Process Algebra and Markov Chains

Ed Brinksma and Holger Hermanns

Formal Methods and Tools Group, Faculty of Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. This paper surveys and relates the basic concepts of process algebra and the modelling of continuous time Markov chains. It provides basic introductions to both fields, where we also study the Markov chains from an algebraic perspective, viz. that of Markov chain algebra. We then proceed to study the interrelation of reactive processes and Markov chains in this setting, and introduce the algebra of Interactive Markov Chains as an orthogonal extension of both process and Markov chain algebra. We conclude with comparing this approach to related (Markovian) stochastic process algebras by analysing the algebraic principles that they support.

1 Introduction

The construction of models for the performance and reliability analysis of systems is a difficult task that requires intelligence and experience. Due to the ever increasing size and complexity of systems, such as e.g. embedded and distributed systems, there is a growing need for powerful methods to master the related complications of the modelling task. Performance models do not only become very large, but because of the intricate interplay between (many) system components they can also have a highly irregular structure that is very hard to understand and control. Traditional performance models like Markov chains and queueing networks are widely accepted as simple but effective models in different areas, yet they lack the notion of hierarchical system (de)composition that has proved so useful for conquering the complexity of systems in the domain of functional system properties. The absence of such techniques seriously hampers the adequate modelling of complicated modern systems.

A prominent example of a (class of) formalism(s) for the compositional, hierarchical description and analysis of functional system behaviour is *process algebra* [37,3,28]. It offers a mathematically well-elaborated framework for reasoning about the structure and behaviour of reactive and distributed systems in a compositional way, including abstraction mechanisms that allow for the treatment of system components as black boxes, encapsulating their internal structure. Process algebras are typically equipped with a formally defined structured operational semantics (SOS [51]) that maps process algebra terms onto *labelled transition systems* in a compositional manner. Such labelled transition systems consist of a set of states and a transition relation that describes how the system evolves from one state to another. These transitions are labelled with action

E. Brinksma, H. Hermanns, and J.-P. Katoen (Eds.): FMPA 2000, LNCS 2090, pp. 183–231, 2001.

names that represent the (inter)actions that may cause the transitions to occur. Such transition systems can be represented as directed edge-labelled graphs, with the states as nodes of and the transitions as edges (labelled with action names).

The labelled transition model is very close to the usual representation of Markov chains as transition systems or automata. Also there system states are connected by directed transition arcs that are labelled. In the case of discrete time Markov chains the labels are probabilities, and in the case of continuous time Markov chains, which are the topic of this paper, the labels are the *rates* that correspond to the exponential distributions that represent the stochastic delays associated with the state transitions. This structural correspondence between the two models motivated the beginning of research in the early 1990's on *stochastic process algebras* [4,27,16], which sought to integrate performance modelling with Markov chains with functional analysis, and to transfer the process algebraic notions of (de)composition and hierarchy to Markov chain theory.

The fruitfulness of this approach to the specification and generation of Markov chains has been demonstrated by a number results. In the stochastic setting, *bisimulation* equivalence [40], a central notion of equivalence for comparing labelled transition systems, has been shown to coincide with *lumpability*, a key concept for the aggregation of Markov chains [27]. Moreover, as bisimulation can be shown to be preserved under system composition operators (algebraically: bisimulation is a *congruence*), Markov chain aggregation can be carried out compositionally, i.e. component-wise. Several (small to medium size) case studies have shown the practicality of this compositional approach, and important progress has been made in exploiting the syntactic structure of specifications for performance analysis purposes, see [26].

In this paper we aim (i) to give an introduction to the essentials of process algebra that are needed for compositional performance modelling, (ii) to introduce the process algebraic approach to Markovian performance modelling using Interactive Markov Chain (IMC) algebra, and (iii) to survey the main algebraic principles that underly related (Markovian) stochastic process algebras. The paper is organised as follows. Section 2 introduces the concepts and features of process algebras, while Section 3 introduces continuous time Markov chains from an algebraic perspective. The algebra of interactive Markov chains is discussed in Section 4. At the end, section 5 compares IMC and other existing stochastic process algebra in terms of the algebraic principles that they support. Section 6, finally, presents the conclusions of the paper.

2 Process Algebra

In this section we introduce a simple process algebraic framework that we will use throughout our paper. Its purpose is to give an intuitive understanding of the key ingredients of process algebra, and prepare for their use in the rest of the paper. We start with the introduction of labelled transition systems, which constitute a simple but powerful operational model for reactive behaviour. We show how these transition systems can be constructed with the aid of three basic operators,

viz. *action-prefixing*, *choice*, and *recursive specification*. Together, these operations give rise to a basic process algebra that can be used for the description of sequential processes. We then extend this algebraic language to concurrent processes with the aid of an operator for *parallel composition*, in combination with an *abstraction operator* to control the scope of the interactions between concurrent processes. It turns out that for many purposes the labelled transition system model is too fine, i.e. there are many different transition systems that display intuitively identical behaviour. This leads us to the definition of useful behavioural equivalences over reactive behaviour, viz. the notions of *strong* and *weak bisimulation*. Finally, we turn to an axiomatic presentation of process algebra by discussing the axiom systems that are induced by the bisimulation equivalences. For a fuller account of the material covered by this section, we refer the reader to the extensive literature of process algebra, e.g. [3,40,5,10].

2.1 Labelled Transition Systems

State-transition diagrams, automata and similar models are widely used to describe the dynamic behaviour of systems. They consists of a set of states S together with a representation of possible state changes. The latter is usually given in the form of some relation (or function) over states, i.e. a subset of the Cartesian product $S \times S$. Intuitively, a pair of states (P, Q) is in this relation if it is possible to change from state P to Q in a single step. Such transition relation are often denoted with an arrow (e.g. $\longrightarrow$), so that $(P, Q) \in \longrightarrow$ can then be conveniently rewritten, in infix notation, as $P \longrightarrow Q$, thus nicely representing the possible state change between P and Q.

In the context of process algebras, transition systems appear in a specific form, viz. that of labelled transition systems. Labelled transition systems form a particular class where state changes are conditioned on occurrences of actions drawn from an *action set*, or *alphabet*, $\mathcal{A}$. A state change between P and Q here entails the occurrence of a related action. Therefore, the transition relation $\longrightarrow$ is a subset of $S \times \mathcal{A} \times S$ rather than a binary relation over just S. Again, it will be convenient to denote $(P, \mathsf{a}, Q) \in \longrightarrow$, using a kind of mixfix notation, as $P \xrightarrow{\mathsf{a}} Q$. Here the action appears as the label of the transition, whence the term 'labelled' transition system.

Definition 1. *A labelled transition system is a triple $(S, \mathcal{A}, \longrightarrow)$, where*

- *S is a nonempty set of states,*
- *$\mathcal{A}$ is a set of actions, and*
- *$\longrightarrow \subset S \times \mathcal{A} \times S$ is a set of action labelled transitions.*

In order to use labelled transition systems as an operational model of systems it is common practice to identify a specific initial state P in the transition system where operation starts. A transition system with an initial state is called a process.

Definition 2. *A process is a quadruple $(S, \mathcal{A}, \longrightarrow, P)$, where $(S, \mathcal{A}, \longrightarrow)$ is a labelled transition system and $P \in S$ is the initial state.*

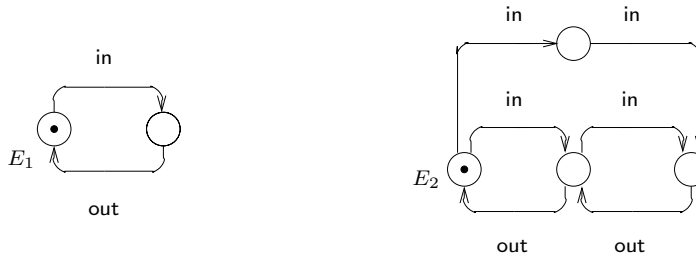


Fig. 1. Two processes.

Example 1. Figure 1 contains two examples of processes. In principle, states are represented as circles labelled with identifiers from S. We adopt the convention, however, to use state labels only if they are required for understanding. We use $\odot$ to denote the initial state. The first process, with initial state E_1, is a simple one-place buffer. It accepts data via the action in and releases them with the action out. The right process is able to buffer two values, but with a slightly unusual restriction. From its initial state E_2 there is a choice between two transitions that are both labelled with in. In the standard interpretation of process algebra this represents a *nondeterministic choice* that is made as part of the execution of the action in, i.e. the receipt of a first datum. The lower branch leads to the usual behaviour of a two-place buffer, whereas after the upper branch no datum can be released, i.e. no out can be executed, before two data have been accepted.

2.2 Basic Processes

Process algebra is a means to specify processes and to reason about them. To achieve this we use an *algebraic language*, based on *combinators*, i.e. operators that compose processes into new ones. The terms of the algebraic language, the *behaviour expressions*, are interpreted as labelled transition systems with a distinguished initial state, i.e. as processes in the sense of Definition 2. This is done using so-called structural operational semantic rules. This semantic interpretation induces equalities between different behaviour expression, yielding an *equational calculus* for reasoning about processes.

We introduce the language by a simple BNF-style grammar. We assume as given a countable set $\mathcal{V}$ of *variables* that are used to express repetitive behaviour, and, as before, a set of actions $\mathcal{A}$. We use a, b, ... for elements of $\mathcal{A}_\tau$. We also assume a distinguished element τ, representing *internal* (or *silent*, or *hidden*) actions, and let $\mathcal{A}_\tau$ denote the set $\mathcal{A} \cup \{\tau\}$.

Definition 3. *Let* $\mathsf{a} \in \mathcal{A}_\tau$ *and* $X \in \mathcal{V}$. *We define the language* PA *as the set of expressions given by the following grammar.*

$$\mathcal{E} ::= \quad 0 \quad | \quad \mathsf{a}.\mathcal{E} \quad | \quad \mathcal{E} + \mathcal{E} \quad | \quad X \quad | \quad [X := \mathcal{E}]_i$$

$[X := \mathcal{E}]$ is a shorthand notation for an arbitrary (finite) set of defining equations of the form $[X_1 := \mathcal{E}_1\ ,\ X_2 := \mathcal{E}_2\ ,\ \ldots\ ,\ X_n := \mathcal{E}_n]$, or in vector notation,

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} := \begin{bmatrix} \mathcal{E}_1 \\ \mathcal{E}_2 \\ \vdots \\ \mathcal{E}_n \end{bmatrix}$$

with $X_i \in \mathcal{V}$, and $\mathcal{E}_i$ complying to the above grammar.

We use E, E_1, E_2, F, ... to range over arbitrary expressions of PA. The intuitive meaning of the language constructs is as follows.

- The terminal symbol 0 describes a *terminated* behaviour that cannot engage in any (inter)action.
- The expression $\mathsf{a}.E$ may interact on action a and afterwards behave as expression E. We shall say that E is *action prefixed* by a.
- The expression $E + F$ combines two alternatives. It either exhibits the behaviour of expression E or the behaviour of expression F. The terminal symbol $+$ is called the *choice operator*. The choice between E and F is resolved in interaction with other processes on the initial actions of E and F.
- The expression $[X := E]_i$ defines a behaviour in terms of the set $[X := E]$ of mutually recursive behaviour definitions . Its meaning is as follows: $[X := E]_i$ behaves as E_i, where the behaviour of the recursion variables is obtained by 'unrolling': wherever a behaviour X_j is reached, it is replaced by (the behaviour of) its definition $[X := E]_j$.

In the sequel, we restrict ourselves to expressions, where each occuring variable X_j is bound by a defining equation $X_j := \ldots$. Such expressions are called *closed expressions*. An expression $E \in \mathsf{PA}$ is closed, if each variable $X_j \in \mathcal{V}$ appearing in E only appears inside the scope of a guarding defining equation set, i.e. inside an expression $[\ldots\ ,\ X_j := \ldots\ ,\ \ldots]_i$. The set of closed expressions is denoted PA^c.

Example 2. An example of an expression that is not closed is $\mathsf{in}.[X_1 := \mathsf{in}.X_2]_1$. The processes of Figure 1 can be specified as follows:

- E_1 is defined by $[X_1 := \mathsf{in}.\mathsf{out}.X_1]_1$
- E_2 is defined by $[X_1 := \mathsf{in}.X_2 + \mathsf{in}.\mathsf{in}.\mathsf{out}.X_2\ ,\ X_2 := \mathsf{in}.\mathsf{out}.X_2 + \mathsf{out}.X_1]_1$

We formalise this intuitive interpretation by giving it an operational semantics in terms of labelled transition systems. The style of definition that we use goes back to Plotkin [51], and is usually referred to as structured operational semantics (SOS), since it defines the operational interpretation of a behaviour expression inductively over its syntactical structure.

We define a semantics for closed expressions of PA by mapping the complete language PA^c onto a universal transition system. The state space of this transition system is the set of all closed expressions according to Definition 3. Since

$$\frac{}{\mathsf{a}.E \xrightarrow{\mathsf{a}} E} \qquad \frac{E \xrightarrow{\mathsf{a}} E'}{E + F \xrightarrow{\mathsf{a}} E'} \qquad \frac{F \xrightarrow{\mathsf{a}} F'}{E + F \xrightarrow{\mathsf{a}} F'} \qquad \frac{E_i\{[X := E]_i / X_i\} \xrightarrow{\mathsf{a}} E'}{[X := E]_i \xrightarrow{\mathsf{a}} E'}$$

Table 1. Operational semantic rules for PA^c.

each $E \in \mathsf{PA}^c$ appears somewhere in this transition system the corresponding semantics is determined by the state space reachable from this expression.

The first SOS-rules that we need are given in Table 1. The rules have the format

$$\frac{B}{C} \quad A,$$

to express that *if A holds, then B implies C*, where A,B and C statements about the existence of labelled state transitions. The notation $E\{F/X\}$ is used to represent the result of a simultaneous substitution of each occurrence of variable X by expression F in expression E.

Definition 4. *The universal transition system $\mathcal{U}$ is given by the triple $(\mathsf{PA}^c, \mathcal{A}, \longrightarrow)$, where $\longrightarrow \subset \mathsf{PA}^c \times \mathcal{A} \times \mathsf{PA}^c$ is the least relation satisfying the operational rules in Table 1.*

This definition provides a semantics for each element of $E \in \mathsf{PA}^c$, via the fragment of $\longrightarrow$ reachable from the state E in $\mathcal{U}$. For a closed expression E, we let $Reach(E)$ denote the set of states reachable from E in the universal transition relation $\mathcal{U}$: $Reach(E) = \{E' \mid (E, E') \in T^*\}$ where T is the unlabelled transition relation in $\mathcal{U}$, i.e. $T = \{(F, F') \in \mathsf{PA}^c \times \mathsf{PA}^c \mid \exists a \in \mathcal{A}.\ F \xrightarrow{\mathsf{a}} F'\}$.

Definition 5. *The semantics of a closed behaviour expression $E \in \mathsf{PA}^c$ is a process $(S, \mathcal{A}, \longrightarrow', E)$, where $S = Reach(E)$ and $\longrightarrow' = \longrightarrow \cap (S \times \mathcal{A} \times S)$.*

Because of this definition we can adopt the fairly general convention of identifying a process with its initial state. Closed expressions are thus also called processes.

Example 3. In order to prove that the process E_1 of Figure 1 possesses an outgoing transition labelled with action in we apply the operational rules of Table 1 to construct the following derivation.

$$\frac{\dfrac{}{\mathsf{in.out}.[X_1 := \mathsf{in.out}.X_1]_1 \xrightarrow{\mathsf{in}} \mathsf{out}.[X_1 := \mathsf{in.out}.X_1]_1}}{[X_1 := \mathsf{in.out}.X_1]_1 \xrightarrow{\mathsf{in}} \mathsf{out}.[X_1 := \mathsf{in.out}.X_1]_1}$$

2.3 Concurrency and Abstraction

Although basic process algebra suffices, at least in principle, for the description of processes, it is too limited to be of great practical value. When specifying and analysing reactive systems it will often be necessary to conceive of them as the concurrent composition of a number of subprocesses. This can be the case because parallelism is a natural feature of the given system, and we wish to represent it. Or it may be that the properties of a system can be understood better if its behaviour is decomposed into a number of smaller components. Many realistic systems are so complicated, in fact, that they can only be understood in terms of a concurrent composition of components.

A key ingredient of concurrency is the possibility for component processes to interact. Processes interact to achieve a common goal, which means that they somehow have to synchronise their activities, e.g. by exchanging data. Different forms of process interaction have been studied in the literature of process algebra [37,40,10,29]. Distinctive features are asynchronous vs. synchronous and binary vs. multiparty interaction. For our purposes it will be convenient to use the synchronous multiparty interaction as defined, for instance, in the ISO specification language LOTOS [30,5].

We introduce a binary parallel composition operator that is indexed with the set of actions that its component processes have to synchronise on. All other actions, i.e. those that are not in the index set of the composition operator, can be performed independently of the other component process. The basic form of interaction therefore is synchronisation on actions: the execution of a synchronised action is a joint activity of all synchronising processes.

If P and Q are two processes, such synchronous parallel composition is denoted

$$P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$$

By varying the set of synchronising actions, parallel composition ranges from *full synchronisation*, when the set comprises all the possible actions, to *arbitrary interleaving*, when the set is the empty (in this case we use the concise notation $P \mid\mid\mid Q$). The intuition behind this operator is summarised by the following informal properties:

- A state change of $P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$ is possible if P may change to, say P', on the occurrence of an action a that is not contained in $\{\mathsf{a}_1 \ldots \mathsf{a}_n\}$. The result of the state change is $P' \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$, since only P has changed state.
- Symmetrically, a state change of $P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$ is also possible if Q may change to some Q', on the occurrence of an action a that is not contained in $\{\mathsf{a}_1 \ldots \mathsf{a}_n\}$, resulting in $P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q'$.
- In order to be able to interact on an action a contained in $\{\mathsf{a}_1 \ldots \mathsf{a}_n\}$, both P and Q have to be able to perform a and thereby evolve to some P' and Q'. If this condition is met $P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$ may in a single step change state to $P' \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q'$.
- No other transitions are possible for $P \shortparallel \mathsf{a}_1 \ldots \mathsf{a}_n \shortparallel Q$.

$$\frac{P \xrightarrow{\mathsf{a}} P'}{P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q \xrightarrow{\mathsf{a}} P' \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q} \mathsf{a} \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

$$\frac{Q \xrightarrow{\mathsf{a}} Q'}{P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q \xrightarrow{\mathsf{a}} P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q'} \mathsf{a} \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

$$\frac{P \xrightarrow{\mathsf{a}} P' \qquad Q \xrightarrow{\mathsf{a}} Q'}{P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q \xrightarrow{\mathsf{a}} P' \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q'} \mathsf{a} \in \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

Table 2. Structural operational rules for parallel composition.

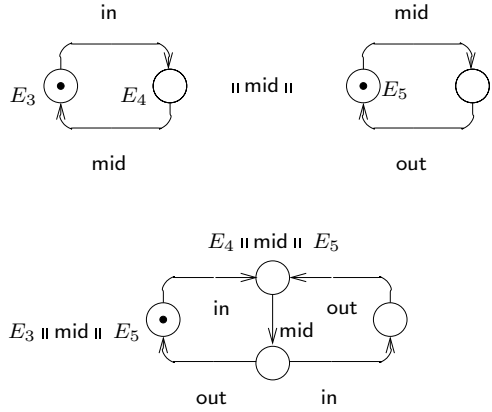


Fig. 2. Parallel composition of two processes.

We extend PA^c with the new operator by stipulating that if P and Q are in PA^c then $P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q$ is in PA^c as well. We can now formalise the above requirements as SOS-rules for the process $P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q$. The first three requirements are reflected in the three derivation rules of Table 2. The last property is automatically fulfilled as the transition relation itself is defined as the *least* relation satisfying the definition, i.e. it does not possess non-derivable transitions.

In the third SOS-rule of Table 2 it can be seen that the result of synchronisation on an action a is a transition of the composite behaviour again labelled with the same action a. This choice (borrowed from [46,5]) is an essential ingredient to enable so-called *multiway* synchronisation, where further processes may synchronise with the a-labelled transition of the composition. This approach, although fairly straightforward, is one of a number of alternatives to interaction and synchronisation, e.g. see [39] for a discussion of this topic.

$$\frac{P \xrightarrow{\mathsf{a}} P'}{\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P \xrightarrow{\mathsf{a}} \texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P'} \quad \mathsf{a} \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

$$\frac{P \xrightarrow{\mathsf{a}} P'}{\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P \xrightarrow{\tau} \texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P'} \quad \mathsf{a} \in \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

Table 3. Structural operational rules for abstraction.

Example 4. Figure 2 shows parallel composition of two processes E_3 and E_5. Below these processes the resulting process $E_3 \parallel \mathsf{mid} \parallel E_5$, obtained by applying the rules of Table 2, is also depicted.

The concept of multiway synchronisation has proven convenient from a specification engineering point of view. It allows for a constraint-oriented style of system specification, where processes add conditions on the occurrence of interactions incrementally using concurrent composition, see e.g. [58]. However, with the operators introduced so far, all actions that occur anywhere in a system specification remain available for further synchronisation with new processes. This is undesirable, since most system design methods try to work with components as black boxes, i.e. as functionality without internal structure. Such an approach calls for mechanisms to *abstract* from internal aspects that are irrelevant at higher design levels.

In process algebra the concept of abstraction must be dealt with in terms of an operator, the *abstraction operator*. The key to this operator is a distinguished action, usually named τ, that symbolises *internal* or *hidden* action, e.g. a state change that does not depend on synchronisation with the environment. Actions other than τ are called *external* or *observable*. For a given process P and actions $\mathsf{a}_1 \ldots \mathsf{a}_n$ abstraction or *hiding* of those actions simply renames them into the internal action τ. We use

$$\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P$$

to denote this operation. Again, we extend PA^c with a closure condition, viz. that if P is in PA^c then so is $\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P$. The semantics of the abstraction operator are given by the operational rules in Table 3.

We would like to point out that in a concurrent composition internal actions of one component process should be completely independent of those of the other. Hence, synchronisation on internal actions is ruled out, i.e. τ cannot occur in the index set $\{\mathsf{a}_1 \ldots \mathsf{a}_n\}$ of a composition $P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q$.

Example 5. In Figure 3 we have depicted the result of internalising the action mid in the process $E_3 \parallel \mathsf{mid} \parallel E_5$ by means of abstraction. The behaviour of the resulting process behaves as a two place buffer composed out of two one-place

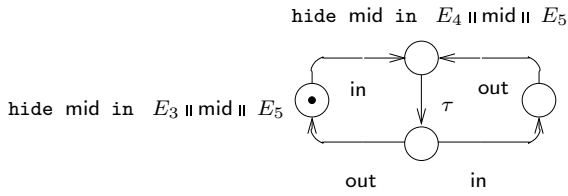


Fig. 3. Abstraction applied to composed processes.

buffers, E_3 and E_5. The first accepts data with action in. These are then passed *internally* to process E_5, which, in turn, can output them. E_3 may accept a second input, but before it can pass it on to E_5 this process has to output the input it accepted earlier.

2.4 Equivalence

An essential part of the development of process algebraic theory has been devoted to the study of suitable notions of behavioural equivalence. Such equivalences are induced by various notions of what constitutes process behaviour, different processes being equivalent iff they display *identical* behaviour. This behaviour-oriented (as opposed to state-oriented) point of view implies that the identity of states cannot be relevant for distinguishing between processes, whereas the labelling of transitions is. All common process algebraic equivalences share this characteristic. Still, there exists an overwhelmingly rich collection of such equivalences. Their variety is caused by the different intuitions about process behaviour. R.J. van Glabbeek has extensively studied the different behavioural equivalences [56,55]. They are classified according to the observational powers that an observer or experimenter must have to distinguish between different processes. In this paper we will confine ourselves to a particular, but very important class of equivalences, the *bisimulation relations*. We will argue why this is a good class of equivalences for our purposes, in this section and also later on, when probabilities and probability distributions come into play. We start by considering the so-called 'strong' equivalence, where internal and external actions are treated on an equal footing. After that we proceed with the 'weak' equivalence, which abstracts from internal transitions.

Strong equivalence. A labelled transition system can be seen as being essentially an automaton, with its finiteness conditions removed and with only success states. Therefore, the notion language equivalence of automata would seem a natural candidate to be considered for the characterisation of process equivalence. Two transition systems are language equivalent iff they accept the same language, i.e. their (finite) execution traces determine the same set of finite sequences over *Act*. In the context of process algebra this relation is called *trace equivalence*.

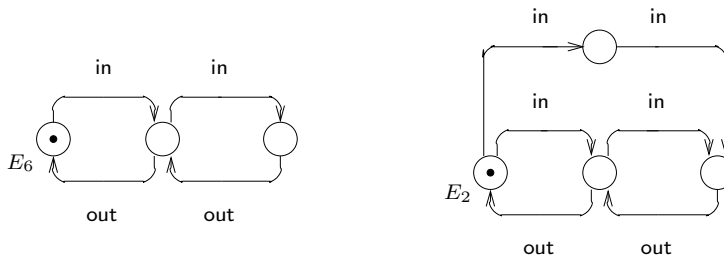


Fig. 4. Two processes with equivalent traces.

Notation. We use $P \xrightarrow{\mathsf{a}_1}\xrightarrow{\mathsf{a}_2} \cdots \xrightarrow{\mathsf{a}_n} P'$ to denote that there exist processes $P_1, P_2, \ldots, P_{n-1}$ such that $P \xrightarrow{\mathsf{a}_1} P_1 \xrightarrow{\mathsf{a}_2} P_2 \cdots P_{n-1} \xrightarrow{\mathsf{a}_n} P'$.

Definition 6. *Let P and Q be processes. P and Q are* strong trace equivalent*, written $P \sim_{tr} Q$, if for all P' and Q',*

$$P \xrightarrow{\mathsf{a}_1}\xrightarrow{\mathsf{a}_2} \cdots \xrightarrow{\mathsf{a}_n} P' \quad \textit{if and only if} \quad Q \xrightarrow{\mathsf{a}_1}\xrightarrow{\mathsf{a}_2} \cdots \xrightarrow{\mathsf{a}_n} Q'.$$

Example 6. Consider the process depicted in Figure 4. E_6 describes the usual two place buffer. E_2 already appeared in Figure 1. Depending on the in branch taken E_2 may loose the possibility to output the first input before a second is accepted. However, both processes are trace equivalent according to Definition 6.

This example gives some insight into the weaknesses of trace equivalence. The process E_2 is not always able to release an input after it has accepted one whereas E_6 always is. This is problematic if E_2 is put in a context where an output is required for synchronisation after every input, (with E_1 of Figure 1, for instance, synchronising on action out and in). E_6 would be able to interact on action out after action in has occurred, whereas E_2 may not, thus forcing a deadlock. In other words, trace equivalence does not preserve deadlocks.

The main reason why trace equivalence is suitable for automata theory, while it does not fit with the process algebraic theory of processes, is the difference between their models of interaction. Automata theory assumes complete control of the automaton over its transitions. In the process algebraic view of processes all observable actions are under the *joint* control of the process and its environment. In this context automata can be seen as processes with only internal actions (but not necessarily labelled with τ), or alternatively, as a process with a completely cooperative environment, i.e. one that is always capable of synchronising on the action of the automatons' choice. The standard interaction between process algebraic processes, however, assumes an interactive resolution of choices, at least between observable transitions. This means that a process cannot select a transition labelled with an observable action if this action is not

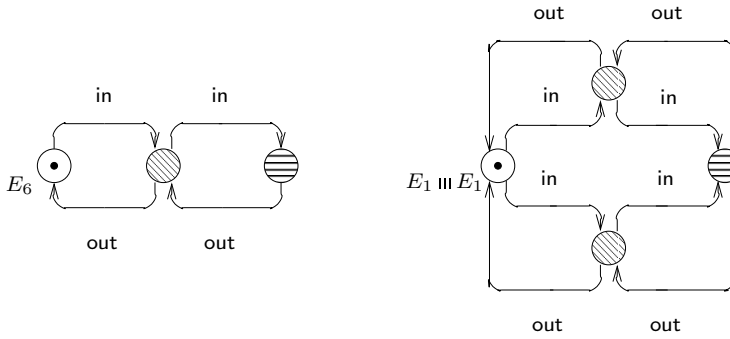


Fig. 5. Two strongly bisimilar processes.

also enabled by the environment. If several such jointly enabled transitions exist, then the choice is made nondeterministically.

In the presence of concurrent composition it is natural that two transition systems should be equivalent if and only if they interact in the same way with arbitrary environments. In view of the above, that means the way in which they constrain the choices between different actions is relevant. This is also referred to as the *branching (time) structure* of processes, as opposed to the *linear (time) structure* of classical automata.

Milner and Park [40,47] have introduced the most important class of equivalence relations that respect the branching structure of a process and therefore are deadlock preserving. This is the class of *bisimulation equivalences* or *bisimilarities*. Two processes are bisimilar if they can simulate each other's behaviour step-by-step. This leads to an inductive definition of bisimulation, based on single steps, that is simple but quite powerful.

Definition 7. *A binary relation $\mathcal{B}$ on PA^c is a strong bisimulation if $(P,Q) \in \mathcal{B}$ implies for all $\mathsf{a} \in \mathcal{A}_\tau$:*

- $P \xrightarrow{\mathsf{a}} P'$ *implies* $Q \xrightarrow{\mathsf{a}} Q'$ *for some Q' such that* $(P',Q') \in \mathcal{B}$,
- $Q \xrightarrow{\mathsf{a}} Q'$ *implies* $P \xrightarrow{\mathsf{a}} P'$ *for some P' such that* $(P',Q') \in \mathcal{B}$.

Two processes, P and Q, are strongly bisimilar, written $P \sim Q$, if they are contained in some strong bisimulation $\mathcal{B}$,i.e. $(P,Q) \in \mathcal{B}$.

Strong bisimilarity, therefore, is the union of all strong bisimulations, i.e.

$$\sim = \bigcup \{\mathcal{B} \mid \mathcal{B} \text{ is a strong bisimulation}\}$$

In words, the above definition says that two states in a transition system are bisimilar if each has transitions labelled with the same actions as the other such that the states after corresponding transitions are bisimilar again.

Example 7. Following Definition 7, the two processes E_6 and $E_1 \,|||\, E_1$ depicted in Figure 5 are bisimilar. To facilitate the inspection of this claim we have shaded

strongly bisimilar states with the same pattern. Note that the right process is obtained by composing in parallel two (one-place) buffers E_1 without any synchronisation. They behave like a two-place buffer E_6 (in fact, they implement a two-place *bag*, but because the data have no identity this is indistinguishable from a buffer).

Strong bisimilarity gives us appropriate means to compare processes with respect to their branching structure. Besides its intuitive content, it also has the correct formal properties that allow for a smooth mathematical treatment.

Proposition 1. *Strong bisimilarity is*

- *an equivalence relation on* PA^c.
- *a strong bisimulation on* PA^c.
- *the* largest *strong bisimulation on* PA^c.

The style of the definition of bisimulation is sometimes called *coinductive*, since it borrows the concept of coinduction from category theory [31]. Roughly speaking, a coinductive definition characterises the largest set satisfying an inductive definition, whereas *in*duction characterises the smallest such set.

Later, we will later also rely on an alternative characterisation of strong bisimilarity, borrowed from [57], which defines $\sim$ as the union of equivalence relations. It makes use of a (boolean) function $\gamma_{\mathrm{o}} : S \times \mathcal{A}_\tau \times 2^S \mapsto \{\mathtt{true}, \mathtt{false}\}$. $\gamma_{\mathrm{o}}(P, \mathsf{a}, C)$ is true iff P can evolve to a state contained in a set of states C by interaction on a.

Definition 8.

$$\gamma_{\mathrm{o}}(P, \mathsf{a}, C) := \begin{cases} \mathtt{true} & \textit{if there is } P' \in C \textit{ such that } P \xrightarrow{\mathsf{a}} P', \\ \mathtt{false} & \textit{otherwise.} \end{cases}$$

With this definition, bisimilarity can be expressed as 'having the same possibilities to interact and make a transition into the same class of behaviours' where these classes are, of course, classes of equivalent behaviour.

Lemma 1. *An equivalence relation $\mathcal{E}$ on S is a strong bisimulation iff $(P, Q) \in \mathcal{E}$ implies for all $\mathsf{a} \in \mathcal{A}_\tau$ and all equivalence classes C of $\mathcal{E}$ that*

$$\gamma_{\mathrm{o}}(P, \mathsf{a}, C) \implies \gamma_{\mathrm{o}}(Q, \mathsf{a}, C).$$

Note that $\mathcal{E}$ is presupposed to be an equivalence relation in this definition, and therefore is symmetric by assumption. Thus, we could equally well replace ' $\Longrightarrow$ ' by ' $\Longleftrightarrow$ ' (or '=') in the above Lemma.

Example 8. If we use  to denote the set of states shaded like  in Figure 5, and similar with  and  , then each of these sets is a class of an equivalence relation $\mathcal{E}$ satisfying Lemma 1. In particular, we compute the following values for each of the states in the respective class. All other combinations return $\mathtt{false}$.

$\gamma_{\mathrm{o}}($  $, \mathsf{in},$  $) = \mathtt{true}$ $\qquad$ $\gamma_{\mathrm{o}}($  $, \mathsf{out},$  $) = \mathtt{true}$

$\gamma_{\mathrm{o}}($  $, \mathsf{in},$  $) = \mathtt{true}$ $\qquad$ $\gamma_{\mathrm{o}}($  $, \mathsf{out},$  $) = \mathtt{true}$

Let us now turn our attention to another important property of bisimilarity. Since we work in a setting with composition operators, we must investigate whether $\sim$ induces a proper algebraic notion of equality. In particular, this means that equality should be preserved under composition. In the above example, we have seen that E_6 and $E_1 \,|||\, E_1$ are bisimilar. They both describe the behaviour of a buffer with two places. However, it is not yet clear whether we can use either of them in a larger composition context and obtain again equivalent overall behaviours. This is, of course, a highly desirable property, because it allows normal equational reasoning, replacing a subterm by an equivalent one, without affecting the resulting behaviour. What we need, in general, is the *substitutivity* of an equivalence relation. In algebraic terms this means that we have to show that $\sim$ is a *congruence* (relation) with respect to the operators.

Theorem 1. *Strong bisimilarity is a congruence relation with respect to the operators of* PA^c*, i.e.*

$$
\begin{array}{lcll}
P_1 \sim P_2 & \textit{implies} & \mathsf{a}.P_1 \sim \mathsf{a}.P_2 \\
P_1 \sim P_2 & \textit{implies} & P_1 + P_3 \sim P_2 + P_3, \\
P_1 \sim P_2 & \textit{implies} & P_3 + P_1 \sim P_3 + P_2, \\
P_1 \sim P_2 & \textit{implies} & P_1 \,|[\mathsf{a}_1 \ldots \mathsf{a}_n]|\, P_3 \sim P_2 \,|[\mathsf{a}_1 \ldots \mathsf{a}_n]|\, P_3, \\
P_1 \sim P_2 & \textit{implies} & P_3 \,|[\mathsf{a}_1 \ldots \mathsf{a}_n]|\, P_1 \sim P_3 \,|[\mathsf{a}_1 \ldots \mathsf{a}_n]|\, P_2, \\
P_1 \sim P_2 & \textit{implies} & \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P_1 \sim \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P_2\,.
\end{array}
$$

Strong bisimilarity shares this substitutivity property with other equivalences, such as trace equivalence. On top of that, it respects the branching structure of processes and therefore preserves deadlocks. Furthermore, it can be defined coinductively. These properties are the main reasons why bisimilarity is a central concept in the theory of process algebraic equivalences. It is easy to define, has a simple proof technique, and is mathematically elegant.

Weak equivalences. So far we have only discussed equivalences that treat internal actions exactly the same way as external actions. In particular, internal actions have to be simulated stepwise to establish strong bisimilarity between two processes. This is counterintuitive, because we ultimately mean to characterise the behaviour of processes by means of their black box, i.e. observable, behaviour. But as internal actions are not observable there seems to be no direct need to be able to simulate each internal transition of an equivalent process.

Example 9. We have discussed before that a serial connection of two one-place buffers as in $\texttt{hide mid in}\ \ E_3 \,|[\mathsf{mid}]|\, E_5$ behaves like a two-place buffer. However, it is not possible to construct a strong bisimulation between this process and E_6 even though E_6 appears as a canonical representation of a two place buffer. The reason is that we have to (bi)simulate internal $\stackrel{\tau}{\longrightarrow}$ transitions that E_6 does not possess (Figure 6).

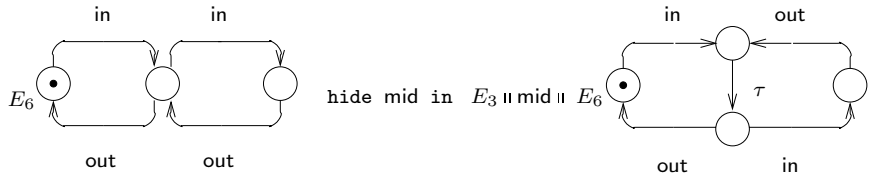


Fig. 6. Not strongly bisimilar processes.

To abstract from internal moves it seems natural to ignore them as far as they do not influence the observable behaviour of a process. To do so, we introduce the notion of an *observable step* of a process, consisting of a single *observable* action preceded and followed by an arbitrary finite number (including zero) of internal steps [40]. This can be seen as deriving a 'weak' transition relation, denoted by $\Longrightarrow$, from the 'strong' transition relation $\longrightarrow$.

Definition 9. *For internal actions,* $\overset{\varepsilon}{\Longrightarrow}$ *is defined as the reflexive and transitive closure* $\overset{\tau}{\longrightarrow}^*$ *of the relation* $\overset{\tau}{\longrightarrow}$*. External weak transitions are then obtained by defining* $\overset{\mathsf{a}}{\Longrightarrow}$ *to denote* $\overset{\varepsilon}{\Longrightarrow} \overset{\mathsf{a}}{\longrightarrow} \overset{\varepsilon}{\Longrightarrow}$*.*

Note that a weak internal transition $\overset{\varepsilon}{\Longrightarrow}$ is possible without actually performing an internal action, because $\overset{\tau}{\longrightarrow}^*$ contains the reflexive closure, i.e. the possibility not to move at all. In contrast, a weak external transition $\overset{\mathsf{a}}{\Longrightarrow}$ must contain exactly one transition $\overset{\mathsf{a}}{\longrightarrow}$ preceded and followed by arbitrary (possibly empty) sequences of internal moves. We use $\mathcal{A}_\varepsilon$ to range over visible actions and ε, i.e. $\mathcal{A}_\varepsilon = \mathcal{A} \cup \{\varepsilon\}$.

Example 10. For the processes E_6 and $\mathtt{hide\ mid\ in}\ \ E_3 \,|\!|\, \mathsf{mid} \,|\!|\, E_5$ the weak transition relation is represented by the arrows in Figure 7.

With this relation, weak trace equivalence and weak bisimilarity are obtained by simply replacing strong by weak transitions in Definition 6 and Definition 7, respectively. Since weak trace equivalence inherits the problems of its strong counterpart, we are not interested in this relation here.

Definition 10. *A binary relation* $\mathcal{B}$ *on* PA^c *is a weak bisimulation if* $(P, Q) \in \mathcal{B}$ *implies for all* $\mathsf{a} \in \mathcal{A}_\varepsilon$:

- $P \overset{\mathsf{a}}{\Longrightarrow} P'$ *implies* $Q \overset{\mathsf{a}}{\Longrightarrow} Q'$ *for some* Q' *such that* $(P', Q') \in \mathcal{B}$,
- $Q \overset{\mathsf{a}}{\Longrightarrow} Q'$ *implies* $P \overset{\mathsf{a}}{\Longrightarrow} P'$ *for some* P' *such that* $(P', Q') \in \mathcal{B}$.

Two processes, P *and* Q*, are weakly bisimilar, written* $P \approx Q$*, if they are contained in some weak bisimulation* $\mathcal{B}$*.*

Weak bisimilarity has the same basic properties as strong bisimilarity (cf. Proposition 1).

Proposition 2. *Weak bisimilarity is*

- *an equivalence relation on* PA^c.
- *a weak bisimulation on* PA^c.
- *the* largest *weak bisimulation on* PA^c.

In addition it is a congruence relation for all operators, except for the choice operator.

Lemma 2. *Weak bisimilarity is a congruence with respect to prefix, parallel composition and abstraction, but not with respect to choice.*

In order to illustrate that $\approx$ is not a congruence with respect to choice we consider the following counterexample [40]. By Definition of $\approx$ it is obvious that

$$\tau.\mathsf{a}.0 \;\approx\; \mathsf{a}.0$$

holds. Supposing that $\approx$ is a congruence with respect to choice, we can conclude that

$$\underbrace{\tau.\mathsf{a}.0 + \mathsf{b}.0}_{P} \;\approx\; \underbrace{\mathsf{a}.0 + \mathsf{b}.0}_{Q}$$

must also hold. But in P there is a transition labelled τ to $\mathsf{a}.0$. In other words, $P \stackrel{\varepsilon}{\Longrightarrow} \mathsf{a}.0$. In order to satisfy Definiton 10, there need to be some Q' with $Q \stackrel{\varepsilon}{\Longrightarrow} Q'$, satisfying $\mathsf{a}.0 \approx Q'$. But this is not the case. The only candidate for Q' – Q itself – obviously differs from $\mathsf{a}.0$. Thus the assumed congruence property turns out to be false.

The problem is that initial internal transitions need to be treated slightly stronger. To heal this problem, we refine weak bisimilarity.

Definition 11. *P and Q are weakly congruent, written $P \approx^c Q$ iff for all $\mathsf{a} \in \mathcal{A}_\tau$*

1. *$P \stackrel{\mathsf{a}}{\longrightarrow} P'$ implies $Q \stackrel{\varepsilon}{\Longrightarrow} \stackrel{\mathsf{a}}{\longrightarrow} \stackrel{\varepsilon}{\Longrightarrow} Q$ for some Q' with $P' \approx Q'$,*
2. *$Q \stackrel{\mathsf{a}}{\longrightarrow} Q'$ implies $P \stackrel{\varepsilon}{\Longrightarrow} \stackrel{\mathsf{a}}{\longrightarrow} \stackrel{\varepsilon}{\Longrightarrow} P'$ for some P' with $P' \approx Q'$.*

Weak congruence and weak bisimilarity only differ in the treatment of initial internal steps of P and Q. Weak bisimilarity requires that an internal transition $\stackrel{\tau}{\longrightarrow}$ is simulated by a weak transition $\stackrel{\varepsilon}{\Longrightarrow}$, which includes the possibility that no internal transition has to be carried out (cf. Definition 9). For initial behaviours, weak congruence strengthens this requirement. It requires that an internal transition $\stackrel{\tau}{\longrightarrow}$ has to be matched by $\stackrel{\tau}{\longrightarrow}^* \stackrel{\tau}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^*$, i.e. by at least on internal transition $\stackrel{\tau}{\longrightarrow}$.

Theorem 2. *Weak congruence is substitutive with respect to the operators of* PA^c*, i.e.*

$$\begin{array}{lll}
P_1 \simeq P_2 & \textit{implies} & \mathsf{a}.P_1 \simeq \mathsf{a}.P_2 \\
P_1 \simeq P_2 & \textit{implies} & P_1 + P_3 \simeq P_2 + P_3, \\
P_1 \simeq P_2 & \textit{implies} & P_3 + P_1 \simeq P_3 + P_2, \\
P_1 \simeq P_2 & \textit{implies} & P_1 \,|\!|_{\mathsf{a}_1 \ldots \mathsf{a}_n}\, P_3 \simeq P_2 \,|\!|_{\mathsf{a}_1 \ldots \mathsf{a}_n}\, P_3, \\
P_1 \simeq P_2 & \textit{implies} & P_3 \,|\!|_{\mathsf{a}_1 \ldots \mathsf{a}_n}\, P_1 \simeq P_3 \,|\!|_{\mathsf{a}_1 \ldots \mathsf{a}_n}\, P_2, \\
P_1 \simeq P_2 & \textit{implies} & \mathtt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \mathtt{in}\ P_1 \simeq \mathtt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \mathtt{in}\ P_2\ .
\end{array}$$

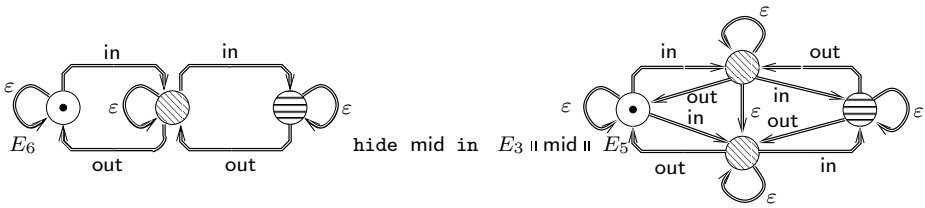


Fig. 7. Two weakly congruent processes.

Indeed, weak congruence is unique in the sense that it turns out to be the *coarsest* congruence contained in weak bisimilarity, as a consequence of the following lemma.

Lemma 3. *$E_1 \simeq E_2$ iff, for each $E_3 \in \mathsf{IMC}^c$, $E_1 + E_3 \approx E_2 + E_3$ and $E_3 + E_1 \approx E_3 + E_2$.*

As a result, we have obtained two substitutive equivalence notions on PA^c: strong bisimilarity and weak congruence, a distinguished subset of weak bisimilarity. The interrelation between these equivalences is expressed in the following lemma.

Lemma 4. $\sim \subset \simeq \subset \approx$.

Example 11. We have pointed out that the processes E_6 and $\mathtt{hide\ mid\ in}\ \ E_3 \,|\!|\, \mathsf{mid} \,|\!|\, E_5$ are not strongly bisimilar. But they are weakly bisimilar according to Definition 10. To illustrate this, Figure 7 shows bisimilar states shaded with the same pattern. A crucial aspect is that the weak internal transition $\overset{\varepsilon}{\Longrightarrow}$ of the right process can be simulated by the left process because $\overset{\varepsilon}{\Longrightarrow}$ contains the reflexive closure.

This example shows that weak congruence is an appropriate notion to compare the behaviour of components when internal actions are present. Furthermore, it is indeed a congruence, it is defined coinductively, and it preserves *observable* deadlocks, i.e. the (in)capacity in a state to execute weak transitions labelled by an observable action.

Nevertheless, weak congruence is not as undisputed among the vast number of weak relations as strong bisimilarity is among the strong relations. Some, e.g. van Glabbeek&Weijland [54] and Montanari&Sassone [41] point out that weak bisimilarity is too coarse to preserve the precise branching structure of a process. Others, like Darondeau [14], Valmari [53], de Nicola&Hennessy [44], Parrow&Sjödin [48], Cleaveland&Natarjan [42], as well as Brinksma *et al.* [9] define again coarser equivalences and argue that these relations characterise the observable behaviour of processes better than $\simeq$ does.

It may be worth to point out that it is desirable to have an equivalence notion that is as coarse as possible, given the criteria for equivalent behaviour (in the form of required preservation properties, for example). An equivalence that is too fine will distinguish between too many processes, and therefore satisfy fewer equations, making verification of certain systems more difficult, if not impossible. From this point of view, (fair) testing equivalences seem to be the right choice [10,49,35,44,42,9] – if one is interested in the preservation of observable deadlocks. Essentially, each of the proposed relations is the coarsest in its category, each corresponding to a natural scenario of what an observer is able to test. However, we will not treat them here, basically they do not have coinductive definitions, which we will need for our stochastic extensions later.

2.5 Algebra of Processes

The previous section has illustrated the usefulness of congruences, i.e. equivalences that are substitutive with respect to the language operators. In the presence of such a congruence, it is interesting to investigate in which sense the congruence can be characterised on PA^c by a set of equational laws.

Example 12. An example of an equational law is the commutativity law $E+F = F+E$. The intuitive meaning of this law is as follows: Whenever a pattern of the form $E+F$ can be found in an expression, it can be replaced by $F+E$. E and F play the roles of meta variables and can be instantiated by arbitrary expressions of PA^c. In this way we may transform $\mathsf{b}.(\mathsf{a}.0 + \mathsf{c}.\mathsf{d}.0)$ into $\mathsf{b}.(\mathsf{c}.\mathsf{d}.0 + \mathsf{a}.0)$: We instantiate $E \equiv \mathsf{a}.0$ and $F \equiv \mathsf{c}.\mathsf{d}.0$ and afterwards substitute $F + E$ for $E + F$.

Formally, an equational law is a pair of expressions of PA^c connected with the symbol '=' where either of the expressions, may contain some 'meta variables' such as E, F, and so on. In technical terms, a law (or a set of laws) induces an equivalence on PA^c, or more precise a *congruence*, since we are allowed to replace sub-expressions inside larger expressions, as in the above example.

The question arises in what sense such an induced congruence is related to the congruences we have defined on the semantics of PA^c, i.e. strong bisimilarity and weak congruence. Indeed we are aiming to provide laws that are *sound* with respect to, say, strong bisimilarity. A law is sound with respect to an equivalence, if any application of the law does not alter the equivalence class of the expression. The converse direction is called *completeness*. A set of laws is complete with respect to an equivalence, if two expressions can be transformed into each other by (iterative) application of laws whenever they are equivalent.

Example 13. The law $0 = 0$ is sound for any equivalence relation on PA^c. However, this law is, as it stands alone, far from providing a complete set of laws for any nontrivial equivalence. On the other hand, a law $E = F$ is complete for any equivalence relation on PA^c, but it is sound only for the trivial relation $\mathsf{PA}^c \times \mathsf{PA}^c$ that equates all processes.

(C)	$E + F = F + E$
(A)	$(E + F) + G = E + (F + G)$
(I)	$E + E = E$
(N)	$E + 0 = E$

Table 4. Axioms for strong bisimilarity.

(C)	$E + F = F + E$
(A)	$(E + F) + G = E + (F + G)$
(I)	$E + E = E$
(N)	$E + 0 = E$
$(\tau 1)$	$\mathsf{a}.\tau.E = \mathsf{a}.E$
$(\tau 2)$	$E + \tau.E = \tau.E$
$(\tau 3)$	$\mathsf{a}.(E + \tau.F) + \mathsf{a}.F = \mathsf{a}.(E + \tau.F)$

Table 5. Axioms for weak congruence.

So, our ultimate goal is to provide sets of laws that are *sound as well as complete* with respect to strong bisimilarity, respectively weak congruence. We shall say that such a set *axiomatises* the respective congruence. This is what turns the set PA^c into a true *algebra*.

To give a flavor of this algebraic view on PA^c, Table 4 lists the main equational laws axiomatising strong bisimilarity. The laws state that the choice operator is commutative (C), associative (A), idempotent (I), and that 0 is the neutral element of choice (N). There are four more laws needed to handle recursion and we refer to [38] or [19] for a detailed explanation.

Turning our attention to weak congruence, Table 5 presents a set of laws that form the core of an axiomatisation of weak congruence on PA^c. The upper part of these laws is literally copied from Table 4. This should not be surprising, because strong bisimilarity is a subset of weak congruence (cf. Lemma 4) and therefore every pair that can be proven to be strongly bisimilar has to be weakly congruent, as well. This is a striking reason why the axiomatisation of weak congruence is an extension of the axiomatisation of strong bisimilarity. The law $(\tau 1)$ allows one to skip (action guarded) internal steps. Law $(\tau 2)$ and $(\tau 3)$ expresses that certain behaviours that are preceded by an internal step can happen instantly provided a τ-guarded copy persists.

We shall now discuss the additional operators we have defined on PA^c, abstraction and parallel composition. We present a set of additional laws, that allow one to rewrite parallel composition, as well as abstraction into the basic operators of PA^c. Table 6 lists the necessary laws. Law (X) is usually called the *expansion law*. It states that non-synchronising actions of components can be simply interleaved. Either the left ($\mathsf{a}_j \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$), or the right component

$$(X)\ \sum \mathsf{a}_j.P_j \between \mathsf{a}_1 \ldots \mathsf{a}_n \between \sum \mathsf{b}_l.Q_l = \sum_{\mathsf{a}_j \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}} \mathsf{a}_j.(P_j \between \mathsf{a}_1 \ldots \mathsf{a}_n \between Q) + \sum_{\mathsf{b}_l \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}} \mathsf{b}_l.(P \between \mathsf{a}_1 \ldots \mathsf{a}_n \between Q_l) + \sum_{\mathsf{a}_j = \mathsf{b}_l \in \{\mathsf{a}_1 \ldots \mathsf{a}_n\}} \mathsf{a}_j.(P_j \between \mathsf{a}_1 \ldots \mathsf{a}_n \between Q_l)$$

$$(H1)\quad \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ 0 = 0$$

$$(H2)\quad \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ \mathsf{a}.P = \mathsf{a}.\texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P \quad \textit{provided}\ \mathsf{a} \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

$$(H3)\quad \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ \mathsf{a}.P = \tau.\texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P \quad \textit{provided}\ \mathsf{a} \in \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$$

$$(H4)\quad \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P + Q = \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ P + \texttt{hide}\ \mathsf{a}_1 \ldots \mathsf{a}_n\ \texttt{in}\ Q$$

Table 6. Axioms for rewriting parallel composition and abstraction.

($\mathsf{b}_l \notin \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$) performs a non-synchronising action. In case of synchronisation ($\mathsf{a}_j = \mathsf{b}_l \in \{\mathsf{a}_1 \ldots \mathsf{a}_n\}$), both partner evolve further.

The laws $(H1) - (H4)$ are very simple. They say that abstraction distributes over termination, over choice and over action prefix, where, according to $(H3)$ action a is internalised if it appears in the set $\{\mathsf{a}_1 \ldots \mathsf{a}_n\}$ of actions. With these laws, parallel composition and abstraction can be shifted arbitrarily deep into a specification, until either 0 or some variable X is reached. This is enough to ensure completeness for a language that includes abstraction and parallel composition (but where the use of recursion is restricted, see e.g. [19]).

This concludes our brief summary how bisimulation can be characterised axiomatically. These axioms are particularly handy to reason about PA^c in an abstract fashion. One can capture the essence of the language just by agreeing (or disagreeing) with a particular set of axioms. It is important to mention that a highly influential strand of process algebra research (also known as the *Dutch school* [1,3]) proceeds the other way round than the way we chose here. This school presupposes a specific equational theory, and then investigates the models and equivalences needed to match this theory. We will follow this way in Section 4 when we introduce an algebra of Interactive Markov Chains.

3 Markov Models

Continuous time Markov chains (MCs) are a particular class of stochastic models that forms a cornerstone of contemporary performance and dependability evaluation methodology [25,18]. This section reviews the main ingredients of MCs from an algebraic perspective, i.e. we proceed similar to the preceding section. After introducing Markov chains and their basic properties, we discuss a bisimulation style equivalence on such chains, which is also known as *lumpability*. We discuss equational properties of this equivalence by developing a small algebra

of MCs, to illustrate the relation to standard process algebra. Broad background material on Markov chains and their analysis can be found in [18].

3.1 Continuous Time Markov Chains

A continuous time Markov chain is a stochastic process $\{X(t) \mid t \in \mathbb{R}\}$ with discrete state space satisfying the so called Markov property. This means that the random variable X takes values of some discrete set S (the state space), and the values of X vary continuously as time passes, satisfying that for $t_n + \Delta t > t_n > t_{n-1} > t_{n-2} > \ldots > t_0$,

$$\begin{aligned} &Prob\{X(t_n + \Delta t) = P' \mid X(t_n) = P, X(t_{n-1}) = P_{t_{n-1}}, \ldots, X(t_0) = P_{t_0}\} \\ &= Prob\{X(t_n + \Delta t) = P' \mid X(t_n) = P\} \\ &= Prob\{X(\Delta t) = P' \mid X(0) = P\}. \end{aligned}$$

Thus, the fact that the process was in state P_{n-1} at time t_{n-1}, in state P_{n-2} at time t_{n-2}, and so on, up to the fact that it was in state P_0 at time t_0 is completely irrelevant. The state $X(t_n)$ contains all relevant history information to determine the random distribution on S at time t_{n+1}. This probability is independent of the actual time instant t_n (or t' or 0) of observation. Nevertheless it *does* depend on the length of the time *interval* Δt. It requires some limit calculation to deduce that we are facing a *linear dependence* [34]. More precise, for every pair of states P and P', there is some parameter λ such that (for small Δt)

$$Prob\{X(\Delta t) = P' \mid X(0) = P\} = \lambda \Delta t + o(\Delta t)$$

where $o(\Delta t)$ subsumes the probabilities to pass through intermediate states between P and P' during the interval Δt. The quantity λ is thus a transition *rate*, a nonnegative real value that scales how the (one step) transition probability between P and P' increases with time. Here, we have implicitly assumed that state P is different from P'. If, otherwise, state P and P' coincide, the probability to stay in state P during an interval Δt (and hence $Prob\{X(\Delta t) = P \mid X(0) = P\}$) decreases with time, starting from 1 if $\Delta t = 0$. The corresponding transition rate is thus a negative real value. It is implicitly determined by the increasing probability to leave state P; that is, it is the negative sum of the respective transition rates.

The probabilistic behaviour of a MC is completely described by the initial state occupied at time 0 (or an initial probability distribution on S) and the transition rates between distinct states. We therefore fix a MC by means of a specific transition relation, $P \stackrel{\lambda}{\longrightarrow} P'$, defined on a state space S, together with an initial state P.

Definition 12. *A Markov transition system is a tuple* $(S, \longrightarrow)$*, where* S *is a nonempty set of states, and* $\longrightarrow$ *is a* Markov transition relation*, a subset of* $S \times \mathbb{R}^+ \times S$*. A Markov chain is a triple* $(S, \longrightarrow, P)$*, where* $(S, \longrightarrow)$ *is a Markov transition system, and* $P \in S$ *is the initial state.*

Fig. 8. Two Markov chains

Example 14. Figure 8 contains two examples of Markov chains, E_{30} and E_{31}.

The time of staying in a particular state of S is a random variable, usually called sojourn time. The sojourn time for any state of a MC is known to be *exponentially distributed*. We highlight the following important properties enjoyed by exponential distributions. Let D, D_1, and D_2 denote exponentially distributed random variables.

(A) An exponential distribution $Prob\{D \leq t\} = 1 - e^{-\lambda t}$ is characterised by a single parameter λ, a positive real value, usually referred to as the *rate* of the distribution. The mean duration of this delay amounts to $1/\lambda$ time units.

(B) The class of exponential distribution is the only class of *memoryless* continuous probability distribution. The remaining delay after some time t_0 has elapsed is a random variable with the same distribution as the whole delay:

$$Prob\{D \leq t + t_0 \mid D > t_0\} \;=\; Prob\{D \leq t\}. \tag{1}$$

(C) The class of exponential distributions is closed under minimum, which is exponentially distributed with the sum of the rates:

$$Prob\{\min(D_1, D_2) \leq t\} = 1 - e^{-(\lambda_1+\lambda_2)t} \tag{2}$$

if D_1 (D_2, respectively) is exponentially distributed with rate λ_1 (λ_2).

(D) The probability that D_1 is smaller than D_2 (and vice versa) can be directly derived from the respective rates:

$$Prob\{D_1 < D_2\} = \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{3}$$

$$Prob\{D_2 < D_1\} = \frac{\lambda_2}{\lambda_1 + \lambda_2}. \tag{4}$$

(E) The continuous nature of exponential distributions ensures that the probability that both delays elapse at the same time instant is zero.

Property (C) explains why the sojourn time for a state s is exponentially distributed. Every transition $s \xrightarrow{\lambda} s'$ leaving state s can be seen to have an exponentially distributed random variable (with parameter λ) associated that

governs when this transition may happen. A race is assumed to exist between several transitions, i.e., they compete for a state change. The sojourn time in s ends as soon as the first transition is ready to occur, inducing a state change. Due to property (C) this sojourn time is exponentially distributed with the sum of the rates of the transitions involved. Property (D) determines the probability of a specific transition to win such a race.

3.2 Equivalence

Strong and weak bisimilarities, as introduced in Section 2, are central in the theory of process algebraic equivalences. Apart from their theoretical importance, a practical merit is the possibility of behaviour preserving state space aggregation. This is achieved by neglecting the identity of states in favour of equivalence classes of states exhibiting identical behaviours. We follow the same spirit in the context of Markov chains.

Strong equivalence. For a given chain, assume that we are only interested in probabilities of equivalence classes of states with respect to some equivalence $\sim$ (that we are aiming to define) instead of probabilities of states. Any such equivalence preserving view on a Markov chain gives rise to an aggregated stochastic process $\widetilde{X} = \{\widetilde{X}(t) | t \in T\}$. It can be defined on the state space $S/\!\sim$, the set of the equivalence classes with respect to $\sim$, by

$$Prob\{\widetilde{X}_t = C\} := Prob\{X(t) \in C\} \qquad \text{for each } C \in S/\!\sim. \tag{5}$$

$\widetilde{X}$ is a discrete state space stochastic process, but it is not necessarily a MC. However sufficient conditions exist such that $\widetilde{X}$ is again a time homogeneous MC. They impose restrictions on the shape of the sets C and are known as lumping conditions, see [34]. We approach them from a different viewpoint, namely by constraints on the equivalence $\sim$, similar to [11,27]. Anticipating the technical details, we achieve that $\widetilde{X}$ is a MC, if $\sim$ is a variant of bisimulation. The difficulty is that we have to equate not only qualities but also *quantities*, for example transition rates of moving from one state to an equivalence class. In contrast, bisimilarity only talks about a (logical) quality: Either there is a move from a state into a class possible or it is impossible, but *tertium non datur*.

The bridge to *quantify* strong bisimilarity is an alternative characterisation of strong bisimilarity that we have mentioned as Lemma 1. To recall its essentials, note that it uses a predicate $\gamma_{\text{O}} : S \times \mathcal{A}_\tau \times 2^S \mapsto \{\texttt{true}, \texttt{false}\}$ that is true iff P can evolve to a state contained in a set of states C (by interaction on action a). Bisimilarity then occurs as the union of all equivalence relations that equate two states if they posses the same γ_{O} values (for each possible combination of action a and equivalence class C).

We follow this style of definition but replace the predicate γ_{O} by a (nonnegative) real-valued function $\gamma_{\text{M}} : S \times 2^S \mapsto \mathbb{R}^+$, that calculates the cumulative rate to reach a set of states C from a single state R:

$$\gamma_{\text{M}}(R, C) = \sum \{\!|\lambda | R \xrightarrow{\lambda} R' \text{ and } R' \in C|\!\}.$$

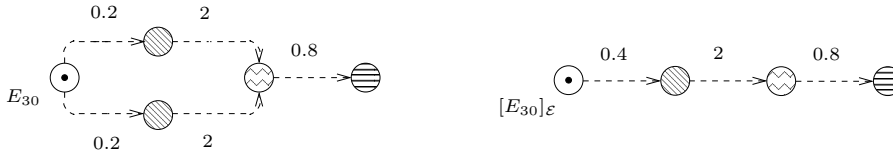


Fig. 9. A Markov chain and its aggregated representative

In this definition we let $\sum \{\!|\ldots|\!\}$ denote the sum of all elements in a multiset (of transition rates), where $\{\!|\ldots|\!\}$ delimits this multiset. The need for this notational burden is best explained by means of an example.

Example 15. Considering Figure 8, the cumulative rate to reach any state in S from state E_{30} is $\gamma_{\mathrm{M}}(E_{30}, S) = \sum \{\!|0.2, 0.2|\!\}$ which amounts to 0.4 due to our definition.

We are now ready to lift bisimilarity to the setting of Markov chains.

Definition 13. *For a given Markov chain $(S, \longrightarrow, P)$, an equivalence relation $\mathcal{E}$ on S is a Markov bisimulation iff $P\mathcal{E}Q$ implies that for all equivalence classes C of $\mathcal{E}$ it holds that*

$$\gamma_{\mathrm{M}}(P, C) \leq \gamma_{\mathrm{M}}(Q, C)$$

Two states P and Q are Markov bisimilar, written $P \sim_{\mathrm{M}} Q$, if (P, Q) is contained in some Markov bisimulation $\mathcal{E}$.

Thus $\sim_{\mathrm{M}}$ is the union of all such equivalences. Indeed, it is itself a Markov bisimulation and therefore the largest such relation (Note that as in Lemma 1 the relation $\mathcal{E}$ is presupposed to be an equivalence, and thus we could write '=' instead of '$\leq$').

Definition 14. *For a given Markov chain $(S, \longrightarrow, P)$ and a Markov bisimulation $\mathcal{E}$ on S, define an aggregated chain $(S/\mathcal{E}, \longrightarrow_{\mathcal{E}}, [P]_{\mathcal{E}})$ where the Markov transition relation $\longrightarrow_{\mathcal{E}}$ is given by*

$$[P']_{\mathcal{E}} \xrightarrow{\lambda}_{\mathcal{E}} [Q']_{\mathcal{E}} \qquad \textit{iff} \qquad \gamma_{\mathrm{M}}(P', [Q']_{\mathcal{E}}) = \lambda.$$

Example 16. With the notation introduced in Chapter 2 each of the sets  ,  ,  and  appearing in Figure 9 is a class of an equivalence relation $\mathcal{E}$ on the state space of E_{30} satisfying Definition 13. In particular, we compute the values

$\gamma_{\mathrm{M}}($ $) = 0.4$ $\qquad$ $\gamma_{\mathrm{M}}($ $) = 2$ $\qquad$ $\gamma_{\mathrm{M}}($  $) = 0.8$

for the states in the respective classes, all other values of γ_{M} are zero. The aggregated Markov chain $[E_{30}]_{\mathcal{E}}$ obtained by applying Definition 14 is depicted on the right.

Theorem 3. *Let P be a Markov chain, describing the CTMC X and let $\mathcal{E}$ be a Markov bisimulation on the state space of P. The aggregated chain $P_{\mathcal{E}}$ describes a homogeneous CTMC $\{\widetilde{X}(t) \mid t \in \mathbb{R}\}$ such that for all equivalence classes C of $\mathcal{E}$,*

$$Prob\{\widetilde{X}(t) = C\} = Prob\{X(t) \in C\}.$$

Proof. The conditions imposed on a Markov bisimulation can be matched with the definition of lumpability [34], see [27].

As a particular consequence, the stochastic process induced by factorising with respect to a Markov bisimulation is again a MC.

As mentioned above this kind of aggregation is known as *lumping*. Lumping is usually formulated with respect to a *suitable* partitioning of the state space. Here, we have defined a *suitability criterion* in a coinductive way. Our partitioning is obtained by means of a factorisation with respect to a bisimulation. This coinductive definition can be exploited for an algorithmic computation of the best possible lumping, see [24].

Weak equivalence. Hitherto we have studied only *strong* bisimilarity on Markov chains. It seems to be equally worthwhile to investigate *weak* bisimilarity. For this purpose, several questions have to be addressed.

First, what is the counterpart of a *weak transition* in terms of Markovian transitions? In the non-stochastic setting we have used a weak transition relation to successfully define weak bisimilarity. It has been based on the distinction between internal actions (labelled τ) and external, observable actions. Such a distinction is not obvious for Markovian chains, because there is no notion of interaction with the external environment.

We may therefore refuse to think about weak relations on Markovian chains at all. Alternatively we may decide that either none, or all of the Markovian transitions are internal. In the former case, a weak Markovian bisimulation will not differ from its strong counterpart, because there is no internal transition that could be abstracted away. So, what about assuming that all Markovian transitions are internal? The corresponding weak transition relation would then combine sequences of Markovian transitions into a single 'weak' transition, in the same way as $\stackrel{\varepsilon}{\Longrightarrow}$ combines sequences of $\stackrel{\tau}{\longrightarrow}$ transitions. For instance, a sequence $P \stackrel{\lambda}{-\!\!\rightarrow} P' \stackrel{\mu}{-\!\!\rightarrow} P''$ could be combined to a weak transition from P to P'' with a parameter ν. This parameter subsumes the exponentially distributed sojourn times in P and P', and it may, in general, be defined as a function $\phi(\lambda, \mu)$.

Unfortunately, the sequence of two (or more) exponentially distributed delays is no longer exponentially distributed. So, any particular choice of a function ϕ will introduce (a possibly severe) error in the model. In other words, replacing a sequence of Markovian transitions by a single *weak* Markovian transitions will lead to a CTMC where it is impossible to reconstruct the stochastic behaviour of the original chain. A result similar to Theorem 3 is thus not possible for any kind of weak Markovian bisimulation. Approximate solutions to this problem have been proposed, and we refer to [27] for a discussion of this topic.

$$\frac{}{(\lambda).E \xrightarrow{\lambda} E} \qquad \frac{E \xrightarrow{\lambda} E'}{E + F \xrightarrow{\lambda} E'} \qquad \frac{F \xrightarrow{\lambda} F'}{E + F \xrightarrow{\lambda} F'} \qquad \frac{E_i\{[X := E]_i / X_i\} \xrightarrow{\lambda} E'}{[X := E]_i \xrightarrow{\lambda} E'}$$

Table 7. Operational semantic rules for MC^c.

3.3 Algebra of Markov Chains

We now develop an algebraic view on Markov chains. To do so, we first define a small, action less algebra, that allows us to generate Markov chains.

Definition 15. *Let $\lambda \in \mathbb{R}^+$ and $X \in \mathcal{V}$. We define the language MC^c as the set of* closed *expressions given by the following grammar.*

$$\mathcal{E} ::= \quad 0 \quad | \quad (\lambda).\mathcal{E} \quad | \quad \mathcal{E} + \mathcal{E} \quad | \quad X \quad | \quad [X := \mathcal{E}]_i$$

The expression $(\lambda).E$, a *delay prefixed* expression, has the intuitive meaning that a process $(\lambda).P$ has to delay for some time before turning into the process P. The amount of time needed to delay is determined by λ, which serves as a parameter of an exponential distribution. In other words, the probability that $(\lambda).P$ has to wait less than t units of time before turning into process P equals $1 - e^{\lambda t}$.

The semantics of MC^c is depicted in Table 7, defining a Markov transition relation $\longrightarrow \subset \mathsf{MC}^c \times \mathbb{R}^+ \times \mathsf{MC}^c$ as the least multi-relation given by the rules. A particular expression E then gives rise to a Markov chain with initial state E and a discrete state space S, determined by the states reachable from E.

Note that, as before, an expression E like $(\lambda).E' + (\mu).E''$ has two alternatives. But different from Section 2 where the decision which alternative to take has been nondeterministic this is not the case here. Instead, the decision is governed by the probabilistic evolution of $(\lambda).E'$ and $(\mu).E''$, since a race is assumed to exist between the different branches. The sojourn time of E, i.e. the time until the state changes (to either E' or E'') is then exponentially distributed with rate $(\lambda + \mu)$. As a consequence of property (D) of exponential distributions, E' (resp. E'') will win the race with probability $\lambda/\lambda + \mu$ (probability $\mu/\lambda + \mu$).

In other words, imagine we want to add a probabilistic choice operator $\oplus_p$ – that selects its left-hand side with probability p, and it's right-hand side with $1-p$. We could indeed do so easily, as long as it is guarded by some delay prefix. We could define

$$(\lambda).(E' \oplus_p E'') = ((1-p)\lambda).E' + (p\lambda).E'' \tag{6}$$

to manifest the probabilistic effect that the operator '+' has due to the race condition.

This remark leads us to the algebraic properties of bisimilarity (or lumpability) in this context. From the above discussion, it is obvious that the laws in

(C)	$E + F = F + E$
(A)	$(E + F) + G = E + (F + G)$
(I')	$(\lambda).E + (\mu).E = (\lambda + \mu).E$
(N)	$E + 0 = E$

Table 8. Axioms for MC bisimilarity.

Table 4 cannot be valid in MC^c without change. The idempotence law (I) clearly contradicts the race condition we assume. Instead, a revised law (I') is needed that reflects the minimum property (C) of exponential distributions. It is listed in Table 8 together with the main equational laws characterising lumpability [20].

4 Interactive Markov Chains

This section joins the models of the preceding two sections, continuous time Markov chains and labelled transition system in an orthogonal fashion. It does so by means of two types of prefixes. The action-prefixed expression $\mathsf{a}.P$ may interact on action a and afterwards behave as expression E. The delay-prefixed expression $(\lambda).F$ has to delay for an exponentially distributed time according to rate λ before turning into expression F. We first introduce the language and discuss the equational properties we expect to hold for this language. Then we match the equational theory with a corresponding operational definition (in SOS style) and appropriate notions of semantic equivalences, based on strong bisimulation and weak congruence.

4.1 Algebra of Interactive Markov Chains

Definition 16. *Let* $\mathsf{a} \in \mathcal{A}_\tau$, $\lambda \in \mathbb{R}^+$, *and* $X \in \mathcal{V}$. *We define the language* IMC^c *as the set of* closed *expressions given by the following grammar.*

$$\mathcal{E} ::= \quad 0 \quad | \quad \mathsf{a}.\mathcal{E} \quad | \quad (\lambda).\mathcal{E} \quad | \quad \mathcal{E} + \mathcal{E} \quad | \quad X \quad | \quad [X := \mathcal{E}]_i$$

Example 17. A simple example of an expression of IMC^c is

$$\mathsf{a}.(\lambda).(\mu).\mathsf{a}.(\mu).0$$

Our intuition is as follows: The expression initially is ready to interact on action a. Afterwards, it delays the next possible interaction on a by a sequence of exponential delays, the first given by rate λ, the second given by rate μ. After a second interaction on a it delays for another exponentially distributed duration, before turning into the terminated expression.

This small example of what we intend to do with IMC^c does not cover all possibilities. In general, we can combine delays and actions, such as in

$$\mathsf{a}.P + (\lambda).Q$$

As a consequence, we have to develop an unambiguous view of the interplay of actions and delays. To do so, we discuss the meaning of IMC^c from an algebraic perspective, by stating which expressions of IMC^c can be equated with respect to an intuitive notion of equality. It is important to observe that we have some freedom with respect to what we consider to be intuitive, but there are also constraints.

Strong bisimulation. First of all we intend to inherit the algebras PA^c and IMC^c i.e., the laws we established earlier should remain valid. As a consequence, our equational theory for strong bisimulation for IMC is based on the union of the axioms listed in Table 4 and in Table 8. We could decide that this union is precisely covering all the cases we want to equate with a strong bisimulation. This would mean that the meaning of $\mathsf{a}.P + (\lambda).Q$ has to be obtained somehow by interpreting the class of all algebraically equivalent expressions, i.e. all expressions into which it can be rewritten using the axioms. Here, we decide not to follow this purely algebraic approach, but instead to add one further axiom that corresponds to an operational intuition. This is the notion of *maximal progress*: we assume that intuitively actions can happen as soon as possible. This means that a behaviour such as $\mathsf{a}.P + (\lambda).Q$ will not be delayed at all if the action a is instantaneously enabled. In this case, $\mathsf{a}.P + (\lambda).Q$ will behave just like $\mathsf{a}.P$ (since the probability of the delay to finish is $1 - e^{-\lambda 0} = 0$). But how do we know that action a is indeed enabled? We do *not* know this in general, because the action may depend on some interaction with the environment which is delayed. But in the specific case where action a is the distinguished internal action τ the environment cannot influence its occurrence, and therefore it can occur instantaneously. Hence we can add an axiom

$$(P)\ \ (\lambda).E + \tau.F \ = \ \tau.F$$

to our equational theory, reflecting our *maximal progress assumption*. This assumption is often made in real-time process algebra [45,59,52,12].[1]

The resulting set of core axioms for strong bisimilarity on IMC is listed in Table 9. All of them have appeared in the subalgebras of PA^c and MC^c, except for (P) and for the law (I''). The latter restricts the standard idempotence law (I) (i.e., $E + E = E$) to action prefixed expressions, such that it is compatible with (I'). Recall that (I) contradicts the race condition assumed in the MC context, and hence needs a refinement.

[1] In the context of generalised stochastic Petri nets a similar assumption is present: by definition, immediate transition are assumed to have a higher priority level than Markov timed transitions [2].

$$
\begin{array}{ll}
(C) & E + F = F + E \\
(A) & (E + F) + G = E + (F + G) \\
(I') & (\lambda).E + (\mu).E = (\lambda + \mu).E \\
(I'') & \mathsf{a}.E + \mathsf{a}.E = \mathsf{a}.E \\
(N) & E + 0 = E \\
 & \\
(P) & (\lambda).E + \tau.F = \tau.F
\end{array}
$$

Table 9. Axioms for IMC strong bisimilarity.

$$
\begin{array}{ll}
(C) & E + F = F + E \\
(A) & (E + F) + G = E + (F + G) \\
(I) & \mathsf{a}.E + \mathsf{a}.E = \mathsf{a}.E \\
(I') & (\lambda).E + (\mu).E = (\lambda + \mu).E \\
(N) & E + 0 = E \\
 & \\
(P) & (\lambda).E + \tau.F = \tau.F \\
 & \\
(\tau 1) & \mathsf{a}.\tau.E = \mathsf{a}.E \\
(\tau 1') & (\lambda).\tau.E = (\lambda).E \\
(\tau 2) & E + \tau.E = \tau.E \\
(\tau 3) & \mathsf{a}.(E + \tau.F) + \mathsf{a}.F = \mathsf{a}.(E + \tau.F)
\end{array}
$$

Table 10. Axioms for IMC weak congruence.

Weak congruence. Even though the axiom (P) of strong bisimilarity involves a specific treatment of internal actions, the axiom system in Table 9 does not provide means to abstract from sequences of internal actions. Recall that Table 5 presents axioms $(\tau 1)$–$(\tau 3)$ that reflect the power of weak congruence to eliminate sequences of internal actions. A similar treatment of internal actions is desirable for IMC, and we therefore postulate some axioms for a weak congruence on IMC. The axioms are listed in Table 10. They extend the ones we have postulated for strong bisimulation on IMC (because we want to preserve the equations of strong bisimulation) with additional axioms that take care of internal actions. The axioms $(\tau 1)$–$(\tau 3)$ are known from the PA^c context already, and axiom $(\tau 1')$ is an obvious adaption of $(\tau 1)$ to the delay prefixed case: if we can do an internal move after some delay, we can also skip the internal move, but not the delay.

Studying the equational theory in Table 10 raises the question why axioms $(\tau 2)$ and $(\tau 3)$ do not require a similar adaptation to the delay-prefix case as $(\tau 1)$ does. For $(\tau 2)$, the answer is easy, because it is not specific for the action-prefix case, it also covers the cases where E involves delay prefixes. But for $(\tau 3)$ the answer is more involved. The adapted candidate law

$$(\tau 3') \quad (\lambda).(E + \tau.F) + (\lambda).F = (\lambda).(E + \tau.F)$$

$$\frac{}{\mathsf{a}.E \xrightarrow{\mathsf{a}} E} \qquad \frac{E \xrightarrow{\mathsf{a}} E'}{E + F \xrightarrow{\mathsf{a}} E'} \qquad \frac{F \xrightarrow{\mathsf{a}} F'}{E + F \xrightarrow{\mathsf{a}} F'} \qquad \frac{E_i\{[X := E]_i / X_i\} \xrightarrow{\mathsf{a}} E'}{[X := E]_i \xrightarrow{\mathsf{a}} E'}$$

$$\frac{}{(\lambda).E \xrightarrow{\lambda} E} \qquad \frac{E \xrightarrow{\lambda} E' \quad F \not\xrightarrow{\tau}}{E + F \xrightarrow{\lambda} E'} \qquad \frac{E \xrightarrow{\lambda} E' \quad F \not\xrightarrow{\tau}}{F + E \xrightarrow{\lambda} E'} \qquad \frac{E_i\{[X := E]_i / X_i\} \xrightarrow{\lambda} E'}{[X := E]_i \xrightarrow{\lambda} E'}$$

Table 11. Operational semantic rules for IMC^c.

is *not* sound for IMC weak congruence, since on the left hand side, the time needed before being able to behave as F is governed by an exponential distribution with rate 2λ, while the process on the right is slower, since it evolves into F after a delay with a rate of only λ. The fact that law $(\tau 3')$ is invalid shades some interesting light on our definition, and suggests a resemblance to the behavioural equivalence known as branching bisimulation [54]. The interested reader is referred to [19] for a detailed discussion concerning this similiarity.

We conclude our an axiomatic view on IMC by pointing out that this perspective still has to be matched by an operational definition of the semantics that matches these axioms – up to appropriate notions of equalities, which also need to be defined.

4.2 Semantics

Interactive Markov Chains involve two types of prefixes. On the semantic level this leads to a model with a twofold transition relation $\longrightarrow$ and $-\!\!\rightarrow$. The former represents action transitions, the latter represents Markov transitions. This should not be surprising, since we strive for an orthogonal extension of PA^c and MC^c.

To give a meaning to elements of IMC^c we define an operational semantics on the basis of the SOS-rules introduced for PA^c (Table 2) and MC^c (Table 7), except for one change.

Definition 17. *The* action transition *relation* $\longrightarrow \subset \mathsf{IMC}^c \times \mathcal{A}_\tau \times \mathsf{IMC}^c$ *is the least relation and the* Markov transition *relation* $-\!\!\rightarrow \subset \mathsf{IMC}^c \times \mathbb{R}^+ \times \mathsf{IMC}^c$ *is the least multi relation given by the rules in Table 11, where* $E \not\xrightarrow{\tau}$ *denotes that there is no* $E' \in \mathsf{IMC}^c$ *such that* $E \xrightarrow{\tau} E'$.

Compared to the rules in Table 2 and, Table 7, two rules are now equipped with negative premises of the form $E \not\xrightarrow{\tau}$ meaning that no internal transition can be performed by E. Only in this case, a Markov transition can happen in the context of choice. This negative premise[2] is used to encode *maximal progress*:

[2] The use of a negative premise is not mandatory to make the above axiom system sound. Alternatively, one can take the operational rules as the plain union of the ones

An expression is only allowed to delay, if it has nothing internal to do instantaneously. Note that the additional negative premise is only influencing the behaviour of expressions that involve both delay prefixes and action prefixes. So, if restricted to the sublanguages PA^c and MC^c, the operational semantics reduces to the ones introduced in Table 2 and Table 7.

Example 18. The semantics of the process E_{65} defined by $(2\lambda).(\tau.0 + \mathsf{a}.0)$ is depicted in the upper left of Figure 10. As another example, the semantics of the process E_{66} defined by $[X_1 := \tau.X_2\ ,\ X_2 := \tau.X_1 + (2\lambda).(\tau.0 + \mathsf{a}, 0)]_1$ is depicted in the upper right of the figure.

4.3 Equivalence

We now investigate how equivalences on IMC^c can be defined. Action transitions and Markov transitions coexist in IMC. Meaningful equivalences for IMC should thus reflect their coexistence. Strong and weak bisimilarities will therefore be based on the respective notions for PA^c and MC^c. Additionally, the interrelation of action and Markov transitions has to be captured as well, according to the axioms we have postulated in Section 4.1.

Strong bisimulation. We introduce strong bisimilarity based on Definition 7 and 13.

Definition 18. *An equivalence relation $\mathcal{E}$ on IMC^c is a strong bisimulation iff $P\ \mathcal{E}\ Q$ implies for all $\mathsf{a} \in \mathcal{A}_\tau$*

1. *$P \xrightarrow{\mathsf{a}} P'$ implies $Q \xrightarrow{\mathsf{a}} Q'$ for some Q' with $P'\ \mathcal{E}\ Q'$,*
2. *$\gamma_{\mathrm{M}}(P, C) \leq \gamma_{\mathrm{M}}(Q, C)$ for all equivalence classes C of $\mathcal{E}$.*

Two processes P and Q are strongly bisimilar (written $P \sim Q$) if they are contained in some strong bisimulation.

This definition amalgamates strong bisimilarity for PA^c and for MC^c. In order to compare the stochastic timing behaviour, the cumulative rate function γ_{M} is used, as motivated in Section 3.2. Formally speaking bisimilarity conservatively extends [13] the respective notions on basic process algebra processes and Markov chains. This answers indeed why we have reused the symbol $\sim$, that has been used in Section 2.4 already to denote strong bisimilarity on PA^c.

We obtain the following desirable results:

for PA^c and MC^c, as done in [20]. In this case a more involved definition of strong and weak bisimulation is needed that must now incorporate maximal progress. The way we proceed here is sketched in [19, Sec. 6.1] and elaborated in [7]. The solution is didactically more appealing, but has the drawback that divergence may imply the awkward phenomeon of a time deadlock: If a state is on a cycle of internal transitions, this implies that no Markov transition (indicating time progress) can be derived, even though the system may return to this state via the internal steps ad infinitum.

Proposition 3. *Strong bisimilarity is*

- *an equivalence relation on* IMC^c.
- *a strong bisimulation on* IMC^c.
- *the* largest *strong bisimulation on* IMC^c.

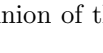
In addition, strong bisimulation turns out to be the desired notion of equivalence relative to the equational theory we postulated.

Theorem 4. *Strong bisimilarity is a congruence relation with respect to the operators of* PA^c*, and it satisfies the axioms in Table 9.*

By adding additional laws to handle recursion the equational theory induced by Table 9 can be shown to completey characterise strong bisimulation, see [19].

Weak congruence. Strong bisimilarity does not abstract from sequences of internal transitions like *weak* bisimilarity does (cf. Section 2.4). We will therefore try to find a corresponding definition of a weak relation for IMC, that is, a weak relation that complies to the axioms we have postulated in Table 10.

A few questions have to be addressed in order to define weak bisimulation on IMC properly. While the treatment of action transitions can follow the lines of Section 2.4, the treatment of Markov transitions in a weak bisimulation has to be clarified. As remarked in Section 3.2 it is impossible to replace a sequence of Markov transitions by a single Markov transition without affecting the probability distribution of the total delay. So, we are forced to demand that Markov transitions have to be bisimulated in the *strong* sense, using function γ_{M}, even for a weak bisimulation. However, we allow them to be preceded and followed by arbitrary sequences of internal action transitions. These sequences are, according to Definition 9 given by $\stackrel{\varepsilon}{\Longrightarrow}$, the reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$. To incorporate these sequences into the definition of weak bisimulation is a bit involved technically. For strong bisimilarity, γ_{M} has been used to cumulate rates of Markov transitions that directly lead from a state P into a specific equivalence class C. We broaden this treatment in order to keep track of the impact of internal transitions that *follow* a Markov transition: We cumulate all rates of Markov transitions leading to states that can internally evolve into an element of class C. For this purpose, we define the *internal backward closure* C^τ as the set of processes that may internally evolve into an element of a set C, i.e. $C^\tau = \{P' \mid \exists P \in C : P' \stackrel{\varepsilon}{\Longrightarrow} P\}$.

Example 19. Concerning the IMC E_{67} in Figure 10, the internal backward closure  of the set  is the union of the sets and  .  is and  is  .

The treatment of internal sequences *preceding* a Markov transitions can follow the style of Definition 10. Thus, whenever there is a sequence of internal transitions to some state P, the cumulative rate $\gamma(P, C^\tau)$ should be taken into account for comparison purposes. This requirement will be made more precise in the following definition.

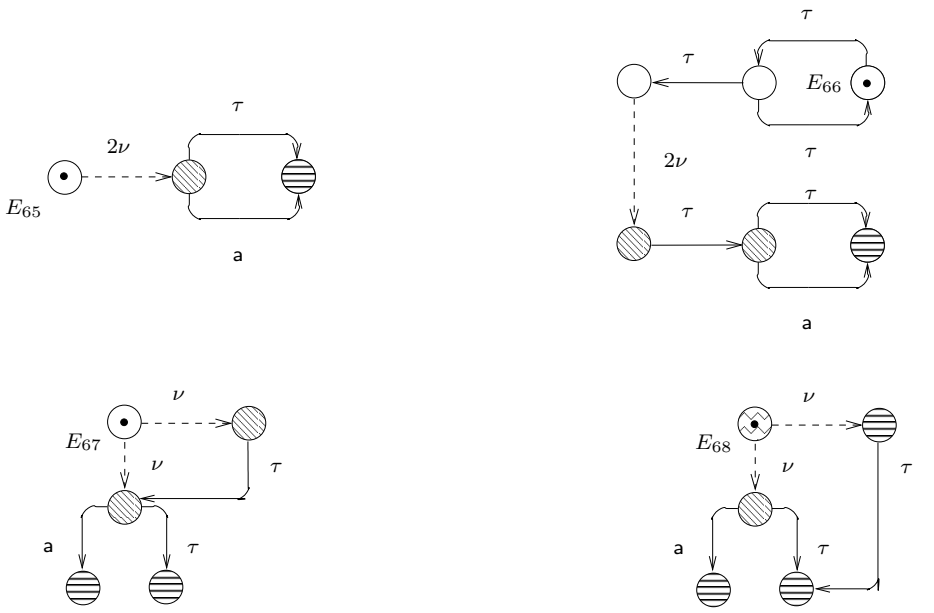


Fig. 10. Some characteristic examples for weak bisimilarity.

Definition 19. *An equivalence relation $\mathcal{E}$ on IMC^c is a weak bisimulation iff $P\ \mathcal{E}\ Q$ implies for all $\mathsf{a} \in \mathcal{A}_\varepsilon$*

1. *$P \stackrel{\mathsf{a}}{\Longrightarrow} P'$ implies $Q \stackrel{\mathsf{a}}{\Longrightarrow} Q'$ for some Q' with $P'\ \mathcal{E}\ Q'$,*
2. *$P \stackrel{\varepsilon}{\Longrightarrow} P'$ imply $Q \stackrel{\varepsilon}{\Longrightarrow} Q'$ for some Q' with $\gamma_{\mathrm{M}}(P', C^\tau) \leq \gamma_{\mathrm{M}}(Q', C^\tau)$ for all equivalence classes C of $\mathcal{E}$.*

Two processes P and Q are weakly bisimilar (written $P \approx Q$) if they are contained in some weak bisimulation.

We illustrate the distinguishing power of $\approx$ by means of some examples.

Example 20. E_{65} and E_{66} depicted in Figure 10 are equivalent even though the precise argument is somewhat involved. We have for E_{65} that $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS}) = 2\nu$, as well as $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS}) = 2\nu$. E_{66} has value 0 for both classes, but this does not violate the conditions imposed by clause (2) of Definition 19. Instead, we have to find a state reachable via τ inside class $\circledS$ satisfying $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS}) \geq 2\nu \leq \gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS})$. Indeed we get the values 2ν precisely for the leftmost state $\bigcirc$ reachable from E_{66}, and hence clause (2) of Definition 19 is satisfied. On the other hand, also the rate of E_{66} has to be investigated. We see that for E_{66}, $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS}) = 0 = \gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circledS})$, which is at most the values of E_{65}. Note that in this reasoning, it is important that we do not demand equality of cumulated rates in clause (2), but instead demand to find a matching

state with *at least* ($\leq$) the cumulated rates of the state we have to check. Hence $E_{65} \approx E_{66}$.

The process E_{67} is equivalent to the former two, because $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circlearrowleft}) = 2\nu$ and $\gamma_{\mathrm{M}}(\bigcirc, \overset{\tau}{\circlearrowleft}) = 2\nu$. In contrast, $\gamma_{\mathrm{M}}(\circledcirc, \overset{\tau}{\circlearrowleft}) = \nu$ whence we have that E_{68} is not weakly bisimilar to the former three processes.

We get the following desirable properties of $\approx$.

Proposition 4. *Weak bisimilarity is*

- *an equivalence relation on* IMC^c.
- *a weak bisimulation on* IMC^c.
- *the* largest *weak bisimulation on* IMC^c.
- *a congruence with respect to all operators of* IMC^c except the choice operator '$+$'.

The fact that weak bisimilarity is not substitutive with respect to choice is inherited from non-stochastic weak bisimilarity, cf. Section 2.4. In order to rectify this situation we identify a specific congruence contained in $\approx$, along the lines of Definition 11.

Definition 20. *P and Q are weakly congruent, written $P \simeq Q$, iff for all $\mathsf{a} \in \mathcal{A}_\tau$ and all $C \in \mathsf{IMC}^c / \approx$:*

1. $P \xrightarrow{\mathsf{a}} P'$ *implies* $Q \overset{\varepsilon}{\Longrightarrow} \xrightarrow{\mathsf{a}} \overset{\varepsilon}{\Longrightarrow} Q$ *for some* Q' *with* $P' \approx Q'$,
2. $Q \xrightarrow{\mathsf{a}} Q'$ *implies* $P \overset{\varepsilon}{\Longrightarrow} \xrightarrow{\mathsf{a}} \overset{\varepsilon}{\Longrightarrow} P'$ *for some* P' *with* $P' \approx Q'$,
3. $\gamma_{\mathrm{M}}(P, C) = \gamma_{\mathrm{M}}(Q, C)$,

Weak congruence strengthens the requirement of weak bisimilarity of initial internal transitions in precisely the same way as in Definition 11. It requires that an initial internal transition has to be matched by at least one internal transition. This small change it is again sufficient to fix the congruence problem with respect to choice: weak congruence is a proper substitutive relation with respect to all language operators.

Theorem 5. *Weak congruence is a congruence with respect to all operators of* IMC^c*, and it satisfies the axioms in Table 10.*

By adding further laws the equational theory induced by Table 9 can be shown to be sound and complete with respect to weak congruence on IMC^c [19,7] . Furthermore, weak congruence is the *coarsest* congruence contained in weak bisimilarity, as a consequence of the following lemma (cf. Lemma 3).

Lemma 5. *$E_1 \simeq E_2$ iff, for each $E_3 \in \mathsf{IMC}^c$, $E_1 + E_3 \approx E_2 + E_3$ and $E_3 + E_1 \approx E_3 + E_2$.*

As a result, we have obtained two substitutive equivalence notions on IMC^c: strong bisimilarity and weak congruence, a distinguished subset of weak bisimilarity. The interrelation between these equivalences is expressed in the following lemma (cf. Lemma 4).

$$\frac{P \xrightarrow{\lambda} P' \qquad Q \not\xrightarrow{\tau}}{P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q \xrightarrow{\lambda} P' \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q} \qquad \frac{Q \xrightarrow{\lambda} Q' \qquad P \not\xrightarrow{\tau}}{P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q \xrightarrow{\lambda} P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q'}$$

$$\frac{P \xrightarrow{\lambda} P' \qquad \texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P \not\xrightarrow{\tau}}{\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P \xrightarrow{\lambda} \texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P'}$$

Table 12. Structural operational rules for parallel composition and abstraction.

Lemma 6. $\sim \subset \simeq \subset \approx$.

Summarizing, we have managed to integrate basic process algebra – where strong bisimulation and weak congruence are reference notions – and Markov chain algebra – where lumpability is central – in a single algebra. An obvious next step now is to extend this algebraic core with the other operators from process algebra to enable the concurrent composition of IMC expressions and abstraction from observable actions.

4.4 Concurrency and Abstraction

The two operators for abstraction and parallel composition, cf. Section 4, can be added to IMC^c without disturbing any of the theory. We extend IMC^c with these operators by stipulating that if P and Q are in IMC^c then $P \parallel \mathsf{a}_1 \ldots \mathsf{a}_n \parallel Q$ and $\texttt{hide } \mathsf{a}_1 \ldots \mathsf{a}_n \texttt{ in } P$ are is in IMC^c as well. The semantics of these operators are given by the operational rules in Table 2, Table 3, and Table 12.

According to this definition, action transitions are treated precisely as in the PA^c setting. Markov transitions $-\!\!\rightarrow$ are only possible if maximal progress is assured, which is incorporated via negative premises. Negative premises in such rule schemata have to be treated carefully in general, since they may affect the well-definedness of the induced transition relation [17]. In this case, however, it is not difficult to show that the rule schemata are well-defined.

In the case of parallel composition, it should be noted that the Markovian delay transitions are interleaved as if they were standard action transitions, in particular without adjusting rates. This is a consequence of the memoryless property (cf. property **B** on page 204), and one of the principal reasons why exponential distributions fit so well to process algebra. In the following Section 5 we will elaborate on the appropriateness of this combination.

One of the consequences of this independent delaying is that the expansion law (X) (cf. Table 6) can be extended in a rather straightforward way to Interactive Markov Chains. Table 13 lists the resulting law, together with an additional law for abstraction, that (together with the ones in Table 6) allow one to rewrite parallel composition, as well as abstraction into the basic operators of IMC^c.

Example 21. In order to exercise the modelling of concurrent behaviour with IMC we consider two processes, E_{71} and E_{72}, depicted in Figure 11. They are

$$(X') \quad P \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q = \sum(\lambda_i).(P_i \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q) + \sum_{\mathtt{a}_j \notin \{\mathtt{a}_1 \ldots \mathtt{a}_n\}} \mathtt{a}_j.(P_j \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q) +$$

$$\overbrace{\underbrace{\sum(\lambda_i).P_i + \sum \mathtt{a}_j.P_j}_{P} \quad \underbrace{\sum(\mu_k).Q_k + \sum \mathtt{b}_l.Q_l}_{Q}}^{} \qquad \sum(\mu_k).(P \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q_k) + \sum_{\mathtt{b}_l \notin \{\mathtt{a}_1 \ldots \mathtt{a}_n\}} \mathtt{b}_l.(P \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q_l) +$$

$$\sum_{\mathtt{a}_j = \mathtt{b}_l \in \{\mathtt{a}_1 \ldots \mathtt{a}_n\}} \mathtt{a}_j.(P_j \,|[\, \mathtt{a}_1 \ldots \mathtt{a}_n \,]|\, Q_l)$$

$$(H2') \quad \mathtt{hide}\ \mathtt{a}_1 \ldots \mathtt{a}_n\ \mathtt{in}\ \ (\lambda).P = (\lambda).\mathtt{hide}\ \mathtt{a}_1 \ldots \mathtt{a}_n\ \mathtt{in}\ \ P$$

Table 13. Axioms for rewriting parallel composition and abstraction on IMC^c.

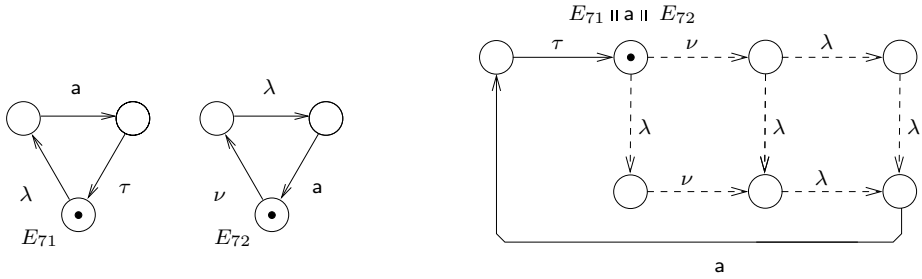


Fig. 11. Parallel composition of IMC.

defined by $[X_1 := (\lambda).\mathtt{a}.\tau.X_1]_1$, respectively $[X_1 := (\nu).(\lambda).\mathtt{a}.X_1\,)]_1$. Their parallel composition $E_{71} \,|[\, \mathtt{a} \,]|\, E_{72}$ is depicted on the right of the figure. Note that maximal progress enforces the τ-transition of E_{71} to take precedence over a delay (with rate ν) of E_{72} prior to reeintering the initial state.

Figure 12 illustrates the semantics of $(\mathtt{hide\ a\ in}\ \ E_{71} \,|[\, \mathtt{a} \,]|\, E_{72}\,)$, a process where all actions are internalised. In this figure, states shaded with equal patterns are weakly congruent. The shading indicates that this process is weakly congruent to a process E_{73} defined by $[X_1 := (\nu).(2\lambda).(\lambda).X_1 + (\lambda).(\lambda).(\nu).X_1\ )]_1$. This is a small Markov chain depicted on the right of Figure 12. The fact that

$$(\mathtt{hide\ a\ in}\ \ E_{71} \,|[\, \mathtt{a} \,]|\, E_{72}\,) \ \simeq\ E_{73},$$

means that the concurrent execution of the Interactive Markov Chains E_{71} and E_{72} can be concisely represented by the Markov chain E_{73}. In other words, we have generated a small Markov chain from the composition of two IMCs.

This is a very simple example showing how Markov chains can be compositionally specified with IMC. Subsequently, the model can be evaluated with standard analysis techniques for Markov chains, cf. [18]. A much larger case study is developed in [22], where a Markov chain model of 720 states is derived from

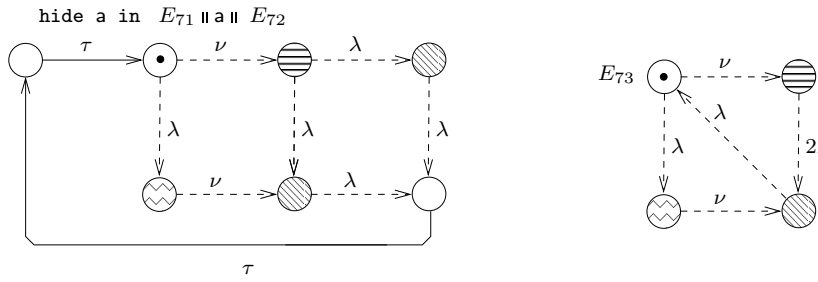


Fig. 12. Compositional specification of a Markov chain.

an IMC specification of the plain old telephone system involving more than 10^7 states. To circumvent the state space explosion problem, the case study makes heavy use of substitutivity (i.e., congruence) properties and algorithms for simplifying (i.e., lumping) the state space (of components) according to weak congruence. We refer to [19] for further reading on IMC.

5 Related Work: A Comparison of Algebraic Principles

As was already mentioned in the introduction a fair number of stochastic process algebras have been developed during the last decade or so, IMC being one of them. In this section we want to make a comparison between them. We will not do so in terms of the complete formalisms, but will organise our discussion around the potential resolution strategies for a number of issues that arise inevitably when trying to combine well-known process algebraic principles with the features of continuous time Markov chains. This will give, we hope, a more generic insight into the (im)possibilities of the various approaches.

We will forego the challenge of defining an integrating semantical model for the various formalisms for a deeper mathematical comparison of the various constructs. Instead, we will try and make our comparison in terms of the various algebraic principles, i.e. in terms of the sort of equational laws that are involved. We think that this is the best level of abstraction to discuss the different options and their consequences.

The three central questions that must be addressed when developing Markovian process algebras are:

1. the meaning of choice
2. the meaning of concurrent composition
3. the meaning of synchronisation

We address these questions in the following sections. Our notational vehicle will be slightly different from that of IMC, because the other stochastic process algebras do not have the separation between actions and delays. We will therefore have only one action-prefix construct, viz.

$$(\mathsf{a}, \lambda).B$$

meaning that action a may take place after an exponentially distributed delay with rate λ, resulting in the behaviour specified by B. As we will also want to discuss some aspects stochastic process algebra in a non-Markovian setting, we also introduce the notation

$$(\mathsf{a}, F).B$$

where F is denotes a (general) distribution of the delay associated with a. We allow F to be denoted by an expression over one or more stochastic variables whose distributions are (implicitly) given, e.g. $(\mathsf{a}, X).B$ implies that a is delayed with the distribution of X, and $(\mathsf{a}, f(X, Y)).B$ means that a is delayed with the distribution defined by $f(X, Y)$ for given distributions of X and Y.

5.1 The Meaning of Choice

The choice or summation operator affects the branching structure of the transition system that is described: its SOS style semantics yields multiple outgoing transitions from a state in the underlying transition system. In CTMCs such outgoing transitions are interpreted as creating a *race condition*, i.e. they are seen as processes that are competing for the fastest service according to their distributions. As we have seen, for exponential distributions the time until the first transition fires is again exponentially distributed, with a rate that is the sum of the rates of the individual transitions.

It stands to reason that in a Markovian process algebra we should somehow be able to add up the rates of all transitions with identical action labels. Indeed, such transitions will all be in a race condition when the environment has enabled the corresponding action. Using the probabilistic choice operator $\oplus_p$ introduced in Section 3.3 we can write this down as an algebraic law, the *race condition principle* (RCP):

$$(\mathsf{a}, \lambda).B + (\mathsf{a}, \mu).C \;=\; (\mathsf{a}, \lambda + \mu).(B \oplus_{\frac{\mu}{\lambda+\mu}} C) \tag{7}$$

Note that the right-hand side of the equation can be interpreted as a process for which an a-transition with the combined rates leads to a *superposition state* that, when it is reached, reduces instantaneously to one of its constituents states. This occurs with a probability proportional to the relative weight of the corresponding rate.

This principle can also be formalised for the non-Markovian case, where a race condition between arbitrary continuous distributions takes place (e.g. as in semi-Markov chains):

$$(\mathsf{a}, X).B + (\mathsf{a}, Y).C \;=\; (\mathsf{a}, \min(X, Y)).(B \oplus_{P\{Y<X\}} C) \tag{8}$$

The above laws make it clear that in a stochastic setting the choice operator has to take the *capacity* of its arguments into account. This is even clearer in the case of IMC, where the delay operator can be interpreted as a scalar multiplication w.r.t. choice:

$$(\lambda).B + (\mu).B \quad = \quad (\lambda + \mu).B \tag{9}$$

This is in stark contrast to the usual interpretation of the choice operator in process algebra, which could be referred to as *structural*, in the sense that only of the arguments is chosen, and therefore there is no interference with properties of the conflicting transitions. This leads to the *idempotency* law for choice, which could be seen as a kind of 'poor man's choice': choosing between identical arguments is as good as no choice at all:

$$B + B \quad = \quad B \tag{10}$$

The difference between the laws (7) and (10) is very important, as the choice operator plays a crucial role in the formulation of other laws in process algebra, the so-called *expansion laws* in particular. Below we will discuss the implications that the capacitive interpretation of choice in the context of parallel composition.

It is possible to interpret (10) as a limit case of (7) by interpreting the former as the behaviour for *immediate* transitions, cf. [27]. They can be thought of as having an infinite rate ∞, with the property $\infty + \infty = \infty$. This approach makes (10) compatible with (7) for actions for which this is a reasonable assumption (e.g. τ-actions).

The process algebra EMPA [4] also wants to apply (10) to so-called *passive* actions, actions that have no associated (finite) rates, but obtain them by synchronising with non-passive actions. This, however, leads to complications, as we will see below.

5.2 Concurrent Composition

The next operator that we consider is the parallel or concurrent composition of processes. As we will consider the issue of synchronisation of actions separately, here we concentrate on 'pure' *interleaving*, i.e. parallel composition without synchronisation of actions between components.

To guide our discussion we consider the following, simple expansion law of standard process algebra:

$$\mathsf{b}.B \,|||\, \mathsf{c}.C \quad = \quad \mathsf{b}.(B \,|||\, \mathsf{c}.C) + \mathsf{c}.(\mathsf{b}.B \,|||\, C) \tag{11}$$

The first thing that we can observe is that for general distributions the interleaving law does not hold, i.e.

$$(\mathsf{b}, X).B \,|||\, (\mathsf{c}, Y).C \quad \neq \quad (b, X).(B \,|||\, (\mathsf{c}, Y).C) + (\mathsf{c}, Y).((\mathsf{b}, X).B \,|||\, C) \tag{12}$$

because the occurrence of b after c has taken time to occur generally has another distribution than b occurring initially.

One way to deal with this complication is to restrict to exponential distributions, i.e. the Markovian case. Because of the memoryless property of these distributions, they are perfectly compatible with the interleaving law as distributions are not affected by the conditional information that one action takes place only after the occurrence of another. So we have

$$(\mathsf{b},\lambda).B \;|||\; (\mathsf{c},\mu).C \;=\; (\mathsf{b},\lambda).(B \;|||\; (\mathsf{c},\mu).C) + (\mathsf{c},\mu).((\mathsf{b},\lambda).B \;|||\; C) \qquad (13)$$

showing again the perfect match of interleaving process algebra and continuous time Markov chains, as was already evident from the elegant theory of IMC.

It is worthwhile to consider also ways out of the complications of (12), as for many applications the assumption of memorylessness is too strong. One straightforward way is to complicate the interleaving law by adding the conditional information that was missing. In this way we obtain:

$$\begin{aligned}(\mathsf{b},X).B \;|||\; (\mathsf{c},Y).C \;=\;& (\mathsf{b},X).(B \;|||\; (\mathsf{c},\langle Y-X|X<Y\rangle).C) + \\ & (\mathsf{c},Y).((\mathsf{b},\langle X-Y|X>Y\rangle).B \;|||\; C)\end{aligned} \qquad (14)$$

where $\langle Y-X|X<Y\rangle$ denotes the distribution of $Y-X$ under the condition that $X<Y$.

The disadvantage of this approach is that the formula complicates very rapidly if more than two actions interleave with nested conditional distributions. This is unattractive, not only algebraically, but also in the sense that the calculation of such complicated conditional distributions is computationally expensive, e.g. when doing simulations. Presumably because of this reasons this approach has not been pursued in (non-Markovian) process algebra.

A second way out is provided by following the separation of concerns between delay transitions and actions, as was also done in IMC. If this idea is combined with the use of stochastic *clocks* to guard the occurrence of transitions, we can formulate interleaving again in terms of an elegant algebraic law, even if one considers the non-Markovian case. Let $\{X_1,\ldots,X_n\}B$ mean that in the initial state of B the clocks $X_1,\ldots,X_n$ are set with random samples according to their associated distributions over $\mathbb{R}^+$, after which they start counting down until they expire (reach 0). Let $(X \to \mathsf{b})$ mean that b is delayed until X has expired. With these additional stochastic clock operators we now get a new interleaving law, viz.

$$\begin{aligned}&\{X,Y\}(X \to \mathsf{b}).B \;|||\; (Y \to \mathsf{c}).C \;= \\ &\qquad \{X,Y\}((X \to \mathsf{b}).(B \;|||\; (Y \to \mathsf{c}).C) + (Y \to \mathsf{c}).((X \to \mathsf{b}).B \;|||\; C))\end{aligned} \qquad (15)$$

where we see that the guarded actions $(X \to \mathsf{b})$ and $(Y \to \mathsf{c})$ are interleaved, but the clock setting $\{X,Y\}$ is not. This approach to non-Markovian process algebra is elaborated in [33].

A final approach that we wish to mention in connection with the interleaving law in a stochastic context is to give up the law altogether. This idea belongs to

the semantic school of the 'true concurrency', which insists that parallel composition is fundamentally different from interleaving and that the two should not be equated. The causal dependencies on the left-hand and right-hand sides of the interleaving law are different (b and c are independent versus b causes/precedes c or vice versa) [36]. It is possible to extend so-called *partial-order semantics* for process algebra with stochastic information to obtain suitable non-interleaving semantical models for stochastic process algebras. We refer to [8,32] for further reading.

5.3 Synchronisation

Probably the most intriguing question in the design of stochastic process algebra is related to the synchronisation of stochastic actions of two concurrent system components. If both actions are subject to stochastic delays, what should be the stochastic delay of their synchronised occurrence?

Looking at this question in its simplest process algebraic form, we consider standard process algebraic law

$$\mathsf{a}.B \,\|_{\mathsf{a}}\, \mathsf{a}.C = \mathsf{a}.(B \,\|_{\mathsf{a}}\, C) \tag{16}$$

and wonder what are reasonable functions '$*$' that would make the corresponding stochastic equation hold true:

$$(\mathsf{a}, X).B \,\|_{\mathsf{a}}\, (\mathsf{a}, Y).C = (\mathsf{a}, X * Y).(B \,\|_{\mathsf{a}}\, C) \tag{17}$$

From a stochastic point of view, one may immediately think of various operationalisations of '$*$', such as the maximum of the distributions, or their convolution, minimum, average etc. Interestingly enough, however, the algebraic properties of the involved operators already impose certain restrictions on '$*$' by themselves.

Let us consider the term $((\mathsf{a}, X).B + (\mathsf{a}, Y).C) \,\|_{\mathsf{a}}\, (\mathsf{a}, Z).D$. The interplay of choice and synchronisation allows us to derive:

$$\begin{aligned}
((\mathsf{a}, X).B + (\mathsf{a}, Y).C) \,\|_{\mathsf{a}}\, (\mathsf{a}, Z).D \;&=\; && \text{(by (8))}\\
(\mathsf{a}, \min(X, Y)).(B \,\oplus_{P\{X>Y\}}\, C) \,\|_{\mathsf{a}}\, (\mathsf{a}, Z).D \;&=\; && \text{(by (17))}\\
(\mathsf{a}, \min(X, Y) * Z).((B \,\oplus_{P\{X>Y\}}\, C) \,\|_{\mathsf{a}}\, D) \;&=\; && \text{(distributing } \oplus\text{)}\\
(\mathsf{a}, \min(X, Y) * Z).((B \,\|_{\mathsf{a}}\, D) \,\oplus_{P\{X>Y\}}\, (C \,\|_{\mathsf{a}}\, D)) && && (18)
\end{aligned}$$

On the other hand, by assuming that we have a (classical) expansion law for synchronisation of the form

$$\begin{aligned}
&((\mathsf{a}, X).B + (\mathsf{a}, Y).C) \,\|_{\mathsf{a}}\, (\mathsf{a}, Z).D = \\
&\qquad (\mathsf{a}, X * Z).(B \,\|_{\mathsf{a}}\, D) + (\mathsf{a}, Y * Z).(C \,\|_{\mathsf{a}}\, D)
\end{aligned} \tag{19}$$

we obtain by applying the RCE (8) to the right-hand side:

$$(\mathsf{a}, \min(X * Z, Y * Z)).((B \,\|_{\mathsf{a}}\, D) \,\oplus_{P\{X*Z>Y*Z\}}\, (C \,\|_{\mathsf{a}}\, D)) \tag{20}$$

By equating the terms of (18) and (20) we can conclude that synchronisation with expansion in the context of RCE necessarily leads to the following two requirements:

$$\min(X, Y) * Z = \min(X * Z, Y * Z) \quad (21)$$

$$P\{X < Y\} = P\{X * Z < Y * Z\} \quad (22)$$

In the face of these requirements, it is interesting to see how the main Markovian process calculi have dealt with them.

PEPA. This stochastic process algebra [27] deals with the situation by rejecting classical expansions like (19). To synthesise choice and synchronisation PEPA takes its recourse to so-called *apparent rates*, which replace the original rates when expanding. Interestingly enough, these rates can be obtained in a general setting by combining RCE with (17) only:

$$\begin{aligned}
((\mathsf{a}, \lambda).B + (\mathsf{a}, \mu).C) \,|\!|_{\mathsf{a}}\!|\, (\mathsf{a}, \nu).D = \quad & \text{(by (7))} \\
(\mathsf{a}, \lambda + \mu).(B \oplus_{\frac{\mu}{\lambda+\mu}} C) \,|\!|_{\mathsf{a}}\!|\, (\mathsf{a}, \nu).D = \quad & \text{(by (17))} \\
(\mathsf{a}, (\lambda + \mu) * \nu).((B \,|\!|_{\mathsf{a}}\!|\, D) \oplus_{\frac{\mu}{\lambda+\mu}} (C \,|\!|_{\mathsf{a}}\!|\, D)) = \quad & \text{(by (7))} \\
\left(\mathsf{a}, \frac{\lambda}{\lambda + \mu}((\lambda + \mu) * \nu)\right).(B \,|\!|_{\mathsf{a}}\!|\, D) + \left(\mathsf{a}, \frac{\mu}{\lambda + \mu}((\lambda + \mu) * \nu)\right).(C \,|\!|_{\mathsf{a}}\!|\, D) \quad & (23)
\end{aligned}$$

The two rate parameters occurring in (23) correspond to Hillston's apparent rates. Note that here, however, they are actually independent of the particular synchronisation function '$*$' that is used. Hillston instantiates '$*$' to the *minimum* of rates (corresponding to the distribution of the *slowest* process): for ν greater than $\lambda + \mu$ we obtain:

$$((\mathsf{a}, \lambda).B + (\mathsf{a}, \mu).C) \,|\!|_{\mathsf{a}}\!|\, (\mathsf{a}, \nu).D = (\mathsf{a}, \lambda).(B \,|\!|_{\mathsf{a}}\!|\, D) + (\mathsf{a}, \mu).(C \,|\!|_{\mathsf{a}}\!|\, D).$$

In the converse case that $\nu < \lambda + \mu$ we get:

$$\begin{aligned}
&((\mathsf{a}, \lambda).B + (\mathsf{a}, \mu).C) \,|\!|_{\mathsf{a}}\!|\, (\mathsf{a}, \nu).D = \\
&\qquad \left(\mathsf{a}, \frac{\lambda\nu}{\lambda + \mu}\right).(B \,|\!|_{\mathsf{a}}\!|\, D) + \left(\mathsf{a}, \frac{\mu\nu}{\lambda + \mu}\right).(C \,|\!|_{\mathsf{a}}\!|\, D).
\end{aligned}$$

TIPP. The requirements (21) and (22) can be reformulated for the Markovian case in terms of rates, where we assume that '$*$' is a function over *rates*, as they uniquely determine exponential distributions. We get:

$$(\lambda + \mu) * \nu = (\lambda * \nu) + (\mu * \nu) \quad (24)$$

$$\frac{\lambda}{\lambda + \mu} = \frac{\lambda * \nu}{(\lambda + \mu) * \nu} \quad (25)$$

An obvious solution to fulfil these requirements is followed in the TIPP algebra [23,21], where '$*$' is interpreted as ordinary multiplication. Although this is a very simple and computationally attractive solution for the synchronisation operator, the operational intuition behind this choice is not at all obvious. There is no useful stochastic interpretation of the multiplication of rates that corresponds to some abstract mechanism for synchronisation. In TIPP, therefore, the interpretation of '$*$' is only motivated by its algebraic simplicity.

Buchholz, who essentially adopts this solution too [11], has given a sophisticated twist to the idea to make it more acceptable. For each action a he stipulates the existence of a systemwide (reference) rate μ_{a}. His action-prefix operators then have the format $(\mathsf{a}, r).B$ where the rate of the associated transition is defined as $r.\mu_{\mathsf{a}}$. In this way r defines the relative capacity of a component w.r.t. an action occurrence. At synchronisation these relative capacities are multiplied, e.g. $(\mathsf{a}, 2) * (\mathsf{a}, 0.5) = (\mathsf{a}, 1)$.

Although, the multiplication idea in most of its forms remains questionable as an operational interpretation of synchronisation, it is attractive from the point of view of system decomposition. When we want to decompose a complicated system into a set of simpler systems, then this may be useful from an analytical point of view, even if its does not have a direct operational (or architectural) interpretation. In much of the work centred around (Kronecker) *product forms*, this approach is therefore, often implicitly, followed [50,15].

EMPA. The stochastic process algebra EMPA [4] deals with synchronisation by imposing some restrictions. It starts from an operational interpretation of synchronisation, viz. that all synchronisations take place in a client/server model, where several clients may request a service represented by synchronisation on a given action and one server grants such requests. In such a setup it is reasonable to assume that the server determines the rate of service, and that the clients are 'passive' in this respect.

Algebraically this can be modelled by assuming that all clients have infinite rates ∞ (in principle they are willing to be served instantaneously), and that synchronisation is interpreted as selecting the minimal rate (i.e. the rate of the slowest process), which ultimately means selecting the rate of the server. In formula's we get:

$$\infty * \infty = \infty \tag{26}$$

$$\lambda * \infty = \infty * \lambda = \lambda \tag{27}$$

Assuming at most one synchronising action carries a rate parameter different from ∞, these properties are consistent with (24) and (25). In this way we obtain expansion laws similar to (19):

$$((\mathsf{a}, \lambda).B + (\mathsf{a}, \mu).C) \parallel \mathsf{a} \parallel (\mathsf{a}, \infty).D = (\mathsf{a}, \lambda).(B \parallel \mathsf{a} \parallel C) + (\mathsf{a}, \mu).(B \parallel \mathsf{a} \parallel C) \tag{28}$$

On the other hand, when applying this principle to multiple passive actions the EMPA approach is not free of complications, as it is unclear how the rate ν should be 'distributed' over the passive components. The naive solution with classical expansion does not work here if one also insists on having the idempotency law (10) for passive actions as EMPA does, e.g.

$$\begin{aligned}
&(\mathsf{a},\lambda).(B \parallel \mathsf{a} \parallel C)\\
&\quad = (\mathsf{a},\infty).B \parallel \mathsf{a} \parallel (\mathsf{a},\lambda).C\\
&\quad = ((\mathsf{a},\infty).B + (\mathsf{a},\infty).B) \parallel \mathsf{a} \parallel (\mathsf{a},\lambda).B\\
&\quad = ((\mathsf{a},\infty).B \parallel \mathsf{a} \parallel (\mathsf{a},\lambda).C + ((\mathsf{a},\infty).B \parallel \mathsf{a} \parallel (\mathsf{a},\lambda).C\\
&\quad = (\mathsf{a},\lambda).(B \parallel \mathsf{a} \parallel C) + (\mathsf{a},\lambda).(B \parallel \mathsf{a} \parallel C)\\
&\quad = (\mathsf{a},2\lambda).(B \parallel \mathsf{a} \parallel C)
\end{aligned}$$

The solution of this problem in the original definition of EMPA was defective; its revision in a more recent definition [6] essentially boils down to the imposition of certain syntactical requirements to avoid such situations.

IMC. Because of its separation between actions and delays IMC essentially manages to avoid the complications with synchronisation of the other calculi. Synchronising actions does not involve the synchronisation of delays, and delay prefixes do not synchronise, but interleave. Consequently, a (rate) synchronisation function '$*$' is not needed.

By itself, however, this does not guarantee that the IMC approach provides a natural model for synchronisation. To see that this is indeed the case, we 'translate' combined prefixes like $(\mathsf{a},\lambda).B$ into their IMC counterparts of the form $(\lambda).\mathsf{a}.B$. If we now look at the induced form of (17) under this translation we get:

$$(\lambda).\mathsf{a}.B \parallel \mathsf{a} \parallel (\mu).\mathsf{a}.C = (\lambda).(\mu).\mathsf{a}.(B \parallel \mathsf{a} \parallel C) + (\mu).(\lambda).\mathsf{a}.(B \parallel \mathsf{a} \parallel C) \qquad (29)$$

The right-hand side indicates that action a will take place after a delay of $(\lambda).(\mu)$ or $(\mu).(\lambda)$, whichever is fastest. This is equivalent to a delay with the distribution of the stochastic value that is the maximum of the two exponential delays. This has a very natural operational interpretation: when synchronising the delay is determined by the slowest synchronisation party. The Markovian process algebras that combine actions and delays cannot handle this situation, because the maximum of two exponential distributions is no longer an exponential distribution itself, and therefore falls outside the scope of the model. As in IMC delays can be represented by combinations of exponential delay transitions, it can accommodate such non-exponential distributions within its model. It can, in fact, represent delays from the much larger class of *phase-type distributions* [43], which can approximate general continuous distributions arbitrarily closely (i.e. it is a *dense* subset of the set of continuous distributions).

6 Conclusion

In this paper we have shown how continuous time Markov chain models can be integrated in the process algebraic framework for the modelling and analysis of reactive systems. To do so, we have reviewed the main ingredients of standard process algebra and introduced the basic concepts of continuous time Markov chains. We have observed how Markov chains can be interpreted as transition systems that can be described by process algebraic means, yielding an algebra of Markov chains.

The proces algebraic treatment of Markov chains immediately induces a compositional framework for their representationa and analysis. Syntactically, (large) Markov chains can be represented as the concurrent composition of simpler chains. Semantically, the stochastic version of strong bisimulation equivalence captures exactly the lumpability criterion for Markov chains that is used to simplify chains by the aggregation of equivalent states. As strong bisimilarity is a congruence relation w.r.t. the process algebraic operators, such simplifications can be carried out componentwise (or compositionally) in the algebraic framework, which greatly improves the practical applicability of the method for large chains.

As a next step we have shown that the algebra of Markov chains itself can be merged successfully with a standard process algebra over actions. In particular we have presented the algebra of Interactive Markov Chains (IMC), which can be used to model systems that have two different types of transitions: Markovian delays (represented by their rates), and actions (represented by their action names). We have shown that in IMC we can define both a strong and a weak variant of stochastic bisimulation. Just like in the standard theory weak bisimilarity is not a congruence w.r.t. the choice operator, but a suitable weak congruence can be identified in the canonical way.

IMC provides a process algebraic framework for the integrated modelling and analysis of both functional and (Markovian) performance aspects of reactive systems. Markov chain models can be obtained from the integrated models by abstraction of all observable system actions and subsequent simplification modulo weak bisimulation. The latter can be done compositionally by applying reduction modulo weak congruence componentwise. Of course, this may involve the resolution of remaining nondeterminism.

In the last part of our survey we have compared IMC with a number of other (Markovian) stochastic process algebra's that have been developed with similar goals. In contrast to IMC the other approaches do not have a separation between action transitions and delays, but instead combine them into composite actions of the form (a, λ), meaning that action a can occur only after an exponentially distributed delay with rate λ. In many respects these algebras are quite comparable to IMC. The main exception is the treatment of action synchronisation, which in IMC is straightforward and follows standard process algebra. The other approaches differ according to the different mechanisms by which the rates of the synchronised actions are determined. In IMC delays are not synchronised but interleaved, which for exponential distributions is equivalent to waiting for

the longest delay. This seems an intuitively natural choice. Because of the separation between delays and actions the treatment of synchronisation in IMC is quite elegant, and in our opinion the preferred approach when the maximal delay interpretation of synchronisation applies.

Acknowledgements. Pedro R. D'Argenio, Ulrich Herzog, Rom Langerak, Joost-Pieter Katoen, Lennard Kerber, Michael Rettelbach, Markus Siegle, and Vassilis Mertsiotakis are all kindly acknowledged for their support, ideas and cooperation in our joint research of stochastic process algebra over the past years. The second author is supported by the Netherlands Organisation of Scientific Research (NWO).

References

1. Jos Baeten and Peter Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Computer Science*. Cambridge University Press, 1990.
2. G. Balbo. Introduction to Stochastic Petri Nets. This volume.
3. J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science Publishers, 2001.
4. M. Bernardo and R. Gorrieri. Extended Markovian Process Algebra. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)*, volume 1119 of *Lecture Notes in Computer Science*. Springer, 1996.
5. T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14, 1987.
6. M. Bravetti and M. Bernardo. Compositional asymmetric cooperations for process algebras with probabilities, priorities, and time. In *Proc. of the 1st International Workshop on Models for Time Critical Systems*, volume 39 (3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2000.
7. M. Bravetti and R. Gorrieri. A complete axiomatisation for observational congruence of prioritized finite state behaviours. In U. Montanari, J.D.P. Rolim, and E. Welzl, editors, *Automata, Languages, and Programming (ICALP)*, volume 1853 of *Lecture Notes in Computer Science*, pages 744–755, Geneva, Switzerland, 2000. Springer.
8. E. Brinksma, J.-P. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. In S. Gilmore and J. Hillston, editors, *Proc. of the 3rd Workshop on Process Algebras and Performance Modelling*. Special Issue of "The Computer Journal", 38(7) 1995.
9. E. Brinksma, A. Rensink, and W. Vogler. Fair Testing. In Insup Lee and Scott Smolka, editors, *Proceedings of 6th International Conference on Concurrency Theory (CONCUR '95, Philadelphia)*, volume 962 of *Lecture Notes in Computer Science*. Springer, 1995.
10. S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.
11. P. Buchholz. Markovian Process Algebra: Composition and Equivalence. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, Regensberg/Erlangen, July 1994. Arbeitsberichte des IMMD, Universität Erlangen-Nürnberg.

12. R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clock. In A. Mazurkievicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory (8th International Conference, Warszaw, Poland, August 1997)*, volume 1243 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 1996.
13. P.R. D'Argenio and C. Verhoef. A conservative extension theorem in process algebras with inequalities. *Theoretical Computer Science*, 177:351–380, 1997.
14. P. Darondeau. An Enlarged Definition and Complete Axiomatisation of Observational Congruence of Finite Processes. In M. Dezani-Ciancaglini and U. Montanari, editors, *International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 1982.
15. S. Donatelli. Superposed Generalised Stochastic Petri Nets: Definition and Efficient Solution. In M. Silva, editor, *Proc. of 15th Int. Conference on Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 258–277. Springer, 1994.
16. N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis using Stochastic Process Algebras. In *Performance'93*, 1993.
17. J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.
18. B. Haverkort. Markovian Models for Performance and Dependability Evaluation. This volume.
19. H. Hermanns. *Interactive Markov Chains*. PhD thesis, Universität Erlangen-Nürnberg, September 1998. Arbeitsberichte des IMMD 32/7.
20. H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 2001. to appear.
21. H. Hermanns, U. Herzog, and V. Mertsiotakis. Stochastic Process Algebras - Between LOTOS and Markov Chains. *Computer Networks and ISDN Systems*, 30(9-10):901–924, 1998.
22. H. Hermanns and J.-P. Katoen. Automated compositional markov chain generation for a plain-old telephone system. *Science of Computer Programming*, 36(1):97–127, 2000.
23. H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, Erlangen-Regensberg, July 1994. IMMD, Universität Erlangen-Nürnberg.
24. H. Hermanns and M. Siegle. Bisimulation Algorithms for Stochastic Process Algebras and their BDD-based Implementation. In J.-P. Katoen, editor, *ARTS'99, 5th Int. AMAST Workshop on Real-Time and Probabilistic Systems*, pages 144–264. Springer, LNCS 1601, 1999.
25. U. Herzog. Formal methods for performance evaluation. This volume.
26. J. Hillston. Exploiting structure in solution: Decomposing compositional models. This volume.
27. J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.
28. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1985.
29. K. Honda and M. Tokoro. On Asynchronous Communication Semantics. In M. Tokoro, O. Nierstrasz, and P. Wegner, editors, *Object-Based Concurrent Computing 1991*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer, 1992.

30. ISO. *LOTOS: A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, 1989.
31. B. Jacobs and J. Rutten. A Tutorial on (Co)Algebras and (Co)Induction. *EATCS Bulletin*, 62:222–259, June 1997.
32. J.-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures*. PhD thesis, Centre for Telematics and Information Technology, Enschede, 1996.
33. J.-P. Katoen and P.R. D'Argenio. General distributions in process algebra. This volume.
34. J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer, 1976.
35. R. Langerak. A testing theory for lotos using deadlock detection. In E. Brinksma, G. Scollo, and C. A. Vissers, editors, *Protocol Specification Testing and Verification IX*, pages 87–98. North-Holland, 1989.
36. R. Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, University of Twente, 1992.
37. R. Milner. Calculi for Synchrony and Asynchrony. *Theoretical Computer Science*, 25:269–310, 1983.
38. R. Milner. A Complete Inference System for a Class of Regular Behaviours. *Journal of Computer and System Science*, 28:439–466, 1984.
39. R. Milner. Process constructors and interpretations. In *Proc. IFIP-WG Information Processing*. North-Holland, 1986.
40. R. Milner. *Communication and Concurrency*. Prentice Hall, London, 1989.
41. U. Montanari and V. Sassone. Dynamic Congruence vs. Progressing Bisimulation for CCS. *Fundamenta Informaticae*, XVI(2):171–199, 1992.
42. V. Natarjan and R. Cleaveland. Divergence and Fair Testing. In *ICALP 95*, volume 944 of *Lecture Notes in Computer Science*, pages 648–659. Springer, 1995.
43. M.F. Neuts. *Matrix-geometric Solutions in Stochastic Models–An Algorithmic Approach*. The Johns Hopkins University Press, 1981.
44. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
45. X. Nicollin and J. Sifakis. An Overview and Synthesis on Timed Process Algebras. In J. W. de Bakker, K. Huizing, and W.-P. de Roever, editors, *Real-Time: Theory in Practice (REX Workshop)*, volume 600 of *Lecture Notes in Computer Science*. Springer, 1990.
46. E.-R. Olderog and C.A.R. Hoare. Specification Oriented Semantics for Communicating Processes. *Acta Informatica*, 23:9–66, 1986.
47. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Fifth GI Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer, 1981.
48. Joachim Parrow and Peter Sjödin. The Complete Axiomatization of cs-Congruence. In P. Enjalbert, E. W. Mayr, and K. W. Wagner, editors, *STACS '94*, volume 775 of *Lecture Notes in Computer Science*, pages 557–568. Springer, 1994.
49. I. Phillips. Refusal testing. *Theoretical Computer Science*, 50(2):241–284, 1987.
50. B. Plateau and K. Atif. Stochastic Automata Network for Modeling Parallel Systems. *IEEE Transactions on Software Engineering*, 17(10), 1991.
51. G.D. Plotkin. A Structured Approach to Operational Semantics. Technical Report DAIMI FM-19, Computer Science Department, Aarhus University, 1981.
52. M. Hennessy und T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.
53. A. Valmari and M. Tienari. Compositional Failure-based Semantic Models for Basic LOTOS. *Formal Aspects of Computing*, 7:440–468, 1995.

54. R. J. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
55. R.J. van Glabbeek. The Linear Time – Branching Time Spectrum II: The Semantics of Sequential Systems With Silent Moves (Extended Abstract). In Eike Best, editor, *Fourth International Conference on Concurrency Theory (CONCUR '93, Hildesheim, Germany)*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.
56. R.J. van Glabbeek. The Linear Time – Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes. In Bergstra et al. [3], pages 3–99.
57. R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121:59–80, 1995.
58. C.A. Vissers, G. Scollo, M. van Sinderen, and E. Brinksma. Specification Styles in Distributed Systems Design and Verification. *Theoretical Computer Science*, 89(1):179–206, 1991.
59. W. Yi. CCS + Time = An Interleaving Model for Real Time Systems. In J. Leach Albert, B. Monien, and M. Rodríguez, editors, *Eighteenth Colloquium on Automata, Languages and Programming (ICALP) (Madrid, Spain)*, volume 510 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 1991.