEE Transaction on Control Systems Technology*, vol. 3, no. 3, pp. 253–268, 1995.

[14] H. Guéguen and N. Bouteille, "Extensions of Grafcet to structure behavioural specifications," *Control Engineering Practice*, vol. 9, no. 7, pp. 743 – 756, 2001.

[15] IEC 61131-3, *Programmable controllers - Part 3: Programming languages*, 2nd ed. International Electrotechnical Commission, 2003.

[16] J. Provost, J.-M. Roussel, and J.-M. Faure, "Translating Grafcet specifications into Mealy machines for conformance test purposes," *Control Engineering Practice*, 2010. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00547891