On approximate throughput computation of Deterministic Systems of Sequential Processes *

Carlos J. Pérez-Jiménez, Javier Campos, and Manuel Silva

Submitted to an international workshop.

Abstract

We concentrate on a family of discrete event systems obtained from a simple modular design principle that include in a controlled way primitives to deal with concurrency, decisions, synchronization, blocking, and bulk movements of jobs. Due to the functional complexity of such systems, reliable throughput approximation algorithms must be deeply supported on a structure based decomposition technique. We present two complementary decomposition techniques and a fixed-point search iterative process based on response time preservation of subsystems. An extensive battery of numerical experiments has shown that the error is less than 3%, and that the state space is usually reduced by more than one order of magnitude.

1 Introduction

Performance analysis of *discrete event systems* is an important subject where growing efforts are being devoted. The emerging problems are really complex if general models are considered, including primitives to deal with *concurrency*, *decisions*, *synchronization*, *blocking*, and *bulk movements* of jobs. In [CCJS94], a throughput approximation technique was developed for the analysis of a class of discrete event systems isomorphous to *Fork-Join Queueing Networks with Blocking*, *stochastic strongly connected marked graphs*. Those systems allow concurrency, synchronization, and blocking but neither decisions nor bulk movements.

Deterministic Systems of Sequential Processes (DSSP, in the sequel) are obtained by the application of a simple modular design principle: several functional units (in this case, sequential processes modeled with state machines, SM's) execute concurrently and cooperate using asynchronous communication by message passing through a set of buffers (places with possibly weighted input and output arcs). Buffers are destination private (i.e., they go to a single sequential process). Thus competition among functional units is prevented. Moreover, buffers do not represent side conditions in conflicts of the functional units (i.e., they do not unbalance conflicts). Several classes have been defined in the literature following this basic modular design principle [Rei82, Sou93, CCST94, RTS95]. In this paper, we deal with approximate throughput computation of DSSP's, that include, in a controlled way, all the above enumerated features (concurrency, decisions, synchronization, blocking, and bulk movements).

The approximation technique that we consider is based on two key points:

(1) *A structure based decomposition of the original model leading to two or more smaller aggregated subsystems.* The decomposition must be such that the qualitative behaviour of the isolated subsystems reproduces as accurate as possible that of the original system projected on the preserved nodes. In some cases, the functional behaviour can be captured (reachable markings). Sometimes, only some weaker logical properties can be preserved. In principle, the better the qualitative behaviour of the original system is represented by the aggregated subsystems, the more accurate the quantitative approximation will be expected.

*This work has been developed within the projects PRONTIC 94-0242 of the Spanish CICYT, and ESPRIT BRA 7269 (QMIPS) and HCM CT94-0452 (MATCH) of the European Community. The work of C.J. Pérez-Jiménez is supported by a grant of the CONAI of the Aragonese Autonomy. The authors are with the Departamento de Ingeniería Eléctrica e Informática, Centro Politécnico Superior, Universidad de Zaragoza, María de Luna 3, E-50015 Zaragoza, Spain; e-mail: cjperez, fjcampos, msilva @mcps.unizar.es.

(2) *An iterative response time approximation method for the computation of the throughput.* At each iteration step, the underlying CTMC of every aggregated subsystems are solved, getting more information for the improvement of the response time approximation of the reduced nodes. The use of a *basic skeleton* of the system in order to tune up the response time approximation was firstly introduced in [JSS92] and plays a very important role in the iterative response time approximation technique. An extensive battery of numerical experiments has shown that this approximation technique is very robust even for the most complex models for which the exact functional aggregation is not possible (provided that some basic logical properties are preserved).

Two complementary techniques are presented in this paper for the structural decomposition of large DSSP's. The first of them, based on *SM's reduction*, is a *local* aggregation technique and it preserves all the connections among functional units (i.e., it preserves the buffers and their input and output transitions). It is specially suited for DSSP's composed from large SM's. A cut is defined through buffers, partitioning the SM's into several subsets. An aggregated subsystem is obtained per each of these subsets by reducing as much as possible all the SM's belonging to the rest of subsets. A method for the SM's reduction is presented such that the functional behaviour of the aggregated subsystems is an exact projection of the original system on the preserved nodes (reachable markings and firing sequences). Thus, logical properties like steps, boundedness, liveness, and the existence of home states are also preserved.

If the SM's of a DSSP are not large, then the previous technique may not lead to an important state space reduction. The second technique presented in this paper deals with such situations: not only the SM's but also the buffers are aggregated, thus being a *global* reduction technique. The basic idea is an elaborate generalization of that presented in [JSS92]. A *single input-single output cut* (SISO-cut) is defined that splits the original net in two (multiple splitting can be considered in a hierarchical way). In order to apply this technique, the buffers must be (not only output-private but also) input-private and the SM's must be fully reducible to a transition preserving some aggregated properties of the functional behaviour like the bounds of buffers, liveness or the existence of home states. In that case, the problem reduces to a subnet aggregation in *weighted T-systems* [TCWCS92] (the weighted extension of marked graphs). A method is presented to preserve the mentioned basic functional properties after the aggregation phase, using the *gain* concept [TCWCS92] defined on (weighted) paths. In order to improve approximation, the structural concept of *resistance* of a path is introduced. It is related to the minimum number of times that the first transition of a weighted path must be fired to allow the firing of the last transition of the path. The aggregation technique preserves also the resistance of the aggregated subnet.

The paper is organized as follows. Some basic notation, the formal definition of DSSP's, and the considered stochastic interpretation are introduced in Section 2. Section 3 is devoted to the first decomposition technique, based on SM's reduction. The decomposition phase and the iteration phase are presented, with some numerical results for a selected example. In Section 4, the single input-single output cut is defined; the decomposition phase is based on an algorithm to compute the gain and the resistance of a subnet; the iteration phase, which is analogous than in Section 3, is applied to a representative example. Some concluding remarks and further work are outlined in Section 5.

2 Basics on Deterministic Systems of Sequential Processes

In this section we formally define the class of Deterministic Systems of Sequential Processes as a subclass of Petri net systems, and we explain how time is introduced in the model. Previously, some basic definitions and notations of Petri nets are recalled.

The class that we consider in this paper is an extension of that in [Sou93], although we keep the same name. In that work, sequential processes are modelled with *safe state machines* while the communication among them is described by their connection through particular places called *buffers*: their buffers are private in the sense that each of them has only one input and only one output state machine. Our extension allows that several state machines deposit messages (tokens) in a buffer.

2.1 Basic notations

We assume the reader is familiar with concepts of P/T nets. In this section we present notations used in later sections. For further extensions the reader is referred to [Mur89, Sil93].

A P/T *net* is a 4–tuple $\mathcal{N} = (P, T, Pre, Post)$, where P and T are disjoint sets of *places* and *transitions* ($|P| = n$, $|T| = m$), and Pre ($Post$) is the *pre- (post-) incidence function* representing the input (output) arcs: $Pre: P \times T \to \mathbb{N} = \{0, 1, 2, \ldots\}$ ($Post: P \times T \to \mathbb{N}$). *Ordinary* nets are Petri nets whose pre- and post-incidence functions take values in $\{0, 1\}$. The incidence function of a given arc in a non-ordinary net is called *weight* or *multiplicity*. The *pre-* and *post-set* of a transition $t \in T$ are defined respectively as ${}^\bullet t = \{p | Pre(p, t) > 0\}$ and $t^\bullet = \{p | Post(p, t) > 0\}$. The *pre-* and *post-set* of a place $p \in P$ are defined respectively as ${}^\bullet p = \{t | Post(p, t) > 0\}$ and $p^\bullet = \{t | Pre(p, t) > 0\}$. The *incidence matrix* of the net is defined as $C = [Post(p_i, t_j) - Pre(p_i, t_j)]$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. *Flows* (*semiflows*) are integer (natural) annullers of C. Right and left annullers are called T- and P-(semi)flows respectively. A semiflow is called *minimal* when its support, $\|X\|$, is not a proper superset of the support of any other, and the greatest common divisor of its elements is one. A net is *consistent* iff it has a T-semiflow $X \geq \mathbb{1}$. A net is *conservative* iff it has a P-semiflow $Y \geq \mathbb{1}$.

A function $M: P \to \mathbb{N}$ (usually represented in vector form) is called *marking*. A P/T *system*, or *marked Petri net*, $(\mathcal{N}, M_0)$, is a P/T net $\mathcal{N}$ with an *initial marking* M_0. A transition $t \in T$ is *enabled* at marking M iff $\forall p \in P$: $M(p) \geq Pre(p, t)$. A transition t enabled at M can *fire* yielding a new marking M' (*reached* marking) defined by $M'(p) = M(p) - Pre(p, t) + Post(p, t)$ (it is denoted by $M[t\rangle M'$). A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is a *firing sequence* in $(\mathcal{N}, M_0)$ iff there exists a sequence of markings such that $M_0[t_1\rangle M_1 [t_2\rangle M_2 \ldots [t_n\rangle M_n$. In this case, marking M_n is said to be *reachable* from M_0 by firing σ, and this is denoted by $M_0[\sigma\rangle M_n$. The *reachability set* $R(\mathcal{N}, M_0)$ is the set of all markings reachable from the initial marking. $L(\mathcal{N}, M_0)$ is the *language of firing sequences* of $(\mathcal{N}, M_0)$ ($L(\mathcal{N}, M_0) = \{\sigma \mid M_0[\sigma\rangle\}$).

A place $p \in P$ is said to be k*–bounded* iff $\forall M \in R(\mathcal{N}, M_0)$, $M(p) \leq k$. A P/T system is said to be (marking) k–bounded iff every place is k–bounded, and bounded iff there exists some k for which it is k–bounded. A P/T system is *live* when every transition can ultimately occur from every reachable marking. M is a *home state* in $(\mathcal{N}, M_0)$ iff it is reachable from every reachable marking.

2.2 Deterministic Systems of Sequential Processes

State machines are ordinary Petri nets such that every transition has only one input and only one output place ($\forall t \in T$: $|{}^\bullet t| = |t^\bullet| = 1$). SM's allow the modelling of sequences, decisions (or conflicts), and re-entrance (when they are marked with more than one token) but not synchronization.

Deterministic Systems of Sequential Processes (DSSP's) are used for the modelling and analysis of distributed systems composed by sequential processes communicating through output-private buffers. Each sequential process is modelled by a safe (1–bounded) SM. The communication among them is described by *buffers* (places) which contain *products/messages* (tokens), which are produced by some processes and consumed by others. Each buffer is *output-private* in the sense that it is an input place of only one SM.

Definition 2.1 *A marked Petri net* $(\mathcal{N}, M_0) = (P_1 \cup \ldots \cup P_q \cup B, T_1 \cup \ldots \cup T_q, Pre, Post, M_0)$ *is a* Deterministic System of Sequential Processes *iff:*

1. $\forall i, j \in \{1, \ldots, q\}, i \neq j$: $P_i \cap P_j = \emptyset$, $T_i \cap T_j = \emptyset$, $P_i \cap B = \emptyset$;
2. $\forall i \in \{1, \ldots, q\}$: $(\mathcal{SM}_i, M_{0i}) = (P_i, T_i, Pre_i, Post_i, M_{0i})$ *is a strongly connected and 1–bounded state machine (where* Pre_i*,*$Post_i$*, and* M_{0i} *are the restrictions of* Pre*,* $Post$*, and* M_0 *to* P_i *and* T_i*);*
3. *The set* B *of buffers is such that* $\forall b \in B$:
   (a) $|{}^\bullet b| \geq 1$ *and* $|b^\bullet| \geq 1$,
   (b) $\exists i \in \{1, \ldots, q\}$ *such that* $b^\bullet \subset T_i$,
   (c) $\forall p \in P_1 \cup \ldots \cup P_q$: $t, t' \in p^\bullet \Rightarrow Pre(b, t) = Pre(b, t')$.

The first two items of the previous definition state that a DSSP is composed by a set of SM's ($\mathcal{SM}_i, i = 1, \ldots, q$) and a set of buffers ($B$). By item *3.a*, buffers are neither source nor sink places. The output-private condition is expressed by condition *3.b*. Requirement *3.c* justifies the word "deterministic" in the name of the class: the marking of buffers does not disturb the decisions taken by a SM, i.e., choices in the SM's are free. This definition generalizes the class of DSSP's defined in [Sou93], where buffers are required to have not only a single output SM (output-private) but also a single input SM (input-private). From a queueing network perspective, DSSP's are a mild generalization of *Fork-Join Queuing Networks with Blocking* where servers are complex (safe SM's with a rich connectivity to buffers) and bulk movements of jobs are allowed.

2.3 Stochastic interpretation

We consider net systems with timed transitions. Marking and time independent exponentially distributed random variables associated to the firing of transitions define their *service time*. The mean values of these variables are denoted s_i for each transition t_i of the net.

For the modelling of conflicts we use *immediate transitions* with the addition of (marking and time independent) *routing rates*. In other words, for the subset of immediate transitions $\{t_1, \ldots, t_k\} \subset T$ being in conflict at each reachable marking, we assume that the constants $r_1, \ldots, r_k \in \mathbb{Q}^+$ are explicitly defined in the system interpretation in such a way that when $t_1, \ldots, t_k$ are simultaneously enabled, transition t_i, $i = 1, \ldots, k$, fires with probability $r_i/(\sum_{j=1}^{k} r_j)$. Consequently, routing is completely decoupled from duration of activities.

The *visit ratio* of transition t_i with respect to t_j, $v_i^{(j)}$, is the average number of times t_i is visited (fired) for each visit to (firing of) the reference transition t_j. For live and bounded DSSP's, the vector of visit ratios can be computed by solving a linear system of equations, using the incidence matrix and the routing rates at conflicts (see [CCST94]).

3 SM reduction and exact projection of reachable markings

In this section we present the first technique for the structural decomposition of large DSSP's and the iterative response time approximation method for the computation of throughput. The decomposition technique is specially suited for DSSP's composed from large SM's. A cut is defined through buffers, classifying the SM's into several subsets. An aggregated subsystem is obtained per each of these subsets by reducing as much as possible all the SM's belonging to the rest of subsets. A method for the SM's reduction is presented such that the functional behaviour of the aggregated subsystems is an exact projection of the original system on the preserved nodes (reachable markings and firing sequences).

3.1 Decomposition phase

3.1.1 Observable behaviour preserving reduction of SM's

First, we present the reduction technique for SM's. Let $\mathcal{SM} = (P_{\mathcal{SM}}, T_{\mathcal{SM}}, F_{\mathcal{SM}})$ be a safe strongly connected SM. Given a subset $T_v \subseteq T_{\mathcal{SM}}$ of *observable transitions* we are interested in constructing other safe and strongly connected SM which preserves the *branching probabilities* among observable transitions. By observable transitions we consider those defining the interface between the SM and the rest of the system (i.e. those connected to buffers, that we assume timed). The branching probability p_{ij} between observable transition t_i and observable transition t_j of a given SM is the probability that transition t_j is the next observable transition in that SM which is fired after the firing of t_i.

To preserve branching probabilities we follow a technique similar to that in [BI82]. This technique consists of several steps. Given a SM, a set of observable transitions and the routing probabilities between them, a non-observable transition is shorted at each step and the new routing probabilities are computed. Once all non-observable transitions have been shorted we obtain the branching probabilities between observable transitions. These probabilities are maintained in the reduced net.

To explain this calculus we need some notation. Let $T_v = \{t_1, \ldots, t_n\}$ be the set of observable transitions and $T_{\mathcal{SM}} \setminus T_v = \{1, \ldots, m\}$ the set of non-observable transitions of a state machine $\mathcal{SM}$. We denote by $\mathcal{P}$ the $(n+m) \times (n+m)$ matrix of routing probabilities of $\mathcal{SM}$ (the routing matrix in queuing networks terminology). We are going to compute from $\mathcal{P}$ the reduced SM routing $(n \times n)$ matrix $\mathcal{P}'$. If we short transition k on step k $(k = 1, \ldots, m)$, the new routing probabilities are:

$$p_{ij}^{(k+1)} = p_{ij}^{(k)} + p_{ik}^{(k)} \left(\sum_{r=0}^{\infty} \left(p_{kk}^{(k)} \right)^r \right) p_{kj}^{(k)} \quad i, j = t_1, \ldots, t_n, k+1, \ldots, m$$

$$= \quad p_{ij}^{(k)} + p_{ik}^{(k)} \left(1 - p_{kk}^{(k)}\right)^{-1} p_{kj}^{(k)} \qquad k = 1, \ldots, m \tag{1}$$

Then, matrix $\mathcal{P}'$ is $\mathcal{P}' = \left(p_{ij}^{(m+1)}\right)$. We will denote the elements $p_{ij}^{(m+1)}$ by p'_{ij}.

The reduced state machine $(\mathcal{SM}_{T_v}, M_0^{T_v})$ is easily constructed as follows: For each observable transition t_i, add an input p_i^{in} and an output place p_i^{out}; for each $p'_{ij} = 1$, make the fusion of places p_i^{out} and p_j^{in}; and for the rest of $p'_{ij} \neq 0$, add an immediate transition $t_{i,j}$ from $\{p_i^{out}\}$ to $\{p_j^{in}\}$, with routing rate equal to p'_{ij}.

From this reduction technique, the next properties trivially follow.

Property 3.1 *Let $(\mathcal{SM}, M_0)$ be a safe strongly connected SM and $T_v \subseteq T_{\mathcal{SM}}$ a subset of observable transitions. Let $(\mathcal{SM}_{T_v}, M_0^{T_v})$ be the SM reduced by previous technique. Then, the branching probabilities among all observable transitions are preserved after the reduction.*

Corollary 3.1 *Let $(\mathcal{SM}, M_0)$ be a safe strongly connected SM and $T_v \subseteq T_{\mathcal{SM}}$ a subset of observable transitions. Let $(\mathcal{SM}_{T_v}, M_0^{T_v})$ be the reduced SM by previous technique. Then:*

i) The visit ratios of observable transitions in $(\mathcal{SM}, M_0)$ and $(\mathcal{SM}_{T_v}, M_0^{T_v})$ are the same.

ii) $L(\mathcal{SM}, M_0)|_{T_v} = L\left(\mathcal{SM}_{T_v}, M_0^{T_v}\right)\Big|_{T_v}$

iii) $R(\mathcal{SM}, M_0)|_{^\bullet T_v} = R\left(\mathcal{SM}_{T_v}, M_0^{T_v}\right)\Big|_{^\bullet T_v}$

Now, from the definition of the class of DSSP's, it is clear that provided liveness the only effect of buffers on a single state machine is a possible additional delay for the firing of transitions $(L(\mathcal{N}, M_0)|_{T_{\mathcal{SM}}} = L\left(\mathcal{SM}, M_0^{\mathcal{SM}}\right))$. Therefore, the functional properties of the reduced state machine are also preserved if it is considered within the whole system, as stated in the next property.

Property 3.2 *Let $(\mathcal{N}, M_0)$ be a live and bounded DSSP, $\mathcal{SM}$ a SM of $\mathcal{N}$ and t_v the subset of observable transitions of $\mathcal{SM}$. Let $(\mathcal{N}', M'_0)$ be the net system obtained by reducing $\mathcal{SM}$. Then we have:*

i) The visit ratios of observable transitions of $\mathcal{SM}$ are preserved after the reduction.

ii) $L(\mathcal{N}, M_0)|_{TAS} = L\left(\mathcal{N}', M'_0\right)|_{TAS}$

iii) $R(\mathcal{N}, M_0)|_{PAS} = R\left(\mathcal{N}', M'_0\right)|_{PAS}$

Where TAS= $(T_\mathcal{N} \setminus T_{\mathcal{SM}}) \cup T_v$ and PAS= $(P_\mathcal{N} \setminus P_{\mathcal{SM}}) \cup (^\bullet T_v \cap P_{\mathcal{SM}})$.

3.1.2 Cut, aggregated subsystems and basic skeleton

The decomposition of a live and bounded DSSP (like that in Fig. 1.a) is based on the splitting in k pieces by a cut defined on buffers. Once the cut and the pieces are selected we construct k *aggregated subsystems* ($\mathcal{AS}_1, \ldots, \mathcal{AS}_k$; see Figs. 1.b, 1.c, and 1.d) and a *basic skeleton system* ($\mathcal{BS}$; see Fig. 1.e). First, we formally define the cut.

Definition 3.1 *Let $(\mathcal{N}, M_0) = (P_1 \cup P_2 \cup \cdots \cup P_q \cup B, T_1 \cup T_2 \cup \cdots \cup T_q, Pre, Post, M_0)$ be a live and bounded DSSP. A subset $Q \subseteq B$ of buffers is said k-cut of $\mathcal{N}$ ($k \geq 2$) iff there exist k subnets $\mathcal{N}_i = (P_{\mathcal{N}_i}, T_{\mathcal{N}_i}, Pre_{\mathcal{N}_i}, Post_{\mathcal{N}_i})$, $i = 1, \ldots, k$, of $\mathcal{N}$ verifying:*

i) $\bigcup_{i=1}^{k} T_{\mathcal{N}_i} = T, \;\; T_{\mathcal{N}_i} \cap T_{\mathcal{N}_j} = \emptyset \;\; \forall i \neq j, \;\; i,j = 1, \ldots, k.$

ii) $\forall i \in \{1, \ldots, q\} \; \exists j \in \{1, \ldots, k\}$ *such that* $T_i \subseteq T_{\mathcal{N}_j}$

iii) $P_{\mathcal{N}_i} = {^\bullet T_{\mathcal{N}_i}} \cup T_{\mathcal{N}_i}{^\bullet} \;\; i = 1, \ldots, k.$

iv) $\bigcup_{i=1}^{k} P_{\mathcal{N}_i} = P$ *and* $\bigcup_{i \neq j} \left(P_{\mathcal{N}_i} \cap P_{\mathcal{N}_j}\right) = Q \;\; i,j \in \{1, \ldots, k\}.$

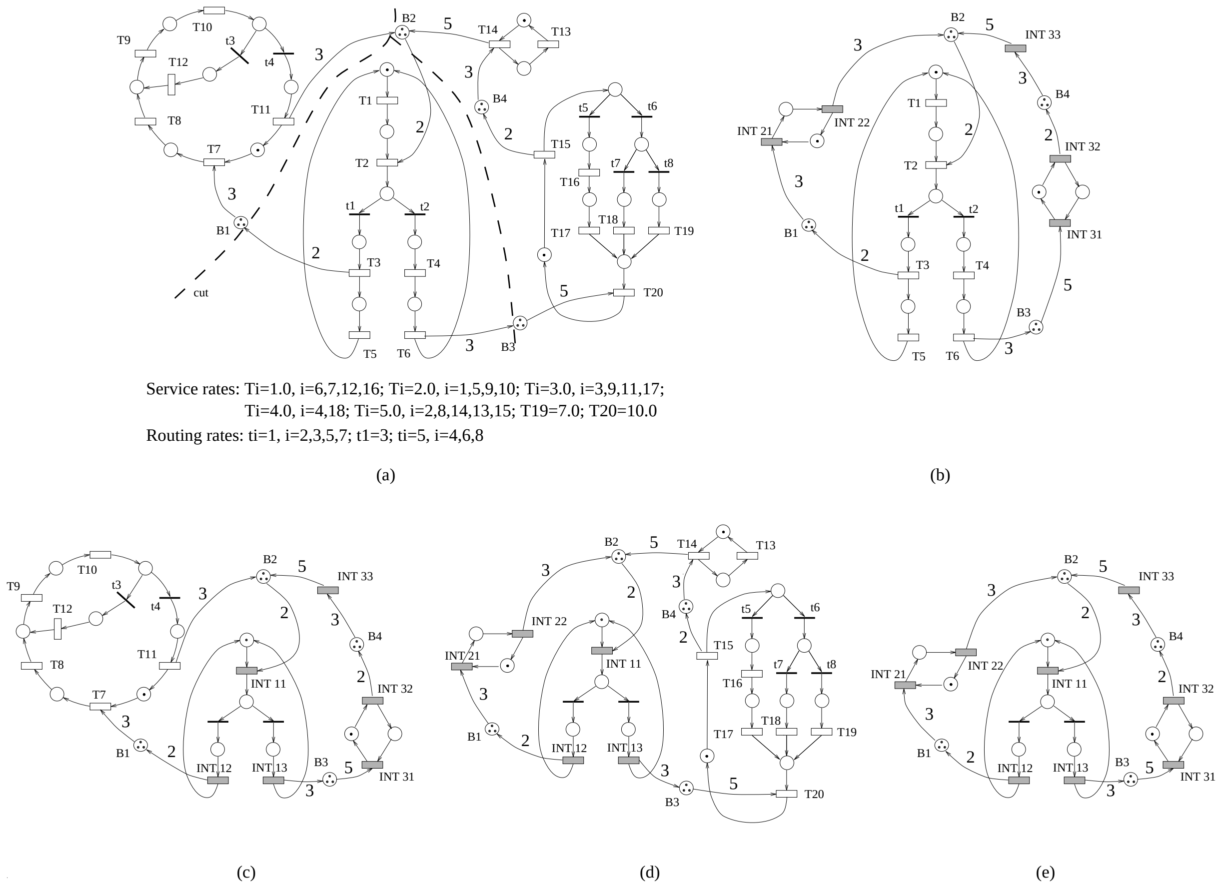

Figure 1: (a) A DSSP. Its decomposition in aggregated subsystems (b) $\mathcal{AS}_1$, (c) $\mathcal{AS}_2$, and (d) $\mathcal{AS}_3$, and (e) the basic skeleton (shadow transitions are the observable transitions summarizing the response time of SM's).

v) $Pre_{\mathcal{N}_i} = Pre|_{P_{\mathcal{N}_i} \times T_{\mathcal{N}_i}}$, $Post_{\mathcal{N}_i} = Post|_{P_{\mathcal{N}_i} \times T_{\mathcal{N}_i}}$, $i \in \{1, \dots, k\}$.

Now we define the aggregated subsystems and the basic skeleton obtained from a k-cut.

Definition 3.2 *Let $(\mathcal{N}, M_0)$ be a live and bounded DSSP, $Q \subseteq B$ a k-cut of $\mathcal{N}$ and $\mathcal{N}_i$ $i = 1, \dots, k$ the k subnets defined by Q. For each $i = 1, \dots, k$ the aggregated subsystem $\mathcal{AS}_i = (\mathcal{AN}_i, M_0^{\mathcal{AN}_i})$ is the system obtained from $(\mathcal{N}, M_0)$ by reducing every SM not included in $\mathcal{N}_i$. All buffers are maintained. The basic skeleton $\mathcal{BS} = (\mathcal{BN}, M_0^{\mathcal{BN}})$ is the system obtained from $(\mathcal{N}, M_0)$ by reducing all SM's. The initial markings of aggregated subsystems and basic skeleton maintain the initial marking of buffers and of non-reduced SM's. The initial marking of reduced SM's is the marking computed by the previous reduction technique.*

The next theorem includes the main result of this section. It relates the behaviour of a DSSP with those of the aggregated subsystems and the basic skeleton. Previously, we introduce some additional notation. For each $i = 1, \dots, k$, let TV_i be the set of observable transitions of $\mathcal{N}_i$ and PV_i the set of input places of transitions of TV_i belonging to a SM.

Theorem 3.1 *Let $(\mathcal{N}, M_0)$ be a live and bounded DSSP, $Q \subseteq B$ a k-cut of $\mathcal{N}$, $\mathcal{AS}_i = (\mathcal{AN}_i, M_0^{\mathcal{AN}_i})$, $i = 1, \ldots, k$, the aggregated subsystems, and $\mathcal{BS} = (\mathcal{BN}, M_0^{\mathcal{BN}})$ the basic skeleton derived from Q. Then:*

i) $L(\mathcal{N}, M_0)|_{TAS_i} = L\left(\mathcal{AN}_i, M_0^{\mathcal{AN}_i}\right)\Big|_{TAS_i}$, *for* $i = 1, \ldots, k$.

ii) $R(\mathcal{N}, M_0)|_{PAS_i} = R\left(\mathcal{AN}_i, M_0^{\mathcal{AN}_i}\right)\Big|_{PAS_i}$, *for* $i = 1, \ldots, k$.

iii) $L(\mathcal{N}, M_0)|_{TBN} = L\left(\mathcal{BN}, M_0^{\mathcal{BN}}\right)|_{TBN}$

iv) $R(\mathcal{N}, M_0)|_{PBN} = R\left(\mathcal{BN}, M_0^{\mathcal{BN}}\right)|_{PBN}$

Where $TAS_i = \left(\bigcup_{j \neq i} TV_i\right) \cup T_{\mathcal{N}_i}$, $PAS_i = \left(\bigcup_{j \neq i} PV_i\right) \cup P_{\mathcal{N}_i} \cup B$, $TBN = \bigcup_{i=1}^{k} TV_i$, *and* $PBN = \bigcup_{i=1}^{k} PV_i$.

Proof sketch: An aggregated subsystem $\mathcal{AS}_i$ can be computed from the original net by reducing one by one the SM's not included in $\mathcal{N}_i$. Applying at each step lemma 3.2, results *(i)* and *(ii)* follow. The same argument is valid for *(iii)* and *(iv)*. Q.E.D.

We remark that the previous result assures an "exact functional aggregation" of the original DSSP, in the sense of state space projection. In this way, good performance approximations should be expected.

3.2 Iterative response time approximation phase

The technique for an approximate computation of the throughput that we present now is, basically, a *response time approximation* method [ABS84, CCJS94]. The observable transitions of $\mathcal{N}_j$ in $\mathcal{AS}_i$ $(j \neq i)$ approximate the response time of all the subsystem $\mathcal{N}_j$. A direct (non-iterative) method to compute a constant service rates of observable transitions in order to represent the aggregation of the subnet gives, in general, low accuracy. Therefore, we are forced to define a *fixed-point search iterative process*. The proposed algorithm is the following:

select a k-cut Q;
derive $\mathcal{AS}_i$, $i = 1, \ldots, k$, and $\mathcal{BS}$;
give an initial service rate μ_t^0 for each $t \in TBN$;
$j := 0$; {counter for iteration steps}
repeat
    $j := j + 1$;
    for $i := 1, \ldots, k$
        solve the aggregated subsystem $\mathcal{AS}_i$ with
            input: μ_t^j for each $t \in TV_l$ $(l = 1, \ldots, i-1)$
                μ_t^{j-1} for each $t \in TV_l$ $(l = i+1, \ldots, k)$
            output: $\mathcal{X}_i^j$
        solve the basic skeleton system $\mathcal{BS}$ with
            input: μ_t^j for each $t \in TV_l$ $(l = 1, \ldots, i-1)$
                μ_t^{j-1} for each $t \in TV_l$ $(l = i+1, \ldots, k)$
                ratios among μ_t^0 of $t \in TV_i$ and $\mathcal{X}_i^j$
            output: scale factor of μ_t^j of $t \in TV_i$
    end for;
until convergence of $\mathcal{X}_1^j, \ldots, \mathcal{X}_k^j$;

In the above procedure, once a k-cut has been selected and given some initial values μ_t^0 for service rates of all observable transitions except those in $\mathcal{N}_1$, the underlying CTMC of aggregated subsystem $\mathcal{AS}_1$ is solved (the computation of appropriate initial values μ_t^0 is addressed below). From the solution of that CTMC, the first estimation $\mathcal{X}_1^1$ of the throughput of $\mathcal{AS}_1$ can be computed. Then, the initial estimated values of service rates of observable transitions TV_1 must be derived. To do that, we take the initial values μ_t^0 for service rates of transitions in TV_1 and we search in the basic skeleton a scale factor for all these rates such that the throughput of the basic skeleton and the throughput of $\mathcal{AS}_1$, computed before, are equal. The same procedure is executed for each aggregated subsystem in a cyclic way. Each time we solve the aggregated subsystem $\mathcal{AS}_i$ we obtain in the basic skeleton a new estimation of the observable transitions rates of TV_i.

The computation of a *scale factor* in the basic skeleton can be implemented with a linear search. Now the net system (the basic skeleton) has considerably fewer states than the original one. In each iteration of this linear search, the underlying CTMC of the basic skeleton is solved. Note that only first iteration the CTMC is completely derived. For later iterations only some values must be changed. In [BLS95], it is proved that in stochastic DSSP's (with time and marking independent rates) the throughput of any transition is a monotonic function of the rate of any other transition. In other words, if we increase (or decrease) the rate of any transition the throughput of any other transition will not be decreased (or not increase). This fact assures that the linear search in the basic skeleton (being a DSSP) has *one and only one solution*.

Now we present a procedure to compute the *initial values* μ_t^0 for service rates of observable transitions in the above algorithm. These initial values are also the relative service rates μ_t^0 of observable transitions used in the basic skeleton during all the approximation process. Consider a state machine $\mathcal{SM}_i$ of the original system. We can approximate its behaviour in *isolation* with its aggregated corresponding state machine $\mathcal{SM}_{T_v}$ in isolation. In particular, we are able to preserve not only the exact throughput of each observable transition but also the relative probabilities of observable transitions being enabled. In this way, in the reduced SM not only observable transitions are fired as they were in the original net but also they "work" the same proportion of time. As the SM is safe, what we want to preserve is the relative probabilities of the states enabling observable transitions.

If Π and Π' are the steady-state probability vectors of $\mathcal{SM}$ and $\mathcal{SM}_{T_v}$ respectively, we want that $\Pi|_{\bullet T_v}$ is proportional to Π' (because Π' must be also a probability vector). Vector Π can be efficiently computed from linear system $\Pi Q = 0$. The elements of matrix Q' (the underlying CTMC of $\mathcal{SM}_{T_v}$) are: $q'_{ij} = p'_{ij}\lambda_i,\ \ \forall i \neq j$, and $q'_{ii} = -\sum_{k=1}^{n} q'_{ij} = -p'_{ii}\lambda_i$. Where $\lambda_1, \ldots, \lambda_n$ are the unknown rates we want to compute. Since $\mathrm{rank}(Q') = n-1$, from linear system $\Pi' Q' = 0$ we can compute the rates $\lambda_1, \ldots, \lambda_n$ but for a scale factor λ (we take, for instance, $\lambda_1 = 1$). The scale factor is computed in such a way that actual throughputs of observable transitions in $\mathcal{SM}$ and $\mathcal{SM}_{T_v}$ are the same. If $\mathcal{X}_1$ is the actual throughput of observable transition t_1 in the original state machine $\mathcal{SM}$ and $\mathcal{X}'_1$ is the actual throughput of the same transition in the reduced SM with transition rates computed before (the rates normalized with $\lambda_1 = 1$), then $\lambda = \mathcal{X}_1/\mathcal{X}'_1$. Therefore, the initial service rates for observable transitions $t \in TV_i$ of state machine $\mathcal{SM}_i$ used in the iterative algorithm are $\mu_t^0 = \lambda \cdot \lambda_t$.

Now the existence and uniqueness of the solution of the entire method should be addressed. Although no formal proof gives positive answers so far to the above questions, extensive testing allows us to conjecture that there exists one and only one solution, computable in a finite number of steps, typically between 2 and 6 if the convergence criterion is that the maximum difference between the two last estimations of the throughput of any aggregated subsystem is less than 0.01%.

3.3 Numerical results

Let us present some numerical values obtained for the system in Fig. 1. The exact value of the throughput of T1 in the original system for the service and routing rates given in the figure is 0.584919. The underlying CTMC has 66816 states. In Table 1, the iterative results are shown. Columns $\mathcal{X}$(Ti) are the estimated values for throughput of transition Ti at each iteration step. Columns 'scale f.' are the scale factors modifying the previous estimated service rates, computed with the basic skeleton. Convergence of the method is usually obtained from the third iteration step. The error for this example was -0.168%. The following additional fact must be remarked: the number of states of the underlying CTMC's of $\mathcal{AS}_1$, $\mathcal{AS}_2$, $\mathcal{AS}_3$, and $\mathcal{BS}$ are 3748, 4241, 12914, and 1733, respectively.

The results obtained for alternative service and routing rates are also depicted in Table 1. If all rates are equal to 1.0 in the original system (representing a symmetric timing case), the exact throughput of T1 is 0.244811. The error of the approximated value is -0.022%. On the other hand, if a very asymmetric timing is considered (service rates differ in three orders of magnitude, as given in the third case of Table 1), the exact and the approximate values are the same (with six decimals of precision). This exceptionally good approximation is usually obtained with systems with time scale decomposition compatible with the cut.

Table 1: Iteration results for the DSSP in Fig. 1.a

Service and routing rates as given in Fig. 1.a					
$\mathcal{AS}_1$		$\mathcal{AS}_2$		$\mathcal{AS}_3$	
$\mathcal{X}$(T1)	scale f.	$\mathcal{X}$(T7)	scale f.	$\mathcal{X}$(T20)	scale f.
0.570681	1.032258	0.292835	1.085594	0.088028	2.109817
0.583688	1.024574	0.291993	1.087649	0.087597	2.098562
0.583931	1.024455	0.291967	1.087671	0.087590	2.098639
0.583934	1.024455	0.291967	1.087671	0.087590	2.098639
Error: -0.168%					

Service and routing rates equal to 1.0					
$\mathcal{AS}_1$		$\mathcal{AS}_2$		$\mathcal{AS}_3$	
$\mathcal{X}$(T1)	scale f.	$\mathcal{X}$(T7)	scale f.	$\mathcal{X}$(T20)	scale f.
0.242410	1.024770	0.081269	1.033071	0.073919	1.790746
0.244743	1.016547	0.081588	1.033610	0.073429	1.790735
0.244757	1.016523	0.081586	1.033611	0.073428	1.792880
0.244758	1.016514	0.081587	1.033697	0.073428	1.792873
Error: -0.022%					

Service rates: $\mathcal{N}_1$ all equal 100.0; $\mathcal{N}_2$ all equal 0.1; $\mathcal{N}_3$ all equal 1.0. Routing rates equal to 1.0					
$\mathcal{AS}_1$		$\mathcal{AS}_2$		$\mathcal{AS}_3$	
$\mathcal{X}$(T1)	scale f.	$\mathcal{X}$(T7)	scale f.	$\mathcal{X}$(T20)	scale f.
0.037500	1.000000	0.012500	1.000000	0.011250	1.000000
0.037500	1.000000	0.012500	1.000000	0.011250	1.000000
Error: 0.000%					

In the extensive numerical experiments that we have made, convergence was achieved after no more than five iteration steps and the error was no more than 3%.

4 DDSP's reducible to weighted T-systems

The technique in the previous section is based on the (partial) reduction of all SM's of a general DSSP. If the SM's of a DSSP are not large, then the previous technique may not lead to a significant state space reduction. In this section we introduce a more general reduction on a subclass of DSSP's that we will call *reducible to weighted T-systems* (WTS's are the weighted generalization of marked graphs; see [TCWCS92]). In particular we assume in this section that the buffers are also input private as in [Sou93](i.e., only a SM gives tokens to it). Moreover, since the functional units of a DSSP are mono-marked SM's, it can be said that buffers with origin and destination to a single functional unit can be ignored *without loosing expressive power*: (a) To preserve consistency and conservativeness, purely private buffers should be structurally implicit [CS91b], and (b) either the buffer is implicit (i.e., it does not constraint at all the behaviour of the global system) or not, and in the last case it kills the global system (and we are looking for live systems only!).

The main differences with respect to the technique in previous section are that now the reduction is more global in the sense that not only full reduction of SM's is considered, but also buffers are reduced. The price to be payed is that the exact projection of the functional behaviour of the original system on the preserved nodes is not longer true. The following properties are preserved: boundedness (essential to the finiteness of the underlying CTMC), liveness (for bounded, strongly connected DSSP's equivalent to deadlock-freeness [CCST94], thus non-null throughput preservation) and the existence of home states (to preserve the CTMC ergodicity [CCST94]). The descomposition phase is performed in *polynomial time*.

The reduction from the original system towards the computation of the aggregated subsystems is explained in two steps: (a) Each SM (with some properties assumed) is reduced to a single transition, and (b) the obtained skeleton (a WTS) is reduced to a given standard scheme.

In order to simplify notations and statements, in the sequel we assume the basic single input-single output (SISO) cut, splitting the original system into two parts through two buffers. The technique is a mild generalization of that introduced in [JSS92], where the SISO-cut is considered but limited to marked graphs (i.e., neither decisions nor bulk services or arrivals).

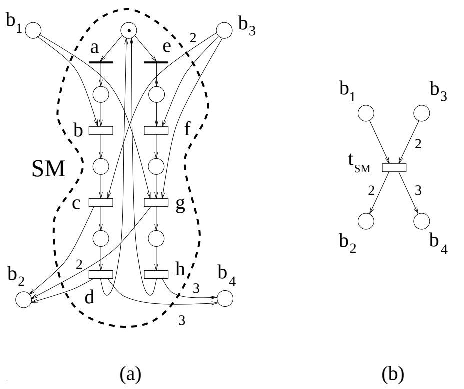

Figure 2: (a) Part of the original system. (b) The SM is reducible to a transition preserving boundedness, liveness, and the existence of home states for the full system.

4.1 State machine reduction to transition

The idea (see Fig. 2.a) is to reduce a SM to a single transition (with single-server semantics!) provided a unique buffer-neighbouring exists in a certain strict sense: depending of the way in which the buffers are connected to the machine and on its initial state, the machine can be reduced or not.

Let $\mathcal{N} = (P, T, Pre, Post)$ be a DSSP net, $\mathcal{N}_{\mathcal{SM}} = (P_{\mathcal{SM}}, T_{\mathcal{SM}}, \Pi^-, \Pi^+)$ a SM subnet of $\mathcal{N}$ (i.e. $\Pi^- = Pre|_{P_{\mathcal{SM}} \times T_{\mathcal{SM}}}, \Pi^+ = Post|_{P_{\mathcal{SM}} \times T_{\mathcal{SM}}}$), $B_{in} = \{b_i | b_i^\bullet \subseteq T_{\mathcal{SM}}\}$ and $B_{out} = \{b_j | {}^\bullet b_j \subseteq T_{\mathcal{SM}}\}$ the input and output buffers of $\mathcal{SM}$ in $\mathcal{N}$. The reduction of $\mathcal{SM}$ leads to a transition $t_{\mathcal{SM}}$ with ${}^\bullet t_{\mathcal{SM}} = B_{in}$ and $t_{\mathcal{SM}}{}^\bullet = B_{out}$, and some weightings $\widetilde{Pre}(b_i, t_{\mathcal{SM}})$ for $b_i \in B_{in}$ and $\widetilde{Post}(b_j, t_{\mathcal{SM}})$ for $b_j \in B_{out}$.

Two conditions should be imposed in the reduction of $\mathcal{SM}$ to get some "similarities" in the behaviour of the original and the reduced systems: (1) a linear token conservation property that allows the calculation of $\widetilde{Pre}(b_i, t_{\mathcal{SM}})$ and $\widetilde{Post}(b_j, t_{\mathcal{SM}})$ (see section 4.1.1) and (2) a token withdrawal (from input buffers)-token throw in (to output buffers) property from the initial marking of the SM (see section 4.1.2), that allows the preservation of liveness and the existence of home states.

4.1.1 The linear token conservation property

Firing $t_{\mathcal{SM}}$ in the reduced system, $M[t_{\mathcal{SM}}\rangle M'$, produces a token variation $\mu = M' - M$. It is obvious (assume, for example, $b_i \in B_{in}$) that

$$\mu(b_i) = k_{ij}\mu(b_j) \tag{2}$$

where:

- $k_{ij} = \widetilde{Pre}(b_i, t_{\mathcal{SM}}) / \widetilde{Pre}(b_j, t_{\mathcal{SM}})$, if $b_j \in B_{in}$.
- $k_{ij} = -\widetilde{Pre}(b_i, t_{\mathcal{SM}}) / \widetilde{Post}(b_j, t_{\mathcal{SM}})$, if $b_j \in B_{out}$.

For the computability of $\widetilde{Pre}$ and $\widetilde{Post}$, it is clear that equations similar to (2) should be written in the original system for "arbitrarily long" sequences projected on $T_{\mathcal{SM}}$:

$$\mu(b_i) = k_{ij}\mu(b_j) + \sum_{p_k \in P_{\mathcal{SM}}} \lambda_k \mu(p_k) \tag{3}$$

where k_{ij} is defined as in (2), while $\sum_{p_k \in P_{\mathcal{SM}}} \lambda_k \mu(p_k)$ represents the contribution of the marking variation induced on the places of the SM. Obviously if the firing sequence projected on the $\mathcal{SM}$ leads to a T-semiflow of the state machine, $\mu(p_k) = 0$.

Property 4.1 *Let* $C^* = \begin{pmatrix} C_{\mathcal{SM}} \\ C_{B_{in}} \\ C_{B_{out}} \end{pmatrix}$ *be the incidence submatrix of the full net* $\mathcal{N}$ *reduced to* $T_{\mathcal{SM}}$, $P_{\mathcal{SM}}$, *input buffers* (B_{in}), *and output buffers* (B_{out}). *Then the relative weightings for the reduced net, are computable iff* $\text{rank}(C^*) = n_{\mathcal{SM}} (= |P_{\mathcal{SM}}|)$.

Proof: (i) $\text{rank}(C_{\mathcal{SM}}) = n_{\mathcal{SM}} - 1$, because $\mathcal{SM}$ is a state machine. Moreover, if $C_1 = \begin{pmatrix} C_{\mathcal{SM}} \\ C(b_i) \end{pmatrix}$, $\text{rank}(C_1) = n_{\mathcal{SM}}$, because b_i (any input, non-output buffer of $\mathcal{SM}$) cannot be a linear combination of the places of the $\mathcal{SM}$.

Consider now $C_2 = \begin{pmatrix} C_{\mathcal{SM}} \\ C(b_i) \\ C(b_j) \end{pmatrix}$, where b_j is another input or output buffer of $\mathcal{SM}$. If $\text{rank}(C_2) = n_{\mathcal{SM}} + 1$, b_j is linearly independent of b_i against (2). Therefore $\text{rank}(C_2) = n_{\mathcal{SM}}$. By induction on the remaining buffers in vecinity of $\mathcal{SM}$ we conclude that $\text{rank}(C^*) = n_{\mathcal{SM}}$.

(ii) If $\text{rank}(C^*) = n_{\mathcal{SM}}$, $\text{rank}(C_{\mathcal{SM}}) = n_{\mathcal{SM}} - 1$ and $\text{rank}(C_1) = n_{\mathcal{SM}}$, it is clear that for any buffer in vecinity of $\mathcal{SM}$, except b_i we can write: $C(b_j) = \sum_{p_k \in P_{\mathcal{SM}}} \lambda_k C(p_k) + k_{ij} C(b_i)$. Postmultiplying by any firable $\vec{\sigma}_{\mathcal{SM}}$ and taking into account the state equation of a net system, (3) is found. Q.E.D.

A dual perspective of the above rank property is the following. Let $X \geq 0$ be such that $C_{\mathcal{SM}} \cdot X = 0$ (i.e. a T-semiflow of $\mathcal{SM}$). Therefore if $C(b_i) \cdot X = \alpha$, then $C(b_j) \cdot X = \alpha K_{ij}$, and so on. In words: Because minimal T-semiflows of SM's are cycles, all cycles of a SM reducible to a transition induce proportional effects (eventually null) on the set of input and output buffers.

From the above reasoning, the value of $\widetilde{Pre}(B_{in})$ or $\widetilde{Post}(B_{out})$ is not fully characterized, but it is known that:

- $\widetilde{Pre}(b_i) = \alpha$
- $\widetilde{Pre}(b_j) = \alpha \cdot k_{ij} \;\; \forall b_j \in B_{in} \setminus \{b_i\}$
- $\widetilde{Pre}(b_q) = -\alpha k_{iq} \;\; \forall b_q \in B_{out}$

The value of α is computed in the next section for the case where the $\mathcal{SM}$ is really reducible.

4.1.2 Token withdrawal and token throw in per cycle

Looking at Fig. 2.b, it is clear that the SM reduction makes sense if, starting in an initially marked place, for all cycles of the SM (i.e., its minimal T-semiflows), all token withdrawal from the input buffers occur before the first token throw in to the output buffers, and additionally the number of withdrawals is independent of the path followed to the first throw in to the output buffers.

Let $\overline{\mathcal{N}_{\mathcal{SM}}}$ be the net obtained from $\mathcal{N}_{\mathcal{SM}}$ by labelling each interface transition t with "$-b_i$" if b_i is an input buffer of t and "$+b_j$" if b_j is an output buffer of t. Moreover, the output places of transitions labelled with "$+b_k$" are blocked.

Definition 4.1 *The state machine $\mathcal{SM}$, initially marked at p_q, is reducible to $t_{\mathcal{SM}}$ if:*

i) rank $\begin{pmatrix} C_{\mathcal{SM}} \\ B_{in} \\ B_{out} \end{pmatrix} = |P_{\mathcal{SM}}|$ *(pure structural condition from property 4.1).*

ii) All transitions labelled with "$-$" in $\overline{\mathcal{N}_{\mathcal{SM}}}$ are reachable (without going through a blocked place) from p_q, and in all elementary paths from p_q to a blocked place the same number of ocurrences of "$-b_k$" appears. The value of α is the number of occurrences of label "$-b_i$" in the paths from p_q to a blocked place.

The condition ii) in the above definition can be computed also in *polynomial time* using, for example, a small variation of a single node to several nodes path following Floyd's algorithm [AHU83].

For instance, for the net in Fig. 3, the SM defined by $T_{\mathcal{SM}} = \{t_5, t_6, T_{14}, T_{15}, T_{16}, T_{17}\}$ should be labelled with $-2B_{11}$ in $\{T_{15}, T_{16}\}$, $-2B_{13}$ in T_{14} and $-B_{13}$ and $+2B_{12}$ in T_{17}. Thus the input place of $t_5 - t_6$ is blocked. From the initial marking T_{16} is not reachable, thus another marking should be considered. Firing backwards t_6 (a persistent firing) does not introduce spurious behaviours and now all cycles appear to be labelled with $-2B_{11}$ and $-3B_{13}$ (and $+2B_{12}$). Thus $\widetilde{Pre}(B_{11}, t_{\mathcal{SM}}) = 2$, $\widetilde{Pre}(B_{13}, t_{\mathcal{SM}}) = 3$ and $\widetilde{Post}(B_{12}, t_{\mathcal{SM}}) = 2$.

Theorem 4.1 *Let $(\mathcal{N}, M_0)$ be a DSSP and $(\widetilde{\mathcal{N}}, \widetilde{M_0})$ another one obtained through the reduction of a SM to a transition, according with the conditions above. $(\widetilde{\mathcal{N}}, \widetilde{M_0})$ is conservative, live, and it has home states if and only if $(\mathcal{N}, M_0)$ does. Moreover, the transformation preserves the marking bounds of the buffers.*

Proof: (1) The reduction is a particular case of the elimination of transitions $T_{\mathcal{SM}}$ with the standard P-semiflow calculation algorithm [CS91a]. Therefore conservativeness is preserved.

(2) According with the reduction conditions, a SM is reducible iff for any T-semiflow (cycle) with external effect the same amount of tokens are withdrawn from the input buffers (possibly in different order) before starting the throw in of tokens on the output buffers, and this also for a quantity independent of the cycle being executed. Thus liveness and existence of home states can be equally considered in a two transitions SM: After the initially marked place, a transition resumes all tokens withdrawal from input buffers; its output place, π, is the SM precondition to a transition in which the postcondition resumes all tokens throw in the output buffers. Both reductions naturally preserve liveness and the existence of home states. Moreover, π can be eliminated fusing both transitions into a single one, also preserving liveness and the existence of home states.

(3) The bounds of the buffers are preserved due to the two phases on the aggregated behaviour, because before giving the first token all consumptions controlled from the SM are done and these input buffers have it as the unique destination. Q.E.D.

The rule presented so far reduces a SM, with the characteristics already considered, to a transition. This reduction technique does not preserve the projection either the languages or the markings over the preserved nodes. For instance, in Fig. 3, the SM defined by $T_{\mathcal{SM}} = \{t_5, t_6, T_{14}, T_{15}, T_{16}, T_{17}\}$ if we fire T_{15} and then T_{14} two tokens are removed from buffer B_{11} and then two tokens from buffer B_{13}. When the state machine is reduced to a transition it is imposible to separate these two states or firings. If this reduction is done for all the SM's in the DSSP we obtain a bounded and live WTS, for which it is also known, from other developments, that there exist home states [TCWCS92]. Moreover, the bounds of all places of the WTS are equal to the bounds of the corresponding buffers of the original DSSP.

4.2 WTS-reduction for SISO-cuts

We consider the subclass of DSSP's such that, according to the procedure previously presented, every SM has been reduced to a single transition and, additionally, buffers are (not only output but also) input private. In this case, the remaining system after reduction of SM's is a WTS.

In order to simplify the presentation we restrict ourselves to a WTS decomposition obtained from a *single input-single output cut* (SISO-cut) through places (buffers of the original DSSP). A SISO-cut is defined by two buffers whose deletion generates a partition of the system into two subsystem [JSS92] (see Fig. 3). Assume, to simplify the exposition, that each buffer of the cut is the only input (output) buffer of its output (input) $\mathcal{SM}$. The technique presented here can be generalized to multiple input-multiple output cuts.

Once a SISO-cut has been decided, the main problem is to derive the aggregated subsystems where either the left or the right part of the original system is reduced to a "minimum number" of nodes (the basic skeleton will be obtained after reduction of both parts). The method is a generalization of that presented in [JSS92] for marked graphs. In Fig. 3.a, the left part from the output SM of B8 to the input SM of B1 is replaced with a single place (see Fig. 3.c). The input and output weights of that place are computed to preserve the *gain* of the paths in original system from the output SM of B8 to the input SM of B1. The gain represents, from a structural point of view, the average number of tokens produced in the output place per each single token taken from the input place. In principle, the same reduction would be done for the right part.

After this reduction, the projection of the state space of the original system on the non-reduced part is not preserved in the aggregated subsystem (this fact differs from the marked graph case [CCJS94]). In other words, spurious markings can be made reachable after the aggregation.

An easy (structural) way to reduce some of the spurious markings is to preserve, in addition to the gain, what we call the maximum *resistance* of the aggregated part. The resistance of a path from a transition to another transition is a structural property of a path related to the minimum number of firings of the first transition that is needed to fire once the second. For the system in Fig. 3.a, to preserve the resistance of the subsystem in the right hand side, we add the immediate transition $r2$ with its corresponding input and output place to obtain the aggregated subsystem in Fig. 3.b.

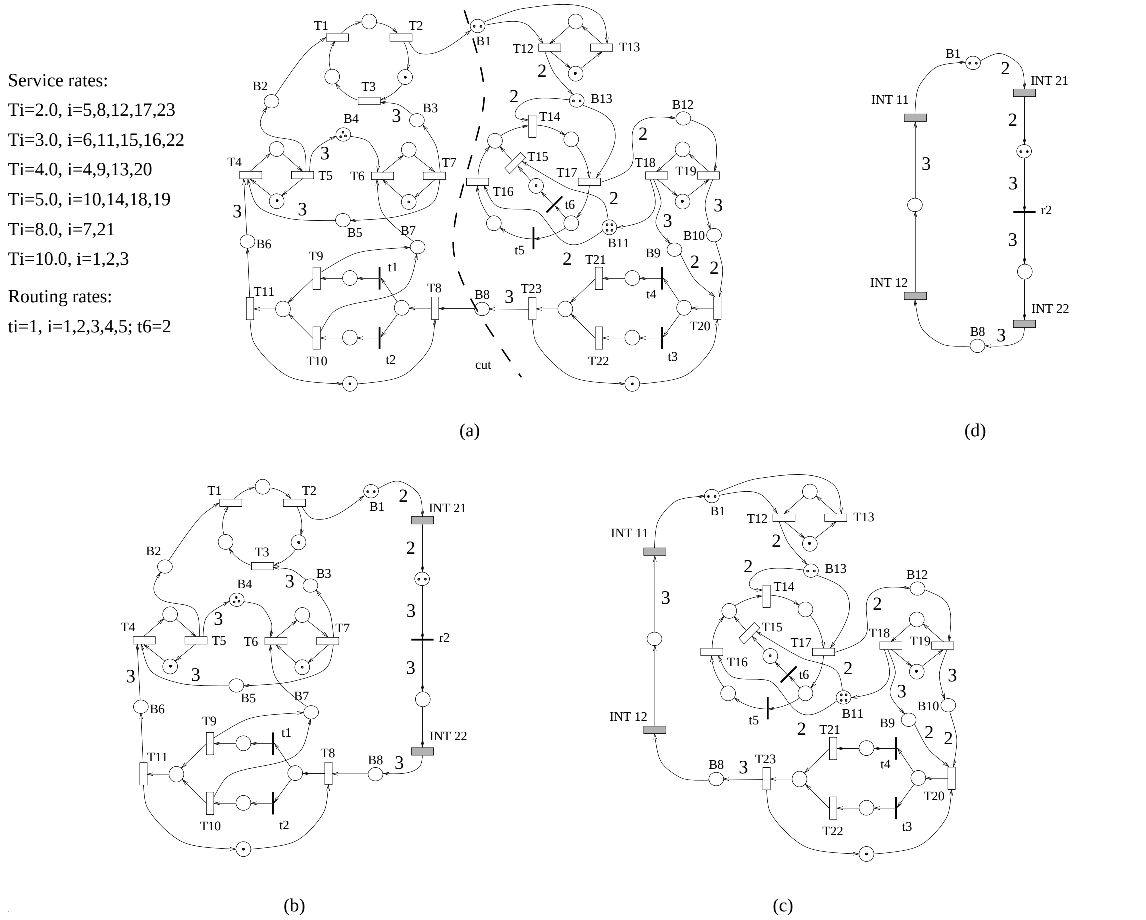

Figure 3: (a) A DSSP reducible to a WTS. Its decomposition in aggregated subsystems (b) $\mathcal{AS}_1$ and (c) $\mathcal{AS}_2$, and (d) the basic skeleton.

Sometimes, the place preserving the gain is implicit with respect to the path composed by the immediate transition preserving the resistance and its input and output place (this is the case for the aggregated subsystem in Fig. 3.b), thus it can be deleted. Sometimes, in order to preserve the resistance, it is not necessary to add the immediate transition with its input and output place because the place added for preserving the gain summarizes also the resistance (as in Fig. 3.c). In general, it can be necessary to include both the place for preserving the gain and the immediate transition for preserving the resistance for summarizing each side of the system.

The formal definitions of gain and resistance follow. Let $wp = t_0p_1t_1p_2\ldots p_nt_n$ denote a weighted path with $Pre(p_i, t_{i-1}) = x_i$ and $Post(p_i, t_i) = y_i$, $i = 1, \ldots, n$.

Definition 4.2 *The gain of wp is* $g(wp) = \prod_{i=1}^{n} \frac{x_i}{y_i}$. *The resistance of wp is* $r(wp) = \max_{i=1,\ldots,n} \prod_{j=1}^{i} \frac{y_j}{x_j}$.

The next lemma shows that the proposed reduction preserves the gain and the resistance.

Lemma 4.1 *Let wp be a weighted path with* $r(wp) = \frac{r_1}{r_2}$ $(\gcd(r_1, r_2) = 1)$, $g(wp) = \frac{g_1}{g_2}$ $(\gcd(g_1, g_2) = 1)$. *Let* $d = \gcd(r_1g_1, r_2g_2)$. *Then, the weighted path* $\overline{wp} = t_0q_1t_{0,n}q_2t_n$ *with* $Pre(q_1, t_0) = r_2, Post(q_1, t_{0,n}) = r_1, Pre(q_2, t_{0,n}) = \frac{r_1g_1}{d}, Post(q_2, t_n) = \frac{r_2g_2}{d}$ *has the same gain and resistance than wp.*

Proof: $g(\overline{wp}) = \frac{r_2 \frac{r_1g_1}{d}}{r_1 \frac{r_2g_2}{d}} = \frac{g_1}{g_2}$. $r(\overline{wp}) = \max\{\frac{r_1}{r_2}, \frac{r_1}{r_2}\frac{r_2g_2}{r_1g_1}\} = \max\{r(wp), g(wp)^{-1}\} = r(wp)$ because by definition 4.2 for every weighted path $r \geq g^{-1}$. Q.E.D.

The problem now is to compute the gain and maximum resistance of a set of paths with the same initial and last transition. Note that all these paths must have the same gain (since we are considering consistent WTS's). We will use a slight modification of Floyd's algorithm to solve the *all-pairs shortest paths* problem [AHU83]. We derive from the part of the WTS that we are going to aggregate a directed graph in which transitions are the nodes and places are the arcs. The cost function on arcs needed for our purpose includes the gain and the resistance between two interconnected transitions. The algorithm computes for every ordered pair of transitions the maximum resistance of a path joining the pair of transitions (the solution is 0 for a pair if there does not exist such a path). Floyd's algorithm can be used taken into account the following expression of the resistance of a path as function of the resistance and gain of two subpaths.

Lemma 4.2 *Let wp be a weighted path with t_0, t_n its first and last transitions. Let t_i be other transition of wp that splits it in two subpaths denoted by $wp_{0,i}$ and $wp_{i,n}$, respectively. Then:*

i) $g(wp) = g(wp_{0,i})g(wp_{i,n})$

ii) $r(wp) = \max\left\{ r(wp_{0,i}), \dfrac{r(wp_{i,n})}{g(wp_{0,i})} \right\}$

Proof: (i) is trivial.
(ii) $r(wp) = \max_{k=1,\ldots,n}\left\{\prod_{j=1}^{k}\frac{y_j}{x_j}\right\} = \max\left\{\max_{k=1,\ldots,i}\left\{\prod_{j=1}^{k}\frac{y_j}{x_j}\right\}, \max_{k=i+1,\ldots,n}\left\{\prod_{j=1}^{k}\frac{y_j}{x_j}\right\}\right\} = \max\left\{r(wp_{0,i}), \prod_{j=1}^{i}\frac{y_j}{x_j}\max_{k=i+1,\ldots,n}\left\{\prod_{j=i+1}^{k}\frac{y_j}{x_j}\right\}\right\} = \max\left\{r(wp_{0,i}), \frac{r(wp_{i,n})}{g(wp_{0,i})}\right\}$ Q.E.D.

Note that there is a transition t of wp such that $r(t_0 t) = r(wp)$ so the reduced path $\overline{wp}$ can be seen as the joining of the implicit place from t_0 to t with implicit place from t to t_n.

With respect to the initial marking of the aggregated subsystems, it can be defined from the following concept of *weighted marking.*

Definition 4.3 *Let $(wp, M_0[wp])$ be a weighted path as in Def. 4.2, with $M_0[p_1], \ldots, M_0[p_n]$ tokens in its places and $g(wp) = \frac{g_1}{g_2}$ ($\gcd(g_1, g_2) = 1$). The weighted marking of wp is $WM(wp, M_0[wp]) = g_1 \sum_{i=1}^{n} \dfrac{\prod_{j=1}^{i-1} y_j}{\prod_{j=1}^{i} x_j} M[p_i]$.*

The marking of aggregated subsystems is as follows. The marking of preserved places is the same that in the original system. The marking of the place p summarizing the gain is the minimum weighted marking of a path joining the interface transitions. As this weighted marking can be non-integer, it can be (optimistically) rounded up (to avoid killing the subsystem) or (pessimistically) rounded down (after checking the subsystem is still live). If the path summarizing the maximum resistance is $t_0 q_1 t_{0,n} q_2 t_n$ then $t_{0,n}$ corresponds to a transition of a path in the original system with maximum resistance. Let $wp = t_0 p_1 t_1 \ldots p_n t_n$ be the path of minimum weighted marking among the paths which have the maximum resistance. If t_i is the transition such that $r(t_0 p_1 \ldots p_i t_i) = r(wp)$, then $M[q_1] = \lceil WM(t_0 \ldots t_i, M_0[t_0 \ldots t_i]) \rceil$ and $M[q_2] = \lceil WM(t_i \ldots t_n, M_0[t_i \ldots t_n]) \rceil$.

The reduction technique presented in this section does not preserve, in general, the projection of state spaces of the original system. Nevertheless, the following result states the preservation of some fundamental logical properties.

Theorem 4.2 *Let $(\mathcal{N}, M_0)$ be a live and bounded DSSP reducible to a WTS, $Q \subseteq P$ a SISO-cut of $\mathcal{N}$ and $\mathcal{AS}_1, \mathcal{AS}_2$ and $\mathcal{BS}$ the aggregated subsystems and the basic skeleton defined by Q, preserving the gain and the resistance. Then, $\mathcal{AS}_1, \mathcal{AS}_2$, and $\mathcal{BS}$ are live and bounded and they have home state.*

Proof: i) To obtain an aggregated subsystem we take $\mathcal{N}$ and we reduce some SM's to a single transition. By theorem 4.1 This reduction preserves liveness and boundedness. Then we can add the reduced paths. As these reduced paths are implicit we preserve liveness and boundedness. If we eliminate the rest of paths to obtain the aggregated subsystem we only eliminate restrictions to the output transition, so liveness is preserved. As the gain of reduced paths is the same than original ones, boundedness is also preserved. With respect to home state existence, it can follows from the fact that live and bounded WTS's have home state [TCWCS92] Q.E.D.

Concerning the iterative approximation phase, a similar scheme to that in the previous section can be applied. Now, the response time approximation of each subsystem is summarized at each iteration step

Table 2: Iteration results for the DSSP in Fig. 3.a

Service and routing rates as given in Fig. 3.a				Service and routing rates equal to 1.0			
$\mathcal{AS}_1$		$\mathcal{AS}_2$		$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)	$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)
0.274443	0.343932	0.248674	0.722283	0.090540	0.101778	0.079229	0.208813
0.253678	0.355383	0.253769	0.723678	0.080421	0.104019	0.080453	0.209198
0.253753	0.355346	0.253753	0.723678	0.080450	0.104014	0.080450	0.209198
0.253753	0.355346	0.253753	0.723678	0.080450	0.104014	0.080450	0.209198
Error: 0.132%				Error: 1.199%			

Service rates: $\mathcal{N}_1$ all equal 100.0; $\mathcal{N}_2$ all equal 0.1 Routing rates equal to 1.0			
$\mathcal{AS}_1$		$\mathcal{AS}_2$	
$\mathcal{X}$(T2)	rate(INT 11)	$\mathcal{X}$(T23)	rate(INT 22)
0.077193	8.353201	0.023543	0.024763
0.023543	8.353201	0.023543	0.024763
0.023543	8.353201	0.023543	0.024763
Error: 0.004%			

only with the output transition of the corresponding aggregated part. The rate of that transition is tuned up by using the basic skeleton. In this way, the service time of the input transition of the aggregated part can be fixed at the beginning (with, for instance, the mean cycle time of the input state machine to that part, considered in isolation).

4.3 Numerical example

Let us present some numerical values obtained for the system in Fig. 3. The exact value of the throughput of T2 in the original system for the service and routing rates given in the figure is 0.253419 (single-server semantics is assumed). The underlying CTMC has 57288 states. In Table 2, the iterative results are shown. Columns 'rates(INT i)' are the computed rates of transition i computed with the basic skeleton. This transitions summarize the successive response time approximations of the corresponding aggregated subsystems. Convergence of the method is also usually obtained from the third iteration step. The error for this example was 0.132% and the number of states of the underlying CTMC's of $\mathcal{AS}_1$, $\mathcal{AS}_2$, and $\mathcal{BS}$ are 6854, 2573, and 133, respectively.

The results obtained for alternative service and routing rates are also depicted in Table 2. If all rates are equal to 1.0 in the original system (representing a symmetric timing case), the exact throughput of T2 is 0.079497. The error of the approximated value is 1.199%. On the other hand, if a very asymmetric timing is considered (service rates differ in three orders of magnitude, as given in the third case of Table 2), the exact throughput of T2 is 0.023542 leading up to an error of 0.004%. As in the example of the previous section, an exceptionally good approximation is obtained for very asymmetric systems with respect to timing.

After extensive numerical examples, we always obtained convergence of the iterative method in four or five steps. The state spaces are usually reduced in more than one order of magnitude and with very little error (less than 3% in most cases). Even if a seed is needed to initiate the iteration, our experience is that the method seems to be robust with respect to the value of the seed.

5 Conclusions

Two complementary aggregation techniques were presented as a basis for system decomposition. The first technique was based on a reduction of state machines, achieved by preserving the interface transitions, called observable. The reduction preserves the projection of firing sequences on the observable transitions and, even more, the branching probabilities among these transitions. The original system is decomposed with a cut defined through buffers. Then, each aggregated subsystem is obtained by reducing a different subset of state machines. In the second method, valid only for a subclass of DSSP's, each state machine of a part is fully reduced to a single transition (interfaces are not preserved), leading to a weighted T-system. And moreover, global aggregations of buffers and transitions are performed to increase the state space reduction. In this case, the obtained aggregated subsystems do not maintain projected state spaces, but basic functional porperties like boundedness, liveness, and the existence of home states are preserved. Both aggregations and decomposition can be done in polynomial time on the net size.

After the decomposition phase, a fixed-point search iterative process is used to approximate the throughput. Extensive numerical examples have shown that the iterative method converges in three to five steps. The state spaces are usually reduced in more than one order of magnitude and with very little error (less than 3% in most cases). In the second aggregation technique, where a seed is needed to initiate the iteration, our experience is that the method seems to be truly robust with respect to the value of the seed.

For the sake of clearness, both aggregation techniques were presented in a separate way. Nevertheless, they can be used in a complementary way for the reduction of large systems.

References

[ABS84] S. C. Agrawal, J. P. Buzen, and A. W. Shum. Response time preservation: A general technique for developing approximate algorithms for queueing networks. In *Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 63–77, Cambridge, MA, August 1984.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[BI82] S. Balsamo and G. Iazeolla. An extension of Norton's theorem for queueing networks. *IEEE Transactions on Software Engineering*, 8(4):298–305, July 1982.

[BLS95] F. Baccelli, Z. Liu, and M. Silva. Local and global monotonicities of stochastic Petri nets. Research report, 1995. In preparation.

[CCJS94] 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.

[CCST94] J. Campos, J. M. Colom, M. Silva, and E. Teruel. Functional and performance analysis of cooperating sequential processes. In O.J. Boxma and G.M. Koole, editors, *Performance Evaluation of Parallel and Distributed Systems: Solution Methods*, volume 106 of *Tract*, pages 233–251. Centrum voor Wiskunde en Informatica, Amsterdam, 1994.

[CS91a] J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. A comparative study of algorithms for computation of minimal p-semiflows. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 79–112. Springer-Verlag, Berlin, 1991.

[CS91b] 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-Verlag, Berlin, 1991.

[JSS92] H. Jungnitz, B. Sánchez, and M. Silva. Approximate throughput computation of stochastic marked graphs. *Journal of Parallel and Distributed Computing*, 15:282–295, 1992.

[Mur89] T. Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[Rei82] W. Reisig. Deterministic buffer synchronization of sequential processes. *Acta Informatica*, 18:117–134, 1982.

[RTS95] L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: The case of DSSP. Research Report GISI-RR-95-1, Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, Spain, January 1995. Submitted for publication.

[Sil93] M. Silva. Introducing Petri nets. In *Practice of Petri Nets in Manufacturing*, chapter 1. Chapman & Hall, 1993.

[Sou93] Y. Souissi. Deterministic systems of sequential processes: A class of structured Petri nets. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 406–426. Springer-Verlag, Berlin, 1993.

[TCWCS92] E. Teruel, P. Chrząstowski-Wachtel, J. M. Colom, and M. Silva. On weighted T-systems. In K. Jensen, editor, *Application and Theory of Petri Nets 1992*, volume 616 of *Lecture Notes in Computer Science*, pages 348–367. Springer-Verlag, Berlin, 1992.