Control of Industrial Systems to Avoid Failures: Application to Electrical System

Yamen EL TOUATI
Department of Computer Science
Faculty of Computing and Information Technology
Northern Border University
Kingdom of Saudi Arabia

Saleh ALTOWAIJRI
Department of Information Systems
Faculty of Computing and Information Technology
Northern Border University
Kingdom of Saudi Arabia

Mohamed AYARI
Department of Information Technology
Faculty of Computing and Information Technology
Northern Border University
Kingdom of Saudi Arabia

***Abstract*—We resolve the control problem for a class of dynamic hybrid systems (DHS) considering electrical systems as case study. The objective is to guarantee that the plan never reaches unsafe states. We consider a subclass class of DHS called Cumulative Preemptive Event-driven DHS (CPE-DHS). This class is distinguished by the dominance of its discrete aspect characterized by features as cumulative continuous variables combined with actions behavior that may be interrupted and restarted. We utilize a subclass of Rectangular Hybrid Automata (RHA), named Constant Slope RHA (CSRHA), as a solution framework to resolve the control problem. The main contribution is a control Algorithm for the class of systems described above. This algorithm ensures that the system meet the requirement specifications by forcing some events. The forcing action is given in the form of restrictions on the transition guards of the CSRHA. The termination/decidability as well as correctness of the algorithm is given by theorems and formal proofs. This contribution ensures that the system will always be safe states and avoid failure due to the reachability of unsafe states. Our approach can be applied to a large category of industrial systems, especially electrical systems that we consider as case study.**

***Keywords*—Dynamic hybrid systems; supervisory control; hybrid automata; electrical systems; safety**

I. Introduction

Dynamic hybrid systems [1]–[4] (DHS) are systems characterized by the interaction of both discrete and continuous components. A large variety of real-time and embedded systems and many computer automated systems as well as industrial and electrical systems are described by both continuous and discrete aspects. In this paper, we concentrate in a particular class of dynamic hybrid systems where system behavior is captured essentially by preemptive activities which can be produced sequentially or in parallel. Besides, these systems are depicted by an interaction of dominant discrete component with a slight continuous one.

DHS are modeled by a large variety of modeling frameworks. We distinguish essentially several timed and hybrid extensions of finite state automata [5] as well as Petri nets [6], [7]. Petri nets extensions benefit a salient graphical modeling power. However, computations are mostly based on similar automata extension. On the other hand, there are many extensions of finite state machines, such as time transition systems [8], timed automata [5] and stop watch automata [9]. In these frameworks, time is included in configurations and transitions in the form of constraints and/or speed rate. In order to deal with dynamic hybrid systems, we consider essentially hybrid automata, linear hybrid automata, and rectangular automata [10]. All the previous frameworks capture various aspects of DHS depending on their modelling power which is generally inversely proportional with the decidability of the accessibility problem. In fact, models that cover more classes of systems become more difficult to manage by a computer due to the undecidability problems [11].

In our case, we use a subclass of RHA: the CSRHA to model our systems. This subclass is better managed from decidability side. The control problem, as one of the highly studied problems in literature [12], will be resolved using CSRHA formalism. One of the important problems in the DHS control theory is related to safety verification. This problem states that the controller has to ensure that all the trajectories of the system do not reach any "unsafe" state. In order to guarantee this safety property, the controller may restrict the scope of some controllable events. By taking such decision, the controller avoids that system trajectories interfere with any undesired state induced by uncontrollable events. However, in this paper, we consider that the computational power of the controller is limited to narrowing time intervals on transitions related to controllable events. Technically speaking, this action is similar to modifying guards on transitions associated to controllable events in the CSRHA model.

This paper is organized as follows. The next section provides background of hybrid automata and a description of the CSRHA. In Section 3, we present and solve the supervisory control problem. We note that throughout this paper we use the same case study of an electrical system to illustrate our supervisory control approach.

II. Background on Hybrid Automata

In the following, we define the retained subclass of RHA: the CSRHA.

A. Constant Piece-wise Rectangular Hybrid Automata

We consider these notations: $\mathcal{X} = \{x_1,\ x_2,\ \ldots,\ x_n\}$ is a finite set of real valued clocks (variables). $\dot{\mathcal{X}} = \{\dot{x},\ x \in \mathcal{X}\}$ denotes the set of first derivative variables of $\mathcal{X}$. A variable x is considered piece-wise linear variable if $\dot{x} \in \mathbb{R}$. $\sim$ denotes an element of operator's set $\{<, \leq, =, \geq, >, \neq\}$. A rectangular inequality over $\mathcal{X}$, is an inequality of the form, $x \sim c$, where $c \in \mathbb{R}$, and $x \in \mathcal{X}$. A rectangular predicate over $\mathcal{X}$ is a conjunction of rectangular inequalities over $\mathcal{X}$. $Rect(\mathcal{X})$ denotes the set of rectangular predicates over $\mathcal{X}$. A polyhedral inequality over $\mathcal{X}$ is an inequality of the form $c_1x_1 + \cdots + c_kx_k \sim c$, where $c, c_1, \ldots, c_k \in \mathbb{R}$, and $x_1, \ldots, x_k \in \mathcal{X}$. A polyhedral predicate over $\mathcal{X}$ is boolean combination of polyhedral inequalities over $\mathcal{X}$. $\Psi(\mathcal{X})$ is the set of polyhedral predicates over $\mathcal{X}$. $\mathbf{v} = (\ v_1, \ldots, v_n\ )$ denotes an element of $\mathbb{R}^n$ that captures clocks valuation, $v_i \in \mathbb{R}$, of every clock $x_i \in \mathcal{X}$. $v(x_i) = v_i$ corresponds to the value of x_i. We denote by *region*, a subset of $\mathbb{R}^n$. For a region z and $x_i \in \mathcal{X}$, $z(x_i) = \{v_i | \mathbf{v} \in z\}$. $\psi(\mathbf{v})$ denotes the boolean function which equals **true** if the predicate ψ is satisfied by the input vector $\mathbf{v}$ and **false** if not. We denote by $[\![\psi]\!]$, the region composed by the set of vectors $\mathbf{v} \in \mathbb{R}^n$, where the predicate ψ is **true** when we substitute each x_i by its corresponding v_i. $[\![\psi]\!](x_i)$ denotes the interval of values captured by v_i, $\forall \mathbf{v} \in [\![\psi]\!]$.

Definition 1: In [13]–[15] A constant piece-wise linear hybrid automata (CSRHA) is a tuple $\mathcal{A} = (\mathcal{X}, \mathcal{Q}, \mathcal{T} \cup \{e_0\}, inv, dyn, guard, assign, l_0)$ where:

- $\mathcal{X}$, is a finite set of variables.
- $\mathcal{Q}$, is a finite set of locations.
- $\mathcal{T}$, is a finite set of transitions. A transition $e = (l, l') \in \mathcal{T}$, leads the system from the source location $l \in \mathcal{Q}$, to the end location $l' \in \mathcal{Q}$. The entry transition of the initial state l_0 is denoted by e_0.
- inv: $\mathcal{Q} \longrightarrow \Psi(\mathcal{X})$ is the location *invariant*, it associates a predicate to each location.
- dyn: $\mathcal{Q} \times \mathcal{X} \longrightarrow \mathbb{R}$, is a function describing the evolution of variables. This evolution is usually of the form $l, \dot{x} = k,\ k \in \mathbb{R}$ or simply $\dot{x} = k$ in the location l. $\dot{\mathcal{X}}(l)$ denotes the evolution of all variables in the location l.
- $guard$: $\mathcal{T} \longrightarrow \Psi(\mathcal{X})$ is the guard function. It associates a predicate, C_e to each transition, e. The guard, C_e should equals true to allow the execution of the transition e.
- $assign$, is the initialization function. It associates a relation, $assign_e$ to each transition e defining the clocks to be reset.
- $l_0 \in \mathcal{Q}$, is the initial location. □

The semantic of a constant piece-wise linear hybrid automata (CSRHA) is given by the following definition:

Definition 2: The semantic of a CSRHA $\mathcal{A} = (\mathcal{X}, \mathcal{Q}, \mathcal{T} \cup \{e_0\}, inv, dyn, guard, assign, l_0)$ is defined by a timed transition system $S_{\mathcal{A}} = (Q,\ q_0, \longrightarrow)$ with

- $Q = \mathcal{Q} \times \mathbb{R}^n$ with $n = |\mathcal{X}|$.
- $q_0 = (l_0, init)$ is the initial state.
- $\longrightarrow \in \Big(Q \times (\mathcal{T} \cup \mathbb{R}_+) \times Q\Big)$ is defined by:

∘ $(l, v) \xrightarrow{a} (l', v')$(**jump transition**) $if\ \ \exists\ e = (l, l') \in \mathcal{A}\ s.t.$

$$\begin{cases} a = e \\ guard(e)(v) = true \\ v' = assign_e(v) \\ inv(l')(v') = true \end{cases}$$

∘ $(l, v) \xrightarrow{\epsilon(t)} (l', v')$(**flow transition**) if

$$\begin{cases} l = l' \\ v' = v + t * \dot{\mathcal{X}}(l) \\ inv(l')(v') = true \end{cases}$$ □

A *run* of CSRHA $\mathcal{A}$ is a path in $S_{\mathcal{A}}$ started from q_0. $[\![\mathcal{A}]\!]$ denotes the set of all runs of $\mathcal{A}$. We note $(l, v) \xrightarrow{\epsilon(t)} (l', v') \xrightarrow{a} (l'', v'')$ is equivalent to $(l, v) \longrightarrow^{\epsilon(t)}_{a} (l'', v'')$. A state (l_i, v_i) is considered as *reachable*, if $\exists (l_0, v_0) \longrightarrow^{\epsilon(t_0)}_{a_0} (l_1, v_1) \longrightarrow^{\epsilon(t_1)}_{a_1} (l_2, v_2) \longrightarrow^{\epsilon(t_2)}_{a_2} \ldots \longrightarrow^{\epsilon(t_i)}_{a_i} (l_i, v_i)$ where $(l_0, v_0) = q_0$. A run $(l_0, v_0) \longrightarrow^{\epsilon(t_0)}_{a_0} (l_1, v_1) \longrightarrow^{\epsilon(t_1)}_{a_1} (l_2, v_2) \longrightarrow^{\epsilon(t_2)}_{a_2} \ldots \longrightarrow^{\epsilon(t_i)}_{a_i} (l_i, v_i) \ldots$ starting from $q_0 = (l_0, v_0)$ is a *timed trace*, denoted as $w = (a_0 = e_0, \delta_0) \rightarrow (a_1, \delta_1) \rightarrow (a_2, \delta_2) \rightarrow \ldots (a_2, \delta_2) \ldots$, where w is a sequence of pairs (a_i, δ_i), with $a_i \in \mathcal{T} \cup \{e_0\}$ a transition, and $\delta_{i+1} \in \mathcal{R}_+$ is the delay between the two successive events a_i and a_{i+1}, where : $\delta_0 = 0$, and $\forall i \geq 1,\ \ \delta_i = \epsilon(t_i) - \epsilon(t_{i-1})$.

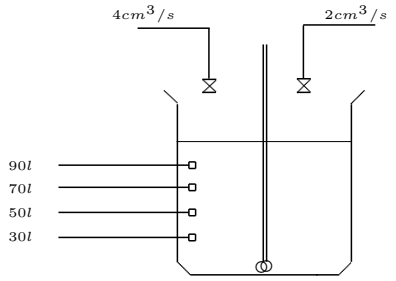

Fig. 1. Electrical system for blending chemical solution.

Example 2.1: Consider the electrical system for mixing chemical solution given in Fig. 1. Filling action is composed of two stages. Firstly, a tray is replenished by a chemical solution with a rate of $2cm^3/s$. We assume that initially the tray is filled by $10dm^3$ of a neutral liquid. This phase is accomplished when the current content of the tray is bounded by 30 and 50 liters. The next phase should be fulfilled before a deadline of $18s$, elapses in order to avoid the risk of obtaining improper solution. An authorization at a random time prompts the second stage which has a deadline of $16s$ once started.

When the next stage is activated, a second chemical solution is replenished with the rate of $4cm^3/s$. The filling process is accomplished when the total content of the tray is bounded by $70dm^3$ and $90dm^3$. The CSRHA modeling of this electrical system is illustrated in Fig. 2.

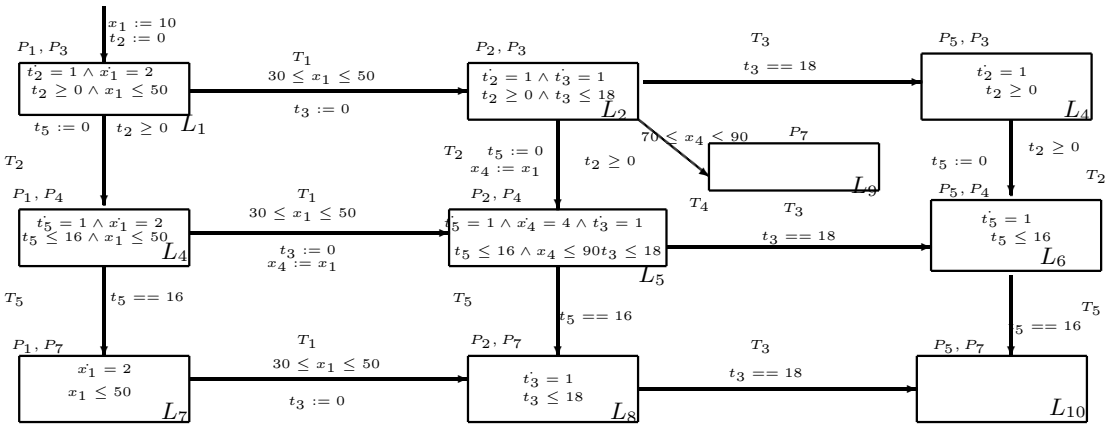

Fig. 2. The CSRHA of the electrical system.

III. Control of CPE-DHS

In the following, we describe our contribution to resolve the control problem. Our solution define a derived space where all trajectories satisfy the requested specifications to avoid system failure. Thus, all unsafe locations will be inaccessible. The safety specification is considered as the set of forbidden locations. The control action acts by reducing transition guard intervals. By nature, some events are not eligible for narrowing their time occurrence scope. Such events are considered uncontrollable from the controller perspective. An event is controllable if the controller has the power to reduce its occurrence time slot. In general, event connected to forbidden locations are uncontrollable, otherwise it becomes trivial to define the control solution. Moreover, the restriction action on the time intervals should be minimalist.

A. Specification of the Control Problem

The inputs are the set of unsafe locations and the partition of events as controllable/uncontrollable. The main steps that we propose to resolve the control problem are as follows:

Steps:

1) Mark all unsafe locations considering the safety specification.
2) Mark all transitions as controllable and uncontrollable considering the input events partition.
3) Perform a computation of the desired space adopted by the controller in all the locations to ensure that the system is not accessing forbidden locations.
4) Reassign the restricted guards of transition related to controllable events and update any necessary location invariant to force that the system remains in safe states.

B. Control Algorithm

Let $\mathcal{A} = (L, l_0, X, \Sigma, E, inv, Dif)$ the CSRHA model of the system to be controlled. $\mathcal{A}^d$ represents the output (controlled) CSRHA. We consider these notations:

- L^F represents the set of forbidden locations (given by the safety specification).
- E^F represents the set of CSRHA transitions where the output location is a forbidden.

$$E^F = \{e \in E | e = (l, \delta, \alpha, Aff, \rho, l'),\ l' \in L^F\}$$

- $e_{l,l'}$ represents a transition $e = (l, \delta, \alpha, Aff, \rho, l')$ where the source location is l and the destination location is l'.
- E_l represents the set of transitions having l as source location.

$$E_l = \{e \in E | e = (l, \delta, \alpha, Aff, \rho, l'), l' \in L\}$$

- $E_l^F = E_l \cap E^F$ represents the forbidden transitions having l as source location.
- $\overline{E_l^F} = E_l - E^F$ represents the non forbidden transitions having l as source location.
- In Algorithm III.1, we consider that $|E^l| = |\underline{E_l^F} \cup \overline{E_l^F}| = |E_l^F| \cup |\overline{E_l^F}| = m$ such as $|\underline{E_l^F}| = k$, $|\overline{E_l^F}| = m - k$. E^l is parted into E_l^F and $\overline{E_l^F}$ as follows:
  - $E_l^F = \{e_1 = (l, \delta_1, \alpha_1, Aff_1, \rho_1, l_1), e_2 = (l, \delta_2, \alpha_2, Aff_2, \rho_2, l_2), \ldots, e_k = (l, \delta_k, \alpha_k, Aff_k, \rho_k, l_k)\}$
  - $\overline{E_l^F} = \{e_{k+1} = (l, \delta_{k+1}, \alpha_{k+1}, Aff_{k+1}, \rho_{k+1}, l_{k+1}), e_{k+2} = (l, \delta_{k+2}, \alpha_{k+2}, Aff_{k+2}, \rho_{k+2}, l_{k+2}), \ldots, e_m = (l, \delta_m, \alpha_m, Aff_m, \rho_m, l_m)\}$
- $L^R \in (2^L)^L$ represents the set of reachable locations from l in $-\mathcal{A}$[1]. In other words, a location $l' \in L^R(l)$ if it exists a run from l' to l. Formally, $L^R(l) = \{l \in L, \exists k \in \mathbb{N},\ (l', v') \xrightarrow{t_1,e_1} (l_1, v_1) \xrightarrow{t_2,e_2} (l_2, v_2) \ldots \xrightarrow{t_k,e_k} (l_k, v_k),\ l_k = l\}$. This corresponds to the closure of the set $\{l\}$ under the relation $\{(p, q)$: there is a transition $e = (p, \delta, \alpha, Aff, \rho, q) \in E,\ q \in L^R(l)\}$.
- $\overline{L^R(l)}$ represents $L - L^R(l)$.

Algorithm III.1 Control Algorithm

[1] $-\mathcal{A}$ is the reversed automate of $\mathcal{A}$ ([16]).

function Control($\mathcal{A}, M^F$):$\mathcal{A}^d$
initialize the output CSRHA by the entry CSRHA. $\mathcal{A}^d := \mathcal{A}$

function initialize()
calculate the set E^F :
for all $e_{l,l'} \in E$ with $l' \in L^F$ **do**
  $E^F := E^F \cup \{e_{l,l'}\}$
end for
calculate $L^R(l)$
initialize $L^R(l) := \{l\}$
while $\exists e = (l', \delta, \alpha, Aff, \rho, l'') \in E$, with $l'' \in L^R(l)$ and $l' \notin L^R(l)$ **do**
  $L^R(l) := L^R(l) \cup \{l'\}$.
end while
for all location $l \in L \backslash L^F$ **do**
  calculate E_l^F and $\overline{E_l^F}$:
  for all $e_{l,l'} \in E^F, l' \in L$ **do**
    $E_l^F := E_l^F \cup \{e_{l,l'}\}$
  end for
  calculate $\overline{E_l^F} := E_l - E^F$
  if $\overline{E_l^F} = \emptyset$ **then**
    $L^F := L^F \cup \{l\}$
  end if
end for
if L^F is modified **then**
  re-invoke **initialize()** to reconsider the new forbidden locations
end if
if $L^F = L$ **then**
  exit (there is no solution)
end if
end function

for all location l where $E_l^F \neq \emptyset$ **do**
  recalculate the guard predicates of all the transitions included into the set $\overline{E_l^F}$:
  for all $e_i \in \overline{E_l^F},\ i \in [\![k+1, m]\!]$ **do**
    calculate the new guard δ_i^n regarding δ_i and guards of transitions in E_l^F : [2]

$$\boldsymbol{\delta_i^n := \delta_i \wedge \neg \delta_1 \ \wedge \neg \delta_2 \ \ldots \ \wedge \neg \delta_k}$$

  end for
end for
do a forward analysis, started at the initial location. We note by $S_l^{forward}$ the reachable space[3] calculated by forward analysis at location l.
for all location l where $E_l^F \neq \emptyset$ **do**
  do a backward analysis started at location l considering $\delta_{k+1}^n \vee \delta_{k+2}^n \vee \ldots \vee \delta_m^n$ as initial entry space. We note by par $S_{l,l'}^{backward}$ the space calculated by backward analysis (from location l') in the location $l \in L_l^R$
end for
for all $l' \in L^R(l)$ where $E_l^F \neq \emptyset$ **do**
  calculate the final space of backward analysis at location l' :

$$\boldsymbol{S_{l'}^{backward} := \bigwedge_{l \in E_l^F} S_{l,l'}^{backward}}$$

end for
for all location l_i **do**
  calculate the desired space S_l^d at location l

$$\boldsymbol{S_l^d = S_l^{backward} \wedge S_l^{forward}}$$

  calculate the new location invariant l given by $\boldsymbol{inv^d(l) := inv(l) \wedge S_l^d}$
  for all transition $e_{l,l'} \in E_l^F$ **do**
    redefine the guards : $\boldsymbol{\delta_{l,l}^d := \delta_{l,l'} \wedge S_l^d}$
  end for
end for
end function

The CSRHA modelling a CPE-DHS system is the input of the Algorithm III.1. Algorithm III.1 produces the output as an updated CSRHA where forbidden states can never be reached. The control algorithm computes the new transition guards and the new location invariants.

Theorem 1: The Algorithm III.1 terminates if the entry CSRHA has no loop.

Proof: 1 The Algorithm III.1 terminates if the computation of reachable space (both backward and forward) terminates. This analysis use discrete and continuous predecessor and successor operators which perform certain geometric calculus on regions [14]. Software like PHAVer [17] and SpaceEx [18], [19] implement such region operations, using polyhedral libraries, to accomplish the reachable space computation. We note that these analysis terminate if the CSRHA is acyclic. Nevertheless, for more general forms, the accessibility problem is known as undecidable [14], [20]. ■

In the following, we present some particular and interesting cases where this problem is decidable.

Theorem 2: The Algorithm III.1 terminates if the input CSRHA satisfies the following proprieties:

1) All derivative variables in the locations are **non negative** or **null**.
2) Guards and invariants are defined by **single non negative** constraints.
3) Assignments are of the form $x' := x$ or $x' = c$.

Proof: This is ensured due to the decidability of accessibility problems in that case [21]. ■

Furthermore, we can ensure the algorithm decidability for these interesting classes of CSRHA:

1) CSRHA where each loop contains at least one initialization of all clocks [22].
2) CSRHA where each loop contains at most one transition guard in the form of "dangerous" test [22].
3) CSRHA where the dynamic changing (the derivative value) of a variable between two locations is accompanied by resetting the variable assignment at the transition between the two locations [16].

[2]Our goal is to reduce the state space in order to avoid the possibility of occurrence of prohibited events.

[3]The reachable space at a given location is a polyhedron with dimension $|X|$ defined the inequalities system $A.XR^{\prec}b$, with $A \in \mathcal{M}_{a,|X|}(\mathbb{R})$ a matrix with a lines and $|X|$ columns, and $X \in \mathbb{R}^n$ the vector of CSRHA variables.

Theorem 3: The automaton $\mathcal{A}^d$ obtained by applying Algorithm III.1 ensures that all reachable spaces respect the safety specification while being maximal permissive.

Proof: Consider the CSRHA $\mathcal{A} = (L, l_0, X, \Sigma, E, inv, Dif)$.

Part 1: We demonstrate (by contradiction) that the reachable space meets the safety specification.

Suppose that $\exists l \in L^F$ such as it exists a run in $\mathcal{A}^d$ from initial state:

$$(l_0, v_0) \xrightarrow{t_0, e_0} (l_1, v_1) \dots (l_a, v_a) \xrightarrow{t_a, e_a} (l, v)$$

We have $l \in L^F \implies e_a \in E^F$. Suppose that $e_a = (l_a, \delta_a, \alpha_a, Aff_a, \rho_a, l)$. According to the TTS of $\mathcal{A}^d$, we have $inv(l)(v_a) = true$ and $\delta(v_a) = true$. However, according to Algorithm III.1, the calculation of $S_l^{backward}$ conclude that $inv^d(l) = inv(l) \wedge \neg\delta_1 \wedge \neg\delta_2 \dots \wedge \neg\delta_k$, $\forall e_i \in E^F_{l_a}, i \in [\![1, k]\!]$. According to the construction of the set $E^F_{l_a}$ in the Algorithm, we have $e_a \in E^F_{l_a}$. Thus, $\exists j \in [\![1, k]\!]$ such as $e_a = e_j$. This implies that $inv(l')(v) = false$, which contradicts the starting assumption.

Part 2: We demonstrate (by contradiction) that the reachable space at $\mathcal{A}^d$ is maximal permissive.

To do this, let us suppose that there is a location $(l, v) \in \mathcal{Q}_A$ such that $(l, v) \notin \mathcal{Q}_{A^d}$ and $l \notin L^F$. Also suppose that (l, v) do not lead to forbidden locations by the specification. As $(l, v) \notin \mathcal{Q}_{A^d}$, we obtain $inv^d(l)(v) = false$. Similarly $(l, v) \in \mathcal{Q}_A \implies inv^d(l)(v) = true$.

The fact that (l, v) does not lead to unauthorized locations, means that there is no run from (l, v) leading to a state (l', v') with $l' \in L^F$.

Let $l^f \in L^F$ a location such that $l \in L^R(l_f)$. Since there is no run from (l, v) leading to forbidden location, thus, (l_f, v_f) is not reachable since (l, v), and that, for any $v_f \in \mathbb{R}^{|X|}$. Similarly, (l, v) is not reachable from (l_f, v_f) at the reverse automaton $-\mathcal{A}$ (or by backward analysis). Let $S'^{backward}_{l,l_f}$ the obtained space at l by backward analysis from (l_f, v_f). Thus, we have $S'^{backward}_{l,l_f}(v) = false$.

According to Algorithm III.1, S_l^d start by the initial space $\neg\delta_1 \wedge \neg\delta_2 \dots \wedge \neg\delta_k$ $\forall e_i \in E^F_{l_f}, i \in [\![1, k]\!], k = |E^F_{l_f}|$. Thus, $S_l^d(v) = true$. Moreover, according to the calculation formula of location invariant, we have $inv^d(l)(v) = true$.

$\implies (l, v) \in \mathcal{Q}^d$. Thus, any location leading exclusively to locations respecting the specification is in the reachable space of $\mathcal{A}^d$. Consequently, $\mathcal{A}^d$ is maximal permissive. ■

Example 3.1: We reconsider the CSRHA of the electrical system illustrated in Fig. 2. According to the safety specification, we consider the following unsafe locations: $\mathcal{SF} = \{l_7, l_8, l_{10}, l_4, l_6\}$. The results related to the reachable space computation by forward and backward analysis are performed by PHAVeR [17] and SpaceEx [18], [19] software. The intersection between backward and forward spaces is illustrated in Table I. The results meets with the safety specification. Thus, the controller defines a derived CSRHA where invariant locations and transition guards are truncated by the new obtained polyhedral equations in each location. This derived automaton is maximal permissive and describes all possible trajectories that obey to the requirements.

TABLE I. INTERSECTION SPACE

l_5	$E_5^c = (-x_4 - 4t_2 + 4t_3 \geq -130) \wedge (-x_4 \geq -90) \wedge (-x_4 + 4t_3 \geq -50) \wedge (-x_4 + 2t_2 + 4t_3 > -42) \wedge (x_4 - 4t_3 > -2) \wedge (-t_5 > -16) \wedge (t_3 \geq 0) \wedge (t_2 \geq 0) \wedge (x_4 - 4t_5 > 6) \wedge (x_4 - 2t_2 + 2t_3 \geq 10) \wedge (x_4 \geq 30) \wedge (7x_4 + 18t_2 - 28t_3 \geq 210) \wedge (-t_3 > -18)$
l_4	$E_4^c = (x_1 - 2t_2 - 2t_5 == 10) \wedge (-x_1 \geq -50) \wedge (-x_1 + 2t_2 > -42) \wedge (t_2 \geq 0) \wedge (x_1 - 2t_2 \geq 10)$
l_1	$E_1^c = (x_1 - 2t_2 == 10) \wedge (-x_1 \geq -50) \wedge (x_1 \geq 10)$
l_9	$E_9^c = (-x_4 - 4t_2 + 4t_3 \geq -130) \wedge (-x_4 \geq -90) \wedge (-x_4 + 4t_3 \geq -50) \wedge (-x_4 + 2t_2 + 4t_3 > -42) \wedge (-t_3 > -18) \wedge (-t_5 > -16) \wedge (t_2 \geq 0) \wedge (x_4 \geq 70) \wedge (7x_4 + 18t_2 - 28t_3 \geq 210)$
l_2	$E_2^c = (x_1 - 2t_2 + 2t_3 == 10) \wedge (-x_1 \geq -50) \wedge (-x_1 + 2t_2 \geq -10) \wedge (3x_1 - 4t_2 > 18) \wedge (x_1 \geq 30)$

Table I illustrates the intersection space, obtained by PHAVer and SpaceEx. This allows capturing the maximal polyhedron that meet with requirements. For example, the updated location invariant of l_4 is given by $I_4^c = I_4 \wedge (x_1 - 2t_2 - 2t_5 == 10) \wedge (-x_1 \geq -50) \wedge (-x_1 + 2t_2 > -42) \wedge (t_2 \geq 0) \wedge (x_1 - 2t_2 \geq 10)$. Besides, the updated guard of transition of $T_{4,5}$ is $g_{4,5}^c = g_{4,5} \wedge t_5 < 16 \wedge (x_1 - 2t_2 - 2t_5 == 10) \wedge (-x_1 \geq -50) \wedge (-x_1 + 2t_2 > -42) \wedge (t_2 \geq 0) \wedge (x_1 - 2t_2 \geq 10)$. Similarly, all guards and invariants will be updated according to the results given by the intersection space. Furthermore, we omit any outgoing transition from a forbidden location (since it becomes unreachable).

IV. CONCLUSION

In this paper, our main contribution is to solve the problem of supervisory control of the particular class of dynamic hybrid systems (DHS) called Cumulative Preemptive Event-driven DHS (CPE-DHS) by narrowing guards and invariants of transitions relative to controllable events in a way that forbidden states remain inaccessible. Our proposed solution can be applied in a systematic way to any system that fits with our requirements. Then we applied this approach to an electrical system as case study. Generally speaking, the control problem is known to be undecidable for this class of complex systems. Nevertheless, in quest of decidability, we propose some restrictions that makes the problem decidable. In our future directions, we will focus on the supervisor generation while considering uncontrollable variables.

ACKNOWLEDGMENT

The authors gratefully acknowledge the approval and the support of this research from the Deanship of Scientific Research study by the grant no 4665-CIT-2016-1-6-F K.S.A, Northern Border University, Arar, KSA.

REFERENCES

[1] A. van der Schaft and H. Schumacher, *An introduction to hybrid dynamical systems Lecture Notes in Control and Information Sciences*. London: Springer-Verlag London Ltd., 2000, vol. 251.

[2] J. Lygeros, K. Johansson, S. Simic, J. Zhang, and S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[3] R. Sanfelice., "Robust hybrid control systems," Ph.D. dissertation, University of California, Santa Barbara, 2007.

[4] M. Kosmykov, "Hybrid dynamics in large-scale logistics networks," Ph.D. dissertation, University of Bremen, 2011.

[5] R. Alur, "Timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1999.

[6] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE transactions on software Engeneering*, vol. 17, no. 3, pp. 259–273, 1991.

[7] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, 1st ed. Springer, Heidelberg, 2004.

[8] T. A. Henzinger, Z. Manna, and A. Pnueli, "Timed transition systems," in *Real-Time: Theory in Practice*, ser. Lecture Notes in Computer Science, J. Bakker, C. Huizing, W. Roever, and G. Rozenberg, Eds. Springer Berlin Heidelberg, 1992, vol. 600, pp. 226–251. [Online]. Available: http://dx.doi.org/10.1007/BFb0031995

[9] F. Cassez and K. Larsen, "The impressive power of stopwatches," in *In Proc. of CONCUR 2000: Concurrency Theory*. Springer, 1999, pp. 138–152.

[10] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1995, pp. 373–382, a revised version appeared in Journal of Computer and System Sciences, vol. 57, p. 94124, 1998.

[11] E. Asarin, V. P. Mysore, A. Pnueli, and G. Schneider, "Low dimensional hybrid systems - decidable, undecidable, don't know," *Inf. Comput.*, vol. 211, pp. 138–159, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.ic.2011.11.006

[12] R. Alur, T. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," *IEEE Transactions on Software Engineering*, vol. 22, no. 3, pp. 181–201, 1996, (A preliminary version appeared in the Proceedings of the 14th Annual Real-Time Systems Symposium (RTSS), IEEE Computer Society Press, 1993, pp. 2-11.).

[13] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and Computation*, vol. 111, pp. 394–406, 1994. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.7422

[14] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995, (A preliminary version appeared in the Proceedings of the 11th International Conference on Analysis and Optimization of Systems: Discrete-Event Systems (ICAOS), Lecture Notes in Control and Information Sciences 199, Springer-Verlag, 1994, pp. 331-351.). [Online]. Available: citeseer.ist.psu.edu/alur95algorithmic.html

[15] Y. E. Touati, N. B. Hadj-Alouane, and M. Yeddes, "Modeling and control of constant speed dynamic hybrid systems using extended time petri networks," in *Control and Decision Conference (CCDC), 2012 24th Chinese*, May 2012, pp. 634–641.

[16] T. A. Henzinger and V. Rusu, "Reachability verification for hybrid automata," in *HSCC 98: Hybrid Systems Computation and Control, Lecture Notes in Computer Science 1386*. Springer-Verlag, 1998, pp. 190–204.

[17] G. Frehse, "Compositional verification of hybrid systems using simulation relations," Ph.D. dissertation, Radboud Universiteit Nijmegen, 10 2005.

[18] G. F. Alexandre Donzé, "Modular, hierarchical models of control systems in spaceex," in *Proc. European Control Conf. (ECC'13)*, Zurich, Switzerland, 2013.

[19] G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '13. New York, NY, USA: ACM, 2013, pp. 203–212. [Online]. Available: http://doi.acm.org/10.1145/2461328.2461361

[20] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Systems*, ser. Lecture Notes in Computer Science, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. Springer Berlin, Heidelberg, 1993, vol. 736, pp. 209–229, 10.1007/3-540-57318-6-30. [Online]. Available: http://dx.doi.org/10.1007/3-540-57318-6-30

[21] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine, "Integration graphs: A class of decidable hybrid systems," in *Hybrid Systems, volume 736 of Lecture Notes in Computer Science*. Springer-Verlag, 1993, pp. 179–208, (appeared in In Proceedings of Workshop on Theory of Hybrid Systems , Lyngby, Denmark, June 1992).

[22] Y. Kesten, A. Pnueli, J. Sifakis, and S.Yovine, "Decidable integration graphs," in *Information and Computation, volume 150(2)*, 1999, pp. 209–243.