7-th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation

Introduction to Generalized Stochastic Petri Nets

Gianfranco Balbo
Dipartimento di Informatica
Università di Torino
Italy

May 29-th, 2007

Outline

- Performance Evaluation of DEDS (Discrete Event Dynamic Systems)
  - Problem statement
  - Petri Nets
  - Timed Petri Net
  - Stochastic Petri Nets
  - Generalized Stochastic Petri Nets
  - Performance Indices
  - Practical Problems
- Case studies
- Advanced Material
  - Net-based solution technique
  - Decomposition and Aggregation
  - General distribution firing times
  - Simulation

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 2

Performance Evaluation of DEDS

Stochastic Petri Nets are a convenient formalism for the representation and evaluation of Discrete Event Dynamic Systems (DEDS)

DEDS are characterized by
- discrete (countable) state space
- Events

DEDS can be considered as views of dynamic systems such as
- flexible Manufacturing Systems
- transport Systems
- organization Systems
- distributed Systems
- telecommunication Systems
- ......

Common to all these systems is the presence of
- Concurrency
- Cooperation
- Competition

(queueing, service, routing, and synchronization)

Modelling DEDS Systems

Modelling plays an important role during the life-cycle of DEDS that includes the following critical issues
- correctness analysis
- performance evaluation
- reliability evaluation
- design optimization
- scheduling (performance control)
- monitoring and supervision
- implementation
- ......

The complexity of the interplaying among DEDS components suggests to consider the time-evolutions of DEDSs as Stochastic Processes that can be used to assess their efficiency and reliability.

Performance Indices (Transient Analysis)

The distribution of the stochastic process representing the time-evolution of a DEDS system at a certain given time is usually the basis for the quantitative evaluation of the behaviour of the system

Often the transient analysis of these systems is mathematically very complex and simulation becomes the only viable technique

Performance indices of interest are

- Probability of reaching particular states
- Probability of satisfying assigned deadlines

Performance Indices (Steady State Analysis)

The stationary distribution of the stochastic process representing the time-evolution of a DEDS system is usually the basis for the quantitative evaluation of the behaviour of the system expressed in terms of *performance indices*

Performance indices can be computed using a unifying approach in which proper index functions (also called *reward functions*) are defined over the states of the stochastic process and an *expected reward* is derived using the stationary distribution of the process

Performance indices of interest are

- Probability of specific state conditions
- Resource utilizations
- Expected flows (throughputs)
- Expected numbers of active resources (or clients)
- Expected waiting times

Petri Nets

Petri nets are abstract formal models of information flow

They have been developed in search for natural, simple, and powerful methods for describing and analyzing the flow of information and control in systems

Petri nets are well suited for the representation of systems in which activities may take place concurrently, under precedence or frequency constraints

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 7

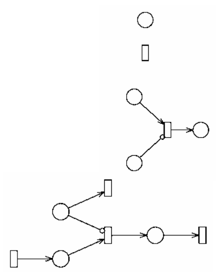
Petri Nets: Definition, Notation and Rules

Petri nets are bipartited directed graphs

NODES — Places, Transitions

ARCS — Input, Output, Inhibition

A PETRI NET



- A marking *M* is an assignment of tokens to places
- A transition is *enabled* if at least one token exists in each of its *input* places, and no tokens exist in its *inhibition* places
- A transition *may fire* if it is *enabled*
- A Petri nets *executes* by *firing* transitions
- A transition *fires* by *removing* tokens from each of its input places and *depositin*g tokens in each of its output places
- Dynamic properties of Petri nets result from their *execution* controlled by the position and movement of tokens

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 8

Petri Nets: Formal Definition

A marked Petri net is formally defined by the following tuple

$$PN = (P, T, F, W, M_0)$$

where

$P = (p_1, p_2, ..., p_P)$	is the set of places
$T = (t_1, t_2, ..., t_T)$	is the set of transitions
$F \subseteq (P \times T) \cup (T \times P)$	is the set of arcs
$W : F \rightarrow (1, 2, ...)$	is a weight function
$M_0 = (m_{01}, m_{02}, ..., m_{0P})$	is the initial marking

Combining the information provided by the flow realtions and by the weight function, we obtain the *Incidence Matrix*

transitions

$$\mathbf{C} = \text{places} \; \boxed{c_{pt}}$$

with $c_{pt} = c_{pt}^{+} + c_{pt}^{-} = w(t,p) - w(p,t)$

Petri Nets: Basic Definitions

$RS(M_0)$	Set of markings reachable from M_0
$E(M)$	Set of transitions enabled in marking M
$M \xrightarrow{s} M'$	M' is reachable from M by firing a sequence S of transitions

a transitions t_r is enabled in marking M iff

$$M \geq [C_r^-]^T$$

$$M \xrightarrow{t_r} M' \equiv M - [C_r^-]^T + [C_r^+]^T = M'$$

a marking M' is said to be a *home state* iff

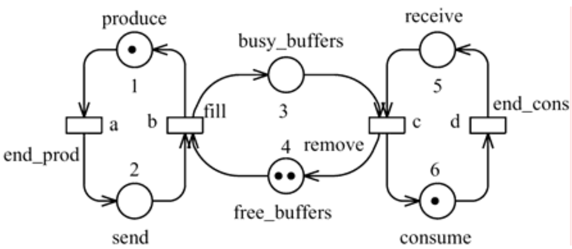
$$\forall M \in RS(M_0),\ \exists s \ : \ M \xrightarrow{s} M'$$

a transition t_r is said to be in conflict with transition t_s in marking M iff

$$t_r, t_s \in E(M); \qquad M \xrightarrow{t_s} M'; \qquad t_r \notin E(M')$$

Petri Nets: Simple example – Producer/Consumer

Petri net model:



Set of places: $P = (p_1, p_2, p_3, p_4, p_5, p_6)$

Set of transitions: $T = (a, b, c, d)$

Incidence matrix:

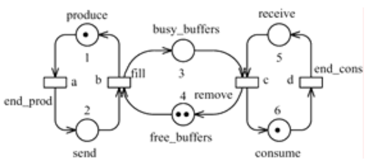
C =

	a	b	c	d
1	-1	+1		
2	+1	-1		
3		+1	-1	
4		-1	+1	
5			-1	+1
6			+1	-1

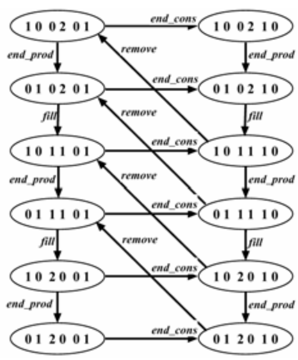
Initial marking: $M_0 = (1, 0, 0, 2, 0, 1)$

Petri Nets: Simple example – Producer/Consumer

Petri net model:



Reachability graph:



Petri Nets: Structural and Behavioural Properties

Structural **properties of Petri nets are obtained from the incidence matrix, independently of the initial marking**

Behavioural **properties of Petri nets depend on the initial marking and are obtained from the reachability graph (finite case) of the net or from the covering tree (infinite case)**

Petri Nets: *P* Semiflows

A Petri net is ***strictly conservative*** **(or strictly invariant) iff**

$$\sum_{p=1}^{P} m_p = \sum_{p=1}^{P} m_{0p}, \qquad \forall M \in RS(M_0)$$

A Petri net is ***conservative*** **(or** ***P*** **invariant) iff**

$$\exists\, Y = (y_1,\ y_2,\ \ldots,\ y_P) > 0 \ such\ that$$

$$\sum_{p=1}^{P} y_p m_p = \sum_{p=1}^{P} y_p m_{0p} \qquad \forall M \in RS(M_0)$$

from this relation it follows that

$$M \xrightarrow{t_r} M' \equiv M' = M + [C_r]^T$$

$$\Rightarrow Y[M']^T = Y[M]^T + Y[C_r]$$

The integer solution ***Y*** **of the equation**

$$YC = 0$$

is called a ***P Semiflow***

Petri Nets: *T* Semiflows

Let $V = (v_1, v_2, ..., v_T)^T$ be the transition count vector associated with a firing sequence *S*

$$M \xrightarrow{t_r} M' \equiv M' = M + [C_r]^T$$
$$M \xrightarrow{s} M" \equiv M" = M + [CV]^T$$

The integer solution *X* of the equation

$$CX = 0$$

is called a *T-Semiflow*

A net covered by *T*-semiflows
may have *home states*

A net with home states is covered by *T*-semiflows

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 15

Petri Nets: Reversibility

A marking M_h is called a *home-state* iff

$$\forall M \in RS(M_0), \quad M_h \in RS(M)$$

The set of the home-states of a Petri net is called its *home-space*

A Petri net is *reversible* whenever its initial marking M_0 is a home-state

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 16

Petri Nets: Boundedness

A place p_i is *bounded* (*k*-bounded) iff

$$\forall M \in RS(M_0), \quad \exists k : m_i \leq k$$

A Petri net is *bounded* (*k*-bounded) iff

$$\exists k : (\forall p_i \in P : is\ k - bounded)$$

A net covered by *P*-semiflows Is bounded

Petri Nets: Liveness

A transition t_r is *live* iff

$$\forall M \in RS(M_0), \quad \exists M' : (M \xrightarrow{s} M' \bigwedge t_r \in E(M'))$$

A Petri Net is *live* iff

$$\forall t_r \in T : t_r\ is\ live$$

A marking *M* is *live* iff

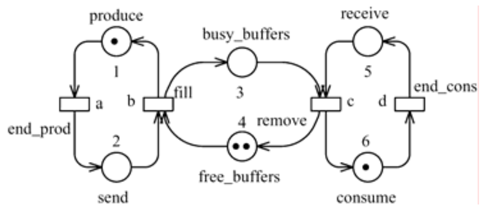
$$\forall t_r \in T, \quad \exists M' : (M \xrightarrow{s} M' \bigwedge t_r \in E(M'))$$

A Petri Net is *live* iff

$$\forall M \in RS(M_0) : M\ is\ live$$

Petri Nets: Simple example – Producer/Consumer

Petri net model:



Incidence matrix:

C =

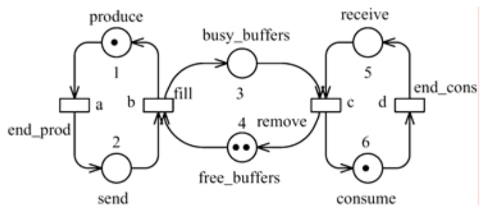
	a	b	c	d
1	-1	+1		
2	+1	-1		
3		+1	-1	
4		-1	+1	
5			-1	+1
6			+1	-1

***P* semiflows (*YC* = 0):** *y* = (1, 1, 0, 0, 0, 0)
y = (0, 0, 1, 1, 0, 0)
y = (0, 0, 0, 0, 1, 1)

***T* semiflows (*CX* = 0):** *x* = (1, 1, 1, 1)

Petri Nets: Simple example – Producer/Consumer

Petri net model:



***P* semiflows (*YC* = 0):** *y* = (1, 1, 0, 0, 0, 0)
y = (0, 0, 1, 1, 0, 0)
y = (0, 0, 0, 0, 1, 1)

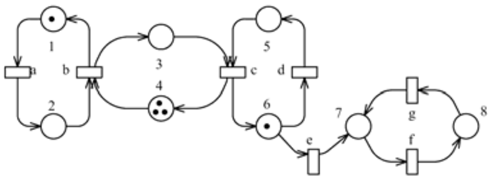
The net is covered by *P*-semiflows, thus is bounded

***T* semiflows (*CX* = 0):** *x* = (1, 1, 1, 1)

The net is covered by *T*-semiflows, this is necessary for liveness

Petri Nets: Simple example – Producer/Consumer

Petri net model:



Incidence matrix:

$C =$

	a	b	c	d	e	f	g
1	-1	+1					
2	+1	-1					
3		+1	-1				
4		-1	+1				
5			-1	+1			
6			+1	-1	-1		
7					+1	-1	+1
8						+1	-1

***P* semiflows** ***(YC = 0)*:** $y = (1, 1, 0, 0, 0, 0, 0, 0)$
$y = (0, 0, 1, 1, 0, 0, 0, 0)$
$y = (0, 0, 0, 0, 1, 1, 1, 1)$

***T* semiflows** ***(CX = 0)*:** $x = (1, 1, 1, 1, 0, 0, 0)$
$x = (0, 0, 0, 0, 0, 1, 1)$

Time and Petri Nets

Timing Specifications

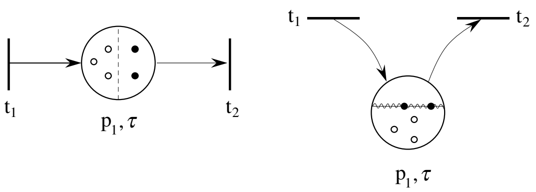
- Introducing time = interpretation of the model
  - observation of the autonomous (untimed) model
  - definition of the non-autonomous model
- Time specifications could/should provide
  - consistency of behaviour of autonomous and non-autonomous model
  - reduction of non-determinism on the basis of time
  - support for the computation of performance indices
  - support for timed properties verification

Timing Specifications

- time is associated to places
- time is associated to tokens
- time is associated to arcs
- time is associated to transitions

Timed places

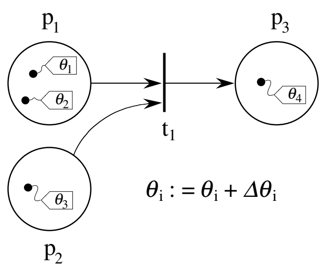
- Tokens generated in the output places become available to fire a transition only after a delay has elapsed.
- Delay is an attribute of the place



- Consequences........

Timed tokens

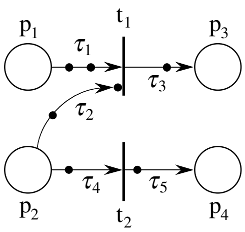
- Tokens carry a time stamp that indicates when they are available to fire a transitions.
- Firing changes the timestamp



- Consequences........

Timed arcs

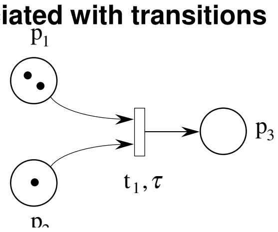
- A travelling delay is associated with each arc
- Tokens are available for firing only when they reach the transition



- Consequences........

Timed transitions (1)

- Time is associated to transitions, that represents "activities"
  - activity start corresponds to enabling
  - activity end corresponds to firing
- Delay is associated with transitions



Timed transitions (2)

- Different *firing policy* are possible:
- Three-phase policy:
  - on enabling tokens are removed from input places
  - delay elapses
  - on firing tokens are generated in the output places
- Atomic firing policy
  - tokens remains in the input places according to the transition delay
  - at firing tokens are removed from input places and tokens are generated in the output places
- Consequences........

From now on TTPN......

- A "time specification" is associated with transitions
- Atomic firing policy is assumed
- TTPN *can* preserve the behaviour (marking space) of the autonomous model
- Logical properties computed on the autonomous model can carry on to the interpreted model
- Why only *can* ?

From now on TTPN……

What is a time specification?

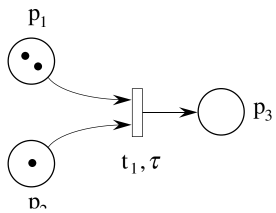
1. A value
2. An interval (min-max pair)
3. A probability distribution

How it is used?

1. A value → fixed delay
2. An interval → a non deterministic value in the interval
3. A probability distribution → value is extracted from a distribution

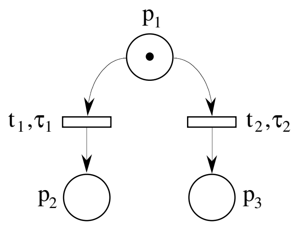
Timed transition behaviour

- a timer is associated with each transition
  - when the transition is enabled the timer is set to a *valid* transition delay value
  - timers are decremented at constant speed
  - when the timer reaches zero the transition fires (*or can fire)*



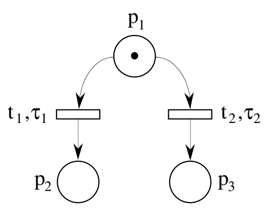
Timed transition behaviour: conflict

- If more than one timed transition is enabled the behaviour is similar, but which transition is going to fire?



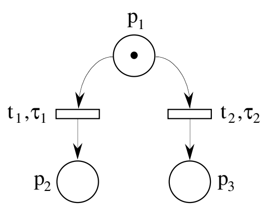
Timed transition behaviour: conflict

- two selection rules
  - *preselection:* the enabled transition that will fire is chosen first, without using the timing specification (according to some metrics, or non deterministically)
  - *race:* the enabled transition that will fire is the one whose timer goes to zero first



Timed transition behaviour: memory policy

- When a timed transition is disabled by a conflicting transition, what happens to the timer of the disabled transition?
- When there is a change of marking what happens to the timer of the transitions that have kept their enabling?
  - ***How does the transition keep track of its past enabling time?***



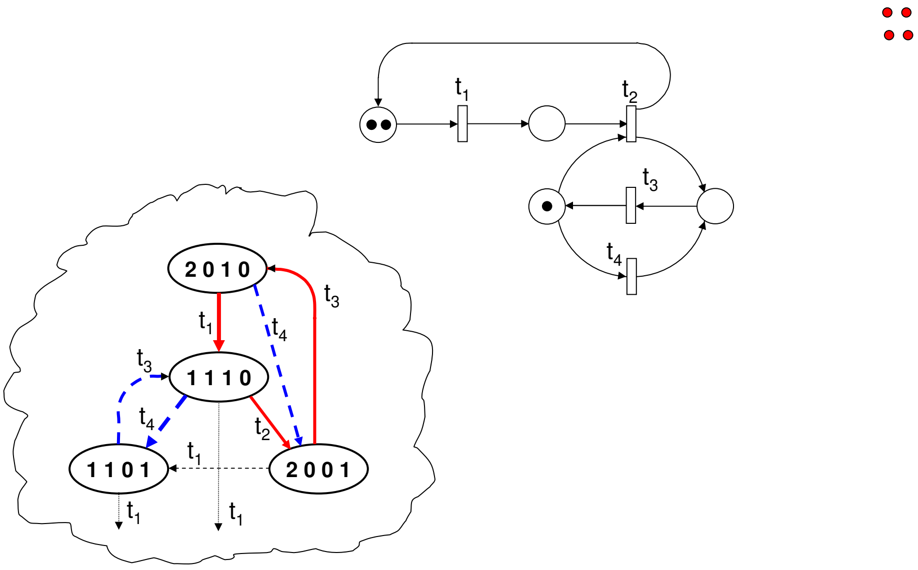
Basic memory mechanisms

- ***Continue:*** the timer value is kept
- ***Restart:*** the timer of the transition is restarted
  - **And these actions can take place depending on whether there was a change of marking that has changed the enabling of the transition or not, giving rise to a few *memory policies***

Transition memory policy: resampling

- At each and every transition firing the timers of all timed transitions are discarded (*restart mechanism)*
- No memory of the past is recorded - regeneration points
- In the new marking a new value of the timer is associated with each enabled transition

Transition memory policy: resampling

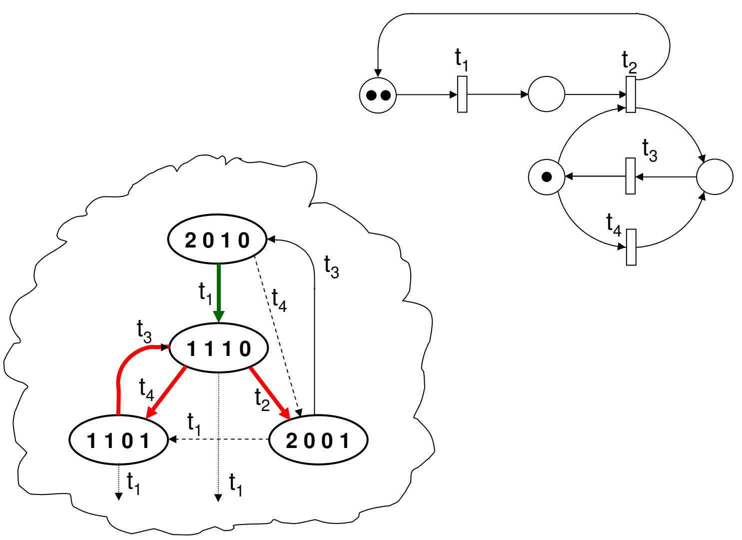


Transition memory policy: enabling memory

- At each and every transition firing:
  - the value of the timer of all timed transitions that are disabled in the new marking are discarded (*restart mechanism)*
  - the value of the timer of all timed transitions that are still enabled in the new marking are kept (*continue mechanism)*
- The memory of the past is recorded into an ***enabling memory variable*** associated with the transition
- Modelling viewpoint: used when the activity looses the work done if it is interrupted (disabled by another transition)

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 39

Transition memory policy: enabling memory



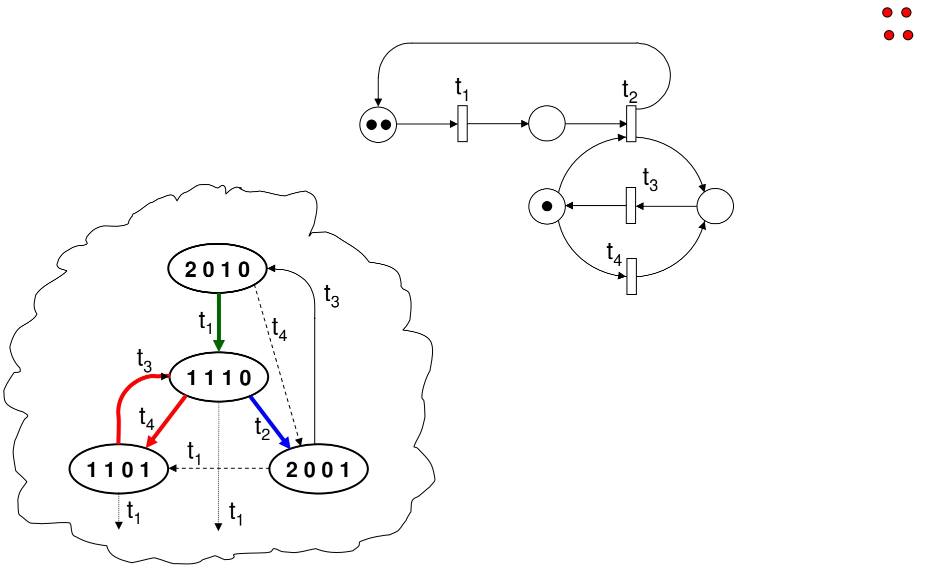
SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 40

Transition memory policy: age memory

- At each and every transition firing:
  - the value of the timer of all timed transitions is kept at each transition firing (*continue mechanism)*
- The memory of the past is recorded into an *age memory variable* associated with the transition
- Modelling viewpoint: used when the activity does not loose the work done if it is interrupted (disabled by another transition)

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 41

Transition memory policy: age memory



SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 42

Transition memory policies

- Resampling:
  - locality is lost (the firing of a non conflicting transition has an impact)
  - no conservation of work in any case
- Enabling
  - locality is kept
  - conservation of work only if no race was present
- Age
  - locality is kept
  - work is always kept

Transition enabling

- The *enabling degree* of a transition is the number of times the transition could fire in a given marking before becoming disabled -- $E(t, M)$
- If $E(t, M) > 1$ then, is a single timer enough?
- It depends on the *server semantics:*
  - single server
  - multiple server (*k*-server)
  - infinite server

Transition enabling: single server

- **Single server == one at a time == sequential processing == self-conflict**
- **A single timer per transition**
- **If *E(t, M) > 1* when the timer of *t* goes to zero the transition fires, a new marking *M'* is reached with *E(t,M')≥1* and the timer of *t* is initialized again**

Transition enabling: infinite server

- **Infinite server == all at the same time == parallel processing == self-concurrency**
- **An infinite number of timers if the net is not bounded**
- **When in a new marking the enabling degree of *t* is incremented a new timer is initialized,**
- **When it is decremented what happens depends on the memory policy and on the choice of the "server" to preempt**

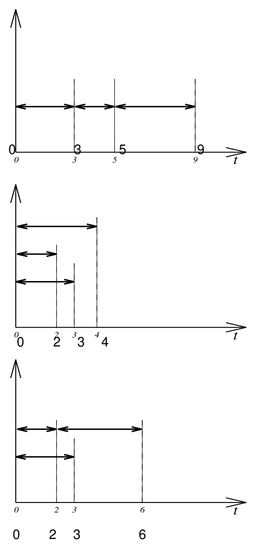
Transition enabling: multiple server

- Multiple *k* server == at most *k* at the same time == limited parallel processing == both self-concurrency and self-conflict
- *k* timers per transition
- Up to an enabling degree ≤ *k* same behaviour as infinite server

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 47

Transition enabling: example

- E(t, M) = 3
- The three "enablings" are associated with firing delay 3, 2 and 4 (in the arrival order)



SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 48

Server semantics and memory policy for nets

- The memory policies (age, enabling, resampling) and server policy (single, multiple, infinite) can be defined separately for each transition of a net
- Attention should be paid when combining multiple or infinite server with preemptive policies (which timer is blocked/reset?)
- Related to the notion of queueing discipline of a transition

Chosing a firing delay

- fixed value: every time the timer of *t* is set, the same value is used -- what happen with conflicts?
- finite/infinite interval: every time the timer of *t* is set, a value from the interval is chosen -- what happen with conflicts?

Chosing a firing delay

- HOW is the value chosen?
  - non deterministically -- *not considered any longer in this tutorial*
  - according to a probability distribution, the delay associated to transition *t* is a random variable -- *main topic of this tutorial* -- defines a stochastic process and leads to performance evaluation

Stochastic (Exponential) Petri Nets

Stochastic (Exponential) Petri Nets

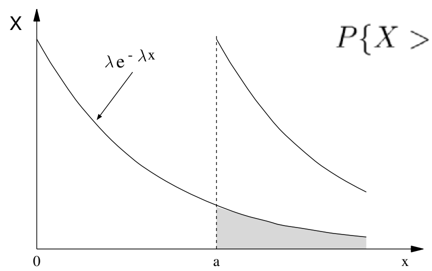
- The delay of a transition is a random variable
- Timed Transition PN with atomic firing and race policy in which transition delays are random variables *exponentially* distributed are called Stochastic Petri Nets (SPN)
- SPN is the name chosen by Molloy in 1982, but a more adequate one could be Exponential Petri Nets

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 53

Exponential distributions

- The exponential pdf is $f_X(x) = \lambda e^{-\lambda x}$ $(x \geq 0)$

it is the only continous distribution for which the ***memoryless property*** holds



$$P\{X > x + \alpha | X > \alpha\} = P\{X > x\}$$

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 54

Exponential distributions

- **The exponential pdf**

$$f_X(x) = \lambda e^{-\lambda x} \qquad (x \geq 0)$$

is defined only by its rate λ, which is the inverse of the average value

$$E[X] = \frac{1}{\lambda}$$

Why Exponential distributions

- **Given two random variables X and Y with exponential pdf**

$$f_X(x) = \lambda e^{-\lambda x} \qquad (x \geq 0)$$
$$f_Y(y) = \mu e^{-\mu y} \qquad (y \geq 0)$$

the new random variable Z = min(X,Y) has *also* an exp. pdf

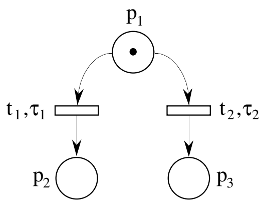
$$f_Z(z) = (\lambda + \mu) e^{-(\lambda+\mu)z} \qquad (z \geq 0)$$

since

$$\begin{aligned} F_Z(z) &= 1 - Pr\{Z > z\} \\ &= 1 - Pr\{X > z, Y > z\} \\ &= 1 - e^{-\lambda z} e^{-\mu z} = 1 - e^{-(\lambda+\mu)z} \end{aligned}$$

Why Exponential distributions

- If X is the random variable for t_1 and
  Y is the random variable for t_2



if the race policy is assumed, then the random variable that describes:

how long the system stays in marking $1 \cdot p_1$,

is defined as $Z = min(X,Y)$

Why Exponential distributions

Moreover the distinction between *continue* and *restart* is irrelevant, since the residual time of the timers has the same distribution as the original assignment

Markov chains

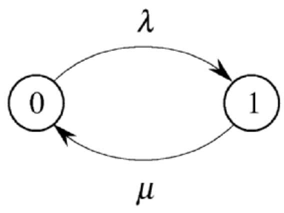
- **Continuous Time Markov chain - CTMC**
  is a simple type of stochastic process with discrete state space
  - **Sojourn times in states are exponentially distributed random variables**

and

  - **future evolution only depends on the present state (no need to keep history information)**

Markov chains

- **CTMCs can be described as automata with labelled transitions; the value of the label describes the rate associated to that change of state**



$$Q = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}$$

State transition rate diagram **Infinitesimal generator**

Markov chains

- If the CTMC is ergodic, the CTMC solution amounts to the computation of the solution of a set of linear equations (as many equations as there are states in the CTMC)
- The solution vector gives the probability of being in each single state of the chain in equilibrium
- Solution at time T is also possible

SPN definition

- An SPN is defined as a 7-tuple

  SPN= (P, T, I(.), O(.), H(.), W(.), M0)

where

- PN = (P, T, I(.), O(.), H(.), M0) is the P/T system underlying the SPN
- Transitions have an exponentially distributed delay
- W(.): T --> $\mathcal{R}$ assigns a rate to each transition

  (inverse of the mean firing time)

SPN definition

- The stochastic process underlying an SPN is a CTMC in which
  - the state transition rate diagram is isomorphic to the reachability graph
  - the transition labels are computed from the W(.) functions of the transitions enabled in a state

- Let's start with two simple cases:
  - SPN w/o choices and synchronization
  - SPN with choices

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 63

SPN w/o synchronization and choices

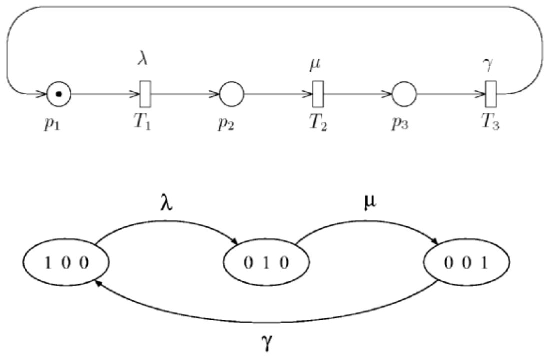
- If we also require that the initial marking is a single place marked with a token, then the SPN is both a Marked Graph (*no place has more than one input and one output transition – no choices*) and a Finite State Machine (*no transition has more than one input – no synchronizations - and one output place*)

- Each place univocally identifies a state of the net (and a state of the CTMC)

- The time spent by the system in a state is determined by the characteristics of the single transition enabled in that state

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 64

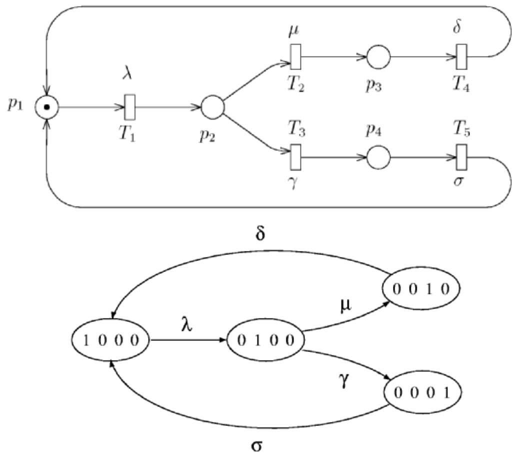
SPN w/o synchronization and choices



SPN with choices

- If we also require that the initial marking is a single place marked with a token, then the SPN is a Finite State Machine
- Race is possible (there can be conflicts)
- The time spent by the system in a state is the minimum among the delays of the enabled transitions

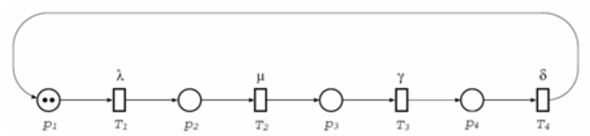
SPN with choices

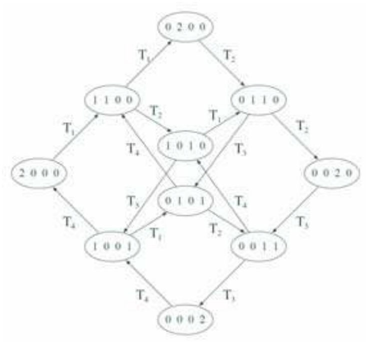


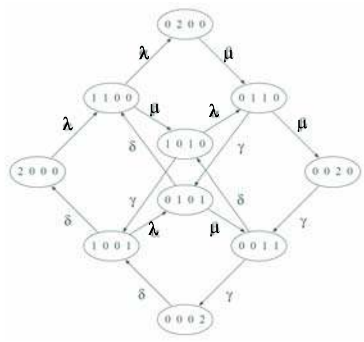
SPN with (self) concurrency

- **If there can be more then one token in the initial marking, then it is necessary to consider:**
  - **transitions that are enabled in the same state, but that are not in conflict**
  - **transitions with k- and infinite server semantics**

SPN with (self) concurrency







Probabilistic characterization

Let

σ_j be the **sojourn** time in marking m_j

and

δ_k the **firing** delay of transition t_k

The Markov property says

$$Pr\{X_{n+1} = m_j, \sigma_j > t | (X_n = m_i, \sigma_i = u), (X_{n-1} = m_k, \sigma_k = v), \ldots\}$$
$$= Pr\{X_{n+1} = m_j, \sigma_j > t | X_n = m_i\}$$

where

$$Pr\{\sigma_j > t\} = Pr\left\{ \bigcap_{k \in E(m_j)} \delta_k > t \right\}$$

Markov dependency

Let

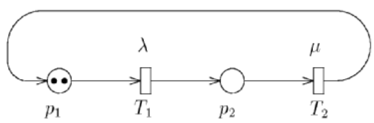
$E(m_j) = NE(m_j) \cup OE(m_j)$

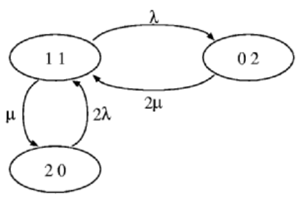
where

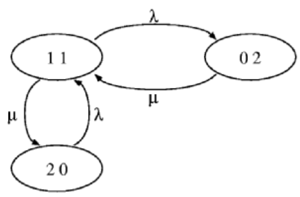
$NE(m_j)$ **Set of transitions newly enabled in** m_j
$OE(m_j)$ **Set of transitions already enabled in previous marking**

$$
\begin{aligned}
& Pr\{X_{n+1} = \mathbf{m}_j, \sigma_j > t | X_n = \mathbf{m}_i, \sigma_i > u\} = \\
&= \frac{Pr\{X_{n+1} = \mathbf{m}_j, \sigma_j > t, \sigma_i > u | X_n = \mathbf{m}_i\}}{\Pr\{\sigma_i > u\}} \\
&= \frac{Pr\left\{X_{n+1} = \mathbf{m}_j, \left(\bigcap_{k \in NE(\mathbf{m}_j)} \delta_k > t\right), \left(\bigcap_{h \in OE(\mathbf{m}_j)} \delta_h > (t+u)\right) | X_n = \mathbf{m}_i\right\}}{Pr\left\{\bigcap_{h \in OE(\mathbf{m}_j)} \delta_h > u\right\}} \\
&= Pr\left\{X_{n+1} = \mathbf{m}_j, \left(\bigcap_{k \in E(\mathbf{m}_j)} \delta_k > t\right) | X_n = \mathbf{m}_i\right\} \qquad (15)
\end{aligned}
$$

SPN with (self) concurrency







Queueing policy

If only exponential distributions are present

and

if the performance figures of interest are based on the moments of the number of tokens in places

then

many queueing policy yield the same results, and random order can be safely assumed

CTMC construction

In general, the CTMC associated with a given SPN system is obtained by applying the following simple rules:

1. The CTMC state space $S = \{s_i\}$ corresponds to the reachability set $RS(m_0)$ of the PN associated with the SPN $(m_i \leftrightarrow s_i)$.
2. The transition rate from state s_i (corresponding to marking m_i) to state s_j (m_j) is obtained as the sum of the firing rates of the transitions that are enabled in m_i and whose firings generate marking m_j.

CTMC construction

Assuming that all the transitions of the net operate with a single-server semantics and marking-independent speeds, and denoting with

- **Q** the *infinitesimal generator*,
- w_k the firing rate of T_k,
- $e_j(m_i) = \{h : T_h \in e(m_i) \wedge m_i[T_h\rangle m_j\}$ the set of transitions that bring the net from m_i to m_j,

the components of **Q** are:

$$q_{ij} = \begin{cases} \sum_{T_k \in e_j(m_i)} w_k & i \neq j \\ -q_i & i = j \end{cases}$$

where

$$q_i = \sum_{T_k \in e(m_i)} w_k$$

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 75

Performance indices

- **The steady state distribution vector π(M) is the basis for the computation of the performance indices, together with the reward function *r(M).***
- **An average reward for an SPN N is derived as**

$$\sum_{M \in RS(N)} r(M)\pi(M)$$

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 76

Performance indices - examples

- Probability of a condition *C(M)*
  - *r(M)* = 1 if *C(M)* holds
  - *r(M)* = 0 otherwise
- Mean number of tokens in place *p*
  - if *M(p)* is the number of tokens in *p*, then

$$\sum_{M \in RS(N)} M(p)\pi(M)$$

  - therefore *r(M)=M(p)* is the right reward function

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 77

Performance indices - examples

- Mean number of firing of transition *t* per unit time
  - if *W(t),* or *W(t,M)* is the firing rate of *t* in *M*, then

$$\sum_{M \in RS(N)} W(t)\pi(M)$$

assuming that *W(t,M)=0* if *t* is not enabled in *M*

  - therefore
    - *r(M)=W(t)* if *t* is enabled in *M* and
    - *r(M) = 0* otherwise

    is the right reward function

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 78

Performance indices - examples

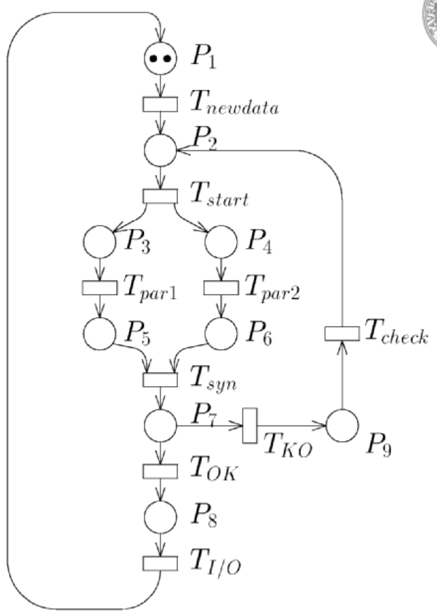
- **The average steady state delay spent in traversing a subnetwork has to be computed using Little's formula**

$$E[T] = \frac{E[N]}{E[S]}$$

where *E[N]* is the average number of (equivalent) tokens in the subnet, and *E[S]* is the average input rate into the network

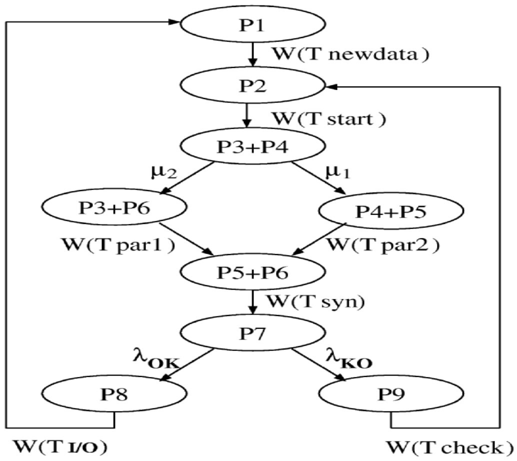
SPN example

- **The SPN description of a simple parallel computation system**



SPN example (1)

The reachability graph for M(P1) = 1



SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 81

SPN example (2)

Total rate out of P3 + P4 is:

$$W(T_{par1}) + W(T_{par2})$$

With what probability T_{par1} is the first to fire?

$$\frac{W(T_{par1})}{W(T_{par1}) + W(T_{par2})}$$

Therefore:

$$\begin{aligned} \mu_1 &= (W(T_{par1}) + W(T_{par2}))\frac{W(T_{par1})}{W(T_{par1}) + W(T_{par2})} \\ &= W(T_{par1}) \end{aligned}$$

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 82

SPN example (3)

- **Computation of success and failure rate (same as before)**

$$\lambda_{OK} = W(T_{OK})$$

$$\lambda_{KO} = W(T_{KO})$$

but what is the meaning of W(Tok) or W(Tko)?

check activity: 0.0001

$\rightarrow$ **rate of** $10,000$

$\rightarrow$ **probability of** OK/KO **is** 99% **vs.** 1%

$\rightarrow$ $W(T_{OK}) = 9,900$ **and** $W(T_{KO}) = 100$

SPN example (4)

transition	rate	value	semantics
$T_{newdata}$	λ	1	infinite-server
T_{start}	τ	1000	single-server
T_{par1}	μ_1	10	single-server
T_{par2}	μ_2	5	single-server
T_{syn}	σ	2500	single-server
T_{OK}	α	9900	single-server
T_{KO}	β	100	single-server
$T_{I/O}$	ν	25	single-server
T_{check}	θ	0.5	single-server

Consistency check operation has 0.0001 time unit duration and has 99% success probability

SPN example (5)

- Throughput of transition Ti/o:
  - 1.504 success per time unit
- Average number of items under test:
  - 0.031
- Average production time:
  - 0.33 time units

Generalized Stochastic Petri Nets GSPN

GSPN definition -- immediate

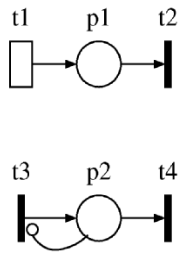
- Two types of transitions
  - timed with an exponentially distributed delay
  - immediate, with constant zero delay
  - immediate have priority over timed
- Why immediate transitions:
  - to account for instantaneous actions (typically choices)
  - to implement logical actions (e.g. emptying a place)
  - to account for large time scale differences (e.g. bus arbitration vs. I/O accesses)

GSPN definition -- immediate

- A priority function is associated to transitions (pri:T --> $\mathcal{N}$)
- Timed transitions have priority 0
- Immediate transition have priority > 0
- The autonomous model is a P/T net with global priorities
- Concession vs. enabling
  - *t* has concession in M *iff* $M \geq I(t)$ and $M > H(t)$
  - *t* is enabled in M *iff*
    it has concession and
    $\forall\, t' \in T$, if t' has concession in M, then $pri(t) \geq pri(t')$

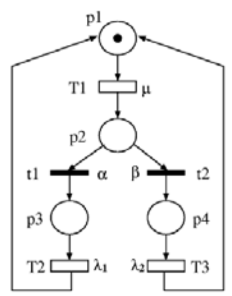
GSPN definition - immediate

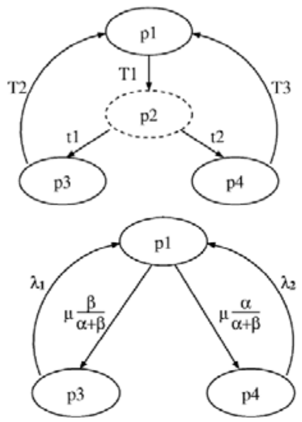
- **RS(priority removed) $\supseteq$ RS(GSPN)**
- **Safety (invariant) properties are maintained:**
  - **absence of deadlock, boundedness, mutual exclusion**
- **Eventuality (progress) properties are not maintained**
  - **reachability, liveness**



GSPN definition - vanishing states

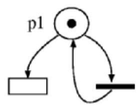
- **Tangible marking (if it enables *only* timed transitions)**
- **Vanishing marking (if it enables *only* immediate transitions)**





GSPN definition - attention

- Need for priority



- Any memory policy, any queueing policy
- No timers can expire at the same time
  (probability of extracting a specific sample is equal to zero)

GSPN definition

- A GSPN is defined as an 8-tuple
  SPN= (P, T, pri(.), I(.), O(.), H(.), W(.), M0)

where

- *PN = (P, T, pri(.), I(.), O(.), H(.), M0)* is the *P/T* system *with priority* underlying the SPN
- Transitions have an exponentially distributed delay
- *W(.): T* --> $\mathcal{R}$

and the net must be confusion free

GSPN definition - rates and weights

- ***W(t)* is called:**
  - ***rate,* if *t* is timed**
  - **weight, if *t* is immediate**

- **Rates define the distribution of the delay associated with *t***

- **Weights are used for the probabilistic resolution of conflicts of immediate**

GSPN definition - behaviour

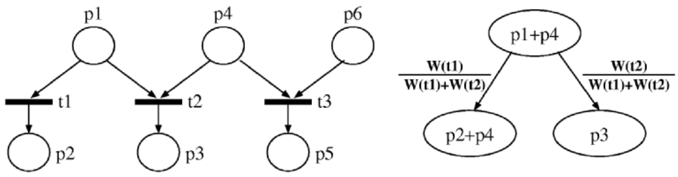
- **Behaviour in tangible states is as in SPN**

- **Behaviour in vanishing states:**

  When a vanishing marking is entered the weights of the *n*-ime mediate enabled transitions are used to probabilistically select the transition to fire

  The time spent in the marking is deterministically equal to zero

GSPN definition - normalization



$W(t_1) = 10$ $W(t_2) = 20$ $W(t_3) = 44$

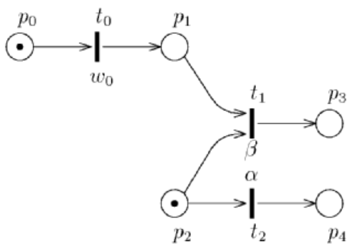
- The weight of t_2 with respect to t_1 is always the same, regardless of whether t_3 is enabled or not.
- Extensions to marking dependent rates have been defined.

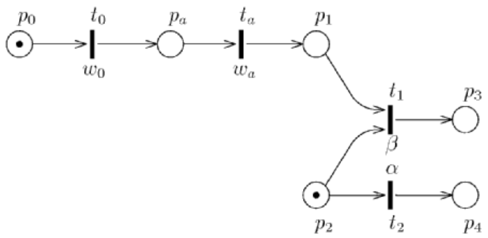
GSPN definition - ECS

- ***Extended conflict set* - ECS: identification of immediate transitions that are enabled in conflict**
- **Probability normalization can be done in within ECS if the GSPN is *confusion-free***

GSPN definition - confusion

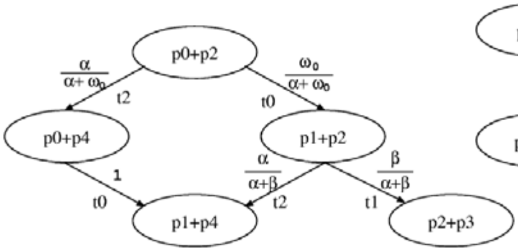
- **Confusion destroys the locality of conflicts**

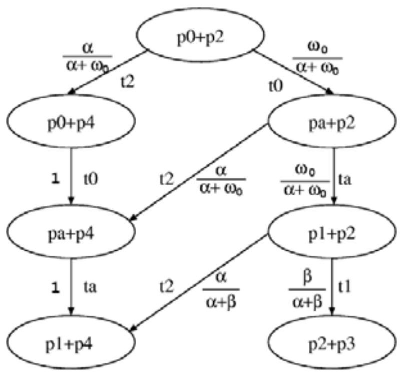




GSPN definition - confusion

- **Confusion produces different transitions probabilities**





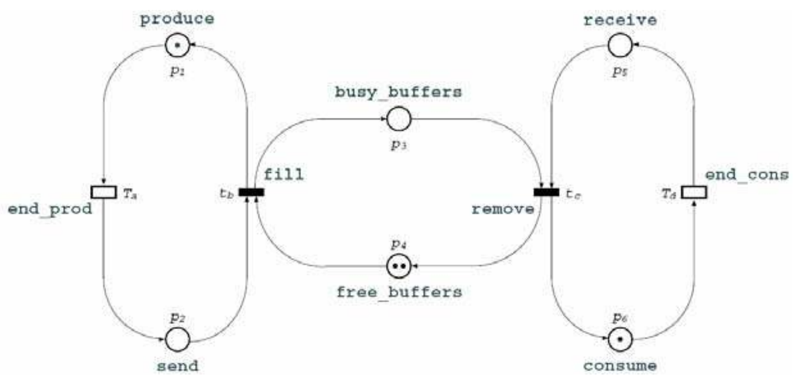
GSPN definition - prob. of firing

- **Probability of firing *t* in a marking M is**

$$P\{t \mid M\} = W(t) / W_{ECS(t)}(M)$$

where $W_{ECS(t)}(M) = \sum_{t' \in ECS(t) \cap E(M)} W(t')$

GSPN model of Producer / Consumer



Formal specification of the Producer / Consumer GSPN

Set of places: $P = (p_1, p_2, p_3, p_4, p_5, p_6)$

Set of transitions: $T = (T_a, t_b, t_c, T_d)$

Priorities: $\Pi = (\pi_0, \pi_1, \pi_1, \pi_0)$

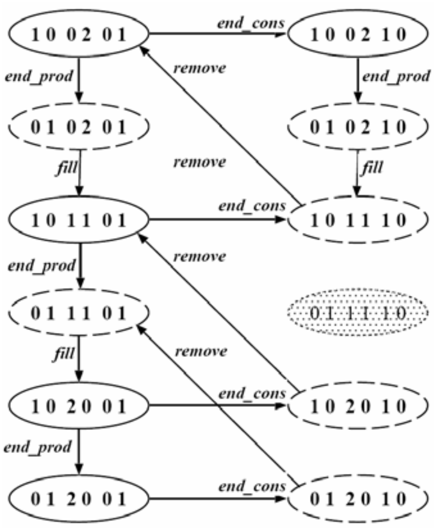
Weights: $W = (a, 1, 1, b)$

Incidence matrix: $C =$

	a	b	c	d
1	−1	+1		
2	+1	−1		
3		+1	−1	
4		−1	+1	
5			−1	+1
6			+1	−1

Initial marking: $\boldsymbol{m}_0 = (1, 0, 0, 2, 0, 1)$

Reachability Graph of the Producer / Consumer GSPN



GSPN solution

- GSPNs are isomorphic to semi-Markov processes
- The analysis can be performed on
  - a reduced Embedded CTMC defined on the set of tangible states or
  - reducing the GSPN to an equivalent SPN

Construction of the Embedded CTMC (EMC)

Recall

RS	Reachability Set
TS	Set of Tangible States
VS	Set of Vanishing States

$$RS = TS \bigcup VS, \quad TS \bigcap VS = \emptyset.$$

$U = [u_{ij}]$ Transition probability matrix of the Embedded Markov Chain

$$u_{ij} = \frac{\sum_{T_k \in E_j(\boldsymbol{m}_i)} w_k}{q_i}$$

$$U = A + B = \begin{bmatrix} C & D \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ E & F \end{bmatrix}$$

EMC of the Producer / Consumer GSPN

C =

	1	2	3	4	5	6
1						
2			1			
3						
4						
5						
6				1		

D =

	7	8	9	10	11
1			1		
2					
3	1				
4				1	
5			1		
6					

E =

	1	2	3	4	5	6
7	b					
8		1				
9			a	b		
10					a	
11						1

F =

	7	8	9	10	11
7		a			
8					
9					
10					b
11					

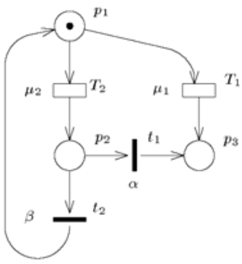
Solution of the EMC

$\boldsymbol{\psi}$ **Probability distribution vector**

$$\boldsymbol{\psi}(n) = \boldsymbol{\psi}(0)\boldsymbol{U}^n$$

$$\begin{cases} \boldsymbol{\psi} = \boldsymbol{\psi}\, \boldsymbol{U} \\ \boldsymbol{\psi}\, \mathbf{1}^{\mathrm{T}} = 1 \end{cases}$$

Construction of the Reduced EMC (REMC): an example



$$u'_{ij} = \frac{\mu_1}{(\mu_1 + \mu_2)} + \frac{\mu_2}{(\mu_1 + \mu_2)} \frac{\alpha}{(\alpha + \beta)}$$

Construction of the REMC: general expressions (1)

$U' = [u'_{ij}]$ **transition probability matrix of the REMC**

$$u'_{ij} = f_{ij} + \sum_{r:\boldsymbol{m}_r \in VS} e_{ir} P\{r \rightarrow s\}\, d_{sj}$$

where $P\{r \rightarrow s\}\, d_{sj}$ **represents the probability of moving from vanishing marking** m_r **to tangible marking** m_j **following a path through vanishing markings only**

In matrix notation

$$\boldsymbol{U}' = \boldsymbol{F} + \boldsymbol{E}\,\boldsymbol{G}\,\boldsymbol{D}$$

where

$$\boldsymbol{G} = \sum_{n=0}^{\infty} \boldsymbol{C}^n$$

Construction of the REMC: general expressions (2)

There are no loops among vanishing states only

$$\exists n_0 \ : \quad \forall \ n \geq n_0 \quad \boldsymbol{C}^n \ = \ 0$$

$$\boldsymbol{G} \ = \ \sum_{k=0}^{\infty} \boldsymbol{C}^k \ = \ \sum_{k=0}^{n_0} \boldsymbol{C}^k$$

There are loops among vanishing states only

$$\lim_{n \to \infty} \boldsymbol{C}^n \ = \ 0$$

$$\boldsymbol{G} \ = \ \sum_{k=0}^{\infty} \boldsymbol{C}^k \ = \ [\boldsymbol{I} \ - \ \boldsymbol{C}]^{-1}.$$

In general

$$\boldsymbol{H} \ = \ \begin{cases} \left(\sum_{k=0}^{n_0} \boldsymbol{C}^k \right) \boldsymbol{D} & \textit{no loops among vanishing states} \\ [\boldsymbol{I} \ - \ \boldsymbol{C}]^{-1} \ \boldsymbol{D} & \textit{loops among vanishing states} \end{cases}$$

from which, the transition proability matrix of the REMC becomes

$$\boldsymbol{U}' \ = \ \boldsymbol{F} \ + \ \boldsymbol{E} \ \boldsymbol{H}$$

The REMC of the Producer/Consumer GSPN (1)

C =

	1	2	3	4	5	6
1						
2			1			
3						
4						
5						
6				1		

D =

	7	8	9	10	11
1			1		
2					
3	1				
4				1	
5			1		
6					

E =

	1	2	3	4	5	6
7	b					
8		1				
9			a	b		
10					a	
11						1

F =

	7	8	9	10	11
7		a			
8					
9					
10					b
11					

The REMC of the Producer/Consumer GSPN (2)

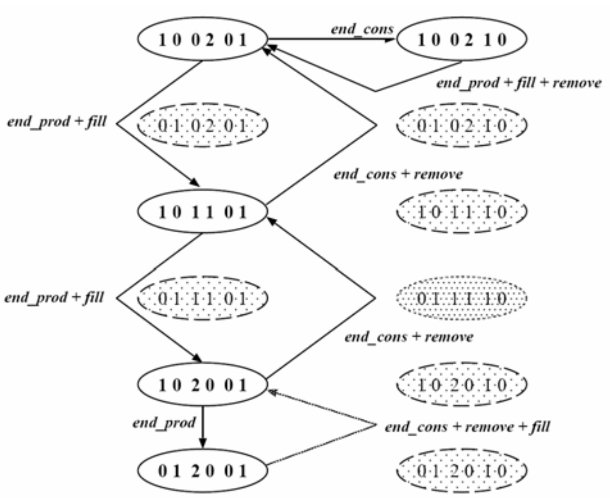
$G = I + C =$

	1	2	3	4	5	6
1	1					
2		1	1			
3			1			
4				1		
5					1	
6				1		1

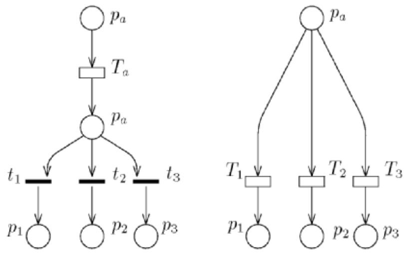
$U' = F + EGD =$

	7	8	9	10	11
7		a	b		
8	1				
9	a			b	
10			a		b
11				1	

The REMC of the Producer/Consumer GSPN (3)



Reducing the GSPN to an equivalent SPN



GSPN tools

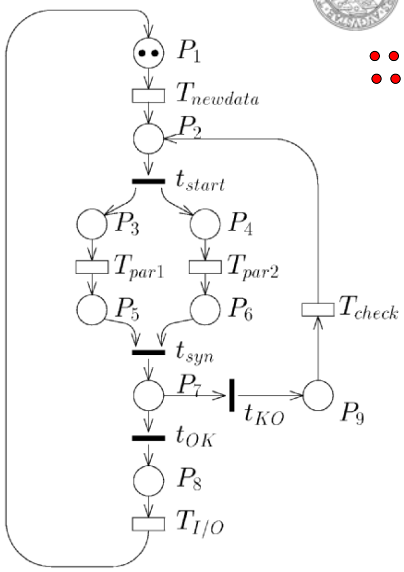
- **Features:**
  - **model construction**
  - **model debugging**
  - **definition of performance indices and logical properties**
  - **model solution**
  - **computation of aggregate results**
  - **display of results**
- **Some addresses:**
  - **GreatSPN: www.di.unito.it/~greatspn**
  - **HiQPN: ls4-www.informatik.uni-dortmund.de/QPN**
  - **SMART: www.cs.ucr.edu/~ciardo/SMART**
  - **UltraSAN and Moebius: www.mobius.uiuc.edu**

Case Studies

GSPN example

transition	rate	value	semantics
$T_{newdata}$	λ	1	infinite-server
T_{par1}	μ_1	10	single-server
T_{par2}	μ_2	5	single-server
$T_{I/O}$	ν	25	single-server
T_{check}	θ	0.5	single-server

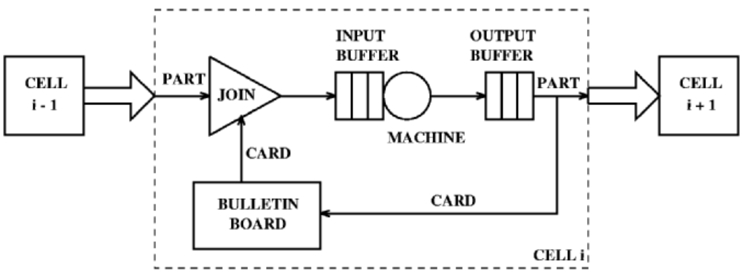
transition	weight	priority	ECS
t_{start}	1	1	1
t_{syn}	1	1	2
t_{OK}	99	1	3
t_{KO}	1	1	3



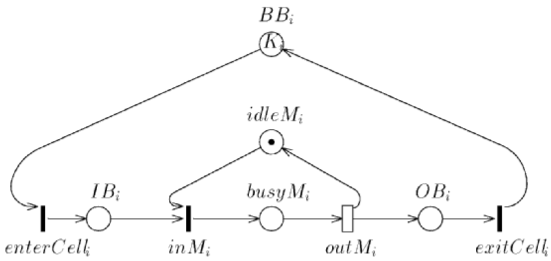
GSPN example

- 20 tangible markings
- 18 vanishing
- Prob. of at least one process waiting for synchronization is 0.238

GSPN case study - kanban

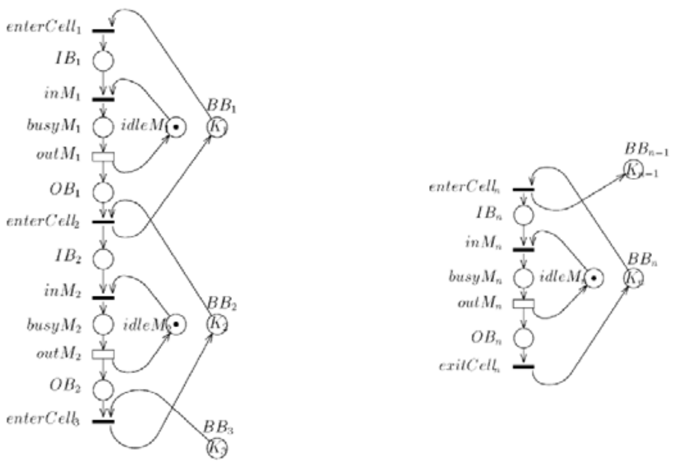


Kanban - basic model



SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 119

Kanban: n-cell sequential composition



SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 120

Kanban: qualitative analysis

- n-cells implies 2n minimal P-semiflow, that generate:

$\forall i, 1 \leq i \leq n$:

$$M(BB_i) + M(IB_i) + M(busyM_i) + M(OB_i) = K_i$$

$$M(idleM_i) + M(busyM_i) = 1$$

- The number of parts in a cell is at most Ki, the number of card in the cell
- Each machine can process only one part at a time
- Place Idle*i* and Busy*i* are mutually exclusive

Kanban - qualitative analysis

- All transitions are covered by a single minimal *T*-semiiflow, representing the deterministic flow of parts
- Behaviour is deterministic (no structural conflicts and therefore non effective conflicts nor confusion can arise)

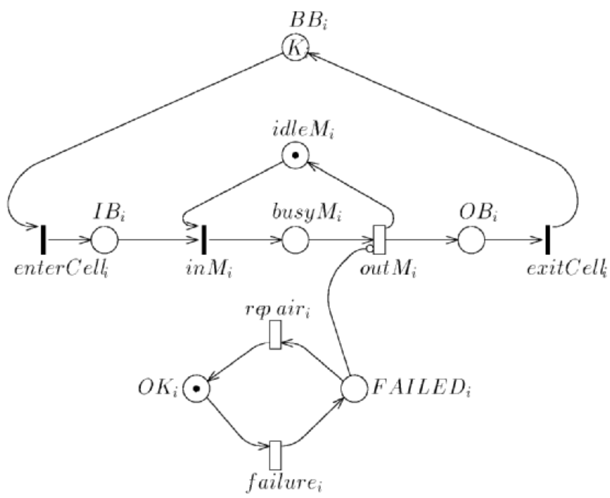
Kanban - quantitative analysis

- K cards, n=5 cells of equal machine time (rate = 4.0)
- Value of the input and output inventory

	Input buffer inventory			Output buffer inventory		
Cell	1 Card	2 Cards	3 Cards	1 Card	2 Cards	3 Cards
1	0.486	1.041	1.474	0.514	0.958	1.526
2	0.486	1.040	1.470	0.383	0.713	1.131
3	0.486	1.047	1.478	0.282	0.524	0.811
4	0.486	1.056	1.490	0.170	0.316	0.472
5	0.486	1.073	1.515	0.000	0.000	0.000

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 123

Kanban with failure

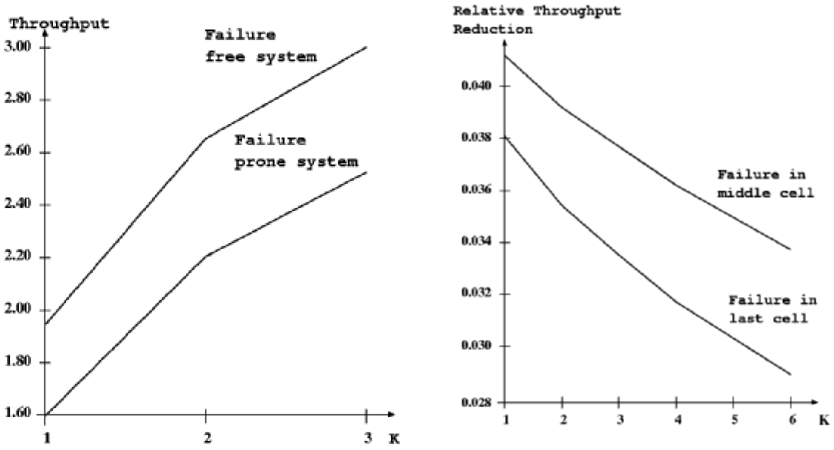


SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 124

Kanban with failure

- Cells can fail independently
- Failure rate = 0.02, repair rate = 0.4
- In a perfectly balanced Kanban system the cell performance is position dependent

Kanban with failure



Advanced Material

Stochastic Petri Nets and Discrete Event Dynamic Systems

In principle

Stochastic Petri Nets are adequate for the representation of any Discrete Event Dynamic System since they

- capture in a very natural way the essence of the dynamic behaviour of these systems
- support the automatic construction of stochastic processes

In practice

The analysis of Stochastic Petri Nets suffers from

- the state space explosion that limits the applicability of all the numerical techniques based on the construction of such state space,
- the constraints of time specifications using the negative exponential distributions

Solution Methodologies

To overcome these problems, many differeny approaches can be adopted

Net-driven Markov Chain Generation

Reduced size Markov chains are generated exploiting structuralfeatures of the model such as submodels and symmetries

- **Net structure allows a ``clever'' Markov Chain generation**
- **Tensor-based methods: Decomposability**
- **Symmetries and exact lumping (= quasi-lumpability)**
- **Combination of Symmetries and Decomposition**
- **Compositional aggregation (using ideas from SPA)**

Net-level Analysis Techniques

Subclasses of models are identified for which the quantitative evaluation can be performed with direct methods that avoid the construction of the state space

- **No Markov Chain generation: Analysis at net-level**
- **Performance Bounds**
- **Product Forms**

Solution Methodologies

Decomposition and Approximation

Approximation results are computed exploiting specific features of the models

- **Divide and Conquer techniques to manage complexity**
- **Response Time Approximations**

General Firing Time Distributions

- **A need exists for including in Petri net representations of random firing delays with low variability (or even constant duration) as well as with high variability, and for addressing the effects that different service policies may have on the behaviour of the real system.**
- **Non-Markovian stochastic processes are generated by Stochastic Petri Nets of this type. Under suitable restrictions, numerical solutions can be computed also for these processes.**
  - **Deterministic Stochastic Petri Nets**
  - **Semi-Regenerative Stochastic Petri Nets**
  - **Phase-Type Distributions**
  - **Fluid stochastic Petri Nets**

Solution Methodologies

Simulation

- Efficient simulation models are derived from the Stochastic Petri Net description of the systems in order to analyze complex real cases
  - Centralized and distributed schemes
  - Acceleration with multiple processors
  - Synchronous schemes
  - Asynchronous schemes

Net-Level Analysis Techniques

Performance Bounds

- Instead of computing exact or approximate results that require dealing with the state space of the model, less computational effort is in general required if we content ourselves with the determination of performance bounds.

- The method is based on:
  - The analysis of the flow of tokens at the net level.
  - The extensive exploitation of structure analysis results which allow to establish flow relationships that are independent of the initial marking of the net.
  - The use of a *linear programming* approach for the computation of upper and lower bounds for linear functions of the average number of tokens in places and for transition throughputs.
  - The study of the *insensitiveness* of the results with respect to the probability distribution of transition firing times.

Net-Level Analysis Techniques

Product Form Stochastic Petri Nets

- **Exact quantitative results can be computed with direct computational methods when the Stochastic Petri Net models satisfy certain conditions that imply a product form expression for the steady-state probability distribution of their markings.**
  - Sufficient conditions are established that allow to easily identify models that exhibit this solution property.
  - Efficient computational algorithms are devised for the solution of these models. *Normalization Constant* and *Mean Value Analysis* approaches have been developed that are direct generalizations of similar methods originally proposed for *Queueing Networks*.
  - Approximation techniques based on Mean Value Analysis are developed that further reduce the computational complexity of these methods thus allowing the analysis of extremely large models.

Decomposition and Approximation

Divide and Conquer

- Decomposition of large models is the process that identifies the existence of component submodels strictly on the basis of their structural properties.
- Net-Driven Decomposition Techniques use the concepts related with P-semiflows, implicit places, etc. to identify net components (and their complements) that can be analyzed within a divide and conquer approach for the solution of the model.
- Iterative procedures are devised in which
  - components are analyzed in isolation to compute equivalent compact representations;
  - improvements in the characterization of the individual submodels are obtained by including in their analysis abstract (compact) representations of the other components of the model
- For strongly connected marked graphs a method exists
  - for automatically splitting the model into two aggregated subsystems and for deriving a basic skeleton system that preserves the properties of the original model
  - for iteratively obtaining an approximate solution of the original problem by characterizing each subsystem with its response time approximation.

General Firing Time Distributions
Generally Distributed SPN

Definition: ***A GDSPN is a marked SPN in which:***

- ***A random variable θ_k is associated with any timed transition t_k2 T (the set of transitions of the net). θ_k models the time needed by the activity represented by t_k to complete, when considered in isolation.***
- ***Each random variable θ_k is characterized by its (possibly marking dependent) cumulative distribution function.***
- ***A set of specifications is given for unequivocally defining the stochastic process associated with the ensemble of all the timed execution sequences T_E This set of specifications is called the execution policy.***
- ***An initial probability is given on the reachability set***

The potential complexity of the behaviour of these nets and the Inclusion of distributions that lack (in general) a memory-less property, require a careful characterization of the probabilistic models associated with these specifications.

General Firing Time Distributions

The inclusion of non-exponential timings destroys the memoryless property and forces the specification of how the system is conditioned upon its past history.

The execution policy comprises two specifications:

- **a criterion to choose the next timed transition to fire (*the firing policy*);**
- **a criterion to keep memory of the past history of the process (the *memory policy*).**

General Firing Time Distributions

Firing policy

When the net enters a new marking, a random sample is extracted for each transition enabled in that marking.

The sample whose value is minimum determines which transition will actually fire, and the sojourn time in this marking equals this minimum sampled value.

We only consider the case in which the random variables associated with the transitions of the SPN are independent.

Memory policy

Assume that an *age variable* a_k, associated with each timed transition t_k, is defined that increases with the time in which the corresponding transition is enabled.

The way in which a_k is related to the past history determines the different memory policies.

We consider the following three alternatives:

- ***Resampling* - The age variable a_k is reset to zero at any change of marking.**
- ***Enabling Memory* - The age variable a_k accounts for the time elapsed from the last epoch in which t_k has been enabled.**
- ***Age Memory* - The age variable a_k accounts for the total time in which t_k has been enabled from its last firing.**

Deterministic SPNs

The Deterministic and Stochastic PN model has been introduced by Ajmone-Marsan and Chiola for describing systems in which some time variables assume constant values

Definition: ***A DSPN is a GDSPN in which:***

- ***The set T of transition is partitioned into a subset T_e of exponential transitions (EXP) and a subset T_d of deterministic transitions (DET), and such that $T=T_e[\ T_d$***
- ***An exponentially distributed random variable θ_j is associated with any EXP transition $t_j 2\ T_e$***
- ***A deterministic firing time d_k is associated with any DET transition $t_k 2\ T_d$***
- ***No more than one DET transition is allowed to be enabled in each marking***
- ***The only allowed execution policy for the DET transition is the race policy with enabling memory***

Deterministic SPNs

Solution approach:

- **During the firing of a DET transition, the marking process can undergo EXP transitions only, thus describing a *Continuous Time Markov Chain (CTMC)* called the subordinated process.**
- **The stationary distribution is computed on the basis of the evaluation of the subordinated *CTMC* at a time corresponding to the duration of the DET transition.**
- **The reachability set is partitioned into disjoint sets of markings according the deterministic transitions that they enable.**

Markov Regenerative SPN (MRSPN)

A natural extension of the DSPN model replaces each DET transition with a GEN transition.

This new model is referred as MRSPN*

Definition: ***A MRSPN* is a GDSPN in which:***

- ***The set T of transition is partitioned into a subset T_e of exponential transitions (EXP) and a subset T_g of generally distributed transitions(GEN), such that $T=T_e[\ T_g$***
- ***An exponentially distributed random variable θ_k is associated***
  - ***with any EXP transition $t_k 2\ T_e$***
- ***A generally distributed random variable \theta_j is associated with any GEN transition $t_j 2\ T_g$***
- ***At most, a single GEN transition is allowed to be enabled in each marking***
- ***The only allowed execution policy for the GEN transition is the* race *policy with* enabling memory**

Markov Regenerative SPN (MRSPN)

In principle, the solution of this more general model can be computed using the approach presented for the DSPN case.

Closed form expressions are presented in the literature when the GEN transitions of the *MRSPN have uniform distributions**

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 141

Phase-Type SPN (PHSPN) [1]

A numerically tractable realization of the GDSPN, is obtained by restricting the random firing times θ_k to have a Phase Type distribution (*PH*).

Definition: ***A PHSPN is a GDSPN in which:***

- ***A PH random variable θ_k is associated with any timed transition t_k2 T***
  - ***The PH model assigned to θ_k has ν_k stages with a single initial stage numbered stage 1 and a single final stage numbered stage ν_k***
- ***A memory policy of resampling, enabling or age memory type, is assigned to any timed transition t_k2 T.***

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 142

Phase-Type SPN [2]

PH distributions are the distributions of the time till absorption of continuous time homogeneous Markov chains with at least one absorbing state.

- **The simplest subclasses of PH distributions, like Erlang, Hyperexponential (and trivially Exponential), are commonly encountered in various areas of applied stochastic modeling.**
- **When the transition firing distributions are of Phase-type, the reachability graph of the PN can be expanded to produce a continuous-time homogeneous Markov chain, equivalent to the original non-Markov process.**
- **The measures pertinent to the original process can then be evaluated by solving the expanded Markov chain.**

Fluid Stochastic Petri Nets (FSPN) [1]

Definition: ***A FSPN is a marked GDSPN in which:***

- ***The set of places is the union of two disjoint set of discrete and continuous places***
  - ***Discrete places contains tokens (characterized by natural numbers)***
  - ***Continuous places contain fluid (characterized by real numbers)***
- ***Arcs are of two different types, depending on the fact they "transport" tokens or fluid***
  - ***A special type of arc transporting fluid are the "flush out arcs" that emty a fluid place when triggered by the firing of the transition they connect to the place***

The analysis has been developed in full details for FSPN with a single fluid places, but suggestions exist in the literature on how to extend it to the case of many fluid places

Fluid Stochastic Petri Nets (FSPN) [2]

Simulation is the common technique usd for the quantitative evaluation of this type of models

Fluid places may be used to represent the supplementary variables needed for the numerical solution of GDSPN thus making the FSPN a practical tool for the evaluation of GDSPN with different type of memory policies.

Tools exist for the evaluation of FSPNs

Simulation

Simulation supplements mathematical/analytical modelling approaches in the analysis of real systems.

Stochastic Petri Net models are naturally suited for discrete event simulation, since they describe the behaviours of real systems in terms of *events* that correspond to transition firings.

The simulation of Stochastic Petri Nets, specified in the most general possible way, requires a precise definition of

- firing delays
- firing semantics
- firing policies

The simulation of complex models is a time expensive analysis technique that must take advantage of any conceivable speed-up method

Net-driven Markov Chain Generation

Net-driven Markov chain generation techniques are emerging as general-purpose methods for controlling the complexity of real-life DEDS models in which both modularity and non-Markovian assumptions play important roles.

The exploitation of peculiar features of these models coming from

- *modularity* due to the compositional development of the models
- *symmetry* induced by the use of high-level model specification formalisms
- *regularity* deriving from the simple structure of the Phase-type expansion of general distributions

yields solution methods that may handle finite, but huge state spaces.

Net-driven Markov Chain Generation

Tensor Based Methods

Basic ideas

- Express the infinitesimal generator of a GSPN in terms of the infinitesimal generators of component submodels
- Avoid the construction of the infinitesimal generator of the whole model by using the component generators during the solution phase

The solution technique is based on the structure of the model and is thus referred as a *structural solution method*

The method may enlarge of one order of magnitude the size (state space) of the models that can be solved by lowering the computational cost of the analysis in terms of space as well as of time complexity

Net-driven Markov Chain Generation

Symmetries [1]

Well Formed Petri Nets are characterized by a structured syntax for the definition of colour domains and arc functions. They allow:

- the automatic exploitation of the symmetries of the model for the generation of a symbolic (aggregated) reachability graph.
- the specification of continuous time Markov chains in which the aggregations of the symbolic reachability graph satisfy the strong lumpability condition.
- the quantitative evaluation of the original model in terms of the solution of the aggregated model.

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 149

Net-driven Markov Chain Generation

Symmetries [2]

There are however cases in which the aggregation of states identified by the symbolic reachability graph typical of Well Formed Nets does not satisfy the lumpability condition, due to particular specifications of the stochastic part of the model

- *quasi-lumpable* well formed stochastic Petri nets are defined in these cases
- approximate results and bounds on the steady-state probability distribution are computed
- Reduced size models are analyzed with a technique that is based on the theory of *Nearly Decomposable Systems* of Simon and Ando.

SFM 07:PE - May 29-th, 2007 - Bertinoro - Italy 150

Net-driven Markov Chain Generation

Compositionality and Aggregation

Often, complex systems are obtained from the composition of symmetric subsystems

Composition and (symmetry) aggregation are not independent processes as synchronization among coloured submodels can involve only certain colours, affecting the original symmetries of the individual components

- development of solution methods that combine the tensor and lumpability approaches
- definition of enlarged components that account for the external synchronizations deriving from the interactions with the other modules.
- equivalence induces lumpability in the Markov chain that represents the probabilistic model associated with a stochastic process algebra representation of a system.
- quantitative evaluation of the original problem performed on a reduced size Markov chain.

Conclusions

- Stochastic Petri nets techniques are attractive because they provide a performance evaluation approach based on formal description
- This allow the same language to be used for
  - specification
  - validation
  - performance evaluation
  - implementation
  - documentation

  of a system
- Real life systems often yield Stochastic Petri net models that are very difficult to analyze, unless proper tools and methods are developed and made available to the researcher

References [1]

Books

- M. Ajmone Marsan, G. Balbo, and G. Conte, *Performance Models of Multiprocessor Systems*, MIT Press, Cambridge, USA, 1986.
- M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, J. Wiley, 1995.
- G. Balbo, and M. Silva, *Performance Models for Discrete Event Dynamic Systems with Synchronizations: Formalisms and Analysis Techniques - Vol.1 and 2*,KRONOS, Zaragoza, Spain, 1998
- F. Di Cesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat, *Practice of Petri Nets in Manufacturing*, Chapman & Hall, 1993.
- P.J. Courtois, *Decomposability: Queueing and Computer Systems Applications*, Academic Press, New York, 1977.
- C. Lindemann, *Performance Modelling with Deterministic and Stochastic Petri Nets*, J. Wiley, New York, NY, 1998.
- M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models*, Johns Hopkins University Press, Baltimore, MD, 1981.
- W. Reisig, *Petri Nets: An Introduction*, Springer, 1985
- T.G. Robertazzi, *Computer Networks and Systems: Queueing Theory and Performance Evaluation,* Springer Verlag, 1991.

References [2]

Stochastic Petri Nets

- A. Blakemore and S.K. Tripathi, “Automated time scale decomposition and analysis of stochastic Petri Nets,” *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, pages 248--257, Toulouse, France, October 1993. IEEE-CS Press.
- A. Bobbio, “Stochastic reward models in performance/reliability analysis,” *Journal on Communications*, XLIII:27--35, January 1992.
- G. Ciardo, *Analysis of large Petri net models*, PhD thesis, Department of Computer Science, Duke University, Durham, NC, USA, 1989, Ph. D. Thesis.
- G. Ciardo, R. German, and C. Lindemann, “A characterization of the stochastic process underlying a stochastic Petri net,” *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, pages 170--179, Toulouse, France, October 1993. IEEE-CS Press.
- G. Chiola, C. Anglano, J. Campos, J.M. Colom, and M. Silva, “Operational analysis of timed Petri nets and applications to the computation of performance bounds,” *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, Toulouse, France, October 1993. IEEE-CS Press.
- G. Chiola, J. Campos, J.M. Colom, and M. Silva, “Operational analysis of timed Petri nets’” *In Proc. 16-th Int. Symp. On Computer Performance, Modeling, Measurements, and Evaluation “PERFORMANCE ‘93”*, Rome, September 1993.
- B.R. Haverkort, “Matrix-geometric solution of infinite stochastic Petri nets,” *In Proceedings of the IPDS International Conference*, pages 72--81, Erlangen, Germany, May 1995.
- B.R. Haverkort and K. Trivedi, “Specification techniques for Markov Reward Models,” *Discrete Event Dynamic Systems: Theory and Applications*, Vol.3, July 1993, Springer Netherlands, pages 219-247.
- A. Reibman, R. Smith, and K.S. Trivedi, “Markov and Markov reward model transient analysis: an overview of numerical approaches,” *European Journal of Operation Research,* 40:257--267, 1989.

References [3]

Generalized Stochastic Petri Nets

- M. Ajmone-Marsan, G. Balbo, and G. Conte, “A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems,” *ACM Transactions on Computer Systems*, 2(1), May 1984.
- A. Blakemore, “The cost of eliminating vanishing markings from generalized stochastic Petri nets,” *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
- G. Ciardo, J. Muppala, and K.S. Trivedi, “On the solution of GSPN reward models,” *Performance Evaluation,* 12(4):237--253, July 1991.
- G. Chiola, M. Ajmone-Marsan, G. Balbo, and G. Conte, “Generalized Stochastic Petri Nets: A definition at the net level and its implications,” *IEEE Transactions on Software Engineering*, 19(2), February 1993, pages 89-107.

References [4]

Well Formed Stochastic Petri Nets

- G. Balbo, G. Chiola, S.C. Bruell, and P. Chen, “An example of modelling and evaluation of a concurrent program using coloured stochastic Petri nets: Lamport's fast mutual exclusion algorithm,” *IEEE Transactions on Parallel and Distributed Systems*, 3(2):221--240, March 1992.
- G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, “Stochastic Well-Formed coloured nets for symmetric modelling Applications,” *IEEE Transactions on Computers*, 42(11), November 1993.
- G. Chiola and G. Franceschinis, “Colored GSPN models and automatic symmetry detection,” *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
- G. Chiola and G. Franceschinis, “A structural colour simplification in Well-Formed coloured nets,” *In Proc. 4-th Intern. Workshop on Petri Nets and Performance Models*, pages 144--153, Melbourne, Australia, December 1991. IEEE-CS Press.
- C. Dutheillet and S. Haddad, “Aggregation and disaggregation of states in colored stochastic Petri nets: Application to a multiprocessor architecture,” *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.

References [5]

Generally Distributed Stochastic Petri Nets

- M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani, "The effect of execution policies on the semantics and analysis of stochastic Petri nets," *IEEE Transactions on Software Engineering*}, 15(7):832--846, July 1989.
- A. Bobbio and M. Telek, Computational restrictions for SPN with generally distributed transition times," *In Proc. First European Dependable Computing Conference LNCS 852*, pages 131--148. 1994.
- A. Bobbio and M. Telek, "Markov regenerative SPN with non-overlapping activity cycles," Technical report, Department of Telecommunications – Technical University of Budapest, Budapest, Hungary, October 1994.
- A. Bobbio and M. Telek "Transient analysis of a preemptive resume M/D/1/2/2 through Petri Nets" Technical report, Department of Telecommunications – Technical University of Budapest, Budapest, Hungary, April 1994.
- H. Choi, V.S. Kulkarni, and K.S. Trivedi, "Markov regenerative stochastic Petri nets," *In Proc. of Performance 93*, Rome, Italy, September 1993.
- J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola, "Extended stochastic Petri nets: Applications and analysis," *In Proc. PERFORMANCE '84*, Paris, France, December 1984.
- R. German, "Transient analysis of deterministic and stochastic Petri nets by the method of supplementary variables," *In Proc. of MASCOT'95*, 1995.
- R. German, C. Kelling, A. Zimmermann, and G. Hommel, "{TimeNET} - a toolkit for evaluating non-Markovian stochastic Petri nets." Technical Report 19, Technische Universitat Berlin, Berlin, Germany, 1994.
- R. German and C. Lindemann, "Analysis of stochastic Petri nets by the method of supplementary Variables," *In Proc. of Performance 93*, Rome, Italy, September 1993.
- P. J. Haas and G. S. Shedler, "Regenerative stochastic Petri nets," *Performance Evaluation*, 6(3):189--204, September 1986.
- V. Mainkar, H. Choi, and K. Trivedi, "Sensitivity analysis of Markov regenerative stochastic Petri Nets," *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, Toulouse, France, October 1993. IEEE-CS Press.

References [6]

Deterministic and Stochastic Petri Nets

- M. Ajmone Marsan and G. Chiola, "On Petri Nets with Deterministic and Exponentially Distributed Firing Times," *volume 266 of LNCS*}, pages 132--145. Springer Verlag, 1987.
- H. Choi, G. Kulkarni, and K.S. Trivedi, "Transient analysis of deterministic and stochastic petri nets," *In Proc. 14-th Intern. Conference on Application and Theory of Petri Nets*, Chicago, Illinois, June 1993. Springer Verlag.
- G. Ciardo and C. Lindemann, "Analysis of deterministic and stochastic Petri nets," *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, pages 160--169, Toulouse, France, October 1993. IEEE-CS Press.
- C. Lindemann, "An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models," *Performance Evaluation*, 18:75--95, 1993.

References [7]

Phase Stochastic Petri Nets

- D. Aldous and L. Shepp. “The least variable phase type distribution is Erlang,” *Stochastic Models*, 3:467--473, 1987.
- G. Balbo, G. Chiola, S.C. Bruell, and P. Chen, “An example of modelling and evaluation of a concurrent program using coloured stochastic Petri nets: Lamport's fast mutual exclusion algorithm,” *IEEE Transactions on Parallel and Distributed Systems*, 3(2):221--240, March 1992.
- A. Bobbio and A. Cumani, “Discrete state stochastic systems with phase type distributed transition times,” *In Proc. AMSE Intern. Conf. on Modelling and Simulation*, Athens, 1984.
- A. Bobbio and M. Telek, “A benchmark for {PH} estimation algorithms: results for Acyclic-PH,” *Stochastic Models*, 10:661--677, 1994.
- P. Chen, S.C. Bruell, and G. Balbo, “Alternative methods for incorporating non-exponential distributions into stochastic Petri nets,” *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, pages 187--197, Kyoto, Japan, December 1989. IEEE-CS Press.
- A. Cumani, “On the canonical representation of processes modelling failure time distributions,” *Microelectron Reliability*, 22:583--602, 1982.
- A. Cumani, “Esp - a package for the evaluation of stochastic Petri nets with phase-type distributed transition times,” *In Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.

References [8]

Fluid Stochastic Petri Nets

- R.German, M. Gribaudo, G. Horvath, M. Telek. “Stationary Analysis of FSPNs with mutually dependent Discrete And Continuous Part,” *In 10-th Intern. Workshop on Petri Nets and Performance Models (PNPM '03)*, Urbana, Illinois, USA, September 2-5, 2003.
- M. Gribaudo, M. Sereno, A. Horvath, and A. Bobbio, “Fluid stochastic Petri nets augmented with flush-out arcs: Modelling and analysis,” *Discrete Event Dynamic Systems*, pages 97-- 117, January 2001
- G. Horton, V. Kulkarni, D. Nicol, K. Trivedi, “Fluid stochastic Petri nets: Theory, application and solution techniques,” *Eur. J. Oper. Res.,* 105(1): 184--201,1998.
- M. Gribaudo, R. Gaeta. “Efficient steady-state analysis of Second-Order Fluid Stochastic Petri Nets,” *to appear in Performance Evaluation.*
- M. Gribaudo, A. Horvath, “Model Checking Functional and Performability Properties of Stochastic Fluid Models,” *In Fourth International Workshop on Automated Verification of Critical Systems : AVoCS'04*, September 4, 2004 London. (To appear in ENTCS)
- M. Gribaudo, A. Horváth, A. Bobbio, E. Tronci, E. Ciancamerla, and M. Minichino, “Fluid Petri nets and hybrid model-checking: A comparative case study,” *In Reliability Engineering & System Safety*, 81(3):239-257, 2003.
- A. Horvath, M. Gribaudo, “Matrix Geometric Solution of Fluid Stochastic Petri Nets,” *In Fourth International Conference on Matrix Analytic Methods in Stochastic Models*, Adelaide, Australia,14 - 18 July 2002.
- M. Gribaudo, A. Horvath, “Fluid Stochastic Petri Nets Augmented with Flush-out Arcs: a Transient Analysis Technique,” *In Proceeding of Ninth International Workshop on Petri Nets and Performance Models 2001, PNPM'01*, pages 145-154, Aachen, Germany, September 2001. IEEE-CS Press

References [9]

Bounds

- J. Campos, G. Chiola, J. M. Colom, and M. Silva, "Properties and performance bounds for timed marked graphs," *IEEE Transactions on Circuits and Systems---1: Fundamental Theory and Applications*, 39(5):386--401, May 1992.
- J. Campos, G. Chiola, J.M. Colom, and M. Silva, "Tight polynomial bounds for steady-state performance of marked Graphs,", *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
- J. Campos, G. Chiola, and M. Silva, "Properties and steady-state performance bounds for Petri nets with unique repetitive firing count vector," *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
- J. Campos, G. Chiola, and M. Silva, "Ergodicity and throughput bounds for Petri nets with unique consistent firing count vector'" *IEEE Transactions on Software Engineering*, 17(2):117--125, February 1991.
- J. Campos, G. Chiola, and M. Silva, "Properties and performance bounds for closed free choice synchronized monoclass queueing networks," *IEEE Transactions on Automatic Control*, 36(12):1368--1382, December 1991.
- J. Campos, J.M. Colom, H. Jungnitz, and M. Silva, "Approximate throughput computation of stochastic marked graphs," *IEEE Transactions on Software Engineering,* 20(7):526--535, July 1994.
- H.J. Jungnitz, B. Sanchez, and M. Silva, "Approximate throughput computation of stochastic marked graphs," *Journal of Parallel and Distributed Computing*, 15:282--295, 1992.

References [10]

Product Form

- G. Balbo, S. C. Bruell, and M. Sereno, "Arrival theorems for product-form stochastic Petri nets," *In Proc. 1994 SIGMETRICS Conference*, Nashville, Tennessee, USA, May 1994. ACM.
- R.J. Boucherie, "A characterization of independence for competing Markov chains with applications to stochastic Petri nets," *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, pages 117--126, Toulouse, France, October 1993. IEEE-CS Press.
- J.L. Coleman, "Algorithms for product-form stochastic Petri nets - a new approach," *In Proc. 5^{th} Intern. Workshop on Petri Nets and Performance Models*, {Toulouse, France}, October 1993. IEEE-CS Press.
- J.L. Coleman, *Stochastic Petri Nets with Product Form Equilibrium Distributions*, PhD thesis, University of Adelaide, Adelaide, Australia, December 1993.
- S. Donatelli and M. Sereno, "On the Product Form Solution for Stochastic Petri Nets," *LNCS, N. 616*, Springer Verlag, Sheffield, England, pages 154—172, June 1992.
- A.A. Lazar and T.G. Robertazzi, "Markovian Petri net protocols with product form solution," *Performance Evaluation,* 12:67--77, 1991.
- M. Sereno and G. Balbo, "Computational algorithms for product form solution stochastic Petri Nets," *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, {Toulouse, France}, October 1993. IEEE-CS Press.

Tensor Products

- J. Campos, S. Donatelli, and M. Silva, "Structured solution of stochastic DSSP systems," *In Proc. 7-th Intern. Workshop on Petri Nets and Performance Models*, pages 91--100, Saint Malo, France, June 1997. IEEE-CS Press.
- S. Donatelli, "Kronecker Algebra and (Stochastic) Petri Nets: Is It Worth the Effort?," *ICATPN 2001*: 1-18.
- S. Donatelli, and P. Kemper, "Integrating synchronization with priority into a Kronecker representation," *Performance Evaluation 44*(1-4): 73-96 (2001)

References [11]

Aggregation / Decomposition

- P. Buchholz, “Hierarchical high level Petri nets for complex system analysis,” *In Proc. 15-th} Intern. Conference on Applications and Theory of Petri Nets, number 185 in LNCS*, Zaragoza, Spain, 1994. Springer-Verlag.
- J. Campos, J.M. Colom, H. Jungnitz, and M. Silva, “A general iterative technique for approximate throughput computation of stochastic marked graphs,” *In Proc. 5-th Intern. Workshop on Petri Nets and Performance Models*, pages 138--147, Toulouse, France, October 1993. IEEE-CS Press.
- J. Campos, M. Silva, and S. Donatelli, "Structured solution of stochastic DSSP systems," *In Proc. 7th Int. Workshop on Petri Nets and Performance Models (PNPM'97),* Saint Malo, France, June 1997, IEEE Comp. Soc. Press, pp. 91--100, 1997.
- W. Henderson and D. Lucic, “Exact results in the aggregation and disaggregation of stochastic Petri nets,” *In Proc. 4-th Intern. Workshop on Petri Nets and Performance Models*, pages 166--175, Melbourne, Australia, December 1991, IEEE-CS Press.
- W. Henderson and P.G. Taylor, “Aggregation methods in exact performance analysis of stochastic Petri nets,” *In Proc. 3-rd Intern. Workshop on Petri Nets and Performance Models*, pages 12--18, Kyoto, Japan, December 1989. IEEE-CS Press.

.

References [12]

Simulation

- H.H. Ammar, and S. Deng, “Time warp simulation of stochastic Petri Nets,” *In Proc. 4-th Int. Workshop on Petri Nets and Performance Models*, Melbourne, Australia,December 1991,pages 186-195, IEEE-CS Press.
- H.H. Ammar and S.M. Rezaul Islam, “Time scale decomposition of a class of generalized stochastic Petri net models,” *IEEE Transactions on Software Engineering*, 15(6):809--820, June 1989.
- F. Baccelli and M. Canales, “Parallel simulation of stochastic Petri nets using recurrence Equations,” ACM *Transactions on Modelling and Computer Simulation*, 20(1):20--41, January 1993.
- G. Chiola and A. Ferscha, “Distributed discrete event simulation of timed transition Petri Nets,” *IEEE Parallel and Distributed Technology, 1 Special Issue on “Parallel and Distributed Systems: from theory to practice,”* 1993.
- G. Chiola and A. Ferscha, “Distributed simulation of timed Petri nets: exploiting the net structure to obtain efficiency,” *In Proc. 14-th Intern. Conference on Application and Theory of Petri Nets*, Chicago, Illinois, June 1993. Springer Verlag.
- G. Chiola and A. Ferscha, “Exploiting Petri net model structure to improve distributed Simulation,” *In 26-th Hawaii Intern. Computer Science Symposium,* Honolulu, Hawaii, January 1993.
- G. Chiola and A. Ferscha, “Performance comparable design of efficient synchronization protocols for distributed simulation,” *In Proc. of MASCOT'95*, pages 343--348. IEEE-CS Press, 1995.
- P.J. Haas and G.S. Shedler, “Regenerative simulation of stochastic Petri nets, *In Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
- D.M. Nicol and S. Roy, “Parallel simulation of timed Petri nets,” *In Proc. 1991 Winter Simulation Conference*, pages 574--583, 1991.
- G.S. Thomas and J. Zahorjan, “Parallel simulation of performance Petri nets: Extending the domain of parallel simulation,” In *Proc. 1991 Winter Simulation Conference*, 1991.

References [13]

Applications

- G. Balbo, S. Donatelli, and G. Franceschinis, “Understanding parallel programs behaviour through Petri net models,” *Journal of Parallel and Distributed Computing,* 15(3), July 1992.
- G. Balbo, S. Donatelli, G. Franceschinis, A. Mazzeo, N. Mazzocca, and M. Ribaudo, “On the computation of performance characteristics of concurrent programs using GSPNs,” *Performance Evaluation*, 19:195--222, 1994.
- S. Donatelli, G. Franceschinis, N. Mazzocca, and S. Russo, “Software architecture of the Epoca integrated environment,” *In Proc. of the 7-th Intern. Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, May.