Current Journal of Applied Science and Technology

Current Journal of Applied Science and Technology

38(6): 1-12, 2019; Article no.CJAST.53246

ISSN: 2457-1024
(Past name: British Journal of Applied Science & Technology, Past ISSN: 2231-0843, NLM ID: 101664541)

SE-LTL Model-checking on Timed Grafcets via ε-TPN

Médésu Sogbohossou[1*], Rodrigue Yehouessi[1], Tahirou Djara[1], Theophile Aballo[1] and Antoine Vianou[1]

[1] *Laboratoire d'Électrotechnique, de Télécommunications et d'Informatique Appliquée (LETIA), École Polytechnique d'Abomey-Calavi (EPAC), UAC, 01 BP 2009 Cotonou, Benin.*

Authors' contributions

This work was carried out in collaboration between all authors. M. Sogbohossou designed the study and wrote the first draft of the manuscript. R. Yehouessi wrote the computer codes, and performed the tests supervised by T. Djara. T. Aballo and A. Vianou made some corrective remarks, and all authors contributed to the overall achievement. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/CJAST/2019/v38i630433
Editor(s):
(1) Dr. Koji Nagata, Professor, Department of Physics, Korea Advanced Institute of Science and Technology, Daejeon, Korea.
Reviewers:
(1) Tomasz Rak, Rzeszow University of Technology, Poland.
(2) David Lizcano, Madrid Open University, Spain.
Complete Peer review History: http://www.sdiarticle4.com/review-history/53246

Received: 12 October 2019
Accepted: 15 December 2019
Published: 26 December 2019

Original Research Article

ABSTRACT

The GRAFCET standard (IEC 60848) is one of the convenient formalisms used to specify the behaviour of the automated systems. Being just a semi-formal language, the usual practice is to go through an unambiguous formalism such as time Petri net (TPN) in order to validate a specification expressed by a GRAFCET model. In this paper, we propose how to perform model-checking on a GRAFCET model translated into a ε-TPN, specifically with State-Event Linear Temporal Logic (SE-LTL). Especially, we provide a way to take into account quantitative time constraints verification by integrating observers in the ε-TPN intermediate model, since TPN state-space abstractions do not allow directly such kind of model-checking.

Keywords: grafcet (IEC 60848); Time Petri Net (TPN); model-checking; SE-LTL; observer.

**Corresponding author: E-mail: medesu.sogbohossou@epac.uac.bj;*

1 INTRODUCTION

Formal verification is essential during the conception of critical automated systems [1]. Several techniques are available for checking properties about such systems: theorem proof, test, simulation and model-checking [2]. A model-checking software takes as input the model of the system (an automaton) and the property to check on it (a formula in Temporal Logic), and aswers if this property is satisfied or not (by providing a counter-example).

The GRAFCET formalism [3] is used to describe by the means of charts the behaviour of the sequential control part of an automated system. This standard is intended for specification purposes, contrary to SFC standard [4] intended for implementation purposes.

However, GRAFCET and SFC semantics are only semi-formal, and so contain some ambiguities to clarify via a translation of the chart into another formalism, such as SMV textual language [5], timed automata [6] or time Petri nets (TPN) [7]: timed automata are used in [8] for SFC, and in [6] for GRAFCET. TPNs are structurally and historically closer to the GRAFCET than the other formalisms, and are adopted here.

A Petri net state (the marking) changes when a transition is fired by modifying token numbers in the input and output places of this transition. For a TPN, delay bounds related to each transition determine the possible relative instant of the successive firings. The work in [7] has proposed a procedure of translating a grafcet into a TPN model called ε-TPN, of which syntax is extended by ε infinitesimal delays as bounds on some transitions, allowing to simulate the synchronous semantics of GRAFCET.

In this paper, we propose how to perform model-checking on a GRAFCET chart (or *grafcet* for short) translated into a ε-TPN, specifically with the temporal logic fragment called State-Event Linear Temporal Logic (SE-LTL) [9]. Compared to the more popular fragment called CTL (Computation Tree Logic) [8], the main interest of SE-LTL is allowing integration of transition firings (i.e. events) in a property formula, offering more user comfort. Especially, we provide a way to take into account quantitative time constraints verification by defining related time events; Indeed, TPN state-space abstractions do not allow directly quantitative time model-checking. Each time event corresponds to an observer [10] (a TPN module) to add to the ε-TPN before computing the state-space abstraction.

The remainder of this paper is organized as follows. Section 2 newly defines the restricted GRAFCET formal syntax used in the context of ε-TPN. Section 3 recalls the principle of the translation of a grafcet into ε-TPN, and the state-space computing. Then, SE-LTL model-checking fragment is presented in Section 4: the application to grafcet is described, by pointing out some specificities due to ε-TPN; Practical implementation with TINA-SELT tool[1] [11] is also evoked. In Section 5, verifications about quantitative time constraints are proposed and illustrated in SE-LTL, by the means of TPN module observers to add to the ε-TPN. Finally, Section 6 summarizes this paper and gives some outlooks.

2 GRAFCET CHARTS

A GRAFCET chart [3] is a graphical representation to model the behaviour of the control part of a system. The representation consists of two parts:

- the *structure* describes the possible evolutions between the situations: a situation is the set of the active steps at a given time. The structure is composed of steps, transitions and directed links;
- the *interpretation* enables the relationship between the literal variables (inputs, outputs, delays, internal variables, ...) and the structure, by means of the transition conditions (containing inputs, rising/falling edges of boolean inputs, delays, ...) and the actions (continuous actions and stored actions).

[1]TIme petri Net Analyzer (TINA): http://projects.laas.fr/tina

2.1 Formal Syntax of a Simple Timed Grafcet

Here is newly proposed the formal syntax of a classical chart, in accordance with the restrictions used to define the translation into a ε-TPN [7]. The semantics (with the required clarifications and the related restrictions) is neither formally defined here, nor recalled: it is the one already described in [7].

The interpretation of a grafcet often involves logical expressions (called conditions) based on the literal variables and associated to transitions or continuous actions. We denote by ϕ such an expression related to a transition or a continuous action.

Definition 2.1. A grafcet is a 5-tuple $(\mathcal{S}, \mathcal{S}_{init}, \mathcal{T}, \mathcal{V}, \mathcal{A})$:

- $\mathcal{S}$ is the non-empty finite set of steps;
- $\mathcal{S}_{init}$ is the non-empty finite subset of initial steps defining the initial situation: $\mathcal{S}_{init} \subseteq \mathcal{S}$;
- $\mathcal{T}$ is the finite set of transitions; each transition $t_j \in \mathcal{T}$ is defined by a 3-tuple $({}^\bullet t_j, t_j^\bullet, \phi_j)$: ${}^\bullet t_j \subseteq \mathcal{S}$ is the non-empty set of input steps of t_j, $t_j^\bullet \subseteq \mathcal{S}$ is the non-empty set of output steps of t_j, and ϕ_j is the condition of t_j ($true$ by default);
- $\mathcal{V}$ is the non-empty finite set of literal variables: $\mathcal{V} \overset{\text{def}}{=} \mathcal{X} \cup \mathcal{I} \cup \mathcal{O} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{D}$ such as:
  - the set $\mathcal{X}$ maps bijectively to $\mathcal{S}$: for each step $S_i \in \mathcal{S}$, $X_i \in \mathcal{X}$ is the boolean variable denoting that S_i is active ($true$) or inactive ($false$) in a given situation;
  - the set $\mathcal{I}$ represents the boolean input variables; especially, for $I_k \in \mathcal{I}$, propositions $\uparrow I_k$ (rising edge of I_k) and $\downarrow I_k$ (falling edge of I_k) also state the boolean value at a given instant;
  - the set of boolean outputs $\mathcal{O} \overset{\text{def}}{=} \mathcal{O}_c \cup \mathcal{O}_s$ (with $\mathcal{O}_c \cap \mathcal{O}_s = \emptyset$): $\mathcal{O}_c$ (the *assigned* outputs) is related to continuous actions and $\mathcal{O}_s$ (the *allocated* outputs) is related to stored actions;
  - the set $\mathcal{B}$ contains the boolean internal variables (*internal* because not intended for output order) to handle by the control program;
  - the set $\mathcal{C}$ represents the counters (internal variables);
  - the set $\mathcal{D}$ represents the defined timed variables about steps: the GRAFCET conventional form of an element in $\mathcal{D}$ is either $T_1/X_i/T_2$ or T/X_i (for a step S_i, and constant time values T, T_1 and T_2).
- $\mathcal{A}$ is the set of actions: $\mathcal{A} \overset{\text{def}}{=} \mathcal{A}_s \cup \mathcal{A}_c$ (with $\mathcal{A}_s \cap \mathcal{A}_c = \emptyset$): $\mathcal{A}_s$ is the set of stored actions and $\mathcal{A}_c$ is the set of continuous actions.

A logical expression ϕ uses literal variables in $\mathcal{V}$ (except $\mathcal{O}$) composed by logical operators (usually $\neg$, $\wedge$ and $\vee$). A non-boolean variable $C \in \mathcal{C}$ is involved as a predicate of the form $C \bowtie n$ (for $\bowtie \in \{<, \geq, =, \neq\}$ and $n \in \mathbb{N}$).

Stored actions ($\mathcal{A}_s$) act on outputs $\mathcal{O}_s$ or on internal variables $\mathcal{B} \cup \mathcal{C}$ by logical or mathematical operations. The set OP of simple operations for stored actions are: set or reset a boolean in $\mathcal{O}_s \cup \mathcal{B}$, reset a counter in $\mathcal{C}$, or increment/decrement a counter in $\mathcal{C}$ by a constant integer. The instant of allocation of a stored action may be at a step activation (denoted Act) or at a step deactivation (denoted $Deact$).

Thus, any stored action $a_s \in \mathcal{A}_s$ is a 3-tuple element in $\mathcal{S} \times OP \times \{Act, Deact\}$.

Continuous actions are only performed in a stable situation, that is when no transition firing is possible before some change on a variable in $\mathcal{I} \cup \mathcal{D}$ due to the environment (inputs or time elapse). A continuous action $a_c \in \mathcal{A}_c$ sets one output in $\mathcal{O}_c$, possibly under a condition ϕ_c. However, an expression ϕ_c for a continuous action excepts event terms such as edges of inputs ($\uparrow I_k$,$\downarrow I_k$). So, $a_c \in \mathcal{A}_c$ is a 3-tuple $(S \in \mathcal{S}, O \in \mathcal{O}_c, \phi_c)$, with $\phi_c = true$ if a_c gets no condition.

It should be noticed that several actions may act on a same variable at the same time, which then allows the possibility of contradictory orders in the case of stored actions: model-checking can detect such contradictions.

2.2 An Example of Grafcet

Fig. 1 shows a grafcet edited with the software JGrafchart[2], which respects only partially the GRAFCET standard. For instance, a continuous action (resp. a stored action on activation) is defined with qualifier N (resp. S) like in the SFC standard; a timed variable T_j/X_i on a step i is denoted by $S_i.s > T_j$. Fig. 1 describes a simple traffic light system at the junction of two streets.

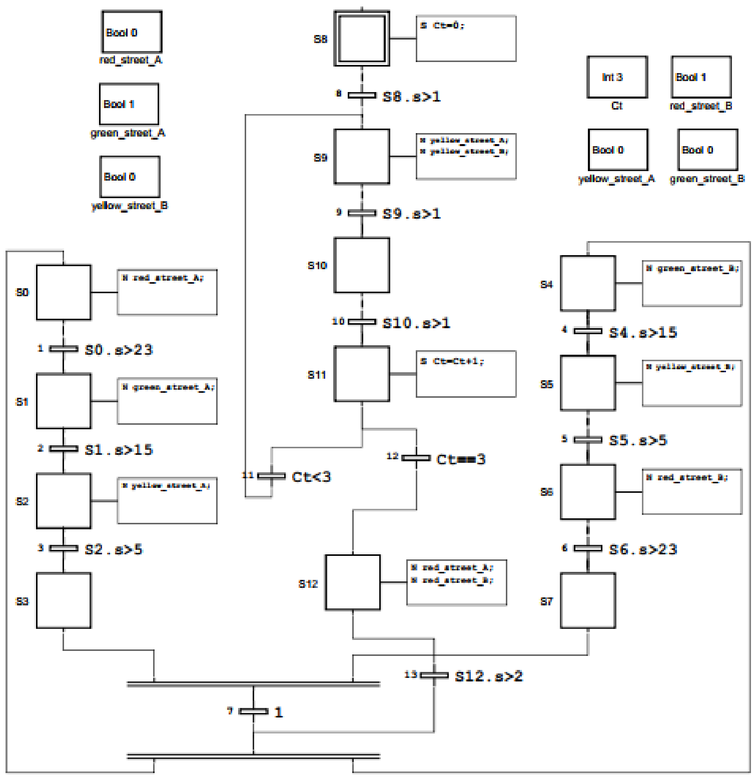


Fig. 1. Case study

3 TRANSLATION OF A GRAFCET INTO ε-TPN

The work [7] has proposed a procedure of translating a grafcet into a TPN model, of which syntax is extended by ε infinitesimal delays as bounds on some transitions, allowing to simulate the synchronous semantics of GRAFCET.

ε_0 is the infinitesimal constant delay: $\varepsilon_n \overset{\text{def}}{=} \varepsilon_0 \times (n+1)$ $(n \in \mathbb{N})$, $\mathcal{E} \overset{\text{def}}{=} \{\varepsilon_0, \varepsilon_1, \varepsilon_2\}$ and $\mathcal{E}_0 \overset{\text{def}}{=} \mathcal{E} \cup \{0\}$.

Definition 3.1. A ε-TPN [12] is a tuple $(P, T, W, W_I, W_R, ED, LD, M_0)$ such as:

- the nodes: P is the set of *places* and T is the set of *transitions* ($P \cap T = \emptyset$);

[2]JGrafchart: http://www.control.lth.se/Research/tools/grafchart.html

- the *regular arcs* and the corresponding weights between nodes, $W : P \times T \cup T \times P \longrightarrow \mathbb{N}$;
- the *read arcs*, $W_R : P \times T \longrightarrow \mathbb{N}$;
- the *inhibitor arcs*, $W_I : P \times T \longrightarrow \mathbb{N}^* \cup \{\infty\}$;
- the TPN initial marking, $M_0 : P \longrightarrow \mathbb{N}$;
- the earliest firing delays of transitions, $ED : T \longrightarrow \mathcal{E} \cup \mathbb{Q}^+$;
- the latest firing delays of transitions, $LD : \ T \longrightarrow \mathcal{E} \cup \mathbb{Q}^+ \cup \{\infty\}$;
- the set of transitions is made of three subsets, $T = T_{\mathcal{E}_0} \cup T_T \cup \{t_\infty\}$ such as:
  - ***a)*** $ED(t) = LD(t) \in \mathcal{E}_0$ when $t \in T_{\mathcal{E}_0}$;
  - ***b)*** $ED(t) = LD(t) \in \mathbb{Q}^{+*}$ when $t \in T_T$;
  - ***c)*** $ED(t) = 0$ and $LD(t) = \infty$ when $t = t_\infty$.

A transition t is *enabled* by a marking $M : P \longrightarrow \mathbb{N}$ when: $\forall p \in P, (M(p) \geq W(p,t) \wedge M(p) \geq W_R(p,t)) \wedge \nexists p \in P, M(p) \geq W_I(p,t)$. The semantics of ε-TPN is defined in [12]: besides the adopted TPN standard semantics [11], transitions in $T_{\mathcal{E}_0}$ always have priority to be fired over transitions in $T_T \cup \{t_\infty\}$.

A ε-TPN is composed with several connected modules translating the different elements constituting a grafcet.

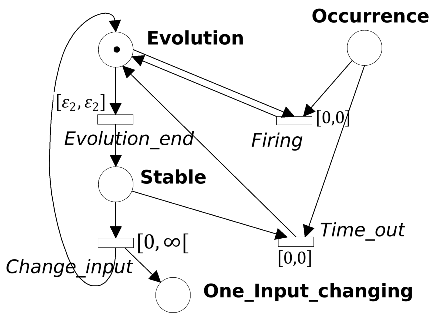


Fig. 2. Phase sequencer

A extra module, called *phase sequencer* (Fig. 2), is necessary to allow a transient evolution without modification of inputs as external events: it forces alternation between the reaction phase (called *evolution* with grafcets) and an external event production (an input change or some timed variable becomes true) in a stable situation. Indeed, the place $Evolution$ is marked during an evolution phase: transition $Firing$ is fired every time a grafcet transition model is fired (causing the marking of the place $Occurrence$) to remain in this evolution state. At the end of the evolution phase (meaning that no more grafcet transition is fireable), firing $Evolution\_end$ transition makes entering the stability phase: either an input change authorizing (firing transition $Change\_input = t_\infty$ marks the place $One\_input\_changing$, which will instantaneously change one input state modelled in some input module) or a timed variable becoming true (which causes the marking of the place $Occurrence$ and then the firing of the transition $Time\_out$).

After adding this first module, the generation of the complete TPN is done by translating sequentially: the steps, the inputs, the timed variables, the outputs, the counter variables, the continuous and conditional actions, the stored actions and the grafcet transitions. These grafcet elements (steps, transitions, input variables, actions, ...) correspond to different but connected blocks in the resulting TPN.

For the gracet at Fig.1, an illustration of the translation of some elements is given in Fig. 3-5: the output $action\_yellow\_street\_A$ with the related continuous actions (Fig. 3), the time variable on step $S8$ (Fig. 4) and the transitions 11 and 12 (Fig. 5). For each figure, related nodes in gray are already contained in a previously generated module. It should be noticed that: $1s\_X\_8\_to\_true \in T_T$ (T_T is the set of delay events for the timed variables); only one firing in the set $\{t_\infty\} \cup T_T$ triggers a reaction from a stable sate in the ε-TPN.

A translation of a grafcet (free of unbounded counters) gives a bounded ε-TPN, thereby building a finite state-space abstraction on which model-checking may be applied [12]. Such an abstraction called state class graph (SCG) is of several kinds: the one called LSCG (Linear SCG) preserves LTL properties. In the context of ε-TPN, computing LSCG copes with the state-space explosion problem, by avoiding useless interleaving of concurrent firings. Moreover, two levels of LSCG were proposed in order to abstract the informations too specific to the target ε-TPN model: the all situations state-space (denoted L1-SCG) containing all the stable and unstable situations of the grafcet, and the stable situations state-space (denoted L2-SCG) containing only the stable situations of the grafcet. L2-SCG is more compact than L1-SCG, but in the sequel we only consider L1-SCG since SE-LTL properties often include events as grafcet transition firings.

For the case study at Fig.1, we have generated the equivalent ε-TPN made of: 75 places, 96 transitions, 205 regular arcs, 110 read arcs and 20 inhibitor arcs. The corresponding L1-SCG (as an **.aut** file for TINA model-checking tool) is made of 42 states and 42 transitions.

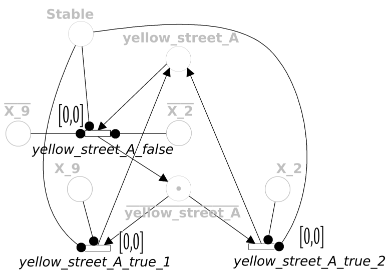


Fig. 3. Continuous action on yellow_street_A

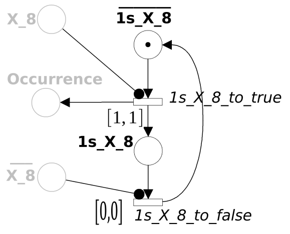


Fig. 4. Timed variable 1s/X_8

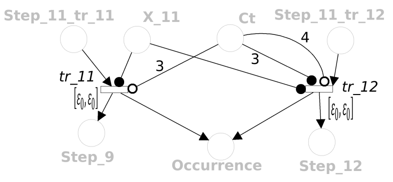


Fig. 5. Transitions 11 and 12

4 MODEL-CHECKING WITH SE-LTL

A model-checking software takes as input an abstraction of the system behaviour (a transition system such as L1-SCG in our case) and a property expressed in Temporal Logic [2] to check on the model, and answers if the abstraction satisfies or not this property. Here, we only focus on the type of temporal logic called LTL (Linear Temporal Logic) to express properties to verify on every path of the transition system. More specifically, State-Event LTL (SE-LTL) [9] allows using events (i.e. transition firings) in properties. SE-LTL is used to express properties about situations and actions of a GRAFCET chart.

4.1 Syntax and Semantics of SE-LTL

A property p verified on a model $\mathcal{M}$ is denoted: $\mathcal{M} \models p$. Here, $\mathcal{M}$ is L1-SCG such as each state is labeled with some atomic propositions (in a set AP) true in this state and a transition between two states is labeled by a subset A of events in Σ. The propositions AP concerns the marking of the different places in the ε-TPN. The set Σ includes the grafcet transition firings.

Model-checking consists in inspecting each path (or execution) of the model $\mathcal{M}$, representing a particular behaviour, to check a property. A path $\pi = (s_0, A_0, s_1, A_1, s_2, A_2, ...)$ of $\mathcal{M}$ is an alternating infinite sequence of states ($s_0, s_1, ...$ with s_0 the initial state) and events ($A_0, A_1, ...$ with $A_i \subseteq \Sigma$). Notation π^i stands for the suffix of π starting in the state s_i.

The syntax of SE-LTL path formula is given by (where p ranges over AP and a ranges over Σ) :

$$\varphi := p \mid a \mid \neg\varphi \mid \varphi\vee\varphi \mid \varphi\wedge\varphi \mid \textbf{X}\,\varphi \mid \textbf{F}\,\varphi \mid \textbf{G}\,\varphi \mid \varphi\,\textbf{U}\,\varphi$$

For the SE-LTL semantics, a path-satisfaction of formulas is defined inductively as follows ($\mathcal{L}(s_0)$ is a subset of AP labeling s_0):

1. $\pi \models p$ iff $p \in \mathcal{L}(s_0)$, and $\pi \models a$ iff $a \in A_0$,
2. $\pi \models \neg\varphi$ iff $\pi \not\models \varphi$,
3. $\pi \models \varphi_1 \vee \varphi_2$ iff $\pi \models \varphi_1$ or $\pi \models \varphi_2$,
4. $\pi \models \varphi_1 \wedge \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$,
5. $\pi \models \textbf{X}\,\varphi$ iff $\pi^1 \models \varphi$,
6. $\pi \models \textbf{F}\,\varphi$ iff $\exists k \geq 0$ s.t. $\pi^k \models \varphi$,
7. $\pi \models \textbf{G}\,\varphi$ iff $\forall k \geq 0, \pi^k \models \varphi$,
8. $\pi \models \varphi_1\,\textbf{U}\,\varphi_2$ iff $\exists k \geq 0$ s.t. $\pi^k \models \varphi_2$ and $\forall\, 0 \leq j < k, \pi^j \models \varphi_1$.

4.2 Kinds of Properties on a GRAFCET Chart

In the context of the translation into ε-TPN, state and event properties are related to TPN places and transitions respectively, modelling the grafcet elements. State propositions are about continuous actions, steps and situations, or other literal variables (such as a counter value, an active or inactive delay, an input edge, ...). Event propositions concern stored actions, grafcet transitions, or else, edges on timed variables.

Besides, some properties may depend on the specificities of ε-TPN. Indeed, some elements report the evolution phase and the stability phase of a grafcet: it is the case of the places called $Stable$ and $Evolution$, and the transition $Evolution\_end$.

Naturally, a more general property may mix up several of these types of propositions.

4.2.1 Examples of State Properties

Examples of elementary state LTL properties related to steps may be the followings:

1. to find out whether the step S_i is permanently active: **G** X_i;
2. to verify that a step S_i is never active: **G** $\neg X_i$;
3. to know if in the future the step S_i could not become permanently active: $\neg$**FG** X_i;
4. to check[3] whether the activeness of the step S_j is reachable since the one of the step S_i: **G** $(X_i \Rightarrow$ **F** $X_j)$;
5. To check that the steps S_i, ..., S_n are always activated simultaneously: **F** $((\neg X_i \wedge ... \wedge \neg X_n) \wedge$ **X** $(X_i \wedge ... \wedge X_n))$.

The meanings of $\bigvee_{X_i \in \mathcal{X}}$ **G** X_i and $\bigvee_{X_i \in \mathcal{X}} \neg$**FG** X_i are straightforward.

To express properties on continuous actions, the specific place $Stable$ in ε-TPN is useful. Let $\mathcal{X}_j \subseteq \mathcal{X}$ be the set of step states associated with the continuous action $action_j$. The proposition corresponding to $action_j$ is: $(\bigvee_{X_i \in \mathcal{X}_j} X_i) \wedge Stable$. For a conditional action $action_j$, with the condition $condition_j$ (a logical expression), that becomes obviously: $(\bigvee_{X_i \in \mathcal{X}_j} X_i) \wedge Stable \wedge condition_j$.

[3] $\phi \Rightarrow \varphi$ is equivalent to $\neg\phi \vee \varphi$.

As an example of action property (continuous or stored), to show that an action $action_2$ always follows an action $action_1$: $action_1 \Rightarrow$ **F** $action_2$.

4.2.2 Examples of event properties

About stored actions, let $T_{j_1} \subseteq T_G$ (resp. $T_{j_2} \subseteq T_G$) be the set of succeeding (resp. preceding) transitions of the steps associated with the action $action_j$. The stored action is:

- on activation : $action_j = \bigvee_{tr_i \in T_{j_2}} tr_i$,
- on deactivation : $action_j = \bigvee_{tr_i \in T_{j_1}} tr_i$.

As examples of properties with events depending on specific elements of ε-TPN (namely, transition $Evolution\_end$ and place $Evolution$):

- to check that there is never a total instability in the system: $\neg$**FG** $\neg Evolution\_end$;
- to check if there is no grafcet situation with deadlock (that is to say a situation from which no more firing of grafcet transition is possible): $\neg$**FG** $(Evolution \Rightarrow Evolution\_end)$.

This last property is equivalent to this one: **GF** $(\bigvee_{tr_j \in T_G} tr_j)$.

4.3 Illustration on Fig. 1

To implement SE-LTL model-checking with TINA software, our implementation of L1-SCG algorithm generates the corresponding automaton as a **.aut** file from the JGrafchart XML file of the grafcet. Then, TINA **ktzio** tool takes this **.aut** file as input to generate a TINA compact binary format as a **.ktz** extension file. Finally, the tool SELT of TINA is launched to check the properties on the **.ktz** file, as shown Fig. 6.

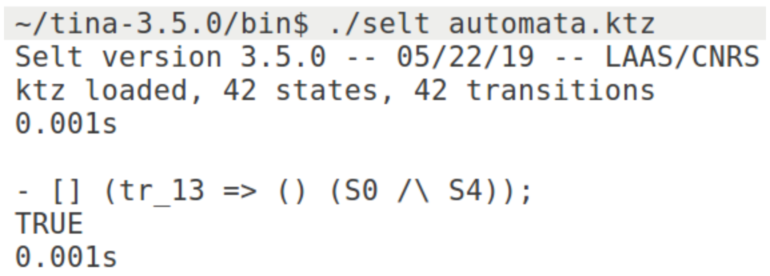
The grafcet at Fig. 1 models two traffic lights located respectively on a street A and a street B. It contains a transient mode (yellow lights blink three times) and a steady state.

Fig. 6 illustrates the following property (TRUE): the system finally leaves the transient mode (firing of transition 13) to enter the steady state, that is[4] **G** $(tr\_13 \Rightarrow$ **X** $(S0 \wedge S4))$.

These three examples of state properties are also TRUE:

- green light cannot be displayed on the two streets simultaneously: $\neg$**F** $(green\_street\_A \wedge green\_street\_B)$;
- light cannot become permanently green on the street A: $\neg$**FG** $green\_street\_A$;
- yellow lights in the transient mode will not blink more than three times: $\neg$**F** $(Ct > 3)$.

Other properties such as no deadlock and no total instability (specific to the ε-TPN) are also TRUE.



```
~/tina-3.5.0/bin$ ./selt automata.ktz
Selt version 3.5.0 -- 05/22/19 -- LAAS/CNRS
ktz loaded, 42 states, 42 transitions
0.001s

- [] (tr_13 => () (S0 /\ S4));
TRUE
0.001s
```

Fig. 6. Verification of the exit from the transient mode

[4]With SELT, operators **G**, **X** and $\wedge$ are denoted resp. $[\,]$, $(\,)$ and $/\backslash$.

5 OBSERVER INTEGRATION TO VERIFY QUANTITATIVE TIME PROPERTIES

Since quantitative time model-checking is not applicable on state-space abstractions of TPN such as SCG, here we propose an event-based method consisting in integrating observers in the grafcet model translation to allow the quantitative time verifications.

An observer [10] is a subnet added in a TPN to observe in an non-intrusive way the occurrence of some particular events about this TPN at a given date. Combined with a non-quantitative state-based and event-based temporal logic like SE-LTL, observers implanted in a ε-TPN will allow quantitative time model-checking, by considering logic propositions related to events of the observers to state on the possibility of a grafcet event or couple of events to occur or not in some time interval.

5.1 Definition of the Observers

The observers proposed here, at Fig. 7 and 8, are modifications of those given in [10], and contain delay transitions respecting the semantics of ε-TPNs; the observer at Fig. 9 is a novel modelling. Three kind of observers are considered here corresponding respectively to the following three types of quantitative time constraints to check on events of the grafcet model:

- absolute constraint applied on the j^{th} occurrence of an event evt_i, at Fig. 7: when the j^{th} occurrence of evt_i is produced at an absolute date d_i such as $d_i < \delta_{i,min}$ or $d_i > \delta_{i,max}$, transitions $OUT\_1\_evt\_i\_j$ or $OUT\_2\_evt\_i\_j$ are respectively fired. But when $\delta_{i,min} < d_i < \delta_{i,max}$, transition $IN\_evt\_i\_j$ is fired. Obviously, the value of j may impact proportionally the size of the state-space;
- relative constraint applied on every two consecutive occurrences (j^{th} and $(j+1)^{th}$) of the same event e_i, at Fig.8: when the delay d_i between any two consecutive occurrences is such as $d_i < \delta_{i,min}$ or $d_i > \delta_{i,max}$, transitions $OUT\_1\_evt\_i$ or $OUT\_2\_evt\_i$ are respectively fired. But when $\delta_{i,min} < d_i < \delta_{i,max}$, transition IN_evt_i is fired;
- relative causality constraint applied on every same order occurrence (j^{th}) of the two different events e_c (the cause) and e_e (the effect), at Fig. 9: if the delay d_{c_e} between the j^{th} instances and subsequent occurrences of e_c and e_e are produced within interval $]\delta_{c\_e,min}, \delta_{c\_e,max}[$ (and without the $(j+1)^{th}$ instance of e_c occurs before), then $IN\_evt\_c\_e$ is fired. $OUT\_1\_evt\_c\_e$ is fired whenever two consecutive firings of e_c are possible without an intermediate occurrence of e_e, and $OUT\_3\_evt\_c\_e$ is fired whenever an instance of e_e is produced before the same instance of e_c: these two cases are evidently violation of the causality between e_c and e_e. $OUT\_2\_evt\_c\_e$ and $OUT\_4\_evt\_c\_e$ are fired when the causality is fulfilled for the instance, but with $d_{c\_e} < \delta_{c\_e,min}$ and $d_{c\_e} > \delta_{c\_e,max}$ respectively. It should be noticed that the authors of [10] rather adopted a two verification steps, not practical for verification with temporal logic since two state-space models must be considered.

Thereby, the proposed observers especially allow checking a delay constraint d, with strict bounds, between a couple of events: the delay may be of the kinds $\delta_{min} < d < \delta_{max}$, $d < \delta_{min}$ or $d > \delta_{max}$. It is assumed that $\delta_{min} < \delta_{max}$, $\delta_{min} > 0$ and $\delta_{max} \neq \infty$ always hold.

An execution of an observer always follows one possible interleaving (no possible concurrency between its firings) in accordance with the triggers induced by the grafcet events. This guarantees the unequivocalness of the answer about the constraint satisfaction to check, by a firing of one of the "IN" and "OUT" transitions ("IN" for delay in the interval $]\delta_{min}, \delta_{max}[$, "$OUT$" for delay out this interval) in the observer.

Definition 5.1. T_O is defined as the set of transitions in the observers with intervals of the form $[\delta_{min}, \delta_{min}]$ or $[\delta_{max} - \delta_{min}, \delta_{max} - \delta_{min}]$.

T_O is similar to the set T_T. So, the extension of ε-TPN definition to integrate observer modules is straightforward. T_O firings are only possible in a stable state, concurrently with transitions in $T_T \cup \{t_\infty\}$.

The grafcet events which may trigger an observer for doing the model-checking on a delay constraint are the followings, for the L1-SCG: firings in T_T related to timed variable modellings, t_∞ (any input change) or firings in T_{input} (particular input changes) and firings in T_G (grafcet transition modellings).

Atomic propositions involving "IN" and "OUT" transitions of the observers is then usable in temporal expressions to state about quantitative time properties. In practice, a list of the events in the ε-TPN to observe should be defined, for instance in a specific file. Then, the corresponding observers will be added to the ε-TPN before computing L1-SCG automaton (the **.ktz** file for TINA-SELT). These events (numbers and corresponding transitions) with their time interval may be defined as lines of the file:

- absolute constraint about the j^{th} occurrence of evt_i (Observer of type 1):

$$evt\_i/j : transition\_i, \delta_1, \delta_2$$

- relative constraint about two consecutive occurrences of evt_i (Observer of type 2):

$$evt\_i : transition\_i, \delta_1, \delta_2$$

- relative causality constraint about two events evt_c and evt_e (Observer of type 3):

$$evt\_c\_e : transition\_c, transition\_e, \delta_1, \delta_2$$

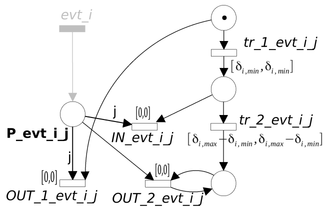


Fig. 7. Observer 1

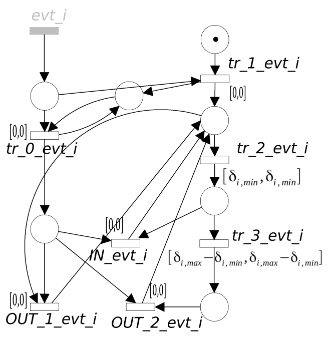


Fig. 8. Observer 2

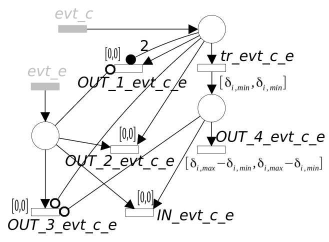


Fig. 9. Observer 3

5.2 Examples of model-checking on time events

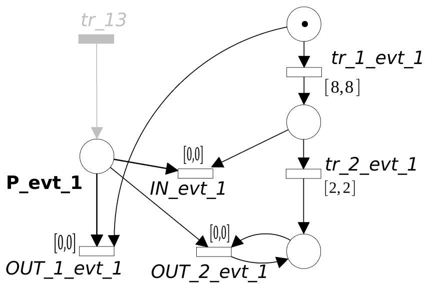
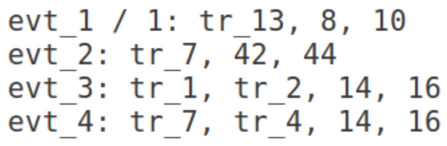


Fig. 10. Observer 1 for the first property (P1)



```
evt_1 / 1: tr_13, 8, 10
evt_2: tr_7, 42, 44
evt_3: tr_1, tr_2, 14, 16
evt_4: tr_7, tr_4, 14, 16
```

Fig. 11. List of timed events

The tool SELT of TINA is used (as in Fig. 6) to check the following three properties on the **.ktz** file:

(P1) the transient mode lasts between 8 and 10 seconds (observer of type 1): event evt_1 on the firing tr_13 (with $j = 1$). To check this property, the related observer is shown at Fig. 10;

(P2) the global cycle of the lights (in the steady state) lasts between 42 and 44 seconds (observer of type 2): event evt_2 on the firing tr_7;

(P3) the green lights do not last less than 14 seconds (observer of type 3):

- on street A: event evt_3 with transitions tr_1 (cause) and tr_2 (effect);
- on street B: event evt_4 with transitions tr_7 (cause) and tr_4 (effect).

So, the list of events to consider is given at Fig. 11.

Formally, the properties to check are respectively:

(P1) **G** $(evt\_1 \Rightarrow$ **F** $IN\_evt\_1)$

(P2) **G** $(evt\_2 \Rightarrow$ **F** $IN\_evt\_2)$

(P3) **G** $(evt\_3 \Rightarrow$ **F** $\neg\ OUT\_2\_evt\_3) \wedge$ **G** $(evt\_4 \Rightarrow$ **F** $\neg\ OUT\_2\_evt\_4)$

These properties are TRUE after model-checking with TINA-SELT tool.

6 CONCLUSION

This paper shows how to check SE-LTL properties on a GRAFCET chart after the translation into an equivalent ε-TPN, and subsequently into an automaton representing the state-space. To take into account properties about quantitative time constraints, observers related to the time events appearing in SE-LTL formulas have to be implanted into the ε-TPN previously, before generating the state-space. SELT-TINA model-checking tool is used for the possibility to experiment the verifications.

A short-term perspective may be to make expressing grafcet LTL properties more independent of the knowledge of the ε-TPN structure, by acting on the resulting SCG before model-checking. Similarly, time events in formulas may be made more user-friendly. Moreover, creating a software with a convivial interface (to be independent of TINA tools) will be another step before to consider an extensive evaluation on a great number of grafcets, with the related complexity analyses.

The subset of temporal logic called SE-LTL is chosen here firstly, because allowing the facility of using events as propositions in a property formula and secondly, because SELT-TINA is a unique tool to perform event-based model-checking tests with TPNs.

Other subsets of temporal logic may be considered for future developments. On the one hand, UCTL (Action/State-Based Temporal Logic) [13] not only takes into account event propositions, but also branching in CTL (Computation Tree Logic) seems more valuable in model-checking practices [8]. On the other hand, quantitative time model-checking in a temporal logic such as TCTL (Timed CTL) [14, 15]) is applicable to TPN [16] via a structural translation to timed automata [17]: this method must be adapted for ε-TPN.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

[1] Darvas D, Majzik I, Viñuela EB. PLC program translation for verification purposes. Periodica Polytechnica, Electrical Engineering and Computer Science. 2017;61:151-165.

[2] Clarke EM, Henzinger TA, Veith H. Introduction to model checking. Springer International Publishing, Cham. 2018;1-26.

[3] IEC 60848. Grafcet specification language for sequential function charts. Technical Report, International Electrotechnical Commission; 2013.

[4] IEC 61131-3. Programmable controllers - part 3: Programming languages. Technical Report, International Electrotechnical Commission; 2013.

[5] Ovatman T, Aral A, Polat D, Ünver AO. An overview of model checking practices on verification of PLC software. Softw. Syst. Model. 2016;15(4):937-960.

[6] L'Her D, Le Parc P, Marcé L. Proving sequential function chart programs using timed automata. Theoretical Computer Science. 2001;267(1-2):141-155.

[7] Sogbohossou M, Vianou A. Formal modelling of grafcets with time petri nets. IEEE Transactions on Control Systems Technology. 2015;23(5):1978-1985.

[9] Chaki S, Clarke EM, Ouaknine J, Sharygina N, Sinha N. State/event-based software model checking. In IFM, Springer. 2004;2999:128-147.

[8] Bauer N, Engell S, Huuck R, Lohmann S, Lukoschus B, Remelhe M, Stursberg O. Verification of PLC programs given as sequential function charts. Integration of Software Specification Techniques for Applications in Engineering. 2004;517-540.

[10] Toussaint J, Simonot-Lion F, Thomesse JP. Time constraint verification methods based on time petri nets. Proceedings of the Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems. 1997;262-267.

[11] Berthomieu B, Vernadat F. Time Petri nets analysis with TINA. In Quantitative Evaluation of Systems, QEST 2006. IEEE. 2006;123-124.

[12] Sogbohossou M, Vianou A. ε-TPN: definition of a Time Petri Net formalism simulating the behaviour of the timed grafcets. ARIMA Journal, Special Issue CARI 2018. Nabil Gmati, Eric Badouel, Bruce Watson, Eds., 2019;31:1-22.

[13] ter Beek MH, Fantechi A, Gnesi S, Mazzanti F. An action/state-based model-checking approach for the analysis of communication protocols for service-oriented applications. In FMICS. Springer. 2007;133-148.

[14] Alur R, Courcoubetis C, Dill D. Model-checking in dense real-time. Information and Computation. 1993;104(1):2-34.

[15] Markey N, Schnoebelen P. TSMV: A symbolic model checker for quantitative analysis of systems. In QEST. 2004;4:330-331.

[16] Boucheneb H, Gardey G, Roux OH. TCTL model checking of time petri nets. Journal of Logic and Computation. 2009;19(6):1509-1540.

[17] Cassez F, Roux OH. Structural translation from time petri nets to timed automata. Journal of Systems and Software. 2006;79(10):1456-1468.

©2019 Sogbohossou et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/53246