Integrating TwoTowers and GreatSPN

M. BERNARDO
Università di Torino, Italy

N. BUSI
Università di Bologna, Italy

M. RIBAUDO
Università di Torino, Italy

Abstract

We describe the integration of the EMPA based software tool TwoTowers and the GSPN based software tool GreatSPN, in order to fully realize a multiparadigm approach to the functional and performance modeling and analysis of concurrent and distributed systems. The integration is achieved via an improved semantics mapping EMPA terms onto GSPN models.

1 Introduction

In [2] a formal approach to the modeling and analysis of concurrent systems has been proposed. The objective is to allow both functional and performance aspects of systems to be considered since the early stages of their design, so that malfunctionings and inefficiency can be detected from the very beginning thereby avoiding project cost increases that would be caused by their late discovery. One of the peculiarities of this approch is the integration of two different description techniques – stochastic process algebras (SPAs) and stochastic Petri nets (SPNs) – which is motivated by the possibility of exploiting the strengths of both. On the one hand, process algebras offer a linguistic support to system modeling which is emphasized by their compositional nature. On the other hand, Petri nets provide a truly concurrent framework where it is possible to analyze functional and performance system properties without constructing the underlying state space.

To integrate the two formalisms, a net semantics for algebraic terms should be provided. In this paper we focus on EMPA (Extended Markovian Process Algebra) [2] and on GSPNs (Generalized Stochastic Petri Nets) [1] for two reasons. First, the structure of EMPA actions has been strongly influenced by the structure of GSPN transitions, so the mapping from EMPA terms to GSPNs should be straightforward. Second, analysis tools – TwoTowers [2] and GreatSPN [5] – have been developed for both formalisms.

This paper is organized as follows. In Sect. 2 we present EMPA and GSPNs. In Sect. 3 we define an extension of the net semantics proposed in [3] – which improves earlier approaches – to our stochastically timed framework. In Sect. 4 we introduce TwoTowers and GreatSPN, then we discuss their integration through the implementation of the previously defined net semantics. In Sect. 5 we provide an example to illustrate the advantages of exploiting GSPN models for the analysis of EMPA descriptions. Finally, Sect. 6 concludes the paper with a discussion of related and future work. [1]

2 Background

Extended Markovian Process Algebra. Each action $<a,\tilde{\lambda}>$ in EMPA [2] is composed of a type and a rate. Based on rates, an action is classified as exponential if its rate is a positive real number, immediate if its rate is infinite (denoted $\infty_{l,w}$ with priority level l and weight w), and passive if its rate is left unspecified (denoted $*$). The set of actions is given by $Act = AType \times ARate$ where $AType$ is the set of types, including τ for invisible actions, and $ARate = \mathbb{R}_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathbf{N}_+ \wedge w \in \mathbb{R}_+\}$, is the set of rates.

Definition 1 *The set $\mathcal{L}$ of process terms of EMPA is generated by the syntax*

$$E ::= \sum_{i \in I} <a_i,\tilde{\lambda}_i>.E_i \mid E/L \mid E[\varphi] \mid E \parallel_S E \mid A$$

where I is a finite set of indexes, $L,S \subseteq AType - \{\tau\}$, φ is an action type relabeling function, and A is a constant. We denote by $sort(E)$ the set of action types occurring in E and by $\mathcal{G}$ the set of closed and guarded terms of $\mathcal{L}$. ■

The functional abstraction E/L, functional relabeling $E[\varphi]$ and parallel composition $E_1 \parallel_S E_2$ operators are said *static* operators. A term is said *sequential* if its outermost operator is the guarded alternative composition $\sum_{i \in I} <a_i,\tilde{\lambda}_i>.E_i$.

The integrated interleaving semantics of EMPA term $E \in \mathcal{G}$, denoted $I[\![E]\!]$, is represented by a labeled transition system (LTS) whose states are in correspondence with terms and whose transition labels are actions. The formal rules to generate such a LTS according to the meaning of the operators are shown in [2].

Generalized Stochastic Petri Nets. GSPNs [1] are an extension of classical Petri nets which comprise inhibitor arcs, exponential transitions with marking dependent rates, immediate transitions equipped with priorities and marking dependent weights. We introduce a slightly different definition of GSPN where inhibitor arcs are removed and passive transitions (depicted as black boxes) are added. This is

[1] Due to lack of space, the reader is referred to [3] for a more explanatory description of our net semantics as well as a comparison with earlier approaches to the definition of net semantics for process algebras.

because, when defining the net semantics for EMPA, inhibitor arcs will not be necessary whereas passive actions will be mapped to passive transitions. [2]

Definition 2 *A GSPN is a tuple* $(P, AType \times ARate^{\mathcal{M}(P)}, T, M_0)$ *where:*

- P *is a set of places.*
- $T \subseteq \mathcal{M}(P) \times (AType \times ARate^{\mathcal{M}(P)}) \times \mathcal{M}(P)$ *is a set of transitions.*
- $M_0 \in \mathcal{M}(P)$ *is called the initial marking.*

The set $\mathcal{M}(P)$ *of possible markings will be ranged over by* M*. We denote by* $\mathcal{RG}[\![N]\!]$ *the LTS representing the reachability graph of* N*.* ■

3 Net Semantics

Net Places. The first step in the definition of the GSPN semantics for EMPA consists of introducing a set of places onto which sequential terms will be mapped as well as defining a function which decomposes terms into their sequential components. Since sequential terms are represented by guarded alternative compositions, these are used to formalize places. In order to keep track of the static operators, we suitably decorate action types within net places. Each action type a becomes $a^{\theta,R,\sigma}$ where θ is the functional abstraction related decoration, R is the functional relabeling related decoration, and σ is the parallel composition related decoration. As we shall see, such decorations are introduced by the decomposition function. In order to avoid undesired substitutions, we identify binders arising from the static operators. In the term E/L, all the action types in L are bound in E. In the term $E[\varphi]$, all the action types in $Dom(\varphi) = \{c \in AType \mid \varphi(c) \neq c\}$ are bound in E. For operational convenience, we rewrite the synchronization set S of term $E_1 \|_S E_2$ as the function $s = \{(a,a) \mid a \in S\}$. In the term $E_1 \|_s E_2$, all the action types in $\pi_1(s) = \{a \mid \exists b.\,(a,b) \in s\}$ are bound in E_1 and E_2.

For the functional abstraction operator, decoration $\theta \in \{\tau, \varepsilon\}$ is used to remember to hide certain action types. As an example, $(<a, \tilde{\lambda}>.\underline{0})/\{a\}$ will be mapped onto place $<a^{\tau,-,-}, \tilde{\lambda}>.\underline{0}$. This decoration provides the information that action type a is hidden. This information will be used when generating net transitions.

For the functional relabeling operator, action types are directly relabeled within net places and conflicts are employed to avoid clashes within the scope of the operator. We define *Conf* as an infinite set of conflicts (ranged over by r) on which the operation $\bar{\ } : Conf \longrightarrow Conf, \bar{\bar{r}} = r$ is defined. Given this definition, term $(<a, \tilde{\lambda}>.\underline{0} + <b, \tilde{\mu}>.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, will be mapped onto place $<b^{\varepsilon,\{r\},-}, \tilde{\lambda}>.\underline{0} + <b^{\varepsilon,\{\bar{r}\},-}, \tilde{\mu}>.\underline{0}$. In this way, the two action types are kept distinct within the scope of the functional relabeling operator by their complementary conflicts r and $\bar{r}$. Such conflicts are necessary whenever parallel composition op-

[2] We use "{|" and "|}" as brackets for multisets, "$\_ \oplus \_$" to denote multiset union, and $\mathcal{M}(S)$ ($\mathcal{P}(S)$) to denote the collection of finite multisets (sets) over set S.

erators occur within the scope of the relabeling operator. Again, this information will be used when generating net transitions.

For the parallel composition operator, combinators are used to correctly handle synchronizations. We define *Comb* as an infinite set of combinators (ranged over by k) and we consider the set $Comb^*$ of combinator strings (ranged over by σ), on which the following two operations are defined: $(i)\ ^- : Comb \longrightarrow Comb, \bar{\bar{k}} = k, (ii) \odot : Comb^* \times Comb^* \not\longrightarrow Comb^*, \sigma k \odot \sigma\bar{k} = \sigma$, which is extended to multisets over $Comb^*$ as follows:

$$\odot m = \begin{cases} \odot(\{\!|\,\sigma\,|\!\} \oplus m') & \text{if } \exists \sigma, k, m'.\, m = \{\!|\,\sigma k, \sigma\bar{k}\,|\!\} \oplus m' \\ m & \text{otherwise} \end{cases}$$

Given this definition, term $<a,\tilde{\lambda}>.\underline{0} \|_{\{a\}} <a,*>.\underline{0}$ will be mapped onto the set of places $\{<a^{\varepsilon,\emptyset,k},\tilde{\lambda}>.\underline{0}, <a^{\varepsilon,\emptyset,\bar{k}},*>.\underline{0}\}$ which can synchronize because they have complementary combinators k and $\bar{k}$.

Definition 3 *The set of places is defined by*
$\mathcal{V} = \{\sum_{i\in I} <a_i^{\theta_i,R_i,\sigma_i},\tilde{\lambda}_i>.E_i \mid \theta_i \in \{\tau,\varepsilon\} \wedge R_i \in \mathcal{P}(Conf) \wedge \sigma_i \in Comb^* \wedge E_i \in \mathcal{G}'\}$
where the summation is associative and commutative and $\mathcal{G}'$ is the set of closed and guarded terms defined on the same syntax as $\mathcal{G}$ with Act replaced by $Act' = Act \times \{\tau,\varepsilon\} \times \mathcal{P}(Conf) \times Comb^*$, where $(<a,\tilde{\lambda}>,\theta,R,\sigma)$ is denoted $<a^{\theta,R,\sigma},\tilde{\lambda}>$ and $<a^{\varepsilon,\emptyset,\varepsilon},\tilde{\lambda}>$ is denoted $<a,\tilde{\lambda}>$. $\mathcal{V}$ will be ranged over by V while $\mathcal{M}(\mathcal{V})$ will be ranged over by Q. ■

For operational convenience, we make explicit the actions occurring in the body E of each constant definition $A \stackrel{\Delta}{=} E$ by rewriting the definition itself as $A(a_1,\ldots,a_n) \stackrel{\Delta}{=} E$, where $\{a_1,\ldots,a_n\} = sort(E)$.

Definition 4 *The syntactical substitutions on $\mathcal{G}'$ are defined by structural induction as follows*

$$\begin{aligned}
(\textstyle\sum_{i\in I} <a_i^{\theta_i,R_i,\sigma_i},\tilde{\lambda}_i>.E_i)\{b^{\theta,R,\sigma}/a\} &= \textstyle\sum_{i\in I} <c_i^{\theta'_i,R'_i,\sigma'_i},\tilde{\lambda}_i>.E_i\{b^{\theta,R,\sigma}/a\} \\
E/L\{b^{\theta,R,\sigma}/a\} &= \begin{cases} E\{b^{\theta,R,\sigma}/a\}/L & \text{if } a \notin L \\ E/L & \text{if } a \in L \end{cases} \\
E[\varphi]\{b^{\theta,R,\sigma}/a\} &= \begin{cases} E\{b^{\theta,R,\sigma}/a\}[\varphi] & \text{if } a \notin Dom(\varphi) \\ E[\varphi] & \text{if } a \in Dom(\varphi) \end{cases} \\
(E_1 \|_s E_2)\{b^{\theta,R,\sigma}/a\} &= \begin{cases} E_1\{b^{\theta,R,\sigma}/a\} \|_s E_2\{b^{\theta,R,\sigma}/a\} & \text{if } a \notin \pi_1(s) \\ E_1 \|_{s'} E_2 & \text{if } a \in \pi_1(s) \end{cases} \\
A(a_1^{\theta_1,R_1,\sigma_1},\ldots,a_n^{\theta_n,R_n,\sigma_n})\{b^{\theta,R,\sigma}/a\} &= A(c_1^{\theta'_1,R'_1,\sigma'_1},\ldots,c_n^{\theta'_n,R'_n,\sigma'_n})
\end{aligned}$$

where

$$s' = \{(d_i, c_i^{\theta'_i,R'_i,\sigma'_i}) \mid \exists a_i^{\theta_i,R_i,\sigma_i}.\,(d_i, a_i^{\theta_i,R_i,\sigma_i}) \in s\}$$

and

$$c_i^{\theta'_i,R'_i,\sigma'_i} = \begin{cases} b^{\theta_i\theta,R_i\cup R,\sigma_i\sigma} & \text{if } a_i = a \\ a_i^{\theta_i,R_i,\sigma_i} & \text{if } a_i \neq a \end{cases}$$

■

Definition 5 *The decomposition function* $dec : \mathcal{G}' \longrightarrow \mathcal{M}(\mathcal{V})$ *is defined by structural induction as follows*

$$\begin{aligned}
dec(\textstyle\sum_{i\in I} <a_i^{\theta_i,R_i,\sigma_i}, \tilde{\lambda}_i>.E_i) &= \{\!|\, \textstyle\sum_{i\in I} <a_i^{\theta_i,R_i,\sigma_i}, \tilde{\lambda}_i>.E_i \,|\!\} \\
dec(E/L) &= dec(E\{a^{\tau,\emptyset,\varepsilon}/a \mid a \in L\}) \\
dec(E[\varphi]) &= dec(E(\{b^{\varepsilon,\{r\},\varepsilon}/a, b^{\varepsilon,\{\bar{r}\},\varepsilon}/b \mid \\
&\qquad a \in Dom(\varphi) \wedge \varphi(a) = b \wedge \\
&\qquad b \notin Dom(\varphi) \wedge r \text{ fresh}\} \cup \\
&\qquad \{b/a \mid a,b \in Dom(\varphi) \wedge \\
&\qquad\qquad \varphi(a) = b \wedge r \text{ fresh}\})) \\
dec(E_1 \|_s E_2) &= dec(E_1\{s(a)^{\varepsilon,\emptyset,k}/a \mid a \in \pi_1(s) \wedge k \text{ fresh}\}) \oplus \\
&\qquad dec(E_2\{s(a)^{\varepsilon,\emptyset,\bar{k}}/a \mid a \in \pi_1(s) \wedge k \text{ fresh}\}) \\
dec(A(b_1^{\theta_1,R_1,\sigma_1}, \ldots, b_n^{\theta_n,R_n,\sigma_n})) &= dec(E\{b_i^{\theta_i,R_i,\sigma_i}/a_i \mid 1 \le i \le n\}) \\
&\qquad \text{if } A(a_1,\ldots,a_n) \stackrel{\Delta}{=} E
\end{aligned}$$

■

Net Transitions. The second step in the definition of the GSPN semantics for EMPA consists of suitably connecting net places by means of net transitions. Term $(<a,\tilde{\lambda}>.\underline{0})/\{a\}$, mapped onto place $<a^{\tau,\emptyset,\varepsilon},\tilde{\lambda}>.\underline{0}$, becomes the preset of a transition labeled with τ by exploiting the information stored in the decoration. Term $(<a,\tilde{\lambda}>.\underline{0} + <b,\tilde{\mu}>.\underline{0})[\varphi]$, where $\varphi(a) = \varphi(b) = b$, mapped onto place $<b^{\varepsilon,\{r\},\varepsilon},\tilde{\lambda}>.\underline{0} + <b^{\varepsilon,\{\bar{r}\},\varepsilon},\tilde{\mu}>.\underline{0}$, becomes the preset of two transitions labeled with $b,\tilde{\lambda}$ and $b,\tilde{\mu}$, respectively. Term $<a,\tilde{\lambda}>.\underline{0} \|_{\{a\}} <a,*>.\underline{0}$, mapped onto the set of places $\{<a^{\varepsilon,\emptyset,k},\tilde{\lambda}>.\underline{0}, <a^{\varepsilon,\emptyset,\bar{k}},*>.\underline{0}\}$, becomes the preset of a transition labeled with $<a,\tilde{\lambda}>$ using the combinators k and $\bar{k}$ stored in the decorations.

$$\{\!|\, \sum_{h=1}^{n_i} <a^{\theta,R_{i,h},\sigma_{i,h}}, \tilde{\lambda}_{i,h}>.E_{i,h} + F_i \mid 1 \le i \le n \,|\!\} \xrightarrow{norm(<b,\tilde{\lambda}>,E,f_1,f_2)}_{\mathcal{N}} \bigoplus_{i=1}^{n} dec(E_{i,1})$$

$$norm(<a,\tilde{\lambda}>,E,f_1,f_2) = \begin{cases} <a,*> & \text{if } \tilde{\lambda} = * \\ <a,\tilde{\lambda}\cdot f_1 \cdot f_2/s> & \text{if } \tilde{\lambda} \in \mathbb{R}_+ \\ <a,\infty_{l,w\cdot f_1\cdot f_2/s}> & \text{if } \tilde{\lambda} = \infty_{l,w} \end{cases}$$

$$s = \sum\{\!|\, f_2' \mid \exists Q_1,E',Q_2.\, Q_1 \xrightarrow{norm(<a,\tilde{\lambda}>,E',f_1,f_2')}_{\mathcal{N}} Q_2 \wedge dec(E') = dec(E) \,|\!\}$$

Table 1: Axiom for EMPA integrated net semantics

Formally, the set of transitions $\longrightarrow_{\mathcal{N}}$ is defined as the least subset of $\mathcal{P}(\mathcal{V}) \times (AType \times ARate^{\mathcal{M}(\mathcal{V})}) \times \mathcal{M}(\mathcal{V})$ generated by the axiom reported in the

first part of Table 1, where $n \in \mathbf{N}_+$ and $n_i \in \mathbf{N}_+$ for all $i = 1, \ldots, n$. The preset of the transition is a finite nonempty multiset of places, from each of which a set of summands is factorized. If the preset contains more than one place, then the generated transition results from the synchronization of the actions of the factorized summands. The factorized summands must have the same action type at the same visibility level (a^{θ}). The postset of the transition is the multiset composed of the places representing the decomposition of the derivative terms of the factorized summands in the preset. The axiom is subject to the following conditions:

1. For all $i = 1, \ldots, n$ and $h, h' = 1, \ldots, n_i$ it must hold $PL(<a^{\theta, R_{i,h}, \sigma_{i,h}}, \tilde{\lambda}_{i,h}>) = PL(<a^{\theta, R_{i,h'}, \sigma_{i,h'}}, \tilde{\lambda}_{i,h'}>)$ and $dec(E_{i,h}) = dec(E_{i,h'})$, whereas
$$F_i \equiv \sum_{h=n_i+1}^{m_i} <a_{i,h}^{\theta_{i,h}, R_{i,h}, \sigma_{i,h}}, \tilde{\lambda}_{i,h}>.E_{i,h}$$
is such that for all $h = n_i + 1, \ldots, m_i$ it must hold $a_{i,h} \neq a$, $\theta_{i,h} \neq \theta$, $PL(<a_{i,h}, \tilde{\lambda}_{i,h}>) \neq PL(<a, \tilde{\lambda}_{i,1}>)$, or $dec(E_{i,h}) \neq dec(E_{i,1})$. [3]

2. For all $i, i' = 1, \ldots, n$, there are no $h = 1, \ldots, n_i$ and $h' = 1, \ldots, n_{i'}$ such that $\{r, \bar{r}\} \subseteq R_{i,h} \cap R_{i',h'}$ whenever $i \neq i'$.

3. $\bigodot_{i=1}^{n} \sigma_i = \{\!|\, \varepsilon \,|\!\}$.

4. $b = \begin{cases} a & \text{if } \theta = \varepsilon \\ \tau & \text{if } \theta = \tau \end{cases}$

5. $\tilde{\lambda} = \begin{cases} Aggr\{\!|\, \tilde{\lambda}_{i,h} \mid 1 \leq h \leq n_i \,|\!\} & \text{if } \exists! i = 1, \ldots, n. \tilde{\lambda}_{i,1} \in \mathbb{R}_+ \cup Inf \\ * & \text{if } \forall i = 1, \ldots, n. \tilde{\lambda}_{i,1} = * \end{cases}$

Cond. 1 states that the summands factorized for each place in the preset of the transition are all and only those involving the same action type a, the same visibility level θ, the same priority level, and derivative terms having the same decomposition. We require terms to have the same decomposition instead of being syntactically identical because *dec* is not injective. Such a factorization, together with Cond. 5, prevents identical transitions from collapsing by summing up their rates. Cond. 2 and 3 ensure that the places in the preset correctly synchronize by considering the decorations of the factorized actions. More precisely, Cond. 2 avoids the generation of synchronization transitions which are forbidden because of a functional relabeling operator. Cond. 3, instead, checks for the correct reduction of combinators to the empty string. Cond. 4 determines the action type labeling the transition according to the functional abstraction related decoration θ of the selected summands. Cond. 5 establishes the basic rate which is used to compute (by means of function *norm*) the actual rate of the transition, provided

[3] The priority level is defined as follows: $PL(<a, \tilde{\lambda}>) = 0, PL(<a, \infty_{l,w}>) = l, PL(<a, *>) = -1$.

that, in case of synchronization of several places, at most one of them contributes with a nonpassive action.

$norm : Act \times (\mathcal{G}' \cup \{\bot\}) \times \mathbf{N}^{\mathcal{M}(\mathcal{V})} \times \mathbf{N}^{\mathcal{M}(\mathcal{V})} \longrightarrow AType \times ARate^{\mathcal{M}(\mathcal{V})}$, defined in the second part of Table 1, computes the actual rate of transitions according to their basic rates and three factors – the nonpassive derivative term E, the nonpassive contribution f_1, and the passive contribution f_2 – where:

$$E \equiv \begin{cases} E_{i,1} & \text{if } \exists! i = 1,\ldots,n.\, \tilde{\lambda}_{i,1} \in \mathbb{R}_+ \cup Inf \\ \bot & \text{if } \forall i = 1,\ldots,n.\, \tilde{\lambda}_{i,1} = * \end{cases}$$

$$f_1 = \begin{cases} M_{\text{curr}}(\sum_{h=1}^{n_i} <a^{\theta,R_{i,h},\sigma_{i,h}}, \tilde{\lambda}_{i,h}>.E_{i,h} + F_i) & \text{if } \exists! i = 1,\ldots,n.\, \tilde{\lambda}_{i,1} \in \mathbb{R}_+ \cup Inf \\ 1 & \text{if } \forall i = 1,\ldots,n.\, \tilde{\lambda}_{i,1} = * \end{cases}$$

$$f_2 = \begin{cases} \prod_{i=1}^{n} \{|\, n_i \cdot M_{\text{curr}}(\sum_{h=1}^{n_i} <a^{\theta,R_{i,h},\sigma_{i,h}}, \tilde{\lambda}_{i,h}>.E_{i,h} + F_i) \mid \tilde{\lambda}_{i,1} = * \,|\} & \text{if } \exists i = 1,\ldots,n.\, \tilde{\lambda}_{i,1} = * \\ 1 & \text{if } n = 1 \wedge \tilde{\lambda}_{1,1} \in \mathbb{R}_+ \cup Inf \end{cases}$$

In case of several transitions coming from the synchronization of a place contributing with a nonpassive action with several places contributing with alternative passive actions of the same type, factor f_2 is suitably normalized (see the second part of Table 1). Such transitions have the same type, the same nonpassive basic rate, and nonpassive derivative terms with the same decomposition. The normalizing factor s for them is computed by summing up their passive contributions.

The factor E is the derivative term appearing in the place (if any) of the preset which contributes to the transition with a nonpassive action. Since transitions coming from the synchronization of a place contributing with a nonpassive action with several places contributing with alternative passive actions of the same type have in general different presets, we need this piece of information beside the basic rate in order to correctly carry out the normalization. The marking dependent factor f_1 is the number of tokens in the place (if any) of the preset which contributes to the transition with a nonpassive action. The marking dependent factor f_2 is the product, over the set of places (if any) of the preset which contribute to the transition with a passive action, of the number of factorized summands multiplied by the number of tokens contained in the related places.

Nets Associated with Terms. The third step consists of associating with each term an appropriate GSPN by exploiting the previous two steps.

Definition 6 *The integrated net semantics of $E \in \mathcal{G}$ is the GSPN*

$$\mathcal{N}[\![E]\!] = (P_{E,\mathcal{N}}, AType \times ARate^{\mathcal{M}(P_{E,\mathcal{N}})}, \longrightarrow_{E,\mathcal{N}}, dec(E))$$

where $P_{E,\mathcal{N}}$ is the least subset of $\mathcal{V}$ such that:

- *$dec(E) \subseteq P_{E,\mathcal{N}}$;*
- *if $Q_1 \subseteq P_{E,\mathcal{N}}$ and $Q_1 \xrightarrow{norm(<a,\tilde{\lambda}>,E,f_1,f_2)}_{\mathcal{N}} Q_2$, then $Q_2 \subseteq P_{E,\mathcal{N}}$;*

and $\longrightarrow_{E,\mathcal{N}}$ *is* $\longrightarrow_{\mathcal{N}}$ *restricted to* $\mathcal{M}(P_{E,\mathcal{N}}) \times (AType \times ARate^{\mathcal{M}(P_{E,\mathcal{N}})}) \times \mathcal{M}(P_{E,\mathcal{N}})$. ■

Retrievability Result. The net semantics for EMPA is correct w.r.t. the integrated interleaving semantics for EMPA, i.e. the integrated net semantics and the integrated interleaving semantics of a given term represent the same system both from the functional and the performance point of view, because for all $E \in \mathcal{G}$ $\mathcal{RG}[\![\mathcal{N}[\![E]\!]]\!]$ is strongly extended Markovian bisimilar [2] to $I[\![E]\!]$.

4 Integrating TwoTowers and GreatSPN

TwoTowers. TwoTowers [2] is a software tool for the analysis of functional and performance properties of distributed systems described in EMPA. TwoTowers is composed of a graphical user interface, a parser, an integrated kernel, a functional kernel, and a performance kernel. The integrated kernel contains a routine to check two EMPA specifications for strong extended Markovian bisimulation equivalence [2] as well as a routine for the simulative analysis. The functional kernel investigates functional properties of EMPA specifications via model checking, equivalence checking, and preorder checking. Finally, the performance kernel computes transient and steady state probability distributions and user defined, reward based performance measures of EMPA specifications.

GreatSPN. GreatSPN [5] is a software tool for the modeling, verification, and performance evaluation of distributed systems using GSPNs. It provides a graphical user interface for model editing and for structural properties and performance results visualization. Once a GSPN model has been specified, its structural analysis can start allowing the investigation of interesting model properties without building its state space. If the GSPN has a finite state space, the modules responsible for its generation and analysis can be invoked. The underlying Markov chain can be derived and transient or stationary marking probability distributions can be computed as well as a number of default and user defined performance measures.

Implementation of the Net Semantics. Given an EMPA specification, the net kernel of TwoTowers generates the corresponding GSPN by initially producing all the places. This is accomplished by computing the decomposition of the initial constant definition of the specification, and by repeating this procedure for each sequential term associated with a place after removing its outermost action prefixes. The places, as well as the sequential terms, are stored in a hash table for efficient retrieval. The bucket of a term points to the buckets of the related places resulting from its decomposition, so it is straightforward to see whether the decomposition of a given term has already been computed or not. Similarly, the bucket of a place points to the bucket of the related sequential term; this allows an efficient reinterpretation at the algebraic level of analysis results computed at the net level.

Once all the places have been generated, the transitions are produced by examining the parallel composition related decorations of the actions within places in order to build transition presets according to Cond. 3. More precisely, in the first round we consider those places having some actions with ε as combinator string. Such places cannot synchronize with others and hence give rise to transitions with a singleton preset. In the next rounds, we consider the remaining places two by two and we try to reduce the combinator strings of their actions. Three cases arise:

- If two places combine and the resulting combinator string is not empty, a new dummy place is generated which is considered in the subsequent rounds instead of the two originating places. The dummy place contains the information about the two places that have originated it.
- If two places combine and the resulting combinator string is empty, a new synchronization transition is generated and the two places will no longer be considered in the subsequent rounds. The preset of the newly generated transition is composed of these places. When one or both of them are dummy places, their expansion is required.
- If two places do not combine, nothing happens.

The number of rounds is bounded by the length of the longest combinator string.

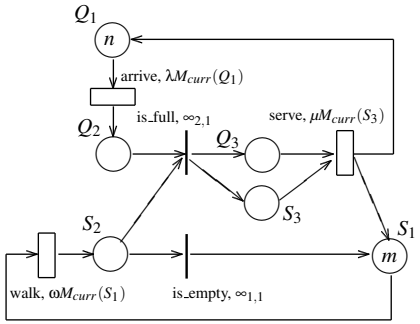
Given an EMPA specification, the corresponding GSPN produced by TwoTowers in the format expected by GreatSPN can be analyzed by invoking via TwoTowers some routines implemented in GreatSPN. First of all, P-invariants and T-invariants can be computed so we can derive – without generating the underlying state space – information about properties of the modeled system such as boundedness and liveness, respectively. Moreover, since places are in correspondence with sequential subterms occurring in the EMPA specification, invariants can be reinterpreted at the algebraic level, thereby showing properties not directly visible on the algebraic specification. On the performance side, it is possible to compute the maximum and the average numbers of tokens in the places as well as upper bounds on the throughput of transitions by solving suitable linear programming problems [4] without generating the underlying state space.

5 Random Polling Systems

In this section we apply the GSPN semantics above to the EMPA specification of a simple multi-server multi-queue (MSMQ) system adopting a random polling service policy. A MSMQ system is composed of a set of waiting lines that receive arrivals from the external world, and a set of servers that attend the queues and provide service when required. Our MSMQ system is composed of $n \in \mathbf{N}_+$ queues, each of capacity one (i.e., only one customer can wait at each station), and $m \in \mathbf{N}_+$ servers (with $m \leq n$). Each server randomly polls a queue and checks if

there is a customer to be served. If so, it removes the customer from the buffer, serves it, and then walks to the next randomly chosen queue, even the one it has just visited. We assume that the servers and the queues have the same timing characteristics: arrival times, service times, and walk times have rates λ, μ, and ω, respectively. Note that the system is symmetric w.r.t. both servers and queues. The EMPA specification of the MSMQ system is:

$$
\begin{aligned}
RPS_{m,n} &\triangleq (S \|_\emptyset S \|_\emptyset \ldots \|_\emptyset S) \|_T (Q \|_\emptyset Q \|_\emptyset \ldots \|_\emptyset Q) \\
T &= \{is\_full, serve\} \\
S &\triangleq <walk, \omega>.(<is\_full, \infty_{2,1}>.<serve, \mu>.S + <is\_empty, \infty_{1,1}>.S) \\
Q &\triangleq <arrive, \lambda>.<is\_full, *>.<serve, *>.Q
\end{aligned}
$$



$$
\begin{aligned}
S_1 &\equiv <walk^{\varepsilon,\emptyset,\varepsilon}, \omega>.S_2 \\
S_2 &\equiv <is\_full^{\varepsilon,\emptyset,k_1}, \infty_{2,1}>.S_3 + \\
&\quad\ <is\_empty^{\varepsilon,\emptyset,\varepsilon}, \infty_{1,1}>.S_1 \\
S_3 &\equiv <serve^{\varepsilon,\emptyset,k_2}, \mu>.S_1 \\
Q_1 &\equiv <arrive^{\varepsilon,\emptyset,\varepsilon}, \lambda>.Q_2 \\
Q_2 &\equiv <is\_full^{\varepsilon,\emptyset,\bar{k}_1}, *>.Q_3 \\
Q_3 &\equiv <serve^{\varepsilon,\emptyset,\bar{k}_2}, *>.Q_1
\end{aligned}
$$

Figure 1: $\mathcal{N}[\![RPS_{m,n}]\!]$

The net corresponding to $RPS_{m,n}$, obtained by applying the three steps described in Sect. 3, is shown in Fig. 1. Place S_1 models the servers before polling a queue; when it is marked, exponential transition *walk* is enabled and its firing represents one server reaching a queue. The rate of transition *walk* varies according to the marking of S_1. Place S_2 represents the servers in front of the queues. The arrival of a new customer is modeled by exponential transition *arrive* whose rate varies according to the number of tokens in Q_1. If there are waiting customers, i.e. place Q_2 is marked, immediate transition *is_full* can fire modeling a synchronization between one server and one queue. Now exponential transition *serve* can fire modeling the provision of service. If there are no waiting customers, immediate transition *is_empty* can instead fire. Note that the symmetry in the roles of the servers and the queues is reflected at the net level by the presence of two subnets, one composed of places S_1, S_2, and S_3, the other composed of places Q_1, Q_2, and Q_3. Changes in the number of servers and/or queues do not alter the structure of the net but only its initial marking.

Given the net description file of the GSPN model of Fig. 1 automatically produced by TwoTowers, we ran both tools to compute the system throughput for $\lambda = 1$, $\mu = 2$, and $\omega = 3$, when varying the numbers m of servers and n of queues

m	n	$\#I$	throughput	$\#\mathcal{RG}$	throughput	upper bound
2	2	29	1.159270	11	1.159270	1.333333
2	3	78	1.636170	16	1.636170	2.000000
3	3	177	1.796080	19	1.796080	2.000000
2	4	200	1.997520	21	1.997530	2.400000
3	4	504	2.318680	26	2.318680	2.666700
4	4	1089	2.443010	29	2.443010	2.666700
2	5	496	2.224840	26	2.224850	2.400000
3	5	1368	2.766000	33	2.766010	3.333333
4	5	3210	2.991610	38	2.991610	3.333333
5	5	6693	3.095440	41	3.095430	3.333333

Table 2: Results of the analysis of $RPS_{m,n}$

from 2 to 5. The results are shown in Table 2. The third and fourth columns describe the number of states of the integrated interleaving semantics and the value of the service throughput computed with TwoTowers. The last three columns describe the number of states of the reachability graph of the integrated net semantics, the system throughput, and an upper bound on the system throughput computed with GreatSPN. For systems exhibiting a high degree of symmetry, working with TwoTowers at the integrated interleaving semantic model level requires an amount of time which grows exponentially with the number of servers and queues. Instead, working with GreatSPN on the GSPN model derived by applying our net semantics requires an amount of time which grows linearly and it is possible to avoid the state space contruction if we restrict ourselves to bounds computation.

6 Conclusion

We have described an extension of the compact net semantics presented in [3], whose contribution lies in the treatment of action rates. We have then presented how this mapping has been implemented in order to integrate TwoTowers and GreatSPN. The approach we have followed to define the GSPN semantics for EMPA has several advantages [3]. For instance, the reachability graph of the resulting nets can be smaller than the state spaces of the corresponding EMPA terms, and structural techniques can be applied to the net representation of algebraic terms thereby avoiding the construction of the underlying state space.

Related Work. The detection of symmetries is not new in the field of SPAs. In [6] a reduced state space underlying an SPA model is obtained by building equivalence classes of states and by considering one element for each class only. The reduction in the size of the state space is not optimal – it is still possible to find states which are Markovian bisimulation equivalent but do not belong to

the same equivalence class – but it is stronger than ours since it works for synchronized replica whose corresponding nets are kept distinct in our case. Another approach to state space reduction is discussed in [7]. Here a new operator is introduced for representing sets of identical processes, all synchronizing on the same sets of actions. New semantic rules, which are consistent with those for the traditional parallel composition operator but avoid the representation of all possible interleavings, have been defined. An informal net semantics is also discussed.

Future Work. The GSPN model of our random polling system is very compact since the net semantics captures its instrinsic symmetries. There are other situations in which we would like to recognize a symmetric behavior, e.g. in the case of a customer faster than the others. Here the customers are still equivalent from a purely functional point of view, but the net semantics would produce a distinct subnet for the fast customer because of the different rate values. The same would happen in a GSPN model but would not happen in a colored GSPN model where it is possible to define transition rates depending on the identity of the tokens, i.e. on the identity of the customers. This suggests the extension of the compact net semantics towards a colored net semantics.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *"Modelling with Generalized Stochastic Petri Nets"*, Wiley & Sons, 1995

[2] M. Bernardo, *"Theory and Application of Extended Markovian Process Algebra"*, Ph.D. Thesis, University of Bologna (Italy), 1999 (`http://www.di.unito.it/~bernardo/`)

[3] M. Bernardo, N. Busi, M. Ribaudo, *"Compact Net Semantics for Process Algebras"*, Tech. Rep. UBLCS-2000-02, University of Bologna (Italy), 2000

[4] G. Chiola, C. Anglano, J. Campos, J.M. Colom, M. Silva, *"Operational Analysis of Timed Petri Nets and Applications to the Computation of Performance Bounds"*, in Proc. of PNPM '93, IEEE-CS Press, pp. 128-137

[5] G. Chiola, G. Franceschinis, R. Gaeta, M. Ribaudo, *"GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets"*, in Performance Evaluation 24:47-68, 1995

[6] S. Gilmore, J. Hillston, M. Ribaudo, *"An Efficient Algorithm for Aggregating PEPA Models"*, to appear in IEEE Trans. on Software Engineering, 2000

[7] H. Hermanns, M. Ribaudo, *"Exploiting Symmetries in Stochastic Process Algebras"*, in Proc. of ESM '98, SCS Europe, pp. 763-770