Forrester Diagrams and Continuous Petri Nets: A Comparative View

E. Jiménez
Dept. Ingeniería Eléctrica
Universidad de La Rioja
Luis de Ulloa 20, E-26004, Logroño (Spain)
E-mail: emilio.jimenezm@die.unirioja.es

L. Recalde and M. Silva *
Dept. Informática e Ingeniería de Sistemas
Universidad de Zaragoza
María de Luna 3, E-50015, Zaragoza (Spain)
E-mail: {lrecalde, silva}@posta.unizar.es

Abstract – **Forrester Diagrams (FD) and Petri Nets (PN) are formalisms introduced in the sixties to model complex systems. This paper explores similarities and differences between FDs and the continuous relaxation of the originally discrete PNs. Historically speaking, the approaches were quite different: the PNs paradigm was introduced at a very abstract level, without timing interpretation, while FDs led to a modelling methodology were the systematic simulation of a set of differential equations was the goal. Strict flow conservation around valves, non explicit fork and join operations, separation of information and material flows, are peculiarities of FDs. In PN models the existence of global conservation laws is a potential for structural analysis.**

I INTRODUCTION

The decade of 1960's sees the consolidation, among others, of two "very different" formalisms and methodologies for modelling dynamic systems. On the one side, Jay W. Forrester, an engineer with an Automatic Control background, working in the modelling of industrial and urban systems, started the System Dynamics Group at MIT, from which Systems Dynamics derives [7, 9]. In essence, a modelling methodology using Causal Diagrams (CD) and, the later called, Forrester Diagrams (FD) allows to systematise the construction of continuous models based on systems of non-linear, multivariable, time dependent differential equations. The focus is in model building, while analysis is basically bounded to simulation. Insufficiencies of the simulation approach were pointed out and formal analysis techniques are also in use from the 1980's [12]. On the other side, C. A. Petri, a mathematician working in Computer Science, defines in 1962 a formalism to deal with concurrency and cooperation relationships in Discrete Event Dynamics Systems (DEDS), computer systems, in particular. This formalism and modelling methodologies were further developed at MIT by A. Holt's group, who baptised it as Petri Nets, and at GMD (Germany) by the Petri's Group. Successive developments in this field led to a family of related formalisms. Different abstraction levels (elementary [21], place/transition [16], colored [10], predicate/transition [11] ...) and different interpretations (timed, stochastic ...) provide a rich modelling paradigm for DEDS [19].

In both cases, the modelling of "general enough" systems was contemplated. Forrester's view leads to continuous models, while Petri's view deals with discrete models. In the first case, modelling was the key issue, while in the second much effort has been devoted to formal analysis techniques (state space exploration, model reduction, mathematical programming ...) [20]. The state explosion problem, inherent to the enumerative analysis of DEDS models, is particularly crucial when large populations are flowing through a system. But large populations usually lead to "relatively small errors", if the discrete model is relaxed to a continuous approximation. This way, in 1987 Petri Nets (PN) were interpreted with markings in the non-negative reals (Continuous PNs) [5]. Continuous nets are particularly interesting in the framework of performance evaluation, in which computing an "educated guess" for some performance indexes is the goal. At the same time the state equation associated to discrete PNs was similarly relaxed for the purpose of analysability, leading usually to semi-decision algorithms [17]. Recently it has been realised that although different, this two relaxations are "essentially" identical in practice [15]. At this point a natural question appears: Which are the similarities and the differences between Forrester Diagrams and continuous Petri Nets? The purpose of this work is to advance in providing answers to this question, a topic that was just brought to mind in [18]. The present paper is structured as follows. Petri Nets and its continuous relaxation are addressed in Section II, while System Dynamics and Forrester Diagrams is the topic of Section III. A simple manufacturing system is considered from both perspectives in Section IV. Finally, Section V presents some preliminary comparative remarks.

II CONTINUOUS PETRI NETS

A. *PNs definitions*

Petri nets (PNs) constitute a well-known formal paradigm for the modelling, analysis, synthesis and implementation of systems that "can be seen" as discrete. We assume the reader is familiar with PNs (see for instance [13, 16, 19] for an introduction of the basic concepts and notations of PNs). We will just remark that a system is an structure

* This work has been partially supported by Project CICYT TAP98-0679.

0-7803-7241-7/01/$10.00 (c)2001 IEEE

$\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ ($\mathbf{Pre}$ and $\mathbf{Post}$ represent the static structure of the model, from which the token flow matrix $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ can be deduced) provided with an initial marking over P, $\mathbf{m_0}$. A Petri net structure can also be represented as a bipartite directed graph, in which places are usually represented as circles and transitions as bars. In a PN, the marking defines the state of the system, and it is changed by the firing of transitions, thanks to the occurrence of their associated events. Starting from a PN system, a state (or fundamental) equation can be written:

$$\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}, \text{ where } \boldsymbol{\sigma} \in \mathbb{N}^{|T|}, \text{ and } \mathbf{m} \in \mathbb{N}^{|P|}$$

The places of a PN system could be seen as the state variables, and the marking vector as the state vector. However, it must be taken into account that there may exist redundancies. That is, it may happen that the marking of a place can be always obtained as a linear combination of the marking of other places.

The set of reachable states of a discrete PN system may easily become extremely large (the so called *state explosion problem*). A way to try to overcome this problem, is to continuize the system, what allows the use of different mathematical tools (linear programming techniques, differential equations ...).

The usual PN system, $\langle \mathcal{N}, \mathbf{m_0} \rangle$, will be said to be *discrete* so as to distinguish it from its *continuous relaxation*. In a discrete PN the marking is restricted to be integer, while in continuous PNs any non-negative real number is allowed. In continuous PNs the firing is modified in the same way, that is, a transition t is *enabled* at $\mathbf{m}$ iff for every $p \in {}^\bullet t$, $\mathbf{m}[p] > 0$. Its *enabling degree* is defined as $\text{enab}(t, \mathbf{m}) = \min_{p \in {}^\bullet t} \{\mathbf{m}[p]/\mathbf{Pre}[p, t]\}$. The firing of t in a certain amount $\alpha \leq \text{enab}(t, \mathbf{m})$ leads to a new marking $\mathbf{m'} = \mathbf{m} + \alpha \cdot \mathbf{C}[P, t]$.

The continuization of a net system is intended as an *approximation*. A first thing to point out is that not all net systems allow a "reasonable" continuization. Examples can be shown for which the lack of relationship between the qualitative properties of the discrete and the continuous system may certainly look surprising [15]. For example, deadlock-freeness of the continuous systems is neither necessary nor sufficient for deadlock-freeness of the discrete system (not even under structural boundedness).

Different timing interpretations can be associated to a (discrete) Petri net. One possibility is to assign a deterministic fixed delay to each transition (deterministic timed nets). Another one is to consider that the delay of each transition is exponentially distributed (markovian stochastic Petri nets). For continuous nets we will use a deterministic approximation for both interpretations, and either the deterministic delay or the mean value of the exponential distribution function will be used to define the firing speed of the transition.

As in discrete systems, in a continuous PN the state equation $\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$ summarises the marking evolution. But, in continuous systems, the marking is continuously changing, so we may consider the derivative of $\mathbf{m}$ with respect to time. This way we obtain that $\dot{\mathbf{m}} = \mathbf{C} \cdot \dot{\boldsymbol{\sigma}}$, plus the initial condition $\mathbf{m}(0) = \mathbf{m_0}$. Let us call $\mathbf{f} = \dot{\boldsymbol{\sigma}}$, since it represents the *flow* through the transitions. In general $\mathbf{f}$ is not constant, but may depend locally on the marking, thus on time. Observe that if a steady state is reached, $\dot{\mathbf{m}} = \mathbf{0}$, and so $\mathbf{C} \cdot \mathbf{f} = \mathbf{0}$ (since $\mathbf{f} \geq \mathbf{0}$, it is a T-semiflow, according to the usual notation)

If $\mathbf{f}(\tau)$ is defined by an *interpretative extension*, the timed evolution of the continuous PN can be obtained. Two particularly interesting semantics are often used in discrete PNs, and they can be extended to the continuous case [14]:

1. *Infinite servers semantics.* In this case, transitions are fired with: $\mathbf{f}(\tau)[t_i] = \boldsymbol{\lambda}[t_i] \cdot \mathbf{e}(\tau)[t_i]$, where $\mathbf{e}(\tau)[t_i] = \min_{p \in {}^\bullet t_i} \{\mathbf{m}[p]/\mathbf{Pre}[p, t_i]\}$ is the enabling degree of t_i, and $\boldsymbol{\lambda}[t_i]$ is the rate associated to t_i. That is, $\mathbf{e}(\tau)[t_i]$ represents the number of active *servers* in the *station* (transition), at instant τ.

   Observe that the fluidified model is a *set of switching systems of linear differential equations* with constant coefficients. In the example of Figure 1, if it is seen as a continuous PN system with infinite-servers semantics, the flow vector is:

$$\begin{aligned} \mathbf{f}[t_1] &= \boldsymbol{\lambda}[t_1] * \min\{\mathbf{m}[p1]/3, \mathbf{m}[p2]\} \\ \mathbf{f}[t_2] &= \boldsymbol{\lambda}[t_2] * \mathbf{m}[p3] \\ \mathbf{f}[t_3] &= \boldsymbol{\lambda}[t_3] * \mathbf{m}[p4]/2 \end{aligned}$$

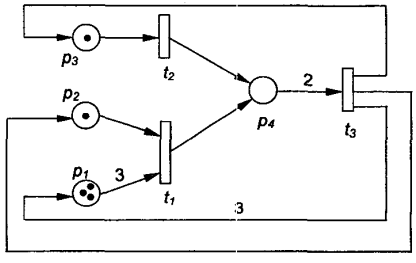


Figure 1: A continuous PN system

2. *Finite servers semantics.* In discrete PNs, the constraint on the number of servers can be made explicit by elementary self-loops around each transition t_i marked with k_{t_i} tokens, as many as the number of servers. However, the meaning of the "servers tokens" and the "client tokens" is very different for continuous systems, since the latter represent large populations while the former are count as units. This immediately suggests that the speed $\mathbf{f}(\tau)][t_i]$ has just an upper bound (k_{t_i} times the speed of a server, $\mathbf{F}[t_i]$). Then $\mathbf{f}(\tau)[t_i] \leq k_{t_i} \cdot \mathbf{F}[t_i]$ (knowing that at least a transition will be in saturation, that is, its utilisation will be equal to 1).

In continuous PNs terminology, infinite servers semantics is "variable speed"; while finite servers semantics is named "constant speed" (see for instance [1]), what in fact corresponds to a "bounded" speed.

B. A basic population model

Let us consider a simple version of the predator/prey model of Volterra-Lotka. This is a so well-known model that, for brevity reasons, is not explicitly introduced here (it can be found, for instance, in [3]).

The colored PN in Figure 2(a) represents the problem using a discrete model (more realistic hypothesis could be introduced in a simple way, but providing a more elaborated model is not our goal here). The use of colored PNs is simply methodological here, to reveal the existence of individuals that can be grouped in homogenous populations. To do that, the model has to be *decoloured* [6] and we have to obtain the firing rates of the new transitions. Let $\mathbf{m}[f]$ and $\mathbf{m}[r]$ be the number of predators and preys (foxes and rabbits, for example). If we consider the colored transition t_3 at a certain instant, it is enabled in $\mathbf{m}[r] \cdot \mathbf{m}[f]$ differently colored ways. For this reason in the decoloured (discrete) model (see Figure 2(b)) t_3 has an associated firing rate equal to $\lambda[t_3] \cdot \mathbf{m}[r] \cdot \mathbf{m}[f]$. Both discrete net systems in Figure 2 are non bounded and non live. In fact, they have two absorbent "states" (or attractors): in both of them $\mathbf{m}[f] = 0$, and either $\mathbf{m}[r] = 0$ or $\mathbf{m}[r] = \omega$ (ω is an arbitrarily large number). Only $\mathbf{m}[r] = \mathbf{m}[f] = 0$ is a steady state.

Observe that with $\mathbf{m}[r] \cdot \mathbf{m}[f]$, the *product of variables* has been introduced as a rate, i.e., a new semantics has appeared from the "decoloration" of the usual infinite servers of the colored transition.

If the constants in Figure 2(a) (death and birth rates) are defined as $\alpha_r = 0, \alpha_f = \alpha, \beta_r = 2, \beta_f = 0$ (Figure 2(b)), the equations associated to the continuous and decoloured PN are the classical Volterra-Lotka equations:

$$\dot{\mathbf{m}}[r] = \lambda[t_1] \cdot \mathbf{m}[r] - \lambda[t_3] \cdot \mathbf{m}[r] \cdot \mathbf{m}[f]$$
$$\dot{\mathbf{m}}[f] = -\lambda[t_2] \cdot \mathbf{m}[f] + (\alpha - 1) \cdot \lambda[t_3] \cdot \mathbf{m}[r] \cdot \mathbf{m}[f]$$

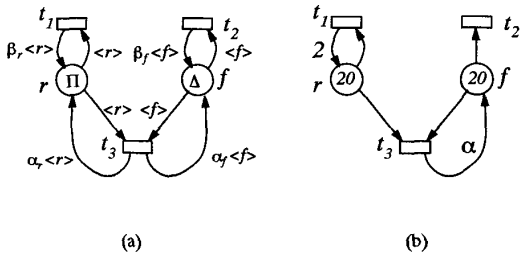


Figure 2: Colored and place/transition net model of a predator/prey system

For $\dot{\mathbf{m}}[r] = \dot{\mathbf{m}}[f] = 0$ the classical equilibrium solution is found: $\mathbf{m}[r] = \lambda[t_2]/(\lambda[t_3](\alpha - 1))$ and $\mathbf{m}[f] = \lambda[t_1]/\lambda[t_3]$. However, it must be noticed that according to this model, the system does not have equilibrium solutions, but oscillates in orbits defined by the initial populations.

In our example, the discrete PNs (colored or not) are *stochastic non bounded and non live* models. In particular, in a "large enough" run, predators will disappear (with probability 1) and preys will either disappear or grow infinitely.

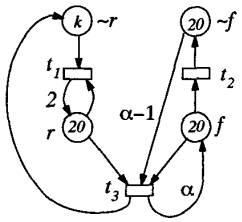


Figure 3: Place/transition net model of a bounded predator/prey system.

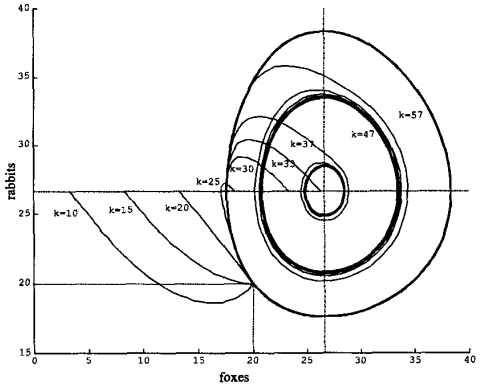


Figure 4: Trajectories obtained for the system in Figure 3 with $\lambda[t_1] = \lambda[t_2] = 20$, $\lambda[t_3] = 0.75, \alpha = 2, \mathbf{m_0}[r] = \mathbf{m_0}[f] = 20$, $\mathbf{m_0}[\sim f] = 40$ and $\mathbf{m_0}[\sim r] = k$.

The continuous PN, which was intended to be an "approximation" of the discrete model, is *deterministic, bounded and live*! One could imagine that boundedness and liveness are due to a "certain equilibrium" between the non boundedness and the deadlocks of the discrete system. To deepen into this question, the discrete PN in Figure 2(b) has been transformed into a bounded net system, just adding complementary places to r ($\sim r$) and f ($\sim f$) (see Figure 3). Seen as discrete, this system is bounded and contains deadlocks. The underlying stochastic process will sooner or latter enter into one of the deadlocks ($\mathbf{m}[f] = 0$, with either $\mathbf{m}[r] = 0$ or $\mathbf{m}[r] = k + 20$). Nevertheless, its continuous approximation is live.

Just as an exercise, Figure 4 shows the trajectories for the case of having a maximal number of preys of $20 + k$. Since, for the given $\mathbf{m_0}$, the place $\sim f$ is always greater than 0, it never restricts the enabling of t_3. Hence, the equation of f in the steady state is: $\lambda[t_3] \cdot \mathbf{m}[r] \cdot \mathbf{m}[f] - \lambda[t_2] \cdot \mathbf{m}[f] = 0$, and so: $\mathbf{m}[r] = \lambda[t_2]/\lambda[t_3] = 80/3$. The behaviour of the system when k decreases shows that in a first transitory phase, the limitation has the effect of placing the system in an orbit closer to the non null equilibrium point. From a critical value, the evolution does not lead the system to an orbit, but it directly goes to an equilibrium point, with $\mathbf{m}[r] = 80/3$ (obtained from the equation $\dot{\mathbf{m}}[f] = 0$). For

$k = 20$, the main constraint for t_1 is always $\sim r$, hence the flow at t_1 is $20 \cdot \mathbf{m}[\ r]$ which (by conservation) must be equal to the flow at t_2, $20 \cdot \mathbf{m}[f]$; and so $\mathbf{m}[f] = \mathbf{m}[\sim r]$. Therefore $\mathbf{m}[r] + \mathbf{m}[\sim r] = 20 + k = 40 \Rightarrow \mathbf{m}[\sim r] = 40 - 80/3 = 40/3$, and so $\mathbf{m}[f] = 40/3$. (In fact, for $k = 20$, $\dot{\mathbf{m}}[r] = -\dot{\mathbf{m}}[f]$, from which the straight line in the figure.)

Some final remarks:

- If the starting point is a colored net, it is necessary *first to decolour* (from which large populations can be obtained) and *later to fluidify*. The reverse, first fluidify and then decolour, does not make sense, specially considering that "addition" and "min" operations do not commute.
- If just infinite and finite-servers semantics are allowed, the continuous PN is in fact a set of switching linear systems. The use of the product of the markings as the firing rate allows to represent more complex behaviours. However still the locality principle is preserved. That is, the rate depends only on the making of the input places (i.e., a local precondition).
- If functions that depend on the *global state* of the system are allowed in $\mathbf{f}(\tau)[t]$, *chaotic* behaviours (even the classical of Lorenz [12]) may be represented with continuous PNs. Since continuization is a relatively strong relaxation, the chaotic trajectories may not be *very representative*. Hence, it is possible that just their *qualitative* properties make sense (see [2, 12] for some reflections about this).

III FORRESTER DIAGRAMS

A. Forrester Diagrams definitions

Forrester Diagrams (FD) are specific modelling tools inside System Dynamics (SD) [7, 8, 9]. SD is a methodology for the study and analysis of complex continuous systems, which tries to build dynamic models of complex systems, by searching the relationships between the subsystems (specially the feedback loops). It looks at the system as a whole, usually using the computer for simulation. The genesis and the development of SD constitute a manifestation of the paradigm of systems.

The methodology to build a model in SD could be summarised in several steps [8], which are applied in an iterative way until the desired adjustment is obtained:

1. *Conceptualisation*, which includes: a) identifying the system and its parts, b) looking for the causal relationship and feedback loops, and c) building the Causal Diagram.
2. *Representation and formulation*, which include: d) building the so called Forrester Diagram, and e) writing the equations of the system.
3. *Analysis and evaluation*, which include: f) model analysis: comparison to the reference model and sensibility analysis, and g) evaluating and implementing the system.

In this methodology two graphical models are used: Causal Diagrams, and Forrester Diagrams. A differential equations based model is straightforwardly derived from the later.

Causal Diagrams (CD) qualitatively show the causal relationships between the parts (subsystems), by means of arrows with a sign that indicates if the relationship is positive (greater/less cause implies greater/less effect) or negative (the opposite). In these diagrams it is not distinguished if the parts will be state variables or another type of variables. Special attention is paid to the feedback loops (a closed chain of causal relationships) because they provide a first idea of how the system will evolve dynamically: positive feedback loops (even number of negative relationships) "indicate" an exponential grow, and negative feedback loops indicate the possibility of balance and equilibrium.

Certain recommendations exist for the construction of Causal Diagrams: avoid the fictitious loops, use easily quantifiable elements, do not use twice the same relationship, avoid redundant loops and do not use time like a causal factor.

Forrester Diagrams provide a graphic representation of dynamic systems (see Figure 5), modelling quantitatively the relationships between the parts by means of some symbols, which correspond to an hydrodynamic interpretation of the system.

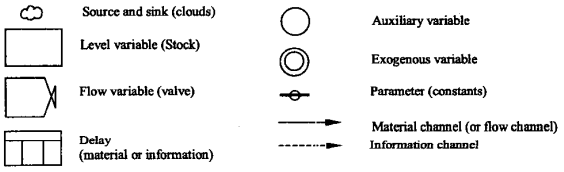


Figure 5: Forrester Diagrams elements

The *levels* (stocks) correspond to the *state variables* in systems theory. They represent the variables whose evolution is more significant for the study of the system. The levels accumulate "material" from material channels, which are controlled by the *valves* (flow variables). This material flow is strictly conservative (balance around the valves). Valves define the behaviour of the system, since they determine the speed of the material flow (through the material channels) according to a set of associated equations. The equations depend on the information that the valves receive from the system (levels, auxiliary variables and parameters) and from the environment (exogenous variables). The information is transmitted instantaneously through *information channels*. *Auxiliary variables* correspond to intermediate steps in the calculation of the functions associated to the valves. They can be used to simplify the process, either because some mathematical calculations are used for several equations (reused computation of flows), or because they have certain physical meaning or interpretation that could

be interesting to observe, but they can always be removed. The *clouds* represent sources and sinks, that is to say, a non determined (infinite) amount of material, and the *parameters* are constant values of the system. The interaction of the system with the exterior is represented with the *exogenous variables*, which have an evolution that is assumed to be independent from the evolution of the system. The *delays* can affect the material or the information transmission, but in both cases they do not introduce more description capacity, because they just correspond to a compact notation of elements that produce these delays (see Figure 6).

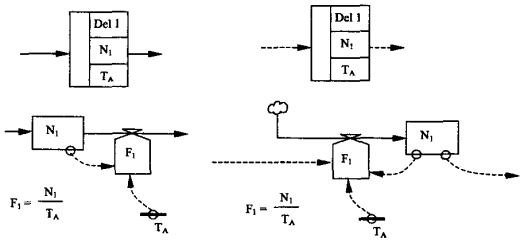


Figure 6: Material and information first order delays in Forrester Diagrams

The interest of the hydrodynamic analogy is that indicates that a FD model is equivalent to a first order (eventually non linear, time dependent) differential equation system, and vice versa. The equations of the model are simply the analytic representation of the FD, and allow not only simulation of the model but also the application of modern control theory techniques. The equations just correspond to the material balance in each deposit:

$$x(t) = x(0) + \int_0^t (\sum f_{IN} - \sum f_{OUT})dt$$
$$\Downarrow$$
$$dx(t)/dt = \sum f_{IN} - \sum f_{OUT}$$

where x are the level variables, and f_{IN} and f_{OUT} represent the functions associated to the valves (flow functions) that introduce or take out respectively material in a level. Since the flow variables, f_{IN} and f_{OUT}, depend both on the levels and on the exogenous variables (the auxiliary variables can always be eliminated), it corresponds to a system of first order differential equations:

$$dx/dt = f(x, u)$$

where u represents the exogenous variables.

B. A basic population model

Let us consider the same very simple predator/prey model as in Section II.B. The goal of this example is just instrumental, to show the process of model construction, but not to provide a real approximation by a complex model. The Causal Diagram of that system is shown in Figure 7. Note that in the diagram the 'Captures' should influence the 'Foxes' through the 'Foxes births' and the 'Foxes deaths' rates, (2) and (3), instead of directly (1), but to simplify the model and make it more similar to the previous one, (2) and (3) relationships have been summarised by their equivalent (1).

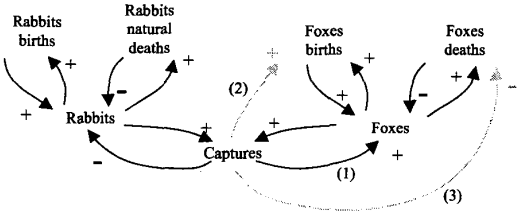


Figure 7: Causal Diagram of basic predator/prey model

The FD obtained after further elaboration is shown in Figure 8. It can be realised that valve r_b includes the effect of births and deaths of rabbits for natural causes, and that f_b includes the effect of births and deaths of foxes for natural causes (excluding the captures effect). In both cases the valves could be divided into two, one for each cause. Observe that flow equations must be included in the FD to fully describe the model. In this case, the equations correspond to a model that is equivalent to the PN model in Section II.B (Figure 2(a)), which comes from a discrete model.

Figure 9 corresponds to the PN model of Figure 3, where the populations capacities are bounded. This limitation, or any other one, can be introduced in models either through the flow equation (for example in bounding the foxes level, by f_{max}), or through its graph (or a chart of values) that describes the function (as in the limitation of the rabbits level, by lim_r). Models in Figures 3 and 9 lead to the same system of equations.

IV A SIMPLE MANUFACTURING SYSTEM

Let us consider the simple manufacturing system sketched in figure 10. It basically consists in the manipulation in machines and the storage in buffers of two types of parts, *a* and *b*, that are assembled to obtain a final product. One of each kind of parts comes to machine 2 (through its respective machines, 1a or 1b, and buffers, 1a or 1b), where they are joined. The resulting part is stored again in another buffer, and waits until the machine 3 generates the final product. Taking a part from a buffer takes 0.2 time unites, and each operation needs 1 time unit. All buffers capacities are 3. There is a limitation in the number of parts of the system, represented by parameter *k*.

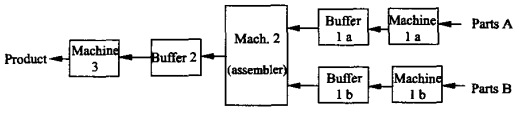
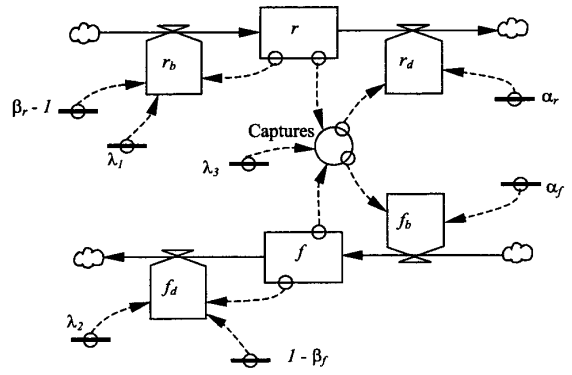


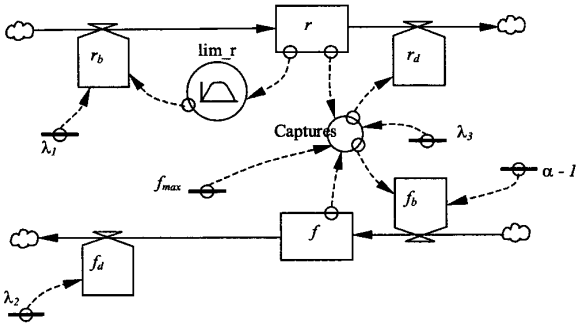
Figure 10: Diagram of the Simple Manufacturing System

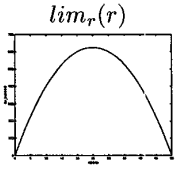
This is a discrete system. A discrete PN model is shown in Figure 11(a), where each element has been modelled by means of two places (a place and its complementary one). Therefore, it can be observed that the number of places is



$$captures = \lambda_3 \cdot r \cdot f$$
$$r_b = \lambda_1 \cdot (\beta_r - 1) \cdot r$$
$$r_d = captures \cdot (1 - \alpha_r)$$
$$f_b = captures \cdot (\alpha_f - 1)$$
$$f_d = \lambda_2 \cdot (1 - \beta_f) \cdot f$$

Figure 8: Forrester Diagram of a basic predator/prey model (equivalent to Figure 2(a))





if $f < f_{max}$ then
$captures = \lambda_3 \cdot r \cdot f$
else $captures = 0$
$$r_b = \lambda_1 \cdot lim_r(r)$$
$$r_d = captures \cdot (1 - \alpha_r)$$
$$f_b = captures \cdot (\alpha_f - 1)$$
$$f_d = \lambda_2 \cdot (1 - \beta_f) \cdot f$$

Figure 9: Forrester Diagram of a bounded basic predator/prey model

different from the number of state variables, because only a state variable is needed for each element. However, for large markings, great computational effort is required to carry out the simulation as discrete, and a continuous approximation may be interesting. The model in Figure 11(a) can be interpreted as a continuous system, with infinite servers semantics (usually used in this paper). The finite servers semantics model can be built too, keeping in mind that a server (machine) cannot simultaneously load and unload parts, and therefore the two delays should be added. In this latter case the system has a very similar interpretation to an hybrid system, as it can be observed in Figure 11(b), where machines have been represented as discrete places, and the remaining places, and all the transitions, are continuous.

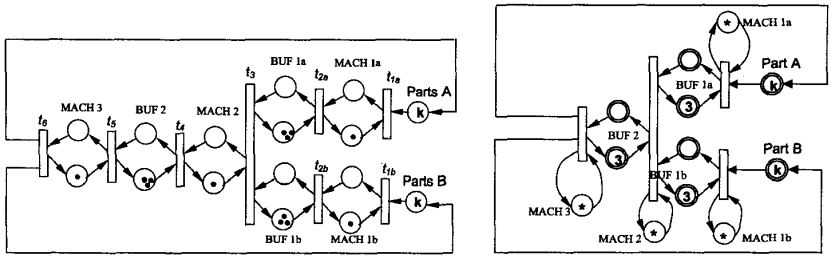
Up to now we have seen that this discrete system can be analysed with a PN using a discrete deterministic model, a continuous model with infinite servers semantics, and a continuous model with finite servers semantics. We could wonder if they all will give "similar" results, and it is not really the case. The results of the referred cases are represented in Table 1. It shows the throughput in steady-state for the initial marking shown in Figure 11, depending on k (the bound on the number of parts in the system). Note that in this case continuous infinite servers and discrete deterministic models provide the same production rates, a general result for strongly connected Marked Graphs.

k	Discrete	Continuous Infinite servers	Continuous Finite servers
1	0.278	0.278	0.833
2	0.555	0.555	0.833
> 3	0.833	0.833	0.833

Table 1: Comparing the throughput of the discrete and the continuous models (under infinite and finite servers semantics) of the manufacturing system

These values indicate that it is necessary to be careful when analysing the behaviour of a model if approximations are used. With the continuous approximations the computation is simplified but some accuracy may be lost. This difference can be observed more clearly in the extreme case of having a single part of each type ($k = 1$). In that case it is evident that as deterministic discrete, the time for producing a new part, i.e., the inverse of the throughput, in steady-state is the sum of the times of the slowest branch, that is, 3.6 seconds. But as continuous with finite server semantics, the time is only that of the slowest transition, that is, 1.2 seconds. Thus, a coefficient of three makes the difference (!).

The behaviour of the system can also be simulated with Forrester Diagrams. When modelling the system by means



(a) (b)

Figure 11: a) PN model of a manufacturing system (can be seen as discrete or as continuous), b) PN model of the manufacturing system under finite servers semantics

of FDs the machines are considered as material delays between the levels, which are the buffers (Figure 12(a)). The greater the order of the delay is, the more it may look like the deterministic discrete system. Nevertheless, this is not crucial, because the usual procedure in FDs is to adjust the time parameters in the delays to get the observed production rate.

Any continuous PN can be translated into a FD, building the FD from the equations that derive from the PN. A direct translation of the PN in Figure 11(a) into a FD is shown in Figure 12(b) (obviously the finite servers semantics model could have been translated too). The FDs in figures 12(a) and 12(b) are "different" although of course both have "equivalent" behaviours. The methodologies of both formalisms, PNs and FDs, have driven in this case to different models.

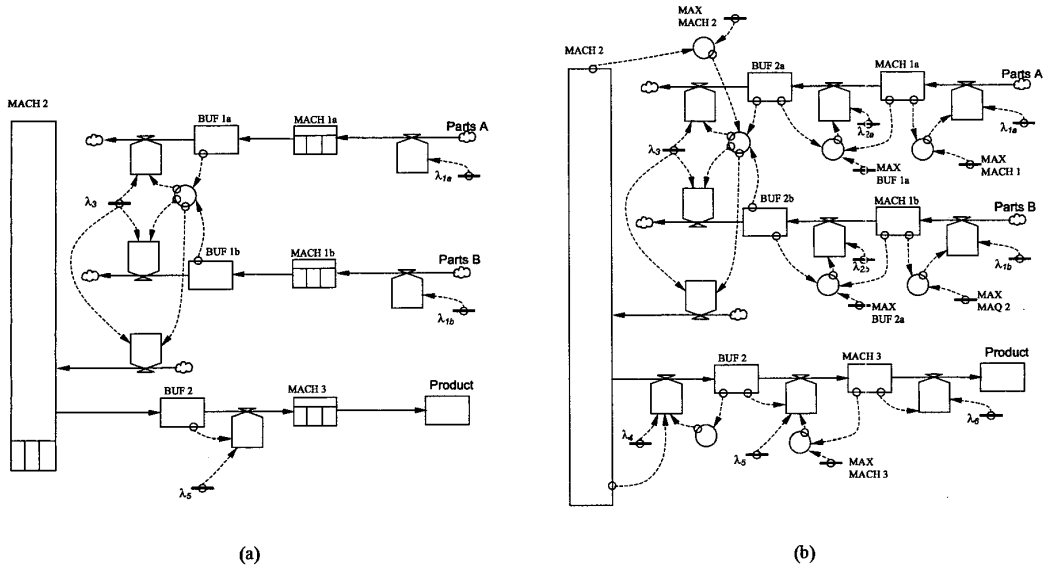
Figures 12(a) and 12(b) show that the parts flow is broken when they join in machine 2. To model that a part of type *a* and a part of type *b* are joined in machine 2 producing a new part, the FDs operate as follows: the information of how many parts come in machine 2 is used to eliminate these parts from buffers *1a* and *1b* and to generate from another source the corresponding number of parts, which represent the parts produced in machine 2. The connectivity (and the synchronisation) in the process is broken down in the structure of the FDs. They are implicitly conserved by the equations. On the contrary, PNs preserve this information in the structure, by means of the and-nodes (fork and joins) and the weights in the arcs.

V CONTINUOUS PNs vs. FDs: SOME REMARKS

In the previous sections continuous PNs and FDs have been briefly presented and some examples analysed from both points of view. Both provide a graphical support for easy generation of systems of differential equations. A clear correspondence exists among the main types of nodes in both: place/level and transition/valve (or firing speed/flow variable). However, this correspondence should not hide the differences that appear:

1. *Marking of places vs. levels.* In FDs each level corresponds, to a state variable. However, although in PNs places are essentially state variables, redundancies may exist due to token conservation laws derived from P-flows (y is a P-flow iff $\mathbf{y} \cdot \mathbf{C} = \mathbf{0}$, thus $\mathbf{y}^T \cdot \mathbf{m} = \mathbf{y}^T \cdot \mathbf{m_0}$). Particular cases are structural implicit places (a place is implicit iff it never restricts the firing of its output transitions) [4] and conservativeness ($\exists \mathbf{y} > \mathbf{0}$ such that $\mathbf{y} \cdot \mathbf{C} = \mathbf{0}$). From conservativeness the existence of a basis of non-negative left annullers of the token flow matrix, C, can be deduced. For example, in Figure 13(a), p_4 is implicit as continuous if $5 \cdot \mathbf{m_0}[p_4] \geq 3 \cdot \mathbf{m_0}[p_2] + 7 \cdot \mathbf{m_0}[p_3]$. In other words, removing p_4 from the system preserves its behaviour for any (temporal) interpretation. Moreover, p_1, p_2 and p_3 form a conservative component, where the token conservation law is $5 \cdot \mathbf{m}[p_1] + 3 \cdot \mathbf{m}[p_2] + 7 \cdot \mathbf{m}[p_3] = 5 \cdot \mathbf{m_0}[p_1] + 3 \cdot \mathbf{m_0}[p_2] + 7 \cdot \mathbf{m_0}[p_3] = 55$. Thus, one of these three places is redundant (for instance, $\mathbf{m}[p_3] = (55 - 5 \cdot \mathbf{m_0}[p_1] - 3 \cdot \mathbf{m_0}[p_2])/7$).

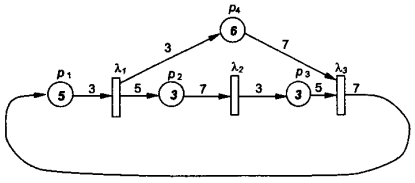
Furthermore, the cycle that results when the implicit place p_4 is removed can be transformed into an *ordinary* cycle (with unitary arc weights), by means of a linear transformation, making a reinterpretation of the net (the places, the transitions and the flow rates). If the following change is carried out in the places: $p'_1 = 5 \cdot p_1$, $p'_2 = 3 \cdot p_2$, and $p'_3 = 7 \cdot p_3$, the resulting net is an ordinary cycle (with the new places and marking interpretation). If the PN has finite servers semantics interpretation, as the firing rate does not depend on the enabling, when the previous transformation is applied, the resulting net is that in Figure 13(b)), and we should name: $\theta'_1 = \theta_1/15$, $\theta'_2 = \theta_2/21$, and $\theta'_3 = \theta_3/35$ (where θ represents times, not speeds), to have the basic cycle. It can be demonstrated that these transformations can be made in general in nets that are conservative and topologically state machines ($|{}^\bullet t| = |t^\bullet| = 1$). This property does not hold for more general subclasses, for instance, the net system in Figure 14, has a simple conservative component ($\mathbf{m}[p_1] + \mathbf{m}[p_2] + \mathbf{m}[p_3] = 3$) and can be simulated with only two state variables, but there



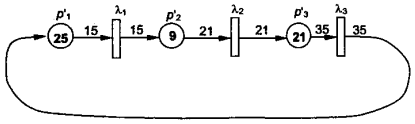
(a) (b)

Figure 12: a) FD model of the manufacturing system, b) Translation of the PN in Figure 11(a) into a FD

are weights that cannot be removed.



(a)



(b)

Figure 13: a) PN with an implicit place p_4 and a conservative component: $5 \cdot \mathbf{m}[p_1] + 3 \cdot \mathbf{m}[p_2] + 7 \cdot \mathbf{m}[p_3] = 55$, and b) Reinterpretation of the PN, after removing p_4

All these cases show that a direct translation of the PN into the FD will usually produce redundant equations. The FD in Figure 9, which corresponds to the prey/predator model, is different from the FD obtained by direct translation of the PN of Figure 3 (see Figure 15). Observe here that there are four levels but only two state variables (there exist only two independent conservation laws).

2. *Transitions vs. valves: flows.* The evolution of FDs

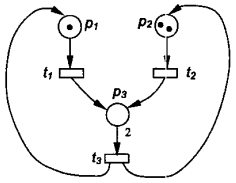


Figure 14: A non topologically state machine PN with a conservative component

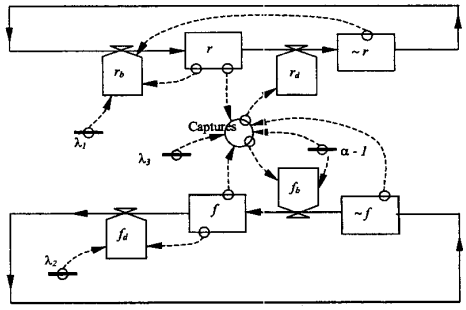


Figure 15: Translation from a PN model of a bounded prey/predator system to a FD model (Figure 3)

(the flow) takes place according to the information that valves receive of the whole system state, through the flow of information. The evolution of the PNs (the firing) takes place according to the information that each transition receives from its input places. That is, FDs *separate* the *material* and the *information* flows, and evolve according to *global* information of the system,

whereas PNs have only a flow of material that carries the information implicitly, and evolve according to information that in standard uses is *local* to each transition (its input places). Figure 16 makes explicit a hypothetical separation of material and information flows in the PN of Figure 2(b).

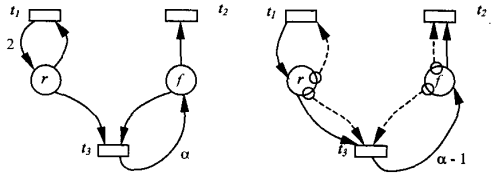


Figure 16: Hypothetical separation of flows in a PN ($\alpha > 1$)

A more detailed comparison can be made between the flows:

(a) *Material.* In FDs material is strictly conservative around the valves, that is, the relationship among input and output flow is always $1:1$, but this does not imply the existence of conservative laws over the set of levels. In this sense valves in FDs act like stations in Jackson or Gordon-Newell Queuing Networks: identity of customers is preserved when a service is provided. On the other hand, in PNs weighted conservation is frequent, even if joins and forks exist. In a general case, around a transition with n input arcs and m output arcs, $n \times m$ ratios $a_i : b_j$ will exist, with $i = 1 \ldots n$, and $j = 1 \ldots m$ (Figure 17).

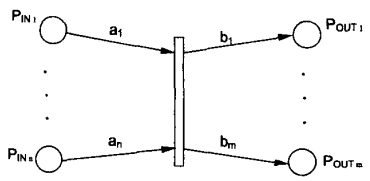


Figure 17: And-node (Join and Fork)

Synchronizations are not structurally and explicitly modelled with FDs: there exist no elements to represent "rendez-vous", and must be simulated by means of flow equations. The simulation of the PN in Figure 17 by means of FDs can be carried out in a generic way according to Figure 18(a), where a cloud and a valve are used to simulate each incoming or salient arc of the PN. It can be seen that the connectivity in the structure of the material graph is completely broken. A more efficient simulation can be carried out if a linear transformation is used in the interpretation of the levels. Thus material connectivity is possible for $min(n, m)$ channels (see Figure 18(b)); simultaneously the same quantity of valves (and clouds) is saved in the simulation.

(b) *Information.* In FDs the information affects to the dynamics of the valves in a global way, as it has been already commented. Moreover it affects in a *generic* way, since arbitrary equations (including any kind of non linearities) can be associated to the valves. However, in PNs the information affects to the dynamics (evolution) of the system not only locally, but also according to a limited number of functions: basic functions depending on the semantics (like *infinite servers* semantics, or *finite server* semantics) and additional functions that may arise for example from the decoloration of a colored net. If basic semantics are kept, some properties of the original discrete systems could subsist. But nothing prevents to define any other global firing functions, and this way the graphical tool that we obtain includes continuous PNs and FDs.

3. *On their typical application domains*

FDs have been traditionally used to model existing complex socio-technical and bio-ecological systems, whose behaviour has to be studied. Usually, those models are used for the analysis, mainly by means of simulation. There exist several computer programs to create and simulate a FD, and are friendly enough to be used even by non experts in the subject.

Continuous PNs have been mostly used in the design of "technical" systems, and much effort has been made in formal analysis techniques. The analysis has been done at two levels:

- As untimed models: if time (firing speed) is not taken into account, some properties of the autonomous PN (liveness, boundedness, reliability, etc.) can be analysed.
- As timed models performance properties have been analysed.

Hence, different application domains and approaches appear. Moreover, the analysis of these two formalisms, PNs and FDs, from the point of view of the other, leads us to appreciate some features that could go unnoticed (information flow in PNs, invariants and synchronizations in FDs ...). Methodological comparisons between the use of both formalisms for model building and the analysis of correspondences and differences can show their advantages and disadvantages, and drive to a deeper knowledge of them. The richer material structure of PNs exhibits some potentials for the analysis of the underlying untimed non-deterministic systems (for example for the analysis of qualitative or logical properties like deadlock-freeness or certain classes of mutual exclusions).

VI REFERENCES

[1] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, 8(1):159–188, 1998.

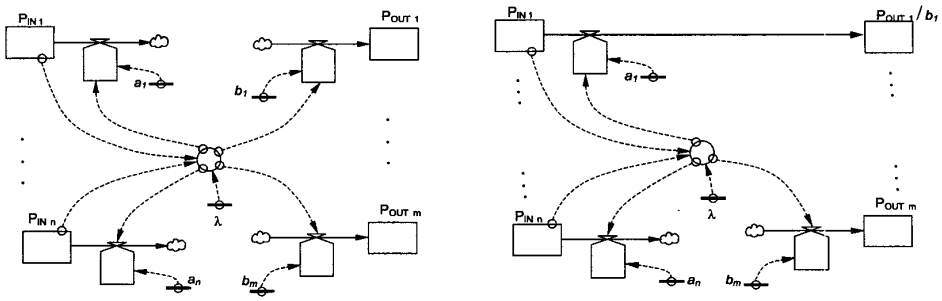


Figure 18: Generic simulation with FD of a PN transition

[2] J. Aracil. Bifurcations and structural stability in the dynamical systems modeling process. *Systems Research*, 3(4):243–252, 1986.

[3] F. E. Cellier. *Continuous System Modeling*. Springer, 1991.

[4] J. M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 113–145. Springer, 1991.

[5] R. David and H. Alla. Continuous petri nets. In *EWATPN 1987: 8th European Workshop on Application and Theory of Petri Nets*, pages 275–294, Zaragoza, 1987.

[6] C. Dutheillet, G. Franceschinis, and S. Haddad. Analysis techniques for colored well-formed systems. In G. Balbo and M. Silva, editors, *Performance Models for Discrete Event Systems with Synchronozations: Formalisms and Analysis Techniques*, chapter 7, pages 233–284. Jaca, Spain, 1998.

[7] Jay W. Forrester. *Industrial Dynamics*. MIT Press, Cambridge, Mass, 1961.

[8] Jay W. Forrester. *Principles of Systems*. Productivity Press, 1968.

[9] Jay W. Forrester. *Urban Dynamics*. Productivity Press, 1969.

[10] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods, and Practical Use*. EATCS Monographs on Theoretical Computer Science. Springer, 1994.

[11] K. Jensen and G. Rozenberg, editors. *High-level Petri Nets*. Springer, 1991.

[12] E. Mosekilde, J. Aracil, and P.M. Allen. Instabilities and chaos in nonlinear dynamic systems. *Systems Dynamics Review*, 4(1-2):14–55, 1988.

[13] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[14] L. Recalde and M. Silva. PN fluidification revisited: Semantics and steady state. In J. Zaytoon S. Engell, S. Kowalewski, editor, *ADPM 2000: 4th Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 279–286, Dortmund, 2000.

[15] L. Recalde, E. Teruel, and M. Silva. Autonomous continuous P/T systems. In J. Kleijn S. Donatelli, editor, *Application and Theory of Petri Nets 1999*, volume 1639 of *Lecture Notes in Computer Science*, pages 107–126. Springer, 1999.

[16] M. Silva. Introducing Petri nets. In *Practice of Petri Nets in Manufacturing*, pages 1–62. Chapman & Hall, 1993.

[17] M. Silva and J.M. Colom. On the structural computation of synchronic invariants in P/T nets. In *EWATPN 1987: 8th European Workshop on Application and Theory of Petri Nets*, pages 237–258, Zaragoza, 1987.

[18] M. Silva and L. Recalde. Réseaux de Petri et relaxations de l'integralité: Une vision des réseaux continus. In *Conférence Internationale Francophone d'Automatique (CIFA 2000)*, pages 37–48, Lille, 2000.

[19] M. Silva and E. Teruel. A systems theory perspective of discrete event dynamic systems: The Petri net paradigm. In P. Borne, J. C. Gentina, E. Craye, and S. El Khattabi, editors, *Symposium on Discrete Events and Manufacturing Systems. CESA '96 IMACS Multiconference*, pages 1–12, Lille, July 1996.

[20] M. Silva and E. Teruel. Petri nets for the design and operation of manufacturing systems. *European Journal of Control*, 3(3):182–199, 1997.

[21] P. S. Thiagarajan. Elementary net systems. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties. Advances in Petri Nets 1986, Part I*, volume 254 of *Lecture Notes in Computer Science*, pages 26–59. Springer, 1987.