I.J. Information Technology and Computer Science, **2014, 10, 1-10**
Published Online September 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.10.01

Linear Hybrid Automaton Generation Using Mapping Algorithm for Hybrid Dynamic Systems

Sekhri Larbi
Industrial Computing and Networking Laboratory, Computer Science Department, University of Oran, BP 1524 Oran, Algeria
Email: larbi.sekhri@univ-oran.dz

Haffaf Hafid
Industrial Computing and Networking Laboratory, Computer Science Department, University of Oran, BP 1524 Oran, Algeria
Email: haffaf.hafid@univ-oran.dz

Abstract—Hybrid dynamic systems are analyzed through linear hybrid automaton. In this paper, we propose a mapping algorithm to deal with a new *Continuous elementary HPN*. The method shown enables us to analyze some system properties using a linear hybrid automaton generated by a mapping process. The application involves a water system of three tanks, which is analyzed by a *PHAVer* (Polyhedral Hybrid Automaton Verifier) software tool. Its effectiveness is illustrated by numerical simulation results.

Index Terms— Hybrid Dynamic System, Hybrid Petri- Nets, Evolution Graph, Linear Hybrid Automaton

I. Introduction

Hybrid dynamic systems (*HDS*) are a class of reactive systems. Modeling such systems and verifying their behavior are current research topics in both the automatic control community and the computer science community. *HDS* concern all industrial domains: automated production systems, traffic systems, energetic systems, telecommunications, etc. *HDS* systems coexist with each other, as well as with discrete behavior and continuous behavior. A system is then characterized by the nature of its state's variables, which can be continuous or discrete.

Many research, such as new specification languages, tools and models have been developed for modeling and analyzing *HDS*. These works may concern efficient performance in monitoring, control, diagnosis, and structural properties analysis; these studies often rely on mathematical models [1] and [2] or graphical models [3].

In order to model complex systems, formal specifications are needed. The latter are subject to validation software tools that are able to check system properties [4], [5], [6], [7], [8], [9], [10] and [11]. In the HDS domain, there are three kinds of models:

The hybrid bond graph model expands on the continuous model, similar to bond graph models, by including specific elements such as switch elements to represent physical TOR transitions (also known as binary transitions) [12], [13] and [14].

- Extensions of discrete models, such as continuous Petri nets, consider marking as real number and transition firing as a continuous process [15]. Examples of such extensions include Hybrid Petri nets (*HPN*) which are composed of a discrete part and a continuous part [16]. Hybrid systems modeling results in *HPN* inheriting all advantages known for Petri nets. In [17], *HPN* are used to model a traffic network control, considered a hybrid system. Another model, the hybrid state Petri net (*HSPN*), offers the analyzing capacity of stochastic hybrid systems as well, the automata formal verification power is proposed in [18].
- Mixed models; that are based on collaboration between two sub-models in the same structure. The first models discrete event aspects, and is generally based on the finite state automaton or on the classical discrete Petri net. The second is based on state equations representation or any other continuous model to describe the continuous part of the *HDS*. Hybrid aspects are taken in the interface between the sub-models [19], or by combining Bond-graph and (Max, +) algebra [20]. Among these models we can cite hybrid automatons which function alternatively between continuous steps (involving state variables and continuous time evolutes) and discrete steps (in which many discrete transition can be fired). The Hybrid automaton is an extension of timed automaton where the continuous dynamics model is represented by means of differential equations.
- For monitoring purposes, the improvement of the decision-making step is realized through a Hybrid Bayesian Network (*HBN*) model using a hybrid inference procedure. Time is seen as a continuous variable and its evolution interacts with discrete transitions. The automaton includes clocks whose values belong to the set of real numbers, and continuously increase [21]. Another model of timed automata has been proposed in [22] for the supervisory control. The method presented in this paper extends Sava's work [22]. Sava proposed an approach to build the timed automaton (*TA*) which models the exact

behavior of a discrete event system (*DES*) modeled by a time Petri net (*TPN*). The forbidden states of *DES* are modeled by forbidden timed automaton locations.

Reachability analysis is one of the major problems encountered in verifying the properties of *HDS* as modeled by the hybrid automaton. In this work, we are interested in modeling a particular class of *HDS*: Systems with continuous flow (materials flow or products flow) supervised by discrete event systems.

This paper has two aims: first, to formally present a new class of *HPN* called *Discrete Continuous Elementary HPN* (*DC elementary HPN*) and, second, to propose a systematic method, based on mapping algorithms, to build a linear hybrid automaton (*LHA*) from an evolution graph of a *DC-elementary HPN*. Properties analysis of *LHA* is evaluated by the *PHAVer* (*Polyhedral Hybrid Automaton Verifier*) software tool developed at *VERIMAG* laboratory of Grenoble (France) [23].

The paper is organized as follows: section 2 introduces some formal definitions (*HDS, HPN* and *DC-elementary HPN)*. In section 3, a formal definition of linear hybrid automaton (*LHA)* is given and a mapping algorithm is presented. Section 4 offers a case study which consists of a system of three water tanks. In this section the mapping algorithm is applied and a detailed simulated analysis is performed using *PHAVer*. Finally, we conclude our work and discuss some potential areas of future research.

II. Fomal Definitions

A. Introduction

Hybrid Petri nets (*HPN*) are developed from continuous Petri nets [24]. *HPN* are characterized by the interaction of two major components: *Discrete Part and Continuous Part.* The Discrete part models logical functionalities while the continuous part models continuous phenomenon. *HPN* presented in previous research literature are formed by two kinds of Petri nets: Discrete timed Petri nets, and continuous Petri nets, which function at a higher speed. Elementary *HPN* are a class of *HPN* where no transformation of marking exists between discrete and continuous parts. Elementary *HPN* combining the time Petri net [25] and the continuous Petri net with constant speed are called *D-elementary HPN*. In this model only the discrete part can influence the continuous part [26] and [27]. In this section we define an extension of *D-elementary HPN* known as *DC-elementary HPN* where the continuous part can also influence the discrete part, and vice versa. This is motivated by two facts: first, in real systems, the continuous part has a direct influence on the discrete part. For instance, when a threshold is reached after introducing a continuous steady state variable, it could stop or automatically trigger buttons that correspond to Boolean variables. In this case, each value would correspond to continuous values in an interval, thus providing discrete state transition. In these transitions, problems can arise concerning initial values in the new state, referred to as the "warm" initialization. The second motivation is that for hybrid systems, two models are always required, where the discrete model governs the transition between the continuous models. With this under consideration, our approach follows a recent trend towards a unified and unique model. In the remainder of the paper, we assume the reader familiar with formal definitions of continuous systems and discrete event systems.

B. Hybrid dynamic systems

Formally a *HDS* is a 5-tuple**:** $HDS= (T, (x, q), (x_0, q_0), U_c \cup U_d, \Phi)$ [28]:

- $T \in \Re$ *is the time interval*
- $(x, q) \subseteq X \times Q$ *represents the complete state of hybrid system*
- (x_0, q_0) *is the set of initial states*
- $U_c \cup U_d$ *is the set of continuous commands and discrete commands*
- $\Phi : X \times Q \rightarrow \Re^n$ *defines a sub-set of trajectories for each discrete state*

C. Hybrid Petri net

Formally, a marked hybrid Petri net is an 8-tuple: $HPN=< P; T; h ; Pre; Post; Tempo; V; M_0 >$ *where:*

$P= \{P_1, P_2, \ldots, P_n\}$ *finite set of places*

$T= \{T_1, T_2, \ldots, T_m\}$ *finite set of transitions*

$P \cap T=\emptyset$

$h: P \cup T \rightarrow \{D, C\}$ *hybrid function indicating for each node if it is a discrete node* (P^D, T^D) *or a continuous node* (P^C, T^C) *of HPN.*

$Pre: P \times T \rightarrow Q^+$ *(if* $P_i \in P^C$*) or N (if* $P_i \in P^D$*) is the forward application*

$Post: P \times T \rightarrow Q^+$ *(if* $P_i \in P^C$*) or N (if* $P_i \in P^D$*) is the backward application*

$Tempo: T^D \rightarrow Q^+$ *application associating to each D-transition* T_j *their duration*

$V: T^C \rightarrow R^+$ *application associating to each C-transition* T_j *their maximal firing speed* V_j

$M_0: P \rightarrow R^+$ *(if* $P_i \in P^C$*) or N (if* $P_i \in P^D$*) is the initial marking*

Pre and Post applications must satisfy the following condition:

if P_i *and* T_j *with* $h(P_i) = D$ *and* $h(T_j) = C$ *then* $Pre (P_i; T_j) = Post (P_i; T_j)$

This condition enables the marking of a *D-place* stay integer in the evolution of *HPN.*

D. DC-elementary hybrid Petri net

Formally, a *DC-elementary HPN* is defined as 9-tuple $<P; T; h; Pre; Post; I; SIM; V; M_0>$ *where*:

$P= \{P_1, P_2, \ldots, P_n\}$ *finite set of places*

$T= \{T_1, T_2, \ldots, T_m\}$ *finite set of transitions*

$P \cap T=\emptyset$

$h: P \cup T \rightarrow \{D, C\}$ *hybrid function indicating discrete nodes* (P^D, T^D) *and continuous nodes* (P^C, T^C)

$Pre: P \times T \rightarrow Q^+$ *(if* $P_i \in P^C$*) or N (if* $P_i \in P^D$*) is the forward application*

Post: $P \times T \rightarrow Q^+$ *(if* $P_i \in P^C$*) or N (if* $P_i \in P^D$*) is the backward application*

Pre and Post satisfy the following condition:

$\forall (P_i,T_j) \in (P^D \times T^C) \cup (P^C \times T^D)$ *then Pre* $(P_i, T_j) = (P_i, T_j)$

I: $(P_i, T_j) \rightarrow R$ *Inhibition application*

SIM: $T^D \rightarrow Q^+ \times (Q^+ \cup \infty)$*: application associating with each discrete transition* T_j *their firing static interval* $[\alpha_j, \beta_j]$*,*

V: $T^C \rightarrow R^+$ application associating with each continuous transition T_j their maximal firing speed V_j

M_0*:* $P \rightarrow R^+$ *(if* $P_i \in P^C$*)* or *N (if* $P_i \in P^D$*)* is the initial marking.

III. From Evolution Graph to Linear Hybrid Automaton

A. Introduction

The state of a linear hybrid automaton (*LHA)* is given by a couple *E=(s, v)* where *s* is a summit and *v* is a valuation, defining the values of the variable *x* at *t* moment. This state can change either by discrete transition firing, or by time passing in the same summit. Contrary to *HPN*, hybrid automata (*HA*) are difficult to integrate into modeling systems. *HPN* have the advantage of clearly and efficiently modeling systems without presenting an exhaustive enumeration of the state space; however, there is no simulation software that can directly handle *HPN*. There are many algorithms in research literature which translate different classes of *HPN* into hybrid automaton [29], [30], [31], [32] and [33]. In [29] an algorithm with the ability to build a hybrid automaton equivalent to *HPN* is presented. The resulting automaton has the same number of summits as the IB- state (Invariant Behavior-state) number of *HPN*. The state variables represent the marking of continuous places and the clocks measure the time of enabled transitions. The obtained automaton is linear and deterministic. In order to ensure the convergence of this mapping algorithm, the *HPN* must be bounded. Sava's algorithm translates a time Petri net into timed automaton by determining for each reachable summit *L*, reachable space, the enabled discrete transitions and clocks [31]. Expanding on ideas presented in [29] and [31], Ghomri, in [26], proposes an algorithm which translates a *Discrete-elementary HPN* into hybrid automaton. The algorithm uses two main steps in its mapping process. At first, it translates the time Petri net into a timed automaton; at the second stage, it maps the continuous constant speed Petri net into a hybrid automaton. The summit number of the hybrid automaton resulting from this algorithm is less or equal to $N.2^n$, where *N* represents the macro-summits number of timed automaton that corresponds to the discrete part, while *n* is the C-places number of the D-elementary *HPN*. However, in practice we are not obligated to explore all the possibilities, as some do not possess physical realizations.

Based on Sava's work, El-Touati et al. propose an algorithm that translates extended time Petri nets into linear hybrid automaton (*LHA*). The obtained temporal behavior of *LHA* is similar to the behavior of the extended temporal Petri net since the summit's number is equal to marking number of the marking graph.

Many problems concerning properties analysis of hybrid automaton are expressed as reachability problem, which are in turn, not definitive. In order to reduce this complexity, strong restrictions are imposed, in order to obtain specific classes of hybrid automaton to which existing tools could be applied. The existence of software tools that enable the resolution reachability analysis for some classes of hybrid automaton leads to the hybrid systems analysis through a mapping process.

B. Linear hybrid automaton

A linear hybrid automaton (*LHA*) is defined by *A= (X, S, δ, Inv, dyn, λ, a, guard, aff) where:*

$X \subseteq R^n$ *is a finite set of variables.*

S is a finite set of summits.

δ is a finite set of synchronization labels

Inv is a function associating for every s $\in$ *S an invariant Inv (s).*

dyn: $S \times X \rightarrow Q$ *is a function describing the evolution of variables in each summit.*

λ is a finite set of transitions, each transition e=(s, s') $\in$ *λ identifies a starting summit s* $\in$ *S and an arrival summit s'* $\in$ *S.*

a $\in$ *δ is a synchronization label associated to transition e=(s, s').*

guard is a function that associates to each transition e=(s, s') a predicate C_e *called guard. If* C_e *is verified then the transition e=(s, s') is executed.*

aff is a function that associates to each transition e=(s, s') an assignment relation.

C. Mapping algorithm

In this section, a translation algorithm for *DC-elementary HPN* is proposed. As input, this algorithm accepts an evolution graph of a *DC-elementary HPN* and gives as output a linear hybrid automaton (*LHA)*. The *LHA* is more compact than an evolution graph, which facilitates the analysis activities. We have developed a software tool using C++ programming language to implement our algorithm. In the rest of this paper, the notation given below is adopted:

L: set of summits,

T: transitions set {T^D: temporal set and T^C: dynamic},

T^d_c: set of temporal transitions depending on marking of continuous places of HPN,

$M = \{M^c, M^d\}$*:* set of markings,

D: set of dynamic continuous places of *HPN*,

N: summits counter,

P: stack saving (memorizing) visited summits not analyzed. Each element of the stack memorizes the summit name, dynamics of continuous place in the summit and the active clocks in this summit.

The main steps of our algorithm can be summarized as follows:

- For each enabled transition in L_{n-1} create a summit L_n,
- Define D_n (dynamics of continuous places in L_n),
- Define active clocks in L_n and compute invariant I_d (L_n) for each enabled $T_j \in T^d$,
- Define a dynamic transition T_j^C and compute invariant $I_c(L_n)$ from firing interval of T_j^C for each enabled $T_j \in T^d_c$ and associate a null clock to T_j *in* L_n,,
- Define a dynamic transition T_{ci} (p_i^c) and compute $I_{ci}(L_n)$ for each p_i^c with $D_n(p_i^c) < 0$,
- Create a transition, $T_{n-1,n}=(L_{n-1}, g_{n-1,n}, A_{n-1,n}, L_{n-1,n})$ and save in P the visit of $(L_n, T_{n-1,n})$.

The algorithm is concluded when all the summits have been visited.

Step 1: Initialization (creating of initial summit L_0)

Initially, all continuous places are marked and consequently all transitions are fired to their maximum speed. From the initial discrete marking, we determine the fireable transitions set. So, we create the initial summit of the hybrid automaton by associating the activity $\dot{M} = Wc.V$ enabling the computation of the dynamics of the C-places (W: incidence matrix; V: maximal firing speed vector). Then we determine the enabled transitions set following the first step of our algorithm.

Algorithm for *LHA* generation

Algorithm of Step 1: Initialization

Begin

1. *Let M_0 $\{M^d, M^c\}$ the initial marking of DC-elementary HPN.*
2. *Create a summit L_0 associated to initial marking,*
3. *Determine D_0 dynamics of continuous places in summit L_0,*
4. *Determine validated transitions by marking M_0^d*
5. ***If** $T_j \in T^d$ **then***
6. *Determine active clocks in summit L_0*
7. *Compute the invariant $I_d(L_0)$ from intervals of validated firing transitions in the marking M_0^d*
8. ***endif***
9. ***If** $T_j \in T^d_c$ **then** Tj is fireable if $M(P^c_i) = S$ (S_{max} or S_{min})*
10. *Associate a nil clock to T_j in summit L_0*
11. *Determine a dynamic transition T_j^C*
12. *Compute his invariant $I_c(L_0)$ from firing interval of T_j^C*
13. *Update $I(L_0)=I_d(L_0) \cup I_c(L_0)$*
14. ***endif***
15. ***For each** i=1 to p **do***
16. ***If** $D_0(p_i^c) < 0$ and $I_{ci}(L_0)$ $(p_i^c) = \emptyset$ **then***
17. *efine a dynamic transition T_{ci} (p_i^c)*
18. *Compute $I_{ci}(L_0)$ of $T_{ci} = Mp_i^c >= 0$*
19. *$I(L_0)=I(L_0) \cup I_{ci}(L_0)$,*
20. ***endif***
21. ***endfor***
22. *Create an entry transition by associating an affectation updating all active clocks to zero in summit L_0 and the continuous initial markings.*
23. *Memorize this visit of summit L_0 in the stack P : $(L_0, T_{0,0})$*
24. *Actualize the sets $M := \{M_0\}$; $D := \{D_0\}$; $L := \{L_0\}$; $T = \{T_{0,0}\}$; $n := 1$.*

End.

Algorithm of Step 2: Analyze the last visit saved in the stack (L_n summit)

Begin

1. *The analysis of summit L_n by firing of $T_{m,n}$*
2. *Take out from the stack element memorizing this visit*
3. *Determine the enabled transitions set*
4. ***For each** firing of T_j **do** determine M^d_{n+1} and D_{n+1}*
5. ***if** $M^d_{n+1} \notin M^d$ **then** go to step 3 **else** go to step 4*
6. ***endif***
7. ***endfor***

End.

Algorihm of Step 3: Creating summit L_{n+1} associated to reachable M_{n+1} marking

Begin

1. *Create summit L_{n+1} associated to reachable marking M_{n+1}*
2. *Determine dynamics of all continuous marking in summit L_{n+1} :D_{n+1}*
3. *Determine validated transitions in marking M^d_{n+1}*
4. ***if** $T_j \in T^d$ **then***
5. *Determine active clocks in summit L_{n+1}*
6. *Compute the invariant I_d (L_{n+1}) from the interval of validated transitions in marking M^d_{n+1}*
7. ***endif***
8. ***if** $T_j \in T^d_c$ (S is equal to S_{max} or S_{min}) **then***
9. ***if** the fired transition $T_{n,n+1}=T^C_j$ **then***
10. *Activate the clock of T_j*
11. *Compute the invariant $I_{dj}(L_{n+1}) = I_{dj}(L_{n+1})$*
12. *Update $I_d(L_{n+1})=I_d(L_{n+1}) \cup I_{dj}(L_{n+1})$*

else

13. *Associate a nill clock to Tj in summit L_{n+1}*
14. *Determine a dynamic transition T^C_j*
15. *Compute his invariant $I_{ci,}(L_{n+1})$ from the firing condition $C_{T^c_j}$, obtained from the dynamic $D_{n+1}(p_i^c)=k$, ($k=cst$), his continuous place p_i^c, and the firing interval of T_j . $I(L_{n+1})=I_d(L_{n+1}) \cup I_{ci,}(L_{n+1})$*
16. ***Endif***
17. ***endif***
18. ***For** i=1 to p **do***
19. ***If** $D_{n+1}(p_i^c) < 0$ and $I_{ci}(L_{n+1}) = \emptyset$ **then***
20. *Define a dynamic transition $T^C(p_i^c)$*
21. *Compute $I_{ci}(L_{n+1})$ equal to $Mp_i^c >= 0$*
22. *$I(L_{n+1})=I(L_{n+1}) \cup I_{ci}(L_{n+1})$,*
23. ***endif***
24. ***endfor***
25. *Create a transition $T_{n,n+1}=(L_n, g_{n,n+1}, A_{n,n+1}, L_{n+1})$ modeling the firing of T_j.*
26. *Add his guard $g_{n,n+1}$ equal to: $[a_j, d_j]$ if $T_j \in T^D$; $[b_i, c_i]$ if $T_j \in T_j^C$; $Mp_i^c=0$ if $T_j \in T_{ci}$ $[a_j, d_j]$ and $Mp^c_{i,Tj,,}=S$ if $T_j \in T^d_c$*

27.
28. Save in stack P the visit of summit L_{n+1} *:* $(L_{n+1}, T_{n,n+1})$
29. Actualize the sets:$M:=M\cup\{M_{n+1}\}$, $D:=D\cup\{D_{n+1}\}$, $L:=L\cup\{L_{n+1}\}$, $T:=T\cup\{T_{n,n+1}\}$, $n:=n+1$.
End.

Algorithm of step 4: Associating transition to an existed summit
Begin
1. ***if*** $M^d_{n+1}= M^d_L$ ***then***
2. ***if*** $D_{n+1} = D_L$ ***then***
3. *Create a transition* $T_{n,l}=(L_n, g_{n,l}, A_{n,l}, L_l)$,
4. *Add his guard* $g_{n,l}$,
5. $T:=T\cup\{T_{n,l}\}$
6. ***else*** *go to* ***step 3***
7. ***endif***
8. ***endif***
9. ***If*** $P\neq\emptyset$ ***then*** *go to* ***step 2.***
10. ***endif***
End.

***Remark** 1:* This algorithm converges for bounded DC elementary HPN. The algorithm concludes when the stack to analyze becomes empty and when all summits are created. The visit of summit is totally characterized by the invariant behavior state (*IB-state*) of the DC-elementary HPN.

A *DC-elementary HPN* is made up by a T-temporal Petri net and a continuous Petri net with constant speed. Many algorithms in previous research have demonstrated a convergence that is due to the bounded aspect of the T-temporal Petri net [31]. The translation of continuous Petri net with constant speed in a hybrid automaton is always a convergent process, even if this Petri net is not bounded. This is due to the fact that we characterize a place not by his marking but by a macro-marking. The *LHA* associated to the *DC-elementary HPN* then has a finite set of summits. The visit of a summit corresponds to a transition execution. If the summit is not yet created in the stack, then we save the summit and the visit together in the stack, or we create the summit without the visit in the stack. Since the summit's number is finite, the transition's number is also finite.

IV. Illustrative Example

A. Case study: Three water tanks system

The method proposed in this paper will be illustrated with a three- tanks system presented in Fig. 1. This is a pedagogical example, in which all kinds of interactions between discrete and continuous parts in the same system. We assume the reader has a general familiarity with continuous Petri nets [15]. In Hybrid Petri nets, continuous places are modeled by double circles and continuous transitions by rectangles.

Let us consider the system of three tanks illustrated by the Fig.1. The flow rates 2, 5, 3, 6 and 7 litres/sec are associated to valve1, valve 2, valve 3, valve 4 and valve 5 respectively. This system is modeled by a DC-elementary HPN illustrated by the Fig. 2. Tanks are represented by continuous places P_5, P_6 and P_7 while the maximum values of speed V_5, V_6, V_7, V_8 and V_9 are associated to continuous transitions T_5, T_6, T_7, T_8 and T_9 respectively.

m_1, m_2 and m_3 represent the initial marking of places P_5, P_6 and P_7 respectively, v_5 (t), v_6 (t), v_7 (t), v_8 (t) and v_9 (t) represent the firing speed of transitions T_5, T_6, T_7, T_8 and T_9 respectively. Valve3 (Valve4) has two discrete working modes (stop, work) and is modeled by two discrete places P_1 and P_2 (P_3 and P_4).

Transition from the open state to the closed state of Valve3 (Valve4) takes 3t.u to 5t.u, therefore the time interval [3, 5] is associated with discrete transitions T_2 and T_3. This transition is constrained by the water level in tank1 (tank2) modeled by the valuation in the arrow $P_5\rightarrow T_2$ ($P_6\rightarrow T_3$). On the other hand, the passage from the closed state to the open state of Valve3 (Valve4) takes place after 10 t. u. from the last opening action, therefore the time interval [10, 10] is associated with discrete transition T_1 and T_4. $P_2\rightarrow T_6$, $P_4\rightarrow T_8$ ($P_5\rightarrow T_2$, $P_6\rightarrow T_3$) represent the influence of discrete (continuous) part on continuous (discrete).

Assume m_1 (0) = 26 litres, m_2 (0) = 10 litres and m_3 (0) = 12 litres:

The markings m_1, m_2 and m_3 are positive thus the continuous transitions T_5, T_6 and T_7 are strongly validated. P_4 is not marked and T_8 is not enabled any more, and v_8 (t) = 0. Markings m_1, m_2 and m_3 evolve according to curves of Fig. 3. The dynamics of markings m_1, m_2 and m_3 is equal to:

$$V_1 = \dot{m}_1 = V_5 - V_6 = -1 litres/sec$$
$$V_2 = \dot{m}_2 = V_7 = 5 litres/sec$$
$$V_3 = \dot{m}_3 = V_6 - V_9 = -4 litres/sec$$

At $t = [3, 3]$: $m_3 = 0$ then its dynamic is null

$$V_3 = \dot{m}_3 = V_6 - V_9 = 0\ litres/sec$$

After $t = [1, 3]$: $20 \leq m_1 \leq 22$ thus T_2 is enabled. After $t = [3, 5]$, T_2 is fireable, $m_1=17$ and its dynamic $V_1 = \dot{m}_1 = V_5 = 2 litres/sec$ (P_2 being not marked thus T_6 m_1, m_2 and m_3 is not enabled any more). The evolution dynamics of continues places evolves according to the curves illustrated by Fig. 4.

The behavior of continuous Petri nets can be defined by the marking and their instantaneous firing speeds vector. For *HPN*, an IB-state (*Invariant Behavior-state*) concept is introduced [26]. An IB-state represents stages where *HPN* evolutions of marking remain constant and continuous. Formally, an IB-state is a time interval where:

1. Marking M^D of D-places is constant,
2. Validation vector e^D of discrete transitions is constant,
3. Instantaneous speeds vector of continuous transitions is constant,
4. Markings vector for discrete and continuous places is constant.
5. When an IB-state is reached, the marking of continuous places M^C is constant.

Consequently, a *DC-elementary HPN* evolution graph is a succession of IB-state modeled by nodes. The evolution between nodes is guided by the occurrence of events like

C_1-event (continuous place marking becomes null), D_1-event (firing of discrete-transition) and *D_2-event* (change validation degree of discrete transition by a marking of a continuous place).

Fig. 5 illustrates the evolution graph of the *DC-elementary HPN* of Fig. 2.

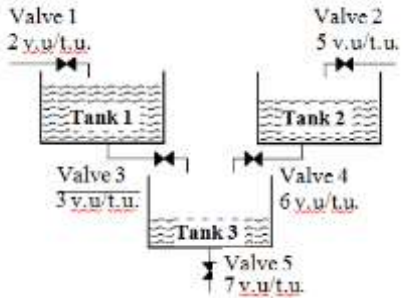

Fig. 1. Three water tanks system

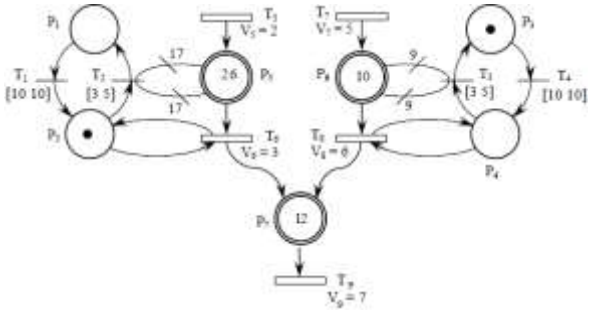

Fig. 2. DC-elementary HPN

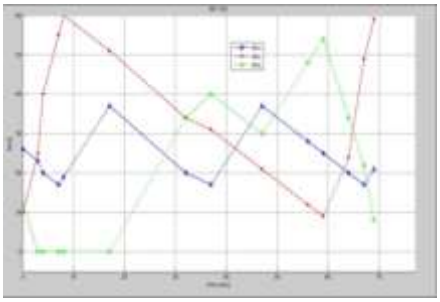

Fig. 3. Evolution marking

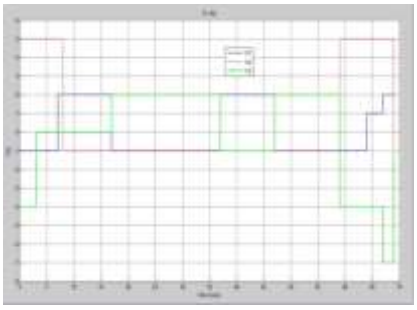

Fig. 4. Evolution dynamics of continuous places

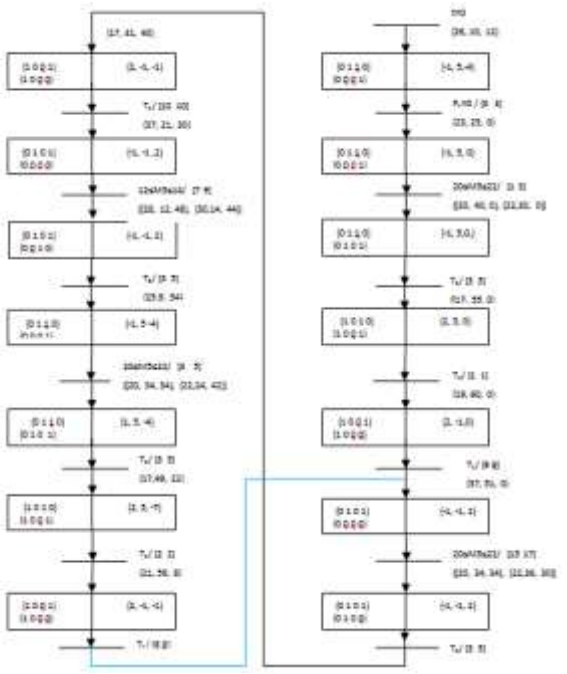

Fig. 5. Evolution graph of the DC-elementary HPN

B. Construction of LHA

Let us illustrate our algorithm from the example of Fig. 1. We consider that the initial marking of the DC-elementary HPN of figure 2 is M_0= [M_d {0, 1, 1, 0}, M_c {26, 10, 12}]. The proposed algorithm generates summits without taking into account the reachabilty of summits, which is a non-decidable problem for LHA. Nevertheless, the existence of software tools like Hytech and PHAVer (Polyhedral Hybrid Automaton Verifier) facilitates the computation of the reachable space. The mapping algorithm applied upon our illustrative example gives the LHA illustrated by the Fig. 6.

C. Analysis of three water tanks system

The set of generated summits keeps the syntax and the semantics of the evolution graph. The results of the simulation, which is based on software tool PHAVer (http://www.cs.ru.nl/~goranf/) for analyzing LHA $\mathcal{A}$, are summarized in Table 1. Reachability space by forward analysis was obtained after 5 iterations and time duration of 0,094s.

Let us now interpret these results, represented as linear constraints in table 1. For instance, the relation -6.x2 – 5.x4 – 5.p5 >= -160 (forward analysis column and L_3 state line) yields remarkable information (an upper limit of accessibility space): all trajectories in the upper part of this limit lead to dangerous states. In this state (summit), minimum water level in tank 1 exceeds and alarms may be triggered.

Comparison between tank 3 and the opening of valve 4

To illustrate the results of Table 1 and to be able to interpret them, the data from PHAVer was used to generate a bi-dimensional graph (Fig. 7) through a Matlab script.

In Fig. 7, we clearly see that the interval of the clock x4 is [0, 3], which is illustrated by the relation (x4 >= 0 and -x4 >= -3). We see also that the marking of place p_7 is positive in this interval (4.x4+p_7=12) (forward analysis column and L1 state line).

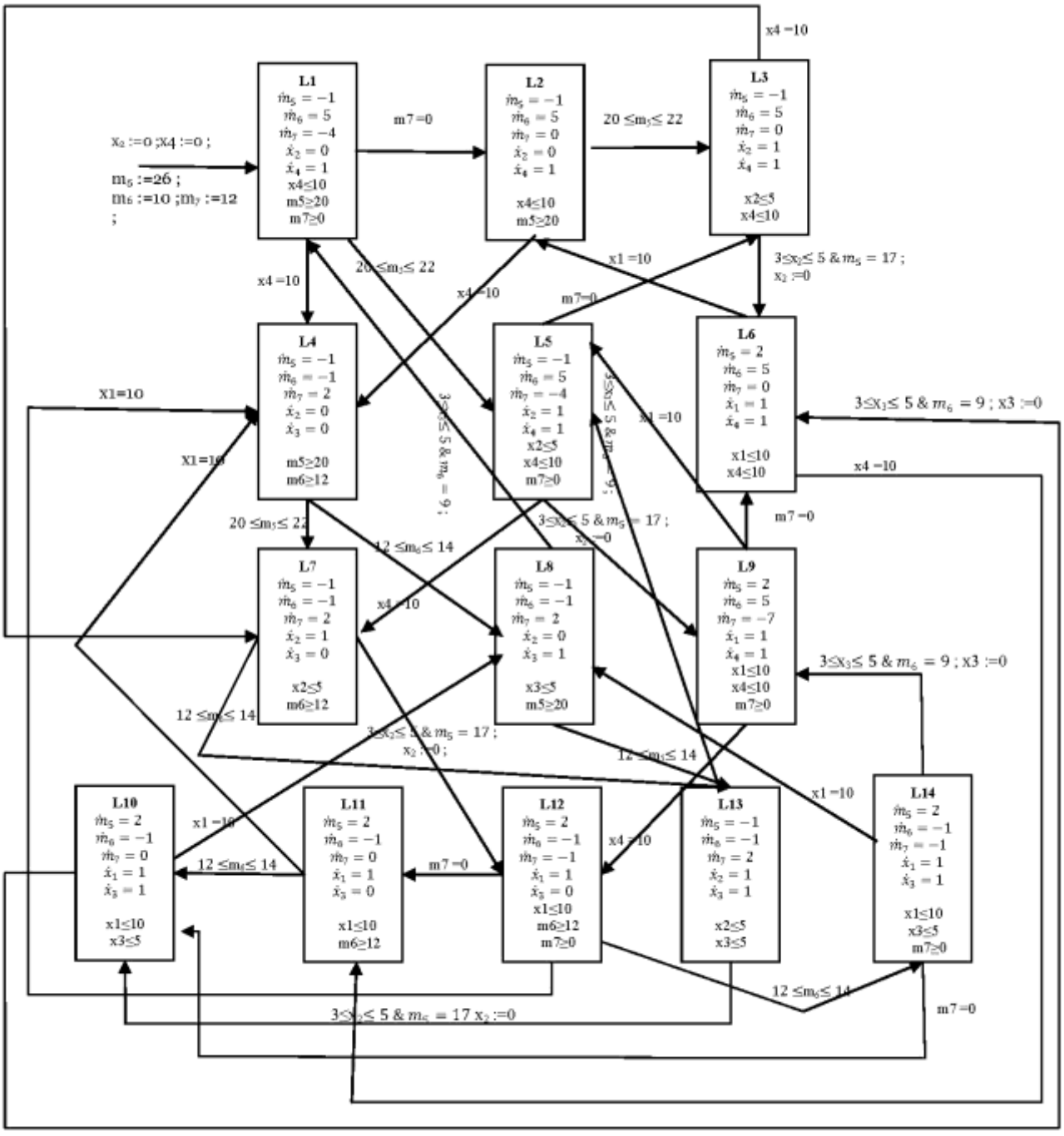

Fig. 6. LHA obtained by the mapping algorithm

Comparison between activity of tank 1 and activities of valve 3 and valve 4

The bi-dimensional graph of clock x_2 in term of place p_5 for all states {L_1... L_{14}} is illustrated in Fig. 8. In the accessible space (authorized trajectories) of summit L_3, we note that the temporal interval of the clock x_2 is *[0,5] (p5 >= 17, x2 + p5 >= 20 and -x2 - p5 >= -22)* where the marking place p_5 in this interval will be greater than or equal to 17. At the same time the valve 4 begins to open, since clock *x4* is activated *(-x4 >= -10).*

Comparison between activity of tank2 and the valve 4 closed

In the accessible state (authorized trajectories) of summit L_{13}, the temporal interval of clock *x3* is *[0, 5] (x3 >= 0, –x3 >= -5, x3+p6>=12, -x3-p6>=-14).* This means that tank 2 has reached a dangerous interval and valve 4 is in the process of opening, in order to prevent tank 2 from bypassing its minimum threshold equal to 9 (see Fig. 9).

Table 1. Reachable space of *LHA* $\mathcal{A}$

	Forward analysis
L_1	*4.x4 + p7 =12, 5.x4 - p6 = -10, x4 + p5 = 26, x2 = 0, x4 >= 0,-x4 >= -3*
L_2	*x2 =0, -x4 >= -10, p5 >= 20*
L_3	*x2 >= 0, -x4 >= -10, -x2 - p5 >= -22, -6.x2 – 5.x4 – 5.p5 >= -160, p5 >= 17, x2 + p5 >= 20, -x2 >= -5*
L_4	*p6 >= 12, p5 >= 20*
L_5	*x4 =0, p7 >= 0, -x2 >= -5*
L_6	*x1 >= 0, -x1 >= -10, -x4 >= -10*
L_7	*p6 >= 12*
L_8	*p6 >= 12, p5 >= 20, -p6 >= -14*
L_9	*x4 =0, x1 = 0, p7 >= 0*
L_{10}	*x3 = 0, -x1 >= -10, p6 >= 12, -p6 >= -14*
L_{11}	*p6 >= 12, -x1 >= -10*
L_{12}	*x1 >= 0, -x1 >= -10, p7 >= 0, p6 >= 12*
L_{13}	*-x3 >= -5, -x3 - p6 >= -14, x3 >= 0, x3 + p6 >= 12, -x2 >= -5*
L_{14}	*x1 >= 0, -x1 >= -10*

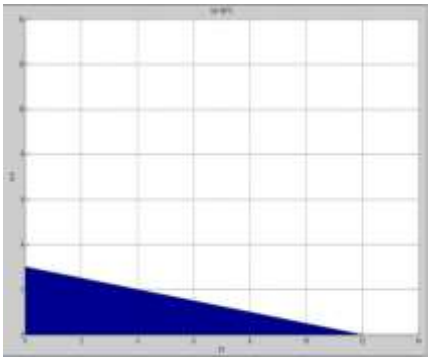

Fig. 7. Graph *x4* versus p_7 for states $\{L_1, \ldots, L_{14}\}$

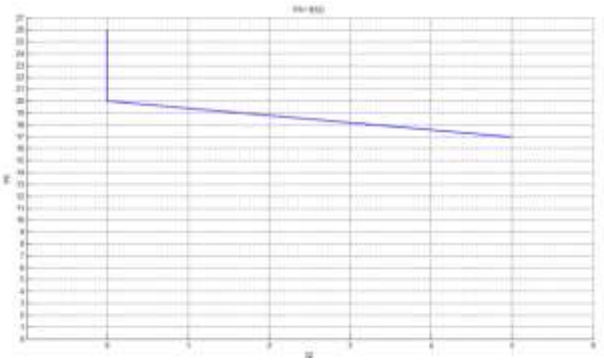

Fig. 8. Accessible space accepted by the hybrid automaton (p_5 in term of x_2)

Comparison between activities of tank 1 and tank 3

We observe that activities of tank 1 and tank 3 are identical since the two tanks emptied (*4.x4 + p7 = 12, 5.x4 - p6 = -10, x4 + p5 = 26)* as shown in figure 10. While no constraint is imposed on tank 3, this tank reaches the 0 level without water alimentation. Contrary to the tank 1, once the water level reaches a minimum threshold of 17, its activity changes (see Fig. 11).

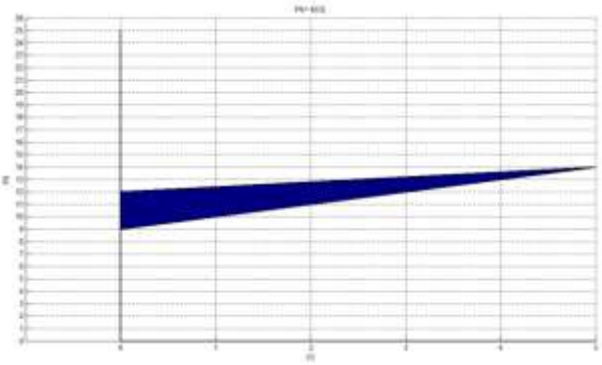

Fig. 9. Accessible space accepted by the hybrid automaton (*p6* in term of *x3*)

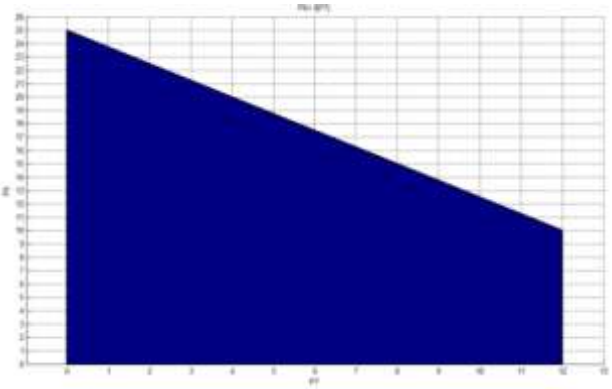

Fig. 10. Trajectories authorized by the hybrid automaton (p5 in term of p7)

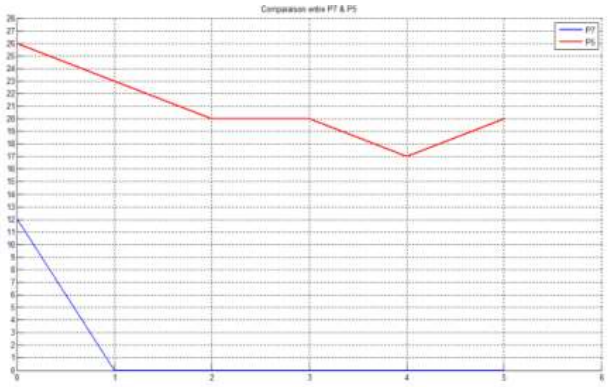

Fig. 11. Accessible states accepted by the hybrid automaton (comparison of p5 and p7)

Comparison between closing process of valve 3 and valve 4

Valve 3 closes at a faster rate than valve 4 (see Fig. 12). This is a consequence of the debit difference of tank 1 and tank 2 respectively.

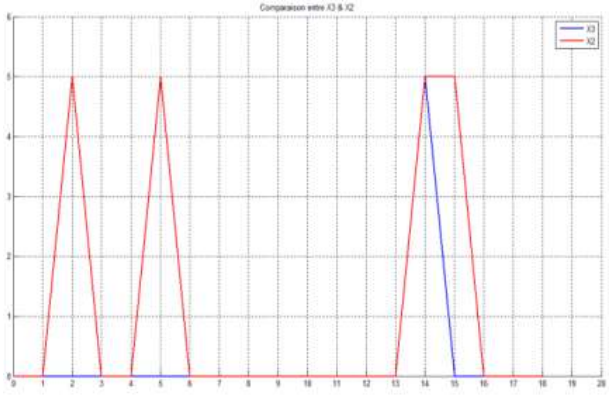

Fig. 12. Accessible space accepted: comparison between x3 and x2

In the same manner we can graphically interpret all information found in the table 1. We have obtained many interesting results [32] and [33].

V. Conclusion and Future Work

This paper analyzes the properties of the Petri net hybrid system. It expands on our previous works on discrete event systems modeled by a functional graph [4], [8] and [9]. The main intent of this work is to introduce a new class of *HPN (DC-elementary HPN)* that will be able to take into account the interaction between discrete part and continuous part in hybrid systems. We have also proposed an algorithm that allows for translation from an evolution graph to a hybrid dynamic system graph called a linear hybrid automaton. This graph is analyzed by a *PHAVer* software tool in order to demonstrate the system's properties. In future research, we will explore the possibility of studying diagnostic properties in hybrid systems, using this new kind of linear hybrid automaton.

Acknowledgment

This work was supported in part by the Industrial Computing and Networking Laboratory of Oran University.

References

[1] A. Zaidi, M. Tagina and B. Ould Bouamama, Reliability Data for improvement of Decision-Making in Analytical Redundancy Relations Bond Graph based Diagnosis, IEEE/ASME International Conference on Advanced Intelligent Mechatronics Montréal, Canada, July 6-9, 2010.

[2] B. Brandin and W.M. Wonham, Supervisory control of timed discrete event systems, IEEE Transactions on Automatic control, vol. 39, 2, pp. 329-341, 1994

[3] A. Zaidi, N. Zandouri and M. Tagina 'Graphical Approaches for Modelling and Diagnosis of Hybrid Dynamic Systems, *WSEAS Transaction on Systems*, 5(10), pp. 2322-2327, 2006, Springer.

[4] A.K.A. Toguyéni, E. Craye and L. Sekhri, Study of the Diagnosability of Automated Production Systems Based on Functional Graphs, *Mathematics and Computers in Simulation*, vol. 70, issues 5-6, 24, pp. 377-393, Elsevier, February 2006.

[5] H. Guéguen, M.A. Lefebvre, O. Nasri and J. Zaytoon, Safety Verification and Reachability Analysis for Hybrid Systems, Proceedings of the 17th World Congress, International Federation of Automatic Control Seoul, Korea, July 6-11, 2008.

[6] J. Zaytoon and J.L. Ferrier, Rappels sur les Systèmes à Evénements Discrets dans les Systèmes Dynamiques Hybrides, Edition Hermès, 2001, France.

[7] L. Sekhri, A.K.A. Toguyeni and E. Craye, A Relational Based Approach for Analysing Functional Graphs of Automated Production Systems, *IEEE International Conference on Systems, Man and Cybernetics, (SMC'02),* October 6-9, 2002, Hammamet, Tunisia.

[8] L. Sekhri, A.K.A Toguyeni and E. Craye, Diagnosability of Automated Production Systems Using Petri Net Based Models. *IEEE International Conference on Systems, Man and Cybernetics, (SMC'04),* October 10-13, 2004, The Hague, Netherlands.

[9] L. Sekhri, A.K.A. Toguyeni and E. Craye, Surveillabilité d'un Système Automatisé de Production Modélisé par un Graphe Fonctionnel, *Journal Européen des Systèmes Automatisés (JESA)*, vol.38, N° 3-4, Pages 243-268, Octobre, 2004, ISSN 1269-6935.

[10] P. Manon and C. Valentin-Roubinet, On the use of trajectory synthesis to return to nominal mode for a class of hybrid systems, *Journal Européen des Systèmes automatisés*, vol. 33, No. 8-9, November, 1999, MSR'99, pp. 995-1014.

[11] Z. Juarez, B. Denis and J.J. Lesage, Réseaux d'automates hybrides à synchronisations typées pour la modélisation des SDH, *Conférence Internationale d'Automatique Francophone, CIFA*, 2008, Bucarest, Roumanie.

[12] B. Ould-Bouamama, R. El Harabi, M.N. Abdelkrim and M.K. Ben Gayed, Bond graph for the Diagnosis of Chemical Processes, *Computers and Chemical Engineering*, 36, 301-324, 2012, Elsevier.

[13] M. Daigle, I. Roychoudhury, G. Biswas and X. Koutsoukos, Efficient simulation of component-based hybrid models represented as hybrid bond graphs, Technical Report ISIS-06-712, 2006, Institute for software integrated Systems, Vanderbilt university, Nashville, USA.

[14] P. Gawthrop and B. Geraint, Bond Graph Modeling, *IEEE Control Systems Magazine*, vol. 27, 2007.

[15] H. Alla and R. David, Continuous and Hybrid Nets, Journal of Circuits, Systems and Computers, vol. 8, No.1, pp. 159-188, 1998.

[16] R. David and H. Alla, Discrete, Continuous and Hybrid Petri Nets, 2005, Springer.

[17] K. Youngwoo, Traffic Network Control Based on Hybrid System Modeling, Petri Nets Applications, pp. 589-623, In:Tech, Ed., P. Pawlewski, 2010, ISBN 978-953-307-047-6.

[18] L. Chang Boon, W. Danwei, A. Shai and Z. Jing Bing, Causality Assignment and Model Approximation for Hybrid Bond Graph: Fault Diagnosis Perspectives' *IEEE Transactions on Automaton, Science and Engineering*, vol. 7, No. 3, July, pp. 570-580, July 2010.

[19] Y. EL-Touati, M. Yeddes, N. Ben Hadj Alouane and H. Halla, Du réseau de Petri temporel étendu vers les automates hybrides linéaires pour l'analyse des systèmes, 2009, *Conférence Internationale d'Automatique Francophone CIFA*, Bucarest, Romania.

[20] M. Tagian and I. Fliss, Diganosing Multiple faults in dynamic hybrid systems, *Intelligent Informatics, AISC* 129-189, 2013, Springer Verlag, Heilderberg.
[21] A. Gouin and J.L. Ferier, Modeling and Supervisory Control of Timed Automata, Journal Européen des Systèmes automatisés, Vol. 33, No. 8-9, MSR'99, pp. 1093-1110, November 1999.
[22] A.T. Sava and H. Alla, A Control Synthesis Approach for Time Discrete Event Systems, *Mathematics and Computers in Simulation*, vol. 70, issues 5-6, 24, pp. 250-265, Elsevier, February 2006.
[23] Freshe, Phaver: Algorithmic Verification of Hybrid Systems Past Hytech'. In: M. Morari and L. Thiele ed., Hybrid Systems: *Computation and Control*: 8th International Workshop, HSCC2005, Zurich, Switzerland, LNCS 3414, 2004, pp. 258–273, Springer.
[24] J. LeBail, Sur les Réseaux de Petri Continus et Hybrides, Thèse de Doctorat, Institut National Polytechnique de Grenoble (France), 1992.
[25] B. Berthomieu and M. Diaz, Modeling and verification of time dependent systems using time Petri nets, *IEEE Transaction on soft. Eng.*, vol. 17, No. 3, pp. 259-273, 1991.
[26] L. Ghomri, Modélisation structurelle utilisant les automates hybrides et les réseaux de Petri hybrides en vue de la synthèse de contrôleur des systèmes dynamiques hybrides, Mémoire de Magister, Université A. Belkaïd, Tlemcen (Algeria), 2005.
[27] L. Ghomri and H. Alla, Modeling and analysis using hybrid Petri nets, Nonlinear Analysis Hybrid Systems, 2007, pp. 141-153, Elsevier.
[28] P. Peter and H. Philips, Modelling Control and Fault Detection of Discretely-Observed Systems, Thesis, Technische Universities Eindhoven, 2001.
[29] H. Alla and R. David, A Modeling and Analysis Tool for Discrete Events Systems: Continuous Petri net, Performances evaluation, vol. 33, No. 3, 175, 1998.
[30] T. Henzinger, P. Peter, W. Kopke, A. Puri and P. Varaiya, Whats decidable about hybrid automata?, The algorithmic analysis of hybrid systems, *Proceedings of 27th annual ACM Symposium on theory of computing*, pp. 373-382, 1995.
[31] A.T. Sava, Sur la synthèse de la commande des systèmes à événements discrets temporisés. Thèse de Doctorat, Grenoble, France, 2001.
[32] R. Hakiki, Etude et Analyse des Systèmes Hybrides : Approche par les Réseaux de Petri Hybrides et Automates Hybrides Linéaires, Thèse de Magister en Informatique, Université d'Oran Sénia, 2010.
[33] R. Hakiki and L. Sekhri, Hybrid Petri Nets Based Approach For Analyzing Complex Dynamic Systems, *First International Conference on Machine and Web Intelligence (ICMWI'2010),* 3-5 October 2010, Algiers, Algeria.

Authors' Profiles

Sekhri Larbi is an Associate Professor at the Computer Science Department of Oran University. His current research area of interests include formal modeling in distributed and mobile systems, wireless ad-hoc and sensor networks, systems modeling using Petri nets, diagnosability and monitoring of automated production systems. He is member of the Industrial Computing and Networking Laboratory at Oran University. He has been a visiting professor at Cedric-CNAM research laboratory, in Paris, France, and Ecole Centrale de Lille (LAGIS) where he worked in Diagnosis of Industrial systems; and LIUPA Laboratory at the University of Pau, France.

Haffaf Hafid Obtained Doctor degree in computer Science in 2000; is a senior lecturer at the University of Oran Es-Senia (Algeria). He actually heads the L.I.I.R Laboratory at Computer science department –Oran University. His researchers concern different domain as Automatic control and diagnosis, optimisation, reconfiguration using matroid theory, system of system approaches and their applications in Bond graph and monitoring. He has many collaborations projects with European laboratory: Polytech lille where he worked in Intelligent transport systems infrastructures- and LIUPA, Pau (France) in the domain of Wireless sensor Networks

How to cite this paper: Sekhri Larbi, Haffaf Hafid,"Linear Hybrid Automaton Generation Using Mapping Algorithm for Hybrid Dynamic Systems", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.10, pp.1-10, 2014. DOI: 10.5815/ijitcs.2014.10.01