Verification of a Hybrid Model of a Manufacturing System using Rectangular Petri Nets

Behzad Bordbar, Luisa Giacomini and David J. Holding
Aston University, Birmingham, UK

Key words: Hybrid Model, Rectangular Petri Nets

Abstract: This paper addresses the problem of deriving a design and verification method for hybrid systems. The paper applies the theory of Rectangular Petri nets in which the discrete event system is modelled using an extended form of Petri net and the continuous system is represented as a differential inclusion and its dynamics are constrained to rectangular regions of the state space. The Rectangular Petri net can be analysed using a Rectangular Reachability Graph, a generalisation of the reachability graph in conventional Petri Net theory. The method is illustrated using a simplified example of a manufacturing process.

1. INTRODUCTION

In many manufacturing and process control systems the discrete event system has a supervisory role. It governs both the sequencing of operations and the control strategy for the each operation. Typically, during each state of the discrete event system, the continuous system is controlled using an appropriate algorithm and is made to follow a particular set-point trajectory or motion profile. When the conditions that determine the end of a specific profile or trajectory are satisfied, the discrete event system selects the mode, profile or algorithm for the next phase of the system behavior. In such hybrid systems, the states will form pairs, consisting of both discrete event states and continuous states, and each evolution of the system can be either via a change in the discrete event state or changes in the continuous states of

the system. This paradigm provides the basis for the selection of the modeling techniques that are used in this paper. The problem is to ensure consistency between the two domains while reasoning about the behaviour of the system. In particular, the method should facilitate reasoning in discrete domain while simultaneously ensuring the behaviour in the continuous domain.

The 'higher level' discrete event part of a hybrid system can be modelled using a variety of techniques, including temporal logic, process algebras, state-charts, automata, or Petri nets (Antsaklis et al., 1997; Le Bail et al., 1991). Our work draws on Rectangular Automata (Herzinger et al., 1998), in which the discrete event part is an Automata.

Rectangular Automata provide a satisfactory starting point for the top-down design of systems, but it is somewhat less useful for bottom-up or compositional designs in which subsystems with well defined behaviour are embedded within the system (Jiang et al., 1996; Lamport, 1997). To generate a more general approach that accomodates top-down design and provides consistency in compositional design, this paper develops and applies the theory of a Petri-Net-based hybrid systems model called the *Rectangular Petri net* or RPN. Also, we introduce the Rectangular Reachability Graph (RRG) which contains information on the feasible states and state transitions of the discrete event part of the system. The RRG can help in verifying hybrid systems modelled as Rectangular Petri nets, in the same way that almost all important queries about DED's modelled as conventional Petri nets can be formulated in term of reachability problems and checked via studying the conventional reachability graph.

To provide insight into the approach, the method is demonstrated by application to a manufacturing process that is common to many production lines.

2. MODELLING HYBRID SYSTEMS USING RPN

The Rectangular Petri net is designed to provide an integrated description of both the discrete event and continuous parts of a hybrid system. It adopts a supervisory control view that is centred on the state of the discrete event system (PN marking) and captures the notion that a hybrid system evolves through either changes in discrete state or changes in continuous state. The states of the continuous parts of a system can be represented as n-dimensional vector functions $\mathbf{x}$: $[0, \infty) \rightarrow \mathbb{R}^n$. Points of $\mathbb{R}^n$ are denoted as $\mathbf{x} = (x_1, \ldots, x_n)$. Let $[\mathbf{x}]_i = x_i$ where $1 \leq i \leq n$ denotes the i-th co-ordinate of $\mathbf{x}$. The states of a continuous system evolve over time and the time variable is indicated by θ. The evolution of the continuous system forms a trajectory in

$\mathbb{R}^n$. Given any discrete event state the associated trajectory is considered to be constrained by rectangular regions[1], specifically the *initial region* which constrains the initial continuous state, the *invariant region* which constrains the continuous state variable, and the *flow region* which constrains the derivatives of the state variables. RPN uses the notion of *scope* to determine the effect of discrete event state changes on the continuous system. For example, a discrete event transition that changes a set point trajectory or control mode will not immediately alter variables associated with the energy or momentum of the physical system. Thus a change in discrete event state normally immediately affects only specific "in-scope" parts of the continuous system, invoking "re-initialisation" of the associated continuous variables, and leaves other "out-of-scope" continuous variables unchanged.

2.1 Rectangular Petri Net

A Petri-net (Silva, 1989) is a graphical notation with an underlying mathematical structure for modelling discrete event dynamic systems or DEDS. An n-dimensional Rectangular Petri net or simply *Rectangular Petri net* (RPN) is a 8-tuple ($\boldsymbol{N}$, $\mathbf{m_0}$, *init*, *inv*, *flow*, *scope*, *pre*, *post*) defined as follows: $\boldsymbol{N}$ is a Petri net with a set of places S, set of transitions T and initial marking $\mathbf{m_0}$; *init*, *inv*, and *flow* are all functions from $S \times \mathbb{N}$ to $\mathfrak{R}^n$; *scope* is a function from $S \cup T$ to $\{1, \ldots, n\}$ and *pre* and *post* are functions from T to $\mathfrak{R}^n$. For the purpose of simplicity, we shall assume that arcs are of weight 1. Thus each place s occupied by k token(s) is assigned values *init*(s, k), *inv*(s, k) and *flow*(s, k), which are the rectangular regions *initial region*, *invariant region* and *flow region* of s respectively. If a transition t *fires*, and changes the marking $\mathbf{m}$ to a new marking $\mathbf{m}'$, we shall write $\mathbf{m}[\boldsymbol{N},t\rangle\mathbf{m}'$ or simply $\mathbf{m}[t\rangle\mathbf{m}'$. A sequence $\mathbf{m}[t_1\rangle\ \mathbf{m}_1\ [t_2\rangle\ \mathbf{m}_2\ \ldots[\ t_k\ \rangle\ \mathbf{m}_k$ is called an *occurrence sequence* and $\mathbf{m_k}$ is *reachable* from $\mathbf{m}$. Each transition t is assigned two rectangular regions *previous region*, or *pre*(t), and the *posterior region*, or *post*(t). For each transition t, *scope*(t) is the set of all indices of the continuous variable $\mathbf{x} = (x_1, \ldots, x_n)$ which are effected by firing of the transition t.

A *state* of an RPN is an ordered pair ($\mathbf{m}$, $\mathbf{x}$) in which $\mathbf{m}$, the *discrete state*, is a reachable marking of the underlying Petri net ($\boldsymbol{N}$, $\mathbf{m_0}$) and $\mathbf{x}$, the *continuous state*, satisfies $\mathbf{x} \in \mathit{INV}(\mathbf{m}) = \cap\{\ \mathit{inv}(s\ ,\ \mathbf{m}(s)) \mid \mathbf{m}(s) > 0\}$ [2] and $\dot{\mathbf{x}} \in \mathit{FLOW}(\mathbf{m}) = \cap\{\ \mathit{flow}(s\ ,\ \mathbf{m}(s)) \mid \mathbf{m}(s) > 0\}$. For $\mathbf{m_0}$, the initial marking of the Petri net $\boldsymbol{N}$ and $\mathbf{x_0} \in \cap\ \{\ \mathit{init}(s\ ,\ \mathbf{m_0}\ (s)) \mid \mathbf{m_0}\ (s) > 0\ \}$, the state $(\mathbf{m_0},\mathbf{x_0})$

[1] Any subset of $\mathbb{R}^n$ is called a *region* and a region is a *rectangular region* if it is a Cartesian product of intervals (all) with rational end points. The set of all rectangular regions in $\mathbb{R}^n$ is denoted by $\mathfrak{R}^n$. Notice that $\mathfrak{R}^n$ includes the empty set because (a,a)=∅.

[2] $\mathbf{m}(s)$ denotes the number of tokens in place s under the marking $\mathbf{m}$.

is referred to as an *initial state* of the RPN ***A***. The continuous dynamics of the RPN must satisfy all rectangular constraints related to all marked places. If any of intersections is empty then the state is not physically attainable.

3. DYNAMICS OF RECTANGULAR PETRI NETS

The state of an RPN can be considered to evolve through *changes of continuous states* and *changes of discrete states.*

A *change of continuous state* of an RPN ***A*** from a state (**m**,**x**) into a new state (**m′**,**y**) over a time θ_0, (θ_0 is a non-negative real number), is denoted by (**m**,**x**) $\xrightarrow{\theta_0}$ (**m′**,**y**) and occurs if and only if the discrete part remains invariable i.e. **m** = **m′** and there is a smooth function ξ:[0, θ_0]$\rightarrow$ *INV*(**m**) such that $\xi(0) = \mathbf{x}$, $\xi(\theta_0) = \mathbf{y}$, and $\dot{\xi}(\theta) \in$ *FLOW*(**m**), for all $\theta \in (0, \theta_0)$. Such a function ξ is a *trajectory* in *INV*(**m**) with *derivatives* in *FLOW*(**m**).

Under a discrete state, continuous system behaviour that satisfies the associated rectangular constraints can terminate as the specific profile or trajectory enters the *pre*(t) condition region of an enabled transition and having entered the *pre*(t) condition region must terminate before the trajectory exists the *pre*(t) condition region.

A *change of discrete state* from (**m**,**x**) to (**m′**,**y**) results from the firing of transition t of ***N***, denoted by (**m**,**x**) $\xrightarrow{t}$ (**m′**,**y**), and occurs if and only if

a) the firing of t of ***N*** changes the marking from **m** to **m′** (i.e. **m**[***N***, t⟩**m′**)**,**

b) $\mathbf{x} \in pre(t)$ and $\mathbf{y} \in post(t)$

If $i \notin scope(t)$ then $[\mathbf{y}]_i = [\mathbf{x}]_i$ and if $i \in scope(t)$, then $[\mathbf{y}]_i \in [post(t)]_i$.

Reachability in RPNs. Consider RPN ***A*** with transitions $t_1, \ldots, t_k$, nonnegative real numbers $\theta_0,\theta_1,\ldots, \theta_k$, markings $\mathbf{m}_1, \mathbf{m}_2,\ldots, \mathbf{m}_k$ and continuous states $\mathbf{y}_0,\mathbf{y}_1,\ldots,\mathbf{y}_{k-1}, \mathbf{x}_1,\ldots,\mathbf{x}_k \in \mathbb{R}^n$. The sequence $(\mathbf{m}_0 , \mathbf{x}_0) \xrightarrow{\theta_0} (\mathbf{m}_0 , \mathbf{y}_0) \xrightarrow{t_1} (\mathbf{m}_1 , \mathbf{x}_1) \xrightarrow{\theta_1} (\mathbf{m}_1 , \mathbf{y}_1) \xrightarrow{t_2} (\mathbf{m}_2 , \mathbf{x}_2) \ldots (\mathbf{m}_{k-1} , \mathbf{y}_{k-1}) \xrightarrow{t_k} (\mathbf{m}_k , \mathbf{x}_k) \xrightarrow{\theta_k} (\mathbf{m}_k , \mathbf{x})$ is called an *occurrence sequence* of RPN ***A***. In this case we shall say (**m** , **x**) is *reachable* from ($\mathbf{m}_0 , \mathbf{x}_0$).

4. DURATION INTERVAL

A duration can be associated with each trajectory that satisfies the rectangular constraints of the system. Consider an n-dimensional RPN ***A*** with a discrete part represented by the Petri net ***N***. Given any A, $D \in \Re^n \setminus \{\varnothing\}$, the time durations of the set of trajectories in A with derivatives in D, create an interval which is characteristic of all trajectories in A with derivatives in D.

Theorem 1 (Existence of the Duration Interval). Assume that A,D,B,C$\in \Re^n\backslash\{\varnothing\}$ such that B$\subseteq$A and C $\subseteq$ A. There is an interval **I** (it can be the empty interval) such that $\theta_0 \in$ **I** if and only if there is a trajectory ξ: $[0,\theta_0] \to$ A with $\dot{\xi}:(0,\theta_0) \to$ D, $\xi(0) = \mathbf{x} \in$ B and $\xi(\theta_0) = \mathbf{y} \in$ C. The interval **I** is unique with respect to A,B,C and D and is denoted by *DurI*(A , D ; B , C).

Sketch of the proof: The set of all time durations of trajectories in A with derivatives in D starting from $\mathbf{x} \in$ B ending in $\mathbf{y} \in$ C is a convex set $\Gamma(\mathbf{x},\mathbf{y})$ of $\mathbb{R}$ and consequently $\Gamma(\mathbf{x},\mathbf{y})$ is an interval. The interval **I** is the union of all $\Gamma(\mathbf{x},\mathbf{y})$ where $\mathbf{x} \in$ B and $\mathbf{y} \in$ C. □

5. CALCULATION OF DURATION INTERVAL

Each trajectory of dimension n, n > 1, can be projected to n trajectories of dimension one. Assume that A,B,C,D are non-empty n-dimensional rectangular regions such that B$\subseteq$A and C $\subseteq$ A, and, for i=1,…,n, let A_i,B_i,C_i and D_i denote the projection of A, B, C and D to the i-th co-ordinate axis. then

$$DurI\ (A,D;B,C) = \bigcap_{1 \le i \le n} DurI(A_i,D_i;B_i,C_i). \qquad (1)$$

Let $A_i = [a_l, a_u]$, $B_i = [b_l, b_u]$ and $C_i = [c_l, c_u]$ be nonempty, bounded real interval such that B_i and $C_i \subseteq A_i$, $b_l \le c_l$ and $b_u \le c_u$. Suppose that $D_i = [d_l, d_u]$, where $d_l \ge 0$.

$$\text{If } c_l \le b_u, \text{ then } \quad DurI(A_i,D_i;B_i,C_i) = [0\ ,\ (c_u - b_l) / d_l] \qquad (2)$$

$$\text{If } b_u < c_l, \text{ then } DurI(A_i,D_i;B_i,C_i) = [(c_l - b_u) / d_u\ ,\ (c_u - b_l) / d_l\ ] \qquad (3)$$

Now, assume that **m** is a reachable marking of ***N***, i.e. there is an occurrence sequence as follows

$$\mathbf{m}_0\ [t_0\rangle\ \mathbf{m}_1\ [t_1\rangle\ \mathbf{m}_2\ \ldots\ \mathbf{m}_k\ [\ t_k\ \rangle\ \mathbf{m} \qquad (4)$$

Moreover, let **m** occur as the discrete part of some state in the RPN ***A***. Hence firing of the transition t_k starts a continuous trajectory ξ from a point **x** ending in **y** in a time duration θ_0, i.e. $(\mathbf{m}\ ,\ \mathbf{x}) \xrightarrow{\theta_0} (\mathbf{m}\ ,\ \mathbf{y})$. The following algorithm determines the set of possible time duration of the continuous state trajectories, related to the discrete event marking **m** as in (4).

.

Algorithm 1. During the discrete event state **m**, the values of ξ lies in A = *INV*(**m**) while the derivatives of ξ belongs to D = *FLOW*(**m**). In order to find the set B of the starting point of the trajectory ξ notice that for each co-ordinate with $i \in$ *scope*(t_k), the i-th co-ordinate of ξ can start at any value in [*post*(t_k)]$_i$. As a result, [B]$_i$ = [*post*(t_k)]$_i$. For $i \notin$ *scope*(t_k), we must find the values of the last re-initialisation of the i-th co-ordinate of ξ, i.e. find the largest index $0 \le r < k$ such that $i \in$ *scope*(t_r). When the discrete state was in $\mathbf{m}_{r+1}$, the i-th co-ordinate of ξ could start from any point of β_r = [*post*(t_r)]$_i$, if $r > 0$ and from β_r = [*INIT*($\mathbf{m}_0$)]$_i$, if $r = 0$. Assuming that we know the duration interval for the marking $\mathbf{m}_{r+1}$, then we can calculate the interval $\beta_{(r+1)}$ of all possible values for the i-th co-ordinate of ξ at the time of firing of the next transition $t_{(r+1)}$. Fig.1 pictures the situation in the (θ , x_i) plane.

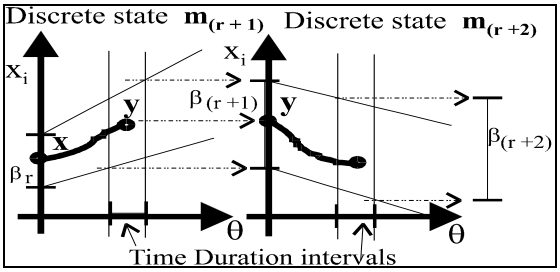


Figure 1. Trajectory evolving through states

Similarly, $\beta_{(r+1)}$, … , β_k can be defined and calculated. The last interval obtained, i.e. β_k which is related to the firing of t_k, is the interval [B]$_i$. We shall refer to the above procedure of finding [B]$_i$ as "back tracking" from the marking **m**. Considering that there are n co-ordinates to back track, we need to go as far as $\mathbf{m}_j$, where $0 \le j \le k$ is the largest index that $\cup$ { *scope*(t_r) | $j \le r \le k$ } = {1, … , n}. Calculating [B]$_i$, leads us to B.

For each enabled transition t under **m** with *scope*(t) $\neq \varnothing$ let C = *pre*(t) the set of possible terminating points for ξ, if t fires and changes the discrete state from **m**. Using (2) and (3) calculate *DurI*(A,D;B,C) and denote it with *DurI*($t_j \ldots t_k$; **m** ; t). For each enabled transition t under **m** with *scope*(t) = $\varnothing$ let *DurI*($t_j \ldots t_k$; **m**; t) = [0 , 0] = {0}. For the case $\mathbf{m} = \mathbf{m_0}$, we shall write *DurI*($-$; $\mathbf{m_0}$; t), where t is enabled under the initial marking.

6. FEASIBLE DURATION INTERVALS

The set of enabled transition under the marking **m** is denoted by *ENABLED*(**m**). Then for each $t \in$ *ENABLED*(**m**) there is an interval *DurI*(

$t_j \ldots t_k$; **m**; t), where ($t_j \ldots t_k$) refers to the occurrence sequence for **m** and the interval is calculated using the Algorithm 1. In the case of conflicting transitions the intervals must be modified. For example, if there are two enabled transitions t and t′ with intervals [1, 2] and [0, 1.5], respectively, then the trajectory ξ "must" end by the time 1.5 as a result of firing of t′ and, the feasible time for firing of t is [1, 1.5] instead of [1, 2]. The following algorithm generates the feasible duration intervals related to the marking **m**.

Algorithm 2. Consider a marking **m** and let β denote the smallest of the final points of the intervals in $\Lambda = \{ DurI(t_j \ldots t_k; \mathbf{m}; t) \mid t \in ENABLED(\mathbf{m})\}$, where $\emptyset \notin \Lambda$. Define the *Modifying Interval*, denoted by $MInt(t_j \ldots t_k; \mathbf{m}; t)$, to be equal to $[0,\beta]$. Note that if Λ contains the empty interval then $MInt(t_j \ldots t_k ;\mathbf{m} ;t) = \emptyset$. Now, a continuous trajectory, starting as a result of firing t with the discrete part **m, '**must' end at the latest at β. Thus one can consider $DurI(t_j \ldots t_k; \mathbf{m}; t) \cap MInt(t_j \ldots t_k; \mathbf{m}; t)$ instead of $DurI(t_j \ldots t_k; \mathbf{m}; t)$ as the set of feasible time duration for a continuous change of state at **m** ending by firing t. If the intersection is empty, firing t is not attainable.

7. RRG AND VERIFICATION OF RPNS

The main concern of this paper is the verification of the functional behaviour of a system. In the following, we shall restrict our consideration to an RPN ***A*** with an underlying bounded[3] Petri net ***N***. Also, for technical reasons, we shall assume there is no reachable marking **m** with $\mathbf{m}[t\rangle\mathbf{m}$. The principal step in the process of verification involves the generation of Rectangular Reachability Graph, RRG, which is a directed graph with nodes labelled by reachable markings of ***N***. Each edge of RRG is labelled by both a transition and an interval of all feasible time durations of continuous trajectories ending by firing of the transition. Many behavioural properties of conventional Petri nets, including liveness[4], reversibility, home state (Juan et al. (1998)) and concurrency set (Azzopardi and Holding (1997)), can be extended to RPNs. Verification of an RPN can be expressed in terms of liveness properties (things that the system should do) and safety properties (things that the system should not do) where these terms are used in the formal methods sense (Lamport, 1997). Such functional and safety properties of the system can be translated to reachability problems and verified by searching the RRG.

[3] A Petri net ***N*** with an *initial* marking $\mathbf{m_0}$ is called *bounded* if there is a fixed number k such that under each reachable marking **m**, the number of tokens in each place does not exceed k.

[4] A Petri net is called *live* if for each transition t and each reachable marking **m** a marking **m**′ that enables t can be reached.

7.1 Construction of RRG(*A*)

To construct RRG(***A***) we shall start with an "*initial node*" $\alpha 0$ marked by the initial marking $\mathbf{m_0}$. Apply Algorithm 1, for each enabled transition t under $\mathbf{m_0}$, calculate *DurI*(−; $\mathbf{m_0}$; t). Apply Algorithm 2 to find feasible duration interval I_0_t. For each t with $\mathbf{m_0}$ [t⟩ $\mathbf{m_0}$_t and I_0_t ≠ ∅, create a node α_0_t marked by $\mathbf{m_0}$_t and connect $\alpha 0$ to α_0_t via an edge and label it with (t , I_0_t). For each newly created node α marked by **m**, apply the Algorithm 1 and calculate all *DurI*(t_j…t_k ; **m** ; t), where t is an enabled transition under **m**. Apply Algorithm 2 to find feasible duration intervals I_t. If there is an already existing node β of the RRG, labelled by **m** and the identical corresponding feasible duration intervals, then cancel α and redirect all inputs edges to α, into β. Otherwise, for each enabled transition t with **m** [t⟩ **m**_t and I_t ≠ ∅, create a node α_t marked by **m**_t and connect α to α_t via an edge and label it with (t , I_t).

8. APPLICATION TO A WRAPPING MACHINE

Consider the simplified wrapping machine of Fig.2 (left). The product (JOB) is supplied via a belt. Proximity sensors detect JOBs. The packaging film is supplied by unwinding a roll of printed foil. The printed image needs to be positioned centrally on the product; this is done using printed marks (TAG) which are detected by sensors. The packaging film is then formed into a tube via a *funnel*, and a *longitudinal sealing* roller welds the two edges of the film together. To produce individually packaged products, the tube is sealed between packs by a *lateral sealer*.

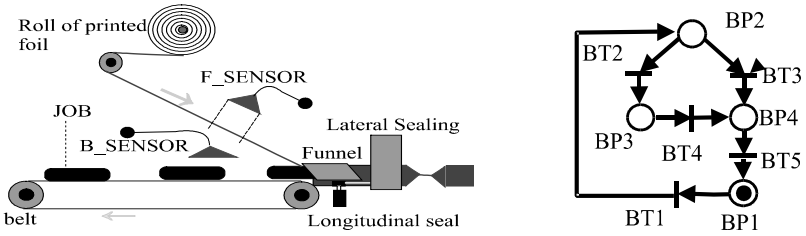


Figure 2. Wrapping machine (left) and the associated Petri net (right)

A PN model of the belt is shown in Fig.2 (right). The belt is driven by a motor, which is controlled to track a reference model. On arrival of a new JOB, BP1, the belt keeps following the previous reference trajectory. On this path there is a *decision point*. When the JOB arrives at the decision

point, BT1, if the film is ready then the Belt *inserts*, BT3, the JOB in the funnel. Otherwise, the decision is taken to 'abort', BT2, and *wait*, BP3, for the film to be re-positioned. When the film is re-positioned, the belt is restarted to follow the usual motion profile and *inserts*, BT4, the JOB in the funnel. During the insert process the Belt is said to be *engaged*, BP4, until the insert operation is *completed*, BT5, and the belt is ready for a new JOB.

Let the film be subject to the same type of control as the belt, thus giving a similar Petri net. The Belt and Film are synchronised together as follows. Assume that Belt is engaged, i.e. BP4 is marked, then the wrapping can terminate (BT5) only if TAG is in position. Therefore SP6 is added as a precondition for firing of BT5. Arrival of the TAG (i.e. firing of BT3 or BT4) puts a token in SP6. However, a new JOB can not start until the Film wraps the previous JOB. Consequently, the place SP1 occupied by a token is added such that firing of FT3 or FT4 (which mark the starting of the process of wrapping) disables BT1. Also, it is not possible to insert the JOB unless the Film is in position. Therefore, SP5 is added as preconditions to FT3. Similarly, places SP4, SP5 and SP1 can be defined. The reachability graph of Petri net ***N*** of Fig.3 is shown in Fig. 4.

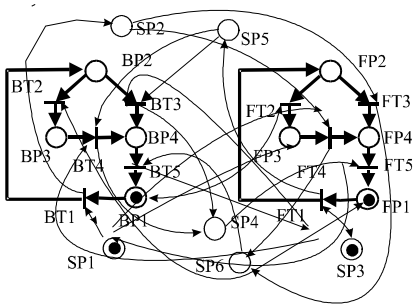
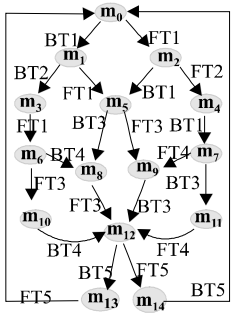


Figure 3. PN Model of the Wrapping Machine (left) and its Reachability Graph (right)



$\mathbf{m_0}$ = BP1 FP1 SP1 SP3
$\mathbf{m_1}$ = BP2 FP1 SP1 SP2 SP3
$\mathbf{m_2}$ = BP1 FP2 SP1 SP3, SP5
$\mathbf{m_3}$ = BP3 FP1 SP1 SP2 SP3
$\mathbf{m_4}$ = BP1 FP3 SP1 SP3 SP5
$\mathbf{m_5}$ = BP2 FP2 SP1 SP2 SP3 SP5
$\mathbf{m_6}$ = BP3 FP2 SP1 SP2 SP3 SP5
$\mathbf{m_7}$ = BP2 FP3 SP1 SP2 SP3 SP5
$\mathbf{m_8}$ = BP4 FP2 SP1 SP2 SP4
$\mathbf{m_9}$ = BP2 FP4 SP3 SP5 SP6
$\mathbf{m_{10}}$ = BP3 FP4 SP3 SP5 SP6
$\mathbf{m_{11}}$ = BP4 FP3 SP1 SP2 SP4
$\mathbf{m_{12}}$ = BP4 FP4 SP4 SP6
$\mathbf{m_{13}}$ = BP1 FP4 SP3 SP4
$\mathbf{m_{14}}$ = BP4 FP1 SP1 SP2

Figure 4. Reachability Graph of the Wrapping machine

Assume that $x_1(\theta)$ and $x_2(\theta)$ denotes the displacement of the JOB and TAG from the origin of an inertial coordinate system at time θ, respectively, and let $\mathbf{x} = (x_1 , x_2)$. To generate representative dynamics and timing for the continuous part of the RPN an associated continuous system model was simulated using SIMULINK3, with a sensor sampling time of 0.005 sec. Now we can calculate *inv* and *flow* of different markings. For example, for the initial marking $\mathbf{m_0}$ = BP1 FP1 SP1 SP3, $inv(\mathbf{m_0}) = ([0, 0.062] \times \mathbb{R}) \cap (\mathbb{R} \times [0, 0.065]) \cap (\mathbb{R} \times \mathbb{R}) \cap (\mathbb{R} \times \mathbb{R}) = [0, 0.062] \times [0, 0.065]$. Since *pre*(BT1) = [0.06, 0.062] × ℝ and *pre*(FT1) = ℝ × [0.06, 0.065], applying (1) with A= $inv(\mathbf{m_0})$, D=$flow(\mathbf{m_0})$, C=pre(BT1) and C=pre(FT1), while B = init($\mathbf{m_0}$), gives $DurI(-; \mathbf{m_0}; BT1) = [0, 0.065]$ and $DurI(-; \mathbf{m_0}; FT1) = [0, 0.069]$. As a result, by Algorithm 2, the Modifying interval is equal to [0, 0.065]. Consequently, under the marking $\mathbf{m_0}$ the feasible firing interval is equal to [0, 0.065].

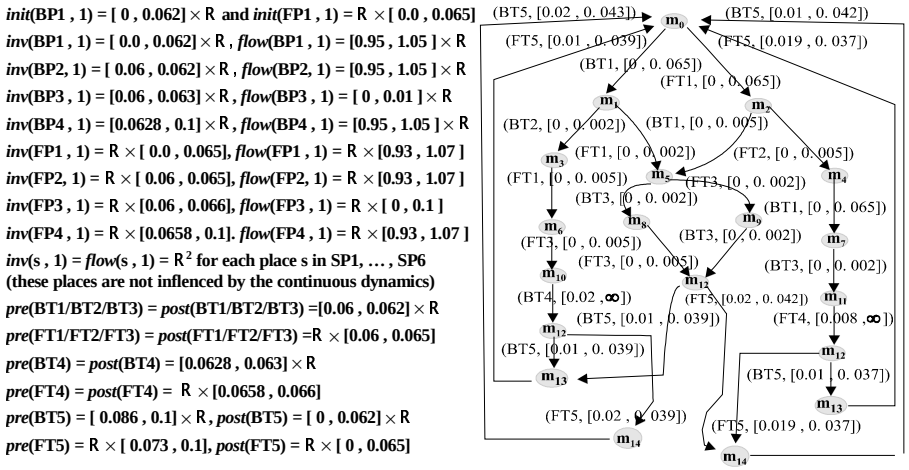


Figure 5. RRG of the Wrapping Machine

Fig.5 denotes the RRG of the wrapping machine. Searching RRG shows that the system is live, 1-bounded and without any deadlock. Further analysis shows that starting from the initial marking, all possible paths end in $\mathbf{m_{12}}$, which is the JOB in funnel and the TAG in position, i.e. the action of wrapping. It can also be checked that no JOB or TAG are wasted. A comparison with Fig.4 shows that the process durations constraint the hybrid system behaviour.

9. CONCLUSION

This paper has shown how Rectangular Petri nets, or RPN's, can be used to describe models of hybrid systems. The paper shows how the RPN brings together and integrates (i) the Petri net that describe the discrete event part of the system and (ii) the differential inclusions (constrained to rectangular regions of the state space) that describe the continuous part of the system. The paper also explains how RPN theory, and the Rectangular Reachability Graph, can be used to analyse the behaviour of the hybrid system. The method has been demonstrated by application to a typical manufacturing process.

ACKNOWLEDGEMENT

This work was supported by EPSRC (UK) Grant GR/L31234.

REFERENCES

Antsaklis, Panos J., Kohn, Wolf, Nerode, Anil, Shastry, Shankar, (eds.), *Hybrid Systems IV*, LNCS 1273, 1997.

Azzopardi D., Holding D.J.. Petri nets and OMT for modelling and analysis of DEDS's. IFAC Jou. Control Engineering Practice 1997; 5:1407–1415

Cheung S., Kramer J. Checking safety properties using compositional reachability analysis. ACM Trans. Soft. Eng. Method 1999; 8:49-78

Henzinger T., Kopke P., Puri A., Varaiya P. What's decidable about Hybrid Automata. Jour. Comput. Sys. Sci.1998; 57:94-124

Jiang J., Azzopardi D., Holding D. J., Carpenter G.F., Sagoo J.S. Real-time synchronisation of multi-axis high-speed machines, from SFC specification to Petri nets verification. IEE Proc. Control Theo. and Application 1996; 143:164-170

Juan E., Tsai J., Murata T. Compositional verification of concurrent systems using Petri-Net-Based Condensation Rules. ACM Trans. Prog. Lang. and Sys. 1998; 20:917-979

Lamport L. Proving the correctness of multiprocess programs. IEEE Trans. Software Eng. 1997; SE-3:125-143

Le Bail J., Alla H., David R. Hybrid Petri nets. Proc. 1st European Control Conference, Grenoble, France, 1991; 187-191

Murata, Tadao. "Modelling and analysis of concurrent systems." In *Handbook of software Engineering*, C. Vick and C. Ramamoorthy eds. Van Nostrand Reinhold 1984.

Silva M. Petri nets and flexible manufacturing. Advances in Petri nets, LNCS 1989