Integrating Performance and Functional Analysis of Concurrent Systems with EMPA

Marco Bernardo  Lorenzo Donatiello  Roberto Gorrieri

Technical Report UBLCS-95-14
September 1995
Revised March 1996

Department of Computer Science
University of Bologna
Piazza di Porta S. Donato, 5
40127 Bologna (Italy)
Recent Titles from the UBLCS Technical Report Series

94-20 Dynamic Allocation of Signature Files in Multiple-Disk Systems, P. Ciaccia, August 1994.
94-21 Parallel Independent Grid Files Based on a Dynamic Declustering Method Using Multiple Error Correcting Codes, P. Ciaccia, November 1994.
95-13 Optimal Multi-Block Read Schedules for Partitioned Signature Files, P. Ciaccia, August 1995.
Integrating Performance and Functional Analysis of Concurrent Systems with EMPA

Marco Bernardo 1  Lorenzo Donatiello 1  Roberto Gorrieri 1

Technical Report UBLCS-95-14

September 1995
Revised March 1996

Abstract

An approach is proposed for modeling and analyzing concurrent systems that integrates different views (abstract vs. concrete) as well as different aspects (functional vs. performance) of their behavior. The approach is based on a stochastic process algebra, a formalism for specifying and studying concurrent systems that is characterized by the property of compositionality. The approach is instantiated to the case of Extended Markovian Process Algebra (EMPA), introduced together with the collection of its four semantics and the notion of equivalence that are required in order to implement the approach. Finally, the case study of the alternating bit protocol is presented to illustrate the approach.

1. Università di Bologna, Dipartimento di Scienze dell’Informazione, Via di Mura Anteo Zamboni 7, 40127 Bologna, Italy. E-mail: {bernardo, donat, gorrieri}@cs.unibo.it.
1 Introduction

The need of integrating the performance modeling and analysis of a concurrent system into the design process of the system itself has been widely recognized (see, e.g., [68, 31, 35, 22]) and stimulated many researchers. The problem is that, in the case when quantitative aspects are neglected, time-critical concurrent systems (such as communication protocols) cannot be given completely satisfactory models and, moreover, these models cannot be used to estimate the system performance.

Unfortunately, it often happens that a concurrent system is first fully designed and tested for functionality, and afterwards tested for efficiency. The major drawback is that, whenever the performance is detected to be poor, the concurrent system has to be designed again, thereby negatively affecting both the design costs and the delivery at a fixed deadline. Another relevant drawback is that tests for functionality and performance are carried out on two different models of the system, so one has to make sure that these two models are consistent, i.e. they really describe (different aspects of) the same system.

In the last two decades a remarkable effort has been made in order to make existing formal description techniques suitable to support performance modeling and analysis. The key feature common to all of the proposals is to enhance the expressiveness of the formal description techniques by introducing the concept of time, represented either in a deterministic way (see, e.g., [46, 57, 58, 49, 32]) or in a stochastic way (see, e.g., [52, 50, 5, 30, 54, 47, 9, 33, 38]).

One of the most mature fields where functional and performance aspects of concurrent systems are both considered is that of stochastic Petri nets, appeared in the 80’s (see [4] and the references therein). Like classical Petri nets (see [59] for an introductory textbook), they are mathematical models for graphically representing concurrent systems: their main feature is that the notion of state is distributed among net places, hence allowing dependencies, conflicts and synchronizations among system activities to be explicitly described. Stochastic Petri nets extend the expressiveness of classical Petri nets by associating with each net transition a random variable determining its duration.

The advantage of using a stochastic Petri net as a model for a given concurrent system is that functional and performance aspects are both taken into account since the beginning of the design process. These two aspects can then be separately analyzed on two different “projected models” (a classical Petri net and, typically, a Markov chain) obtained from the same “integrated model” (the stochastic Petri net), so we are guaranteed that the projected models are consistent. However two problems have to be addressed: (i) lack of compositionality, i.e. the capability of constructing nets by composing smaller ones, and (ii) inability to perform an integrated analysis, i.e. an analysis carried out directly on the integrated model, which can be much more efficient as there is no need of building projected models. Both problems can be overcome by resorting to stochastic process algebras.

Stochastic process algebras appeared in the 90’s (see [1, 2, 3] and the references therein). Like classical process algebras [48, 41, 13, 23], they are algebraic languages whose key feature is compositionality. Compositionality concerns both the syntactical and the semantic level of the language. Syntactical compositionality is related to system modeling: process algebras provide the system designer with a small set of powerful operators whereby it is possible to construct process terms from simpler ones, without incurring in the graphical complexity of nets and making the task of recognizing or modifying system components quite easier. Semantic compositionality is related to system analysis: process algebras enable the system designer to study separately the various system components, provided that an appropriate notion of equivalence over process terms is developed.

Stochastic process algebras extend the expressiveness of classical process algebras by representing each action as a pair composed of a type and a duration. Functional and performance analyses of a process term can be carried out on two consistent projected semantic models (a transition system labeled only on the type of the actions, and, typically, a Markov chain), as well as directly on the integrated semantic model (a transition system labeled on both the type and the duration of the actions) provided that a suitable notion of “integrated equivalence” is
The aim of this paper is to tackle the problem cited at the beginning by means of the proposal of an integrated approach for modeling and analyzing concurrent systems, based on stochastic process algebras and stochastic Petri nets. The approach implements three orthogonal integrations:

(i) The first integration relates the abstract and the concrete views of concurrent systems. The abstract view is provided by process terms: they give an algebraic representation of system components and their interactions, whose semantic model is obtained by interleaving actions of concurrent components. The concrete view is provided by Petri nets: they give a machine-like representation of systems with the explicit description of concurrency. This integration results in the two phases depicted in Figure 1.

(ii) The second integration relates functional and performance aspects of the behavior of concurrent systems. This integration is depicted in Figure 1 by means of the contrast between the nonshaded part and the shaded part.

(iii) The third integration consists of exploiting several tools tailored for specific purposes.

Let us explain in more details the two phases in the light of the three orthogonal integrations mentioned above.

1. The first phase requires the system designer to specify the concurrent system as a term of the stochastic process algebra. Because of syntactical compositionality, the system designer is allowed to develop the algebraic representation of the system in a modular way. From the algebraic representation, an integrated interleaving semantic model is automatically derived in the form of a transition system labeled on both the type and the duration of the actions. The integrated interleaving semantic model can be analyzed as a whole by a notion of integrated equivalence, or is projected on a functional semantic model and a performance semantic model that can be analyzed by means of tools like CW [27] and SHARPE [63], respectively.

The functional analysis can be carried out by resorting to methods such as equivalence checking, preorder checking and model checking [27]. Equivalence checking verifies whether a process term meets the specification of a given system in the case when the specification is a process term as well. Preorder checking requires that the specification is still a process term treated as the minimal requirement to be met, owing to the specification can contain don’t care points. Model checking requires specifications to be formalized as modal logic formulas to be satisfied, expressing assertions about safety, liveness or fairness...
The performance analysis permits to obtain quantitative measures by typically resorting to the study of a Markov chain, since almost all the existing process algebras rely on exponential timing.

2. The second phase consists of automatically obtaining from the algebraic representation of the system an equivalent net representation in the form of a stochastic Petri net. The net representation of the concurrent system is derived from the algebraic one without intervention of the system designer, in order to avoid overhead concerning graphical complexity and absence of compositionality. The net representation turns out to be useful whenever a less abstract representation is required highlighting dependencies, conflicts and synchronizations among system activities, and helpful to detect some properties (e.g., partial deadlock) that can be easily checked only in a distributed setting. Additionally, the net representation is usually more compact than the integrated interleaving semantic model, since concurrency is kept explicit instead of being simulated by alternative computations obtained by interleaving actions of concurrent components. The functional and performance analyses of the net representation can be assisted by tools like GreatSPN [26].

The functional analysis aims at detecting behavioral and structural properties of nets (see, e.g., [51]), i.e. both properties depending on the initial marking of the net and properties depending only upon the structure of the net. Concerning structural analysis, the algebraic technique of net invariants is frequently used. Such a technique (see, e.g., [59]) consists of computing the solutions of linear equation systems based on the incidence matrix of the net under consideration. These solutions single out places that do not change their token count during transition firings, or indicate how often each transition has to fire in order to reproduce a given marking. By means of these solutions, properties such as boundedness, liveness and deadlock can be studied.

The performance analysis aims at determining efficiency measures. Since in most of the cases the random variables used for expressing the transition durations are exponentially distributed, performance indices are determined by solving Markov chains. Tools like GreatSPN [26] provides the designer with facilities for deriving performance results based on either the numerical solution of Markov chains or the event-driven simulation of the net.

Since the two phases above are complementary, the choice between them is made depending on the adequacy of the related representation with respect to the analysis of the concurrent system under consideration, and the availability of the corresponding tools. In any case, the system designer is forced to start with an algebraic representation of the system in order to exploit syntactical compositionality.

In order to implement the integrated approach, we have to choose a class of stochastic Petri nets and then a stochastic process algebra having possibly the same expressive power. The class of stochastic Petri nets we have chosen is that of generalized stochastic Petri nets (GSPNs) [5, 6], because they have been extensively studied and successfully applied. Since in the literature there does not exist a stochastic process algebra having the same expressive power as GSPNs, we have developed a new stochastic process algebra, called Extended Markovian Process Algebra (EMPA), which is endowed with expressive features typical of GSPNs. The name of the algebra stems from the fact that action durations are mainly expressed by means of exponentially distributed random variables (hence Markovian), but it is also possible to express actions having duration zero as well as actions whose duration is unspecified (hence Extended). In order to support the various phases and analyses of the integrated approach, EMPA has been equipped with a collection of semantics as depicted in Figure 2, as well as a notion of integrated equivalence. Each term has an integrated interleaving semantics represented by a labeled transition system (LTS) whose labels consist of both the type and the duration of the actions, and an integrated net semantics represented by a GSPN. From the integrated interleaving semantic model, two

---

2. The original name was MPA [16, 17, 18, 19, 20, 21]. It has been subsequently changed in order to avoid confusion with another MPA [25].
projected semantic models can be obtained: a functional model given by a LTS labeled only on the type of the actions, and a performance model given by a homogeneous continuous-time Markov chain (HCTMC).

Although the integrated approach has in principle a general validity, in this paper we consider only exponential distributions. The restriction to exponentially distributed durations simplifies the performance evaluation, as the performance model turns out to be a HCTMC. Also, such a restriction affects the semantic treatment, because the memoryless property of the exponential distribution allows us to define an integrated semantics for EMPA through the interleaving approach, in the same style as classical process algebras: each parallel execution is simulated by means of the set of the sequential executions obtained by interleaving the actions involved in the parallel execution itself. For instance, suppose we are given an action $a$ whose duration is exponentially distributed with rate $\lambda$, and an action $b$ whose duration is exponentially distributed with rate $\mu$. Then let us consider a term $E_1$ that executes either $a$ followed by $b$ or (operator “$+$”) $b$ followed by $a$, and a term $E_2$ that executes $a$ in parallel with (operator “$|$”) $b$:

$$E_1 \equiv <a, \lambda>, <b, \mu> \cdot 0 + <b, \mu>, <a, \lambda> \cdot 0$$

$$E_2 \equiv <a, \lambda>, 0 || b, \mu> \cdot 0$$

The LTSs representing their integrated interleaving semantics are isomorphic (the unlabeled arrow points to the initial state of the execution):

This is correct from the functional point of view by definition of interleaving, and also from the performance point of view: due to the memoryless property of the exponential distribution, if we assume that $E_2$ completes $a$ before $b$, then the residual time to the completion of $b$ is still exponentially distributed with rate $\mu$, so the rate labeling the transition from state $0 || b, \mu> \cdot 0$ to state $\emptyset || 0$ is $\mu$ itself instead of $\mu$ conditional on $\lambda$.

The paper is organized as follows. In Section 2 we introduce EMPA by giving the syntax of terms and the meaning of operators. In Section 3 the integrated interleaving semantics is defined together with the related functional semantics, while the performance semantics is presented in Section 4. In Section 5 we stress the expressiveness of EMPA, as well as the advantages of

---

3. Actually, in the following we shall see that also phase-type distributions are somehow expressible.
syntactical compositionality, by showing some examples about queueing systems. In Section 6 we
introduce a notion of integrated equivalence. In Section 7 we define the integrated net semantics
and we investigate the relationship with the integrated interleaving semantics. In Section 8 we
apply the integrated approach to a case study: the alternating bit protocol. Finally, in Section 9
we report some concluding remarks on related work, perspectives and open problems.

2 Syntax and informal semantics for EMPA

In this section we introduce EMPA by showing the syntax of its terms and explaining the meaning
of its operators.

This section is organized as follows. In Section 2.1 we introduce the concept of action
together with a classification of actions based on their types and rates. In Section 2.2 we define
the syntax of terms and we informally explain the meaning of each operator. In Section 2.3
we illustrate the race policy, adopted as a mechanism for solving the contention among several
simultaneously executable actions.

2.1 Actions: types and rates

The building blocks of terms are actions that model the activities performed by the systems under
consideration. Each action is a pair “<a, λ>” consisting of the type of the action and the rate of the
action. The type expresses the functional part of the action, i.e. it specifies which kind of action
can be seen by an external observer (e.g. customer arrival, train departure). The rate indicates the
performance part of the action, i.e. the speed at which the action occurs from the point of view of
an external observer: the rate is used as a concise way to denote the random variable specifying
the duration of the action.

Depending on the type, like in classical process algebras, actions are divided into:

- **External or observable actions**, i.e. actions whose type can be seen by an external observer.
- **Internal or invisible actions**, i.e. actions whose type cannot be seen by an external observer.
  
  We denote by τ the only internal action type we use.

Depending on the rate, actions are divided into:

- **Active actions**, i.e. actions whose rate is fixed, in turn divided into:
  
  - **Exponentially timed actions**, i.e. actions whose rate (denoted by λ ∈ R+) is a positive real
    number. Such a number is interpreted as the parameter of the exponential distribution
    specifying the duration of the action.
  
  - **Immediate actions**, i.e. actions whose rate (denoted by ∞) is infinite. Such actions
    have duration zero, and each of them is given a priority level l ∈ N and a weight
    w ∈ R.

- **Passive actions**, i.e. actions whose rate (denoted by *) is undefined. The duration of a
  passive action is fixed only upon a synchronization with an active action of the same type.

The classification of actions based on their rates implies that: (i) exponentially timed actions
model activities that are relevant from the performance point of view, (ii) immediate actions
model logical events as well as activities that are either irrelevant from the performance point of
view or unboundedly faster than the others, (iii) passive actions model activities waiting for the
synchronization with timed activities and are useful to express pure nondeterminism.

We denote the set of actions by

\[ \text{Act} = \text{AType} \times \text{ARate} \]

where \text{AType} is the set of action types and

\[ \text{ARate} = \mathbb{R}^+ \cup \text{Inf} \cup \{\ast\} \]

with \text{Inf} = \{∞l,w | l ∈ N ∧ w ∈ R\},

is the set of action rates. We use a, b, c, ... as metavariables for \text{AType}, λ, μ, γ, ... for \text{ARate}, and
λ, μ, γ, ... for R+.
2.2 Syntax of terms and informal semantics of operators

Before defining the syntax of terms, let us introduce the following. Let \( \text{Const} \) be a set of constants, ranged over by \( A, B, \ldots \). Furthermore, let

\[
Relab = \{ \varphi : \text{AType} \to \text{AType} \mid \varphi(\tau) = \tau \land \varphi(\text{AType} - \{\tau\}) \subseteq \text{AType} - \{\tau\} \}
\]

be a set of relabeling functions that transform action types into other action types preserving their observability.

**Definition 2.1** The set \( \mathcal{L} \) of EMPA process terms is generated by the following syntax

\[
E ::= 0 | <a, \lambda> \cdot E | E/L | E \setminus H | E[\varphi] | E + E | E || S | E | A
\]

where \( L, S \subseteq \text{AType} - \{\tau\} \) and \( H \subseteq \text{AType} \). The set \( \mathcal{L} \) will be ranged over by \( E, F, G, \ldots \). □

In the rest of the section we informally explain the semantics of the operators: the formal semantics will be presented in Section 3.

The **null term** “0” is a zero-ary operator representing a term that cannot execute actions. It can be thought of as either a termination state or a deadlock state.

The **prefix operator** “\(<a, \lambda> \cdot .\)” represents the sequential composition of an action and a term; so, term “\(<a, \lambda> \cdot E\)” can execute action “\(<a, \lambda>\)” and then behaves as term “E”.

The **functional abstraction operator** “\("L\)” expresses the abstraction from the type of actions whenever it is in \( L \), i.e. the action type is turned into \( \tau \). Its meaning is the same as that of the hiding operator of CSP [41], but we prefer to call it functional abstraction operator because also invisible actions consume time. From the point of view of an external observer, the kind of action cannot be seen but the passage of time can.

The **temporal restriction operator** “\("H\)” prevents the execution of passive actions whose type is in \( H \). Based on this operator, we define the notion of **temporal closure**: a term is temporally closed if it cannot execute passive actions. Thus, the temporal closure property singles out terms that are completely specified from the performance viewpoint.

The **functional relabeling operator** “\("\varphi\)” changes the type of the actions (executed by the term to which it is applied) according to \( \varphi \). Its meaning is the same as that of the relabeling operator of CCS [48], but we prefer to call it functional relabeling operator in order to stress that only the action types may be changed, not the rates. Observe that the functional relabeling operator and the functional abstraction operator have clearly distinct roles: the former preserves the observability of the action types and permits to obtain more compact algebraic representations, whereas the latter can only transform observable actions into invisible ones thereby providing an information hiding mechanism.

The **alternative composition operator** “\("+\)” expresses a choice between two terms. Unlike classical process algebras, such a choice is not necessarily nondeterministic since its nature depends upon the rates of the actions executable by the various alternatives (as we shall see in Section 2.3).

The **parallel composition operator** “\("||\)” expresses the parallel execution of two terms, based on two synchronization disciplines:

- The synchronization discipline on action types is the same as that of CSP [41], i.e. term \( E_1 || S \cdot E_2 \), where \( S \) is called the **synchronization set**, can execute asynchronously only actions from \( E_1 \) or \( E_2 \) whose type does not belong to \( S \), and synchronously only actions from \( E_1 \) and \( E_2 \) whose type belongs to \( S \). In case of synchronization, the resulting action has the same type as the involved actions, so multiway synchronizations are allowed.

- The synchronization discipline on action rates states that action “\(<a, \mu>\)” can be synchronized with action “\(<a, \lambda>\)” if and only if \( \min(\lambda, \mu) = * \), and the rate of the resulting action is given by \( \max(\lambda, \mu) \) up to normalization (see Section 3). In other words, in a synchronization at most one active action can be involved and its rate determines the rate of the resulting action, up to normalization. We think that this choice leads to both the adoption of a clearer modular design style and a more intuitive treatment of the parallel composition operator.

---

4. We assume that **\( * < \lambda < \infty_{1, \omega} \) for all \( \lambda \in \mathbb{R}_+ \) and \( \infty_{1, \omega} \in Inf \)**.
as this synchronization discipline on action rates turns out to be based on the client-server paradigm.\(^5\)

Finally, EMPA is equipped with constants. Let \(\text{Def} \subseteq \text{Const} \times \mathcal{L}\) be a set of defining equations. Constant \(A\) can be used as a shorthand for term \(E\) if and only if \(A \triangleq E \in \text{Def}\), meaning that constant \(A\) behaves as term \(E\). The main feature of constants is that they allow for recursion: e.g., in \(E\) above constant \(A\) may appear as a subterm. Whenever a system is modeled by means of a set of recursive definitions, we have to make sure that (i) each constant appearing in these definitions has exactly one defining equation, i.e., occurs exactly once in \(\text{Def}\), and (ii) from each right-hand side term of these definitions having constants as subterms, an equivalent term can be obtained by replacing constants themselves by the right-hand side terms of their defining equations, such that each constant in this term appears in a prefix context (e.g., \(A \triangleq A\) makes no sense, unlike \(A \triangleq \langle a, \lambda \rangle . A\) or \(A \triangleq B, B \triangleq \langle a, \lambda \rangle . B\)). The well-foundedness of such recursive definitions can be then assessed by the notion of guarded closure introduced below.

**Definition 2.2** Let \(E \in \mathcal{L}\) and \(A \triangleq E' \in \text{Def}\), and let us denote by \(\equiv\) the syntactical equivalence between terms. The term \(E(A := E')\) obtained from \(E\) by replacing each occurrence of \(A\) with \(E'\) is defined by induction on the syntactical structure of \(E\) as follows:

- \(0(A := E') \equiv 0\)
- \(\langle a, \lambda \rangle . E(A := E') \equiv \langle a, \lambda \rangle . (E(A := E'))\)
- \(E/L(A := E') \equiv (E(A := E'))/L\)
- \((E \setminus H)(A := E') \equiv (E(A := E')) \setminus H\)
- \((E[\varphi])(A := E') \equiv (E(A := E'))[\varphi]\)
- \((E_1 + E_2)(A := E') \equiv (E_1(A := E')) + (E_2(A := E'))\)
- \((E_1 \parallel_E E_2)(A := E') \equiv (E_1(A := E')) \parallel_E (E_2(A := E'))\)
- \(B(A := E') \equiv \begin{cases} E' & \text{if } B \equiv A \\ B & \text{if } B \not\equiv A \end{cases}\)

**Definition 2.3** Let \(E \in \mathcal{L}\), and let us denote by \(\text{st}\) the relation subterm-of. The set of terms obtained from \(E\) by repeatedly replacing constants by the right-hand side terms of their defining equations is defined by

\[
\text{Subst}(E) = \bigcup_{n \in \mathbb{N}} \text{Subst}^n(E)
\]

where

\[
\text{Subst}^n(E) = \begin{cases} \{E\} & \text{if } n = 0 \\ \{F \in \mathcal{L} \mid F \equiv G(A := E') \land G \in \text{Subst}^{n-1}(E) \land A \text{ st } G \land A \triangleq E' \in \text{Def}\} & \text{if } n > 0 \end{cases}
\]

**Definition 2.4** Let \(E \in \mathcal{L}\). The set of constants occurring in \(E\) is defined by

\[
\text{Const}(E) = \{A \in \text{Const} \mid \exists F \in \text{Subst}(E), A \text{ st } F\}
\]

**Definition 2.5** A term \(E \in \mathcal{L}\) is guardedly closed if and only if for each constant \(A \in \text{Const}(E)\)

- \(A\) is equipped in \(\text{Def}\) with exactly one defining equation \(A \triangleq E'\), and
- there exists \(F \in \text{Subst}(E')\) such that, whenever an instance of a constant \(B\) satisfies \(B \text{ st } F\), then the same instance satisfies \(B \text{ st } \langle a, \lambda \rangle . G \text{ st } F\).

In the following we consider only the set \(\mathcal{G}\) of guardedly closed terms in \(\mathcal{L}\). Moreover, in order to avoid ambiguities, we assume the binary operators to be left-associative and we intro-

\(^5\) For an overview of synchronization disciplines on action rates in stochastic process algebras, the reader is referred to [40].
duce the following operator-precedence relation: functional abstraction = temporal restriction =
functional relabeling > prefix > parallel composition > alternative composition.

2.3 Race policy
Because of the presence of binary operators such as the alternative composition and the parallel
composition ones, the situation in which several active actions are simultaneously executable can
arise. Both in the case of the alternative composition (due to the choice it expresses) and in the
case of the parallel composition (as we have adopted an interleaving model, hence representing
the execution of only one action at a time), we need a mechanism for choosing the action to be
executed. Following the proposal in [5, 34, 39], we adopt the race policy: the action sampling the
least duration succeeds.  

If we consider a term enabling only exponentially timed actions, the race policy establishes
that (i) the random variable describing the sojourn time in the state corresponding to the term at
hand is the minimum of the exponentially distributed random variables describing the duration
of the enabled actions, and (ii) the execution probability of each enabled action is determined as
well by the exponentially distributed random variables describing the duration of the enabled
actions. In order to compute the two quantities above, we exploit the following property.

Proposition 2.6 If \(\{Y_i \mid 1 \leq i \leq n\}\) is a set of independent exponentially distributed random
variables with rates \(\lambda_1, \lambda_2, \ldots, \lambda_n\), respectively, and a random variable \(Y\) is defined by

\[ Y = \min\{Y_i \mid 1 \leq i \leq n\} \]

then \(Y\) is exponentially distributed with rate

\[ \lambda = \sum_{i=1}^{n} \lambda_i \]

As a consequence, if \(n\) exponentially timed actions \(\langle a_1, \lambda_1 \rangle, \langle a_2, \lambda_2 \rangle, \ldots, \langle a_n, \lambda_n \rangle\) are
enabled, then the sojourn time is exponentially distributed with rate \(\lambda = \sum_{i=1}^{n} \lambda_i\) and the execution probability of \(\langle a_k, \lambda_k \rangle, 1 \leq k \leq n\), is given by \(\lambda_k / \lambda\).

If we consider instead a term enabling both exponentially timed and immediate actions, the
race policy establishes that (i) the exponentially timed actions cannot be executed at all because
they cannot sample zero durations, and (ii) the sojourn time in the state corresponding to the
term at hand is zero. Only the enabled immediate actions having the highest priority level are
actually executable, and the choice among them is probabilistically made by giving each of them
an execution probability proportional to its weight. As a consequence, if \(n\) immediate actions
\(\langle a_1, w_1 \rangle, \langle a_2, w_2 \rangle, \ldots, \langle a_n, w_n \rangle\) are enabled and no immediate action with higher
priority level is enabled, then the execution probability of \(\langle a_k, w_k \rangle, 1 \leq k \leq n\), is given by

\[ w_k / w \]

where \(w = \sum_{i=1}^{n} w_i\) is called the total weight of the term at hand.  

As we said at the beginning of the section, the possibility of having several simultaneously
executable active actions in a given term is due to the alternative composition operator and the
parallel composition operator. Concerning the alternative composition operator, we now realize
that in its simpler form it expresses a choice between two active actions (executable by two
alternative terms) whose nature is:

- **Prioritized** if the two active actions have different priority levels. The choice is solved in an:
  - implicit way whenever it concerns an exponentially timed action and an immediate
    action, because the outcome of the choice is implicitly determined by their durations
due to the race policy;
  - explicit way whenever it concerns two immediate actions having different priority
    levels, because the priority levels themselves explicitly determine the outcome of the
    choice.
- **Probabilistic** if the two active actions have the same priority level. The choice is solved in an:

6. For an overview of execution policies in stochastic frameworks, the reader is invited to consult [7].
7. For the sake of uniformity, \(\lambda\) and \(w\) above are called the exit rate of the corresponding term.
- **implicit** way whenever it concerns two exponentially timed actions, because their execution probabilities are implicitly determined by their durations due to the race policy;
- **explicit** way whenever it concerns two immediate actions having the same priority level, because their execution probabilities are explicitly determined by their weights.

Similar considerations hold for the parallel composition operator. Obviously, in this case the choice concerns the interleaving of actions. For example, term

\[ E \equiv <a, \lambda> \cdot F + <b, \infty_{1,1}> \cdot 0 \]

can execute only action \(<b, \infty_{1,1}>, \) while term

\[ F \equiv <a, \lambda> \cdot 0 \| <b, \infty_{1,1}> \cdot 0 \]

can execute action \(<a, \lambda>\) as well but (from an interleaving point of view) only after the completion of action \(<b, \infty_{1,1}>\).

Now we turn our attention to passive actions. Firstly, since their duration is undefined, passive actions cannot be undergone to the race policy: they are not involved in the priority mechanism at all. Secondly, since their duration is fixed only upon synchronizations with active actions of the same type, we must distinguish between two cases. If we consider passive actions that are not synchronized with active actions of the same type, then we can obtain models that are not temporally closed. Nevertheless, these models could be useful to express pure nondeterminism. Since the execution probability of passive actions is undefined, i.e. passive actions can be safely viewed as actions of a classical process algebra, a choice between two passive actions has a **nondeterministic** nature.

If we consider instead passive actions that do engage in synchronizations with active actions of the same type, then their execution probability is assigned a value. Whenever \(n\) passive actions can be separately synchronized with the same active action, we assume that each of the synchronizations is given the same execution probability: such a probability is equal to the execution probability of the active action divided by \(n\). For example, the following terms

\[ E_1 \equiv <a, \lambda> \cdot 0 \| <\cdot>(a, \ast \cdot) \cdot 0 + <a, \ast> \cdot 0 \]
\[ E_2 \equiv <a, \lambda> \cdot 0 \| <\cdot>(a, \ast \cdot) \cdot 0 \| <a, \ast> \cdot 0 \]

comprise two passive actions that can be synchronized with the same active action. The nondeterministic choice between them is solved by a uniform probability distribution that assigns each of the passive actions probability \(1/2\), and this corresponds to the execution probability of each of the resulting synchronizations.

We conclude this brief presentation about the race policy by introducing a problem common to all the stochastic process algebras. Consider a term composed of two alternative identical subterms whose topmost operator is the prefix operator. In the case of a classical process algebra, term

\[ E \equiv a \cdot F + a \cdot F \]

can only perform \(a\) thus becoming \(F\), so it is equivalent to \(a \cdot F\). In the case of a stochastic process algebra such as EMPA, given term

\[ E' \equiv <a, \lambda> \cdot F' + <a, \lambda> \cdot F' \]

we must remember that there are two executable exponentially timed actions because the race policy has been adopted and therefore the exit rate of this term is not \(\lambda\) but \(2 \cdot \lambda\) (by Proposition 2.6).

The same problem arises also in the case of immediate actions. To solve the problem, we can either decorate action executions by auxiliary labels (as proposed in [34] and then in the early version of EMPA [16]), or take into account the multiplicity of action executions (as proposed in [39]). Here we adopt a new solution (illustrated in Section 3) that consists of computing for each term all of its potential action executions and then properly merging those having the same action type, the same priority level and the same derivative term: in the case of \(E'\) above, there are two identical potential action executions \(<<a, \lambda>, F'>\) and \(<<a, \lambda>, F'>\) that are merged together resulting in \(<<a, 2 \cdot \lambda>, F'>\). Since our solution requires to keep track of the multiplicity of every potential action execution, we resort to the formalism of multisets, which is briefly recalled below.

**Definition 2.7** Given a set \(S\), a multiset over \(S\) is a function \(M : S \rightarrow \mathbb{N}\), and a finite multiset over \(S\) is a function \(M : S \rightarrow \mathbb{N}\) such that the set \(\text{dom}(M) = \{s \in S \mid M(s) \neq 0\}\) is finite. The value
$M(s)$ is called the *multiplicity* of element $s$. We denote by $\mathcal{M}_u(S)$ the set of all the multisets over $S$, $\mathcal{M}_{\text{fin}}(S)$ the set of all the finite multisets over $S$, and $\mathcal{P}_{\text{fin}}(S)$ the set of all the finite sets over $S$. 

### Definition 2.8

Given a set $S$, let $M_1, M_2, M \in \mathcal{M}_u(S)$ and $M' \in \mathcal{M}_u(S \times S)$. Then:

- $s \in M \iff M(s) > 0$;
- $M_1 \subseteq M_2 \iff \forall s \in S. M_1(s) \leq M_2(s)$;
- $M = M_1 \oplus M_2 \iff \forall s \in S. M(s) = M_1(s) + M_2(s)$;
- $M = M_1 \odot M_2 \iff \forall s \in S. M(s) = \max(M_1(s) - M_2(s), 0)$;
- $M' = M_1 \odot M_2 \iff \forall (s_1, s_2) \in S \times S. M'(s_1, s_2) = M_1(s_1) \cdot M_2(s_2)$. 

We use “$|$” and “$\|$” as brackets for multisets, and “$\emptyset$” to denote the empty multiset. When describing multisets by means of brackets, we adopt the following conventions about the multiplicity of their elements:

- Let $S_1, \ldots, S_n$ be $n \geq 1$ sets, let $M_1, \ldots, M_n$ be $n$ multisets over $S_1, \ldots, S_n$, respectively, let $M'$ be a multiset over $S_1 \times \ldots \times S_n$, and let $\psi$ be a predicate over $S_1 \times \ldots \times S_n$. Then
  \[
  M' = \{ (s_1, \ldots, s_n) \mid \psi(s_1, \ldots, s_n) \} \quad \text{if and only if for each } s = (s_1, \ldots, s_n) \in S_1 \times \ldots \times S_n
  \]
  \[
  \psi(s) \implies M'(s) = \prod_{i=1}^n M_i(s_i)
  \]
  \[
  \neg \psi(s) \implies M'(s) = 0
  \]

- Let $S_1, \ldots, S_n$ be $n \geq 1$ sets, let $M$ be a multiset over $S_1 \times \ldots \times S_n$, and let $M'$ be a multiset over $S_i$ where $1 \leq i \leq n$, and let $\psi$ be a predicate over $S_i$. Then
  \[
  M' = \{ s \mid \psi(s) \} \quad \text{if and only if for each } s \in S_i
  \]
  \[
  \psi(s) \implies M'(s) = \sum_{s \in S_1 \times \ldots \times S_n \cap S_i \times \ldots \times S_n} M(s_1, \ldots, s_i, \ldots, s_n)
  \]
  \[
  \neg \psi(s) \implies M'(s) = 0
  \]

### Definition 2.9

Let $S_1, \ldots, S_n$ be $n \geq 1$ sets. We call the $i^{th}$ *multiset projection* over the given sets, $1 \leq i \leq n$, the function $\pi_i : \mathcal{M}_u(S_1 \times \ldots \times S_n) \rightarrow \mathcal{M}_u(S_i)$ defined by

\[
\pi_i(M) = \{ s \mid \forall i \neq j. s \in S_j \iff s_j \in S_j \}
\]
### 3.1 Rooted labeled transition systems

In this section we recall the definition of rooted labeled transition system and some related notions [56].

**Definition 3.1** A rooted labeled transition system (LTS) is a quadruple 

\[(S, U, \rightarrow, s_0)\]

such that:
- \(S\) is a set whose elements are called states;
- \(U\) is a set whose elements are called labels;
- \(\rightarrow \subseteq S \times U \times S\) is called transition relation;
- \(s_0 \in S\) is called the initial state.

In the graphical representation of a LTS, states are drawn as black dots and transitions are drawn as arrows between pairs of states with the appropriate labels; the initial state is pointed to by an unlabeled arrow (for an example, see Figure 3). Below we recall two notions of equivalence for LTSs: the first one (isomorphism) concerns the structure of the LTSs, the second one (bisimilarity) concerns their behavior.

**Definition 3.2** Let \(Z_1 = (S_1, U, \rightarrow_1, s_{01})\) and \(Z_2 = (S_2, U, \rightarrow_2, s_{02})\) be two LTSs.
- \(Z_1\) is isomorphic to \(Z_2\) if and only if there exists a bijection \(\beta : S_1 \rightarrow S_2\) such that:
  - \(\beta(s_{01}) = s_{02}\);
  - for each \(s, s' \in S_1\) and for each \(u \in U\)
    \[s \rightarrow_1 s' \iff \beta(s) \rightarrow_2 \beta(s')\]
- \(Z_1\) is bisimilar to \(Z_2\) if and only if there exists a relation \(B \subseteq S_1 \times S_2\) such that:
  - \((s_{01}, s_{02}) \in B\);
  - for each \((s_1, s_2) \in B\) and for each \(u \in U\)
    * whenever \(s_1 \rightarrow_1 s'_1\), then \(s_2 \rightarrow_2 s'_2\) and \((s'_1, s'_2) \in B\);
    * whenever \(s_2 \rightarrow_2 s'_2\), then \(s_1 \rightarrow_1 s'_1\) and \((s'_1, s'_2) \in B\).

### 3.2 Integrated operational interleaving semantics

The main problem in the definition of the integrated operational interleaving semantics for EMPA is that the actions executable by a given term may have different priority levels, and only those having the highest priority level are actually executable. Let us call potential move of a given term a pair composed of (i) an action executable by the term, and (ii) a derivative term obtained by executing that action. We compute inductively all the potential moves of a given term regardless of priority levels, and then we select those having the highest priority level. Computing inductively all the potential moves of a given term instead of a single potential move at a time is motivated, in a stochastic framework, by the fact that the actual executability as well as the execution probability of an action depend upon all the actions that are executable at the same time when it is executable: only if we know all the potential moves of the subterms of a given term, we can correctly determine its transitions and their rates. This is clarified by the following example.

**Example 3.3** Consider term

\[E \equiv <a, \infty_{3,1}>, E_1 + <e, \infty_{2,1}>, A + <g, *>, \emptyset\]

where

\[E_1 \equiv <b, \lambda>, (\|\emptyset\|) + <e, \infty_{1,1}>, E_2\]
\[E_2 \equiv <h, \xi>, E_3 + <h, \xi>, E_3\]
\[E_3 \equiv <d, \mu>, \emptyset, (\|a\|), (\|d, *\|), \emptyset, (\|d, *\|), \emptyset, A\]

Suppose to apply to \(E\) standard rules for classical process algebras, thereby disregarding priority levels. Then we obtain the LTS reported in Figure 3(a) where
Figure 3. Integrated interleaving model of term $E$
\[
E_4 \equiv \emptyset \parallel (\emptyset \parallel <d, \ast \parallel 0) \\
E_5 \equiv \emptyset \parallel (\emptyset \parallel <d, \ast \parallel 0) \parallel \emptyset
\]

Now assume that priority levels are taken into account. Then lower priority transitions must be pruned, thus resulting in the LTS reported in Figure 3(b): note that the passive transition has not been discarded. This is obtained by means of an auxiliary function we shall call Select.

Finally, consider the rate of the transition from \(E_2\) to \(E_3\) and the two transitions from \(E_3\) to \(E_4\) and \(E_5\). In the correct semantic model for \(E\), such rates have to be like in Figure 3(c). Concerning the transition from \(E_2\) to \(E_3\), its rate is \(2 \cdot \xi\) instead of \(\xi\) because in \(E_2\) two exponentially timed actions with rate \(\xi\) occur and the race policy has been adopted (see Proposition 2.6). The problem is that both exponentially timed actions have the same type and results in the same derivative term, so with classical rules only one transition is produced. The same problem arises in the case of immediate actions. To overcome this, we compute the multiset of all the potential moves, and then we construct transitions by using an auxiliary function we shall call Melt that merges together those potential moves having the same action type, the same priority level and the same derivative term. The rate of transitions deriving from the merging of potential moves is computed by means of another auxiliary function we shall call Min.

Concerning the transitions from \(E_3\) to \(E_4\) and \(E_5\), their rate is \(\mu/2\) instead of \(\mu\) because in \(E_3\) only one exponentially timed action with rate \(\mu\) occurs hence its exit rate is \(\mu\) the value \(\mu/2\) stems from the assumption about the execution probability of passive actions made in Section 2.3. If the rate of these transitions were \(\mu\), then the exit rate would be \(2 \cdot \mu\) (by virtue of Proposition 2.6) and this would contradict the definition of \(E_3\). The same considerations hold if in \(E_3\) we have an immediate action instead of an exponentially timed action, or alternative passive actions instead of independent passive actions. In all of these cases a normalization of rates is required, and this is carried out by means of an auxiliary function we shall call Norm.

The reader is invited to look again at this example after examining the formal definition of the semantics, in order to verify that the LTS of Figure 3(c) is exactly the result of the application of the rules reported in Table 1 equipped with the auxiliary functions mentioned above.

Now let us turn our attention to the formal definition of the integrated operational interleaving semantics for EMPA. The transition relation \(\longrightarrow\) is the least subset of \(\mathcal{G} \times \mathcal{A} \times \mathcal{G}\) satisfying the inference rule reported in the first part of Table 1. This rule selects the potential moves having the highest priority level, and then merges together the remaining potential moves having the same action type, the same priority level and the same derivative term. The first operation is carried out through functions \(\text{Select} : \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G}) \rightarrow \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G})\) and \(PL : \mathcal{A} \rightarrow \mathcal{P}_{\text{Set}},\) with \(\mathcal{P}_{\text{Set}} = \{-1\} \cup \mathbb{N},\) which are defined in the third part of Table 1. The second operation is carried out by functions \(\text{Melt} : \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G}) \rightarrow \mathcal{P}_{\text{fin}}(\mathcal{A} \times \mathcal{G})\) and \(\text{Min} : (\mathcal{A} \mathcal{R} \rightarrow \mathcal{A} \mathcal{R} \rightarrow \mathcal{A} \mathcal{R} \rightarrow \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G}) \times \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G})) \rightarrow \mathcal{A} \mathcal{R},\) which are defined in the fourth part of Table 1. Function Min is the operation computing the minimum of a set of random variables at the same priority level: with abuse of notation, it is extended to multisets of action rates having the same priority by assuming both it acts as the identity function when applied to singletons, and it is commutative and associative when applied to multisets with more than one element.

The multiset \(PM (E)\) of potential moves of term \(E \in \mathcal{G}\) is defined by structural induction as the least element of \(\mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G})\) satisfying the rules reported in the second part of Table 1. The way in which such a multiset is worked out should result clear from the informal explanation of the semantics of operators given in Section 2.2. The only caution concerns the need of normalizing the rates of potential moves resulting from the synchronization of the same active action with several passive actions that are either independent (i.e., occurring in terms composed in parallel with a synchronization set not containing the action type at hand), or alternative (i.e., occurring in terms composed in alternative). This operation is carried out through functions \(\text{Norm} : (\mathcal{A} \mathcal{R} \rightarrow \mathcal{A} \mathcal{R} \times \mathcal{A} \mathcal{R} \times \mathcal{A} \mathcal{R} \times \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G}) \times \mathcal{M}_{\text{fin}}(\mathcal{A} \times \mathcal{G})) \rightarrow \mathcal{A} \mathcal{R}\) and \(\text{Split} : (\mathcal{A} \mathcal{R} \times \mathcal{R}_{[0,1]}) \rightarrow \mathcal{A} \mathcal{R},\) which are defined in the fifth part of Table 1. It is worth noting that \(\text{Norm}(a, \lambda, \mu, PM_1, PM_2)\) is defined if and only if \(\min(\lambda, \mu) = \ast,\) which is the condition on
\[
\frac{\langle a, \lambda, E' \rangle \in \text{Melt}(\text{Select}(PM(E)))}{E \xrightarrow{a, \lambda} E'}
\]

\begin{align*}
PM(\emptyset) &= \emptyset \\
PM(\langle a, \lambda, E \rangle) &= \emptyset \cup \{ (\langle a, \lambda, E \rangle) \} \\
PM(E/L) &= \emptyset \cup \{ (\langle a, \lambda, E'/L \rangle) | (\langle a, \lambda, E' \rangle) \in PM(E) \land a \notin L \} \cup \{ (\langle \tau, \lambda, E'/L \rangle) | (\langle a, \lambda, E' \rangle) \in PM(E) \land a \in L \} \\
PM(E/H) &= \emptyset \cup \{ (\langle a, \lambda, E'/H \rangle) | (\langle a, \lambda, E' \rangle) \in PM(E) \land \neg(a \in H \land \lambda = *) \} \\
PM(E[\varphi]) &= \emptyset \cup \{ (\varphi(a), \lambda, E'[\varphi]) | (\langle a, \lambda, E' \rangle) \in PM(E) \} \\
PM(E_1 + E_2) &= PM(E_1) \oplus PM(E_2) \\
PM(\langle a, \lambda, E _1 \parallel E _2 \rangle) &= \emptyset \cup \{ (\langle a, \lambda, E _1 \parallel E _2 \rangle) | a \notin S \land \neg(\langle a, \lambda, E _1 \rangle) \in PM(E_1) \} \cup \{ (\langle a, \lambda, E _1 \parallel E _2 \rangle) | a \notin S \land (\langle a, \lambda, E _1 \rangle, E _2 \rangle) \in PM(E_2) \} \cup \{ (\langle a, \lambda, E _1 \parallel E _2 \rangle) | a \in S \land (\langle a, \lambda, E _1 \rangle, E _2 \rangle) \in PM(E_1) \land (\langle a, \lambda, E _2 \rangle, E _1 \rangle) \in PM(E_1) \land \gamma = \text{Norm}(a, \lambda, E_1, PM(E_1), PM(E_2)) \} \\
PM(A) &= PM(E) \quad \text{if} \ A \overset{\Delta}{=} E \\
S_{\text{Select}}(PM) &= \emptyset \cup \{ (\langle a, \lambda, E \rangle) \in PM \mid PL(\langle a, \lambda \rangle) = -1 \lor \forall(\langle b, \mu \rangle, E') \in PM \cdot PL(\langle a, \lambda \rangle) \geq PL(\langle b, \mu \rangle) \} \\
PL(\langle a, * \rangle) &= -1 \quad PL(\langle a, \lambda \rangle) = 0 \quad PL(\langle a, \infty_{l,w} \rangle) = 1 \\
\text{Melt}(PM) &= \{ (\langle a, \lambda, E \rangle) \mid (\langle a, \mu, E \rangle) \in PM \land \lambda = \text{Min}(\{ \gamma \mid (\langle a, \gamma, E \rangle) \in PM \land PL(\langle a, \gamma \rangle) = PL(\langle a, \mu \rangle) \}) \} \\
* \text{Min} * &= \quad \lambda \text{Min} \lambda' = \lambda + \lambda' \quad \infty_{l,w} \quad \text{Min} \infty_{l,w} = \infty_{l,w + w} \\
\text{Norm}(a, \lambda, E_1, E_2) &= \{ \langle \text{Split}(\lambda, 1/(\lambda_1( PM_2 ))) (\langle a, * \rangle) \rangle \quad \text{if} \ \mu = * \\
&\langle \text{Split}(\lambda, 1/(\lambda_1( PM_1 ))) (\langle a, * \rangle) \rangle \quad \text{if} \ \lambda = * \\
\text{Split}(*, \alpha) &= \quad \text{Split}(\lambda, \alpha) = \lambda \cdot \alpha \quad \text{Split}(\infty_{l,w}, \alpha) = \infty_{l,w} \cdot \alpha \\
\}
\end{align*}

Table 1. Inductive rules for EMPA integrated interleaving semantics
action rates we have required in Section 2.2 in order for a synchronization to be permitted. We also would like to point out that, if we restrict ourselves to exponentially timed and passive actions, the determination of the rate in case of synchronization by means of function \textit{Norm} can be viewed as a particular case of the formula given in [39].

**Example 3.4** Consider term

$$E \equiv E_1 \parallel [a] (E_2 \parallel E_3)$$

where

$$E_1 \equiv <a, \lambda > 0$$
$$E_2 \equiv <a, * > 0 + <a, * > 0$$
$$E_3 \equiv <a, * > 0$$

Then \(E_1\) has one potential move \((<a, \lambda > 0, E_2)\) with multiplicity two, and \(E_3\) has one potential move \((<a, * > 0, E_3)\). As a consequence, \(E_2 \parallel E_3\) has both potential move \((<a, * > 0, E_3)\) with multiplicity two and potential move \((<a, * >, E_2 \parallel 0)\). Thus, when computing the potential moves for \(E_1\), function \textit{Norm} produces both \((<a, \lambda > 0, E_2 \parallel [a] (0 \parallel E_3))\) with multiplicity two and \((<a, \lambda > 0, E_2 \parallel [a] (E_3))\), and subsequently function \textit{Melt} produces both \((<a, 2 \cdot \lambda > 0, E_2 \parallel [a] (0 \parallel E_3))\) and \((<a, \lambda > 0, E_2 \parallel [a] (E_3))\), as expected.

We are now in a position of introducing the integrated operational interleaving semantics of a given term.

**Definition 3.5** The \textit{integrated operational interleaving semantics} of a term \(E \in \mathcal{G}\) is the LTS

$$\mathcal{I}[E] = (\uparrow E, \text{Act}, \rightarrow_{\mathcal{I}}, E)$$

where:

- \(\uparrow E\) is the least subset of \(\mathcal{G}\) such that:
  - \(E \in \uparrow E\);
  - if \(E_1 \in \uparrow E\) and \(a, \lambda \rightarrow_{\mathcal{I}} E_2\), then \(E_2 \in \uparrow E\);
- \(\rightarrow_{\mathcal{I}}\) is the restriction of \(\rightarrow\) to \(\uparrow E \times \text{Act} \times \uparrow E\).

Borrowing the terminology of generalized stochastic Petri nets [5], a state of \(\mathcal{I}[E]\) is called \textit{tangible} if it has at least one exponentially timed transition, \textit{vanishing} if it has at least one immediate transition. Because of function \textit{Select}, a tangible state has only exponentially timed transitions and, possibly, passive transitions; likewise, a vanishing state has only immediate transitions of the same priority level and, possibly, passive transitions.

Given a term \(E \in \mathcal{G}\), its integrated operational interleaving semantics \(\mathcal{I}[E]\) fully represents the behavior of \(E\) because transitions are decorated by both the action type and the action rate, hence both the functional aspect and the performance aspect are described. In order to implement fully the first phase of the integrated approach of Figure 1, we need to derive two projected semantic models concerning functionality and performance, respectively. Evidently, one can think of obtaining the \textit{functional semantics} \(\mathcal{F}[E]\) and the \textit{performance semantics} \(\mathcal{P}[E]\) of term \(E\) from its integrated operational interleaving semantics \(\mathcal{I}[E]\) by simply dropping action rates and action types, respectively. As a matter of fact, this is the case for the functional semantics, and also for the performance semantics whenever only exponentially timed transitions are involved. Below we introduce the definition of the functional semantics, while the definition of the performance semantics is deferred to Section 4 since it requires a more careful treatment due to the possible presence of immediate and passive transitions.

**Definition 3.6** The \textit{functional semantics} of a term \(E \in \mathcal{G}\) is the LTS

$$\mathcal{F}[E] = (\uparrow E, \text{AType}, \rightarrow_{\mathcal{F}}, E)$$

where \(\rightarrow_{\mathcal{F}}\) is the restriction of \(\rightarrow\) to \(\uparrow E \times \text{AType} \times \uparrow E\).

Based on the operational interleaving semantics, we formalize the property of temporal closure.
Definition 3.7 A term $E \in G$ is temporally closed if and only if $I[E]$ is isomorphic to $I[E \setminus AType]$.

A temporally closed term can contain passive actions in its syntax but its integrated interleaving semantic model does not contain passive transitions, thereby representing a system whose behavior is completely specified from the performance viewpoint. We denote by $T$ the set of terms in $L$ that are temporally closed. We also denote by $\mathcal{E}$ the set of terms in $L$ that are guardedly and temporally closed, i.e. $\mathcal{E} = G \cap T$.

4 Performance semantics of EMPA terms

In this section we complete the description of the implementation of the first phase of the integrated approach of Figure 1 by showing the performance projection of the integrated operational interleaving semantics, i.e. the performance semantics.

Since in EMPA the durations of exponentially timed actions are expressed through exponentially distributed random variables, it is natural to associate with each term a homogeneous continuous-time Markov chain (HCTMC) acting as a performance model. Given a term $E \in \mathcal{E}$, the idea is to obtain its performance semantics $P[E]$, hereafter called Markovian semantics and denoted by $\mathcal{M}[E]$, by adequately manipulating $I[E]$. Formally, $\mathcal{M}[E]$ represents the state transition rate diagram of the HCTMC associated with $E$, so it turns out to be defined as a variant of a LTS, called probabilistically rooted labeled transition system, in which there is no initial state but a probability mass function defined over the set of states that determines for each of them the probability that it is the initial state.

This section is organized as follows. In Section 4.1 we introduce some notions about probabilistically rooted labeled transition systems, since they are the means whereby the semantic model is expressed in this framework. In Section 4.2 we recall some notions about Markov chains. In Section 4.3 we define the Markovian semantics of EMPA terms.

4.1 Probabilistically rooted labeled transition systems

In this section we present the definition of probabilistically rooted labeled transition system as well as the related definitions of p-isomorphism and p-bisimilarity.

Definition 4.1 A probabilistically rooted labeled transition system (PLTS) is a quadruple $(S, U, \longrightarrow, P)$ such that:

- $S, U, \longrightarrow$ are defined as for a LTS;
- $P : S \rightarrow [0, 1]$ is called initial state probability function and is such that $\sum_{s \in S} P(s) = 1$.

In the graphical representation of a PLTS, states and transitions are drawn as in a LTS, and each state is labeled with its initial state probability unless it is zero (for an example, see Figure 4(b) and (c)). In this paper we consider only PLTSs whose set of labels is contained in $\mathbb{R}_+ \cup \text{Inf}$, such that the transitions leaving a state are either all labeled with elements of $\mathbb{R}_+$ or all labeled with elements of Inf having the same priority level. The notion of p-isomorphism for such PLTSs carries over from the corresponding notion for LTSs, while the definition of p-bisimilarity is developed in the style of [45] by considering the aggregated rate with which from a state it is possible to reach a class of states.

Definition 4.2 Let $Z_1 = (S_1, \mathbb{R}_+ \cup \text{Inf}, \longrightarrow_1, P_1)$ and $Z_2 = (S_2, \mathbb{R}_+ \cup \text{Inf}, \longrightarrow_2, P_2)$ be two PLTSs.

- $Z_1$ is p-isomorphic to $Z_2$ if and only if there exists a bijection $\beta : S_1 \rightarrow S_2$ such that:
  - for each $s \in S_1$
    $$P_1(s) = P_2(\beta(s))$$
for each \( s, s' \in S_1 \) and for each \( \hat{\lambda} \in \mathbb{R}_+ \cup \text{Inf} \)
\[
\hat{\lambda} \xrightarrow{\beta(s)} \hat{\lambda} \quad \iff \quad \beta(s) \xrightarrow{\beta(s')} \hat{\lambda}
\]

- \( Z_1 \) is p-bisimilar to \( Z_2 \) if and only if there exists an equivalence relation \( B \subseteq (S_1 \cup S_2) \times (S_1 \cup S_2) \), whose quotient set is denoted by \((S_1 \cup S_2)/B\), such that:
  - for each \( C \in (S_1 \cup S_2)/B \)
    \[
    \sum_{s \in C \cap S_1} P_1(s) = \sum_{s \in C \cap S_2} P_2(s)
    \]
  - whenever \((s_1, s_2) \in B \cap (S_1 \times S_2)\), then for each \( C \in (S_1 \cup S_2)/B \)
    \[
    \min\{\|\hat{\lambda} \mid s_1 \xrightarrow{\hat{\lambda}} s_1' \wedge s_1' \in C \cap S_1\} = \min\{\|\hat{\lambda} \mid s_2 \xrightarrow{\hat{\lambda}} s_2' \wedge s_2' \in C \cap S_2\}
    \]

The following proposition states a simple necessary condition for p-bisimilarity based on exit rates; it will be useful in the following.

**Proposition 4.3** Let \( Z_1 = (S_1, \mathbb{R}_+ \cup \text{Inf}, \longrightarrow_1, P_1) \) and \( Z_2 = (S_2, \mathbb{R}_+ \cup \text{Inf}, \longrightarrow_2, P_2) \) be two PLTSs, and let \( B \) be a p-bisimulation for them. If \((s_1, s_2) \in B \cap (S_1 \times S_2)\), then
\[
\min\{\|\hat{\lambda} \mid s_1 \xrightarrow{\hat{\lambda}} s_1' \wedge s_1' \in C \cap S_1\} = \min\{\|\hat{\lambda} \mid s_2 \xrightarrow{\hat{\lambda}} s_2' \wedge s_2' \in C \cap S_2\}
\]

### 4.2 Markov chains and lumping

In this section we recall the definition of HCTMC as well as some ways of representing HCTMCs and some related properties [43, 44].

**Definition 4.4** A *continuous-time Markov chain* (CTMC) is a continuous-time stochastic process

\[
X = \{X(t) \mid t \in T\}
\]

with discrete state space \( S_X \) such that, for each \( n \in \mathbb{N}_+ \), \( i_0, \ldots, i_{n-1}, i_n \in S_X \), \( t_0, \ldots, t_{n-1}, t_n \in T \) where \( t_0 < \ldots < t_{n-1} < t_n \), it turns out

\[
\Pr\{X(t_n) = i_n \mid X(t_{n-1}) = i_{n-1} \wedge \ldots \wedge X(t_0) = i_0\} = \Pr\{X(t_n) = i_n \mid X(t_{n-1}) = i_{n-1}\}
\]

**Definition 4.5** Let \( X \) be a CTMC.

- The *transition matrix* of \( X \) from time \( t \in T \) to time \( t' \in T \) is matrix \( P_X(t, t') \) defined by
  \[
P_X(t, t') = \{\Pr(X(t') = j \mid X(t) = i)\}_{i,j \in S_X}
  \]

- The *infinitesimal generator* of \( X \) at time \( t \in T \) is matrix \( Q_X(t) \) defined by
  \[
  Q_X(t) = [q_{i,j}(t)]_{i,j \in S_X} = \lim_{\Delta t \to 0} \frac{P_X(t, t + \Delta t) - I}{\Delta t}
  \]
  where \( I \) is the identity matrix.

- \( X \) is a *homogeneous CTMC* (HCTMC) if and only if its infinitesimal generator is independent of the time.

- The *state probability distribution function* of \( X \) at time \( t \in T \) is vector \( \pi_X(t) \) defined by
  \[
  \pi_X(t) = \{\Pr(X(t) = i)\}_{i \in S_X}
  \]

- The *steady-state probability distribution function* of \( X \) is vector \( \pi_X \) defined by
  \[
  \pi_X = \lim_{t \to \infty} \pi_X(t)
  \]

A HCTMC \( X \) is represented by means of its infinitesimal generator when we wish to determine its state probability distribution functions, from which performance indices of interest can be derived. Whenever the steady-state probability distribution function exists, it can be determined by solving
It is worth noting that, since every HCTMC $X$ is a memoryless continuous-time stochastic process by definition, and since the exponential distribution is the only memoryless continuous distribution, entries $q_{i,j}$ of the infinitesimal generator of $X$ can be interpreted as the rates of the exponential distributions describing the duration of state transitions. As a consequence, the HCTMC $X$ can equivalently be represented by means of the PLTS $(S_X, \Xi, \{(i, q_{i,j}, j) \in S_X \times \mathbb{R}_+ \times S_X \mid q_{i,j} > 0\}, \pi_X(0))$ which corresponds to the state transition rate diagram of $X$.

We conclude with the notion of ordinary lumping. The state space of a HCTMC is ordinarily lumpable with respect to one of its partitions whenever, given two classes of the partition, all the states in one of the two classes have the same aggregated rate to reach the other class.

**Definition 4.6** Let $X$ be a HCTMC. A partition $\Lambda$ of $S_X$ is an *ordinary lumping* of $X$ if and only if for every $C_i, C_j \in \Lambda$ and $h, l \in C_i$:

$$\sum_{k \in C_j} q_{h,k} = \sum_{k \in C_j} q_{l,k}$$

If this is the case, the *ordinarily lumped* HCTMC $X'$ obtained from $X$ has state space $\Lambda$ and infinitesimal generator $Q'_{X}$, where $q'_{i,j} = \sum_{k \in C_j} q_{h,k}$ for some $h \in C_i$. ■

It is easily seen that, if $X$ is a HCTMC and $X'$ is the HCTMC obtained from $X$ via the ordinary lumping $\Lambda$, then the PLTSs underlying $X$ and $X'$ are $p$-bisimilar via the reflexive, symmetric and transitive closure of the relation that associates each state of $X$ with the state of $X'$ that contains it. The following proposition states a simple necessary condition for ordinary lumpability based on exit rates; it will be useful in the following.

**Proposition 4.7** Let $X$ be a HCTMC and let $\Lambda$ be an ordinary lumping of $X$. If $h, l \in C \in \Lambda$, then

$$\sum_{k \in S_X} q_{h,k} = \sum_{k \in S_X} q_{l,k}$$

The notion of ordinary lumping is a kind of state aggregation that allows an exact analysis to be carried out on a smaller HCTMC. This means that, whenever the steady-state probability distribution function exists, the steady state probability of each partition is the sum of the steady-state probabilities of the states it contains. Other kinds of exact aggregations for HCTMCs, such as the *exact lumping* [64] which guarantees that all the states in the same partition have the same steady-state probability, have appeared in the literature but are not considered in this paper.

### 4.3 Markovian semantics

In this section we present the algorithm transforming $T[\mathcal{E}]$ into $\mathcal{M}[\mathcal{E}]$ for any $E \in \mathcal{E}$. It is organized in two phases.

1. The first phase drops action types and eliminates all the immediate transitions occurring in $T[\mathcal{E}]$. The removal of the immediate transitions and the related vanishing states is justified from a stochastic point of view by the fact that the sojourn time in a vanishing state is zero.
2. The second phase detects and merges states that are equivalent according to the notion of ordinary lumping, in order to reduce the state space of the HCTMC obtained at the end of the previous phase. This second phase can be viewed as optional, since it may lead to information loss because of state merging. As a consequence, it can be safely applied only when such an information loss does not happen. This is the case of, e.g., queueing systems, as it has been proved in [17].

Due to its generality, such an algorithm can be regarded as an alternative to the technique of the embedded Markov chain, which has been used, e.g., to define the HCTMC underlying a generalized stochastic Petri net [5].
Example 4.8 Consider term

\[
E \equiv \langle a, \lambda \rangle . E_1
\]

where

\[
E_1 \equiv \langle b, \infty_{1,2} \rangle . A + \langle c, \infty_{0,1} \rangle . B
\]

\[
A \triangleq \langle e, \mu \rangle . B
\]

\[
B \triangleq \langle f, \mu \rangle . A
\]

The LTS \( I[E] \) is reported in Figure 4(a). To obtain the Markovian semantics of \( E \), we have to eliminate the vanishing state \( E_1 \) together with its immediate transitions. This is carried out by splitting the exponentially timed transition from \( E \) to \( E_1 \) into two exponentially timed transitions from \( E \) to \( A \) and \( B \) whose rates are \( 2 \cdot \lambda/3 \) and \( \lambda/3 \), respectively: factors \( 2/3 \) and \( 1/3 \) are the execution probabilities of the two immediate transitions. Note that the splitting operation preserves the exit rate of \( E \). The resulting PLTS is reported in Figure 4(b).

Afterwards, we discover that states \( A \) and \( B \) have the same aggregated rate to reach any of the other states, so they can be ordinarily lumped together. The resulting PLTS is reported in Figure 4(c). Note that the two exponentially timed transitions from \( E \) to \( A \) and \( B \) have been merged into a single transition from \( E \) to \( AB \) whose rate is the sum of the rates of the two transitions, while the transition from \( A \) to \( B \) and the transition from \( B \) to \( A \) have just been merged into a single one from \( AB \) to itself.

The reader is invited to look again at this example after examining the formal definition of the semantics, in order to verify that the PLTSs in Figure 4(b) and 4(c) are the result of the first and the second phase of the algorithm, respectively.
4.3.1 First phase: eliminating immediate transitions

The first phase of the algorithm drops action types and eliminates immediate transitions. Such an elimination causes vanishing states to be removed and transitions entering vanishing states to be split. This phase comprises several steps.

The first step consists of
1. dropping action types,
2. removing selfloops composed of an immediate transition (hereafter called immediate selfloops for short),
3. changing the weight of each immediate transition into the corresponding execution probability, and
4. determining the initial state probability function.

More accurately, the first step consists of obtaining from the LTS \( I = (S, E, \text{Act}, \longrightarrow_E, E) \) the PLTS

\[
PLTS_{E,1} = (S_{E,1}, \bigcup \mathbb{R}_+ \cup \mathbb{I}n, \longrightarrow_{E,1}, P_{E,1})
\]

where:

- \( S_{E,1} = \downarrow E \)
- let us define for every \( s \in S_{E,1} \)

\[
PM_1(s) = \{ \langle \lambda, s' \rangle \mid s \xrightarrow{a,\lambda} E s' \}
\]

then \( \longrightarrow_{E,1} \) is the least subset of \( S_{E,1} \times (\mathbb{R}_+ \cup \mathbb{I}n) \times S_{E,1} \) such that:

- if \( s \) is tangible and \( \langle \lambda, s' \rangle \in Melt(PM_1(s)) \), then

\[
s \xrightarrow{E,1} s'
\]

- if \( s \) is vanishing and in \( Melt(PM_1(s)) \) there are exactly \( m \geq 1 \) potential moves \( \langle \infty, w_j, s_j \rangle, 1 \leq j \leq m \), such that \( s_j \neq s \), then there are \( m \) transitions

\[
s \xrightarrow{E,1} s_j, 1 \leq j \leq m;
\]

where \( w = \sum_{j=1}^m w_j \);
- \( P_{E,1} : S_{E,1} \rightarrow \mathbb{R}_{[0,1]} \) such that

\[
P_{E,1}(s) = \begin{cases} 
1 & \text{if } s \equiv E \\
0 & \text{if } s \not\equiv E
\end{cases}
\]

The \( k \)-th step, \( k \geq 2 \), exploits the fact that the sojourn time in a vanishing state is zero, and consists of applying the graph reduction rule reported in Figure 5 to a given vanishing state \( s_0 \in S_{E,k-1} \). With this step we thus consider a fork of immediate transitions that is treated by

1. eliminating the related vanishing state as well as the immediate transitions themselves,
2. splitting the transitions entering the state upstream the fork,
3. removing immediate selfloops created by splitting immediate transitions leaving one of the states downstream the fork and entering the state upstream the fork, and
4. distributing the initial state probability associated with the state upstream the fork among the states downstream the fork.

---

**Figure 5. Graph reduction rule**
More accurately, assuming that the fork considered at the \( k \)-th step is the one shown in Figure 5, let

\[
P_k[E] = (S_{E,k}, \mathbb{R}_+ \cup \text{Inf}, \longrightarrow_{E,k}, P_{E,k})
\]

where:

- \( S_{E,k} = S_{E,k-1} - \{s_0\} \);
- let us define for every \( s \in S_{E,k} \)
  \[
  PM_k(s) = \{ (\lambda', s') | s \longrightarrow_{E,k-1} s' \land s' \neq s_0 \} \oplus \{ (\text{Split}(\lambda', p_i), s_i) | s \longrightarrow_{E,k-1} s_0 \land 1 \leq i \leq n \}
  \]

then \( \longrightarrow_{E,k} \) is the least subset of \( S_{E,k} \times (\mathbb{R}_+ \cup \text{Inf}) \times S_{E,k} \) such that:

- if \( s \) is tangible, or vanishing but \( s \not\in \{s_i | 1 \leq i \leq n\} \), and \((\lambda', s') \in \text{Melt}(PM_k(s))\), then
  \[
  s \longrightarrow_{E,k} s'
  \]

- if \( s \) is vanishing, \( s \equiv s_i \) and in \( \text{Melt}(PM_k(s)) \) there are exactly \( m \geq 1 \) potential moves \((\infty_{i,j}, s_j), 1 \leq j \leq m \), such that \( s_j \neq s \), then there are \( m \) transitions
  \[
  s \longrightarrow_{E,k} s_j, 1 \leq j \leq m
  \]

where \( p = \sum_{j=1}^{m} p_i \);
- \( P_{E,k} : S_{E,k} \longrightarrow \mathbb{R}_{[0,1]} \) such that
  \[
  P_{E,k}(s) = \begin{cases} 
  P_{E,k-1}(s) & \text{if } s \not\in \{s_i | 1 \leq i \leq n\} \\
  P_{E,k-1}(s) + P_{E,k-1}(s_0) \cdot p_k & \text{if } s \equiv s_i
  \end{cases}
  \]

It is worth noting that the splitting preserves the exit rate of states having transitions entering the state upstream the fork.

Now we prove the correctness of the first phase of the algorithm. Let \( \mathcal{M}_0[E] \) be the PLTS produced by this phase.

**Theorem 4.9** Let \( E \in \mathcal{E} \).

(i) For every \( k \in \mathbb{N}_+ \) and \( s \in S_{E,k} \) vanishing, \( \sum p | s \longrightarrow_{E,k} s' \| = 1 \).

(ii) For every \( k \in \mathbb{N}_+ \), \( \sum_{s \in S_{E,k}} P_{E,k}(s) = 1 \).

(iii) The elimination of immediate self-loops is correct from the performance viewpoint.

(iv) The graph reduction rule is confluent.

(v) If \( \mathcal{T}[E] \) has finitely many states, then the first phase of the algorithm terminates and \( \mathcal{M}_0[E] \) has no immediate transitions, has finitely many states, and is unique.

**Proof** Let \( E \in \mathcal{E} \).

(i) We proceed by induction on \( k \in \mathbb{N}_+ \):

- If \( k = 1 \) then the result immediately follows from the definition of \( \longrightarrow_{E,1} \).

- Let \( k > 1 \) and let the result hold for \( k - 1 \). Suppose that the fork considered at step \( k \) is the one depicted in Figure 5, and let \( s \in S_{E,k} \):

  * If \( s \not\in \{s' \in S_{E,k-1} | s' \longrightarrow_{E,k-1} s_0\} \) then either \( s \) is tangible (hence the result is not concerned with it), or \( s \) is vanishing but none of its immediate transitions enters \( s_0 \), so the result holds by the induction hypothesis or, if it is downstream the fork, by the renormalization performed at step \( k \).

  * Let \( s \in \{s' \in S_{E,k-1} | s' \longrightarrow_{E,k-1} s_0\} \). If \( s \) is downstream the fork, then the result trivially follows by the renormalization carried out at step \( k \). Assume that \( s \) is not downstream the fork. Since by the induction hypothesis at step \( k - 1 \) it turns out that \( \sum p | s \longrightarrow_{E,k-1} s_i \| = 1 \) and \( \sum p | s \longrightarrow_{E,k-1} s' \| = 1 \), at step \( k \) we have
    \[
    \sum p | s \longrightarrow_{E,k} s' \land s' \neq s_0 \| + \sum p | s \longrightarrow_{E,k} s_0 \| + \sum p | s \longrightarrow_{E,k} s_i \| = 1
    \]

UCLCS-95-14 22
Figure 6. Confluence of the graph reduction rule in absence of immediate selfloops
Figure 7. Confluence of the graph reduction rule in presence of immediate selfloops
We proceed by induction on the length of the longest cycle of immediate transitions. Let us modify the fork of immediate transitions depicted in Figure 5 by assuming that the immediate selfloop is replaced by a transition labeled with \( - \). Starting from \( s_0 \), the probability of reaching \( s_i \) after \( j \) transition executions is \( q^j \), while the probability of reaching \( s_i \) within \( j \) transition executions is \( \sum_{h=0}^{j-1} p_h \cdot q^h \). As \( j \) grows, these probabilities approach 0 and \( p_h / (1 - q) \), respectively.

(iii) Let us modify the fork of immediate transitions depicted in Figure 5 by assuming that \( s_0 \) has also an immediate selfloop labeled with \( \infty_{1,q} \), where \( \sum_{i=1}^{n} p_i + q = 1 \) due to (i). Let us unfold the immediate selfloop by introducing the set of states \( \{ s_{0,j} \mid j \in \mathbb{N}_+ \} \) such that:

- the immediate selfloop is replaced by a transition labeled with \( \infty_{1,q} \) from \( s_0 \) to \( s_{0,1} \);
- for all \( j \in \mathbb{N}_+ \), \( s_{0,j} \) has a transition labeled with \( \infty_{1,p} \), reaching \( s_i \), and a transition labeled with \( \infty_{0,q} \), reaching \( s_{0,j+1} \).

Starting from \( s_0 \), the probability of reaching \( s_{0,j} \) after \( j \) transition executions is \( q^j \), while the probability of reaching \( s_i \) within \( j \) transition executions is \( \sum_{h=0}^{j-1} p_h \cdot q^h \). As \( j \) grows, these probabilities approach 0 and \( p_h / (1 - q) \), respectively.

(iv) We proceed by induction on the length of the longest cycle of immediate transitions in \( T[\mathcal{E}] \).

- If the length of the longest cycle of immediate transitions is \( \leq 1 \), then the first step eliminates all the cycles of immediate transitions (if any). In this case, at each step no immediate selfloop arises, thus making unnecessary the possible renormalization of execution probabilities at states downstream the fork. The confluence of the graph reduction rule then follows: given two forks of immediate transitions, there are the following three cases:

  * There exists a state downstream a fork and upstream the other fork. Figure 6(a) shows that confluence holds in this case. This is achieved by property \( \text{Split}(\text{Split}(\lambda, p), p') = \text{Split}(\lambda, p + p') \).
  
  * There exists at least one state downstream both forks. Figure 6(b) shows that confluence holds in this case as well.
  
  * There is no state shared by the two forks. In such a case, it is obvious that the order in which the two forks are considered is irrelevant.

- Suppose that the length of the longest cycle of immediate transitions is \( c \geq 2 \), and assume that the result holds whenever the length of the longest cycle of immediate transitions is \( < c \). Consider the application of the graph reduction rule to one of the states in the cycle:

  * If no immediate selfloop arises, the confluence is preserved by this step as shown above.
  
  * If an immediate selfloop arises, the confluence is still preserved by this step as shown in Figure 7 due to property \( \text{Split}(\lambda, p) \text{ Min Split}(\lambda, p) = \text{Split}(\lambda, p + p') \). In fact, by exploiting (i), it turns out that

\[
\begin{align*}
\sum_{1 \leq h \leq m \land h \neq i} p'_j \cdot p_h & + \sum_{1 \leq r \leq m \land r \neq j} p'_r = \\
\sum_{1 \leq h \leq m} p'_i \cdot p_h - p'_j \cdot p_i + \sum_{1 \leq r \leq m \land r \neq j} p'_r &=
\end{align*}
\]


\[
\sum_{1 \leq r \leq m} p_r - p_j \cdot p_k = 1 - p_j \cdot p_k
\]
and
\[
d' = \sum_{1 \leq h \leq n, h \neq i} p_h + \sum_{1 \leq r \leq m} p_r \cdot p_j = \sum_{1 \leq h \leq n} p_h - p_i + \sum_{1 \leq r \leq m} p_r \cdot p_j - p_i \cdot p_j =
\]

\[
1 - p_i + p_i - p_i \cdot p_j = 1 - p_i \cdot p_j
\]

and for each \( h = 1, \ldots, n \) such that \( h \neq i \)
\[
p_h + p_i \cdot p_j \cdot p_h/d = p_h(1 + p_i \cdot p_j/(1 - p_j \cdot p_i)) = p_h(1 - p_j \cdot p_i + p_i \cdot p_j)/(1 - p_j \cdot p_i) = p_h/d
\]

The effect of such an application of the graph reduction rule is to shorten the longest cycle of immediate transitions, so the induction hypothesis can be exploited.

(v) If \( \mathcal{T}[E] \) has finitely many states, then \( \mathcal{T}[E] \) has finitely many transitions (\( E \) being guardedly closed). Since each application of the graph reduction rule causes the elimination of \( n \geq 1 \)
immediate transitions, the first phase of the algorithm terminates and \( \mathcal{M}_0[E] \) has no immediate transitions. Moreover, it has finitely many states (as the initial number of states can only decrease during the first phase of the algorithm), and it is unique due to (iv).

4.3.2 Second phase: ordinary lumping

The second phase of the algorithm detects and merges states that are ordinarily lumpable, in order to reduce the number of states of the HCTMC obtained at the end of the first phase.

This phase computes an ordinary lumping \( \Lambda_{E, \mathcal{M}_0} \) of \( \mathcal{M}_0[E] = (S_{E, \mathcal{M}_0}, \mathbb{R}_+, \rightarrow_{E, \mathcal{M}_0}, P_{E, \mathcal{M}_0}) \) by using the procedure reported in Table 2. Such a procedure starts with constructing the partition of \( S_{E, \mathcal{M}_0} \) having as few classes as possible, such that all the states in the same class have the same exit rate (see the necessary condition for ordinary lumpability expressed by Proposition 4.7). Afterwards, the procedure iteratively refines each of these classes: class \( C' \) remains unchanged if and only if, for any other class \( C'' \), all the states in \( C' \) have the same aggregated rate to reach \( C'' \).

<table>
<thead>
<tr>
<th>Table 2. Ordinary lumping procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin</td>
</tr>
<tr>
<td>repeat</td>
</tr>
<tr>
<td>( \Lambda_{old} := \Lambda );</td>
</tr>
<tr>
<td>( \Lambda := \emptyset );</td>
</tr>
<tr>
<td>foreach ( C \in \Lambda_{old} ) do begin</td>
</tr>
<tr>
<td>( \sum{| \lambda \mid s_p \xrightarrow{\lambda} \rightarrow_{E, \mathcal{M}<em>0} s' \land s' \in C'' |} = \sum{| \lambda \mid s_q \xrightarrow{\lambda} \rightarrow</em>{E, \mathcal{M}_0} s' \land s' \in C'' |} );</td>
</tr>
<tr>
<td>( \Lambda := \Lambda \cup \Lambda_C );</td>
</tr>
<tr>
<td>end until ( \Lambda = \Lambda_{old} );</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Once determined the ordinary lumping \( \Lambda_{E, \mathcal{M}_0} \) of \( \mathcal{M}_0[E] \), we finally are in a position of presenting the Markovian semantics of a given term.

**Definition 4.10** The Markovian semantics of a term \( E \in \mathcal{E} \) is the PLTS
\[
\mathcal{M}[E] = (S_{E, \mathcal{M}}, \mathbb{R}_+, \rightarrow_{E, \mathcal{M}}, P_{E, \mathcal{M}})
\]
where:
- \( S_{E, \mathcal{M}} = \Lambda_{E, \mathcal{M}_0} \);
• let us define for every \( C \in S_{E,\mathcal{M}} \)
\[
PM_{\mathcal{M}}(C) = \{ (\lambda, C') \mid s \xrightarrow{\lambda} E,\mathcal{M}_0, s' \wedge s' \in C' \}
\]
where \( s \) is fixed in \( C \); then \( C \xrightarrow{\lambda} E,\mathcal{M} \) is the least subset of \( S_{E,\mathcal{M}} \times \mathbb{R}_+ \times S_{E,\mathcal{M}} \) such that
\[
C \xrightarrow{\lambda} E,\mathcal{M} C'
\]
whenever \( (\lambda, C') \in \text{Melt}(PM_{\mathcal{M}}(C)) \);

• \( P_{E,\mathcal{M}} : S_{E,\mathcal{M}} \rightarrow \mathbb{R}_{[0,1]} \) such that, for every \( C \in S_{E,\mathcal{M}} \), \( P_{E,\mathcal{M}}(C) = \sum_{s \in C} P_{E,\mathcal{M}}(s) \).

Now we prove the correctness of the second phase of the algorithm.

**Theorem 4.11** Let \( E \in \mathcal{E} \).

(i) \( \Lambda_{E,\mathcal{M}_0} \) is an ordinary lumping of \( \mathcal{M}_0[E] \).

(ii) \( \sum_{C \in S_{E,\mathcal{M}}} P_{E,\mathcal{M}}(C) = 1 \).

(iii) If \( \mathcal{M}[E] \) has finitely many states, then also the second phase of the algorithm terminates and \( \mathcal{M}[E] \) has finitely many states and is unique.

**Proof** Let \( E \in \mathcal{E} \).

(i) Each partition \( \Lambda \) of \( S_{E,\mathcal{M}_0} \) constructed in the repeat-until cycle is such that for every \( C' \in \Lambda \), for every \( C'' \in \Lambda_{\text{id},\Lambda} \), and for every \( s_p, s_q \in C' \)
\[
\sum_{s \in \Lambda} \{ \lambda \mid s_p \xrightarrow{\lambda} E,\mathcal{M}_0, s \wedge s' \in C' \} = \sum_{s \in \Lambda} \{ \lambda \mid s_q \xrightarrow{\lambda} E,\mathcal{M}_0, s \wedge s' \in C' \}
\]
Since after the last iteration partition \( \Lambda \) is equal to partition \( \Lambda_{\text{id},\Lambda} \), it follows that \( \Lambda \) is an ordinary lumping of \( \mathcal{M}_0[E] \).

(ii) It immediately stems from Theorem 4.9(ii).

(iii) If \( \mathcal{M}[E] \) has finitely many states then, by virtue of Theorem 4.9(v), the first phase of the algorithm terminates and \( S_{E,\mathcal{M}_0} \) is finite. Since the second phase of the algorithm consists of repeatedly partitioning \( S_{E,\mathcal{M}_0} \), since this proceeding terminates at most when it reaches the partition whose sets contain each only one element in \( S_{E,\mathcal{M}_0} \), and since such a partition is reached after a number of repeat-until cycles bounded by \( |S_{E,\mathcal{M}_0}| \), also the second phase of the algorithm terminates. The previous argument furthermore allows us to deduce that \( \mathcal{M}[E] \) has finitely many states. The uniqueness of \( \mathcal{M}[E] \) stems from the fact that each partition must be minimum.

5 **Describing queueing systems with EMPA**

Before continuing with the presentation of the integrated approach of Figure 1, we wish to dwell upon EMPA in order to show its expressiveness and stress the advantages gained by exploiting syntactical compositionality in the modeling process. To see this, we consider some examples concerning queueing systems.

A *queueing system (QS)* [44] is a model largely used for performance evaluation purposes to represent a service center composed of a waiting queue and a given number of servers; they provide a certain service (following a given discipline) to the customers arriving at the service center.

Since the customer arrival process and the customer service process are described as stochastic processes, in Section 5.1 we show how to express with EMPA some frequently occurring probability distributions. Then, in Sections 5.2 – 5.7 we model various classes of QSs.

5.1 **Phase-type distributions**

Using EMPA, one can express actions having exponentially distributed durations as well as zero durations. From the stochastic point of view (hence for performance evaluation purposes), only exponential distributions are expressible in a straightforward manner. However, it is worth noting that with the interplay of exponentially timed actions and immediate actions, all the phase-type distributions are expressible by means of EMPA.
(a) Exponential distribution

(b) Hypoexponential distribution

(c) Hyperexponential distribution

(d) Coxian distribution

Figure 8. Phase-type distributions
A phase-type distribution [53] is a continuous distribution function describing the time to absorption in a finite-state HCTMC having exactly one absorbing state. Well known examples of phase-type distributions are the exponential distribution, the hypoexponential distribution, the hyperexponential distribution, and finally the Coxian distribution. All these distributions are characterized in terms of time to absorption in a finite-state HCTMC with an absorbing state as outlined in Figure 8. As an absorbing state can be modeled by term 0, the distributions above can be easily represented as follows:

- An exponential distribution with rate \( \lambda \in \mathbb{R}_+ \) can be modeled by means of term

\[
\text{Exp}_\lambda \equiv \langle a, \lambda \rangle > 0
\]

whose Markovian semantics is p-isomorphic to the HCTMC reported in Figure 8(a).

- An \( n \)-stage hypoexponential distribution with rates \( \lambda_i \in \mathbb{R}_+, 1 \leq i \leq n \), can be modeled by means of inductively defined terms

\[
\text{Hypexp}_{m_1, \ldots, \lambda_m} \equiv \langle a, \lambda_i \rangle \cdot \text{Hypexp}_{m_2, \ldots, \lambda_m}, \quad 2 \leq m \leq n;
\]

\[
\text{Hypexp}_{1, \lambda} \equiv \text{Exp}_\lambda
\]

whose Markovian semantics is p-isomorphic to the HCTMC reported in Figure 8(b).

- An \( n \)-stage hyperexponential distribution with rates \( \lambda_i \in \mathbb{R}_+, 1 \leq i \leq n \), and branching probabilities \( p_i \in [0, 1], 1 \leq i \leq n \), where \( \sum_{i=1}^{n} p_i = 1 \), can be modeled by means of the inductively defined terms

\[
\text{Hyperexp}_{n, \lambda_1, \ldots, \lambda_n; p_1, \ldots, p_n} \equiv \text{Hypexp}_{n, \lambda_1, \ldots, \lambda_n; p_1, \ldots, p_n};
\]

\[
\text{Hypexp}_{1, \lambda} \equiv \text{Exp}_\lambda
\]

whose Markovian semantics is p-isomorphic to the HCTMC reported in Figure 8(c).

- An \( n \)-stage Coxian distribution with rates \( \lambda_i \in \mathbb{R}_+, 1 \leq i \leq n \), and branching probabilities \( p_i, q_i \in [0, 1] \) where \( p_i + q_i = 1 \), \( 1 \leq i \leq n - 1 \), can be modeled by means of the inductively defined terms

\[
\text{Cox}_{m_1, \lambda_1, \ldots, \lambda_m; p_1, \ldots, p_{m-1}, q_1, \ldots, q_{m-1}} \equiv \langle a, \lambda_i \rangle \cdot \langle a, \infty_{1, q_1} \rangle > 0 + \langle a, \infty_{1, p_1} \rangle ;
\]

\[
\text{Cox}_{1, \lambda} \equiv \text{Exp}_\lambda
\]

whose Markovian semantics is p-isomorphic to the HCTMC reported in Figure 8(d).

The capability of expressing phase-type distributions is quite important since many frequently occurring distribution functions can be approximated by series-parallel combinations of exponential distributions [28]. However, it must be noticed that in EMPA phase-type distributions cannot be described in a direct manner because they require the interplay of several exponentially timed and immediate actions. As a consequence, they have to be used carefully due to the lack of atomicity. For example, if we consider term

\[
\text{Exp}_\lambda + \text{Hyperexp}_{2, \lambda_1, \lambda_2; p_1, p_2}
\]

then we realize that the right-hand side term takes precedence over the left-hand side term, so the whole term cannot be used to express the choice between an activity whose duration is exponentially distributed and another activity whose duration is hyperexponentially distributed. To overcome this drawback, the system designer has to be enabled to describe directly each kind of distribution: this problem will be addressed in Section 9(v).

5.2 Queueing systems M/M/1/q

A QS \( M/M/n/q/m \) with arrival rate \( \lambda \) and service rate \( \mu \) [44] is a service center defined as follows:

1. The customer interarrival time is exponentially distributed with rate \( \lambda \).
2. The customer service time is exponentially distributed with rate \( \mu \).
3. There are \( n \) independent servers.
4. There is a FIFO queue with \( q - n \) seats.
5. There are \( m \) independent customers.
In the following examples we concentrate on QSs $M/M/1/q$: the absence of the value of the fifth parameter means that the number of customers is unbounded. How can we model a QS $M/M/1/q$ with arrival rate $\lambda$ and service rate $\mu$? Let $a$ be the action type “a customer arrives at the queue of the service center”, $d$ be the action type “a customer is delivered by the queue to the server”, and $s$ be the action type “a customer is served by the server”. Then the QS under consideration can be modeled with EMPA as follows:

- $\text{System}_{M/M/1/q} \triangleq \text{Arrivals} \parallel_{\{a\}} (\text{Queue}_0 \parallel_{\{d\}} \text{Server})$
  - $\text{Arrivals} \triangleq <a, \lambda>, \text{Arrivals}$
  - $\text{Queue}_0 \triangleq <a, *> , \text{Queue}_1$
  - $\text{Queue}_h \triangleq <a, *> , \text{Queue}_{h+1}$
  - $\text{Queue}_{q-1} \triangleq <d, *> , \text{Queue}_{q-2}$
  - $\text{Server} \triangleq <d, \infty, 1>, <s, \mu>, \text{Server}$

It is worth noting that we have described the whole system as the composition of the arrival process with the composition of the queue and the server (using action types $a$ and $d$ as interfaces among components), and that then we have separately modeled the arrival process, the queue and the server. Since the queue is independent of both the arrival rate and the service rate, passive actions have been exploited to represent it. As a consequence, if we want to modify the description by changing the arrival rate or the service rate, only component $\text{Arrivals}$ or $\text{Server}$ needs to be modified while component $\text{Queue}$ is not affected. Additionally, the delivery of a customer to the server can be neglected from the performance point of view: this is achieved by means of the immediate action occurring in component $\text{Server}$.

The operational interleaving semantics $I[\text{System}_{M/M/1/q}]$ is given by the LTS reported in Figure 9(a), where for $0 \leq h \leq q-1$ the shorthand $A Q_h S$ stands for $\text{Arrivals} \parallel_{\{a\}} (\text{Queue}_h \parallel_{\{d\}} \text{Server})$ and the shorthand $A Q_{q-1} S'$ stands for $\text{Arrivals} \parallel_{\{a\}} (\text{Queue}_{q-1} \parallel_{\{d\}} <s, \mu>, \text{Server})$.

The Markovian semantics $M[\text{System}_{M/M/1/q}]$ is given by the HCTMC reported in Figure 9(b), and it is p-isomorphic to the HCTMC underlying a queueing system $M/M/1/q$ [44]. In [17] we proved that this result holds for each QS of the class $M/M$, thus supporting our claim that we have captured the correct Markovian semantics.

The QS at hand can be made complex at will. Nevertheless, it will always be relatively easy to model its variants thanks to syntactical compositionality. For instance, in the next three sections we show the algebraic descriptions of a QS $M/M/1/q$ whose service rate depends upon the number of customers in the queue, a QS $M/M/1/q$ serving customers requiring different service
rates, and a QS \( M/M/1/q \) serving customers having different priorities associated with them: all of these representations are obtained by changing locally the components of \( System_{M/M/1/q} \). Moreover, in the last two sections we show how to combine the descriptions of QSs \( M/M/1/q \) in order to model QSs with forks and joins as well as queueing networks.

5.3 Queueing systems \( M/M/1/q \) with scalable service rate

Assume that a QS \( M/M/1/q \) with arrival rate \( \lambda \) provides service at a speed depending on the number of customers in the queue. Let us denote by \( \mu \) the basic service rate, and by \( sf : N_+ \rightarrow \mathbb{R}_+ \) the function describing the scaling factor. This QS can be modeled as follows:

\[
SSRSystem_{M/M/1/q} \triangleq \text{Arrivals} \# (\text{Queue}_0 \# (d_{\leq h} \leq q-1) \text{ Server});
\]

- \( \text{Arrivals} \triangleq < a, \lambda >. \text{Arrivals} \);
- \( \text{Queue}_0 \triangleq < a, \ast >. \text{Queue}_1 \),
  \( \text{Queue}_h \triangleq < a, \ast >. \text{Queue}_{h+1} + < d_h, \ast >. \text{Queue}_{h-1}, 0 < h < q-1, \)
  \( \text{Queue}_{q-1} \triangleq < d_{q-1}, \ast >. \text{Queue}_0; \)
- \( \text{Server} \triangleq < d_1, \infty >. \text{Server}_1 + \ast; + < d_{q-1}, \infty >. \text{Server}_{q-1}; \)
  * \( \text{Server}_h \triangleq < s, sf(h) \cdot \mu >. \text{Server}_1, 1 \leq h \leq q-1. \)

It is worth noting that the structure of \( SSRSystem_{M/M/1/q} \) is the same as that of \( System_{M/M/1/q} \). Only component \( \text{Server} \) has been significantly modified in order to be able to provide service at a rate depending on the queue length.

5.4 Queueing systems \( M/M/1/q \) with different service rates

Assume that a QS \( M/M/1/q \) with arrival rate \( \lambda \) must serve two different types of customers. Red customers require a service rate \( \mu_r \) whereas black customers require a service rate \( \mu_b \). Such a situation can arise, e.g., in a computer system where the central unit can be viewed as the server and the various devices can be viewed as the customers. In this case, service requests can arrive from several different points and may require different service rates; the type associated with each request singles out the routing of the request itself. This QS can be modeled as follows:

\[
DSSSystem_{M/M/1/q} \triangleq \text{Arrivals} \# (\text{Queue}_r \# (d_{\leq d}) \text{ Server});
\]

- \( \text{Arrivals} \triangleq < a, \lambda >. \text{Arrivals} + < a_b, \lambda >. \text{Arrivals} \);
- \( \text{Queue}_r \triangleq < a_r, \ast >. \text{Queue}_r + < a_b, \ast >. \text{Queue}_b, \)
  \( \text{Queue}_{rw} \triangleq < a_r, \ast >. \text{Queue}_{rw} + < a_b, \ast >. \text{Queue}_{rw} + < d_r, \ast >. \text{Queue}_{rw}, 0 \leq |w| < q-2, \)
  \( \text{Queue}_{bw} \triangleq < a_r, \ast >. \text{Queue}_{bw} + < a_b, \ast >. \text{Queue}_{bw} + < d_b, \ast >. \text{Queue}_{bw}, 0 \leq |w| < q-2, \)
  \( \text{Queue}_{w} \triangleq < d_r, \ast >. \text{Queue}_{w}, |w| = q-2; \)
- \( \text{Server} \triangleq < d_r, \infty >. < s_r, \mu_r >. \text{Server}_r + < d_b, \infty >. < s_b, \mu_b >. \text{Server}_b. \)

Again, note that the structure of \( DSSSystem_{M/M/1/q} \) is the same as that of \( System_{M/M/1/q} \). Only the components have been locally modified in order to be able to treat the two types of customers.

5.5 Queueing systems \( M/M/1/q \) with different priorities

Assume that a QS \( M/M/1/q \) with arrival rate \( \lambda \) and service rate \( \mu \) must serve two different types of customers requiring the same service rate. Red customers are assigned a priority level \( r > b \), where \( b \) is the priority level assigned to black customers. There are two cases.

In the first case, we assume that the priority mechanism only affects the queueing discipline, i.e. we assume that possible preemption on the customer being served cannot be exercised. This QS can be modeled as follows:

\[
PSSystem_{M/M/1/q} \triangleq \text{Arrivals} \# (\text{Queue}_{0,0} \# (d_{\leq d}) \text{ Server});
\]

- \( \text{Arrivals} \triangleq < a, \lambda >. \text{Arrivals} + < a_b, \lambda >. \text{Arrivals} \);
In this section we want to model a QS with a fork and a join that is composed of two QSs $sr_i$ and $sr_j$. The service request of each customer arrived at the QS is divided into two immediate actions with different priority levels occurring in $Server$. The two subrequests $sr_i$ and $sr_j$ are delivered to the join; here they are merged together in $Server$. Note that the precedence of red customers over black ones has been enforced by means of the two immediate actions with different priority levels occurring in $Server$.

Note that the precedence of red customers over black ones has been enforced by means of the two immediate actions with different priority levels occurring in $Server$.

In the second case, we assume that preemption on a black customer being served can be exercised by red customers. This QS can be modeled as follows ($Arrivals$ and $Queue_{i,j}$ are omitted since they stay the same):

- $PPSystem_{M/M/1/q} \triangleq Arrivals ||\{sr_i, sr_j\} (Queue_{0,0} ||| {d_r, d_i}) Server$;
- $Server \triangleq <d_r, \infty, d_i>.Server + <d_i, \infty, d_r>.Server$;
- $Server_r \triangleq <s, \mu>.Server$;
- $Server_b \triangleq <s, \mu>.Server + <d_r, \infty, d_i>.<s, \mu>.Server$.

Note that, due to the memoryless property of the exponential distribution, there is no difference between the preemptive-restart policy (i.e., the preempted customer restarts from the beginning) and the preemptive-resume policy (i.e., the preempted customer resumes from the point at which it has been interrupted).

5.6 Queueing systems with forks and joins

In this section we want to model a QS with a fork and a join that is composed of two QSs $M/M/1/q$ operating in parallel. The service request $sr$ of each customer arrived at the QS is divided into two subrequests $sr_1$ and $sr_2$ by the fork, that are sent to the two QSs $M/M/1/q$. After being served, the two subrequests $sr_1$ and $sr_2$ are delivered to the join; here they are merged together in $\tau'$ and the whole request is considered fulfilled. Once denoted by $Id_S$ the identity function over set $S$, this QS can be modeled as follows:

- $FJSys \triangleq In \|\{\tau\} Fork \|\{sr_i, sr_j\} Center \|\{sr_i', sr_j\} Join \|\{\tau'\} Out$;
- $In \triangleq \tau, \lambda, In$;
- $Fork \triangleq (F[\varphi_1] ||\{\tau\} (F[\varphi_2]));$
  * $F \triangleq \tau, *>, <sr_i, \infty, d_i>.F$;
  * $\varphi_i = \{(sr_i, sr_i) \cup Id_AType_{-sr_i}, i \in \{1, 2\}$;
- $Center \triangleq (C[\varphi_i]) ||| (C[\varphi_i])$;
  * $C \triangleq Queue_0 ||| Id_Server$;
    * $Queue_0 \triangleq <sr, *>.Queue_1$;
    * $Queue_1 \triangleq <sr, *>.Queue_1 + <d_i, *>.Queue_{-1}, 0 < h < q - 1$;
    * $Queue_{-1} \triangleq <d_i, *>.Queue_{-2}$;
    * $Server \triangleq <d_r, \infty, d_i>.<s, \mu>.<sr_i, \infty, d_i>.Server$;
    * $\varphi_i = \{(sr_i, sr_i, sr_i') \cup Id_AType_{-sr_i}, i \in \{1, 2\}$;
- $Join \triangleq (J_0[\varphi_i']) \|\{\tau'\} (J_0[\varphi_i''])$;
  * $J_0 \triangleq <sr_i', *>.J_1$;
  * $J_1 \triangleq <sr_i', *>.J_{h+1} + <\tau', *>.J_{h-1}, h > 0$;
  * $\varphi_i'' = \{(sr_i', sr_i') \cup Id_AType_{-sr_i'}, i \in \{1, 2\}$.
In Sections 2–4 we have defined the syntax of EMPA and the integrated operational interleaving semantics of its terms together with the two projections resulting in the functional and the Markovian semantics: in other words, we have developed the means whereby it is possible to implement the first phase of the integrated approach of Figure 1. Then in Section 5 we have shown the expressiveness of EMPA by describing QSs, and we have stressed its syntactical compositionality. Before moving to the second phase of the integrated approach, we want to equip EMPA with a notion of integrated equivalence in order to gain (i) the capability of performing an integrated analysis without building projected semantic models, and (ii) semantic compositionality, i.e., the possibility of studying separately the various system components. Note that the integrated equivalence allows for a qualitative analysis, because it tells us whether two terms represent two concurrent systems with the same functional and performance properties regardless of their values. In order to know whether a functional property holds, or the value of a performance measure, we have to study the projected semantic models of (the simplest) one of the two terms.

This section is organized as follows. In Section 6.1 we start by defining two projected equivalences based on the two projected semantic models, and then we introduce a notion of integrated equivalence, called strong extended Markovian bisimulation equivalence, defined over the integrated semantic model. In Section 6.2 we address the semantic compositionality issue by showing that the strong extended Markovian bisimulation equivalence is a congruence. In Section 6.3 we investigate the relationship between the strong extended Markovian bisimulation equivalence and other notions of semantic compositionality.

Note that the availability of the functional relabeling operator has allowed us to obtain more compact algebraic representations of components having the same structure but differing for some action types only.

5.7 Queueing networks

A queueing network (QN) [42] is composed of a set of QSs linked to each other. In general, every QS can receive customers from the outside (external sources), from the other QSs in the network, and from itself (feedback paths). The case of open QNs, where interactions with the outside are allowed, is particularly interesting because this kind of QNs can be used to describe store-and-forward packet-switched communication networks [66].

Let us focus our attention on an open QN composed of n QSs $M/M/1/q$ with service rates $\mu_1, \mu_2, \ldots, \mu_n$, respectively. Assume that there are n external sources of customers with rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively. Let us denote by $r_{i,j}$ and $p_{i,j}$ the routing action type and the routing probability, respectively, from QS $i$ to QS $j$ or the outside ($j = n + 1$). This QN can be modeled as follows:

- **QN** $\stackrel{\Delta}{=} QS_1 || R_2 || QS_2 || R_3 || \cdots || R_n || QS_n$;
- **$QS_i$** $\stackrel{\Delta}{=} \text{Arrivals}_i || (a_i) (\text{Queue}_{i,0} || (d_i, r_{i,i}) \text{Server}_i), \ 1 \leq i \leq n$;
  - **Arrivals** $\stackrel{\Delta}{=} <a_i, \lambda_i> \text{Arrivals}_i$;
  - **Queue$_i,0$** $\stackrel{\Delta}{=} <a_i, \lambda_i> \text{Queue}_{i,0} + <r_{i,1}, \lambda_i> \text{Queue}_{i,1} + \cdots + <r_{n,1}, \lambda_i> \text{Queue}_{i,1}$;
  - **Queue$_i,h$** $\stackrel{\Delta}{=} <a_i, \lambda_i> \text{Queue}_{i,h} + <r_{i,1}, \lambda_i> \text{Queue}_{i,h+1} + \cdots + <r_{n,1}, \lambda_i> \text{Queue}_{i,h+1}$;
  - **Queue$_i,q-1$** $\stackrel{\Delta}{=} <d_i, \lambda_i> \text{Queue}_{i,q-2}$;
- **Server** $\stackrel{\Delta}{=} <d_i, \mu_i> \text{Server}_i$;
- **Router** $\stackrel{\Delta}{=} <r_{i,j}, \mu_{i,j}> \text{Router}_i + \cdots + <r_{i,n+1}, \mu_{i,n+1}> \text{Server}_i$;
- $R_j = \{r_{i,j} | 1 \leq i < j\}, \ 2 \leq j \leq n$.

Observe that the description of the QN has been obtained by simply composing the descriptions of the single QSs.

6 A notion of integrated equivalence for EMPA

In Sections 2–4 we have defined the syntax of EMPA and the integrated operational interleaving semantics of its terms together with the two projections resulting in the functional and the Markovian semantics: in other words, we have developed the means whereby it is possible to implement the first phase of the integrated approach of Figure 1. Then in Section 5 we have shown the expressiveness of EMPA by describing QSs, and we have stressed its syntactical compositionality. Before moving to the second phase of the integrated approach, we want to equip EMPA with a notion of integrated equivalence in order to gain (i) the capability of performing an integrated analysis without building projected semantic models, and (ii) semantic compositionality, i.e., the possibility of studying separately the various system components. Note that the integrated equivalence allows for a qualitative analysis, because it tells us whether two terms represent two concurrent systems with the same functional and performance properties regardless of their values. In order to know whether a functional property holds, or the value of a performance measure, we have to study the projected semantic models of (the simplest) one of the two terms.

In Section 6.1 we start by defining two projected equivalences based on the two projected semantic models, and then we introduce a notion of integrated equivalence, called strong extended Markovian bisimulation equivalence, defined over the integrated semantic model. In Section 6.2 we address the semantic compositionality issue by showing that the strong extended Markovian bisimulation equivalence is a congruence. In Section 6.3 we investigate the relationship between the strong extended Markovian bisimulation equivalence.
6.1 Strong extended Markovian bisimulation equivalence

The purpose of the notion of equivalence is to relate terms describing systems that are indistinguishable from the point of view of an external observer. To this aim, a well-suited notion is bisimilarity [56, 48]: two terms are bisimilar if they are able to simulate each other. Using this approach, it is straightforward to define two projected equivalences based on the two projected semantic models.

**Definition 6.1** Let $E_1, E_2 \in \mathcal{G}$. We say that $E_1$ is functionally equivalent to $E_2$, written $E_1 \simF E_2$, if and only if $\mathcal{F}[E_1]$ is bisimilar to $\mathcal{F}[E_2]$.

**Definition 6.2** Let $E_1, E_2 \in \mathcal{E}$. We say that $E_1$ is performance equivalent to $E_2$, written $E_1 \simP E_2$, if and only if $\mathcal{M}[E_1]$ is $p$-bisimilar to $\mathcal{M}[E_2]$.

A natural candidate notion of equivalence may be $\simFP = \simF \cap \simP$. However, the examples below show that $\simFP$ is not useful as it is not a congruence.

**Example 6.3** Consider terms

\[
\begin{align*}
E_1 & \equiv <a, \lambda> \cdot 0 + <b, \mu> \cdot 0 \\
E_2 & \equiv <a, \mu> \cdot 0 + <b, \lambda> \cdot 0
\end{align*}
\]

It turns out that $E_1 \simFP E_2$ but $E_1 \not\simF E_2$ because the left-hand side term has exit rate $\lambda$ while the right-hand side term has exit rate $\mu$, thereby violating the necessary condition for $p$-bisimilarity expressed by Proposition 4.3. Note that the action with type $a$ of $E_1$ has execution probability $\lambda / (\lambda + \mu)$, while the action with type $a$ of $E_2$ has execution probability $\mu / (\lambda + \mu)$, and this is not detected by $\simFP$.

**Example 6.4** Consider terms

\[
\begin{align*}
E_1 & \equiv <a, \infty_{1,1}> \cdot 0 \\
E_2 & \equiv <a, \infty_{2,1}> \cdot 0
\end{align*}
\]

It turns out that $E_1 \simFP E_2$ but $E_1 + <b, \infty_{1,1}> \cdot 0 \not\simF E_2 + <b, \infty_{1,1}> \cdot 0$ because the left-hand side term can execute an action with type $b$ while the right-hand side term cannot.

**Example 6.5** Consider terms

\[
\begin{align*}
E_1 & \equiv <a, \infty_{1,1}> \cdot 0 \\
E_2 & \equiv <a, \infty_{1,1}> \cdot 0
\end{align*}
\]

It turns out that $E_1 \simFP E_2$ but $E_1 + <b, \infty_{1,1}, <b, \lambda> \cdot 0 \not\simP E_2 + <b, \infty_{1,1}, <b, \lambda> \cdot 0$ because state $<b, \lambda> \cdot 0$ has initial state probability $1/2$ in the Markovian semantics of the left-hand side term, $1/3$ in the Markovian semantics of the right-hand side term.

The examples above show that $\simFP$ is unable to keep track of the link between the functional part and the performance part of the actions. This means that to achieve semantic compositionality, it is necessary to define an equivalence based on the integrated semantic model. Incidentally, this is even convenient with respect to $\simFP$, since it avoids the need of building the two projected semantic models and checking them for bisimilarity and $p$-bisimilarity, respectively.

In order to define an integrated equivalence $\simI$ in the bisimulation style, we can follow the guideline below:

- Active actions should be treated by following the notion of probabilistic bisimulation proposed in [45], which consists of requiring a bisimulation to be an equivalence relation
such that two bisimilar terms have the same aggregated probability to reach the same equivalence class by executing actions of the same type and priority level.

- For exponentially timed actions, the notion of probabilistic bisimulation must be refined by requiring additionally that two bisimilar terms have identically distributed sojourn times. For example, if we consider terms

\[
E_1 \equiv <a, \lambda>.E + <a, \mu>.G
\]
\[
E_2 \equiv <a, 2 \cdot \lambda>.E + <a, 2 \cdot \mu>.G
\]

then both transitions labeled with \(a, \lambda\) and \(a, 2 \cdot \lambda\) have execution probability \(\lambda/(\lambda + \mu)\), and both transitions labeled with \(a, \mu\) and \(a, 2 \cdot \mu\) have execution probability \(\mu/(\lambda + \mu)\), but the average sojourn time of \(E_1\) is twice the average sojourn time of \(E_2\). Due to the race policy, requiring that two bisimilar terms have identically distributed sojourn times and the same aggregated probability to reach the same equivalence class by executing exponentially timed actions of the same type, amounts to requiring that two bisimilar terms have the same aggregated rate to reach the same equivalence class by executing exponentially timed actions of the same type. For example, it must hold that

\[
<a, \lambda>.F + <a, \mu>.F \sim_1 <a, \lambda + \mu>.F
\]

This coincides with the notion of Markovian bisimulation proposed in [36, 39, 25].

- For immediate actions, the notion of probabilistic bisimulation must be restated in terms of weights. As a consequence, two bisimilar terms are required to have the same aggregated weight to reach the same equivalence class by executing immediate actions of the same type and priority level. For example, it must hold that

\[
<a, \lambda_1.w>.F + <a, \lambda_2.w>.F \sim_1 <a, \lambda_1 + \lambda_2.w>.F
\]

This coincides with the notion of direct bisimulation proposed in [67].

- Passive actions should be treated by following the classical notion of bisimulation (see, e.g., [48]). Thus, bisimilar terms are required to have the same passive actions reaching the same equivalence class, regardless of the actual number of these passive actions. For example, it must hold that

\[
<a, *.>.F + <a, *.>.F \sim_1 <a, *.>.F
\]

Before introducing the integrated equivalence, we must have a look at action priority levels. It might seem useful to be able to write equations like

\[
<a, \lambda>.E + <b, \lambda>.F \sim_1 <b, \lambda>.F
\]
\[
<c, \lambda>.E + <d, \lambda>.F \sim_1 <d, \lambda>.F \quad \text{if} \quad l' > l
\]

The problem is that the applicability of such equations depends on the context: e.g., terms

\[
E_1 \equiv (a, \lambda).E + b, \lambda\rangle >.F \parallel [i] \langle 0
\]
\[
E_2 \equiv (b, \lambda).F \parallel [i] \langle 0
\]

are not equivalent because \(E_1\) can execute one action while \(E_2\) cannot execute actions at all. To solve the problem, we follow the proposal of [10] by introducing a priority interpretation operator \(\Theta\). We then consider the language \(L_{\Theta}\) generated by the following syntax

\[
E ::= 0 | \langle a, \lambda \rangle.E | E/L | E \backslash H | E[\varphi] | \Theta(E) | E + E | E \parallel_s E | A
\]

whose semantic rules are those reported in Table 1 except that the rule in the first part is replaced by

\[
\frac{E \frac{\alpha, \lambda}{\alpha, \lambda} \rightarrow E'}{(\langle a, \lambda \rangle, E) \in \text{Melt}(PM(E))}
\]

and the following rule for \(\Theta\) is introduced in the second part

\[
PM(\Theta(E)) = S\text{dect}(PM(E))
\]

It is easily seen that EMPA coincides with the set of terms \(\{\Theta(E) | E \in L\}\).

To keep the definition of integrated equivalence as simple as possible, we formalize the key concept of conditional exit rate in a uniform way for all kinds of action by means of partial function

\[
E\text{Rate} : (G_\Theta \times AT \times PL) \rightarrow A\text{Rate}
\]

defined by

\[
E\text{Rate}(E, a, l, C) = \text{Min}(\lambda) \left[ E \frac{\alpha, \lambda}{\alpha, \lambda} \rightarrow E' \wedge PL(\langle a, \lambda \rangle) = l \wedge E' \in C \right]
\]

8. For convenience, we pose \(\text{Min}(\emptyset) = \bot, \lambda \text{Min} \bot = \lambda, \text{Split}(\bot, p) = \bot\).
Now we are in a position of introducing the integrated equivalence, based on the following notion of strong extended Markovian bisimulation.

**Definition 6.6** An equivalence relation \( B \subseteq G_\Theta \times G_\Theta \) is a strong extended Markovian bisimulation (strong EMB) if and only if, whenever \( (E_1, E_2) \in B \), then for all \( a \in \text{AType}, l \in \text{PLSet} \) and \( C \in G_\Theta / B \)

\[
\text{ERate}(E_1, a, l, C) = \text{ERate}(E_2, a, l, C)
\]

In this case we say that \( E_1 \) and \( E_2 \) are strongly extended-Markovian bisimilar (strongly EMB).

As an example, the identity relation \( 1_{G_\Theta} \) over \( G_\Theta \) is a strong EMB, and it is contained in any strong EMB due to reflexivity. We now prove that the largest strong EMB is the union of all the strong EMBs, and we define the integrated equivalence as the largest strong EMB.

**Lemma 6.7** Let \( \{B_i \mid i \in I\} \) be a family of strong EMBs. Then \( B = (\cup_{i \in I} B_i)^+ \) is a strong EMB.

**Proof** Once observed that \( B \) is an equivalence relation because it is the transitive closure of the union of equivalence relations, assume that \( (E_1, E_2) \in B \). Since \( B = \cup_{i \in \mathbb{N}} B^{(n)} \) where \( B^{(n)} = (\cup_{i \in I} B_i)^+ \), we have \( (E_1, E_2) \in B^{(n)} \) for some \( n \in \mathbb{N} \). The result follows by proving by induction on \( n \in \mathbb{N} \) that, whenever \( (E_1, E_2) \in B^{(n)} \), then \( \text{ERate}(E_1, a, l, C) = \text{ERate}(E_2, a, l, C) \) for all \( a \in \text{AType}, l \in \text{PLSet}, C \in G_\Theta / B \).

- If \( n = 1 \), then \( (E_1, E_2) \in B_i \) for some \( i \in I \). Let \( G_\Theta / B_i = \{C_{i,j} \mid j \in J_i\} \). Since \( (E_1, E_2) \in B_i \), implies \( (E_1, E_2) \in B_i \), we have that for each \( C_{i,j} \in G_\Theta / B_i \), there exists \( C \in G_\Theta / B \) such that \( C_{i,j} \subseteq C \), so each equivalence class of \( B \) can be written as the union of a set of equivalence classes of \( B_i \). As a consequence, for all \( a \in \text{AType}, l \in \text{PLSet} \) and \( C \in G_\Theta / B \), if \( C = \cup_{i \in J'} C_{i,j} \) where \( J' \subseteq J \), then \( \text{ERate}(E_1, a, l, C) = \text{ERate}(E_2, a, l, C) \).

- Let \( n > 1 \). From \( (E_1, E_2) \in B^{(n)} \) we derive that there exists \( F \in G_\Theta \) such that \( (E_1, F) \in B^{(n-1)} \) and \( (F, E_2) \in B_i \) for some \( i \in I \). Thus for all \( a \in \text{AType}, l \in \text{PLSet} \) and \( C \in G_\Theta / B \), it turns out \( \text{ERate}(E_1, a, l, C) = \text{ERate}(F, a, l, C) \) by the induction hypothesis, and \( \text{ERate}(F, a, l, C) = \text{ERate}(E_2, a, l, C) \) by applying the same argument as the previous point.

**Proposition 6.8** Let \( \sim_{\text{EMB}} \) be the union of all the strong EMBs. Then \( \sim_{\text{EMB}} \) is the largest strong EMB.

**Proof** By definition, \( \sim_{\text{EMB}} \) contains the largest strong EMB, so \( \sim_{\text{EMB}} \) is at least as large as the largest strong EMB. It remains to prove that \( \sim_{\text{EMB}} \) is a strong EMB. This is achieved by applying Lemma 6.7 to the fact that \( \sim_{\text{EMB}} = \sim_{\text{EMB}}^+ \). In fact, \( \sim_{\text{EMB}} \subseteq \sim_{\text{EMB}}^+ \) trivially holds, and \( \sim_{\text{EMB}}^+ \subseteq \sim_{\text{EMB}} \) is due to the fact that \( \sim_{\text{EMB}}^+ \) is the largest strong EMB (by virtue of Lemma 6.7) and that \( \sim_{\text{EMB}} \) contains all the strong EMBs.

**Definition 6.9** We call \( \sim_{\text{EMB}} \) the strong extended Markovian bisimulation equivalence (strong EMBE), and we say that \( E_1, E_2 \in G_\Theta \) are strongly extended-Markovian bisimulation equivalent (strongly EMBE) if and only if \( E_1 \sim_{\text{EMB}} E_2 \).

In other words, two terms \( E_1, E_2 \in G_\Theta \) are strongly EMB if and only if they are strongly EMB, i.e. there exists a strong EMB containing the pair \((E_1, E_2)\).

We conclude the section by exhibiting two necessary conditions and one sufficient condition in order for two terms to be strongly EMBE. The necessary conditions are based on conditional exit rates independent of equivalence classes, so they are easily checkable.

**Proposition 6.10** Let \( E_1, E_2 \in G_\Theta \). If \( E_1 \sim_{\text{EMB}} E_2 \), then

1. for all \( l \in \text{PLSet} \), 
\[
\min \{\text{ERate}(E_1, a, l, \{E_2\}) \mid a \in \text{AType} \land E \in G_\Theta \} = \min \{\text{ERate}(E_2, a, l, \{E_1\}) \mid a \in \text{AType} \land E \in G_\Theta \}
\]
The first necessary condition guarantees that the states associated with strongly EMBE terms have identically distributed sojourn times, if tangible, or identical total weights, if vanishing. The second necessary condition will be used in Section 6.3.

The sufficient condition is based on the notion of strong EMB up to $\sim_{EMB}$.

**Definition 6.11** An equivalence relation $B \subseteq G_\emptyset \times G_\emptyset$ is a strong EMB up to $\sim_{EMB}$ if and only if, whenever $(E_1, E_2) \in B$, then for all $a \in A_{Type}, l \in PLSet$ and $C \in G_\emptyset / (\sim_{EMB}) B \sim_{EMB}$

$$ERate(E_1, a, l, C) = ERate(E_2, a, l, C)$$

**Proposition 6.12** Let $B \subseteq G_\emptyset \times G_\emptyset$. If $B$ is a strong EMB up to $\sim_{EMB}$, then $B \subseteq \sim_{EMB}$.

**Proof** Given $B \subseteq G_\emptyset \times G_\emptyset$, we first prove that if $B$ is a strong EMB up to $\sim_{EMB}$, then $\sim_{EMB} B \sim_{EMB}$ is a strong EMB. Let $(E_1, E_2) \in \sim_{EMB} B \sim_{EMB}$, i.e. $E_1 \sim_{EMB} E_i E_i B \sim_{EMB} E_2$, and $a \in A_{Type}, l \in PLSet$ and $C \in G_\emptyset / (\sim_{EMB}) B \sim_{EMB})$. Then $ERate(E_1, a, l, C) = ERate(E_2, a, l, C)$. Since $\sim_{EMB} \subseteq \sim_{EMB} B \sim_{EMB}$, $C$ is the union of some equivalence classes with respect to $\sim_{EMB}$. As a consequence, for $j \in \{1, 2\}$ we have $ERate(E_j, a, l, C) = ERate(E_1, a, l, C)$, hence the result follows.

To complete the proof, we observe that $B \subseteq \sim_{EMB} B \sim_{EMB} \because id_{G_\emptyset} \subseteq \sim_{EMB}$, and $\sim_{EMB} B \sim_{EMB} \subseteq \sim_{EMB}$ because $\sim_{EMB} B \sim_{EMB}$ is a strong EMB. 

The notion of strong EMB up to $\sim_{EMB}$ is helpful to avoid redundancy in strong EMBS: for example, if $B$ is a strong EMB and $(E_1 || s E_2, E_3) \in B$, then also $(E_2 || s E_1, E_3) \in B$ although it may be retrieved from the fact that $E_2 || s E_1 \sim_{EMB} E_1 || s E_2$. The sufficient condition above states that in order for two terms $E_1, E_2 \in G_\emptyset$ to be strongly EMB, it suffices to find out a strong EMB up to $\sim_{EMB}$ containing the pair $(E_1, E_2)$. The notion of strong EMB up to $\sim_{EMB}$ will be used in Sections 6.2 and 6.4.

### 6.2 Congruence property

In this section we show that the strong EMB enriches EMPA with semantic compositionality. This stems from the congruence property of the strong EMBE, i.e. from the fact that the strong EMBE is preserved by all the operators.

**Theorem 6.13** Let $E_1, E_2 \in G_\emptyset$ be such that $E_1 \sim_{EMB} E_2$.

(i) For every $<a, \lambda> \in Act$, $<a, \lambda>.E_1 \sim_{EMB} <a, \lambda>.E_2$.

(ii) For every $L \subseteq A_{Type} - \{\tau\}$, $E_1 \downarrow L \sim_{EMB} E_2 \downarrow L$.

(iii) For every $H \subseteq A_{Type}$, $E_1 \downarrow H \sim_{EMB} E_2 \downarrow H$.

(iv) For every $\varphi \in Relab$, $E_1[\varphi] \sim_{EMB} E_2[\varphi].$

(v) $\Theta(E_1) \sim_{EMB} \Theta(E_2)$.

(vi) For every $G \subseteq G_\emptyset$, $E_1 + F \sim_{EMB} E_2 + F$.

(vii) For every $F \subseteq G_\emptyset$ and $S \subseteq A_{Type} - \{\tau\}$, $E_1 || F \sim_{EMB} E_2 || F$.

**Proof** Let $E_1, E_2 \in G_\emptyset$ be such that $E_1 \sim_{EMB} E_2$.

(i) Let $B \subseteq G_\emptyset \times G_\emptyset$ be a strong EMB such that $(E_1, E_2) \in B$. Given $<a, \lambda> \in Act$, we prove that $B' = (B \cup \{(<a, \lambda>.E_1, <a, \lambda>.E_2), (<a, \lambda>.E_2, <a, \lambda>.E_1)\})^+$ is a strong EMB. Observed that $B'$ is an equivalence relation, we have two cases.

- If $(<a, \lambda>.E_1, <a, \lambda>.E_2) \in B$, then $B' = B$ and the result trivially follows.
Given last subcase the result follows from the fact that for $\lambda \in \text{EMB}$ is similar to the one developed in $F$. Observed that $L \subseteq \text{AType} - \{\tau\}$, we prove that

$$B' = B \cup l \& E_0$$

is a strong EMB. Observed that $B'$ is an equivalence relation, and that either each of the terms of an equivalence class has $\sim L$ as topmost operator or none of them has, let $(F_1, F_2) \in B'$ and $a \in \text{AType}$, $l \in \text{PLSet}$, $C \in G_0 / B'$.

- If $(F_1, F_2) \in B'$ and $C \in G_0 / - \{[<a, \lambda >, E_1]_B, [<a, \lambda >, E_2]_B\},$ then trivially $\forall C \in G_0 / B'$.

- If $(F_1, F_2) \in B'$ and $C \in \{[<a, \lambda >, E_1]_B, [<a, \lambda >, E_2]_B\},$ then for $j \in \{1, 2\}$ we have

$$\forall C \in G_0 / B'$$

- If $(F_1, F_2) \in B'$, i.e. $F_1 \in [<a, \lambda >, E_1]_B$ and $F_2 \in [<a, \lambda >, E_2]_B$, then for $j \in \{1, 2\}$ we have

$$\forall C \in G_0 / B'$$

(ii) Given $L \subseteq \text{AType} - \{\tau\}$, we prove that

$$B' = B \cup Id_{E_0},$$

is a strong EMB. Observed that $B'$ is an equivalence relation, and that either each of the terms of an equivalence class has $\sim L$ as topmost operator or none of them has, let $(F_1, F_2) \in B'$ and $a \in \text{AType}$, $l \in \text{PLSet}$, $C \in G_0 / B'$.

- If $(F_1, F_2) \in Id_{E_0}$, then trivially $ERate(F_1, a, l, C) = ERate(F_2, a, l, C)$.

- If $(F_1, F_2) \in B'$, then $F_1 \equiv E_1 / L$ and $F_2 \equiv E_2 / L$ where $E_1 \sim E_2$.

- If none of the terms in $C$ has $\sim L$ as topmost operator, then trivially $ERate(F_1, a, l, C) = ERate(F_2, a, l, C)$.

(iii) Given $H \subseteq \text{AType}$, the proof that

$$B' = B \cup Id_{E_0},$$

is a strong EMB is similar to the one developed in (ii). The main difference is that in the last subcase the result follows from the fact that for $j \in \{1, 2\}$ we have

$$ERate(F_j, a, l, C) = \begin{cases} \text{ERate}(E_j, a, l, [E]_{\sim EMB}) & \text{if } - (a \in H \land l = -1) \\ \bot & \text{if } a \in H \land l = -1 \end{cases}$$

- Given $\varphi \in \text{Relab}$, the proof that

$$B' = B \cup Id_{E_0},$$

is a strong EMB is similar to the one developed in (ii). The main difference is that in the last subcase the result follows from the fact that for $j \in \{1, 2\}$ we have

$$ERate(F_j, a, l, C) = \begin{cases} \text{ERate}(E_j, a, l, [E]_{\sim EMB}) & \varphi(b) \neq a \\ \bot & \text{otherwise} \end{cases}$$

(v) The proof that

$$B' = B \cup Id_{E_0},$$

is a strong EMB is similar to the one developed in (ii). The main difference is that in the last subcase the result follows from the fact that for $j \in \{1, 2\}$ we have

$$ERate(F_j, a, l, C) = \begin{cases} \text{ERate}(E_j, a, l, [E]_{\sim EMB}) & \varphi(b) \neq a \\ \bot & \text{otherwise} \end{cases}$$
(vi) Let $B \subseteq G_\emptyset \times G_\emptyset$ be a strong EMB such that $(E_1, E_2) \in B$. Given $F \in G_\emptyset$, the proof that 

$$B' = (B \cup \{(E_1 + F, E_2 + F), (E_2 + F, E_1 + F)\})^+$$

is a strong EMB is similar to the one developed in (i). The main difference is that in the last subcase the result follows from the fact that for $j \in \{1, 2\}$ we have 

$$ERate(F_j, b, l, C) = ERate(E_j, b, l, C) \text{ or } ERate(F, b, l, C)$$

and from the following considerations:

- If $C \in G_\emptyset / B = \{(E_1 + F)_{[b]}(E_2 + F)_{[b]}\}$, then from $(E_1, E_2) \in B$ we derive $ERate(E_1, b, l, C) = ERate(E_2, b, l, C)$ so $ERate(F, b, l, C) = ERate(F_2, b, l, C)$.

- If $C = [E_1 + F]_{[b]} \cup [E_2 + F]_{[b]}$, then for $j \in \{1, 2\}$ we have 

$$ERate(E_j, b, l, C) = ERate(E_j, b, l, C) \text{ or } ERate(F, b, l, C) \text{ so } ERate(F_1, b, l, C) = ERate(F_2, b, l, C).$$

(vii) Given $S \subseteq AType - \{\tau\}$, the proof that 

$$B' = B \cup Id_{G_\emptyset}, \text{ where } B = \{(E_1 || F, E_2 || F) | E_1 \sim_{EMB} E_2 \land F \in G_\emptyset\},$$

is a strong EMB is similar to the one developed in (iii). The main difference is that in the last subcase, where given $E || F \in C$ it turns out that $C = \{E' || G, E' \in [E]_{\sim_{EMB}}\}$, the result follows from the considerations below:

- If $a \notin S$, then for $j \in \{1, 2\}$ we have that 

$$ERate(F_j, a, l, C) = \begin{cases} ERate(E_j, a, l, C) & \text{if } E \sim_{EMB} F \land E \neq G \land F = E \land [E]_{\sim_{EMB}} \land F \neq G. \text{ Since } E_1 \sim_{EMB} E_2, \text{ it follows that } \text{ERate}(F_1, a, l, C) = \text{ERate}(F_2, a, l, C). \\
\end{cases}$$

- If $a \in S$, then for $j \in \{1, 2\}$ we have that 

$$ERate(F_j, a, l, C) = \begin{cases} ERate(E_j, a, l, C) & \text{if } E \sim_{EMB} F \land E \neq G \land F = E \land [E]_{\sim_{EMB}} \land F \neq G. \text{ Since } E_1 \sim_{EMB} E_2, \text{ it follows that } \text{ERate}(F_1, a, l, C) = \text{ERate}(F_2, a, l, C). \\
\end{cases}$$

We conclude this section by proving that the strong EMB is preserved by recursive definitions as well. To do so, we first extend the definition of $\sim_{EMB}$ to terms that are guardedly closed up to constants devoid of defining equation. Note that such constants act as variables.

**Definition 6.14** A constant $A \in Const$ is free if and only if, for any $E \in L_\emptyset$, $A \Delta E \notin Def_\emptyset$. 

**Definition 6.15** A term $E \in L_\emptyset$ is partially guardedly closed (pgc) if and only if for each constant $A \in Const(E)$ either $A$ is free or

- $A$ is equipped in $Def_\emptyset$ with exactly one defining equation $A \Delta E'$, and
- there exists $F \in Subst(E')$ such that, whenever an instance of a nonfree constant $B$ satisfies $B st F$, then the same instance satisfies $B st \langle a, \lambda >. G st F$. 

**Definition 6.16** Let $E \in L_\emptyset$, $A \in Const$ free, and $B \in G_\emptyset$. The term $E(\langle A := B \rangle)$ obtained from $E$ by replacing each occurrence of $A$ with $B$ is defined by induction on the syntactical structure of $E$ as follows:

UBLCS-95-14 39
Theorem 6.18 Let $E_1, E_2 \in \mathcal{L}_\Theta$ be pgc, and suppose that $\text{Const}(E_1) \cup \text{Const}(E_2)$ contains only $A \in \text{Const}$ as a free constant. Let $A_1 \overset{\Delta}{=} E_1\langle A := A_1 \rangle$ and $A_2 \overset{\Delta}{=} E_2\langle A := A_2 \rangle$ be in $\mathcal{G}_\Theta$. If $E_1 \sim_{\text{EMB}} E_2$, then $A_1 \sim_{\text{EMB}} A_2$.

Proof It suffices to prove that $B' = B \cup B^{-1}$ where

\[ B = \{ (F_1, F_2) \mid F_1 \equiv F(\langle B := A_1 \rangle) \land F_2 \equiv F(\langle B := A_2 \rangle) \land \]

\[ F \in \mathcal{L}_\Theta \text{ pgc with at most } B \in \text{Const}(F) \text{ free} \]

is a strong EMB up to $\sim_{\text{EMB}}$: the result will follow by taking $F \equiv B$. We first observe that $B' \subseteq \mathcal{G}_\Theta \times \mathcal{G}_\Theta$ is reflexive (because if $F$ does not contain free variables then $F \in \mathcal{G}_\Theta$ and $F_1 \equiv F \equiv F_2$), symmetric (by definition), and transitive (for any $F \in \mathcal{L}_\Theta$ pgc with at most $B \in \text{Const}(F)$ free we have $(F_1, F_2) \in B'$ and $(F_2, F_1) \in B'$, and transitivity is guaranteed by $F_1, F_2 \in \mathcal{G}_\Theta$ and reflexivity). Given $(F_1, F_2) \in B'$, $a \in \text{Atype}$, $l \in \text{PLSet}$, and $C \in \mathcal{G}_\Theta / (\sim_{\text{EMB}} B' \sim_{\text{EMB}})$, we prove that $\text{ERate}(F_1, a, l, C) = \text{ERate}(F_2, a, l, C)$ by proceeding by induction on the maximum depth $d$ of the inference (based on the rules reported in the second part of Table 1) of a potential move for $F_1$ having type $a$, priority level $l$, and derivative term in $C$.

- If $d = 1$, then only the rule for the prefix operator has been used to deduce the potential move. Therefore $F \equiv \langle <a, \lambda> \rangle F'$ with $\text{PL}(\langle <a, \lambda> \rangle) = l$, and for $j \in \{1, 2\}$ we have $F_j \equiv \langle <a, \lambda> \rangle \langle F'(\langle B := A_j \rangle) \rangle$. Since $(F'\langle \langle B := A_1 \rangle \rangle, F'\langle \langle B := A_2 \rangle \rangle) \in B$, it turns out that

\[ C = [F'\langle \langle B := A_1 \rangle \rangle]_{\sim_{\text{EMB}}} [F'\langle \langle B := A_2 \rangle \rangle]_{\sim_{\text{EMB}}} \]

hence $\text{ERate}(F_1, a, l, C) = \lambda = \text{ERate}(F_2, a, l, C)$.

- If $d > 1$, then several subcases arise depending on the syntactical structure of $F$.

  - If $F \equiv F'/L$, then for $j \in \{1, 2\}$ we have $F_j \equiv (F'/\langle B := A_j \rangle)/L$. Since $F_1$ has a potential move having type $a$ (with $a \not\in L$), priority level $l$, and derivative term in $C$, such that the depth of its inference is $d$, $F'\langle \langle B := A_1 \rangle \rangle$ has a potential move having type $b$ (with $b = a$ if $a \neq \tau$, $b \in L \cup \{\tau\}$ if $a = \tau$), priority level $l$, and derivative term $G \in C' \in \mathcal{G}_\Theta / (\sim_{\text{EMB}} B' \sim_{\text{EMB}})$, such that the depth of its inference is $d - 1$ and $C = [G/L]_{\sim_{\text{EMB}}} [B' \sim_{\text{EMB}}]$. For $j \in \{1, 2\}$ we have

\[ \text{ERate}(F_j, a, l, C) = \begin{cases} \text{ERate}(F'\langle \langle B := A_j \rangle \rangle, a, l, C') \\ \text{ERate}(F'\langle \langle B := A_j \rangle \rangle, \tau, l, C') \end{cases} \]

depending on whether $a \not\in L \cup \{\tau\}$ or $a = \tau$. From the induction hypothesis, it follows that $\text{ERate}(F_1, a, l, C) = \text{ERate}(F_2, a, l, C)$. 

UBLCS-95-14 40
If \( F \equiv F' \setminus H \), then the proof is similar to the one developed in the first subcase. The result follows by applying the induction hypothesis to the fact that for \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C) = \text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C')
\]

If \( F \equiv F'[\phi] \), then the proof is similar to the one developed in the first subcase. The result follows by applying the induction hypothesis to the fact that for \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C) = \text{Min}\{\text{ERate}(F'(\langle B := A_j \rangle), b_i, l, C') \mid \phi(b) = a_i\}
\]

If \( F \equiv \Theta(F') \), then the proof is similar to the one developed in the first subcase. The result follows by applying the induction hypothesis to the fact that for \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C) = \text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C')
\]

If \( F \equiv F' + F'' \), then for \( j \in \{1, 2\} \) we have \( F_j \equiv (F'(\langle B := A_j \rangle) + F''(\langle B := A_j \rangle)) \). Since \( F_j \) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( C' \), such that the depth of its inference is \( d \), \( F'(\langle B := A_j \rangle) \) (\( F''(\langle B := A_j \rangle) \)) has the same move but the depth of its inference is \( d - 1 \). For \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C) = \text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C') \text{ Min} \text{ERate}(F''(\langle B := A_j \rangle), a_i, l, C')
\]
From the induction hypothesis, it follows that \( \text{ERate}(F_1, a_i, l, C) = \text{ERate}(F_2, a_i, l, C) \).

If \( F \equiv F'[s], F'' \), then for \( j \in \{1, 2\} \) we have \( F_j \equiv (F'(\langle B := A_j \rangle)[s] F''(\langle B := A_j \rangle)) \). Suppose that \( F_1 \) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( C' \), such that the depth of its inference is \( d \).

* If \( a_i \notin S \), then \( F'(\langle B := A_j \rangle)(F''(\langle B := A_j \rangle)) \) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( G' \in C' \times G' \sim EMB B' \sim EMB \), such that the depth of its inference is \( d - 1 \) and \( C' = [G' \times G' \sim EMB B' \sim EMB]) \). For \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C') = \text{Min}(\text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C'), \text{ERate}(F''(\langle B := A_j \rangle), a_i, l, C'))
\]
depending on whether \( F'(\langle B := A_j \rangle) \in C' \) or \( F''(\langle B := A_j \rangle) \notin C' \). From the induction hypothesis, it follows that \( \text{ERate}(F_1, a_i, l, C) = \text{ERate}(F_2, a_i, l, C) \).

* If \( a_i \in S \), then \( F'(\langle B := A_j \rangle)(F''(\langle B := A_j \rangle)) \) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( G' \in C' \times G' \sim EMB B' \sim EMB \), such that the depth of its inference is at most \( d - 1 \), and \( F'(\langle B := A_j \rangle) \) (\( F''(\langle B := A_j \rangle) \)) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( G' \in C' \times G' \sim EMB B' \sim EMB \), such that the depth of its inference is at most \( d - 1 \); besides, \( C' = [G' \times G' \sim EMB B' \sim EMB]) \). For \( j \in \{1, 2\} \) we have
\[
\text{ERate}(F_j, a_i, l, C') = \text{Min}(\text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C'), \text{ERate}(F''(\langle B := A_j \rangle), a_i, l, C'))
\]
depending on whether \( \text{ERate}(F'(\langle B := A_j \rangle), a_i, l, C')) \notin \cup \) or \( \text{ERate}(F''(\langle B := A_j \rangle), a_i, l, C')) \notin \cup \). From the induction hypothesis, it follows that \( \text{ERate}(F_1, a_i, l, C) = \text{ERate}(F_2, a_i, l, C) \).

If \( F \equiv B' \), then for \( j \in \{1, 2\} \) we have \( F_j \equiv B'(\langle B := A_j \rangle) \).

* If \( B' \equiv B \), then for \( j \in \{1, 2\} \) we have \( F_j \equiv A_i \). Since \( F_1 \) has a potential move having type \( a_i \), priority level \( l \), and derivative term in \( C' \), such that the depth of its inference is \( d \), then \( E_1(\langle A := A_i \rangle) \) has the same potential move but the depth of its inference is \( d - 1 \). For \( j \in \{1, 2\} \) we have \( \text{ERate}(F_j, a_i, l, C) = \text{ERate}(F_j, a_i, l, C) \). From the induction hypothesis and the fact that \( E_1 \sim EMB E_2 \), it follows that \( \text{ERate}(F_1, a_i, l, C) = \text{ERate}(F_2, a_i, l, C) \).

* If \( B' \neq B \), then \( B' \in G_E \) and the result trivially follows from the fact that \( F_1 \equiv B' \equiv F_2 \).
6.3 Relationship with the projected equivalences

In this section we investigate the relationship between \( \sim_{EMB} \) and \( \sim_{FP} \). The first result we prove is that the inclusion \( \sim_{EMB} \subseteq \sim_{FP} \) holds in \( E \times E \).

**Theorem 6.19** Let \( E_1, E_2 \in G \). If \( E_1 \sim_{EMB} E_2 \) then \( E_1 \sim_{F} E_2 \).

**Proof** It follows immediately from the definitions of \( \sim_{EMB} \) and \( \sim_{F} \).

**Theorem 6.20** Let \( E_1, E_2 \in E \). If \( E_1 \sim_{EMB} E_2 \) then \( E_1 \sim_{F} E_2 \).

**Proof** Let \( E_1, E_2 \in E \) and let \( B \subseteq G_0 \times G_0 \) be a strong EMB such that \( (E_1, E_2) \in B \) (it is understood that \( (\Theta(E_1), \Theta(E_2)) \in B \)). The proof is divided into two parts.

(1st part) We demonstrate that from the hypothesis it follows that \( \mathcal{M}_0[E_1] \) and \( \mathcal{M}_0[E_2] \) are p-bisimilar. Let us group the steps of the first phase of the algorithm for determining the Markovian semantics into macrosteps, where a given macrostep results in the elimination of the forks of immediate transitions whose upstream vanishing states belong to the same vanishing equivalence class of \( (\uparrow E_1 \cup \uparrow E_2)/B \). Let us denote by \( \mathcal{P}_0[E_1] \) and \( \mathcal{P}_0[E_2] \) the two PLTSs produced by the execution of the first step (up to immediate selfloop removal), and by \( \mathcal{P}_1[E_1] \) and \( \mathcal{P}_1[E_2] \) the two PLTSs produced by the execution of macrostep \( h \geq 1 \) related to vanishing equivalence class \( C_h \) (up to removal of immediate selfloops incident on states not belonging to \( C_h \)). We prove that \( \mathcal{P}_h[E_1] \) and \( \mathcal{P}_h[E_2] \) are p-bisimilar by proceeding by induction on the number \( h \in \mathbb{N} \) of vanishing equivalence classes of \( (\uparrow E_1 \cup \uparrow E_2)/B \).

- Let \( h = 0 \). We prove that \( B_0 = B \cap (\{E_1, E_2\} \times \{E_1, E_2\}) \) is a p-bisimulation between \( \mathcal{P}_0[E_1] \) and \( \mathcal{P}_0[E_2] \). Observe that \( B_0 \) is an equivalence relation, and that \( (\{E_1, E_2\})/B_0 = \{C \mid C' \cap (\{E_1, E_2\} \times \{E_1, E_2\}) \neq \emptyset \land C' \in G_0/B \} \)

- Let \( C \in (\{E_1, E_2\})/B_0 \). For \( j \in \{1, 2\} \) we have \( P_1(s_1, s_2) = \frac{1}{2} \) if \( E_j \in C \) and \( 1 \) if \( E_j \notin C \), so \( \sum_{s \in C \cap S_{E_j,1}} P_1(s_1, s_2) = \frac{1}{2} \)

- Let \( (s_1, s_2) \in B_0 \cap (\{E_1, E_2\} \times \{E_1, E_2\}) \). Let \( C \in (\{E_1, E_2\})/B_0 \), and let \( C' \) be the corresponding equivalence class in \( G_0/B \). Since \( h = 0 \), and \( s_1 \) and \( s_2 \) are tangible. For \( j \in \{1, 2\} \) we have \( \Min \{ \lambda \mid s_j \xrightarrow{\lambda} E_{j,1} s_j \} \leq C \cap S_{E_j,1} \) as \( S_{E_j,1} = \uparrow E_j \). Since \( (s_1, s_2) \in B \), and \( C' \in G_0/B \), it turns out \( \Min \{ \lambda \mid s_1 \xrightarrow{\lambda} E_{1,1} s_1 \} \leq C \cap S_{E_1,1} \) as \( S_{E_1,1} = \uparrow E_1 \).
Let $C \in (S_{E_1,k_1} \cup S_{E_2,k_2})/B_h$, and $C'$ be the corresponding equivalence class in $(S_{E_1,k_1} \cup S_{E_2,k_2})/B_{h-1}$.

* If $C_h \cap (S_{E_1,k_1} \cup S_{E_2,k_2}) \neq \emptyset$, then there is no state in $C_h$ having transitions to states not in $C_h$, so for $j \in \{1,2\}$ we have

$$\sum_{s \in C \cap S_{E_1,k_1}} P_{E_j,k_1}(s) = \sum_{s \in C' \cap S_{E_1,k_1}} P_{E_j,k_1}(s)$$

From the induction hypothesis, it follows that

$$\sum_{s \in C \cap S_{E_2,k_2}} P_{E_j,k_1}(s) = \sum_{s \in C' \cap S_{E_2,k_2}} P_{E_j,k_1}(s)$$

* If $C_h \cap (S_{E_1,k_1} \cup S_{E_2,k_2}) = \emptyset$, given $s_h \in C_h \cap S_{E_1,k_1}$ let $p, q \in \mathbb{R}_{[0,1]}$ be defined by

$$\infty_{l,p} = \min\{\infty_{l,p} \mid s_h \xrightarrow{*_{E_1,k_1}} s' \land s' \notin C' \cup S_{E_1,k_1}\}$$

and

$$\infty_{l,q} = \min\{\infty_{l,q} \mid s_h \xrightarrow{*_{E_1,k_1}} s' \land s' \notin C' \cup S_{E_1,k_1}\}$$

Note that $q$ is well defined because each state in $C_h$ has transitions to states not in $C_h$, whereas $p$ could be undefined and in this case $p$ is taken to be 0 for convenience. Then for $j \in \{1,2\}$ we have

$$\sum_{s \in C \cap S_{E_1,k_1}} P_{E_j,k_1}(s) = \sum_{s \in C' \cap S_{E_2,k_2}} P_{E_j,k_1}(s)$$

From the induction hypothesis, it follows that

$$\sum_{s \in C \cap S_{E_2,k_2}} P_{E_j,k_1}(s) = \sum_{s \in C' \cap S_{E_2,k_2}} P_{E_j,k_1}(s)$$

Let $(s_1,s_2) \in B_h \cap (S_{E_1,k_1} \times S_{E_2,k_2})$. Let $C \in (S_{E_1,k_1} \cup S_{E_2,k_2})/B_h$, and $C'$ be the corresponding equivalence class in $(S_{E_1,k_1} \cup S_{E_2,k_2})/B_{h-1}$. Note that $(s_1,s_2) \in B_{h-1} \cap (S_{E_1,k_1} \times S_{E_2,k_2})$.

* If $C_h \cap (S_{E_1,k_1} \cup S_{E_2,k_2}) \neq \emptyset$, then there is no state in $C_h$ having transitions to states not in $C_h$, so for $j \in \{1,2\}$ we have

$$\min\{\hat{\lambda} \mid s_j \xrightarrow{E_j,k_1} s'_j \land s'_j \in C \cap S_{E_j,k_1}\} = \min\{\hat{\lambda} \mid s_j \xrightarrow{E_j,k_1} s'_j \land s'_j \in C' \cap S_{E_j,k_1}\}$$

From the induction hypothesis, it follows that

$$\min\{\hat{\lambda} \mid s_1 \xrightarrow{E_1,k_1} s'_1 \land s'_1 \in C \cap S_{E_1,k_1}\} = \min\{\hat{\lambda} \mid s_1 \xrightarrow{E_1,k_1} s'_1 \land s'_1 \in C' \cap S_{E_1,k_1}\}$$

* If $C_h \cap (S_{E_1,k_1} \cup S_{E_2,k_2}) = \emptyset$, given $s_h \in C_h \cap S_{E_1,k_1}$ let $p, q \in \mathbb{R}_{[0,1]}$ be defined by

$$\infty_{l,p} = \min\{\infty_{l,p} \mid s_h \xrightarrow{*_{E_1,k_1}} s' \land s' \notin C' \cup S_{E_1,k_1}\}$$

and

$$\infty_{l,q} = \min\{\infty_{l,q} \mid s_h \xrightarrow{*_{E_1,k_1}} s' \land s' \notin C' \cup S_{E_1,k_1}\}$$

Note that $q$ is well defined because each state in $C_h$ has transitions to states not in $C_h$, whereas $p$ could be undefined and in this case $p$ is taken to be 0 for convenience. Then for $j \in \{1,2\}$ we have

$$\min\{\hat{\lambda} \mid s_j \xrightarrow{E_j,k_1} s'_j \land s'_j \in C \cap S_{E_j,k_1}\} = \min\{\hat{\lambda} \mid s_j \xrightarrow{E_j,k_1} s'_j \land s'_j \in C' \cap S_{E_j,k_1}\}$$
Split(Min[λ | s_j →_{E_j,k_j} \overrightarrow{s}_h \land s_h \in C_h \cap S_{E_j,k_j} ] \triangleright q)

From the induction hypothesis, it follows that
\[ \text{Min}[\lambda | s_1 \rightarrow_{E_1,k_1} \overrightarrow{s}_1 \land s'_1 \in C \cap S_{E_1,k_1} ] = \]
\[ \text{Min}[\lambda | s_2 \rightarrow_{E_2,k_2} \overrightarrow{s}_2 \land s'_2 \in C \cap S_{E_2,k_2} ] = \]

\textbf{(2nd part)} We demonstrate that, from the result proved in the first part, it follows that \( M \llbracket E_1 \rrbracket \) and \( M \llbracket E_2 \rrbracket \) are p-bisimilar. Let \( B_{M_0} \) be a p-bisimilarity between \( M_0 \llbracket E_1 \rrbracket \) and \( M_0 \llbracket E_2 \rrbracket \). We prove that
\[ B_{M} = \{(C_1, C_2) \in (S_{E_1,M} \cup S_{E_2,M}) \times (S_{E_1,M} \cup S_{E_2,M}) | \exists s_1 \in C_1, \exists s_2 \in C_2, (s_1, s_2) \in B_{M_0}\} \]
is a p-bisimulation between \( M \llbracket E_1 \rrbracket \) and \( M \llbracket E_2 \rrbracket \). Note that \( B_{M} \) is an equivalence relation because so is \( B_{M_0} \).

- Let \( D \in (S_{E_1,M} \cup S_{E_2,M}) / B_{M} \). For \( j \in \{1, 2\} \) we have
\[ \sum_{C \in D \cap S_{E_jM}} P_{E_j,M}(C) = \sum_{C \in D \cap S_{E_jM}} \sum_{s \in C} P_{E_j,M_0}(s) \]
Since \( \cup_{C \in D} C \) is the union of some equivalence classes with respect to \( B_{M_0} \), and \( B_{M_0} \) is a p-bisimulation, it follows that
\[ \sum_{C \in D \cap S_{E_jM}} P_{E_j,M}(C) = \sum_{C \in D \cap S_{E_jM}} P_{E_j,M_0}(C) \]

- Let \( (C_1, C_2) \in B_{M} \cap (S_{E_1,M} \times S_{E_2,M}) \) due to the existence of \( s_1 \in C_1 \) and \( s_2 \in C_2 \) such that \( (s_1, s_2) \in B_{M_0} \). Let \( D \in (S_{E_1,M} \cup S_{E_2,M}) / B_{M} \). For \( j \in \{1, 2\} \) we have
\[ \text{Min}[\lambda | C_j \rightarrow_{E_j,M} C'_j \land C'_j \in D \cap S_{E_jM} ] = \]
\[ \text{Min}[\lambda | s_j \rightarrow_{E_j,M_0} s'_j \land s'_j \in C'_j \in D \cap S_{E_jM} ] = \]
\[ \text{Min}[\lambda | s_j \rightarrow_{E_j,M_0} s'_j \land s'_j \in (\cup_{C \in D} C) \cap S_{E_jM} ] \]
where the first equality holds whichever is \( s_j \in C_1 \) because \( M \llbracket E_1 \rrbracket \) is obtained from \( M_0 \llbracket E_1 \rrbracket \) via ordinary lumping. Since \( \cup_{C \in D} C \) is the union of some equivalence classes with respect to \( B_{M_0} \), and \( B_{M_0} \) is a p-bisimulation, it follows that
\[ \text{Min}[\lambda | C_1 \rightarrow_{E_1,M} C'_1 \land C'_1 \in D \cap S_{E_1M} ] = \]
\[ \text{Min}[\lambda | C_2 \rightarrow_{E_2,M} C'_2 \land C'_2 \in D \cap S_{E_2M} ] \]

The inclusion \( \sim_{EMB} \subseteq \sim_{FP} \) in \( \mathcal{E} \times \mathcal{E} \) is strict, as one can see by considering the examples below. Additionally, such examples show that \( \sim_{EMB} \) cannot abstract from either priority levels or weights of immediate actions; otherwise, the congruence property would get lost.

\textbf{Example 6.21} Consider terms \( E_1 \) and \( E_2 \) of Example 6.3. Then \( E_1 \sim_{FP} E_2 \) but \( E_1 \not\sim_{EMB} E_2 \) because \( ERATE(E_1, a, 0, [\emptyset \sim_{EMB}]) \neq ERATE(E_2, a, 0, [\emptyset \sim_{EMB}]) \).

\textbf{Example 6.22} Consider terms \( E_1 \) and \( E_2 \) of Example 6.4. Then \( E_1 \sim_{FP} E_2 \) but \( E_1 \not\sim_{EMB} E_2 \) because \( ERATE(E_1, a, 1, [\emptyset \sim_{EMB}]) \neq ERATE(E_2, a, 1, [\emptyset \sim_{EMB}]) \). If we relax Definition 6.6 to abstract from the priority level of immediate actions, then \( E_1 \sim_{EMB} E_2 \) but this new strong EMBE would be closed under neither the alternative composition operator nor the parallel composition operator. For example, \( E_1 + <b, \infty, 1, >.0 \not\sim_{EMB} E_2 + <b, \infty, 1, >.0 \) and \( E_1 || <b, \infty, 1, >.0 \not\sim_{EMB} E_2 || <b, \infty, 1, >.0 \) because the left-hand side terms can execute an action with type \( b \) while the right-hand side terms cannot.
Example 6.23 Consider terms $E_1$ and $E_2$ of Example 6.5. Then $E_1 \sim_{FP} E_2$ but $E_1 \not\simeq_{EMB} E_2$ because $E_{Rate}(E_1, a, 1, \emptyset) \neq E_{Rate}(E_2, a, 1, \emptyset)$. If we relax Definition 6.6 to consider execution probabilities instead of weights for immediate actions (see the notion of relative bisimulation proposed in [67]), then $E_1 \sim_{EMB} E_2$ but this new strong EMB would be closed under neither the alternative composition operator nor the parallel composition operator. For example, $E_1 + (b, 0, 1, >) \not\simeq_{EMB} E_2$ and $E_1 \parallel (b, 0, 1, >) \not\simeq_{EMB} E_2$ because the left-hand side terms can execute actions having type $a$ with probability $1/2$ while the right-hand side terms can execute actions having type $a$ with probability $2/3$.

As a matter of fact, the second result we prove is that $\sim_{EMB}$, restricted to the set $\mathcal{E}_{\tau=\infty}$ of terms in $\mathcal{E}$ whose interleaving semantics has no immediate internal transitions, is the coarsest congruence contained in $\sim_{FP}$.

Theorem 6.24 Let $E_1, E_2 \in \mathcal{E}_{\tau=\infty}$. Then $E_1 \sim_{EMB} E_2$ if and only if, for all $F \in \mathcal{G}$ and $S \subseteq \mathcal{AType} - \{\tau\}$ such that $E_1 + F, E_2 + F, E_1 \parallel_{S} F, E_2 \parallel_{S} F \in \mathcal{E}_{\tau=\infty}$, it turns out that $E_1 + F \sim_{FP} E_2 + F$ and $E_1 \parallel_{S} F \sim_{FP} E_2 \parallel_{S} F$.

Proof ($\Rightarrow$) Since $E_1 \sim_{EMB} E_2$ and $\sim_{EMB}$ is a congruence, for all $F \in \mathcal{G}$ and $S \subseteq \mathcal{AType} - \{\tau\}$ we have $E_1 + F \sim_{EMB} E_2 + F$ and $E_1 \parallel_{S} F \sim_{EMB} E_2 \parallel_{S} F$. Since $\sim_{EMB} \subset \sim_{FP}$ in $\mathcal{E} \times \mathcal{E}$, the result follows.

($\Leftarrow$) We prove the contrapositive, so we assume that $E_1 \not\simeq_{EMB} E_2$ and we demonstrate that $E_1$ and $E_2$ are distinguishable with respect to $\sim_{FP}$ by means of an appropriate context based on the alternative composition operator or the parallel composition operator. We proceed by induction on the number $n$ of actions that $E_1$ and $E_2$ have to execute in order to become $E'_1$ and $E'_2$, respectively, such that $E'_1 \not\simeq_{EMB} E'_2$ because they violate the necessary condition expressed by Proposition 6.10($\tilde{a}$). 9

- Let $n = 0$, i.e., assume that there exist $a \in \mathcal{AType}$ and $l \in \mathcal{PLSet}$ such that $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} \neq \exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$. Since both sides of the inequality cannot be $\perp$, we assume that $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} \neq \perp$. There are several cases:
  - Assume that $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} \neq \perp$.
    - Assume that $l = 0$.
      - If $a = \tau$ or $a \neq \tau$ and $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} \neq \exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$.
        - $E_1 \parallel_{\mathcal{AType} - \{\tau\}} \not\simeq_{FP} E_2 \parallel_{\mathcal{AType} - \{\tau\}} \not\simeq_{FP}$ because the exit rates of these two terms are $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$ and $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$, respectively, hence the necessary condition for p-bisimilarity expressed by Proposition 4.3 is violated.
      - If $a \neq \tau$ and $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} = \exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$.
        - $E_1 \parallel_{\mathcal{AType} - \{\tau\}} \not\simeq_{FP} E_2 \parallel_{\mathcal{AType} - \{\tau\}} \not\simeq_{FP}$ because the exit rates of these two terms are $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$ and $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$, respectively, hence the necessary condition for p-bisimilarity expressed by Proposition 4.3 is violated.
    - Assume that $l \geq 1$. From $E_1, E_2 \in \mathcal{E}_{\tau=\infty}$, it follows that $a \neq \tau$ and $\exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \} \neq \exists \{E\} \mid E \in \mathcal{E}_{\tau=\infty} \}$. As a consequence, given $b \in \mathcal{AType} - \{\tau\}$ occurring neither in $E_1$ nor in $E_2$, and $a \in \mathbb{R}_+$ smaller than the smallest rate of the possible exponentially timed actions occurring in $E_1$ and $E_2$, we have that $\ldots$

9. If such $E'_1$ and $E'_2$ did not exist, then $E_1 \not\simeq_{EMB} E_2$ would not hold.
because the state having the transition labeled with \( h \) has initial state probability \( \omega/(\text{MIN}(\text{ERATE}(E_1, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} + \omega) \) in the Markovian semantics of the right-hand side term, \( \omega/(\text{MIN}(\text{ERATE}(E_2, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} + \omega) \) in the Markovian semantics of the right-hand side term.

- Assume that \( \text{MIN}(\text{ERATE}(E_2, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} = \perp \).
  * Assume that there exists \( l' \in \text{PLSet} - \{\} \) such that \( \text{MIN}(\text{ERATE}(E_2, a, l', \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} \neq \perp \). From \( E_1, E_2 \in \mathcal{E}_{\infty} \{\} \), it follows that \( a \neq \tau \). Then, given \( b \in \text{ATYPE} - \{\tau\} \) occurring neither in \( E_1 \) nor in \( E_2 \), we have that

\[
E_1 \|_{\text{ATYPE} - \{\tau\}} (a, *, \Omega) + (b, \infty, \Omega, b, \Omega) \neq F
\]

\[
E_2 \|_{\text{ATYPE} - \{\tau\}} (a, *, \Omega) + (b, \infty, \Omega, b, \Omega) \neq F
\]

because one of these two terms can execute an action with type \( b \) while the other term cannot.

* Assume that \( \text{MIN}(\text{ERATE}(E_2, a, l', \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} = \perp \). If \( a = \tau \), then

\[
E_1 \|_{\text{ATYPE} - \{\tau\}} \Omega \neq F
\]

\[
E_2 \|_{\text{ATYPE} - \{\tau\}} \Omega \neq F
\]

because the left-hand side term can execute an action with type \( \tau \) while the right-hand side term cannot.

- Let \( u \geq 1 \), i.e. assume that for all \( a \in \text{ATYPE} \) and \( l \in \text{PLSet} \) it turns out that \( \text{MIN}(\text{ERATE}(E_1, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} = \text{MIN}(\text{ERATE}(E_2, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} \).\ From \( E_1 \neq \text{EMB} \) and the hypothesis above, it follows that there exist \( a \in \text{ATYPE} \), \( l \in \text{PLSet} \) and \( C, C' \in \mathcal{E}_{\infty} / \text{EMB} \) such that \( \text{ERATE}(E_1, a, l, C) \neq \text{ERATE}(E_2, a, l, C') \), \( \text{ERATE}(E_1, a, l, C') \neq \text{ERATE}(E_2, a, l, C') \), \( C \cap C' = \emptyset \), and \( \text{ERATE}(E_1, a, l, C) \text{MIN} \text{ERATE}(E_1, a, l, C') \neq \perp \neq \text{ERATE}(E_2, a, l, C) \text{MIN} \text{ERATE}(E_2, a, l, C') \). As a consequence, there exist \( F_1 \in C(C') \) and \( F_2 \in C'(C) \) reachable from \( E_1 \) and \( E_2 \), respectively, by executing an action having type \( a \) and priority level \( l \) such that we can apply the induction hypothesis to \( F_1 \) and \( F_2 \). Let \( \text{MIN}(\text{ERATE}(E_1, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} = \text{MIN}(\text{ERATE}(E_2, a, l, \{E\})) \mid E \in \mathcal{E}_{\infty} \{\} \). If \( a = \tau \), then

\[
E_1 \|_{\text{ATYPE} - \{\tau\} - S} F \neq F
\]

\[
E_2 \|_{\text{ATYPE} - \{\tau\} - S} F \neq F
\]

provided that the constraint on the possible element of \( S \), and the constraint on the rate of the possible exponentially timed action occurring in \( F \), are satisfied.

In the following example we show the problems that arise when immediate internal transitions come into play.

**Example 6.25** Consider terms

\[
E_1 \equiv \langle a, \infty, 1, 2 \rangle . A
\]

\[
E_2 \equiv \langle a, \infty, 1, 2 \rangle . B
\]

where

\[
A \equiv \langle \tau, \infty, 1, 2 \rangle . A
\]

\[
B \equiv \langle \tau, \infty, 1, 2 \rangle . B
\]
It turns out that $E_1 \not\sim_{EMB} E_2$ because $A$ and $B$ violate the necessary condition expressed by Proposition 6.10(ii), and that $E_1$ and $E_2$ cannot be distinguished with respect to $\sim_{FP}$ by means of a context based on the alternative composition operator or the parallel composition operator.

Concerning the alternative composition operator, it does not allow us to introduce a choice at the level of $A$ and $B$.

Concerning the parallel composition operator, in principle it allows us to introduce a choice at the level of $A$ and $B$. However, we also have to introduce an exponentially timed action, in such a way that the state having a transition labeled with this action has different initial state probabilities in the Markovian semantics of the two resulting terms. The problem is that such an action cannot be executed at all because $A$ and $B$ always have an higher priority level action ready to be executed, and this action cannot be blocked by means of an appropriate synchronization set as it is internal.

We conclude by observing that $\sim_{FP}$ is defined over temporally closed terms, while $\sim_{EMB}$ is more general because it handles also terms that are not temporally closed. Furthermore, the treatment of these terms, which contain pure nondeterminism, is consistent with the treatment in classical process algebras.

### 6.4 Granularity of queueing system description

QSs $M/M/n/g/m$ can be given different descriptions with EMPA whenever $1 < n < \infty$ or $1 < m < \infty$: in the first case there are finitely many independent servers, in the second case finitely many independent customers. If we restrict ourselves to the case $1 < n < \infty$, then servers can be given:

- A **fine grain** representation comprising $n$ terms composed in parallel, each modeling a single server. Every state of the integrated semantic model of this representation keeps the information about which servers are busy.
- A **coarse grain** representation comprising $n + 1$ indexed terms, each modeling the situation where $0 \leq i < n$ servers are busy. Every indexed term is composed of two alternative terms describing the two possible evolutions: either an idle server becomes busy, or a busy server becomes idle. Every state of the integrated semantic model of this representation keeps only the information about how many servers are busy.

One expects that these two differently grained representations of QSs are equivalent. To prove that this is the case, we exploit the strong EMB. For the sake of simplicity, let us consider a QS $M/M/n/n, n \geq 2$, with arrival rate $\lambda$ and service rate $\mu$; recall that such a QS has no queue and unboundedly many customers. Its fine grain representation is the following:

- $\text{System}_{M/M/n/n} \overset{\Delta}{=} \text{Arrivals} \parallel_{|a|} \text{Servers}_n$:
  - $\text{Arrivals} \overset{\Delta}{=} <a, \lambda>.\text{Arrivals}$;
  - $\text{Servers}_n \overset{\Delta}{=} S \parallel S \parallel \ldots \parallel S$;

* $S \overset{\Delta}{=} <a, \lambda>.S$;

whereas its coarse grain representation is the following:

- $\text{System}'_{M/M/n/n} \overset{\Delta}{=} \text{Arrivals} \parallel_{|a|} \text{Servers}'_0$:
  - $\text{Arrivals} \overset{\Delta}{=} <a, \lambda>.\text{Arrivals}$;
  - $\text{Servers}'_0 \overset{\Delta}{=} <a, \lambda>,\text{Servers}'_1$,
  - $\text{Servers}'_i \overset{\Delta}{=} <a, \lambda>,\text{Servers}'_{i+1} + <<\lambda, i, \mu>,\text{Servers}'_{i+1}, 1 \leq i \leq n - 1$,
  - $\text{Servers}'_n \overset{\Delta}{=} <<\lambda, n, \mu>,\text{Servers}'_{n+1}$.

Since in these representations immediate actions do not occur, we have $\Theta(\text{System}_{M/M/n/n}) \sim_{EMB} \text{System}_{M/M/n/n}$ and $\Theta(\text{System}'_{M/M/n/n}) \sim_{EMB} \text{System}'_{M/M/n/n}$. We then take advantage of the fact that $\sim_{EMB}$ is a congruence: to prove $\text{System}_{M/M/n/n} \sim_{EMB} \text{System}'_{M/M/n/n}$ it suffices to prove $\text{Servers}_n \sim_{EMB} \text{Servers}'_0$. This is the case because of the strong EMB up to $\sim_{EMB}$ given by the reflexive, symmetric and transitive closure of the relation made out of the following pairs of
terms

\[ S \parallel S \parallel \ldots \parallel S, \quad Servers'_0 \]
\[ <s, \mu> \parallel S \parallel S \parallel \ldots \parallel S, \quad Servers'_1 \]
\[ <s, \mu> \parallel <s, \mu> \parallel S \parallel \ldots \parallel S, \quad Servers'_2 \]
\[ \ldots \]
\[ <s, \mu> \parallel S \parallel <s, \mu> \parallel S \parallel \ldots \parallel <s, \mu> \parallel S, \quad Servers'_{n} \]

A similar result can be shown for every QS \( M/M/n/q/m \) such that \( 1 < n < \infty \) or \( 1 < m < \infty \).

Incidentally, we recall that in [17] it has been proved that the Markovian semantics of every fine grain representation is \( p \)-isomorphic to the Markovian semantics of the corresponding coarse grain representation, which in turn is \( p \)-isomorphic to the HCTMC underlying the corresponding QS; since the ordinary lumping procedure comes into play during the computation of the Markovian semantics of fine grain representations, the ordinary lumping phase does not cause information loss in the case of QSs. Additionally, in the particular case of the above fine grain representation of a QS \( M/M/n/n \), \( n \geq 2 \), we observe that the need of normalizing rates in the semantic rule for the parallel composition operator (see Table 1) becomes evident: if we ignore function \( \text{Norm} \), then we get a wrong HCTMC.

7 Integrated net semantics of EMPA terms

In order to implement the second phase of the integrated approach of Figure 1, we must provide each EMPA term with a net semantics accounting for both functional and performance aspects. As explained in Section 1, a good candidate for the integrated net model is the class of generalized stochastic Petri nets, because they take into account performance aspects since the beginning of the design process, and are endowed with tools for the analysis of projected models.

This section is organized as follows. In Section 7.1 we recall some notions about generalized stochastic Petri nets and we focus our attention on an extension of them, acting as semantic model in this framework. In Section 7.2 we define the integrated operational net semantics for EMPA. The consistency of this semantics with respect to the integrated operational interleaving one is assessed in Section 7.3 by showing that it satisfies the functional and performance retrievability principles, while its completeness is evaluated in Section 7.4 by showing that it meets the concurrency principle. In Section 7.5 we illustrate an example based on QSs.

7.1 Passive generalized stochastic Petri nets

In this section we shall be concerned with the class of the generalized stochastic Petri nets (GSPNs) [5, 6]. They are essentially place/transition nets [59] equipped with inhibitor arcs whose transitions are either exponentially timed (i.e. their durations are expressed by means of exponentially distributed random variables) or immediate (i.e. their durations are zero, so they take precedence over exponentially timed ones). Furthermore, immediate transitions are divided into priority levels and have weights that can depend on the current marking \( M_{\text{curr}} \). The race policy is adopted whenever several transitions are simultaneously executable.

Since GSPNs do not admit passive transitions, and since we need passive transitions to carry out the translation of EMPA passive actions, we propose below an extension of GSPNs where passive transitions (not involved in the priority mechanism) are included.

Definition 7.1 A passive generalized stochastic Petri net (PGSN) is a tuple

\((P, U, T, M_0, L, W)\)

such that:

- \( P \) is a set whose elements are called places;
- \( U = U^{|M_{\text{fin}}(P)|} \) is a set whose elements are called labels;
- \( T \subseteq M_{\text{fin}}(P) \times P_{\text{fin}}(P) \times U \times M_{\text{fin}}(P) \) whose elements are called transitions;
- \( M_0 \in M_{\text{fin}}(P) \) is called the initial marking;
- \( L : T \rightarrow P \text{Set} \), called priority function, is such that:


UBLCS-95-14 48
Let \( N = (P, U, T, M_0, L, W) \) be a PGSPN.

- A marking of \( N \) is an element of \( \mathcal{M}_{\text{fin}}(P) \).
- A transition \( t \) is enabled at marking \( M \) if and only if \( *t \subseteq M \) and \( \text{dom}(M) \cap \{ t \} = \emptyset \). We denote by \( E(M) \) the set of transitions enabled at marking \( M \).
- Transition \( t \) enabled at marking \( M \) can fire if and only if either \( L(t) = -1 \) or \( L(t) \) is the highest priority level among the transitions in \( E(M) \).
- The reachability set \( \mathcal{R}(M) \) of a marking \( M \) is the least subset of \( \mathcal{M}_{\text{fin}}(P) \) such that:
  - \( M \in \mathcal{R}(M) \);
  - if \( M_1 \in \mathcal{R}(M) \) and \( M_1 \{ u \} M_2 \), then \( M_2 \in \mathcal{R}(M) \).
- The reachability graph (or interleaving marking graph) of \( N \) is the LTS \( \mathcal{RG}[N] = (\mathcal{R}(M_0), \{ \} \cup \{ }, M_0) \).

If \( \widehat{U} \) is the set of active places, then from \( \mathcal{RG}[N] \) we can extract the functional semantics of \( N \) as the LTS \( \mathcal{F}[N] \) obtained by keeping only the action type information. And, provided that \( \mathcal{RG}[N] \) has no passive transitions, also the Markovian semantics as the HCTMC \( \mathcal{M}[N] \) by applying the algorithm defined in Section 4 to \( \mathcal{RG}[N] \). Since in the following inhibitor arcs will not come into play, i.e. inhibitor sets will be empty, each transition \( t \) is written as \( *t \xrightarrow{u} t* \).

### 7.2 Integrated operational location-oriented net semantics

The integrated operational semantics of a term \( E \in \mathcal{G} \) is obtained by resorting to a suitable extension of the approach followed in Section 3. The idea [29, 55] consists of associating with every term \( E \) a net such that:

1. Net places correspond to the sequential subterms of \( E \) and its derivative terms.
2. Net transitions are defined by induction on the syntactical structure of the sets of sequential terms.
3. Net markings correspond roughly to \( E \) and its derivative terms.

This approach is called location-oriented because all the information about the syntactical structure of terms is encoded inside places.

---

11. Note that the firing rule includes the race policy.
In this section we adapt the proposal in [55] to our stochastic framework. To be more precise, we firstly introduce the syntax of net places, then we inductively define net transitions, and finally we present nets associated with EMPA terms. The construction of the integrated operational net semantics is illustrated by means of a running example.

**Example 7.3** The integrated operational net semantics will be clarified by considering term

\[ E \equiv (<a, \lambda>, Q) \parallel_b <b, \mu>, Q) \parallel <c, \gamma>, Q \]

The LTS \( I[E] \) is reported below:

![LTS Diagram]

Observe that from this semantic model it is not possible to understand whether actions with type \( a \) and \( b \), respectively, are causally dependent or completely independent of each other. The reason is that, in an interleaving model, a parallel execution is simulated by means of alternative sequential executions obtained by interleaving the actions involved in the parallel execution. ■

### 7.2.1 Net places

The first step in the definition of the integrated operational net semantics consists of establishing a correspondence between net places and sequential terms, thereby inducing a correspondence between net markings and terms.

**Definition 7.4** The set \( V \) of places is generated by the following syntax

\[ V ::= \emptyset | <a, \lambda>, E \mid V/L \mid V \backslash H \mid V[\gamma] \mid V + V \mid V \mid s \ id \mid s \ id \mid s \ V \mid A \]

where \( L, S \subseteq \text{AType} - \{\tau\} \) and \( H \subseteq \text{AType} \). We use \( V, V', V'', \ldots \) as metavariables for \( V \), and \( Q, Q', Q'', \ldots \) as metavariables for \( \mathcal{M}_{fin}(V) \).

The main difference with respect to the syntax of EMPA terms (Definition 2.1) is that the binary operator “\( \parallel_s \)” has been replaced by the two unary operators “\( \parallel s \ id \)” and “\( \parallel s \langle \cdot \rangle \).” This is the means whereby it is possible to express the decomposition into sequential terms. Terms are then mapped onto places through the following function.

**Definition 7.5** The **decomposition function**

\[ \text{dec} : G \rightarrow \mathcal{M}_{fin}(V) \]

is defined by induction on the syntactical structure of the terms in \( G \) as follows:

- \( \text{dec}(\emptyset) = \emptyset \)
- \( \text{dec}(<a, \lambda>, E) = \emptyset <a, \lambda>, E \)
- \( \text{dec}(E/L) = \text{dec}(E) / L = \emptyset V/L \mid V \in \text{dec}(E) \)
- \( \text{dec}(E \backslash H) = \text{dec}(E) \backslash H = \emptyset V \backslash H \mid V \in \text{dec}(E) \)
- \( \text{dec}(E[\gamma]) = \text{dec}(E)[\gamma] = \emptyset V[\gamma] \mid V \in \text{dec}(E) \)
- \( \text{dec}(E_1 + E_2) = \text{dec}(E_1) + \text{dec}(E_2) = \emptyset V_1 + V_2 \mid V_1 \in \text{dec}(E_1) \wedge V_2 \in \text{dec}(E_2) \)
- \( \text{dec}(E_1 \parallel E_2) = \text{dec}(E_1) \parallel s \ id \parallel s \ \text{dec}(E_2) = \emptyset V \parallel s \ id \parallel V \in \text{dec}(E_1) \parallel s \ id \parallel s \ V \in \text{dec}(E_2) \)
- \( \text{dec}(A) = \text{dec}(E) \) if \( A \cong E \)

where \( Q \in \mathcal{M}_{fin}(V) \) is **complete** if and only if there exists \( E \in G \) such that \( \text{dec}(E) = Q \).

The decomposition function is well defined because we consider only guardedly closed terms. It is injective as well if we identify each constant with the right-hand side term of its defining equation, and it assigns place sets, rather than multisets, to terms. Note that the decomposition function embeds the syntactical structure of terms into places.
Example 7.6 If we consider term $E$ defined in Example 7.3, we have that
\[
\text{dec}(E) = \{ (\langle a, \lambda > \| \bullet \| \bullet \| \text{id} ) + (\langle \epsilon, \gamma > \| \langle \text{id} || \| \bullet ) + (\langle \text{id} || \| \bullet , \| \mu > \| \bullet ) + (\langle \epsilon, \gamma > \| \bullet \} \}
\]

7.2.2 Net transitions

The second step in the definition of the integrated operational net semantics consists of introducing an appropriate relation over net places whereby net transitions are constructed. Following the guideline of Section 3, we define the transition relation $\rightarrow$ as the least subset of $M_u_{fin}(V) \times \text{Act}^{u_{fin}(V)} \times M_u_{fin}(V)$ generated by the inference rule reported in the first part of Table 3, which in turn is based on the multiset $P\text{M}(Q)$ of potential moves of $Q \in M_u_{fin}(V)$ defined by structural induction as the least element of $M_u_{fin}(\text{Act}^{M_u_{fin}(V)} \times M_u_{fin}(V))$ generated by the rules reported in the second part of Table 3.

These rules are strictly related to the rules reported in Table 1 for the integrated operational interleaving semantics of EMPA terms. The major differences are listed below:

1. There are three rules for the alternative composition operator, instead of one. In the first two rules only a part of the sequential terms needs to have an alternative, and such a part is not complete whereas its alternative is complete. This guarantees that none of the sequential terms in the complete alternative has been previously involved in an execution, so the noncomplete alternative has not been discarded yet due to an action previously executed by a sequential term in the complete alternative (see Example 7.7).

2. There are three rules for the parallel composition operator, instead of one. This is a consequence of the distributed notion of state typical of nets (see Example 7.8).

3. There are no rules for constants. The treatment of constants has been already embodied in function $\text{dec}$ (see Definition 7.5).

4. Function $\text{Select}$ does not appear because it is both unnecessary, since the race policy is included in the net firing rule, and difficult to be implemented, due to the distributed notion of state (see Example 7.9).

5. Rate normalization is carried out through function $\text{norm} : (\text{Act} \times V \times \mathbb{N}_+ ) \rightarrow \text{Act}^{u\text{fin}(V)}$ defined in the third part of Table 3, where $V'$ is generated by the same syntax as $V$ except that $V + V$ is replaced by $V + \text{id}$ and $\text{id} + V$. In order to determine the correct rate of transitions deriving from the synchronization of the same active action with several passive actions of the same type that are either independent or alternative, function $\text{norm}$ considers for each transition three parameters: the basic action, the basic place and the passive contribution. The basic action is the action that will label the transition after the normalization of its rate. The basic place is the place contributing with the basic action to the transition (see Example 7.10). The passive contribution is the product of the number of alternative passive actions of places contributing to the transition with such actions (see Example 7.11). These three parameters are initialized by the rule for the prefix operator and then modified by the third rule for the parallel composition operator: the second parameter is modified by every rule. The normalizing factor for a given transition is the ratio of its passive contribution to the sum of the passive contributions of the enabled (see Example 7.12) transitions having the same basic action and the same basic place as the transition at hand.

Unlike function $\text{Norm}$, function $\text{norm}$ comes into play not only in the case of a synchronization. Again, this is a consequence of the distributed notion of state.

6. Potential move merging is carried out through functions $\text{md}1 : M_u_{fin}(\text{Act}^{M_u_{fin}(V)} \times M_u_{fin}(V)) \rightarrow P_{fin}(\text{Act}^{M_u_{fin}(V)} \times M_u_{fin}(V))$ and $\text{md}2 : P_{fin}(\text{Act}^{M_u_{fin}(V)} \times M_u_{fin}(V)) \rightarrow P_{fin}(\text{Act}^{M_u_{fin}(V)} \times M_u_{fin}(V))$ defined in the fourth part of Table 3. Function $\text{md}1$ merges the potential moves having the same basic action, the same basic place and the same postset by summing their passive contributions (see Example 7.11). Function $\text{md}2$ merges the potential moves having the same basic action type, the same priority level, the same passive contribution and the same postset by applying operation $\text{Min}$ to their basic action rates: since the basic places of these potential moves can differ only due to “$+$ id” or “$-$ id” operators, and since the basic place of the resulting potential move must be
\[
\frac{(\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q')) \in \text{melt}_2(\text{melt}_1(\text{PM}(Q))))}{Q \xrightarrow{\text{norm}(\langle a, \hat{\lambda}, V, f \rangle)} Q'}
\]

\[
\begin{align*}
\text{PM}(\{\langle a, \hat{\lambda}, E \rangle\}) = \{\langle (\text{norm}(\langle a, \hat{\lambda}, < a, \hat{\lambda}, E, 1 \rangle), \text{dec}(E)) \rangle \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}(Q \mid L) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V/L, f \rangle, Q'/L) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \wedge a \notin L) \} \oplus \\
&\{\langle (\text{norm}(\langle a, \hat{\lambda}, V/L, f \rangle, Q'/L) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \wedge a \in L \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}(Q \setminus H) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V\setminus H, f \rangle, Q'\setminus H) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \wedge \\
&\quad \neg(a \in H \wedge \hat{\lambda} = \ast) \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}(Q[\varphi]) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V[\varphi], f \rangle, Q'[\varphi]) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}((Q_1 + Q_2) \oplus Q_3) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V + id, f \rangle, Q') \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q_1 \oplus Q_3) \} \oplus \\
&\quad \{\langle (\text{norm}(\langle a, \hat{\lambda}, id + V, f \rangle, Q') \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q_2 \oplus Q_3) \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}(Q_1 + Q_2) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V + id, f \rangle, Q') \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q_1) \} \oplus \\
&\quad \{\langle (\text{norm}(\langle a, \hat{\lambda}, id + V, f \rangle, Q') \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q_2) \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}((Q_1 \oplus Q_2) \cap Q_3) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V \mid s \mid id, f \rangle, Q' \mid s \mid id) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \wedge a \notin S \} \\
&\quad \{\langle (\text{norm}(\langle a, \hat{\lambda}, id \mid s \mid V, f \rangle, Q' \mid s \mid id) \mid (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q') \in \text{PM}(Q) \wedge a \notin S \}
\end{align*}
\]

\[
\begin{align*}
\text{PM}(Q_1 \parallel s \mid id, Q_2) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V \mid s \mid id \oplus id \mid s \rangle Q_2) \mid \\
&\quad a \in S \wedge \min(\hat{\lambda}, \hat{\mu}) = \ast \wedge \\
&\quad (\text{norm}(\langle a, \hat{\lambda}, V, f_1 \rangle, \langle V', f' \rangle) \in \text{PM}(Q_1) \wedge \\
&\quad (\text{norm}(\langle a, \hat{\mu}, V', f_2 \rangle, Q_2) \in \text{PM}(Q_2) \wedge \\
&\quad (\hat{\lambda} = \hat{\mu} = \ast \wedge V \equiv V_1 \parallel s \mid id \wedge f = f_1 \cdot f_2) \wedge \\
&\quad (\lambda \in \mathbb{R}_+ \cup \text{Inf} \wedge V \equiv V_1 \parallel s \mid id \wedge f = f_2) \wedge \\
&\quad (\mu \in \mathbb{R}_+ \cup \text{Inf} \wedge V \equiv V_1 \parallel s \mid id \wedge V \equiv V_2 \parallel f = f_1)) \}
\end{align*}
\]

\[
\text{norm}(\langle a, \hat{\lambda}, V, f \rangle) = \langle a, \text{Split}(\hat{\lambda}, f / \sum f' \mid Q_1 \xrightarrow{\text{norm}(\langle a, \hat{\lambda}, V, f' \rangle)} Q_2 \cap Q_1 \subseteq \text{Mcurr} \}
\]

\[
\begin{align*}
\text{melt}_1(\text{PM}) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q) \mid \\
&\quad (\text{norm}(\langle a, \hat{\lambda}, V, f' \rangle, Q) \in \text{PM} \wedge \\
&\quad f = \sum \| f'' \mid (\text{norm}(\langle a, \hat{\lambda}, V, f'' \rangle, Q) \in \text{PM} \}
\}\end{align*}
\]

\[
\begin{align*}
\text{melt}_2(\text{PM}) &= \{\langle (\text{norm}(\langle a, \hat{\lambda}, V, f \rangle, Q) \mid \\
&\quad (\text{norm}(\langle a, \hat{\mu}, V, f \rangle, Q) \in \text{PM} \wedge \\
&\quad \hat{\lambda} = \text{Min} \| \hat{\gamma} \mid (\text{norm}(\langle a, \hat{\gamma}, V''', f \rangle, Q) \in \text{PM} \wedge \text{PL}(\langle a, \hat{\gamma} \rangle) = \text{PL}(\langle a, \hat{\mu} \rangle) \wedge \\
&\quad V = \text{inner}_{+ \mid \id}(V''' \mid (\text{norm}(\langle a, \hat{\gamma}, V''', f \rangle, Q) \in \text{PM} \wedge \text{PL}(\langle a, \hat{\gamma} \rangle) = \text{PL}(\langle a, \hat{\mu} \rangle)) \}
\end{align*}
\]

Table 3. Inductive rules for EMPA integrated location-oriented net semantics
uniquely defined in order for function \( \text{norm} \) to work correctly, the choice is made by taking
the basic place having the innermost “\(+ id\)” operator (see Example 7.13).
The differences above are clarified by the following examples.

**Example 7.7** If we consider term \( E \) defined in Example 7.3 and the places in its decomposition
shown in Example 7.6, we obtain the following transitions
\[
\begin{align*}
\{ (<a, \lambda>, \emptyset || id) &+ <c, \gamma>, \emptyset \} \quad \xrightarrow{\text{norm}(<c, \gamma>, \emptyset || id + id, 1)} \quad \{ \emptyset || id \} \\
\{ (id || <b, \mu>, \emptyset) + <c, \gamma>, \emptyset \} \quad \xrightarrow{\text{norm}(<c, \gamma>, id + <c, \gamma>, \emptyset || id + id, 1)} \quad \{ id || \emptyset \}
\end{align*}
\]
\[\text{dec}(E)\]

Note that if \( \text{dec}(E) \) is the current marking, then all the transitions above are enabled and firing
the first transition results in the marking \( \{ \emptyset || id, (id || <b, \mu>, \emptyset) + <c, \gamma>, \emptyset \} \) which cannot be
the preset of any transition labeled with action type \( c \), because the alternative \( id || <b, \mu>, \emptyset \) of
\( <c, \gamma>, \emptyset \) is not complete. This is consistent with the definition of \( E \): the execution of either \( <a, \lambda> \)
or \( <b, \mu> \) prevents \( <c, \gamma> \) from being executed.

Let us now slightly modify term \( E \) in the following way
\[
E' \equiv (<a, \lambda>, <b, \ast, \emptyset || (\emptyset) <b, \mu>, \emptyset) + <c, \gamma>, \emptyset
\]
where
\[\text{dec}(E') = \{ (<a, \lambda>, <b, \ast, \emptyset || (\emptyset) id) + <c, \gamma>, \emptyset (id || (\emptyset) <b, \mu>, \emptyset) + <c, \gamma>, \emptyset \}\]
By applying the rules in Table 3, we get the two transitions
\[
\begin{align*}
\{ (<a, \lambda>, <b, \ast, \emptyset || (\emptyset) id) + <c, \gamma>, \emptyset \} \quad \xrightarrow{\text{norm}(<c, \gamma>, \emptyset || id + id, 1)} \quad \{ <b, \ast, \emptyset || (\emptyset) id \} \\
\{ <b, \ast, \emptyset || (\emptyset) id \} \quad \xrightarrow{\text{norm}(<c, \gamma>, id + <c, \gamma>, \emptyset || id + id, 1)} \quad \{ \emptyset \}
\end{align*}
\]
\[\text{dec}(E')\]

If \( \text{dec}(E') \) is the current marking, then all the transitions above are enabled and firing the first
transition results in marking \( \{ <b, \ast, \emptyset || (\emptyset) id, (id || (\emptyset) <b, \mu>, \emptyset) + <c, \gamma>, \emptyset \} \) which is the preset
of the following transition
\[
\{ <b, \ast, \emptyset || (\emptyset) id, (id || (\emptyset) <b, \mu>, \emptyset) + <c, \gamma>, \emptyset \} \quad \xrightarrow{\text{norm}(b, \mu, (id || (\emptyset) <b, \mu>, \emptyset) + id + id, 1)} \quad \{ \emptyset || (\emptyset) id, id \}
\]
This example motivates the presence of \( Q_3 \) in the first two rules for the alternative composition
operator: if \( Q_3 \) were not taken into account, then the transition above would not be constructed.

**Example 7.8** Consider term
\[
E \equiv <a, \lambda>, \emptyset || b, \bar{\mu}, \emptyset
\]
whose decomposition is given by
\[\text{dec}(E) = \{ <a, \lambda>, \emptyset || id, id || b, \bar{\mu}, \emptyset \}\]
By applying the rules in Table 3, we get the two independent transitions
\[
\begin{align*}
\{ <a, \lambda>, \emptyset || id \} \quad \xrightarrow{\text{norm}(a, \lambda, \emptyset || id + id, 1)} \quad \{ \emptyset || id \} \\
\{ id || b, \bar{\mu}, \emptyset \} \quad \xrightarrow{\text{norm}(b, \bar{\mu}, (id || \emptyset) + id, 1)} \quad \{ id || \emptyset \}
\end{align*}
\]
as expected. If we replaced the three rules for the parallel composition operator with a single
rule similar to the one reported in Table 1, then we would get instead the two alternative transitions
\[
\begin{align*}
\text{dec}(E) \quad \xrightarrow{\text{norm}(a, \lambda, \emptyset || id + id, 1)} \quad \{ \emptyset || id, id || b, \bar{\mu}, \emptyset \} \\
\text{dec}(E) \quad \xrightarrow{\text{norm}(b, \bar{\mu}, (id || \emptyset) + id, 1)} \quad \{ <a, \lambda>, \emptyset || id, id || b, \bar{\mu}, \emptyset \}
\end{align*}
\]
which are not consistent with the fact that the two subterms of \( E \) are independent, thereby
resulting in a violation of the concurrency principle (see Section 7.4).

UBLCS-95-14 53
Example 7.9 Function \textit{Select} does not appear in the rules of Table 3. Even if we introduced it before applying function \textit{melt} in order to rule out lower priority transitions, we would not be able to capture all the cases. Consider for instance term

$$E \equiv \langle a, \lambda, \rho + \langle c, \omega_1, \mu + \langle c, \rho, \omega \rangle \rangle$$

whose decomposition comprises places \(V_1 \parallel [c] \text{id} \text{ and } id \parallel [c] V_2\) where

\[
\begin{align*}
V_1 & \equiv \langle a, \lambda, \rho + \langle c, \omega_1, \mu \rangle \rangle \\
V_2 & \equiv \langle b, \mu, \rho + \langle c, \rho, \omega \rangle \rangle
\end{align*}
\]

By applying the first rule for the parallel composition operator to \(\parallel \text{id} [c] V_2\) we get one transition labeled with \(\text{norm}(\langle a, \lambda, \rho + \langle c, \omega_1, \mu \rangle \rangle, \langle a, \lambda, \rho + \langle c, \omega_1, \mu \rangle \rangle, \text{id}, 1)\). By applying the second rule to \(\parallel \text{id} [c] V_2\) we get one transition labeled with \(\text{norm}(\langle a, \rho + \langle c, \rho, \omega \rangle \rangle, \text{id}, [c] V_2, \text{id}, 1)\). Furthermore, by applying the third rule to \(\parallel \text{id} [c] V_2\) we get one transition labeled with \(\text{norm}(\langle a, \omega_1, \mu \rangle, \text{id} + <c, \omega_1, \mu \rangle, [c] \text{id}, 1)\). The third transition prevents both the first one and the second one from firing, but this could not be caught by function \textit{Select} because the three transitions have different presets.

Example 7.10 Consider term

$$E \equiv \langle a, \lambda, \rho \rangle \parallel [a] \langle a, \rho, \omega \rangle + \langle a, \lambda, \rho \rangle \parallel [a] \langle a, \rho, \omega \rangle$$

whose decomposition comprises places \((V_1 \parallel [a] \text{id}) \parallel (V_1 \parallel [a] \text{id})\), \(V_2 \parallel [a] \text{id} \parallel id \parallel [a] V_2\) and \(id \parallel [a] V_2\) where

\[
\begin{align*}
V_1 & \equiv \langle a, \lambda, \rho \rangle \\
V_2 & \equiv \langle a, \rho, \omega \rangle + \langle a, \rho, \omega \rangle
\end{align*}
\]

By applying the rules in Table 3, we get the following two transitions

\[
\begin{align*}
dec(E) & \quad \frac{\text{norm}(\langle a, \lambda, \rho \rangle, \text{id} \parallel id \parallel [a] \text{id}, 1)}{[a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id}} \\
dec(E) & \quad \frac{\text{norm}(\langle a, \lambda, \rho \rangle, \text{id} + [a] \text{id} \parallel [a] \text{id}, 1)}{[a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id}}
\end{align*}
\]

If \(\text{dec}(E)\) is the current marking, then both transitions are enabled and the normalizing factor is 1 for both transitions, as expected. This example motivates the use of \(V'\) instead of \(V\) for expressing the basic place: if \(V\) were used, then the two transitions above would have the same basic place (beyond the same basic action), so they would be given the wrong normalizing factor 1/2 by function \textit{norm}.

Example 7.11 Consider term

$$E \equiv \langle a, \lambda, \rho \rangle \parallel [a] (\langle a, \rho, \omega \rangle + \langle a, \rho, \omega \rangle)$$

whose decomposition comprises places \(V_1 \parallel [a] \text{id} \parallel [a] (V_2 \parallel id)\) and \(id \parallel [a] (id) \parallel V_5\) where

\[
\begin{align*}
V_1 & \equiv \langle a, \lambda, \rho \rangle \\
V_2 & \equiv \langle a, \rho, \omega \rangle + \langle a, \rho, \omega \rangle \\
V_3 & \equiv \langle a, \rho, \omega \rangle
\end{align*}
\]

By applying the rules in Table 3, we get the following two transitions

\[
\begin{align*}
\parallel \text{id} V_1 \parallel [a] \text{id} \parallel [a] (V_2 \parallel id) & \quad \frac{\text{norm}(\langle a, \lambda, \rho \rangle, \text{id} \parallel id \parallel [a] \text{id}, 2)}{[a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id} \parallel V_5} \\
\parallel \text{id} V_1 \parallel [a] \text{id} \parallel id \parallel [a] V_5 & \quad \frac{\text{norm}(\langle a, \lambda, \rho \rangle, \text{id} \parallel id \parallel [a] \text{id}, 1)}{[a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id} \parallel [a] \text{id}}
\end{align*}
\]

where value 2 for the passive contribution of the first transition is determined by function \textit{melt}. If \(\text{dec}(E)\) is the current marking, then both transitions are enabled and the normalizing factor is 2/3 for the first transition, and 1/3 for the second transition, as expected. This example motivates the use of passive contributions: if the normalizing factor were computed as the inverse of the number of enabled transitions having the same basic action and the same basic place as the transition at hand, then we would obtain the wrong normalizing factor 1/2 for the two transitions above.
Example 7.12  Consider term 
\[ E \equiv \langle a, \lambda \rangle \cdot \langle a, \ast \rangle \cdot \langle a, \ast \rangle \cdot \emptyset \]
whose decomposition comprises places
\[ V_1, V_2, V_3 \]
and if also the third transition were taken into account, then we would obtain the wrong normalizing factor 1/2 as expected. This example motivates the use of marking-dependent rates: if also the third transition were taken into account, then we would obtain the wrong normalizing factor 1/3 for the first two transitions.

Example 7.13  Consider term 
\[ E \equiv (\langle a, \lambda \rangle \cdot \langle a, \ast \rangle \cdot \emptyset + \langle a, \ast \rangle \cdot \emptyset \cdot \langle a, \ast \rangle \cdot \emptyset) \]
whose decomposition comprises places
\[ V_1, V_2, V_3 \]
By applying the rules in Table 3, we get the following two transitions
\[ \text{Dec}(E) \]
Example 7.12

7.2.3 Nets associated with terms
The third step in the definition of the integrated operational net semantics consists of associating with each term an appropriate PGSPN by exploiting the previous two steps.

Definition 7.14  The integrated operational location-oriented net semantics of a term \( E \in G \) is the PGSPN 
\[ \mathcal{N}_{lo}(E) = (P, U, T, M_0, L, W) \]
where:
- \( P \) is the least subset of \( V \) such that:
  - \( \text{dom}(\text{Dec}(E)) \subseteq P \);
  - \( \text{dom}(Q_1) \subseteq P \) and \( \text{Q}_1 \rightarrow Q_2 \), then \( \text{dom}(Q_2) \subseteq P \);
- \( U = \text{Act}^\text{M}_{\text{fin}}(P) \);
- \( T \) is the restriction of \( T : \text{M}_{\text{fin}}(P) \times \text{Act}^\text{M}_{\text{fin}}(P) \times \text{M}_{\text{fin}}(P) \);
- \( M_0 = \text{Dec}(E) \);
- \( L : T \rightarrow \text{PLSet} \) such that:
  - \( L(Q_1, \text{dom}(\langle a, \ast \rangle \cdot \emptyset, V, f), Q_2) = -1 \);
\( L(Q_1, \text{norm}(<a, \lambda>, V, f), Q_2) = 0; \)
\( L(Q_1, \text{norm}(<a, \infty_{t,w}>, V, f), Q_2) = 1; \)

\[ W : T \longrightarrow (\{\ast\} \cup \mathbb{R}_{\text{Misp}}^{\ast}(P)) \text{ such that:} \]
\( W(Q_1, \text{norm}(<a, \ast>, V, f), Q_2) = \ast; \)
\( W(Q_1, \text{norm}(<a, \lambda>, V, f), Q_2) = \lambda' \text{ if } \text{norm}(<a, \lambda>, V, f) = <a, \lambda'>; \)
\( W(Q_1, \text{norm}(<a, \infty_{t,w}>, V, f), Q_2) = w' \text{ if } \text{norm}(<a, \infty_{t,w}>, V, f) = <a, \infty_{t,w}'>. \]

---

**Example 7.15** If we take term \( E \) introduced in Example 7.3, we have that the PGSPN below represents \( N_{\text{loc}}[E] \):

For the sake of simplicity, transition labels already take into account the current marking. Unlike \( T[E] \), in this model it is clear that actions with type \( a \) and \( b \) are independent. It is easy to verify that \( RG[N_{\text{loc}}[E]] \) is isomorphic to \( T[E] \).

Now we show two properties of the integrated operational net semantics, which can be demonstrated with a proof similar to that provided in [55].

**Theorem 7.16** Let \( E \in \mathcal{G} \).

(i) \( N_{\text{loc}}[E] \) is safe, i.e. every marking reachable from the initial one is a set.

(ii) \( N_{\text{loc}}[E] \) is finite if each subterm of \( E \) of the form \( E'/L, E'\backslash H, E'[\varphi], E_1 \parallel E_2 \) is without constants.

It is interesting to identify a class of terms in \( \mathcal{G} \) such that for each term \( E \) in this class it turns out that \( N_{\text{loc}}[E] \) is a GSPN. As we can obviously expect, the above class is given by \( \mathcal{E} \) and this will be proved later.

### 7.3 Retrievability principles

In this section we assess the soundness of the integrated operational net semantics with respect to the integrated operational interleaving semantics. To this aim, we adapt the proposal in [55] to our stochastic framework by resorting to the following two principles:

- **Functional retrievability principle:** the functional semantics of each term should be retrievable from its integrated operational net semantics. Such a principle can be formalized by requiring that, for each term, its functional semantics is isomorphic or bisimilar to the functional semantics of its integrated operational net semantics.

- **Performance retrievability principle:** the performance semantics of each term should be retrievable from its integrated operational net semantics. Such a principle can be formalized by requiring that, for each term, its performance semantics is \( p \)-isomorphic or \( p \)-bisimilar to the performance semantics of its integrated operational net semantics.

These two principles guarantee that each term and its integrated operational net semantics describe the same system both from the functional point of view and from the performance point of view.

**Theorem 7.17** Let \( E \in \mathcal{G} \). Then \( RG[N_{\text{loc}}[E]] \) is isomorphic to \( T[E] \).

**Proof** The proof is divided into three parts.
(1st part) Suppose that priority levels are taken into account neither in EMPA nor in PGSPNs. More accurately, assume that the active transitions of PGSPNs are not divided into different priority levels, and consider $T'[E]$ instead of $T[E]$, i.e. consider the LTS (whose set of states is denoted by $\uparrow E$) representing the integrated operational interleaving semantics of $E$ if function Select were not applied. Then we can demonstrate, by following the proof developed in [55] Theorem 3.7.18, that $RG[N_{loc}[E]]$ is bisimilar to $T'[E]$ through relation

$$B = \{(F, Q) \in \uparrow E \times R(\text{dec}(E)) \mid Q \text{ swf} \land \text{dec}(F) = \text{upd}(Q)\}$$

where:

- The definition of strongly well formed (swf) marking is the following:
  - $\downarrow \{0\}$ and $\downarrow \{<a, \lambda>.E\}$ are swf;
  - if $Q$ is swf, then so are $Q/L, Q\setminus H$ and $Q[\varphi]$;
  - if $Q_1 \oplus Q_3$ is swf, $\text{dom}(Q_1) \cap \text{dom}(Q_3) = \emptyset$, either $Q_3 = \emptyset$ or not all components in $Q_3$ contain “$+$” as their topmost operator, and $Q_2$ is complete, then $(Q_1 + Q_2) \oplus Q_3$ and $(Q_2 + Q_1) \oplus Q_3$ are swf;
  - if $Q_1$ and $Q_2$ are swf, then so is $Q_1 \|_s id \cup id \|_s Q_2$.

This property is satisfied by complete elements of $\mathcal{M}_{upd}(V)$ and is invariant for transition firing.

- The definition of the update operation (upd) on swf markings is the following:
  - if $Q$ is complete, then $\text{upd}(Q) = Q$;
  - if $Q \equiv Q'/L$ is incomplete, then $\text{upd}(Q) = \text{upd}(Q')/L$;
  - if $Q \equiv Q' \setminus H$ is incomplete, then $\text{upd}(Q) = \text{upd}(Q' \setminus H)$;
  - if $Q \equiv Q'[\varphi]$ is incomplete, then $\text{upd}(Q) = \text{upd}(Q')[\varphi]$;
  - if $Q \equiv (Q_1 + Q_2) \oplus Q_3$ or $Q \equiv (Q_2 + Q_1) \oplus Q_3$ is incomplete, and $Q_2$ is complete, then $\text{upd}(Q) = \text{upd}(Q_1 + Q_3)$;
  - if $Q \equiv Q_1 \|_s id \cup id \|_s Q_2$ is incomplete, then $\text{upd}(Q) = \text{upd}(Q_1) \|_s id \cup id \|_s \text{upd}(Q_2)$. For each swf marking $Q$, it turns out that $\text{upd}(Q)$ is complete.

(2nd part) Now we want to prove, under the same assumption made at the beginning of the previous part, that bisimulation $B$ is actually an isomorphism between $RG[N_{loc}[E]]$ and $T'[E]$.

Firstly, we have to prove that $B$ is a function. Given $F \in \uparrow E$, since $F$ is reachable from $E$ and $B$ is a bisimulation, there must exist $Q \in R(\text{dec}(E))$ such that $(F, Q) \in B$, i.e. $\text{dec}(F) = \text{upd}(Q)$. It remains to prove the uniqueness of such a swf reachable marking $Q$. Suppose that there exist $Q_1, Q_2 \in R(\text{dec}(E))$ swf and different from each other such that $\text{upd}(Q_1) = \text{upd}(Q_2) = \text{dec}(F)$. This can stem only from the fact that there exists at least a pair composed of a subterm $G$ of a place $V_1$ in $Q_1$ and a subterm $G' \setminus H$ of a place $V_2$ in $Q_2$ that reside in the same position of the syntactical structure of $V_1$ and $V_2$ (if such a pair did not exist, $Q_1$ and $Q_2$ could not be different from each other). The existence of this pair contradicts the reachability of $Q_1$. In fact, we recall that the decomposition function $\text{dec}$ distributes all the alternative composition operators between all the appropriate places and when one of these places is part of a marking involved in a transition firing, either it remains unchanged or it gives rise to a new place where the alternative composition operator disappears and only the alternative involved remains after it has been transformed (see the rules for the alternative composition operator).

Secondly, we have to prove that $B$ is injective. This is trivial, because if there exist $F_1, F_2 \in \uparrow E$ and $Q \in R(\text{dec}(E))$ such that $\text{dec}(F_1) = \text{upd}(Q) = \text{dec}(F_2)$, then necessarily $F_1 = F_2$ as $\text{dec}$ is injective.

Thirdly, we have to prove that $B$ is surjective. This is true because given $Q \in R(\text{dec}(E))$, since $Q$ is reachable from $\text{dec}(E)$ and $B$ is a bisimulation, there must exist $F \in \uparrow E$ such that $(F, Q) \in B$.

Finally, we have to prove that $B$ satisfies the isomorphism clauses. This follows immediately from the fact that $B$ is a bijection fulfilling the bisimilarity clauses.

(3rd part) Now let us take into account the priority levels. Since the priority mechanism for EMPA actions is exactly the same as the priority mechanism for PGSPN transitions, from the previous step it follows that $RG[N_{loc}[E]]$ is isomorphic to $T[E]$.


UBLCS-95-14 57
Corollary 7.18 Let \( E \in \mathcal{G} \). Then \( \mathcal{F}[\mathcal{N}_{\text{loc}}[E]] \) is isomorphic to \( \mathcal{F}[E] \).

Corollary 7.19 Let \( E \in \mathcal{E} \). Then \( \mathcal{M}[\mathcal{N}_{\text{loc}}[E]] \) is p-isomorphic to \( \mathcal{M}[E] \).

From retrievability, the following result immediately follows.

Theorem 7.20 Let \( E \in \mathcal{G} \). Then \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN if and only if \( E \in \mathcal{E} \).

Proof (\( \Rightarrow \)) Suppose that \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN, i.e. suppose that \( \mathcal{N}_{\text{loc}}[E] \) has no passive transitions. Then \( \mathcal{R}_G[\mathcal{N}_{\text{loc}}[E]] \) has no passive transitions hence, by virtue of Theorem 7.17, \( \mathcal{I}[E] \) has no passive transitions. Thus \( E \in \mathcal{E} \).

(\( \Leftarrow \)) Suppose that \( E \in \mathcal{E} \), i.e. suppose that \( \mathcal{I}[E] \) has no passive transitions; we prove that \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN by proceeding by induction on the syntactical structure of \( E \):

- If \( E \equiv \emptyset \) then \( \mathcal{N}_{\text{loc}}[E] \) is obviously a GSPN.
- Let \( E \equiv \langle a, \lambda \rangle . E' \). From \( E \in \mathcal{E} \) it follows that \( \hat{\lambda} \neq \star \) and \( E' \in \mathcal{E} \), so by the induction hypothesis we have that \( \mathcal{N}_{\text{loc}}[E'] \) is a GSPN hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN too.
- Let \( E \equiv E' / L \). From \( E \in \mathcal{E} \) it follows that \( E' \in \mathcal{E} \), so by the induction hypothesis we have that \( \mathcal{N}_{\text{loc}}[E'] \) is a GSPN hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN too.
- Let \( E \equiv E' \setminus H \). There are two cases:
  - If \( E' \in \mathcal{E} \) then, by the induction hypothesis, we have that \( \mathcal{N}_{\text{loc}}[E'] = \mathcal{N}_{\text{loc}}[E] \) is a GSPN.
  - If \( E' \notin \mathcal{E} \) then \( E' \) can execute some passive actions which, due to the fact that \( E \in \mathcal{E} \), have type in \( H \). By the rule for the temporal restriction operator, the passive transitions in \( \mathcal{N}_{\text{loc}}[E'] \) cannot be present in \( \mathcal{N}_{\text{loc}}[E] \); hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN.
- Let \( E \equiv E' [ \varphi ] \). From \( E \in \mathcal{E} \) it follows that \( E' \in \mathcal{E} \), so by the induction hypothesis we have that \( \mathcal{N}_{\text{loc}}[E'] \) is a GSPN hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN too.
- Let \( E \equiv E_1 + E_2 \). From \( E \in \mathcal{E} \) it follows that \( E_1 \in \mathcal{E} \) and \( E_2 \in \mathcal{E} \), so by the induction hypothesis we have that \( \mathcal{N}_{\text{loc}}[E_1] \) and \( \mathcal{N}_{\text{loc}}[E_2] \) are two GSPNs hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN too.
- Let \( E \equiv E_1 || E_2 \). There are two cases:
  - If \( E_1 \in \mathcal{E} \land E_2 \in \mathcal{E} \) then, by the induction hypothesis, we have that \( \mathcal{N}_{\text{loc}}[E_1] \) and \( \mathcal{N}_{\text{loc}}[E_2] \) are two GSPNs hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN too.
  - If \( E_1 \notin \mathcal{E} \lor E_2 \notin \mathcal{E} \) then \( E_1 \) or \( E_2 \) can execute some passive actions which, due to the fact that \( E \in \mathcal{E} \), have types in \( S \) and either do not synchronize at all or synchronize with active actions of the same type present in the other subterm. By the rules for the parallel composition operator, the passive transitions present in \( \mathcal{N}_{\text{loc}}[E_1] \) or in \( \mathcal{N}_{\text{loc}}[E_2] \) cannot be present in \( \mathcal{N}_{\text{loc}}[E] \); hence \( \mathcal{N}_{\text{loc}}[E] \) is a GSPN.

7.4 Concurrency principle

In this section we assess the completeness of the integrated operational net semantics by resorting to the concurrency principle[55], which requires that the intended concurrency of each term should be represented by its integrated operational net semantics. The introduction of this principle is due to the fact that retrievability deals only with individual transitions so it does not reject net semantics exhibiting too little concurrency.

To formalize the concurrency principle, we adapt to our stochastic framework some standard operators on nets generally accepted as representing the intended concurrency of terms. In other words, following a standard practice (see, e.g., [55]), we develop an integrated denotational net semantics for EMPA and then we investigate whether the integrated operational net semantics admits the same concurrent computations as the integrated denotational net semantics.

The operators on safe PGSPNs with no inhibitor arcs are presented below: the definitions of the set of labels, the priority function and the weight function for the resulting net of each operator are omitted because they are similar to those reported in Definition 7.14.

- \( \emptyset = (\{p\}, \emptyset, 0, \emptyset, 0, \emptyset) \);
- \( \langle a, \lambda, >, (P, U, T, M_0, I, W) = (P', U', T', M'_0, I', W') \) where:
Consider now term to which it is applied. Root unwinding allows the correct interplay of alternative composition and unwinding which ensures that there are no cycles left at initially marked places; it then uses the composition one. It combines the standard alternative composition operator with the idea of root unwinding.

\[
\begin{align*}
\text{Example 7.21} & \quad \text{Consider terms} \\
A & \triangleq \langle a, \lambda \rangle . A \\
B & \triangleq \langle b, \mu \rangle . B
\end{align*}
\]

The integrated denotational net semantics of $A$ is a net with one place $p_A$ and one transition

\[
\{p_A\} \xrightarrow{\text{norm}_{\langle a, \lambda \rangle . A}} \{p_A\}
\]

The integrated denotational net semantics of $B$ is a net with one place $p_B$ and one transition

\[
\{p_B\} \xrightarrow{\text{norm}_{\langle b, \mu \rangle . B}} \{p_B\}
\]

Consider now term

\[
E \equiv A + B
\]

If we used the Cartesian product construction without root unwinding, then the integrated denotational net semantics of $E$ would be a net with one place $(p_A, p_B)$ and two transitions.
This net is not the right integrated denotational net semantics of \( E \) since \( E \) can perform either infinitely many actions \( <a, \lambda> \) or infinitely many actions \( <b, \mu> \), whereas the net above allows the two different actions to be arbitrarily interleaved. 

Function \( \text{norm}_a \) plays the same role as function \( \text{norm} \). The main difference between them is that function \( \text{norm}_a \) does not consider passive contributions. This is due to the fact that the integrated denotational net semantics generates a new place whenever 0 or a prefix operator is encountered. As a consequence, it is not possible that two or more transitions constructed by the alternative composition operator with the same preset, have the same postset. This is reflected by the definition of the normalizing factor: it is simply the inverse of the number of enabled transitions having the same basic actions and the same basic place as the transition at hand.

**Example 7.22** Consider term \( E \) of Example 7.11 and let \( E_1 \equiv V_1, E_2 \equiv V_2, E_3 \equiv V_3 \). The integrated denotational net semantics of \( E_1 \) is a net with two places \( p_{1,1}, p_{1,2} \) and one transition

\[
\begin{align*}
\{p_{1,1}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{1,1})} \{p_{1,2}\}
\end{align*}
\]

The integrated denotational net semantics of \( E_2 \) is a net with three places \( (p_{2,1}, p_{2,2}), p_{2,3}, p_{2,4} \) and two transitions

\[
\begin{align*}
\{p_{2,1}, p_{2,2}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{2,1})} \{p_{2,2}\} \\
\{p_{2,1}, p_{2,2}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{2,2})} \{p_{2,4}\}
\end{align*}
\]

The integrated denotational net semantics of \( E_3 \) is a net with two places \( p_{3,1}, p_{3,2} \) and one transition

\[
\begin{align*}
\{p_{3,1}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{3,1})} \{p_{3,2}\}
\end{align*}
\]

Finally, the integrated denotational net semantics of \( E \) is a net having the same places as the previous nets and three transitions

\[
\begin{align*}
\{p_{1,1}, p_{2,1}, p_{2,2}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{1,1})} \{p_{1,2}, p_{2,3}\} \\
\{p_{1,1}, p_{2,1}, p_{2,2}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{1,1})} \{p_{1,2}, p_{2,4}\} \\
\{p_{1,1}, p_{3,1}\} & \xrightarrow{\text{norm}_a(a, \lambda, p_{3,1})} \{p_{1,2}, p_{3,2}\}
\end{align*}
\]

In the initial marking all the transitions above are enabled, and their normalizing factor is 1/3 as expected. This example motivates the fact that passive contributions are unnecessary.

Using the notion of place-based bisimilarity (pl-bisimilarity) on safe nets of [55] Definition 2.3.8, provided that it is properly modified in order to take into account conditional exit rates as in Definition 6.6, and following a demonstration similar to that of [55] Theorem 3.8.3, we can now prove that for each \( n \)-ary operator \( op \) we have that \( N_{loc}[\{op_{MEPA}(E_1, \ldots, E_n)\}] \) is pl-bisimilar to \( op_{PSGPN}(N_{loc}[E_1], \ldots, N_{loc}[E_n]) \). By virtue of [55] Theorem 2.3.10, this means that the two nets have the same causal semantics, i.e. they have the same concurrent computations.

**Theorem 7.23** It turns out that:

(i) For every \( E \in \mathcal{G} \) and \( <a, \lambda> \in \text{Act}, N_{loc}[E] \) is pl-bisimilar to \( <a, \lambda> N_{loc}[E] \).

(ii) For every \( E \in \mathcal{G} \) and \( L \subseteq \text{Atype} - \{\tau\} \), \( N_{loc}[E/L] \) is pl-bisimilar to \( N_{loc}[E]/L \).

(iii) For every \( E \in \mathcal{G} \) and \( H \subseteq \text{Atype}, N_{loc}[E/H] \) is pl-bisimilar to \( N_{loc}[E]/H \).

(iv) For every \( E \in \mathcal{G} \) and \( \varphi \in \text{Relab}, N_{loc}[E]\{\varphi\} \) is pl-bisimilar to \( N_{loc}[E]\{\varphi\} \).

(v) For every \( E_1, E_2 \in \mathcal{G} \), \( N_{loc}[E_1 + E_2] \) is pl-bisimilar to \( N_{loc}[E_1] + N_{loc}[E_2] \).

(vi) For every \( E_1, E_2 \in \mathcal{G} \) and \( S \subseteq \text{Atype} - \{\tau\} \), \( N_{loc}[E_1|S E_2] \) is pl-bisimilar to \( N_{loc}[E_1]|S N_{loc}[E_2] \).
7.5 Net semantics of queueing system description

In this section we give an example of net semantics construction based on QSs. Let us consider a QS $M = M_{1}/M_{2}/M_{3}/M_{4}$ with arrival rate $\lambda$ and service rate $\mu$: we recall that such a QS has two independent servers, a queue with one seat, and four independent customers. Once denoted the action type “a customer leaves the service center” by $l$, its fine grain representation is the following:

- $C_4 \triangleq C||C||C||C||C$;
- $Q \triangleq <a, \lambda>.<l, \mu>.Q$;
- $S \triangleq <d, \infty, 1, q, \infty, 1, 1, \infty>.S$.

Its integrated operational net semantics $N_{lo} = [System_{M/M/2/3/4}]$ is the GSPN reported in Figure 10, where the following shorthands have been used:

- $p_1 = ((id||C||id)||id||id)||id||id$;
- $p_2 = ((id||id||id)||id||id)||id||id$;
- $p_3 = (id||id||id)||id||id||id$;
- $p_4 = (id||id||id)||id||id||id$;
- $p_5 = ((<l, \mu>).<d, \infty, 1, q, \infty, 1, 1, \infty>.S$;
- $p_6 = ((id||id||id)||id||id)||id||id||id$;
- $p_7 = (id||id||id)||id||id||id||id$;
- $p_8 = (id||id||id)||id||id||id||id$;

- $p_9 = id||id||id||id||id$;
- $p_{10} = id||id||id||id||id$;
- $p_{11} = id||id||id||id||id||id||id$;
- $p_{12} = id||id||id||id||id||id||id$.
vantage from syntactical compositionality, we figure out to deal with three interacting entities: 

Sender and Receiver, and the performance part, we have to remember that EMPA allows us to express only exponentially distributed

Furthermore, there are action types describing activities local to the single entities. Local activities of Sender are described by action types gm standing for “generate message”, and to standing for “timeout”. The only local activity of Receiver is described by action type cm standing for “consume message”. The local activities of Channel, which is composed of two independent half-duplex lines Line_m and Line_a, are described by action types pm standing for “propagate message tagged with 0”, pm standing for “propagate message tagged with 1”, pa standing for “propagate acknowledgement tagged with 0”, and pa standing for “propagate acknowledgement tagged with 1”. Additionally, there are other two activities local to Channel that are described by action type τ and represent the fact that a message or an acknowledgement gets lost or not.

So far we have considered the functional part of the protocol. Concerning the performance part, we have to remember that EMPA allows us to express only exponentially distributed

The alternating bit protocol

In this section we illustrate an application of the integrated approach proposed in Section 1 to the alternating bit protocol. The protocol is modeled by means of an EMPA term, and then analyzed by studying the semantic models associated with the term. 12

The alternating bit protocol [12] is a data-link level communication protocol that establishes a means whereby two stations, one acting as a sender and the other acting as a receiver, connected by a full-duplex communication channel that may lose messages, can cope with message loss. The name of the protocol stems from the fact that each message is augmented with an additional bit: since consecutive messages that are not lost are tagged with additional bits that are pairwise complementary, it is easy to distinguish between an original message and its possible duplicates. Initially, if the sender obtains a message from the upper level, it augments the message with an additional bit set to 0, sends the tagged message to the receiver, and starts a timer: if an acknowledgement tagged with 0 is received before the timeout expires, then the subsequent message obtained from the upper level will be sent with an additional bit set to 1, otherwise the current tagged message is sent again. On the other side, the receiver waits for a message tagged with 0: if it receives such a tagged message for the first time, then it passes the message to the upper level, sends an acknowledgement tagged with 0 to the sender, and waits for a message tagged with 1, whereas if it receives a duplicate tagged message (due to message loss, acknowledgement loss, or propagation taking an arbitrarily long time), then it sends an acknowledgement tagged with the same additional bit to the sender.

How to model the alternating bit protocol with EMPA? Since it is helpful to take advantage from syntactical compositionality, we figure out to deal with three interacting entities: Sender, Receiver, Channel. The interaction between Sender and Channel is described by action types tm standing for “transmit message tagged with 0”, tm standing for “transmit message tagged with 1”, dm standing for “deliver acknowledgement tagged with 0”, and dm standing for “deliver acknowledgement tagged with 1”. The interaction between Receiver and Channel is described by action types dm standing for “deliver message tagged with 0”, dm standing for “deliver message tagged with 1”, tm standing for “transmit acknowledgement tagged with 0”, and tm standing for “transmit acknowledgement tagged with 1”.

Furthermore, there are action types describing activities local to the single entities. Local activities of Sender are described by action types gm standing for “generate message”, and to standing for “timeout”. The only local activity of Receiver is described by action type cm standing for “consume message”. The local activities of Channel, which is composed of two independent half-duplex lines Line_m and Line_a, are described by action types pm standing for “propagate message tagged with 0”, pm standing for “propagate message tagged with 1”, pa standing for “propagate acknowledgement tagged with 0”, and pa standing for “propagate acknowledgement tagged with 1”. Additionally, there are other two activities local to Channel that are described by action type τ and represent the fact that a message or an acknowledgement gets lost or not.

So far we have considered the functional part of the protocol. Concerning the performance part, we have to remember that EMPA allows us to express only exponentially distributed
durations as well as zero durations. Supposing that the length of a message and the length of an
acknowledgement are exponentially distributed, it turns out that the message/acknowledgement
transmission/propagation/delivery times are exponentially distributed. However, the three
phases given by message/acknowledgement transmission/propagation/delivery are temporally
overlapped, i.e., they constitute a pipeline. As a consequence, in order to determine correctly
the time taken by a message/acknowledgement to reach Receiver(Sender), we model actions
related to transmission and delivery as immediate and we associate the actual timing (i.e., the
duration of the slowest stage of the pipeline) with actions related to propagation: we assume
that the message propagation time is exponentially distributed with rate $\delta$, the acknowledgement
propagation time is exponentially distributed with rate $\gamma$, and the loss probability is $p \in \mathbb{R}_{[0,1]}$.
Furthermore, we assume that the message generation time is exponentially distributed with rate
$\lambda$, and that message consumption is an immediate action in that irrelevant from the performance
viewpoint. Finally, we assume that the timeout period is exponentially distributed with rate
$\theta$: this is not realistic, but EMPA does not enable us to express deterministic durations, and a
Markovian analysis would not be possible otherwise.

The alternating bit protocol can be modeled with the following term $ABP$:

- $ABP \triangleq Sender_0 \parallel Channel \parallel R Receiver_0$:
  - $Sender_0 \triangleq <gm, \lambda>, <tm_0, \infty_1, 1>, Sender_0'$,
  - $Sender_0' \triangleq <da_0, *, Sender_1 + <da_1, *>, Sender_0' + <to, \theta>, Sender_0''$,
  - $Sender_0'' \triangleq <tm_0, \infty_1, 1>, Sender_0' + <da_0, *>, Sender_1 + <da_1, *>, Sender_0'$,
  - $Sender_1 \triangleq <gm, \lambda>, <tm_1, \infty_1, 1>, Sender_1'$,
  - $Sender_1' \triangleq <da_1, *>, Sender_0 + <da_0, *>, Sender_1' + <to, \theta>, Sender_0''$,
  - $Sender_1'' \triangleq <tm_1, \infty_1, 1>, Sender_1' + <da_1, *>, Sender_0 + <da_0, *>, Sender_1';
  - $Channel \triangleq Line_m \parallel Line_a$:
  - $Line_m \triangleq <tm_0, *>, <pm_0, \delta >, (<tau, \infty_1, 1, p >, <dm_0, \infty_1, 1 >, Line_m + <tau, \infty_1, p >, Line_m) + <tm_1, *>, <pm_1, \delta >, (<tau, \infty_1, 1, p >, <dm_1, \infty_1, 1 >, Line_m + <tau, \infty_1, p >, Line_m);$
  - $Line_a \triangleq <ta_0, *>, <pa_0, \gamma >, (<tau, \infty_1, 1, p >, <da_0, \infty_1, 1 >, Line_a + <tau, \infty_1, p >, Line_a) + <ta_1, *>, <pa_1, \gamma >, (<tau, \infty_1, 1, p >, <da_1, \infty_1, 1 >, Line_a + <tau, \infty_1, p >, Line_a);$
  - $Receiver_0 \triangleq <dm_0, *>, <cm_0, \infty_1, 1 >, <ta_0, \infty_1, 1 >, Receiver_1 + <dm_1, *>, <ta_1, \infty_1, 1 >, Receiver_0,$
  - $Receiver_1 \triangleq <dm_1, *>, <cm_0, \infty_1, 1 >, <ta_1, \infty_1, 1 >, Receiver_0 + <dm_0, *>, <ta_0, \infty_1, 1 >, Receiver_1$;

- $S = \{tm_0, tm_1, da_0, da_1\}$;
- $R = \{dm_0, dm_1, ta_0, ta_1\}$.

It has been simple to model the protocol thanks to syntactical compositionality: we have first
described the whole protocol as the parallel composition of the three entities it involves, then
we have modeled each entity separately. Moreover, the probabilistic choice between the reception
and the loss of a message/acknowledgement has been easily represented by means of immediate
actions. An important observation concerns terms $Sender_0''$ and $Sender_1''$. Since they model the
situation after a timeout expiration, they should comprise the retransmission action only in order
to be consistent with the definition of the protocol. The problem is that a deadlock may occur
whenever, after a sequence of premature timeouts (i.e., timeouts expired although nothing gets
lost), the sender is waiting to be able to retransmit the message, the receiver is waiting to be able to
retransmit the corresponding acknowledgement, the message line is waiting to be able to deliver
a previous copy of the message, and the acknowledgement line is waiting to be able to deliver a
previous copy of the acknowledgement. To destroy deadlock, $Sender_0''$ and $Sender_1''$ are allowed
Figure 11. Integrated operational interleaving semantics of ABP
Figure 12. Markovian semantics of ABP
Figure 13. Integrated operational net semantics of ABP
| $s_1$ | Sender$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_3$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_5$ | Sender$_0$ || $S$ Line$_m$,0 || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_7$ | Sender$''$$_0$ || $S$ Line$_m''$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_9$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{11}$ | Sender$''$$_0$ || $S$ Line$_m''$,0 || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{13}$ | Sender$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{15}$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{17}$ | Sender$_0$ || $S$ Line$_m$,0 || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{19}$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{21}$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{23}$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{25}$ | Sender$'$$_0$ || $S$ Line$_m$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{27}$ | Sender$'$$_0$ || $S$ Line$_m'$,0 || $||$ Line$_a$ || $R$ Receiver$_1$ |
| $s_{29}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{31}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{33}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{35}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{37}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_1$ |
| $s_{39}$ | Sender$'$$_0$ || $S$ Line$_m'$,0 || $||$ Line$_a$ || $R$ Receiver$_1$ |
| $s_{41}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{43}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{45}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{47}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{49}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{51}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{53}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{55}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{57}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{59}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{61}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{63}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{65}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{67}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{69}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{71}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{73}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{75}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{77}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{79}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{81}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{83}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{85}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{87}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{89}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |
| $s_{91}$ | Sender$'$$_0$ || $S$ Line$_m'$ || $||$ Line$_a$ || $R$ Receiver$_0$ |

Table 4. Shorthands for state names and place names (i)
<table>
<thead>
<tr>
<th>Rule Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>((\text{Sender}_0 \mid s \mid id) \mid R \mid id)</td>
</tr>
<tr>
<td>p2</td>
<td>((\text{Sender}_1 \mid s \mid id) \mid R \mid id)</td>
</tr>
<tr>
<td>p3</td>
<td>((\text{Sender}_0 \mid s \mid id) \mid a \mid id)</td>
</tr>
<tr>
<td>p4</td>
<td>((\text{Sender}_1 \mid s \mid id) \mid a \mid id)</td>
</tr>
<tr>
<td>p5</td>
<td>((\text{Sender}_0 \mid s \mid id) \mid R \mid id)</td>
</tr>
<tr>
<td>p6</td>
<td>((\text{Sender}_1 \mid s \mid id) \mid R \mid id)</td>
</tr>
<tr>
<td>p7</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p8</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p9</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p10</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p11</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p12</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p13</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p14</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p15</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p16</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p17</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p18</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p19</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p20</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p21</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p22</td>
<td>((\text{id} \mid s \mid (\text{Line}_m \mid s \mid id)) \mid R \mid id)</td>
</tr>
<tr>
<td>p23</td>
<td>(id \mid R \mid \text{Receiver}_0)</td>
</tr>
<tr>
<td>p24</td>
<td>(id \mid R \mid \text{Receiver}_1)</td>
</tr>
<tr>
<td>p25</td>
<td>(id \mid R \mid \text{Receiver}_0)</td>
</tr>
<tr>
<td>p26</td>
<td>(id \mid R \mid \text{Receiver}_1)</td>
</tr>
<tr>
<td>p27</td>
<td>(id \mid R \mid \text{Receiver}_0)</td>
</tr>
<tr>
<td>p28</td>
<td>(id \mid R \mid \text{Receiver}_1)</td>
</tr>
</tbody>
</table>

**Table 5. Shorthands for state names and place names (ii)**
to receive possible acknowledgements, thereby avoiding unnecessary retransmissions.

Before moving to the analysis of ABP, we would like to stress that the reason why we have chosen the alternating bit protocol as a case study to illustrate the integrated approach, is that such a protocol has become a standard example in the literature (see, e.g., [48, 27, 50, 54, 8]), so it can be used to compare the EMPA model with other models. For example, it turns out that: in [48, 27] performance aspects are completely neglected because a classical process algebra is used, in [50] a stochastic Petri net model is adopted but the unrealistic assumption that the timeout expires only if a loss actually occurs is made, and in [54, 8] stochastic process algebraic descriptions are given but the division into three temporally overlapped phases is not accurately taken into account.

The integrated operational interleaving semantics of ABP is presented in Figure 11: the names of the states have been shortened as reported in Table 4 and Table 5. The LTS $I[ABP]$ has 302 states, 76 of which are tangible: due to the symmetry of the protocol, only half states have been drawn (dashed transitions depict the link with the remaining states). Whenever neither losses nor premature timeouts occur, the states visited by the protocol are 1, 3, 5, 9, 15, 21, 25, 45, 51, 55 and the corresponding symmetric ones, i.e. 2, 4, 6, 10, 16, 22, 26, 46, 52, 56: all the other states can be deemed as faulty states. Following the proposed approach, we can use the LTS $F[ABP]$ (obtained from $I[ABP]$ by dropping action rates) to detect some functional properties. For example, we see that each state has at least one incoming transition and one outgoing transition: this means that the protocol is deadlock free. Observe that, if $Sender^n$ and $Sender^m$ had not been carefully designed, then we would have obtained eight deadlock states: 113, 197, 213, 301 and the corresponding symmetric ones. As another example, by resorting to equivalence checking we can prove that, whenever all the action types occurring in ABP are hidden except for $gn$ and $cm$, then ABP behaves as a buffer with capacity one that can engage in a sequence of alternating actions with the two observable types. The LTS $F[ABP]$ could be fully analyzed by using tools like CW [27].

The Markovian semantics of ABP is presented in Figure 12. The PLTS $M[ABP]$ has 33 states. It has been obtained from the LTS $I[ABP]$ by discarding action types, removing the 226 vanishing states and ordinarily lumping the remaining 76 states: note that the phase of ordinary lumping has been able to detect all the symmetry of the protocol. Since $M[ABP]$ is finite and strongly connected, it represents a HCTMC for which the steady-state probability distribution function exists. Following the proposed approach, we can exploit such a HCTMC for assessing some performance indices. For example, the throughput of the protocol is given by $\lambda$ times the sum of the steady-state probabilities of the states having a transition labeled with $\lambda$, while the utilization of the message line is given by the sum of the steady-state probabilities of the states having a transition labeled with $p\cdot\delta$ and a transition labeled with $(1-p)\cdot\delta$. The HCTMC $M[ABP]$ could be fully analyzed by using tools like SHARPE [63].

The integrated operational net semantics of ABP is presented in Figure 13, where the names of the places have been shortened as reported in Table 5. The GSPN $N_{loc}[ABP]$ comprises 28 places and 36 transitions. Since the operational net semantics for EMPA meets the retrievability principles, ABP and $N_{loc}[ABP]$ model exactly the same protocol in two different ways: the algebraic description is compositional and more readable, the net description is more concrete and highlights dependencies, conflicts and synchronizations among activities. Also, $N_{loc}[ABP]$ is more compact than $I[ABP]$ because the concurrency is kept explicit instead of being simulated by means of interleaving. Following the proposed approach, we can exploit such a GSPN for studying functional and performance properties. This can be assisted by tools like GreatSPN [26].

9 Conclusion

In this paper we have proposed an approach for modeling and analyzing concurrent systems that integrates different views (abstract vs. concrete) as well as different aspects (functional vs. performance) of their behavior. In order to implement the integrated approach, we have developed a new stochastic process algebra called EMPA.
Related work: The idea underlying the integrated approach comes from the proposal in [55], where complementary views of concurrent systems, each one describing the systems at a different level of abstraction, are brought together in one uniform framework by establishing the appropriate semantic links. This realizes the stepwise development of complex systems through various levels of abstraction, which is good practice in software and hardware design. We have then extended the proposal in [55] by considering an orthogonal form of integration that relates functional and performance aspects of concurrent systems.

The development of EMPA has been affected by the stochastic process algebras MTIPP [34] and PEPA [39], and by the formalism of GS/Ns [5, 6]. While designing EMPA, emphasis has been placed on expressiveness and formal semantics.

In EMPA, action durations are mainly expressed by means of exponentially distributed random variables (like in MTIPP and PEPA), but it is also possible to express immediate actions (like in MTIPP) each of which is assigned a priority level and a weight (like in GS/Ns). Immediate actions permit to model activities associated with logical events (see, e.g., the delivery of a customer to the server in Section 5 and Section 6) as well as activities that are irrelevant from the performance viewpoint (see, e.g., message consumption in Section 8), thereby providing a mechanism for performance abstraction in the same way as action type \( \tau \) provides a mechanism for functional abstraction: they also supply the designer with a good degree of flexibility (see, e.g., the description of the pipeline in Section 8). Furthermore, immediate actions allow to model concurrent systems whose activities may have different priorities (see, e.g., the QS in Section 5), and can be used to describe explicitly probabilistic choices avoiding the need of a new operator (see, e.g., message and acknowledgement loss in Section 8). Finally, the interplay of exponentially timed and immediate actions makes it possible (though not in an atomic manner) the description of activities whose durations follow a phase-type distribution (see Section 5). It is worth noting that, e.g., hyperexponential distributions cannot be represented without weighted immediate actions, since there is no term in which only exponentially timed actions occur such that its Markovian semantics is p-isomorphic to the HCTMC reported in Figure 8(c). Actually, like weighted immediate transitions in GS/Ns, weighted immediate actions are essential in order to model HCTMCs where more than one state can be initial.

EMPA is also endowed with passive actions (somewhat different from those of MTIPP and PEPA). Passive actions play a prominent role in EMPA because they allow for nondeterministic choices, and are essential in the synchronization discipline on action rates since it requires that at most one active action is involved. MTIPP and PEPA allow for more general kinds of synchronization [40], but we think that our choice based on the client-server paradigm leads to a more intuitive treatment of the interaction among processes. The presence of passive actions has forced us to introduce the temporal restriction operator (missing both in MTIPP and PEPA) in order to single out terms that cannot execute passive actions, i.e. terms completely specified from the performance viewpoint.

Another operator has been introduced with respect to both MTIPP and PEPA: the functional relabeling operator. The main motivation is that it turns out to be helpful to obtain more concise algebraic representations of concurrent systems (see, e.g., the QS in Section 6).

The presence of immediate actions (with priority levels and weights) and passive actions has required a great care in the definition of the integrated operational interleaving semantics: this is reflected by the use of function \( \textit{Select} \) (immediate actions) and function \( \textit{Norm} \) (passive actions) in Table 1. The Markovian semantics has required a cautious handling of immediate transitions, and comprises the application of an ordinary lumping procedure in order to reduce the state space: moreover, in [17] it has been checked for correctness by means of a benchmark composed of QSs of the class \( M/M \). The integrated operational net semantics, developed together with EMPA itself to support the integrated approach, is defined in the style of [29, 55]; our contribution has been the handling of marking-dependent rates. Subsequently, integrated denotational net semantics for stochastic process algebras have been proposed in [62].

Finally, the notion of integrated equivalence, i.e. \( \sim_{EMB} \), has been set up by assembling complementary proposals [45, 36, 39, 25, 67, 48] in the simplest possible way, and it has turned out to be the coarsest congruence contained in \( \sim_{FP} \) for a large class of terms. Concerning the property
of congruence, it is worth noting that the presence of different priority levels is critical. We have tackled this problem by introducing a priority interpretation operator in the style of [10]. Another approach has been followed in [60, 37], where the congruence issue is separately investigated for MTIPP in the case when immediate actions implement probabilistic choices and in the case when immediate actions describe management activities. The strong EMBE is strong in the sense that it abstracts from neither internal actions nor immediate actions. The main motivation to set up weaker forms of equivalence is the possibility of reducing algebraically the state space by eliminating transitions that are irrelevant from the functional or the performance point of view. This aspect has been addressed in [39] for internal actions, and in [37] for internal immediate actions.

**Perspective:** As the various semantics for EMPA can be fully mechanized, we are presently designing a software tool that associates with each EMPA term the collection of its semantics. On this basis, the next step should consist of integrating this tool with the various ones, already available, tailored for specific purposes. For instance, given an EMPA term, the LTS representing its functional semantics can be input for tools like CW [27]; the HCTMC representing its Markovian semantics can be given as input to tools like SHARPE [63]; and the GSPN representing its integrated operational net semantics can be input for tools like GreatSPN [26]. Therefore, our work opens a perspective to a fully integrated tool that can help in making computer aided - and possibly automatic - the functional and performance analysis of concurrent systems.

**Future research:** Finally we outline some open problems left for future research.

(i) The strong EMBE is preserved by all the operators of EMPA as well as recursive definitions. This is the starting point to axiomatize the notion of equivalence, in order to make equational reasoning possible thereby avoiding the generation of the state space. Moreover, since EMPA (as well as the other stochastic process algebras) allows the system designer to model and analyze the system performance, it would be useful to develop a notion of preorder whereby comparing algebraic descriptions based on the performance of the systems they model.

(ii) The Markovian semantics for EMPA is defined by means of an algorithm that manipulates the LTSs produced by the integrated operational interleaving semantics. It would be interesting to find a compositional definition for the Markovian semantics that generates the minimum HCTMC associated with a term by considering the syntactical structure of the term itself. This would contribute to the struggle against state space explosion. This problem has been solved in [39, 61, 25] in the case of absence of immediate actions. Additionally, for the time being the algorithm to compute the Markovian semantics can be applied only to temporally closed terms. However it could be extended to all the terms, provided that passive transitions are treated as parametric active transitions. As a consequence, we would obtain parametric HCTMCs suitable for sensitivity analysis of performance.

(iii) The integrated operational net semantics for EMPA is developed according to the location-oriented approach, i.e. the syntactical structure of terms is encoded inside places. From the applicative viewpoint, its major drawback is that the resulting nets are safe, hence huge. In [15] this problem has been solved by resorting to the label-oriented approach: terms are decomposed into places that ignore the syntactical structure of terms themselves, notably the presence of parallel composition operators, so that e.g. term \( <a, \lambda>, 0 \parallel \| <a, \lambda>, 0 \) needs only one place \( <a, \lambda>, 0 \) (instead of \( <a, \lambda>, 0 \parallel id \| id \| <a, \lambda>, 0 \)) marked with two tokens. Another optimization concerns choices: alternative compositions are translated by linear constructions instead of Cartesian product constructions. Given a term \( E \), its integrated operational label-oriented net semantics \( N_{lab}[E] \) is in general smaller than \( N_{loc}[E] \) and sometimes even finite instead of infinite: for instance, while \( N_{loc}[System_{MT}/M_{1234}] \) (depicted in Figure 10) has 16 places, 16 transitions and 60 arcs, \( N_{lab}[System_{MT}/M_{1234}] \) (depicted in [15]) has only 7 places, 4 transitions and 14 arcs. The price to pay is that
inhibitor arcs come into play, except for terms in which all the choices are guarded. A different approach for obtaining smaller net representations of stochastic process algebra terms is proposed in [62], where an integrated denotational net semantics based on colored stochastic Nets is outlined.

(iv) In [11] it has been shown that stochastic Petri nets can be used to assess both the correctness and the performance of concurrent algorithms, provided that translation rules are given in order to derive a GSPN model from the code of the algorithm. Of course, the same idea can be applied to EMPA (and the other stochastic process algebras). In particular, the translation rules may be set up by following the guideline in [48] Chapter 8, where an imperative concurrent programming language is defined and its semantics is given by translation into a classical process algebra.

(v) How to scale the integrated approach to general distributions? Probably, this is the most challenging open problem because we cannot exploit any more the memoryless property of exponential distributions. The passage to general distributions affects first of all the definition of the integrated operational interleaving semantics. The pure interleaving style of classical process algebras is no more sufficient, because we need more information in the LTSs as described in [34, 65, 8]. An alternative approach proposed in [24] is to use instead an integrated truly concurrent semantics. We could follow this approach since we already have an integrated truly concurrent semantics based on stochastic Petri nets, but this could make the definition of the performance semantics more difficult. Actually, the second important consequence of the passage to general distributions is that, given a term, we have to understand which kind of performance model it represents (e.g., a Markov process, a semi-Markov process or a generalized semi-Markov process), and then we have to construct such a performance model. Whenever this performance model cannot be analyzed by means of mathematical techniques, we have to resort to a simulation study: it is worth noting that stochastic process algebras fit in a natural way the paradigm of process-oriented simulation.

Acknowledgements
We wish to thank Rance Cleaveland, Alessandro Fabbri, Roberto Segala and Rick Sheldon for the valuable discussions, and Alessandro Fabbri and Roberto Segala for their careful proofreading. Furthermore, we are grateful to Nadia Busi for her suggestions about the presentation of the integrated operational interleaving semantics, and Mario Bravetti for his remarks about rate normalization in the integrated operational net semantics. This research has been partially funded by MURST, CNR and ESPRIT BRA n. 8130 LOMAPS.

References

UBLCS-95-14


[37] H. Hermanns, M. Rettelbach, T. Weiß, “Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching”, in [3], pages 530-541
[40] J. Hillston, “The Nature of Synchronisation”, in [2], pages 51-70
[60] M. Rettelbach, “Probabilistic Branching in Markovian Process Algebras”, in [3], pages 590-599