# Modelling Automated Manufacturing Systems with Hybrid Automata*

Fabio Balduzzi (*) Alessandro Giua (**), Carla Seatzu (**)

(*) Dip. di Automatica ed Informatica, Politecnico di Torino, Italy
Email: balduzzi@polito.it

(**) Dip. di Ingegneria Elettrica ed Elettronica, Università di Cagliari, Italy
Email: (giua,seatzu)@diee.unica.it

**Abstract**

In this paper we present a hybrid formulation for the modelling and control of automated manufacturing systems. We define a hybrid manufacturing system as a dynamical system that combines both discrete and continuous dynamics. Each service (a machine coupled with a finite buffer) is described as an elementary hybrid automaton and the whole network of services, that is subject to configuration–based specifications, is described in a modular fashion as the concurrent composition of elementary services. The paper focuses on essential questions associated with regularization and composition of hybrid automata as related to the possibility of efficiently dealing with complex hybrid systems.

## 1 Introduction

In this paper we develop a hybrid model of discrete event dynamic processes that combines aspects of automata and systems theory, thus exploring potential interconnections between those classes of systems. We consider automated manufacturing systems with unreliable machines, buffers of finite capacity, arbitrary service time distributions and routing policies, where several parts of a single class of products are circulated and processed. A machine coupled with a buffer for storing arriving parts defines an *elementary service* (ES).

The main contribution of this work is to develop a mathematical description of the process which specifies a mechanism for composing elementary services. Each elementary service is described as a hybrid automaton (HA) [1, 7]. A general queueing network is obtained by combining ES in a modular fashion by means of the concurrent composition operator which allows the definition of the hybrid automaton representing the whole network of services.

The dynamics of a hybrid manufacturing system is described by splitting the discrete event processes into two hierarchical layers and defining what we call *macro–events* and *macro–states*. At the lower level, the microscopic behavior of arrivals and departures of parts to(from) each

*Published as: F. Balduzzi, A. Giua, C. Seatzu, "Modelling Automated Manufacturing Systems with Hybrid Automata," *Proc. Work. on Formal Methods and Manufacturing* (Zaragoza, Spain), pp. 33-48, September, 1999.

machine is modeled using first order fluid approximations. Such model allows the mathematical formulation of the continuous dynamics of the system. At the higher level a discrete event model will represent the transitions through a sequence of macro–states at the occurrence of a limited number of events, the macro–events. Both levels of the process are represented by a hybrid automaton which exhibits both continuous and discrete changes.

## 2 Description of the Model

The production process considered in this work consists of a set of $n$ single–server stations, denoted $M_i$, for $i = 1, \ldots, n$, serving a single class of products. Parts move from machine $M_i$ to $M_j$ according to their production cycle and are queued in buffers, one for each machine, with the initial one (input buffer) acting as an unlimited supply of parts and the final buffer acting as a limited storage area for collecting finished products, thus representing the production target. The buffers have finite capacity $C_i$ and the machines are unreliable.

We consider operation-dependent failures and we define for each machine the *production volumes before a machine fails* and the *repair times*, denoted $w_i$ and $d_i$ respectively, both assumed independent identically distributed random variables. Machine service times are assumed independent random variables with identical distribution with finite mean and variance. The maximum average production rate of machine $M_i$ is denoted $V_i$.

The evolution in time of the production process is discussed within a framework that distinguishes two levels of aggregation. The lower layer represents the microscopic behavior of arrivals and departures of parts to/from each machine (micro–events). It will be modeled in an aggregate view by using first order fluid approximations [3]. At the higher layer a discrete event model will represent the transitions of the process through a sequence of macro–states, at the occurrence of the macro–events.

### 2.1 The Microscopic Layer

Let $\Delta_k = [t_k, t_{k+1})$, for $k = 0, 1, 2, \ldots$, be the interval of time between the occurrence of consecutive macro–events at time $t_k$ and $t_{k+1}$, that we call *macro–period*, and let $v_{i,j}(k)$ be the constant average flow rates of parts from machines $M_i$ to $M_j$.

The microscopic behavior of a production system during a macro–period can be approximated by the following three processes defined for each machine $M_i$:
(I) the buffer levels

$$x_i(t) = x_i(t_k) + [v_{in,i}(k) - v_{out,i}(k)](t - t_k), \tag{1}$$

(II) the production volume processed by the machine since the last repair (used to evaluate the machine breaking time)

$$\chi_i(t) = \chi_i(t_k) + v_{out,i}(k)(t - t_k), \tag{2}$$

(III) the time spent by the machine under repair since the last failure

$$s_i(t) = s_i(t_k) + (t - t_k) \tag{3}$$

where $t \in [t_k, t_{k+1})$ for all these processes, and

$$v_{in,i}(k) = \sum_h v_{h,i}(k), \quad v_{out,i}(k) = \sum_j v_{i,j}(k)$$

are the inflow and outflow rates of parts of each machine.

Note that the value of $\chi_i(t)$ in Equation (2) will be reset to 0 after each failure and its value will not increase until the machine is repaired. Similarly, the value of $s_i(t)$ in Equation (3) will be reset to 0 after each repair and its value will not increase until the machine fails again.

### 2.2 The Macroscopic Layer

At the macroscopic level the evolution in time of the system through a sequence of macro–states can be described by a finite automata with states given by a finite set of admissible configurations of machines status (*operational* or *broken*) and buffer status (*full*, *not full–not empty*, *empty*) and with transitions represented by the macro–events (*failure*, *repair*, *buffer full* and *buffer empty*).

We denote the set of macro–event types by $\mathcal{E}_i = \{f_i, r_i, be_i, bf_i, bn_{e,i}, bn_{f,i}\}$ whose elements are defined as follows:

- $f_i$ (*failure of machine* $M_i$). After a machine is repaired, failures will occur after the production volume $w_i$.
- $r_i$ (*repair of machine* $M_i$). When a machine fails, it will be repaired after $d_i$ time units.
- $bf_i$ (*buffer full at machine* $M_i$). The buffer level reaches its capacity $C_i$ while $v_{in,i}(t) \geq v_{out,i}(t)$.
- $be_i$ (*buffer empty at machine* $M_i$). The buffer level reaches 0 while $v_{out,i}(t) \geq v_{in,i}(t)$.
- $bn_{e,i}$ (*buffer not empty at machine* $M_i$). Upon the occurrence of exogenous macro–events, apart from the machine status, changes will occur in the inflow and outflow rates of parts of machine $\mathrm{M}_i$, thus changing its buffer content.
- $bn_{f,i}$ (*buffer not full at machine* $M_i$). Upon the occurrence of exogenous macro–events, when the machine is operational changes will occur in the inflow and outflow rates of parts of machine $\mathrm{M}_i$, thus changing its buffer content.

The machine status takes the following symbolic values: O (*machine operational*) and B (*machine broken*). Note that when a machine breaks down then its production rate is $v_{out,i} = 0$, while it must result $v_{out,i} \leq V_i$ if $\mathrm{M}_i$ is operational. The buffer status takes the following symbolic values: F (*buffer full*), E (*buffer empty*) and N (*Buffer not full–not empty*).

Let $\mathcal{M} = \{\mathrm{O}, \mathrm{B}\}$ and $\mathcal{B} = \{\mathrm{F}, \mathrm{E}, \mathrm{N}\}$ be the machine status and the buffer status set types respectively. The finite set of admissible macro–states (operational modes) is denoted by $Q = \{(mb_1, \ldots, mb_n) \mid m \in \mathcal{M}, b \in \mathcal{B}\}$. Each element $q \in Q$ combines the functional status of all services and may only change due to the occurrence of the macro–events $e \in \mathcal{E}$. In Fig. 1 we have depicted the finite automata $\mathcal{Z} = (Q, \mathcal{E}, \delta, q_0)$ for the elementary service $\mathrm{S}_i$. Here $Q$ is the finite set of admissible macro–states as previously defined, $\mathcal{E}$ is the input alphabet,

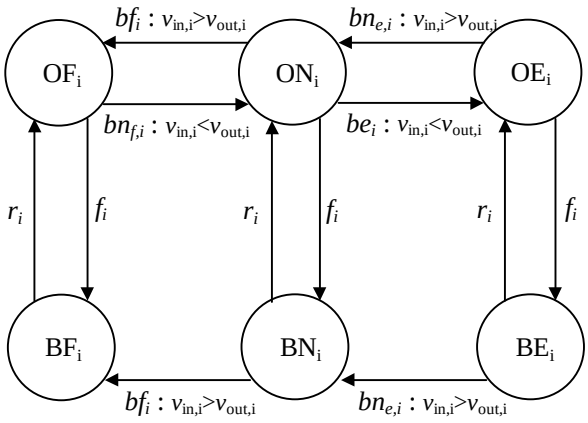


Figure 1: The finite automaton of an elementary service.

$\mathcal{E} = \{\ldots, f_i, r_i, bf_i, be_i, bn_{e,i}, bn_{f,i}, \ldots\}$, $\delta : Q \times \mathcal{E} \to Q$ is the transition function and $q_0 \in Q$ is the initial state.

The transitions of the finite automaton define the interlacing of Equations (1)–(3) and drive the evolution of the system whose macro–behavior can be described by a hybrid automaton with inputs. The continuous dynamics of the HA is given by Equations (1)–(3). We define an input vector

$$\mathbf{u}(k) = [\ldots, v_{i,j}(k), \ldots]^T \tag{4}$$

with entries given by the constant average machine flow rates.

A macro–event leads from one discrete state to another. When a machine fails or gets repaired then it must result $\chi_i(t_{k+1}) = w_i$ or $s_i(t_{k+1}) = d_i$. When a buffer gets full or empty then the condition $x_i(t_{k+1}) = C_i$ or $x_i(t_{k+1}) = 0$ will be satisfied. For simplicity of presentation we consider in this work production systems with a single class of parts flowing through and in what follows we do not take into account fluctuations in the average values of state variables $x_i$, $\chi_i$ and $s_i$ due to the approximations introduced by the fluid model. The complete stochastic state variable model can be found in [2].

Note that each macro–state defines the feasible region, denoted $\mathcal{S}(k)$, for the average machine production rates $v_{i,j}(k)$ that enter this model as input variables. We indicate with $I_o(k)$ and $I_d(k)$ the sets of indices of operational and down machines, $I_f(k)$ and $I_e(k)$ the sets of indices of full and empty buffers, during the $k$–th macro–period, respectively.

**Definition 2.1** *An input vector* $\mathbf{u}(k) \in \mathcal{S}(k)$ *is* admissible *if it is a feasible solution of the following set of linear inequalities:*

$$\begin{cases} (a) & 0 \leq \sum_j v_{i,j}(k) \leq V_i, \quad \forall i \in I_o(k) \\ (b) & \sum_j v_{i,j}(k) = 0, \quad \forall i \in I_d(k) \\ (c) & \sum_h v_{h,i}(k) \leq \sum_j v_{i,j}(k), \forall i \in I_f(k) \\ (d) & \sum_j v_{i,j}(k) \leq \sum_h v_{h,i}(k), \forall i \in I_e(k) \\ v_{i,j} \geq 0 \end{cases} \tag{5}$$

*The* consistency constraint set *(CCS) (5) will be denoted* $\mathbf{g}(k, \mathbf{u}(k)) \leq \mathbf{0}$. ■

Constraints (a) and (b) bound the machine production rates according to the current machine status, while constraints (c) and (d) bound the inflow and outflow rates of parts according to the current buffer status. The region $\mathcal{S}(k)$ defined by $\mathbf{g}(k, \mathbf{u}(k)) \leq 0$ is a convex polyhedron whose vertices are basic solutions of any linear programming problem with objective function of the form $J = \mathbf{a}^T(k)\mathbf{u}(k)$ and subject to the CCS. Any admissible input vector $\mathbf{u}(k)$ corresponds to a point within the feasible region $\mathcal{S}(k)$ and the boundary represents all those input policies aimed at maximizing a given linear objective function.

Let us now consider a dynamic control policy that generates a solution for the average machine production rates obtained for each macro–period. Any solution represents the average flow rates vector $\mathbf{u}(k)$ to be expected for the macro–period $\Delta_k$.

# 3 Hybrid Automata Model

In this section we show how it is possible to derive a HA [1, 7, 4] model of the manufacturing system described in the previous sections.

We consider a system with a continuous state vector $\mathbf{x} \in \mathbb{R}^n$ and a continuous input vector $\mathbf{u} \in \mathbb{R}^p$. The system may be in a finite number of discrete states called locations. In each location the state is constrained to belong to a subset of $\mathbb{R}^n$ called X_invariant.

As in the standard definition of HA we consider a guard $g \subset \mathbb{R}^n$ and a jump relation $j \subset \mathbb{R}^n \times \mathbb{R}^n$. An edge is enabled when $\mathbf{x} \in g$ and the state jumps from $\mathbf{x}$ to $\mathbf{y}$ according to the jump condition, i.e., if $(\mathbf{x}, \mathbf{y}) \in j$.

While in the standard definition of HA the activity function associates to each location a differential inclusion, in our definition we associate to each location a differential equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. However, the input vector $\mathbf{u}$ may take values in a set $\mathcal{S} \subset \mathbb{R}^p$ called U_invariant. Thus we still have that the set of all possible derivatives is not a single vector but, for a given value of $\bar{\mathbf{x}}$, may take values in a subset of $\mathbb{R}^n$ defined by $\{\mathbf{f}(\bar{\mathbf{x}}, \mathbf{u}) \mid \mathbf{u} \in \mathcal{S}\}$.

The set of differential equations, X_invariants, U_invariants, guards and jump relations of interest are called *Diff_Eq*, *X_inv*, *U_inv*, *Guard* and *Jump* respectively.

**Definition 3.1** *A HA is a structure* $H = (L, \Sigma, act, x\_inv, u\_inv, E)$ *where:*

- $L$ *is a finite set of locations.*
- $\Sigma$ *is a finite alphabet of symbols used to label the edges.*
- $act : L \rightarrow$ *Diff_Eq specifies the dynamics at each location. We denote* $act(l) = \mathbf{f}(\mathbf{x}, \mathbf{u})$ *the differential equation* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ *at location* $l$.
- $x\_inv : L \rightarrow X\_inv$ *specifies the set of all admissible state vectors at each location.*
- $u\_inv : L \rightarrow U\_inv$ *specifies the set of all admissible input vectors at each location.*
- $E \subset L \times \Sigma \times Guard \times Jump \times L$ *is the set of edges. An edge* $e = (l, \sigma, g, j, l')$ *is an edge from location* $l$ *to* $l'$*, labelled with the symbol* $\sigma$*, with guard* $g$ *and jump relation* $j$.

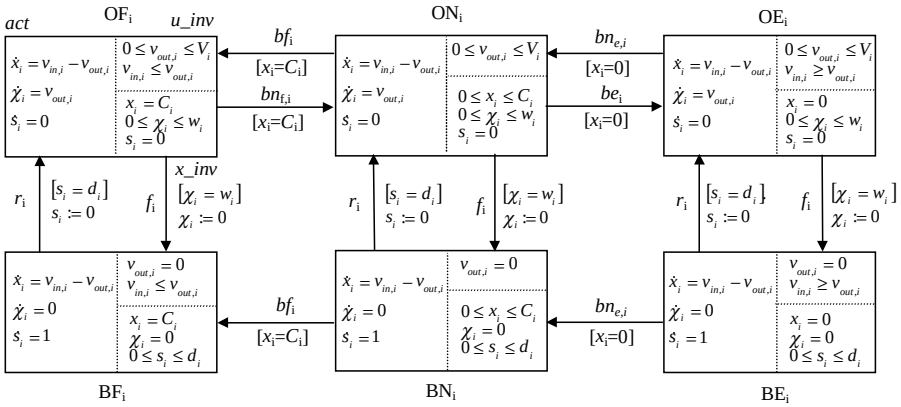


Figure 2: The hybrid automaton $H$ of an elementary service.

### 3.1 The model of an elementary service

We now consider an elementary service $\mathrm{S}_i$ composed of a buffer of finite capacity and of an unreliable machine.

Note that in this model we use $v_{in,i}$ and $v_{out,i}$ to denote the input/output flows: this is the case of a single-input single-output service. In the case of multiple-input multiple-output services we would also need to specify each single input flow $v_{h,i}$ from machine $h$ to machine $i$, and each single output flow $v_{i,j}$ from machine $i$ to machine $j$.

Such a service can be described by the HA $H_i = (L_i, \Sigma_i, act_i, x\_inv_i, u\_inv_i, E_i)$ shown in Fig. 2. The continuous state is $\mathbf{x} = [x_i, \chi_i, s_i]^T$ while the input vector is $\mathbf{u} = [v_{in,i}, v_{out,i}]^T$.

The set of locations is $L = \{OF_i, ON_i, OE_i, BF_i, BN_i, BE_i\}$ with obvious notation as described in section 2.2 and Fig. 1. Each location represents the discrete state of the service (machine and buffer).

The alphabet $\Sigma = \{f_i, r_i, be_i, bf_i, bn_{e,i}, bn_{f,i}\}$ corresponds to the set of macro–events $\mathcal{E}$ described in section 2.2.

The dynamics at each location $act(l)$ can be obtained by differentiating Equations (1)-(3). In particular

$$act(l) = \mathbf{B}_l \mathbf{u} + \mathbf{C}_l$$

where

$$\mathbf{B}_l = \begin{bmatrix} 1 & -1 \\ 0 & \alpha \\ 0 & 0 \end{bmatrix} \qquad \text{where} \quad \alpha = \begin{cases} 1 & \text{if } l \in \{OF_i, ON_i, OE_i\} \\ 0 & \text{if } l \in \{BF_i, BN_i, BE_i\} \end{cases}$$

$$\mathbf{C}_l = \begin{bmatrix} 0 \\ 0 \\ \beta \end{bmatrix} \qquad \text{where} \quad \beta = \begin{cases} 0 & \text{if } l \in \{OF_i, ON_i, OE_i\} \\ 1 & \text{if } l \in \{BF_i, BN_i, BE_i\} \end{cases}$$

We note that in this hybrid automaton there is no explicit dependency of the activity set from the state $\mathbf{x}$.

The set $x\_inv$ of admissible state vectors changes from one location to another. As an example in location $ON_i$ this set takes the form:

$$x\_inv(ON_i) = \begin{cases} 0 \leq x_i \leq C_i \\ 0 \leq \chi_i \leq w_i \\ s_i = 0 \end{cases}$$

since this location represents the discrete state in which the buffer is not full–not empty and the machine is operational.

The set $u\_inv$ of admissible input vectors also changes from one location to another. It corresponds to the feasible solutions $\mathcal{S}$ of the constraints set (5). As an example in location $OF_i$ this set takes the form:

$$u\_inv(OF_i) = \begin{cases} 0 \leq v_{out,i} \leq V_i \\ v_{in,i} \leq v_{out,i} \end{cases}$$

since this location represents the discrete state in which the buffer is full and the machine is operational.

Each edge in the set $E$ is represented as an arrow with associated its symbol, its guard (between brackets) and its jump relation. As an example the edge from location $OF_i$ to location $BF_i$ has symbol $f_i$, guard $[\chi_i = w_i]$ and jump relation $\chi_i := 0$ because it resets the continuous state variable $\chi_i$.

This model represents exactly the hybrid counterpart of the finite automaton depicted in Fig. 1. It should be noted that a reduced HA equivalent to this may be obtained by removing locations $OF_i, BF_i, OE_i, BE_i$. In the reduced HA, the constraints in $u\_inv$ that derive from the fact that the buffer can be full or empty are implicitly enforced by the $x\_inv$ constraints. As an example when the buffer is full, i.e., $x_i = C_i$, no input vector with $v_{in,i} > v_{out,i}$ is allowed since this will lead to $\dot{x}_i > 0$ and it would violate the $x\_inv$ constraint $x_i \leq C_i$. In spite of the simpler structure of the reduced HA, we think that the complete model with six locations is better because it allows a neat separation between control inputs and state variables. In fact $u\_inv$ does not depend on the state values and the state constraints $x\_inv$ are only used to validate the guards and force the transitions among locations.

### 3.2 Dynamic control policy and execution

The dynamic control policy adopted in this work assumes that in each location $l \in L$ the choice of the input vector $\mathbf{u}$ is obtained as the solution of a linear programming problem of the form

$$\begin{array}{ll} \max_{\mathbf{u}} & \mathbf{c}_l^T \cdot \mathbf{u} \\ \text{s.t.} & \mathbf{u} \in u\_inv(l) \end{array}$$

where the objective function vector $\mathbf{c}_l$ as well as the constraint set $u\_inv(l)$ depend on the current location. The optimal solution $\mathbf{u}_l^o$ will always lay on the boundary of the feasible region defined by the linear constraint set, and it is usually unique (if it is not we may always introduce additional constraints to resolve the ambiguity). The input vector will be maintained constant at the value $\mathbf{u}_l^o$ until the system leaves the current location. Such a control policy is called *locally optimal* since it provides the optimal solution for a single time step within a given location.

This dynamic control policy results in an activity set $act(l) = \mathbf{f}(\mathbf{u}_l^o)$ that consists only of a single point in $\mathbb{R}^n$. Such an hybrid automaton is called in the literature *multirate automaton* [4]. Thus we can state this proposition.

**Proposition 3.2** *The hybrid automaton of an elementary service controlled with a locally optimal policy is a multirate hybrid automaton.*

Another property of a HA is that of being initialized. An automaton is said to be *initialed* if for each event step $l \xrightarrow{\sigma_i} l'$ the $j$–th component of the state vector $\mathbf{x}$ is reset whenever the $j$–th component of $act(l)$ is different from the $j$–th component of $act(l')$. Initialed multirate automata are a special class of HA that can be easily analyzed [4]. As an example it has been shown that the reachability problem is decidable for this class of HA.

Unfortunately, it can be easily seen that the HA in Fig. 2 is not initialized. In fact, let us consider the event step $ON_i \xrightarrow{f_i} BN_i$. The activity set of each state variable changes. However, while $s_i$ and $\chi_i$ may be initialized to a constant value, this is not possible for $x_i$ which may have any values in the interval $[0, C_i]$. The same reasoning applies to the event step $BN_i \xrightarrow{r_i} ON_i$.

### 3.3 Regularization

A *step* of an hybrid automaton $H$ is a transition denoted by $\sigma$ from state $(l, \mathbf{x}) \in L \times \mathbb{R}^n$ to $(l', \mathbf{x}')$

$$(l, \mathbf{x}) \xrightarrow{\sigma} (l', \mathbf{x}').$$

An *event step* $\sigma \in \Sigma$ corresponds to the occurrence of a discrete event: there is an edge $e = (l, \sigma, g, j, l') \in E$, $\mathbf{x} \in g$ and $(\mathbf{x}, \mathbf{x}') \in j$. A *time step* $\sigma \in \mathbb{R}^+$ corresponds to a continuous time evolution where $l = l'$ and $\mathbf{x}'$ is an admissible solution for the differential inclusion $act(l)$. An *execution* of length $k$ of an hybrid automaton is a sequence of steps

$$\nu = (l_0, \mathbf{x}_0) \xrightarrow{\sigma_1} (l_1, \mathbf{x}_1) \cdots (l_{k-1}, \mathbf{x}_{k-1}) \xrightarrow{\sigma_k} (l_k, \mathbf{x}_k).$$

Its duration is

$$d(\nu) = \sum_{i=1}^{k} d(\sigma_i) \quad \text{where} \quad d(\sigma_i) = \begin{cases} 0 & \text{if} \quad \sigma_i \in \Sigma \\ \sigma_i & \text{if} \quad \sigma_i \in \mathbb{R}^+ \end{cases}$$

We say that an infinite execution does not diverge if it holds

$$\lim_{k \to \infty} \sum_{i=1}^{k} d(\sigma_i) < \infty$$

Such an execution is called a *Zeno execution*. An hybrid automaton is called Zeno if it possesses Zeno executions [5]. Otherwise is called non–Zeno. Clearly Zeno HA may execute an unbounded number of transitions in a finite time interval.

The HA of an elementary service is Zeno since it can have an infinite execution consisting only of event steps. As an example consider the cycle from location $ON_i$ to $OF_i$ and back.

There exist several techniques to regularize a Zeno automaton $H$, i.e., to convert it into a non–Zeno automaton [6]. The idea is that of adding a small perturbation $\epsilon > 0$ to $H$ to make the system non–Zeno. The regularized automaton is denoted $H_\epsilon$ and we assume that $H_\epsilon \to H$ as $\epsilon \to 0$.

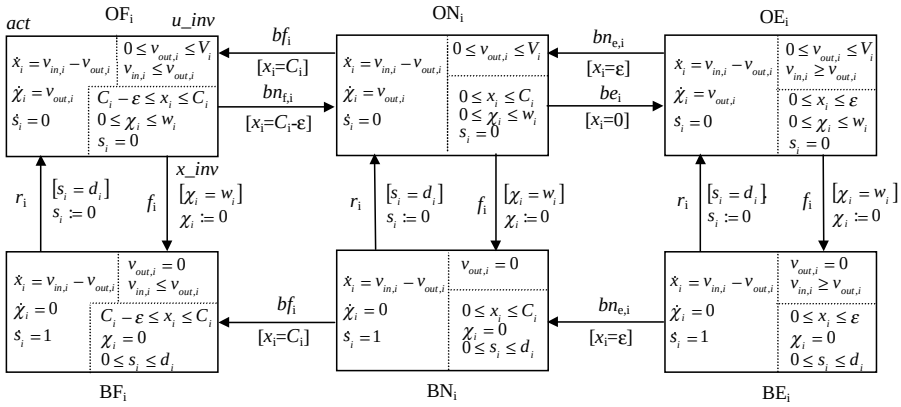


Figure 3: The regularized hybrid automaton $H_\epsilon$ of an elementary service with hysteresis.

To regularize the HA shown in Fig. 2 we add an hysteresis to the two cycles

$$ON_i \overset{bf_i}{\rightarrow} OF_i \overset{bn_i}{\rightarrow} ON_i, \qquad ON_i \overset{be_i}{\rightarrow} OE_i \overset{bn_i}{\rightarrow} ON_i$$

Thus we assume that the buffer is full when $C_i - \epsilon \leq x_i \leq C_i$ and that it is empty when $\epsilon \leq x_i \leq 0$. The resulting automaton $H_\epsilon$ is shown in Fig. 3.

# 4 Composition of hybrid automata

We now want to describe with a single HA a general manufacturing system composed of $m$ single–server stations with arbitrary routing. As it is difficult to write a hybrid automaton model of a complex system directly, a modeling strategy must be elaborated. Let us consider a modular approach, which models a complex system by sub–models. To do this we need to introduce the concurrent composition operator and to give the formal rules to compose two or more HA by concurrent composition.

## 4.1 The concurrent composition operator

As in the case of classic automata we assume that the set of locations of a concurrent composition is given by the cartesian product of the locations of all composed automata. Furthermore if a symbol belongs to the alphabet of two or more automata, it represents a synchronized event that must be executed simultaneously by all corresponding automata. However in the case of HA we must also take into account the possibility that a continuous state variable and/or an input variable may be shared among two or more automata.

To formalize this, we define the set of all continuous states and input variables of the automaton $H$ as $\mathcal{X}(H)$ and $\mathcal{U}(H)$ respectively. Note that if $H$ is obtained as the concurrent composition of two HA $H_1$ and $H_2$, we have that $\mathcal{X}(H) = \mathcal{X}(H_1) \cup \mathcal{X}(H_2)$ and if there are shared state variables $\text{card}\mathcal{X}(H) < \text{card}\mathcal{X}(H_1) + \text{card}\mathcal{X}(H_2)$ (the same holds for the input variables).

We also need the following definition [1].

[1] Here $\mathbb{R}^{\mathcal{X}}$ is the set of real vectors that have as many components as there are elements in $\mathcal{X}$.

**Definition 4.1** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets of state variables with $\mathcal{X} \supset \mathcal{Y}$. Given a set $A \subseteq \mathbb{R}^{\mathcal{X}}$ and a set $B \subseteq \mathbb{R}^{\mathcal{Y}}$ we denote the* projection *of $A$ over $\mathcal{Y}$ as the set*

$$A \uparrow_{\mathcal{Y}} = \left\{ y \in \mathbb{R}^{\mathcal{Y}} \mid (\exists z \in \mathbb{R}^{\mathcal{X}-\mathcal{Y}}) : y \times z \in A \right\}$$

*and the* counterimage *of $B$ over $\mathcal{X}$ the set*

$$B \downarrow_{\mathcal{X}} = \left\{ y \times z \in \mathbb{R}^{\mathcal{X}} \mid y \in B,\ z \in \mathbb{R}^{\mathcal{X}-\mathcal{Y}} \right\}$$

We now define the concurrent composition of hybrid automata.

**Definition 4.2** *Let $H_i = (L_i, \Sigma_i, act_i, x\_inv_i, u\_inv_i, E_i)$, for $i = 1, \ldots, m$, be HA as in Definition 3.1. Their* concurrent composition *denoted $H = H_1 \parallel \cdots \parallel H_m$ is a HA $H = (L, \Sigma, act, x\_inv, u\_inv, E)$, where:*

- *The continuous state and input variable sets are*

  $$\mathcal{X}(H) = \cup_{i=1}^{m} \mathcal{X}(H_i) \qquad \text{and} \qquad \mathcal{U}(H) = \cup_{i=1}^{m} \mathcal{U}(H_i),$$

  *respectively.*

- *The set of locations $L = L_1 \times \cdots \times L_m$ is given by the cartesian product of all $L_i$. Note that some of these locations may be unreachable in $H$ and thus may be deleted.*

- *The dynamics at location $l = l_1 \times \cdots \times l_m$ is given by*

  $$act(l) = \left(act(l_1) \downarrow_{\mathcal{X}(H)}\right) \cap \cdots \cap \left(act(l_m) \downarrow_{\mathcal{X}(H)}\right)$$

- *The set of admissible state vectors at location $l = l_1 \times \cdots \times l_m$ is given by*

  $$x\_inv(l) = \left(x\_inv(l_1) \downarrow_{\mathcal{X}(H)}\right) \cap \cdots \cap \left(x\_inv(l_m) \downarrow_{\mathcal{X}(H)}\right)$$

- *The set of admissible input vectors at location $l = l_1 \times \cdots \times l_m$ is given by*

  $$u\_inv(l) = \left(u\_inv(l_1) \downarrow_{\mathcal{U}(H)}\right) \cap \cdots \cap \left(u\_inv(l_m) \downarrow_{\mathcal{U}(H)}\right)$$

- *For a given $\sigma \in \Sigma$ we define $I_\sigma = \{i \in \{1, \ldots, m\} \mid \sigma \in \Sigma_i\}$ be the set of indices of the automata that have $\sigma$ in their alphabet, and let $\mathcal{X}_\sigma = \cup_{i \in I_\sigma} \mathcal{X}(H_i)$ be the union of the state variables of all these automata.*

  *Then there will be an edge $e = (l, \sigma, g, j, \bar{l}) \in E$ if:*

  - *$l = l_1 \times \cdots \times l_m$ and $\bar{l} = \bar{l}_1 \times \cdots \times \bar{l}_m$.*
  - *For all $i \notin I_\sigma$, $l_i = \bar{l}_i$, i.e., the event $\sigma$ does not change the state of an automaton that does not have $\sigma$ in its alphabet.*
  - *For all $i \in I_\sigma$, there exists an edge $e_i = (l_i, \sigma, g_i, j_i, \bar{l}_i) \in E_i$, i.e., the event $\sigma$ is defined at location $l_i$ for each automaton $H_i$ that has $\sigma$ in its alphabet and its occurrence changes the location to $\bar{l}_i$.*
  - *The guard $g$ must satisfy:*

    $$g = \cap_{i \in I_\sigma} \left(g_i \downarrow_{\mathcal{X}_\sigma}\right).$$

– *If we write the jump relation of each edge $e_i$ as $j_i = A_i \times B_i$ then the jump relation $j$ must satisfy*

$$j = \cap_{i \in I_\sigma} \big( A_i \downarrow_{\mathcal{X}_\sigma} \times B_i \downarrow_{\mathcal{X}_\sigma} \big).$$

This definition implies that the activity set *act* of the concurrent composition automaton $H$ is given by the intersection of the counterimages of the activity sets of all $H_i$. Note that if these sets are defined by a collection of constraints $\mathcal{C}_i$ , the corresponding set of $H$ will be defined by a set of constraints $\mathcal{C} = \cup_i \mathcal{C}_i$. The same applies also to the admissible state set $x\_inv$ and to the admissible input set $u\_inv$.

On the contrary, the guard $g$ of $H$, enabling the event step $\sigma \in \Sigma$, is given by the intersection of the counterimages of the guards $g_i$ and this intersection is taken over the automata that have $\sigma$ in their alphabet. The jump relation $j$ is given by the intersection of the cartesian products of the counterimages of the start and target sets that define the jump relation of each automaton $H_i$ and this intersection is taken over the automata that have $\sigma$ in their alphabet. Thus, when computing the guard and the jump relation of the automaton $H$ relative to a given $\sigma \in \Sigma$, only those automata that have $\sigma$ in their alphabet have to be taken into account. In fact, for all other automata, both the guard and the jump relation are not defined (they can be considered as $\mathbb{R}^{\emptyset}$), thus their counterimages over $\mathcal{X}_\sigma$ are equal to $\mathbb{R}^{\mathcal{X}_\sigma}$ and introduce no constraint on the corresponding set.

### 4.2 Composition of elementary services

As an example we consider a simple transfer line composed of two services $S_1$ and $S_2$, and a final buffer $B_3$ for the collection of finished products. The initial buffer is $B_1$ that is filled up at the production target volume. This system is shown in Fig. 4.

In this example, for sake of simplicity we have assumed that no failure may occur (i.e., in the continuous state we disregard the variables $\chi_i$ and $s_i$, $i = 1, 2$). Thus, the continuous state variable set of the two elementary services are $\mathcal{X}(H_1) = \{x_1\}$, $\mathcal{X}(H_2) = \{x_2\}$ and for the composed automaton $H = H_1 \parallel H_2$ we have $\mathcal{X}(H) = \{x_1, x_2\}$.

The input variable sets for the two ES are $\mathcal{U}(H_1) = \{v_{in,1}, v_{out,1}\}$, $\mathcal{U}(H_2) = \{v_{in,2}, v_{out,2}\}$. The interconnection between the two ES and the fact that the first one is not fed, pose the following constraints:

$$v_{in,1} = 0$$
$$v_{out,1} = v_{in,2}.$$

If we disregard $v_{in,1}$ and rename $v_1 = v_{out,1} = v_{in,2}$, $v_2 = v_{out,2}$, we have that $\mathcal{U}(H) = \{v_1, v_2\}$.

By applying the concurrent composition operator to the elementary services connected in series, we obtain the hybrid automaton $H$ represented in Fig. 5.

Obviously, the cardinality of $L$, the set of locations of $H$, is nine, being card $L_1$=card $L_2 = 3$. The initial location is $OF_1$,$OE_2$.

The sets *act*, $x_{inv}$ and $u_{inv}$ of $H$ have been obtained by combining the constraints that define the corresponding sets for $H_1$ and $H_2$. As an example, the set $u_{inv}(ON_1, OF_2)$ contains the constraint $0 \leq v_1 \leq V_1$ that defines the set $act(ON_1)$ and the constraints $0 \leq v_2 \leq V_2$, $v_1 \leq v_2$, that define the set $act(OF_2)$.

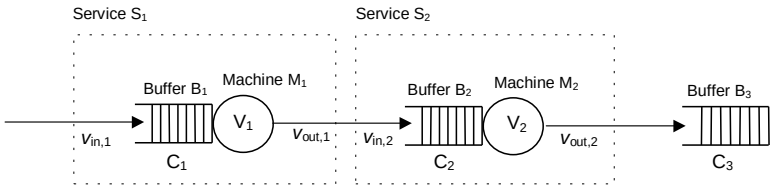


Figure 4: A 2–services transfer line.

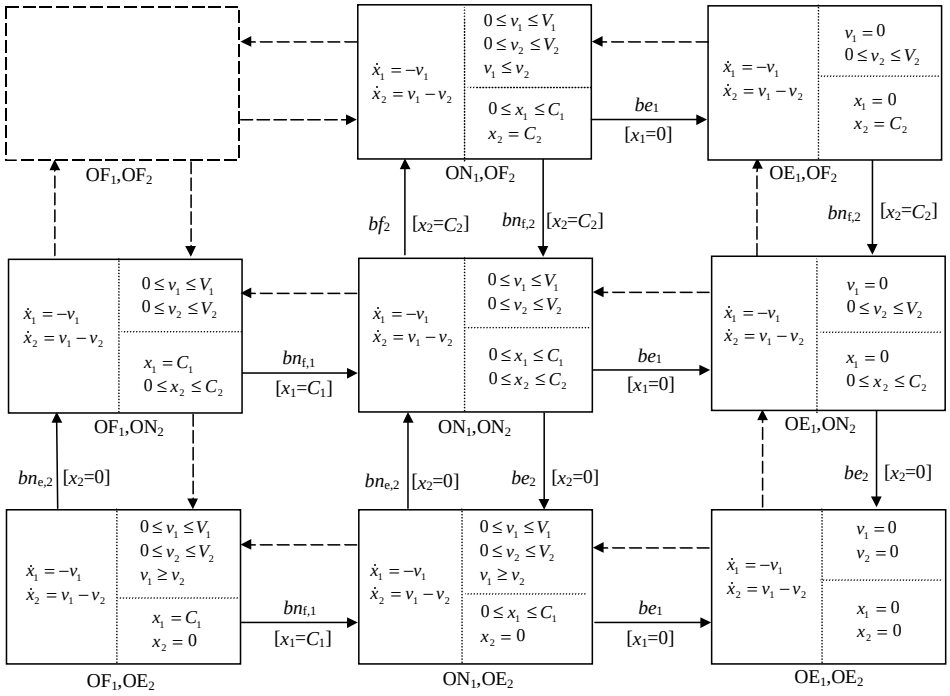


Figure 5: Composition.

In the composed automaton some edges correspond to event steps that can never be executed, regardless of the choice of the input vector. In fact, it may be possible that for all $\mathbf{x}_0 \in x_{inv}(l)$ there exists no $\mathbf{u} \in u_{inv}$ such that in a time step we reach a value of the continuous state $\mathbf{x} \in g$. As an example, because the first buffer is not fed, it is not possible to go from a location in which the buffer $B_1$ is empty ($OE_1$) to a location in which it is not empty ($ON_1$). This type of edges are shown with dashed arcs in Fig. 5.

Note also that the location $OF_1, OF_2$ can never be reached (all edges leading to it are dashed) and thus it is shown as an empty dashed box.

Now, let us examine some possible evolution of the composed system. We first observe that at the initial location $OF_1, OE_2$, the only admissible input vector is $\mathbf{u} = \mathbf{0}$, i.e., $v_1 = v_2 = 0$, and two event steps may fire: $bn_{e,2}$ and $bn_{f,1}$. If $bn_{e,2}$ fires, thus reaching the discrete location $OF_1, ON_2$, the input vector $\mathbf{u}$ is once again the null vector, thus producing no variation on the continuous state. At this point, the firing of $bn_{f,1}$ leads to $ON_1, ON_2$ and at this location both a time step and an event step ($be_2$) are possible. If a time step is executed, depending on the chosen objective function vector at location $ON_1, ON_2$ and on the value of the buffer capacities

$C_1$ and $C_2$, the next event step will be $bf_2$ (buffer $B_2$ is filled up), or $be_1$ (buffer $B_1$ becomes empty). If the event step $be_2$ is executed the system will reach the state $ON_1, OE_2$ where a time step is possible with $v_1 = v_2$. Note that a similar discussion can be repeated if $bn_{f,1}$ is the first event step to fire when the system is in its initial location $OF_1, OE_2$. Eventually, the HA will reach the home state $OE_1, OE_2$.

# 5 Conclusions

In this paper a hybrid formulation for the modelling and control of automated manufacturing systems has been developed. In particular, automated manufacturing systems with unreliable machines, buffers of finite capacity, arbitrary service time distributions and routing policies, where several parts of a single class of products are circulated and processed, have been considered. Each machine coupled with a finite buffer constitutes an elementary service (ES) and is described as a hybrid automaton. A general queueing network is obtained by combining ES in a modular fashion. The concurrent composition operator allowed us the definition of the hybrid automaton representing the whole network of services.

The evolution in time of the production process has been discussed within a framework that distinguishes two levels of aggregation. The lower layer represents the microscopic behaviour of arrivals of parts to/from each machine (micro–events). It has been modeled in an aggregate view by using first order fluid approximations. At the higher layer a discrete event model has been used to represent the transitions of the process through a sequence of macro–states, at the occurrence of the macro–events.

# References


[1] R. Alur, D. L. Dill, *A Theory of Timed Automata*, Theoretical Computer Science, 126: 183–235, 1994.

[2] F. Balduzzi, G. Menga, "A State Variable Model for the Fluid Approximation of Flexible Manufacturing Systems," *IEEE International Conference on Robotics and Automation* (Leuven, Belgium), May 1998, pp. 1172–1778.

[3] H. Chen, A. Mandelbaum, "Discrete Flow Networks: Bottleneck Analysis and Fluid Approximations," *Math. Oper. Res.*, Vol. 16, pp. 408–446, 1991.

[4] T.A. Henzinger, "The Theory of Hybrid Automata,", *Proc. 11–th Annual Symposium on Logic in Computer Science*, pp. 278–292, 1996.

[5] M. Hyemann, F. Lin, G. Meyer, "Viability of Controllers for Hybrid Machines,", *Proc. 36–th Conf. Decision and Control* (San Diego, California), Dec 1997, pp. 714–719.

[6] K.H. Johansson, M. Egerstedt, J. Lygeros, S. Sastry, "Regularization of Zeno Hybrid Automata,", *Systems Control Letters, special issue on Hybrid Systems*, submitted, 1999.

[7] A. Puri, P. Varaiya, *Decidable Hybrid Systems*, *Computer and Mathematical Modeling*, 23(11/12): pp. 191–202, 1996.