Discrete-Event State Equations and Petri Nets

Enrico Canuto[†], Fabio Balduzzi[†]
Dipartimento di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

In this paper we present a novel formulation for the modeling and control of discrete event dynamic systems. This original approach leads to a discrete-event state equations formulation satisfying Kalman axioms, where the state is defined as the sequence of potential events (enabled transitions in terms of Petri net language) forced by the occurrence of state events (free evolution) or by arbitrary input events (forced evolution).

The proposed formulation is considered to be very general and appropriate to any discrete event systems. This conviction is supported by the analysis performed by comparing discrete-event state equations with classical discrete-event models like untimed and timed Petri nets, finite-state timed and untimed automata. We show that all these models can be formulated as a sub-class of the discrete-event state equations.

1 Introduction

In this paper a novel theory of discrete-event dynamic systems (DEDS) is compared with classical approaches such as Petri nets (PN) and finite-state automata. This novel approach has been recently developed within the ESPRIT Basic Research HIMAC, dedicated to a new mathematical framework, the Manufacturing Algebra, for the modeling and control of discrete manufacturing systems (Canuto, 1998a; Canuto, 1998b; Donati *et al.*, 1996; Vallauri, 1997). After a short review of the previous work about DEDS, we present a synthesis of the novel theory which transposes the dynamic system concept, originally developed by Kalman (Kalman *et al.*, 1969) in terms of input, state and output variables to the field of discrete-event dynamics.

A DEDS is formulated as an operator which transforms input event sequences into output event sequences satisfying the causality constraint. Causality implies that a state variable $x(t)$ exists at any time t and can be described as a (*potential*) event occurring at a time later than t. The evolution in time of the potential event described by $x(t)$ does not depend explicitly on time t but only on the occurrence of other events, like the potential event itself or input events.

To provide a simple example, let us consider an alarm-clock endowed with keys for registering an 'alarm' =[*alarm type* (ring, music, ...), *alarm time*] and a function which automatically sets a new 'alarm time' only when an alarm rings. The asynchronous and unpredictable sequence of registrations made by the user is the *input event sequence*. The sequence of alarms effectively ringing is the *output event sequence*. Any event is described as a pair [*fact, occurrence time*]. Output events can also be defined as pairs [alarm type, alarm time]. Input events can also be defined as pairs [alarm, registration time]. The clock state is defined at any real time t and corresponds to the last 'alarm' registered in the

---

[†] Email: {*enrico.canuto,balduzzi*}*@polito.it*

clock memory. Therefore the clock state is a potential event which will occur and modify itself only if no external registration occurs before its ‘alarm time’.

The alarm-clock is a simple but complete example of DEDS types treated in this paper that have the following properties:

1. the time variables of the input events, like the alarm registration times, or of the output events, like times when alarms ring, are countable but with real and unpredictable values. Therefore the input-output operator cannot be described by discrete-time state equations;
2. the state is a variable defined at any real time t and contains the last registered ‘alarm’ (a potential event) which may or may not occur;
3. the state can be only modified at the occurrence of input events (forced evolution) or of the potential event itself (free evolution), i.e., when a registered ‘alarm’ occurs;
4. input and output time functions are discrete-event sequences belonging to well-defined sequence spaces.

In this paper we show that such DEDS formulation is very general and can be realized with the block-diagram depicted in Figure 1, where:

- an asynchronous delay makes the potential event registered as $x(t)$ to occur at a later time, if not modified meanwhile;
- a feedback static function modifies the state $x(t)$ when the potential event occurs;
- an input static function, possibly depending on the state, forces the input events to modify the state;
- an output static function makes output events to occur when a potential event occurs.

Each block will be formulated in terms of event functions and the ensemble will define a discrete-event state equation (DESE) which satisfies Kalman axioms and therefore is appropriate for applying state-space control theory.

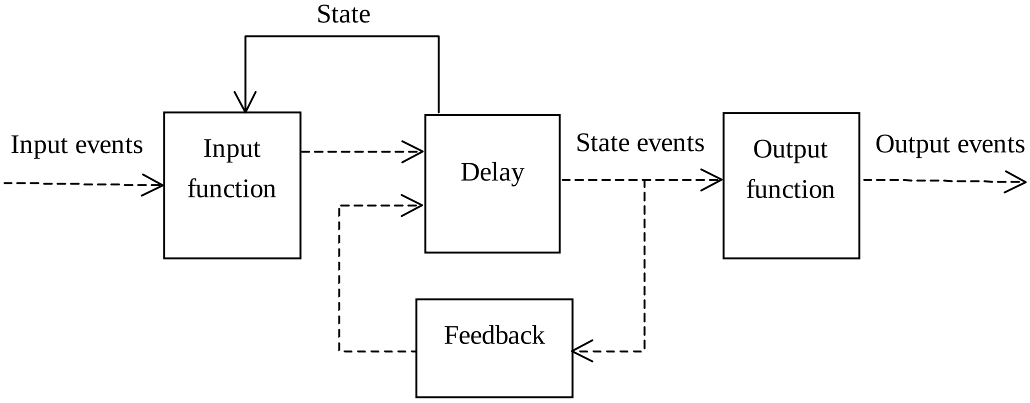

Figure 1: Block-diagram of a discrete-event state equation

The proposed formulation is considered to be very general and appropriate to any DEDS. This conviction is supported by the analysis performed by comparing DESE with classical discrete-event models like untimed and timed Petri nets, finite-state timed and untimed automata.

It will be shown that all these models can be formulated as a sub-class of the discrete-event state equations. Specifically finite-state automata and untimed Petri Nets correspond to forced state evolution, timed automata and timed Petri nets correspond to free state evolution. The analysis developed in this paper will be limited to deterministic DEDS, therefore we shall not treat stochastic models like Markov chains or Generalized Semi-Markov Processes (GSMP).

1.1 Previous Work

A substantial body of literature about DEDS is concerned with the problem of how to formulate the dynamics of a discrete-event system. Even so it seems to the authors that there is not a universally accepted formulation like there is for continuous (discrete) variable dynamic systems.

The most important formulations of DEDS are provided by state automata and Petri nets, either timed or untimed. For these models, however, the general concepts of event and state do not correspond to the definition we will provide in the following sections. Thus in this section we will refer to events as the facts introduced in our formulation of DEDS. Petri net models have been widely considered for describing DEDS, see for instance (Murata, 1989) and the references therein. Other popular models include the automata model of Ramadge and Wonham (Ramadge, Wonham, 1989) and the path-based models used for perturbation analysis (Cassandras, 1993). Other formalisms such as the GSMP formulation (Glynn, 1989) and the model described in (Ziegler, 1989) deal also with simulation issues for DEDS. However analytic results for the more general settings discussed here are available only for simple systems, while the analysis of more complex involves the use of approximate decomposition techniques and Markov models as suggested by Gershwin (Gershwin, 1994).

A novel modeling approach is proposed in (Passino, Burgess, 1998) where a non-linear difference equation is formulated for modeling non-deterministic behavior, in the sense that if the system is at any state, then there is a set of events that can occur and depending on which one occurs the system can evolve through transitions to different states. In a recent work Balduzzi and Menga (Balduzzi, Menga, 1998) developed a discrete-time, time-varying linear stochastic state variable model for the fluid approximation of flexible manufacturing systems. Then, by using perturbation analysis techniques they obtained average values and variances of both performance measures and of their gradients with respect to the system parameters to perform optimal design of the system configuration.

All these formulations, however, do not make a clear distinction among free and forced responses as well as input, output and state variables. Our main interests are towards control theory for DEDS, hence the general framework we provide attempts to properly define all those system concepts by mean of discrete-event state equations which satisfy Kalman axioms and therefore is appropriate for applying state-space control theory. The theory developed in this paper is based on the works produced by Donati *et al.* (Donati *et al.*, 1996; Vallauri, 1997) and the results we will show are obtained starting from the basic ideas presented in (Canuto, 1998a; Canuto, 1998b; Donati *et al.*, 1996; Vallauri, 1997).

2 Discrete-Event State Equations

In this section we introduce the theory of discrete-event state equations for the modeling and control of DEDS. For continuous(discrete)-time dynamic systems (strictly causal), the most general formulation is to assume that the state function $\mathbf{x}=\{\mathbf{x}(t),\ t\geq 0\}$ and the input-output dynamic relation satisfy relations of the form:

$$\begin{aligned} &\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ &\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), t) \end{aligned} \tag{1}$$

for some prescribed family of mappings $\{\mathbf{f}(t,\cdot,\cdot),\ \mathbf{g}(t,\cdot),\ t\geq 0\}$, initial condition $\mathbf{x}_0$ and input function $\mathbf{u}=\{\mathbf{u}(t),\ t\geq 0\}$.

Intuitively we may state an analogous formulation for DEDS, that is we require the existence of a dynamic function $\mathbf{x}(t)$ to describe the evolution in time of the state of the system, and which satisfies Kalman axioms (Kalman *et al.*, 1969). Such a formulation can be obtained provided that an appropriate choice of the state space is made. As it will be clear in what follows, we will define input, output and state variables in a suitable space of event sequences.

2.1 Events and their Functions

Let t be the real time variable and let $T\in\Re$ be any real time interval $T=[t_1,t_2)$ finite or infinite, called *time set*. Let Ξ be a countable set of elements ξ, called *facts*, including the null element 0. The fact set may be:

- a finite set with appropriate algebraic properties like a set of symbols for a language. For instance, the list of the different parts that a workstation can process, the list of the possible functional status of a workstation (*working, idle, broken*);
- a set endowed with algebraic operations like addition and scalar multiplication. For instance, integer vectors defining for each part of a list the quantity stored in a buffer.

Definition 2.1.1 An event e is defined as a pair [*fact, time*], i.e., $e=[\xi,t]$ where $\xi\in\Xi$ and $t\in T$. ■

The time t is called the *occurrence time* of the event e. The set of events is denoted $E=\Xi\times T$. An event is said to occur when its occurrence time t equals the monotone increasing clock function $c(\tau)$, $\tau\in\Re$. Special events are the *null event* $e_0=[0,t]$, the *impossible event* $e_{0\infty}=[0,\infty]$ and the *pending event* $e_\infty=[\xi,\infty]$, i.e., an event missing the occurrence time.

In our formulation, differently from the literature, an event is always tagged by a real time instant and called simply event (not timed event). In order to properly define the time evolution of a DEDS, we need to specify a mechanism for generating the event sequences. Thus we assume that events can be made function of the current time t as long as their occurrence times are greater than t. Such events are called *potential events* and play an essential role in our developments.

Definition 2.1.2 A *potential event* is an event $e_x(t)=[\eta_x,\tau_x>t]$ whose occurrence time satisfies the constraint $\tau_x\geq t$. ■

Events may be transformed into other events by appropriate event functions. We distinguish among static and dynamic functions. For example, the input, output and feedback blocks in Figure 1 are static functions while the delay block is a dynamic function.

Definition 2.1.3 Given two event sets, the input set $E_u=U\times T_u$ and the output set $E_y=Y\times T_y$, the *static event function* $F_e:E_u\to E_y$, mapping an input event $e_u=[\xi_u,t_u]$ into an output event $e_y=[\xi_y,t_y]$, is defined by the pair of relations: $\xi_y=f_e(e_u)$ and $t_y=t_u$, with the constraint that a null event is mapped into a null event, $0=f_e(e_0)$. ■

In the alarm-clock model, when an alarm rings, the clock will register a new alarm. The input and output events are $e_u=[\xi_u=\text{‘alarm type’},t_u=\text{‘alarm time’}]$ and $e_y=[\xi_y=\text{‘alarm’},t_y=\text{‘registration time’}]$ with $t_y=t_u$ and, for instance, $\xi_y=f_e(e_u)=[\xi_u=\text{‘same alarm type’},t_u+\lambda=\text{‘new alarm time’}]$.

Definition 2.1.4 Given two event sets, the input set $E_u=U\times T_u$ and the output set $E_y=Y\times T_y$, the *dynamic event function* $\Gamma_e:E_u\to E_y$, mapping an input event $e_u=[\xi_u,t_u]$ into an output event $e_y=[\xi_y,t_y]$, is defined by the pair of relations $\xi_y=\gamma_e(e_u)$ and $t_y=\psi_e(e_u)>t_u$. The output event will occur delayed with respect to the input event. A null event is mapped into a null event, $0=\gamma_e(e_0)$. ■

When an alarm has been registered and no further registrations are made, alarm will ring at a later time expressed by the registration. The fact of an event might be an event itself. For instance when a potential event $e_x(t)$ occurs at the current time $t=\tau_x(t)$, it can be useful to generate a new potential event $e_x'(t)$, which is embedded as the fact of an event called the *next potential event* and obtained as the output of a static function N_e : $[e_x'(t),\tau_x(t)]=N_e[e_x(t)]$.

2.2 Event Sequences and Operations

Event sequences in our framework correspond to time functions for classical state equations and allow us to describe the evolution of input, output and state variables.

Definition 2.2.1 An *event sequence* σ is a countable set of events $\{e(i)=[\xi(i),t(i)]\in E,\ i\in K(\sigma)\}$ that does not contain simultaneous events, i.e., the sequence σ is subject to the constraint $t(i)\neq t(j)$ for any pair of counters $i,j\in K(\sigma)=\{1,2,\ldots,i,\ldots\}$, where $K(\sigma)$ is the counter set of the sequence. ■

Note that the absence of simultaneous events implies that the events $e(i)\in\sigma$ are strictly ordered by their occurrence times. The set of all event sequences defined by the event set E is denoted $\Sigma(E)$. A sequence will be denoted either by the collective symbol σ or by the generic event $\varepsilon(i)$.

A *null sequence* is the event sequence made of null events. A *normal sequence* is an event sequence without null events. Normal sequences are equal when they include the same events. Note that, as already stated, null events cannot map into non null events. Several operations can be defined over $\Sigma(E)$: addition and scalar multiplication, which depends on the corresponding operation defined for the fact set, time-shift and time restriction.

Definition 2.2.2 *Addition*. Given a pair of event sequences $\sigma_1,\sigma_2\in\Sigma(E)$ with generic events $e_1(i)=[\xi_1(i),t_1(i)]$ and $e_2(j)=[\xi_2(j),t_2(j)]$, the event sequence (sum) $\sigma=\sigma_1+\sigma_2=\{e(k)=[\xi(k),t(k)]\}$ is defined as (1) the union of non simultaneous events, $t_1(i)\neq t_2(j)$, (2) the addition of the facts of two simultaneous events, being admissible only when addition is defined for the fact set. Then $e(k)=[\xi_1(i)+\xi_2(j),t_1(i)=t_2(j)]$. ■

Definition 2.2.3 *Restriction*. Given an event sequence $\sigma\in\Sigma(E)$ and a time interval $[t_1,t_2)$, the restricted sequence σ' is defined by the set of events restricted to the time set $T\cap[t_1,t_2)$. An event sequence σ restricted to the future, i.e., to (t,∞), where t denotes the current time, is called *causal sequence* and indicated by $\sigma(t)$. In practice, it is the *list* of the events not yet occurred. To avoid a causal sequence to become empty as far as $t\to\infty$, we add to the sequence the impossible event $e_{0\infty}$. ■

Definition 2.2.4 *Time-shift*. Given an event sequence $\sigma\in\Sigma(E)$ and a time τ, the shifted sequence σ' is obtained by shifting the time set T and the occurrence times. A causal sequence $\sigma(0)$ shifted by τ is denoted by $\sigma(0,\tau)$. ■

Remark 2.2.5 The simplest way to build a potential event $e_x(t)$ is to restrict a causal sequence $\sigma(0)=\{e(i)=[\xi(i),t(i)]\}$ to the current time $t>0$. The fact is the restricted causal sequence itself, $\eta_x=\sigma(t)$ and the occurrence time is the next one, $\tau_x=\min_i\{t(i)>t\}$.

2.3 Event Sequences and Time Functions

We have already stated that the state of a DEDS is a time function $x(t)$, that is modified only at the occurrence of appropriate events, and the input and output variables are event sequences. The only constraint we impose on the input event sequences is the concatenation axiom, meaning that there are no links between the future and the past events. Therefore, suitable operators have to be defined, mapping an input event sequence σ_u into an output time function $x(t)$.

Concatenation axiom. Let $\Sigma(E_u)$ be the set of all input event sequence σ_u. This set must be closed under *concatenation* of arbitrary pairs of input event sequences σ_{u1} and σ_{u2} restricted over two non overlapping but contiguous time sets $T_1=(t_0,t_1]$ and $T_2=(t_1,t_2]$.

Note that the concatenation axiom, which is defined as $\sigma_{u1}(t_0,t_1]\cup\sigma_{u2}(t_1,t_2]$, is the basic property of input functions in the Kalman formulation (Kalman *et al.*, 1969). The most important operators that map an input event sequence σ_u into an output time function $x(t)$ are the *event register R* and the *event adder* Σ.

Definition 2.3.1 *Event register* $R: E_u=U\times T_u\rightarrow U$, where U is the set of input facts. Given an input event sequence $\sigma_u=\{e_u(j)=[\xi_u(j),t_u(j)]\}$, the event register R is defined by the equation $x(t^+)=R[e_u(j)]=\xi_u(j)$, where $t=t_u(j)$ and t^+ denotes the time interval $t_u(j)<\tau\leq t_u(j+1)$. ■

The output of an event register is a step-like time function $x(t)\in U$, whose value is equal to the fact of the last occurred event. Therefore $x(t)$ is a state function and the register equation plays the role of a state equation with initial condition $x(0)$. As it will be shown later, event registers allow us to formulate in our theory finite state automata. When the fact set of input events is additive, input facts can be registered into a time function $x(t)$ in an additive way.

Definition 2.3.2 *Event adder* Σ: $E_u\times X\rightarrow X$, where X is the set of state facts. Given an input event sequence $\sigma_u=\{e_u(j)=[\xi_u(j),t_u(j)]\}$, with input fact set U endowed with addition operations, the event adder is defined by the equation $x(t^+)=\Sigma[e_u(j)]=x(t_u(j))+\xi_u(j)$, that holds in the time interval $t_u(j)<\tau\leq t_u(j+1)$. ■

Figure 2 shows the graphical symbols of event registers and event adders. Event sequences are indicated with dashed arrows while time functions with solid arrows.

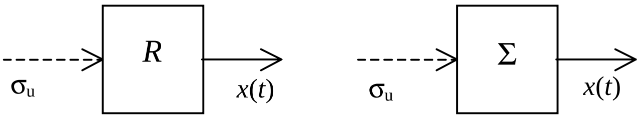

Figure 2: Symbols of event register and adder

2.4 Transitions and Delay Functions

The composition of an event register R with a static event function B_e, can be used to define transition functions. For the sake of simplicity, in this presentation we will only consider time-invariant transition functions.

Definition 2.4.1 *Transition functions*. Given a time function $x(t)\in X$ and a finite set of input event sequences $e_l(j_l)=[\xi_l(j_l),t_l(j_l)]\in E_l$, $l=1,2,\ldots,L$, a vector of transition functions $\Phi_l:E_l\times X\rightarrow X$, is defined by the equations $x(t^+)=\Phi_l[e_l(j_l),x(t)]$, where $t=t_l(j)$ and t^+ corresponds to the interval of time $t_l(j_l)<t<\min_l\{t_l(k)>t_l(j_l)\}$. ■

Transition functions may not depend explicitly on $x(t)$, which means that $x(t^+)=\Phi_l[e_l(j_l)]$. Moreover we can show that transition functions satisfy Kalman axioms and consequently $x(t)$ is a state function.

Proposition 2.4.2 The transition functions Φ_l satisfy the following axioms of strictly causal dynamic systems:

1. *Consistency*. The time function $x(t)$ only changes after an event occurrence.
2. *Causality*. Given any initial value $x(t_0)$ and applying two event sequences σ and σ' restricted to $T_0=[t_0,t_1)$, it yields $x(t_1)$ if it holds $\sigma(T_0)=\sigma'(T_0)$.
3. *Composition*. Given any initial value $x(t_0)$ and a sequence σ, consider the restrictions σ_1 and σ_2 defined for the consecutive intervals $[t_0,t_1)$ and $[t_1,t_2)$ respectively and the concatenation $\sigma_{12}=\sigma_1\cup\sigma_2$. Then sequentially applying σ_1 and σ_2 or applying their concatenation σ_{12} it yields the same final state value $x(t_2)$. ■

A transition function receives a sequence of events and registers their facts, possibly altered, in a time function. If the facts to be registered are potential events, like the ‘alarms’ set in an alarm clock, then we need a dynamic function making such events to occur. We now introduce the event delay operator.

Definition 2.4.3 *Event delay* $\Delta_e: E_u \rightarrow E_x \times X$, where $E_u = E_x \times T$ and $X = E_x$. Given an input event sequence $\sigma_u = \{e_u(h) = [e_x(h), t_u(h)] \in E_u\}$, such that $e_x(h,t) = [\eta_x(h), \tau_x(h) > t]$ is a potential event at time t, a state function $x(t): T \rightarrow X$ and an output event sequence $\sigma_y = \{e_x(k) \in E_x\}$ can be obtained as follows:

1. *Potential event registration*. The potential event $e_x(h)$ embedded as the fact in the input event $e_u(h)$ is registered in the state variable. Formally $x(t^+) = e_x(h)$ where $t = t_u(h)$ and t^+ means $t_u(h) < t < min\{t_u(h+1), \tau_x(h)\}$.
2. *Potential event occurrence*. When a potential event $e_x(k)$ occurs, first the state content is cancelled, formally $x(\tau^+) = e_{0\infty}$ where $\tau = \tau_x(k)$ and τ^+ means $\tau_x(k) < \tau < min_h\{t_u(h) > \tau_x(k)\}$; second an output event will occur matching the potential event, formally $e_x(k) = x(t)$ if and only if $\tau_x(k) = t$, where $t = c(\tau)$, the delay clock.
3. *Initial condition*. The initial condition is given by $x(0) = e_x(0)$. ■

The event delay operator plays a key role in our development, thus it deserves a complete investigation. First we observe that the input and output counters h and k are generally different, since the facts $e_x(h)$ of some input events may not occur, being cancelled by subsequent events. This feature is essential for closed-loop control design, since it allows to correct potential events forced either by previous decisions or by external disturbances. Second, two consecutive output events $e_x(k)$ and $e_x(k+1)$ define a set $H(k) = \{h_k+1, \ldots, h_{k+1}-1\}$ of input counters corresponding to input events that have never occurred (*dreams*). The set $H(k)$ is called the *hidden set* and becomes empty when $h_{k+1} = h_k+1$. The counters h_k corresponding to the occurring events are called *outcome counters*. Finally, the facts $e_x(h)$ of the input events which may or may not occur will be indicated hereafter as $e_{xd}(h)$. The outcomes will be denoted by $e_x(k)$. If we only consider the input and output sequences of potential events effectively occurring, we can write the delay equation as $e_x(k) = \Delta_e([e_{xd}(h_k), t_u(h_k)])$.

In Figure 3 we have shown the symbols of transition functions and event delays.

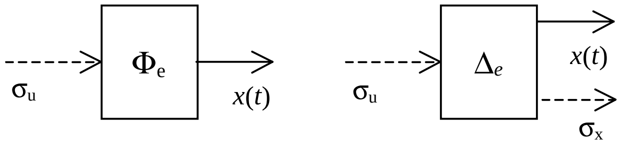

Figure 3: Symbols of transition functions and event delays

3 Discrete-Event State Equations

Let T be the usual positive time set $\{t \geq 0\}$ with t the current time. We now introduce the input and output event sequences and the state function.

- *Input event sequence*. Denote with σ_u a sequence of input events $e_u(j) = [u(j), t_u(j)] \in E_u = U \times T$, with $\Omega(\Sigma_u)$ the subset of the admissible input sequences in the set of the input sequences $\Sigma_u = \Sigma(E_u)$. We assume that $\Omega(\Sigma_u)$ is closed under concatenation.
- *Output event sequence*. Denote with σ_y a sequence of output events $e_y(k) = [y(k), t_y(k)] \in E_y = Y \times T$, with $\Sigma_y = \Sigma(E_y)$ the set of the output event sequences.

Note that output and input events do not need to be synchronous, and in general they have different occurrence times and different counter sets.

- *State event and state function*. The set of state events $E_x = X \times T$ is the set of potential events $e_x(t) = [\eta_x, \tau_x > t]$, whose fact is a causal sequence $\eta_x \in X$. The state function $x(t) \in E_x$ corresponds to the potential event $e_x(t)$ at the current time t.

Definition 3.1 A DEDS is a dynamic operator mapping input event sequences $\sigma_u \in \Omega(\Sigma_u)$ into output event sequences $\sigma_y \in \Sigma_y$. It is defined by a pair of state transition functions Φ_0 and Φ_f and an output static function C_e such that:

1. *Free evolution*: $x(\tau^+)=\Phi_0[e_x(k),\tau]$ where $\tau=t_x(k)$ is the occurrence time of the k-th state event $e_x(k)=x(\tau)$.
2. *Forced evolution*: $x(t^+)=\Phi_f[e_u(j),x(t)]$ where $t=t_u(j)$ is the occurrence time of the j-th input event.
3. *Output equation*: $e_y(k)=C_e[e_x(k),t_x(k)]$.
4. *Initial condition*: $x(0)=[\eta_{x0},\tau_{x0}<0]$.
5. *Time interval between transitions*: τ^+ means $t_x(k)<\tau<\min_j\{t_x(k+1),t_u(j)>t_x(k)\}$ and t^+ means $t_u(j)<t<\min_k\{t_u(j+1),t_x(k)>t_u(j)\}$. ■

We can now state the following proposition.

Proposition 3.2 The equations given by *Definition 3.1* satisfy Kalman axioms and consequently the time function $x(t)$ is a state function. ■

The state variable of the state equations is the time function $x(t)$. By means of event delays it is possible to write down state equations in terms of the potential events $e_x(k)$ registered in $x(t)$.

Proposition 3.3 The state equations given by *Definition 3.1* can be rewritten using a single event delay. ■

We write the state equations with a single event delay by introducing the sequence $\sigma_{xd}=\{e_{xd}(h)\}$ of the potential events registered in the state function $x(t)$ and the sequence of the next potential events $[e_{xd}(h),t_{ud}(h)]$ forcing the event delay. The state equations can be written as follows:

- *Free evolution*: $[e_{xd}(h_k+1),t_x(k)]=F_e(e_x(k))$, i.e., the next potential event is generated at the occurrence of a state event.
- *Forced evolution*: $[e_{xd}(h+1),t_u(j)]=B_e(e_u(j),x(t_u(j)))$, i.e., the next potential event is generated at the occurrence of an input event.
- *State event occurrence*: $e_x(k+1)=\Delta_e([e_{xd}(h_{k+1}),t_{ud}(h_{k+1})])$.
- *State function*: $x(t^+)=R([e_{xd}(h),t])$, where $t=t_{ud}(h)$ is equal to either $t_u(j)$ or $t_x(k)$.
- *Output equation*: $e_y(k)=C_e[e_x(k),t_x(k)]$.

The new set of equations can be represented with a block-diagram as shown in Figure 4.

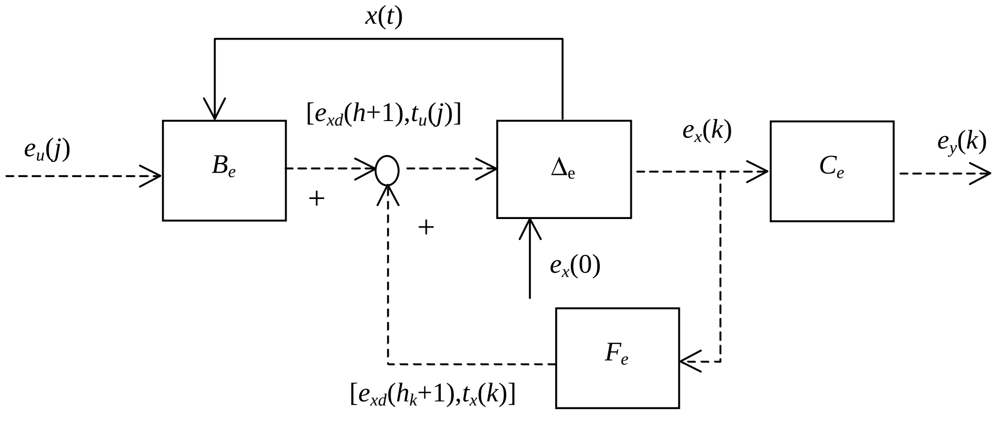

Figure 4: Block diagram of DESE

Linear DESE have been already developed and stability analysis has been performed (Vallauri, 1997) even if not reported in this work.

4 Automata and Untimed Petri Nets

In the remainder of this paper we shall formulate in terms of our theory classical DEDS models like automata and Petri nets. To this end the analysis will first concern with untimed (or logical) models and then with timed models. We shall limit ourselves to the deterministic case. The main results we have obtained can be summerized as follows.

First, untimed automata and Petri nets cannot describe a generic DEDS mapping input event sequences into output event sequences, because their input sequences (events in the case of automata and transitions in the case of Petri nets) do not satisfy the concatenation axiom, being constrained by the state itself (*enabling* or *feasible conditions*) in an ambiguous way (for each marking several transitions can be enabled). In other words, given an initial state (an initial marking for PN) the state evolution is not uniquely specified, being dependent on the input sequence. Furthermore, given an arbitrary input sequence the corresponding forced evolution may not exist, because the sequence does not respect the enabling conditions.

Second, this peculiarity leaves designers to specify an external mechanism of their own in order to solve such an ambiguity, which is often appropriate to model conflicts. If the (internal) input sequence is made dependent on an external one satisfying concatenation, i.e., external disturbance or reference, the state of the untimed DEDS will be only subject to forced evolution. If the internal input sequence is made dependent on the state through a not ambiguous relation, i.e., feedback control policy, the state will be subject to free evolution. Of course mixed cases are possible.

A typical mechanism to solve input ambiguity is to make enabled transitions (or feasible events) to occur (fire) only after a deterministic or random delay. The corresponding models are called timed and we shall show they possess only a free state evolution.

4.1 Finite-State Automata

Consider a finite-state automata defined by the set $\{U, X, Y, \Gamma, \beta, \gamma, x_0\}$, where U, X and Y are the input, state and output sets, respectively, $\Gamma(x)$ denotes a feasible set defining the enabled input elements $u \in U$ for each state element $x \in X$, $\beta{:}X \times U \rightarrow X$ denotes the transition function, $\gamma{:}X \times U \rightarrow Y$ is the output function and x_0 is the initial state.

Theorem 4.1.1 Any finite-state automata can be formulated as a dynamic system mapping an input event sequence $\sigma_u=\{e_u(k)=[u(k),t_u(k)]\}$ belonging to an admissible set Ω_u into a state function $x(t)$ and an output event sequence $\sigma_y=\{e_y(k)=[y(k),t_y(k)]\}$. Input and output sequences are synchronous, i.e., they have the same counter k and the same set $\{t_y(k)=t_u(k)\}$ of occurrence times.

Proof.

(1) The set X is the set of values that the state function $x(t)$ may assume at time t. The state function can vary only at a countable sequence of time instants belonging to the time set $T=[0,\infty)$. It holds $x(0)=x_0$.

(2) The set U is the fact set of the input event sequences $\sigma_u \in \Sigma(E_u)$ defined over the event set $E_u=U \times T$. Input event sequences must belong to the admissible subset $\Omega_u \subset \Sigma(E_u)$ defined by the feasible set $\Gamma(x)$. Such input sequences and their events will be called *enabled*.

(3) The state function $x(t)$ is obtained from the enabled input events $\{e_u(k)\}$ through a static input function B_e expressing the transition function β, and a register R. The function B_e forces the sequence of the intermediate events $e_x(k)=[x(k),t]=B_e(e_u(k),x(t))$ where $t=t_u(k)$, $x(t)=x(k\text{-}1)$ and $x(k)=\beta(x(k\text{-}1), u(k))$. The last equation is the usual automaton state equation. The register function R saves the fact $x(k)$ of the event $e_x(k)$ within $x(t^+)$.

(4) The set Y is the fact set of the output event sequence $\sigma_y=[e_y(k)=[y(k),t_y(k)]\}$, defined over the set $E_y=Y\times T$. The output sequence is generated by the sequence $\sigma_x=\{e_x(k)\}$ through a static function C_e, defined by the equation $y(k)=\gamma'(x(k))$. By replacing the expression of $x(k)$, one can obtain the automaton output equation $y(k)=\gamma'(\beta(x(k\text{-}1), u(k)))=\gamma(x(k\text{-}1), u(k))$.

This completes the proof. ■

The block diagram of an automaton is shown in Figure 5. The dependence on the state $x(t)$ for the admissible set Ω_u of input sequences is sketched with a cloud related to $x(t)$.

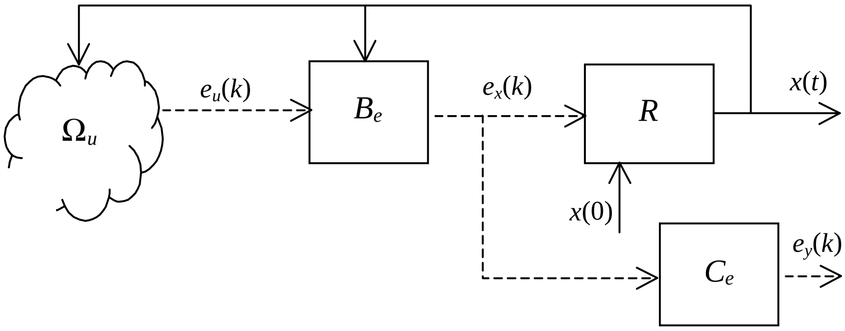

Figure 5: Block diagram of a finite-state automata

It can be shown that the admissible input set Ω_u is not closed under concatenation, and therefore automata do not satisfy all Kalman axioms. This means that the state evolution cannot be expressed in terms of forced and free evolution, since they cannot be distinguished among each others.

Lemma 4.1.2 The admissible set $\Omega_u\subset\Sigma(E_u)$ of input sequences is not closed under concatenation.

Proof. Consider two enabled input sequences σ_{u1} and σ_{u2} leading to different state values x_1 and x_2 at time t and such that they have disjoint feasible sets $\Gamma(x_i)$, $i=1,2$. Consider the concatenation $\sigma_{u1}[0,t)\cup\sigma_{u2}[t,\infty)$. By construction it does not belong to $\Omega_u\subset\Sigma(E_u)$. ■

We observe that the simplest mechanism to distinguish between forced and free state evolution is to select among the enabled input events by means of an input sequence σ_u' belonging to a sequence set Σ_u' closed under concatenation. The occurrence at time $t_u'(j)$ of the new input events $e_u'(j)$ may or may not make to occur one of the enabled events through some static function B_u. In this case the input sequence may have more event occurrences than the output sequence, meaning that counters j and k do not coincide; the state evolution is defined by a forced evolution, since $x(t)$ may only change when an input event occurs. Moreover whenever only enabled input events are considered, all event sequences are synchronous and therefore occurrence times $t_y(k)$ are not essential to describe state evolution.

The simplest automaton is a read/write memory. Input events are read and write events. The state function is the memory content. A write event modifies the state. A read event does not modify the state, but creates an output event. Enabling can be included by constraining memory writing to memory content.

4.2 Untimed Petri Nets

Consider an untimed Petri nets defined by the set $\{P,\Lambda,Pre,Post,\mathbf{m}_0\}$ where P is the finite set of n places p, Λ is the finite set of m transitions λ, $Pre:P\times\Lambda\rightarrow\mathbb{N}^n$ and $Post:P\times\Lambda\rightarrow\mathbb{N}^n$ are the *pre-* and *post-incidence functions* that specify the arcs. We denote the preset (postset) of transition λ as $^\bullet\lambda$ ($\lambda^\bullet$). Similar notation may be used for presets and postsets of places. The *incidence matrix* of the net is defined as $\mathbf{C}(p,\lambda)=Post(p,\lambda)-Pre(p,\lambda)$. A marking $\mathbf{m}:P\rightarrow\mathbb{N}^n$ is a function that assigns to each place a non-negative number of tokens; m_p denotes the marking of place p. The value of a marking at time t is denoted $\mathbf{m}(t)$ and the initial marking is $\mathbf{m}_0$. A transition λ is *enabled* at $\mathbf{m}$ if for all $p\in{}^\bullet\lambda$, $m_p\geq Pre(p,\lambda)$. An enabled transition λ fires yielding the new marking $\mathbf{m}'=\mathbf{m}+\mathbf{C}(\cdot,\lambda)$.

Untimed Petri nets can be considered as a generalization of finite-state automata in the sense that the state function $x(t)$ is multivariable and corresponds to the marking $\mathbf{m}(t)$, and the state function can be modified in an additive way. As a consequence enabling conditions may depend on several places (state components) and when a transition fires several places might change their content.

In our formulation the set of input facts U correspond to the transition set Λ and for each marking $\mathbf{m}$ there is a set $\Gamma(\mathbf{m})$ of enabled transitions defining the admissible set Ω_u of input event sequences. When an enabled input event $e_x(k)$ occurs (a transition λ fires) a new event $e_m(k)=[\Delta\mathbf{m}(k),t_m(k)]$ is generated through a static function B_m. The fact $\Delta\mathbf{m}(k)$ of such an event is the marking variation $\Delta\mathbf{m}(k)=\boldsymbol{C}(.,\lambda)$ forced by the firing transition. The marking variation event modifies the marking of the net by an event adder. Therefore the following proposition can be proved.

Proposition 4.2.1 Any untimed Petri net with an additive marking can be formulated as a dynamic system mapping an input event sequence σ_u belonging to an admissible set Ω_u, into a state function $\mathbf{m}(t)$ and an output event sequence σ_y. The dynamic mapping can be realized as a cascade of two static functions B_e and B_m and an event adder. The admissible input set is not closed under concatenation. ■

In Figure 6 we have depicted the block-diagram of a untimed Petri net represented as a DESE.

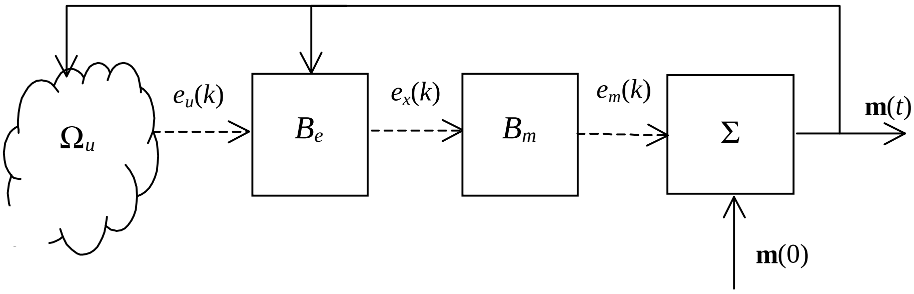

Figure 6: Block diagram of an untimed Petri net

5 Timed Automata and Timed Petri Nets

Timed Petri nets and automata include an internal mechanism for selecting among enabled transitions. The mechanism assigns a deterministic or random delay to each enabled transition, such that the transition with the least delay fires. When a transition fires two events occur: (1) the marking is updated; (2) new transitions might be enabled. Thus the future state evolution at any time t is completely determined by the marking $\mathbf{m}(t)$ and by the set of enabled transitions, which therefore defines at any time t the state of a timed PN or automata. As a consequence timed PN and automata do possess only free state evolution.

5.1 Timed Petri Nets

Timed Petri nets are untimed nets $\{P,T,Pre,Post,\mathbf{m}_0,D\}$ where a function D: $T\rightarrow\Re^+$ specifies the time delay $\tau=D(\lambda)>0$ associated to each transition $\lambda\in\Lambda$. We assume only deterministic delays, but possibly time-varying. An enabled transition λ fires (after its delay) yielding the marking $\mathbf{m}(t^+)=\mathbf{m}(t)+\mathbf{C}(\cdot,\lambda)$. Timed Petri Nets can be mapped into DESE according to the following guidelines:

- The set Y of the output facts corresponds to the set of transitions Λ. Output events $e_y(k)\in E_y=Y\times T$ are defined as $e_y(k)=[y(k)=\lambda,t_y(k)]$ and their occurrences satisfy a one-to-one relationship with transition firings.
- Marking $\mathbf{m}(t)\in \mathrm{N}^n$ defines at any time t a state function vector of dimension n.
- The events $e_m(k)=[\Delta\mathbf{m}(k),t_m(k)]$ have been already defined. They are synchronous with transition firings, that implies $t_m(k)=t_y(k)$. They can be expressed as the output of a static function $e_m(k)=B_m(e_y(k))$.
- Enabled transitions are mapped into potential events $e_\Lambda(i,t)=[\lambda(i),t_\Lambda(i)=t_e+\tau(i)>t]\in E_\lambda=\Lambda\times T$, whose fact is a transition λ and whose occurrence time is given by the enabling time t_e plus the firing delay. A finite sequence $\{e_\Lambda(i,t)\}\in\Sigma(E_\lambda)$ of enabled transitions at time t is denoted by $\eta_x(t)$. Since we assume that any transition can be enabled only once before firing, any transition can appear only once in such sequences.
- Any finite sequence of enabled transitions defines a potential event $e_x(k,t)=[\eta_x(k),t_x(k)>t]$, whose occurrence time $t_x(h)$ is the least occurrence time $t_\Lambda(i)$ of the sequence. The set of potential events $e_x(k,t)$ is denoted with X. Note that all potential events do occur and therefore the hidden set $H(k)$ is always empty. A state function $x(t)$ with values in X defines a further state function besides the marking $\mathbf{m}(t)$.
- Transitions firing corresponds to the occurrence of a potential event $e_x(k)=[\eta_x(k),t_x(k)]$ which in turn generates an output event $e_y(k)=[y(k),t_y(k)=t_x(k)]$ and the next potential event $[e_x(k+1),t_x(k)]$. The next potential event is created through a static function $e_x(k+1)=F_e(e_x(k),\mathbf{m}(t))$, $t=t_x(k)$, which drops from the sequence $\eta_x(k)$ of the enabled transitions that one just fired and possibly add a new enabled transition.

We can now state the following theorem.

Theorem 5.1.1 A timed Petri net with additive marking can be formulated as a DESE that has only free state evolution and it is defined by the following components:

1. Two state functions: the marking $\mathbf{m}(t)$ and the function $x(t){:}T\rightarrow X$ corresponding to the potential event $e_x(k+1)=[\eta_x(k+1),t_x(k+1)]$ at times $t_x(k)<t\le t_x(k+1)$.
2. The state event sequence $\sigma_x=\{e_x(k)\}$ of the occurred potential events.
3. The output event sequence $\sigma_y=\{e_y(k)\}$ of the fired transitions.
4. The input static function B_m that generates the events $e_m(k)$ forcing the adder to modify the marking at any state event occurrence.
5. The feedback function (free evolution) F_e generating the new potential event $[e_x(k+1),t_x(k)]$, thus making the sequence of enabled transitions to evolve.
6. The output function C_e making the first event of the state event sequence to occur (fire). ■

The discrete-event state equations corresponding to a timed Petri nets are defined as follows:

$$\begin{aligned}
&e_x(k+1)=\Phi_e\big(e_x(k),\mathbf{m}(t_x(k))\big)\\
&x(t)=e_x(k+1),\; t_x(k)<t\le t_x(k+1),\; x(0)=x_0\\
&e_y(k)=C_e(e_x(k))\\
&\mathbf{m}(t_x(k)^+)=\mathbf{m}(t_x(k))+B_m\big(e_y(k)\big),\; \mathbf{m}(0)=\mathbf{m}_0\\
&\Phi_e\big(e_x(k),\mathbf{m}(t_x(k))\big)=\Delta_e\big(\big[F_e\big(e_x(k),\mathbf{m}(t_x(k))\big),t_x(k)\big]\big)
\end{aligned}\tag{2}$$

Petri nets that have no transitions with no input places, i.e., their firings simply take place unconditionally, do not have input event sequences and therefore they describe an autonomous DESE, having only free evolution.

Moreover the feedback function F_e does not need to know all marking components in order to enable transitions. All transitions which are enabled by a single place, are actually enabled by the firing of the input transition of such place. In practice tokens are added and simultaneously used for enabling, which is against causality. Therefore such places should be dropped. That means that Petri nets are not minimal state equations, because some state components can be dropped.

The presence of conflicting transitions does not pose any problem. In such case, enabled transitions are included in the potential event with that occurrence time (possibly random). The first transition which fires, makes the conflicting one to be dropped from the next potential event. In Figure 7 we have shown the block-diagram of a timed Petri net represented as a DESE.

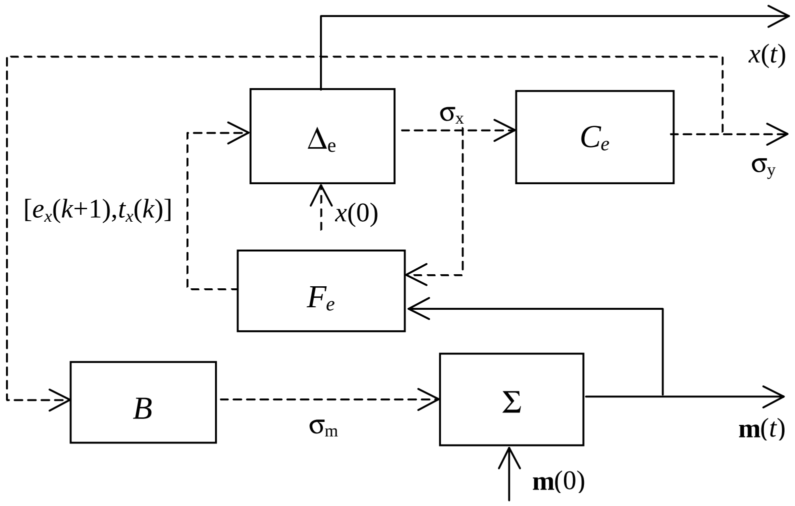

Figure 7: Block diagram of a timed Petri net in terms of DESE

5.2 A Simple Petri Net

Let us now introduce a simple Petri net (Figure 8). We consider a set of transitions $\Lambda=\{a,s,c,b,r\}$: *a*, part arrival; *s*, service starts; *c*: service completion (and part leaves); *b*, server breaks down; *r*, server repaired.

The set of places is $P=\{A,Q,I,B,D\}$. Arrival process is represented by the input place *A* for the transition *a*; since such a place is always marked *a* is kept enabled.

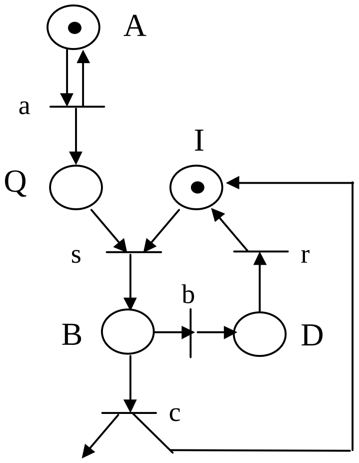

Figure 8: Petri net model of a simple queueing system

Enabling of transition *s* depends on places *Q* and *I*, which implies the presence of a part in the input queue and the server being idle. Transition *c* requires that the server be busy (a token in place *B*). Transitions *b* and *r* model server fail/repair as a an operation-dependent failure. Thus an initial marking may be $\mathbf{m}_0$=[1,0,1,0,0,0], which implies that the queue *Q* is empty and the server is idle. Conflict may arise among transitions *c* and *b* whose enabling depend on the same input place *B*. The conflict is solved by the first enabled transition that fires. The delay time of transition *b* can be progressively reduced with the number of cycles or by the absolute time. It is reset to a larger value after repair (firing of transition *r*). Note that such information is not explicit in the Petri net formalism, but it can be easily included in DESE.

The DESE which define the Petri net model depicted in Figure 8 is obtained as: (1) the output fact set *Y* corresponds to the transition set $\Lambda=\{a,s,c,b,r\}$; (2) the state set *X* is the set of the sequences of enabled transitions, such that each transition can appear only once; (3) the marking time function $\mathbf{m}(t){:}P\times T\rightarrow N^5$ maps the pair (place, time) into a vector of non negative integer values of dimension 5. Actually the set *P* is not minimal, since the three places $\{A,B,D\}$ can be dropped, and a minimal place set $P_0=\{I,Q\}$ can be defined. The static function B_m transforms the four firing transitions $\{a,s,c,r\}$ formulated as output events into three input events for the marking adder. Their facts are $\Delta\mathbf{m}=\{\Delta\mathbf{m}_1=[-1,-1],\ \Delta\mathbf{m}_2=[1,0],\ \Delta\mathbf{m}_3=[0,1]\}$. Hence the four relations are $\Delta\mathbf{m}_1=b_1(\mathrm{s})$, $\Delta\mathbf{m}_2=b_2(\mathrm{c})$, $\Delta\mathbf{m}_2=b_3(\mathrm{r})$, $\Delta\mathbf{m}_3=b_4(\mathrm{a})$. They can be still represented graphically with the Petri net symbols as shown in Figure 9.

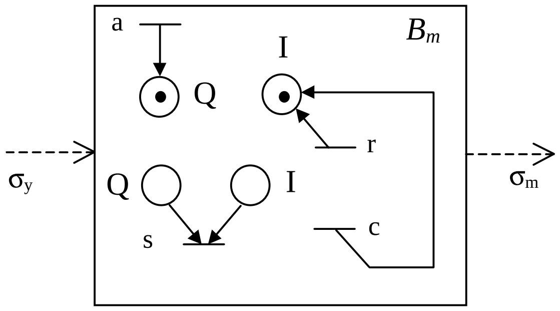

Figure 9: Petri net symbols of function B_m

The feedback function F_e receives the occurred potential events (sequences of enabled transitions at the firing time of the first one) and generates the new potential event, i.e., the new sequence of enabled transitions. The firing transition is dropped from the sequence. If the case, a new transition is enabled. Using the minimal set of places, five functions are necessary (depending on the minimal marking $\mathbf{m}_0$(t)) to generate the new enabled ones. The fact relations are: $(a,s)=f_1(a,m_1(t))$, $(c,b,\underline{s})=f_2(s)$, $(r,\underline{b},\underline{c})=f_4(b)$, $(s,\underline{r})=f_5(r,m_2(t))$ $(s,\underline{b},\underline{c})=f_3(c,m_2(t)$, where underline means that the corresponding transition has to be dropped. They can be graphically represented with a formalism similar to a Petri net as shown in Figure 10.

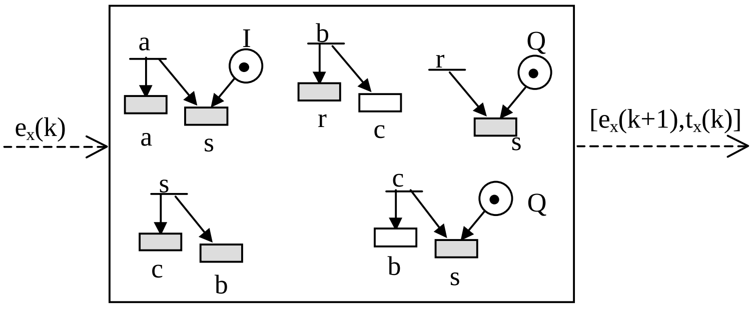

Figure 10: Petri net symbols of function F_e

The colored rectangles represent new enabled transitions while the white ones dropped transitions (conflicting). Here each transition when fires drops itself, except *a*.

5.3 Timed Finite State Automata

Timed automata, like timed Petri nets, includes an internal mechanism for selecting among the enabled (or feasible or activated) transitions (we prefer the name transition instead of event, not to make confusion with our definition of event) (Cassandras 1993). The time delay mechanism allows to distinguish between enabling and firing times.

The state is twofold: (1) the state $x(t)\in X$ of the finite automata, (2) a time function $z(t)\in Z$ registering the potential (or state) event $e_x(k)$ whose fact is the finite sequence η_x of enabled transitions at time t. When a state event $e_x(k)$ occurs at time $t_x(k)$ the following actions are taken:

- The first transition (least occurrence time) of the state event fires, and an output event $e_y(k)$ is generated by a static function C_e.
- The state $x(t)$ is updated as $x(k+1)=x(t_x(k)^+)$ depending on the previous state $x(k)$ and on the output event $e_y(k)$ by a static function B_e.
- The state $z(t)$ is updated according to some rules (Donati *et al.*, 1996): all enabled transitions of $e_x(k)$ are dropped; the transitions enabled by the new state value $x(k+1)$ are included in $e_x(k+1)$; the occurrence times of the new enabled transitions which belong to the previous state event $e_x(k)$ do not change, except for the fired transition. Such rules are described by a feedback (free evolution) function F_e forcing the new potential event to a delay operator.

We can now state the following proposition.

Proposition 5.3.1 Any timed finite automaton can be described by the following discrete-event state equations:

$$
\begin{aligned}
& e_x(k+1) = \Phi_e\left(e_x(k), x(k)\right) \\
& z(t) = e_x(k+1),\ t_x(k) < t \le t_x(k+1),\ z(0) = z_0 \\
& e_y(k) = C_e(e_x(k)) \\
& x(k+1) = R\left(B_m\left(e_y(k), x(k)\right)\right),\ \ x\left(t_x(k)^+\right) = x(k),\ x(0) = x_0 \\
& \Phi_e\left(e_x(k), x(k)\right) = \Delta_e\left(\left[F_e\left(e_x(k), x(k)\right), t_x(k)\right]\right)
\end{aligned}
\tag{3}
$$

■

Finally, in Figure 11 we have shown the block-diagram of a timed finite automaton represented as a DESE.

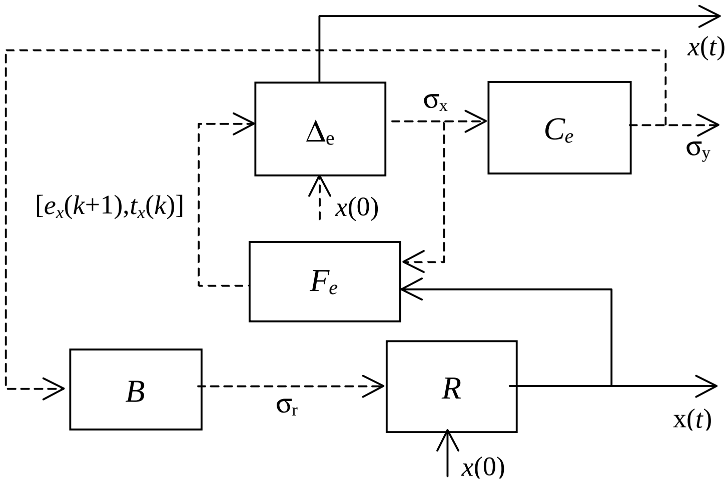

Figure 11: Block diagram of a timed finte automaton in terms of DESE

6 Conclusions

A novel theory of discrete-event dynamic systems (DEDS) has been shortly presented in this paper. The theory allows us to describe any classical DEDS model in terms of discrete-event state equations (DESE) satisfying Kalman axioms (Kalman *et al.*, 1969), therefore exhibiting forced and free state evolutions.

The state is defined as the sequence of potential events (enabled transitions in terms of Petri nets language) forced by the occurrence of state events themselves (free evolution) or by arbitrary input events (forced evolution). The theory puts in evidence that automata and Petri nets cannot be considered the most general DEDS since they miss either free or forced evolution. The theory has been already applied to real-time control of Manufacturing Systems (Vallauri, 1997). Hybrid dynamic systems can be easily defined over this theory.

Our main future goal will be to explore potential application fields of this theory for the optimal design and control of discrete event dynamic processes.

7 Acknowledgments

The authors are grateful to Prof. Francesco Donati, Politecnico di Torino, who has been the initiator of the novel theory presented in this work.

References

Balduzzi, F. and G. Menga (1998). "A State Variable Model for the Fluid Approximation of Flexible Manufacturing Systems," *IEEE Int. Conf*erence *on Robotics and Automation* (Leuven, Belgium), pp. 1172-1178.

Canuto, E. (1998a). "Discrete-Event Models of Manufacturing Systems," *Proc. 6th IEEE Mediterranean Conf. on Control and Systems* (Alghero, Italy).

Canuto, E. (1998b). "Discrete-Event Modelling and Control of Manufacturing Systems," *Proc. 1998 IEEE Conf. on Control Applications* (Trieste, Italy).

Cassandras, C.G. (1993). *Discrete Event Systems*: *Modeling and Performance Analysis*, R.D. Irwin, Inc. and Aksen Associates, Inc.

Donati, F., E. Canuto, and M. Vallauri (1996). "A new Approach to Discrete-Event Dynamic System Theory," in *Periodica Polytechnica*, *Ser. El. Eng.*, Vol. 41, pp. 1-11.

Gershwin, S.B. (1994). *Manufacturing Systems Engineering*, Prentice Hall, Inc., Englewood Cliffs.

Glynn, P.W. (1989). "A GSMP Formalism for Discrete Event Systems," *Proc. IEEE*, Vol. 77, No. 1, pp. 14-23.

Kalman, R.E., P.L. Falb, and M.A. Arbib (1969). *Topics in Mathematical System Theory*, McGraw-Hill.

Murata, T. (1989). "Petri Nets: Properties, Analysis and Applications," *Proc. IEEE*, Vol. 77, No. 4, pp. 541-580.

Passino, K.M. and K.L. Burgess, (1998). *Stability Analysis of Discrete Events Systems*, J. Wiley & Sons, Inc.

Ramadge, P.J.G. and W.M. Wonham, (1989). "The Control of Discrete Event Systems," *Proc. IEEE*, Vol. 77, No. 1, pp. 81-98.

Vallauri M. (1997). "A new methodology for Manufacturing System Engineering," *Proc. 2nd HIMAC Workshop*, CELID, Torino.

Ziegler, B.P. (1989). "DEVS Representation of Dynamical Systems: Event Based Intelligent Control," *Proc. IEEE*, Vol. 77, No. 1, pp. 72-80.