BISIMULATION OF LABELLED STATE-TO-FUNCTION TRANSITION SYSTEMS COALGEBRAICALLY

DIEGO LATELLA a, MIEKE MASSINK b, AND ERIK P. DE VINK c

a, b CNR – Istituto di Scienza e Tecnologie dell’Informazione ‘A. Faedo’, Pisa
e-mail address: {Diego.Latella, Mieke.Massink}@isti.cnr.it

c Department of Mathematics and Computer Science, Eindhoven University of Technology
Centrum voor Wiskunde en Informatica, Amsterdam
e-mail address: evink@win.tue.nl

ABSTRACT. Labeled state-to-function transition systems, FuTS for short, are characterized by transitions which relate states to functions of states over general semirings, equipped with a rich set of higher-order operators. As such, FuTS constitute a convenient modeling instrument to deal with process languages and their quantitative extensions in particular. In this paper, the notion of bisimulation induced by a FuTS is addressed from a coalgebraic point of view. A correspondence result is established stating that FuTS-bisimilarity coincides with behavioural equivalence of the associated functor. As generic examples, the equivalences underlying substantial fragments of major examples of quantitative process algebras are related to the bisimilarity of specific FuTS. The examples range from a stochastic process language, PEPA, to a language for Interactive Markov Chains, IMcL, a (discrete) timed process language, TPC, and a language for Markov Automata, MAL. The equivalences underlying these languages are related to the bisimilarity of their specific FuTS. By the correspondence result coalgebraic justification of the equivalences of these calculi is obtained. The specific selection of languages, besides covering a large variety of process interaction models and modelling choices involving quantities, allows us to show different classes of FuTS, namely so-called simple FuTS, combined FuTS, nested FuTS, and general FuTS.

1. INTRODUCTION

In the last couple of decades, qualitative process languages have been enriched with quantitative information. In the qualitative case, process languages equipped with formal operational semantics have proven to be successful formalisms for the modeling of concurrent systems and the analysis of their behaviour. Generally, the operational semantics of a qualitative process language are given by means of a labeled transition system (LTS), with states being process terms and actions decorating the transitions between states. Typically, based on the induced
transition system relation, a notion of process equivalence is defined, providing means to compare systems and to reduce their representation to enhance subsequent verification.

Extensions of qualitative process languages allow a deterministic as well as stochastic representation of time, or the use of discrete probability distributions for resolving (some) forms of non-determinism. Among them, languages based on stochastic modeling of action duration or delays, usually referred to as stochastic process algebras, or stochastic process calculi (SPC), are one of the quantitative enrichments of process languages that have received particular attention. For SPC, the main aim has been the integration of qualitative descriptions with quantitative ones in a single mathematical framework, building on the combination of LTS and continuous-time Markov chains (CTMC). The latter is one of the most successful approaches to modeling and performance analysis of (computer) systems and networks. An overview of SPC, equivalences and related analysis techniques may, for example, be found in [28, 5, 6]. A common feature of many SPC is that actions are augmented with the rates of exponentially distributed random variables that characterize their duration. Alternatively, actions are assumed to be instantaneous, in which case random variables are used for modeling delays, as in [27]. Although exploiting the same class of distributions, the models and techniques underlying the definition of the calculi turn out to be significantly different in many respects. A prominent difference concerns the modeling, by means of the choice operator, of the race condition arising from the CTMC interpretation of process behaviour, and its relationship to the issue of transition multiplicity. In the quantitative setting, multiplicities can make a crucial distinction between processes that are qualitatively equivalent. Several different approaches have been proposed for handling transition multiplicity. The proposals range from multi-relations [31, 27], to proved transition systems [45], to LTS with numbered transitions [24, 28], and to unique rate names [19], just to mention a few.

In [15, 17, 16], Latella, Massink et al. proposed a variant of LTS, called Rate Transition Systems (RTS). In LTS, a transition is a triple \((P, \alpha, P')\) where \(P\) and \(\alpha\) are the source state and the label of the transition, respectively, while \(P'\) is the target state reached from \(P\) via a transition labeled with \(\alpha\). In RTS, a transition is a triple of the form \((P, \alpha, \mathcal{P})\). The first and second component are the source state and the label of the transition, as in LTS, while the third component \(\mathcal{P}\) is a continuation function (or simply a continuation in the sequel), which associates a non-negative real value with each state \(P'\). A non-zero value for state \(P'\) represents the rate of the exponential distribution characterizing the average time for the execution of the action represented by \(\alpha\), necessary to reach \(P'\) from \(P\) via the transition. If \(\mathcal{P}\) maps \(P'\) to 0, then state \(P'\) cannot be reached from \(P\) via this transition. The use of continuations provides a clean and simple solution to the transition multiplicity problem and make RTS particularly suited for SPC semantics. In order to provide a uniform account of the many SPC proposed in the literature, in previous joint work of the first two authors, see [18], State-to-Function Labeled Transition Systems (FuTS) have been introduced as a natural generalization of RTS. In FuTS the codomain of the continuations are arbitrary semirings, rather than just the non-negative reals. This provides increased flexibility while preserving basic properties of primitive operations like sum and multiplication. Furthermore, FuTS are equipped with a rich set of (generic) operations on continuation functions, which makes the framework very well suited for a compositional definition of the operational semantics of process calculi, including SPC and models where both non-deterministic behaviour and stochastic delays are modelled, like in the Language of Interactive Markov Chains [27], or even in combination with probabilistic distributions.
over behaviours, as in languages for Markov Automata \cite{53}, besides calculi for deterministic (discrete) timed systems \cite{3}.

In this paper we extend the work presented in \cite{18} in two directions. The first contribution concerns the extension of the FuTS framework by introducing the notions of combined FuTS and nested FuTS. Given label sets $\mathcal{L}_i$ and semirings $\mathcal{R}_i$, a combined FuTS takes the general format $S = (S, (\rightarrow_i)_{i=1}^n)$ with transition relations $\rightarrow_i \subseteq S \times \mathcal{L}_i \times \mathcal{F}\mathcal{S}(S, \mathcal{R}_i)$. In the degenerated case of $n = 1$, we speak of a simple FuTS, which coincides with the definition of FuTS proposed in \cite{18}. Here, $\mathcal{F}\mathcal{S}(S, \mathcal{R}_i)$ is the set of total functions from $S$ to $\mathcal{R}_i$ with finite support, a sub-collection of functions also occurring in other work combining coalgebra and quantitative modeling (see, e.g. \cite{35, 10}). So, a combined FuTS is characterized by the presence of multiple transition relations which allow for a clean definition of the FuTS semantics of languages which integrate different aspects of behaviour, such as non-determinism vs. stochastic time, as is characteristic for Interactive Markov Chains. Using a single transition relation in such a setting requires additional proof obligations ensuring type correctness of transition elements, in particular the continuations, as can be seen in \cite{18}, for example. Instead, for combined FuTS this is ensured by construction. The general format of a so-called nested FuTS over the label set $\mathcal{L}$ and semirings $\mathcal{R}_1, \ldots, \mathcal{R}_n$, for $n > 1$, is a tuple $S = (S, \rightarrow)$ with $\rightarrow \subseteq S \times \mathcal{L} \times \mathcal{F}\mathcal{S}(\ldots \mathcal{F}\mathcal{S}(S, \mathcal{R}_1) \ldots, \mathcal{R}_n)$. For the purposes of the present paper, $n = 2$ suffices; the nested FuTS we consider here are of the form $S = (S, \rightarrow)$ with $\rightarrow \subseteq S \times \mathcal{L} \times \mathcal{F}\mathcal{S}(S, \mathcal{R}_1, \mathcal{R}_2)$. For nested FuTS the transition relation relates functions over states, instead of just states, to continuations. This makes it easy, for instance, to represent non-deterministic choices between probabilistic distributions over behaviours, as it is the case for (the non-timed fragment of languages for) Markov Automata. Finally, product construction for combined FuTS and sequencing construction for nested FuTS can easily be combined giving rise to what one may call general FuTS (or just FuTS, in the sequel), which prove useful for a concise definition of the operational semantics of Markov Automata languages.

We will briefly show how the various types of FuTS can be used conveniently for a clean and compact definition of the fragments of interest of major process languages (more details on this can be found in \cite{18}, which the interested reader is referred to). For combined FuTS, as well as nested FuTS and general FuTS, we also present FuTS bisimilarity, a general notion of bisimilarity, which will also be shown to coincide with the standard bisimilarity of the relevant process languages.

The second direction of investigation presented in this paper consists of a coalgebraic treatment of the various type of FuTS. We will see that a combined FuTS $(S, (\rightarrow_i)_{i=1}^n)$ is a coalgebra of the product of the functors $\mathcal{F}\mathcal{S}(\cdot, \mathcal{R}_i)$. For this to work, we need the relations $\rightarrow_i$ to be total and deterministic for the coalgebraic modeling as a function. This is not a severe restriction at all in the presence of continuation functions: as we will see, the zero-continuation function, which maps every state $s'$ to 0 will be associated to a state $s$ and a transition, in order to indicate that no state $s'$ is reachable from $s$ via that transition, in the usual LTS-sense: if $s$ allows a transition to some state $s_1$ as well as to a state $s_2$, then the continuation function will simply yield a non-zero value for $s_1$ and for $s_2$. Therefore, it is no essential limitation to restrict our investigations to total and deterministic FuTS. For example, by using Boolean functions, we can model non-deterministic behaviour, as done in
Similarly, we see that a (two-level) nested FuTS $(S, \to)$ is a coalgebra of functor $\mathcal{FS}(\mathcal{FS}(\cdot, R_1), R_2)$. 

Next, the notion of $S$-bisimilarity that arises from a FuTS $S$ is compared to the coalgebraic notion of behavioral equivalence. Following a familiar argument, we first establish that the functor associated with a FuTS $S$ possesses a final coalgebra and therefore has an associated notion of behavioural equivalence. Then it is shown that behavioural equivalence of the functor coincides with $S$-bisimilarity, bisimilarity for FuTS. Pivotal for the proof is the absence of multiplicities in the FuTS treatment of quantities at the level of the transitions. In fact, quantities are accumulated in the function values of the continuations and hidden at the higher level of abstraction. It is noted, in the presence of a final coalgebra for FuTS a more general definition of behavioural equivalence based on cospans coincides with the one given here, cf. [37, 51]. Finally, it is worth noting that for the coalgebraic treatment itself of FuTS we propose here it is not necessary for the co-domain of continuations to be semirings; working with monoids would be sufficient. However, the richer structure of semirings is convenient, if not essential, when using continuations and their operators in the formal definition of the FuTS semantics of SPC.

Using the bridge established by the FuTS bisimulation vs. coalgebraic behavioral equivalence correspondence results, we continue by showing for two well-known stochastic process algebras, viz. Hillston’s PEPA [31] and Hermanns’s IML [27], that the standard notions of PEPA strong equivalence and IML strong bisimilarity coincide with bisimilarity of the associated proper simple and combined FuTS, respectively. In turn, this means that the standard notions of strong equivalence and strong bisimilarity coincide with behavioural equivalence when cast in a coalgebraic framework.

PEPA stands out as one of the prominent Markovian process algebras, and IML specifically provides separate prefix constructions for actions and for delays. In passing, the issue of transition multiplicity has to be dealt with. Appropriate lemmas are provided relating the relation-based cumulative treatment with FuTS to the multi-relation-based explicit treatment of PEPA and IML. It is noted that in our treatment below we restrict to the key-fragment of these two SPC. We furthermore provide a combined FuTS semantics for a simple language of deterministically-timed processes, viz. TPC [3] and we show the coincidence between FuTS bisimilarity and the standard equivalence of timed bisimilarity for the language. Finally, we provide a general FuTS semantics for a process language which incorporates non-determinism, discrete probabilities and Markovian randomized delays, i.e. a language for Markov Automata [22, 23]. Also in this case we prove that FuTS bisimulation and Markov Automata bisimulation coincide, adding to the claim that FuTS bisimulation is a natural notion of process identification for SPC.

Related work on coalgebra includes the papers [56, 55, 50]. Additionally, these papers cover measures and congruence formats, a topic not touched upon in the present paper. For what concerns the discrete parts, regarding the correspondence of bisimulations, our work is in line with the approach of the papers mentioned. In the treatment below the bi-algebraic perspective of SOS and bisimulation [55] is left implicit. In [41] an approach similar to ours has been applied to the ULTraS model, a model which shares some features with simple FuTS. In ULTraS posets are used instead of semirings, although a monoidal structure is then implicitly assumed when process equivalences are taken into consideration [7]. Furthermore, in [41] a general GSOS specification format is presented which allows for a ‘syntactic’ treatment of continuations involving so-called weight functions. An interesting direction of
research combining coalgebra and quantities studies various types of weighted automata, including linear weighted automata, and associated notions of bisimulation and languages, as well as algorithms for these notions \[11, 34, 49, 10\]. Klin considers weighted transition systems, labelled transition systems that assign a weight to each transition and develops Weighted GSOS, a (meta-)syntactic framework for defining well-behaved weighted transition systems. For commutative monoids the notion of a weighted transition system compares with our notion of a FuTS, and, when cast in the coalgebraic setting, the associated concept of bisimulation coincides with behavioral equivalence. Weights of transitions of weighted transition systems are computed by induction on the syntax of process terms and by taking into account the contribution of all those GSOS rules that are triggered by the relevant (apparent) weights. Note that such a set of rules is finite. So, in a sense, the computation of the weights is distributed among (the instantiations of) the relevant rules with intermediate results collected (and integrated) in the final weight. In comparison, as mentioned before, in the FuTS approach, the relevant values are manipulated in a more direct way, using the higher-order operators on continuation functions, applying them directly to the continuations in the transitions within the same the semantics definition rules. So, in a sense, the FuTS approach is better suited for a compositional definition of the operational semantics of a wide range of process calculi due to the suitable choice of a rich set of generic operations on continuation functions. In [39] the investigation on the relationship for nested FuTS between FuTS bisimilarity, and behavioural equivalence, and also coalgebraic bisimilarity is presented. In particular, it is shown that the functor type involved preserves weak pullbacks when the underlying semiring satisfies the zero-sum property.

The process languages with stochastic delays we consider in the sequel, involve a multi-way CSP-like parallel operator; components proceed simultaneously when synchronization on an action from the synchronization alphabet that indexes the parallel operator is possible. However, here we do not distinguish between internal and external non-determinism, cf. [33], since an explicit representation of such a distinction is not relevant for the subject of this paper. A coalgebraic treatment of this distinction is proposed in [57], which uses a functor for so-called non-deterministic filter automata, viz. \(\mathcal{P}(\mathcal{P}(\mathcal{A})) \times [\mathcal{A} \rightarrow \mathcal{P}(\cdot)]\) involving partial functions from a set of actions \(\mathcal{A}\) to a finite power-set. Via currying, this can be brought into the form \(\mathcal{F}\mathcal{S}(\cdot, \mathcal{B})^L\) for \(L = \mathcal{P}(\mathcal{P}(\mathcal{A})) \times \mathcal{A}\), fitting the format of the functor for the (simple) FuTS considered here. In [12] processes are interpreted as formal power-series over a semiring in the style of [17]. This allows to compare testing equivalence for a CSP-style language and bisimulation in a Moore automaton. Note that the notions of equivalence addressed in this paper, as often in coalgebraic treatments of process relations, are all strong bisimilarities.

An extended abstract of part of this paper has appeared as [38] where the coalgebraic view of the FuTS approach and its application to PEPA and IML was originally presented. The workshop contribution [40] gives an account of bisimulation of FuTS of specific type and provides a general correspondence result with of FuTS-bisimulation and behavioral equivalence. The present paper covers these ideas in a structured way, going from simple FuTS to combined FuTS and nested FuTS. It includes the presentation of the use of combined FuTS for the definition of the semantics of a language of deterministically timed
processes and the treatment of nested FuTS for the integration of stochastically timed, non-deterministic and probabilistic processes, as in Markov Automata.

For the present paper we assume the reader to have some familiarity with SPC and the application of FuTS for the definition of their semantics. The reader is referred to [18] for an introduction on the subject. Furthermore, in [39] an illustrative definition of a simple, qualitative, process calculus in the FuTS framework is shown. Section 2 provides basic concepts and notation. Simple FuTS are introduced in Section 3 followed by their coalgebraic treatment in Section 4. Simple FuTS are illustrated by the case of PEPA in Section 5 which also covers the correspondence of the respective notions of bisimulation. Section 6 introduces combined FuTS as well as their coalgebraic representation. Sections 7 and 8 treat IMI and TPC. For both SPC, semantics based on combined FuTS are given, and FuTS bisimulation is compared to standard bisimulation. Next, Section 9 introduces nested as well as general FuTS, again tying up with behavior equivalence. In Section 10 a general FuTS is used for the semantics of a Markov Automata language, for which the notion of bisimulation is related to the standard one. Section 11 wraps up and discusses closing remarks.

2. Preliminaries

A tuple $\mathcal{R} = (R, +, 0, \ast, 1)$ is called a semiring if $(R, +, 0)$ is a commutative monoid with neutral element 0, $(R, \ast, 1)$ is a monoid with neutral element 1, $\ast$ distributes over $+$, and $0 \ast r = r \ast 0 = 0$ for all $r \in R$. As examples of a semiring we will use the Booleans $\mathbb{B} = \{\text{false, true}\}$ with disjunction as sum and conjunction as multiplication, the non-negative reals $\mathbb{R}_{\geq 0}$ with the standard operations, and the powerset construct $2^X$ for a set $X$ with intersection and union as sum and multiplication, respectively. We will consider, for a semiring $\mathcal{R}$ and a function $\varphi : X \to \mathcal{R}$, (countable) sums $\sum_{x \in X'} \varphi(x)$ in $\mathcal{R}$, for $X' \subseteq X$. For such a sum to exist we require $\varphi$ to be of finite support, i.e. the support set $spt(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$ is finite. We use the notation $\oplus \varphi$ to denote the value $\sum_{x \in X} \varphi(x)$ in $\mathcal{R}$.

We use the notation $\mathcal{FS}(X, \mathcal{R})$ for the collection of all functions of finite support from the set $X$ to the semiring $\mathcal{R}$. A construct $[x_1 \mapsto r_1, \ldots, x_n \mapsto r_n]$, or more compactly $[x_i \mapsto r_i]_{i=1}^n$, with $x_i \in X$ all distinct and $r_i \in \mathcal{R}$, denotes the mapping that assigns $r_i$ to $x_i$, $i = 1, \ldots, n$, and assigns 0 to all $x \in X \setminus \{x_1, \ldots, x_n\}$. In particular $[]$, or more precisely $[\varnothing]_\mathcal{R}$, is the constant function $x \mapsto 0$ and $D_\mathcal{R} = [x \mapsto 1]$ is the Dirac function on $\mathcal{R}$ for $x \in X$; in the sequel we will often drop the subscript or superscript $\mathcal{R}$ from $[\cdot]_\mathcal{R}$ and $D_\mathcal{R}$, when the semiring $\mathcal{R}$ is clear from the context.

For $\varphi, \psi \in \mathcal{FS}(X, \mathcal{R})$, the function $\varphi + \psi$ is the pointwise sum of $\varphi$ and $\psi$, i.e. $(\varphi + \psi)(x) = \varphi(x) + \psi(x) \in \mathcal{R}$. Clearly, $\varphi + \psi$ is of finite support as are $\varphi$ and $\psi$. Slightly more generally, for functions $\varphi_i \in \mathcal{FS}(X, \mathcal{R})$ where $i = 1, \ldots, n$, we define the function $\sum_{i=1}^n \varphi_i$ in $\mathcal{FS}(X, \mathcal{R})$ by $(\sum_{i=1}^n \varphi_i)(x) = \sum_{i=1}^n \varphi_i(x)$. Given an injective operation $| : X \times X \to X$, we define $\varphi | \psi : X \to \mathcal{R}$, by $(\varphi | \psi)(x) = \varphi(x_1) \ast \psi(x_2)$ if $x = x_1 | x_2$ for some $x_1, x_2 \in X$, and $(\varphi | \psi)(x) = 0$ otherwise. Injectivity of the operation $|$ guarantees that $\varphi | \psi$ is well-defined. Again, $\varphi | \psi$ is of finite support as are $\varphi$ and $\psi$. Such an operation is used in the setting of
syntactic processes \( P \) that may have the form \( P_1 \mid P_2 \) for two processes \( P_1 \) and \( P_2 \) and a syntactic operator \(||\).

We recall some basic definitions from coalgebra. See e.g. [46] for more details. For a functor \( F : \text{Set} \to \text{Set} \) on the category \( \text{Set} \) of sets and functions, a coalgebra \( \mathcal{X} \) of \( F \) is a set \( X \) together with a mapping \( \alpha : X \to F(X) \). A homomorphism between two \( F \)-coalgebras \( \mathcal{X} = (X, \alpha) \) and \( \mathcal{Y} = (Y, \beta) \) is a function \( f : X \to Y \) such that \( F(f) \circ \alpha = \beta \circ f \). An \( F \)-coalgebra \( (\Omega_F, \omega_F) \) is called final or terminal, if there exists, for every \( F \)-coalgebra \( \mathcal{X} = (X, \alpha) \), a unique homomorphism \( [\cdot]_F^X : (X, \alpha) \to (\Omega_F, \omega_F) \). Two elements \( x_1, x_2 \) of an \( F \)-coalgebra \( \mathcal{X} \) are called behavioural equivalent with respect to \( F \) if \( [x_1]_F^X = [x_2]_F^X \), denoted \( x_1 \approx_F^X x_2 \). In the notation \( [\cdot]_F^X \), as well as \( \approx_F^X \), the indication of the specific coalgebra \( \mathcal{X} \) will be omitted when clear from the context.

A functor \( F \) is called accessible if it preserves \( \kappa \)-filtered colimits for some cardinal number \( \kappa \). However, in the category \( \text{Set} \), we have the following characterization of accessibility: for every set \( X \) and any element \( \xi \in FX \), there exists a subset \( Y \subseteq X \) with \( |Y| < \kappa \), such that \( \xi \in FY \). It holds that a functor has a final coalgebra if it is \( \kappa \)-accessible for some cardinal number \( \kappa \). See [2, 1].

A number of proofs of results on process languages \( P \) in this paper rely on so-called guarded induction [30]. Typically, constants \( X \), also called process variables, are syntactic ingredients in these languages. As usual, if \( X := P \), i.e. the constant \( X \) is declared to have the process \( P \) as its body, we require \( P \) to be prefix-guarded, i.e. any occurrence of a constant in the body \( P \) is in the scope of a prefix-construct of the language. Guarded induction assumes the existence of a ‘complexity’ function \( c : P \to \mathbb{N} \) such that \( c(P) = 1 \) if \( P \) is a prefix construct, \( c(P_1 \cdot P_2) > \max\{ c(P_1), c(P_2) \} \) for all other syntactic operators \( \cdot \) of \( P \), and, in particular, \( c(X) > c(P) \) if \( X := P \). For all concrete process languages treated in this paper such a complexity function can be given straightforwardly. See [14] for more detail.

For convenience we collect here a number of abbreviations used in the sequel: \( \text{CTMC} \) and \( \text{DTMC} \) for the standard notions of Continuous-Time Markov Chains and Discrete-Time Markov Chains, respectively; \( \text{LTS} \) for Labelled Transition System, \( \text{RTS} \) for Rate Transition System, and \( \text{FuTS} \) for Labelled State-to-Function Transition System, the extension of \( \text{LTS} \) we focus on in this paper; \( \text{SPC} \) for Stochastic Process Calculus, referring to the class of process algebras featuring a choice construct based on a non-negative exponential distribution; for specific process calculi and semantic models, viz. \( \text{PEPA} \) for Performance Evaluation Process Algebra, \( \text{IMC} \) for Interactive Markov Chains and \( \text{IML} \) for the \( \text{IMC} \)-based language used in this paper, \( \text{TPC} \) for an example Timed Process Calculus, \( \text{MA} \) for Markov Automata and \( \text{MAL} \) for the MA-based language used in this paper.

3. Simple State-to-Function Labelled Transition Systems

Below we introduce simple \( \text{FuTS} \), i.e. \( \text{FuTS} \) with a single transition relation, which are sufficient for the definition of the semantics of many of the relevant stochastic process languages proposed in the literature (see [18] for details).

**Definition 3.1.** A *simple* \( \text{FuTS} \) \( \mathcal{S} \), in full ‘a simple state-to-function labelled transition system’, over label set \( \mathcal{L} \) and semiring \( \mathcal{R} \), is a tuple \( \mathcal{S} = (S, \to) \) where \( \to \) is the transition relation, with \( \to \subseteq S \times \mathcal{L} \times \mathcal{FS}(S, \mathcal{R}) \).
In the sequel we omit the word ‘simple’ when this cannot give rise to confusion. Similar as for state-to-state transitions of LTS, for state-to-function transitions of FuTS we write \( s \xrightarrow{\ell} v \) for \((s, \ell, v) \in \to\). Note that a (simple) FuTS over a label set \( \mathcal{L} \) and a semiring \( \mathcal{R} \) is reminiscent of a weighted automaton [21]. However, for a FuTS no output function is given, as is for a weighted automaton. To stress the relationship between LTS and FuTS we stick to the terminology and notion stemming from LTS.

For a FuTS \( S = (S, \to) \) the set \( S \) is called the set of states or the carrier set. We refer to \( \to \) as the state-to-function transition relation of \( S \) or just as the transition relation. A FuTS \( S \) is called total and deterministic if, for all \( s \in S \) and \( \ell \in \mathcal{L} \), we have \( s \xrightarrow{\ell} v \) for exactly one \( v \in \mathcal{F}\mathcal{S}(S, \mathcal{R}) \). In such a situation, the state-to-function relation \( \to \) corresponds to a function \( S \to \mathcal{L} \to \mathcal{F}\mathcal{S}(S, \mathcal{R}) \). For the remainder of the paper, all FuTS we consider will be total and deterministic. It is noted that Definition 3.1 slightly differs in formulation from the one provided in [18].

As an example, Figure 1 displays a simple FuTS over the action set \( \mathcal{A} \) and the semiring \( \mathbb{R}_{\geq 0} \) of the non-negative real numbers with standard sum and multiplication. The functions \( v_0 \) to \( v_3 \) used in the example have the property that \( \oplus v_i(s) = 1 \), for \( i = 0, \ldots, 3 \). More explicitly, we have

\[
\begin{align*}
    s_0 &\xrightarrow{a} [s_0 \mapsto \frac{1}{2}, s_1 \mapsto \frac{1}{2}] & s_2 &\xrightarrow{a} [s_2 \mapsto \frac{1}{2}, s_3 \mapsto \frac{1}{2}] & s_3 &\xrightarrow{a} [s_0 \mapsto \frac{1}{2}, s_3 \mapsto \frac{1}{2}] \\
    s_1 &\xrightarrow{a} [s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}] & s_1 &\xrightarrow{b} [s_0 \mapsto \frac{1}{6}, s_2 \mapsto \frac{1}{3}, s_3 \mapsto \frac{1}{3}] \\
    s_i &\xrightarrow{b} [\bot] & \text{for } i = 0, 2, 3
\end{align*}
\]

Usually, such a FuTS over \( \mathbb{R}_{\geq 0} \), with its weights adding up to 1, is called a (reactive) probabilistic transition system [24].

Below it will be notationally convenient to consider a (total, deterministic and simple) FuTS as a tuple \((S, \theta)\) with transition function \( \theta : S \to \mathcal{L} \to \mathcal{F}\mathcal{S}(S, \mathcal{R}) \), rather than using the form \((S, \to)\) that occurs more frequently for concrete examples in the literature. We will use the notation with transition functions \( \theta : S \to \mathcal{L} \to \mathcal{F}\mathcal{S}(S, \mathcal{R}) \) to introduce the notion of bisimilarity for a simple FuTS.
Definition 3.2. Let \( S = (S, \theta) \) be a simple FuTS over label set \( \mathcal{L} \) and semiring \( \mathcal{R} \). An equivalence relation \( R \subseteq S \times S \) is called an \( S \)-bisimulation if \( R(s_1, s_2) \) implies
\[
\sum_{t \in [t]_R} \theta(s_1)(\ell)(t') = \sum_{t \in [t]_R} \theta(s_2)(\ell)(t')
\]
for all \( t \in S \) and \( \ell \in \mathcal{L} \), where we use the notation \([t]_R\) to denote the equivalence class of \( t \in S \) with respect to \( R \). Two elements \( s_1, s_2 \in S \) are called \( S \)-bisimilar if \( R(s_1, s_2) \) for some \( S \)-bisimulation \( R \) for \( S \). Notation \( x_1 \preceq x_2 \).

Note that the sums in equation (3.1) exist since the functions \( \theta(s_1)(\ell), \theta(s_2)(\ell) \in \mathcal{F}\mathcal{S}(S, \mathcal{R}) \) are of finite support.

4. Simple FuTS coalgebraically

In this section we will cast simple FuTS in the framework of coalgebras and prove a correspondence result of FuTS bisimilarity and behavioural equivalence for functors of the form \( \mathcal{F}\mathcal{S}(\cdot, \mathcal{R})^\mathcal{L} \) on \textbf{Set}, with \( \mathcal{R} \) a semiring and \( \mathcal{L} \) a set of labels.

Definition 4.1. Let \( \mathcal{L} \) be a set of labels and \( \mathcal{R} \) a semiring. Functor \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} : \textbf{Set} \to \textbf{Set} \) assigns to a set \( X \) the function space \( \mathcal{F}\mathcal{S}(X, \mathcal{R})^\mathcal{L} \) of all functions \( \varphi : \mathcal{L} \to \mathcal{F}\mathcal{S}(X, \mathcal{R}) \) and assigns to a mapping \( f : X \to Y \) the mapping \( \mathcal{U}_{\mathcal{R}}^\mathcal{L}(f) : \mathcal{F}\mathcal{S}(X, \mathcal{R})^\mathcal{L} \to \mathcal{F}\mathcal{S}(Y, \mathcal{R})^\mathcal{L} \) where
\[
\mathcal{U}_{\mathcal{R}}^\mathcal{L}(f)(\varphi)(\ell)(y) = \sum_{x \in f^{-1}(y)} \varphi(\ell)(x)
\]
for all \( \varphi \in \mathcal{F}\mathcal{S}(X, \mathcal{R})^\mathcal{L}, \ell \in \mathcal{L} \) and \( y \in Y \).

Working in the context of FuTS we include the label set \( \mathcal{L} \) in the notation for the functor \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} \). The functor \( \mathcal{F}\mathcal{S}(\cdot, \mathcal{X}) \) itself, for \( \mathcal{X} \) not necessarily a semiring, but a commutative monoid or field instead, have been studied frequently in the literature, see e.g. [25, 34, 10].

Again we rely on \( \varphi(\ell) \in \mathcal{F}\mathcal{S}(X, \mathcal{R}) \) having a finite support for the sum to exist and for \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} \) to be well-defined. We observe that for any simple FuTS \( (S, \theta) \) over \( \mathcal{L} \) and \( \mathcal{R} \) we have \( \theta : S \to \mathcal{L} \to \mathcal{F}\mathcal{S}(S, \mathcal{R}) \). Thus \( (S, \theta) \) can be interpreted as a \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} \)-coalgebra. In the sequel, we will abbreviate \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} \) with \( \mathcal{U} \) whenever \( \mathcal{L} \) and \( \mathcal{R} \) are clear from the context.

As we aim at comparing our notion of bisimilarity for simple FuTS with behavioural equivalence for the functor \( \mathcal{U}_{\mathcal{R}}^\mathcal{L} \), \( \mathcal{U} \) for short, given a set of labels \( \mathcal{L} \) and a semiring \( \mathcal{R} \), we need to check that \( \mathcal{U} \) possesses a final coalgebra. For this, one may adapt the proof for the functor \( \mathcal{F}\mathcal{S}(\cdot, \mathcal{M}) : \textbf{Set} \to \textbf{Set} \) where \( \mathcal{M} \) is a monoid (rather than a semiring) as sketched in [18, 49] to the setting here. An alternative route to showing the existence of a final coalgebra is to verify accessibility. We directly apply the results of [2, Section 5].

Lemma 4.2. For a set of labels \( \mathcal{L} \) and a semiring \( \mathcal{R} \), the functor \( \mathcal{U} \) has a final coalgebra.

Proof. It suffices to show that the \textbf{Set}-functor \( \mathcal{U} \) is accessible for some suitable cardinal number. In fact, \( \mathcal{U} \) is \(|\mathcal{L}| \times \omega\)-accessible: Consider \( \varphi : \mathcal{L} \to \mathcal{F}\mathcal{S}(X, \mathcal{R}) \) in the image of the set \( X \). Let \( Y_\ell \subseteq X \) be the support of \( \varphi(\ell) \in \mathcal{F}\mathcal{S}(X, \mathcal{R}) \) and \( Y = \bigcup_{\ell \in \mathcal{L}} Y_\ell \subseteq X \). Then \( \varphi \) can be seen as an element of \( \mathcal{L} \to \mathcal{F}\mathcal{S}(Y, \mathcal{R}) \), since outside of \( Y \) it holds that \( \varphi \) equals 0 ∈ \( \mathcal{R} \).
Next we establish, for a given simple FuTS $S$, the correspondence of $S$-bisimulation as given by Definition 3.2 and behavioural equivalence induced by $U$. The proof is similar to [10, Theorem 1].

**Theorem 4.3.** Let $S = (S, \theta)$ be a simple FuTS over the label set $L$ and semiring $R$, and $U$ as in Definition 4.1. Then $s_1 \simeq_S s_2 \iff s_1 \approx_U s_2$, for all $s_1, s_2 \in S$.

**Proof.** Let $s_1, s_2 \in S$. We first prove $s_1 \simeq_S s_2 \implies s_1 \approx_U s_2$. Let $R \subseteq S \times S$ be an $S$-bisimulation with $R(s_1, s_2)$. Note $(S, \theta)$ is a $U$-coalgebra. We turn the collection of equivalence classes $S/R$ into a $U$-coalgebra $S_R = (S/R, \varrho_R)$ where

$$\varrho_R([s]_R)(\ell)([t]_R) = \sum_{t' \in [t]_R} \theta(s)(\ell)(t')$$

for $s, t \in S$, and $\ell \in L$. This is well-defined since $R$ is an $S$-bisimulation: if $R(s, s')$ then we have $\sum_{t' \in [t]_R} \theta(s)(\ell)(t') = \sum_{t' \in [t]_R} \theta(s')(\ell)(t')$. The canonical mapping $\varepsilon_R : S \to S/R$ is a $U$-homomorphism: For $\ell \in L$ and $t \in S$, we have

$$U(\varepsilon_R)(\theta(s))(\ell)([t]_R) = \sum_{t' \in \varepsilon_R^{-1}([t]_R)} \theta(s)(\ell)(t') \quad \text{by definition of } U$$

$$= \sum_{t' \in \varepsilon_R^{-1}([t]_R)} \theta(s)(\ell)(t') \quad \text{by definition of } \varepsilon_R$$

$$= \varrho_R([s]_R)(\ell)([t]_R) \quad \text{by definition of } \varrho_R$$

Thus, $U(\varepsilon_R) \circ \theta = \varrho_R \circ \varepsilon_R$, i.e. $\varepsilon_R$ is a $U$-homomorphism. Therefore, by uniqueness of a final morphism, we have $[s]_U = [s]_U \circ \varepsilon_R$. In particular, with respect to $S$, this implies $[s_1]_U = [s_2]_U$ since $\varepsilon_R(s_1) = \varepsilon_R(s_2)$. Thus, $s_1 \approx_U s_2$.

For the reverse, $s_1 \approx_U s_2 \implies s_1 \simeq_S s_2$, assume $s_1 \approx_U s_2$, i.e. $[s_1]_U = [s_2]_U$. Since the map $[\cdot]_U : (S, \theta) \to (\Omega, \omega)$ is a $U$-homomorphism, the equivalence relation $R_S$ with $R_S(s', s'') \iff [s']_U = [s'']_U$ is an $S$-bisimulation: Suppose $R_S(s', s'')$, i.e. $s' \approx_U s''$, for some $s', s'' \in S$. Pick $\ell \in L$, $t \in S$ and assume $[t]_U = w \in \Omega$. Since $[\cdot]_U : (S, \theta) \to (\Omega, \omega)$ is a $U$-homomorphism we have that $\omega \circ [\cdot]_U = U([\cdot]_U) \circ \theta$. Hence, for $s \in S$, it holds that

$$\omega([s]_U)(\ell)(w) = U([s]_U)(\theta(s))(\ell)(w) = \sum_{t' \in [t]_U^{-1}(w)} \theta(s)(\ell)(t') \quad \text{(4.1)}$$

Therefore we have

$$\sum_{t' \in [t]_S} \theta(s')(\ell)(t')$$

$$= \sum_{t' \in [t]_U^{-1}(w)} \theta(s')(\ell)(t') \quad \text{by definition of } R_S \text{ and } w$$

$$= \omega([s']_U)(\ell)(w) \quad \text{by equation (4.1)}$$

$$= \omega([s'']_U)(\ell)(w) \quad s' \approx_U s'' \text{ by assumption}$$

$$= \sum_{t' \in [t]_U^{-1}(w)} \theta(s'')(\ell)(t') \quad \text{by equation (4.1)}$$

$$= \sum_{t' \in [t]_S} \theta(s'')(\ell)(t') \quad \text{by definition of } R_S \text{ and } w$$

Thus, if $R_S(s', s'')$ then $\sum_{t' \in [t]_S} \theta(s')(\ell)(t') = \sum_{t' \in [t]_S} \theta(s'')(\ell)(t')$ for all $t \in S$ and $\ell \in L$, and therefore $R_S$ is an $S$-bisimulation. Since $[s_1]_U = [s_2]_U$, it follows that $R_S(s_1, s_2)$. Thus $R_S$ is an $S$-bisimulation relating $s_1$ and $s_2$. Conclusion, it holds that $s_1 \simeq_S s_2$. \qed
In the next section we will provide FuTS semantics for a fragment of PEPA, a representative process language. For this language we will establish that its standard notion of strong equivalence as known in the literature coincides with the notion of strong bisimulation as induced by the FuTS semantics. The results of this section form the basis for showing that the standard notions of strong equivalence on the one hand, and behavioural equivalence on the other hand, are all the same. The notion of bisimulation for FuTS plays an intermediary role: it bridges between the standard notion of concrete equivalence and the abstraction notions from coalgebra.

5. FuTS Semantics of PEPA

In this section we consider a significant fragment of the Performance Evaluation Process Algebra, PEPA, [31]—which we still call PEPA for simplicity— including the parallel operator implementing the scheme of so-called minimal apparent rates, and provide a FuTS semantics for it. We point out that there is no technical difficulty in extending the FuTS approach to the full language; we do not do so here since its treatment does not yield a conceptual benefit for this paper. We present a FuTS semantics for PEPA in line with [18] and show that PEPA’s notion of equivalence \( \sim_{\text{pepa}} \), called strong equivalence in [31], fits with the bisimilarity induced by the FuTS semantics.

Definition 5.1. The set \( P_{\text{pepa}} \) of PEPA processes is given by the grammar below:

\[
P ::= \text{nil} \mid (a,\lambda).P \mid P + P \mid P \not< A \mid X
\]

where \( a \) ranges over the set of actions \( A \), \( \lambda \) over \( \mathbb{R}_{>0} \), \( A \) over the set of finite subsets of \( A \), and \( X \) over the set of constants \( X \).

For \( X \in X \), the notation \( X := P \) indicates that the process \( P \) is associated with the process constant \( X \). It is required that each occurrence of a process constant in the body \( P \) of the definition \( X := P \) is guarded by a prefix.

PEPA, like many other SPC, e.g. [29] [8], couples actions and rates. The prefix \((a,\lambda)\) of the process \((a,\lambda).P\) expresses that the duration of the execution of the action \( a \in A \) is sampled from a random variable with an exponential distribution of rate \( \lambda \). The CSP-like parallel composition \( P \not< A Q \) of a process \( P \) and a process \( Q \) for a set of actions \( A \subseteq A \) allows for the independent, asynchronous execution of actions of \( P \) or \( Q \) not occurring in the subset \( A \), on the one hand, and requires the simultaneous, synchronized execution of \( P \) and \( Q \) for the actions occurring in \( A \), on the other hand. The transition rules of the FuTS-semantics of the fragment of PEPA we consider here is given in Figure 2, on which we comment below.

Characteristic for the PEPA language is the choice to model parallel composition, or cooperation in the terminology of PEPA, scaled by the minimum of the so-called apparent rates. By doing so, PEPA’s strong equivalence becomes a congruence [31]. Informally, the apparent rate \( r_a(P) \) of an action \( a \) for a process \( P \) is the sum of the rates of all possible \( a \)-executions for \( P \). The apparent rate \( r_a(P) \) can easily be defined recursively on the structure of \( P \) (see [31] Definition 3.3.1) for details). Accordingly, in the sequel we will refer to \( r_a(P) \) as the ‘syntactic’ apparent rate. When considering the parallel composition \( P \not< A Q \), with cooperation set \( A \), an action \( a \) occurring in \( A \) has to be performed by both \( P \) and \( Q \). The rate of such an execution is governed by the slowest of the two processes, on average, in this respect. (One cannot take the slowest process per sample, because such an operation
cannot be expressed as an exponential distribution in general.) Thus \( r_a(P \parallel Q) \) for \( a \in A \) is the minimum \( \min \{ r_a(P), r_a(Q) \} \). Now, if \( P \) schedules an execution of \( a \) with rate \( r_1 \) and \( Q \) schedules a transition of \( a \) with rate \( r_2 \), in the minimal apparent rate scheme the combined execution yields the action \( a \) with rate \( r_1 \cdot r_2 \cdot \text{arf}(P, Q) \). Here, the ‘syntactic’ scaling factor \( \text{arf}(P, Q) \), the apparent rate factor, is defined by

\[
\text{arf}(P, Q) = \frac{\min\{ r_a(P), r_a(Q) \}}{r_a(P) \cdot r_a(Q)}
\]

assuming \( r_a(P), r_a(Q) > 0 \), otherwise \( \text{arf}(P, Q) = 0 \). Organizing the product \( r_1 \cdot r_2 \cdot \text{arf}(P, Q) \) differently as \( r_1/r_a(P) \cdot r_2/r_a(Q) \cdot \min\{ r_a(P), r_a(Q) \} \) we see that for \( P \parallel Q \) the minimum of the apparent rates \( \min\{ r_a(P), r_a(Q) \} \) is adjusted by the relative probabilities \( r_1/r_a(P) \) and \( r_2/r_a(Q) \) for executing \( a \) by \( P \) and \( Q \), respectively.

The FuTS we consider for the semantics of PEPA has been proposed originally in \([18]\). The transition relation is given by the rules in Figure 2. The set of labels involved is \( \Delta_A \) defined by \( \Delta_A = \{ \delta_a \mid a \in A \} \). In the context of the FuTS semantics considered in this paper, we conventionally use the special symbol \( \delta \) for denoting that there is a random delay, with an negative exponential distribution, associated with the action. The underlying semiring for the FuTS for PEPA is the semiring \( \mathbb{R}_{\geq 0} \) of non-negative reals.

**Definition 5.2.** The simple FuTS \( S_{\text{pepa}} = (\mathcal{P}_{\text{pepa}}, \rightarrow_{\text{pepa}}) \) over \( \Delta_A \) and \( \mathbb{R}_{\geq 0} \) has as transition relation the smallest relation satisfying the axioms and rules of Figure 2.

We discuss the rules of \( S_{\text{pepa}} \). The FuTS semantics provides \( \text{nil} \rightarrow_{\text{pepa}} [\|]_{\mathbb{R}_{\geq 0}} \), for every action \( a \), with \( [\|]_{\mathbb{R}_{\geq 0}} \) the 0-function of \( \mathbb{R}_{\geq 0} \). Therefore we have \( \theta_{\text{pepa}}(\text{nil})(\delta_a)(P') = 0 \) for every \( a \in A \) and \( P' \in \mathcal{P}_{\text{pepa}} \), or, in standard terminology, \text{nil} has no transition. For the rated action prefix \( (a, \lambda) \) we distinguish two cases: (i) execution of the prefix in rule (RAPF1); (ii) no execution of the prefix in rule (RAPF2). In the case of rule (RAPF1) the label \( \delta_a \) signifies that the transition involves the execution of the action \( a \). The continuation \([P \mapsto \lambda]\) is the function that assigns the rate \( \lambda \) to the process \( P \). All other processes are assigned 0, i.e. the zero-element of the semiring \( \mathbb{R}_{\geq 0} \). In the second case, rule (RAPF2), for labels \( \delta_b \) with \( b \neq a \), we do have a state-to-function transition, but it is a degenerate one. The two rules for the prefix, in particular having the ‘null-continuation’ rule (RAPF2), support the unified treatment of the choice operator in rule (CHO) and the parallel operator in rules (PAR1) and (PAR2). The treatment of constants is as usual.
The semantics of the choice operator is defined by rule (CHO), where the continuation of process $P + Q$ is given by direct composition—using pointwise sum—of the continuation $\mathcal{P}$ of $P$ and the continuation $\mathcal{Q}$ of $Q$.

Regarding the parallel operator $\parallel$, with respect to some cooperation set $A \subseteq \mathcal{A}$ there are two rules. Now the distinction is between interleaving and synchronization. In the case of a label $\delta_a$ involving an action $a$ not in the subset $A$, either the $P$-operand or the $Q$-operand of $P \parallel Q$ makes progress. For example, the effect of the pattern $\mathcal{P} \parallel \mathcal{D}_Q$ is that the value $\mathcal{P}(P') \cdot 1$ is assigned to a process $P' \parallel Q$, the value $\mathcal{P}(P') \cdot 0 = 0$ to a process $P' \parallel Q'$ for all $Q' \neq Q$, and the value 0 for a process not of the form $P' \parallel Q'$. Note that the syntactic constructor $\parallel : \mathcal{P}_{pepa} \times \mathcal{P}_{pepa} \to \mathcal{P}_{pepa}$ is clearly injective; so, for all functions $\mathcal{P}$ and $\mathcal{Q}$ in $\mathcal{FS}(\mathcal{P}_{pepa}, \mathbb{R}_{\geq 0})$, we can define $\mathcal{P} \parallel \mathcal{Q}$, as described in Section 2.

Here, as in all other rules, the right-hand sides of the transitions only involve functions in $\mathcal{FS}(\mathcal{P}_{pepa}, \mathbb{R}_{\geq 0})$ and operators on them.

For the synchronization case of the parallel construct, assuming $P \overset{\delta_a}{\to}_{pepa} \mathcal{P}$ and $Q \overset{\delta_a}{\to}_{pepa} \mathcal{Q}$, the ‘semantic’ scaling factor $arf(\mathcal{P}, \mathcal{Q})$ is applied to $\mathcal{P} \parallel \mathcal{Q}$. This scaling factor for continuation in $\mathcal{FS}(\mathcal{P}_{pepa}, \mathbb{R}_{\geq 0})$, is, very much similar to its ‘syntactic’ counterpart, given by

$$arf(\mathcal{P}, \mathcal{Q}) = \min\{\oplus \mathcal{P}, \oplus \mathcal{Q}\}$$

provided $\oplus \mathcal{P}, \oplus \mathcal{Q} > 0$, and $arf(\mathcal{P}, \mathcal{Q}) = 0$ otherwise. For a process $R = R_1 \parallel R_2$ we obtain the value $arf(\mathcal{P}, \mathcal{Q}) \cdot (\mathcal{P} \parallel \mathcal{Q}) \cdot (R_1 \parallel R_2) = arf(\mathcal{P}, \mathcal{Q}) \cdot \mathcal{P} \cdot \mathcal{Q} \cdot R_1 \cdot R_2.$

The following lemma establishes the relationship between the ‘syntactic’ and ‘semantic’ apparent rate factors defined on processes and on continuation functions, respectively.

**Lemma 5.3.** Let $P \in \mathcal{P}_{pepa}$ and $a \in \mathcal{A}$. Suppose $P \overset{\delta_a}{\to}_{pepa} \mathcal{P}$. Then $r_a(P) = \oplus \mathcal{P}$. \(\square\)

The proof of the lemma is straightforward (relying on the obvious definition of $r_a(P)$, omitted above, which can be found in [31]). It is also easy to prove, by guarded induction, that the FuTS $\mathcal{S}_{pepa}$ given by Definition 5.2 is total and deterministic.

**Lemma 5.4.** The FuTS $\mathcal{S}_{pepa}$ is total and deterministic. \(\square\)

In view of the lemma it is justified to write $\mathcal{S}_{pepa} = (\mathcal{P}_{pepa}, \theta_{pepa})$. We use the abbreviated notation $\simeq_{pepa}$ for denoting $\simeq_{\mathcal{S}_{pepa}}$, the bisimulation equivalence induced by $\mathcal{S}_{pepa}$.

**Example 5.5.** To illustrate the ease to deal with multiplicities in the FuTS semantics, consider the PEPA processes $P_1 = (a, \lambda).P$ and $P_2 = (a, \lambda).P + (a, \lambda).P$ for some $P \in \mathcal{P}_{pepa}$.

We have that $P_1 \overset{\delta_a}{\to}_{pepa} [P \mapsto \lambda]$ by rule (RAPF1), but $P_2 \overset{\delta_a}{\to}_{pepa} [P \mapsto 2\lambda]$ by rule (RAPF1) and rule (CHO). The latter makes us to compute $[P \mapsto \lambda] + [P \mapsto \lambda]$, which equals $[P \mapsto 2\lambda]$. Thus, in particular we have $P_1 \not\simeq_{pepa} P_2$. Intuitively it is clear that, in general we cannot have $P + P \sim P$ for any reasonable quantitative process equivalence $\sim$ in the Markovian setting. Having twice as many $a$-labelled transitions, the average number for $(a, \lambda).P + (a, \lambda).P$ of executing the action $a$ per time unit is double the average of executing $a$ for $(a, \lambda).P$.

The standard operational semantics of $\text{PEPA}$ is given in Figure 3. The transition relation $\to_{pepa} \subseteq \mathcal{P}_{pepa} \times (\mathcal{A} \times \mathbb{R}_{\geq 0}) \times \mathcal{P}_{pepa}$ is the least relation satisfying the rules. For an
appropriate treatment of the rates, the transition relation is considered as a multi-transition system, where also the number of possible derivations of a transition $P \xrightarrow{a,\lambda}_{pepa} P'$ matters. We stress that such bookkeeping is not needed in the FuTS-approach. In rule (PAR2) we use the 'syntactic' apparent rate factor for PEPA processes.

The so-called total conditional transition rate $q[P,C,a]$ of a PEPA-process \cite{31,32} for a subset of processes $C \subseteq \mathcal{P}_{pepa}$ and $a \in A$ is given by

$$q[P,C,a] = \sum_{Q \in C} \sum_{\lambda} |P \xrightarrow{a,\lambda} Q|.$$  

Here, $|P \xrightarrow{a,\lambda} Q|$ is the multiset of transitions $P \xrightarrow{a,\lambda} Q$ and $|\lambda| P \xrightarrow{a,\lambda} Q$ is the multiset of all $\lambda$'s involved. The multiplicity of $P \xrightarrow{a,\lambda} Q$ is to be interpreted as the number of different ways the transition can be derived using the rules of Figure 3.

We are now ready to define PEPA’s notion of strong equivalence.

**Definition 5.6.** An equivalence relation $R \subseteq \mathcal{P}_{pepa} \times \mathcal{P}_{pepa}$ is called a strong equivalence if

$$q[P_1, [Q]_R, a] = q[P_2, [Q]_R, a]$$

for all $P_1, P_2 \in \mathcal{P}_{pepa}$ such that $R(P_1, P_2)$, all $Q \in \mathcal{P}_{pepa}$ and all $a \in A$. Two processes $P_1, P_2 \in \mathcal{P}_{pepa}$ are strongly equivalent if $R(P_1, P_2)$ for a strong equivalence $R$, notation $P_1 \sim_{pepa} P_2$.

The next lemma couples, for a PEPA-process $P$, an action $a$ and a continuation function $\mathcal{P} \in FS(\mathcal{P}_{pepa}, \mathbb{R}_{\geq 0})$, the evaluation $\mathcal{P}(P')$ with respect to the FuTS-semantics to the cumulative rate for $P$ of reaching $P'$ by a transition involving the label $a$ in the standard operational semantics. The lemma is pivotal in relating FuTS bisimulation and standard bisimulation for PEPA in Theorem 5.8 below.

**Lemma 5.7.** Let $P \in \mathcal{P}_{pepa}$ and $a \in A$. Suppose $P \xrightarrow{\delta_a}_{pepa} \mathcal{P}$. The following holds: $\mathcal{P}(P') = \sum |\lambda| P \xrightarrow{a,\lambda} P'|$ for all $P' \in \mathcal{P}_{pepa}$.

**Proof.** Guarded induction on $P$. We only treat the cases for the parallel composition. Note, the operation $\boxplus : \mathcal{P}_{pepa} \times \mathcal{P}_{pepa} \to \mathcal{P}_{pepa}$ with $\boxplus(P_1, P_2) = P_1 \boxplus P_2$ is injective. Recall, for $\mathcal{P}_1, \mathcal{P}_2 \in FS(\mathcal{P}_{pepa}, \mathbb{R}_{\geq 0})$, we have $(\mathcal{P}_1 \boxplus \mathcal{P}_2)(P_1 \boxplus P_2) = \mathcal{P}_1(P_1) \cdot \mathcal{P}_2(P_2)$.

Suppose $a \notin A$. Assume $P_1 \xrightarrow{\delta_a}_{pepa} \mathcal{P}_1$, $P_2 \xrightarrow{\delta_a}_{pepa} \mathcal{P}_2$, $P_1 \boxplus P_2 \xrightarrow{\delta_a}_{pepa} \mathcal{P}$. We distinguish three cases.
With the lemma in place we can prove the following correspondence result for \( S_{\text{pepa}} \)-bisimilarity and strong equivalence as given by Definition 5.6.

Case (I), \( P' = P_1' \not\cong R P_2, P_1' \neq P_1 \). Then we have
\[
\sum \{ \lambda \mid P_1 \not\cong \not\cong_a P_2 \xrightarrow{a,\lambda} \text{pepa} \ P' \} \\
= \sum \{ \lambda \mid P_1 \xrightarrow{a,\lambda} \text{pepa} \ P_1' \} \quad \text{by rule (PAR1a)} \\
= \mathcal{P}(P_1') \quad \text{by the induction hypothesis} \\
= (\mathcal{P}(P_1') \cdot \mathcal{D}(P_2)) + (\mathcal{D}(P_1') \cdot \mathcal{P}(P_2)) \\
= (\mathcal{P}(P_1') \cdot \mathcal{D}(P_2)) + (\mathcal{D}(P_1') \cdot \mathcal{P}(P_2)) \\
= (\mathcal{P}(P_1') \cdot \mathcal{D}(P_2)) + (\mathcal{D}(P_1') \cdot \mathcal{P}(P_2)) \\
= \mathcal{P}(P') \quad \text{again by rule (PAR1)}
\]

Case (II), \( P' = P_1 \not\cong \not\cong_a P_2', P_1' \neq P_2 \): similar.

Case (III), \( P' = P_1 \not\cong \not\cong_a P_2 \). Then we have:
\[
\sum \{ \lambda \mid P_1 \not\cong \not\cong_a P_2 \xrightarrow{a,\lambda} \text{pepa} \ P' \} \\
= (\sum \{ \lambda \mid P_1 \xrightarrow{a,\lambda} \text{pepa} \ P_1 \}) + (\sum \{ \lambda \mid P_2 \xrightarrow{a,\lambda} \text{pepa} \ P_2 \}) \quad \text{by rules (PAR1a) and (PAR1b)} \\
= \mathcal{P}(P_1) + \mathcal{P}(P_2) \quad \text{by the induction hypothesis} \\
= (\mathcal{P}(P_1) \cdot \mathcal{D}(P_2)) + (\mathcal{D}(P_1) \cdot \mathcal{P}(P_2)) \\
= (\mathcal{P}(P_1) \cdot \mathcal{D}(P_2)) + (\mathcal{D}(P_1) \cdot \mathcal{P}(P_2)) \\
= \mathcal{P}(P') \quad \text{again by rule (PAR1)}
\]

Suppose \( a \in A \). Assume \( P_1 \xrightarrow{\delta_a} \text{pepa} \mathcal{P}_1, P_2 \xrightarrow{\delta_a} \text{pepa} \mathcal{P}_2, P_1 \not\cong \not\cong_a P_2 \xrightarrow{\delta_a} \text{pepa} \mathcal{P} \). Without loss of generality, \( P' = P_1' \not\cong \not\cong_a P_2' \) for suitable \( P_1', P_2' \in \mathcal{P}_{\text{pepa}} \).
\[
\sum \{ \lambda \mid P_1 \not\cong \not\cong_a P_2 \xrightarrow{a,\lambda} \text{pepa} \ P' \} \\
= \sum \{ \lambda \mid \text{arf}(P_1, P_2) \cdot \lambda_1 \cdot \lambda_2 \mid P_1 \xrightarrow{a,\lambda_1} \text{pepa} \ P_1', P_2 \xrightarrow{a,\lambda_2} \text{pepa} \ P_2' \} \quad \text{by rule (PAR2)} \\
= \text{arf}(P_1, P_2) \cdot (\sum \{ \lambda \mid P_1 \xrightarrow{a,\lambda_1} \text{pepa} \ P_1' \}) \cdot (\sum \{ \lambda \mid P_2 \xrightarrow{a,\lambda_2} \text{pepa} \ P_2' \}) \quad \text{by distributivity} \\
= \text{arf}(P_1, P_2) \cdot (\mathcal{P}(P_1') \cdot \mathcal{P}(P_2')) \quad \text{by the induction hypothesis} \\
= \text{arf}(\mathcal{P}_1, \mathcal{P}_2) \cdot \mathcal{P}(P_1') \cdot \mathcal{P}(P_2') \quad \text{by Lemma 5.3} \\
= \text{arf}(\mathcal{P}_1, \mathcal{P}_2) \cdot (\mathcal{P}_1 \not\cong \not\cong_a \mathcal{P}_2)(P_1' \not\cong \not\cong_a P_2') \quad \text{definition \( \not\cong \) on } \mathcal{F}(\mathcal{P}_{\text{pepa}}, \mathbb{R}_{\geq 0}) \\
= \mathcal{P}(P') \quad \text{by rule (PAR2)}
\]

The other cases are simpler and omitted here. \( \square \)

With the lemma in place we can prove the following correspondence result for \( S_{\text{pepa}} \)-bisimilarity and strong equivalence as given by Definition 5.6.
Theorem 5.8. For PEPA-processes $P_1, P_2 \in P_{pepa}$, it holds that $P_1 \sim_{pepa} P_2$ iff $P_1 \sim_{pepa} P_2$.

**Proof.** Let $R$ be an equivalence relation on $P_{pepa}$. Choose $P, Q \in P_{pepa}$ and $a \in A$. Suppose $P \xrightarrow{\delta_a} P \Rightarrow R \Rightarrow \delta_{a}$. Thus $\theta_{pepa}(P)(\delta_a) = \emptyset$. We have

$$q[P, [Q]_R, a] = \sum Q' \in [Q]_R \sum \{ \| P \xrightarrow{a, \lambda} pepa Q' \} \text{ by definition } q[P, [Q]_R, a]$$

$$= \sum Q' \in [Q]_R \emptyset(Q') \text{ by Lemma 5.7}$$

$$= \sum Q' \in [Q]_R \theta_{pepa}(P)(a)(Q') \text{ by definition } \theta_{pepa}$$

Therefore, for PEPA-processes $P_1$ and $P_2$ it holds that $q[P_1, [Q]_R, a] = q[P_2, [Q]_R, a]$ for all $Q \in P_{pepa}$, $a \in A$ iff $\sum Q' \in [Q]_R \theta_{pepa}(P_1)(a)(Q') = \sum Q' \in [Q]_R \theta_{pepa}(P_2)(a)(Q')$ for all $Q \in P_{pepa}$, $a \in A$. Thus, the equivalence relation $R$ is a strong equivalence (Definition 5.6) iff $R$ is an $S_{pepa}$-bisimulation (Definition 3.2), from which the theorem follows. \hfill \square

By the theorem the FuTS semantics for PEPA of Definition 5.2 is correct with respect to PEPA’s standard semantics of Figure 3. However, because of the use of continuation functions, the former does not involve implicit counting, decorations or multisets. From the general results on FuTS of the previous section, we also obtain a coalgebraic semantics for PEPA for which behavioral equivalence coincides with strong equivalence as defined in [31].

6. Combined FuTS

In the sequel of this article we will deal with a number of calculi and models that mix non-deterministic behaviour with stochastic or deterministic time or with probabilistic delays. In this section, we introduce the notion of a combined FuTS, which allows for a clean definition of the semantics of calculi where different aspects of behaviour are integrated in an orthogonal way. Prominent examples of such calculi are IML, a language for IMC where non-determinism is integrated with stochastic continuous delays (see Section 7) and TPC, a language where non-determinism is integrated with deterministic discrete delays (see Section 5).

Definition 6.1. A combined FuTS $S$, in full ‘a combined state-to-function labeled transition system’, over a number of label sets $L_i$ and semirings $R_i$, $i = 1, \ldots, n$, is a tuple $S = (S, \rightarrow_i)_{i=1}^n$ with set of states $S$ and such that $\rightarrow_i \subseteq S \times L_i \times FS(S, R_i)$, for $i = 1, \ldots, n$.

Combined FuTS of Definition 6.1 extend the simple ones of Definition 3.1. Note, a combined FuTS is defined over a number of label sets and semirings, and, accordingly, gives rise to the same number of transition relations. Thus, a combined FuTS can be seen as a multi-dimensional simple FuTS. The underlying idea is that the behaviour model given by a combined FuTS is such that one can identify different types of labels, assuming disjoint label sets $L_1, \ldots, L_n$. Then, the continuation function of a transition labeled with an element of $L_i$ is taken from $FS(S, R_i)$, expressing the association of the label set $L_i$ with the semiring $R_i$.

For example, in the case of IML, with set of processes $P_{iml}$, both non-deterministic behaviour and stochastically-timed behaviour are treated. Furthermore, action execution is intended to be instantaneous, while stochastic time is characterized by the rates of negative exponential distributions. Consequently, it is convenient to use two label sets, namely a set
of actions $\mathcal{A}$ and a singleton set $\Delta = \{ \delta \}$ where the symbol $\delta$ is used as label to indicate that the transition involves some exponentially distributed delay. The relevant semirings will be $\mathbb{B}$, used for modeling the purely non-deterministic aspects of behaviour, and $\mathbb{R}_{\geq 0}$, used for the rates characterizing the stochastic aspects of behaviour, as in the case of PEPA, but here without any association of delay and actions. Consequently, for IML there will be two transition relations: $\rightarrow_1 \subseteq \mathcal{P}_{\text{iml}} \times \mathcal{A} \times \mathcal{FS}(\mathcal{P}_{\text{iml}}, \mathbb{B})$ modeling non-deterministic behaviour, and $\rightarrow_2 \subseteq \mathcal{P}_{\text{iml}} \times \Delta \times \mathcal{FS}(\mathcal{P}_{\text{iml}}, \mathbb{R}_{\geq 0})$ modeling stochastic-time behaviour.

It is worth pointing out here that one could use an alternative approach instead of taking resort to combined FuTS, namely one based on disjoint unions of label sets, and respectively, continuation functions. Letting $\bigoplus_{i=1}^{n} X_i$ denote the disjoint union of sets $X_i, i = 1, \ldots, n$, one could use a single transition relation

$$
\rightarrow \subseteq S \times \bigoplus_{i=1}^{n} L_i \times \bigoplus_{i=1}^{n} \mathcal{FS}(S, R_i)
$$

satisfying the additional property that $v \in \mathcal{FS}(S, R_i)$ if $\ell \in L_i$, for all transitions $s \xrightarrow{\ell} v$.

As a matter of fact, this approach based on disjoint unions and a single transition relation has been used in [18]. Technically, the two approaches are equivalent. On the other hand, in the definition with a single transition relation, type compatibility between labels and continuation functions yields an additional proof obligation for the well-definedness the definition of the operational semantics for every specific process calculus (the interested reader is referred to [18] for details). The use of an approach with multiple transition relations instead, automatically guarantees type compatibility, viz. by definition. Furthermore, the approach based on disjoint unions appears less amenable to a category-theoretical treatment. For the reasons mentioned we stick to the format of Definition 6.1 in this paper.

As we will see, for the purposes of the present paper it is sufficient to consider only total and deterministic combined FuTS, i.e. those where every transition relation $\rightarrow_i$ is a total function. Consequently, it will be notionally convenient to consider a combined FuTS $S = (S, \langle \rightarrow_i \rangle_{i=1}^{n})$ as a tuple $(S, \langle \theta_i \rangle_{i=1}^{n})$ with transition functions $\theta_i : S \rightarrow L_i \rightarrow \mathcal{FS}(S, R_i)$, for $i = 1, \ldots, n$, rather than using the form $(S, \langle \rightarrow_i \rangle_{i=1}^{n})$ that occurs more frequently for concrete examples in the literature. In the sequel, we occasionally omit the qualification ‘combined’ for a combined FuTS when this cannot cause confusion. All relevant definitions and results presented in Sections 3 and 4 can be extended straightforwardly to combined FuTS. We refer to [39] for details on the extension of definitions, results and their proofs. Here we recall the most important ones.

**Definition 6.2.** For a combined FuTS $S = (S, \langle \theta_i \rangle_{i=1}^{n})$, an $S$-bisimulation is an equivalence relation $R \subseteq S \times S$ such that $R(s_1, s_2)$ implies

$$
\sum t' \in [t]_R \theta_i(s_1)(\ell)(t') = \sum t' \in [t]_R \theta_i(s_2)(\ell)(t')
$$

for all $t \in S$ and $\ell \in L_i, i = 1, \ldots, n$. Two elements $s_1, s_2 \in S$ are called $S$-bisimilar for the combined FuTS $S$ if $R(s_1, s_2)$ for some $S$-bisimulation $R$ for $S$. Notation $s_1 S s_2$.

Working with total and deterministic FuTS, we can interpret a combined FuTS $S = (S, \langle \theta_i \rangle_{i=1}^{n})$ over the label sets $L_i$ and semirings $R_i, i = 1, \ldots, n$, as a product $\theta_1 \times \cdots \times \theta_n : S \rightarrow \prod_{i=1}^{n} (L_i \rightarrow \mathcal{FS}(S, R_i))$ of functions $\theta_i : S \rightarrow L_i \rightarrow \mathcal{FS}(S, R_i)$. To push this idea a bit further, we want to consider the combined FuTS $S = (S, \langle \theta_i \rangle_{i=1}^{n})$ as a coalgebra of a suitable product functor on sets.
**Definition 6.3.** Let $L = \langle \mathcal{L}_1, \ldots, \mathcal{L}_n \rangle$ be an $n$-tuple of label sets and $R = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ be an $n$-tuple of semirings. The functor $\nu^L_R$ on $\text{Set}$ is defined by $\nu^L_R = \prod_{i=1}^n \mathcal{S} \times \mathcal{R}_i$. Referring to Definition 4.1, we have $\mathcal{F} \mathcal{S}(\cdot, \mathcal{R}_i)^{\mathcal{L}_i} = \mathcal{U}^{\mathcal{R}_i}_n$, for $i = 1, \ldots, n$. Therefore, $\nu^L_R = \prod_{i=1}^n \mathcal{U}^{\mathcal{R}_i}_n$. We note that any combined FuTS $\mathcal{S} = (S, \langle \theta_i \rangle_{i=1}^n)$ over label sets $\mathcal{L}_i$ and semirings $\mathcal{R}_i$, for $i = 1, \ldots, n$, is in fact a $\nu^L_R$-coalgebra. Reversely, every $\nu^L_R$-coalgebra, for $L = \langle \mathcal{L}_1, \ldots, \mathcal{L}_n \rangle$ and $R = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, corresponds to a combined FuTS over the label sets $\mathcal{L}_i$ and semirings $\mathcal{R}_i$, for $i = 1, \ldots, n$. Below we shall use $\mathcal{V}$ as an abbreviation for $\nu^L_R$ whenever $L = \langle \mathcal{L}_1, \ldots, \mathcal{L}_n \rangle$ and $R = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ are clear from the context. Similarly, for the sake of readability, we shall often abbreviate $\mathcal{U}^{\mathcal{R}_i}_n$ by $\mathcal{U}_i$.

As product of accessible functors, the functor $\mathcal{V}$ of Definition 6.3 is accessible and possesses a final coalgebra, $(\Omega, \omega)$ say. So, we can speak of the behavioural equivalence $\approx_\mathcal{V}$ on any $\mathcal{V}$-coalgebra or, equivalently, of any combined FuTS $\mathcal{S}$. Moreover, writing $[\cdot]_\mathcal{V}$ for the final morphism of a $\mathcal{V}$-coalgebra $\mathcal{S}$ into $(\Omega, \omega)$, we have

$$[\cdot]_\mathcal{V} = [\cdot]_{\mathcal{U}_1} \times \cdots \times [\cdot]_{\mathcal{U}_n}$$

Next we establish for a given FuTS $\mathcal{S}$ over $\mathcal{L}_1, \ldots, \mathcal{L}_n$ and $\mathcal{R}_1, \ldots, \mathcal{R}_n$ the correspondence of $\mathcal{S}$-bisimulation $\simeq_\mathcal{S}$ and the behavioural equivalence $\approx_\mathcal{V}$ for the functor $\mathcal{V}$. Thus, one may argue, Definition 6.2 provides an explicit description of behavioral equivalence. The proof of the theorem below for combined FuTS is an adaptation of the proof of Theorem 4.3 for simple ones (see [39] for details).

**Theorem 6.4.** Let $\mathcal{S} = (S, \langle \theta_i \rangle_{i=1}^n)$ be a FuTS over the label sets $\mathcal{L}_i$ and semirings $\mathcal{R}_i$, $i = 1, \ldots, n$, and $\mathcal{V}$ as in Definition 6.3. Then $s_1 \simeq_\mathcal{S} s_2 \Leftrightarrow s_1 \approx_\mathcal{V} s_2$, for all $s_1, s_2 \in S$. □

In the sequel of the paper we will consider combined FuTS, as well as a so-called general FuTS, for concrete process languages. We will show for each process language that the notion of bisimulation of its FuTS coincides with the notion of strong bisimulation that is associated in the literature with the language. Consequently, as a corollary of Theorem 6.4 we obtain that the notions of strong bisimulations align with behavioral equivalence.

### 7. FuTS Semantics of IML

In this section we provide a FuTS semantics for a relevant part of IML, the language of Interactive Markov Chains [27], IMC for short, and compare the notion of bisimulation induced by its FuTS to the standard notion of bisimulation based on the SOS-semantics as reported in the literature.

IMC are automata that combine two types of transitions: interactive transitions that involve the execution of actions, and Markovian transitions that represent the progress of time governed by exponential distributions. As a consequence, IMC embody both nondeterministic behaviour and stochastic, i.e. stochastically timed, behaviour. System analysis using IMC proves to be a powerful approach because of the orthogonality of qualitative and quantitative dynamics, their logical underpinning and tool support, cf. [11][30][13]. Such orthogonality makes it natural to use a combined FuTS for the semantics of IML. A number of behavioural equivalences, both strong and weak, are available for IMC [22]. In our treatment here, we discuss a sublanguage of IML, which we still call IML for simplicity. In particular we do not deal with internal $\tau$-steps, since we focus on strong bisimilarity here. The FuTS semantics we consider in the sequel has been originally proposed in [18].
\[(\text{NIL1}) \quad \frac{a \in A}{\text{nil} \rightarrow_1 \emptyset} \quad \text{(NIL2)} \quad \frac{\delta}{\text{nil} \rightarrow_2 \mathbb{R}_{\geq 0}}\]

\[(\text{RPF1}) \quad \frac{a \in A}{\lambda.P \rightarrow_1 \emptyset} \quad \text{(RPF2)} \quad \frac{\lambda.P \rightarrow_2 [P \mapsto \lambda]}{\delta}\]

\[(\text{APF1}) \quad \frac{a.P \rightarrow_1 [P \mapsto \text{true}]}{\lambda.P \rightarrow_1 \emptyset} \quad \text{(APF2)} \quad \frac{\delta \neq a}{a.P \rightarrow_2 \emptyset} \quad \text{(APF3)} \quad \frac{a.P \rightarrow_2 [\lambda \mapsto \lambda]}{\delta} \]

\[(\text{CHO1}) \quad \frac{P \rightarrow_1 \rho}{P + Q \rightarrow_1 \rho + \delta} \quad \text{(CHO2)} \quad \frac{P \rightarrow_2 \rho}{P + Q \rightarrow_2 P + \rho}\]

\[(\text{PAR1}) \quad \frac{P \parallel_A Q \rightarrow_1 (\rho \parallel_A D_Q) + (D_P \parallel_A \delta)}{P \parallel_A Q \rightarrow_1 (\rho \parallel_A D_Q) + (D_P \parallel_A \delta)} \quad \text{(PAR2)} \quad \frac{P \parallel_A Q \rightarrow_2 \rho \parallel_A \delta}{P \parallel_A Q \rightarrow_2 \rho \parallel_A \delta} \quad \text{(PAR3)} \quad \frac{P \parallel_A Q \rightarrow_2 \rho \parallel_A \delta}{P \parallel_A Q \rightarrow_2 \rho \parallel_A \delta}\]

\[(\text{CON1}) \quad \frac{X \rightarrow_1 Y}{X \rightarrow_2 Y} \quad \text{(CON2)} \quad \frac{X \rightarrow_1 Y}{X \rightarrow_2 Y}\]

Figure 4: FuTS Transition Deduction System for IML.

**Definition 7.1.** The set $\mathcal{P}_\text{iml}$ of IML processes is given by the grammar

\[P ::= \text{nil} | a.P | \lambda.P | P + P | P \parallel_A P | X\]

where $a$ ranges over the set of actions $A$, $\lambda$ over $\mathbb{R}_{>0}$, $A$ over the set of finite subsets of $A$ and $X$ over the set of constants $X$.

We assume the same notation and (action) guardedness requirements for constant definitions and usage as in Section 5 for PEPA.

In line with the discussion above, in IML there are separate prefix constructions for actions $a.P$ (meaning that the process *instantaneously performs* action $a$ and then behaves like $P$) and for time-delays $\lambda.P$ (meaning that the process is *delayed* for a period of time governed by a random variable with negative exponential distribution with rate $\lambda$ and then behaves like $P$). No restriction is imposed on the alternative and parallel composition of processes. For example, in IML, we have the process $a.\lambda.\text{nil} + \mu.b.\text{nil}$. With respect to the FuTS semantics to be defined below, we will see that this process admits both a non-trivial interactive transition and a non-trivial Markovian transition,

\[a.\lambda.\text{nil} + \mu.b.\text{nil} \rightarrow_1 [\lambda.\text{nil} \mapsto \text{true}] + \emptyset = [\lambda.\text{nil} \mapsto \text{true}]\]

\[a.\lambda.\text{nil} + \mu.b.\text{nil} \rightarrow_2 [\mathbb{R}_{\geq 0} + [b.\text{nil} \mapsto \mu] = [b.\text{nil} \mapsto \mu]\]

leading to an interactive continuation and a Markovian continuation, respectively.

**Definition 7.2.** The FuTS semantics of $\mathcal{P}_\text{iml}$ is given by the FuTS $\mathcal{S}_\text{iml} = (\mathcal{P}_\text{iml}, \rightarrow_1, \rightarrow_2)$, a combined FuTS over the label sets $A$ and $\Delta = \{\delta\}$ and the semirings $\mathbb{B}$ and $\mathbb{R}_{\geq 0}$ with transition relations $\rightarrow_1 \subseteq \mathcal{P}_\text{iml} \times A \times \mathcal{FS}(\mathcal{P}_\text{iml}, \mathbb{B})$ and $\rightarrow_2 \subseteq \mathcal{P}_\text{iml} \times \Delta \times \mathcal{FS}(\mathcal{P}_\text{iml}, \mathbb{R}_{\geq 0})$ defined as the least relations satisfying the rules of Figure 4.
Actions $a \in \mathcal{A}$ decorate $\rightarrow_1$, the special symbol $\delta$, with $\delta$ for delay, decorates $\rightarrow_2$. Note that rule (APF3) and rule (RPF1) involve the null-functions of $\mathbb{R}_{\geq 0}$ and of $\mathbb{B}$, respectively, to express that a process $a.P$ does not trigger a delay and a process $\lambda.P$ does not execute any action. In Figure 5 and in the rest of this section we use $\mathcal{P}, \mathcal{Q} \in \mathcal{FS}(\mathcal{P}_{iml}, \mathbb{R}_{\geq 0})$ as typical interactive continuations, and $\mathcal{P}, \mathcal{Q} \in \mathcal{FS}(\mathcal{P}_{iml}, \mathbb{R}_{\geq 0})$ as typical Markovian continuations.

Below we use functions $T$ and $R$ based on $\rightarrow$ and $\rightarrow$-, cf. [30]. We have $T: \mathcal{P}_{iml} \times \mathcal{A} \times 2^{\mathcal{P}_{iml}} \rightarrow \mathbb{B}$ given by $T(P, a, C) = \text{true}$ if the set $\{ P' \in C : P \overset{a}{\rightarrow} P' \}$ is non-empty, for all $P \in \mathcal{P}_{iml}$, $a \in \mathcal{A}$ and any subset $C \subseteq \mathcal{P}_{iml}$. For $R: \mathcal{P}_{iml} \times \mathcal{P}_{iml} \rightarrow \mathbb{R}_{\geq 0}$ we put

$$
\text{Example 7.3. For } a.(\lambda.\text{nil} + b.\text{nil}), \mu.a.\text{nil} \in \mathcal{P}_{iml} \text{ and } A = \{a\} \text{ we have}
$$

$$
a.(\lambda.\text{nil} + b.\text{nil}) \parallel_A \mu.a.\text{nil}
$$

$$
\rightarrow_2 \begin{cases}
\parallel_{\mathbb{R}_{\geq 0}} A \cdot D_{\mu.a.\text{nil}} + D_{a.(\lambda.\text{nil} + b.\text{nil})} & \text{if } R = R_1 \parallel_A R_2 \text{ for some } R_1, R_2 \in \mathcal{P}_{iml} \\
\text{false} & \text{otherwise}
\end{cases}
$$

and, on the other hand,

$$
(\mathcal{P} \parallel_A \mathcal{Q})(R) = \begin{cases}
\mathcal{P}(R_1) \cdot \mathcal{Q}(R_2) & \text{if } R = R_1 \parallel_A R_2 \text{ for some } R_1, R_2 \in \mathcal{P}_{iml} \\
0 & \text{otherwise}
\end{cases}
$$

where $\cdot$ is the product in $\mathbb{R}_{\geq 0}$.

It is not difficult to verify that $\mathcal{S}_{iml}$ is a total and deterministic combined $\text{FuTS}$.\hfill\square

**Lemma 7.4.** The $\text{FuTS} \mathcal{S}_{iml}$ is total and deterministic.

Below we use $\mathcal{S}_{iml} = (\mathcal{P}_{iml}, \theta_1, \theta_2)$ and write $\approx_{iml}$ rather than $\approx_{\mathcal{S}_{iml}}$, the bisimulation equivalence induced by $\mathcal{S}_{iml}$. The standard SOS semantics of $\text{IML}$ [27] is given in Figure 5 involving the transition relations

$$
\rightarrow \subseteq \mathcal{P}_{iml} \times \mathcal{A} \times \mathcal{P}_{iml} \quad \text{and} \quad \rightarrow \subseteq \mathcal{P}_{iml} \times \mathbb{R}_{\geq 0} \times \mathcal{P}_{iml}
$$

Below we will use functions $T$ and $R$ based on $\rightarrow$ and $\rightarrow$-, cf. [30]. We have $T: \mathcal{P}_{iml} \times \mathcal{A} \times 2^{\mathcal{P}_{iml}} \rightarrow \mathbb{B}$ given by $T(P, a, C) = \text{true}$ if the set $\{ P' \in C : P \overset{a}{\rightarrow} P' \}$ is non-empty, for all $P \in \mathcal{P}_{iml}$, $a \in \mathcal{A}$ and any subset $C \subseteq \mathcal{P}_{iml}$. For $R: \mathcal{P}_{iml} \times \mathcal{P}_{iml} \rightarrow \mathbb{R}_{\geq 0}$ we put
Figure 5: Standard Transition Deduction System for IML.

\[ R(P, P') = \sum \{ \lambda \mid P \xrightarrow{\lambda} P' \} \]. Here, as common for probabilistic and stochastic process algebras, the comprehension is over the multiset of transitions leading from \( P \) to \( P' \) with label \( \lambda \). Alternatively, one could define an explicit cut-function, \( \text{cut} : \mathcal{P}_{\text{iml}} \times \mathbb{R}_{\geq 0} \times \mathcal{P}_{\text{iml}} \to \mathbb{R}_{\geq 0} \) returning the number of multiplicities of a transition \( P \xrightarrow{\lambda} P' \), or other means of decorations. We extend \( R \) to \( \mathcal{P}_{\text{iml}} \times 2^{\mathcal{P}_{\text{iml}}} \) by \( R(P, C) = \sum_{P' \in C} \sum \{ \lambda \mid P \xrightarrow{\lambda} P' \} \), for \( P \in \mathcal{P}_{\text{iml}} \), \( C \subseteq \mathcal{P}_{\text{iml}} \).

For IML we have the following notion of strong bisimulation \([27, 30]\) that we will compare with the notion of bisimulation associated with the FuTS \( S_{\text{iml}} \).

**Definition 7.5.** An equivalence relation \( R \subseteq \mathcal{P}_{\text{iml}} \times \mathcal{P}_{\text{iml}} \) is called a strong bisimulation for IML if, for all \( P_1, P_2 \in \mathcal{P}_{\text{iml}} \) such that \( R(P_1, P_2) \), it holds that

- for all \( a \in A \) and \( Q \in \mathcal{P}_{\text{iml}} \): \( \mathcal{T}(P_1, a, [Q]_{R}) \iff \mathcal{T}(P_2, a, [Q]_{R}) \)
- for all \( Q \in \mathcal{P}_{\text{iml}} \): \( R(P_1, [Q]_{R}) = R(P_2, [Q]_{R}) \).

Two processes \( P_1, P_2 \in \mathcal{P}_{\text{iml}} \) are called strongly bisimilar if \( R(P_1, P_2) \) for a strong bisimulation \( R \) for IML, notation \( P_1 \sim_{\text{iml}} P_2 \).

To establish the correspondence of FuTS bisimilarity \( \simeq_{\text{iml}} \) for \( S_{\text{iml}} \) as given by Definition 7.2 and strong bisimilarity \( \sim_{\text{iml}} \) for IML as given by Definition 7.5, we need to connect the state-to-function relation \( \rightarrow_1 \) and the transition relation \( \rightarrow \) as well as the state-to-function relation \( \rightarrow_2 \) and the transition relation \( \rightarrow \).

**Lemma 7.6.**

(a) Let \( P \in \mathcal{P}_{\text{iml}} \) and \( a \in A \). If \( P \xrightarrow{a} \mathcal{P} \) then \( P \xrightarrow{a} P' \iff \mathcal{P}(P') = \text{true} \).

(b) Let \( P \in \mathcal{P}_{\text{iml}} \). If \( P \xrightarrow{\delta} \mathcal{P} \) then \( \sum \{ \lambda \mid P \xrightarrow{\lambda} P' \} = \mathcal{P}(P') \).
Proof.

(a) Guarded induction. Let \( a \in \mathcal{A} \). We treat two typical cases, viz. \( \lambda.P \) and \( P_1 \parallel_A P_2 \) for \( a \notin \mathcal{A} \).

Case \( \lambda.P \). Suppose \( \lambda.P \xrightarrow{a} P' \). Then we have \( \mathcal{P} = [\|_{\mathcal{B}} \). We have \( \lambda.P \xrightarrow{a} P' \) for no \( P' \in \mathcal{P} \text{ml} \), as no transition is provided in \( \rightarrow \), and we have \( \mathcal{P}(P') = \text{false} \) by definition of \( [\|_{\mathcal{B}} \), for all \( P' \in \mathcal{P} \text{ml} \).

Case \( P_1 \parallel_A P_2 \), \( a \notin \mathcal{A} \). Suppose \( P_1 \xrightarrow{a} P_1' \parallel P_2 \), \( P_2 \xrightarrow{a} P_2' \) and \( P_1 \parallel_A P_2 \xrightarrow{a} P_2' \). Then it holds that \( \mathcal{P} = (\mathcal{P}_1 \parallel_A \mathcal{D}_{P_2}) + (\mathcal{D}_{P_1} \parallel_A \mathcal{P}_2) \). Recall, for \( Q \in \mathcal{P} \text{ml} \), by definition of \( \mathcal{D}_Q \in \text{FS}(\mathcal{P} \text{ml}, \mathcal{B}) \), \( \mathcal{D}_Q(\mathcal{P}') = \text{true} \) iff \( \mathcal{P}' = Q \), for \( Q' \in \mathcal{P} \text{ml} \). We have

\[
P_1 \parallel_A P_2 \xrightarrow{a} P'
\]

\[
\iff \left( P_1 \xrightarrow{a} P_1' \land P' = P_1' \parallel_A P_2 \right) \lor \left( P_2 \xrightarrow{a} P_2' \land P' = P_1 \parallel_A P_2' \right)
\]

by analysis of \( \rightarrow \)

\[
\iff \left( \mathcal{P}_1(P_1') = \text{true} \land P' = P_1' \parallel_A P_2 \right) \lor \left( \mathcal{P}_2(P_2') = \text{true} \land P' = P_1 \parallel_A P_2' \right)
\]

by the induction hypothesis

\[
\iff \left( \mathcal{P}_1(P_1') \cdot \mathcal{D}_{P_2}(P_2) = \text{true} \land P' = P_1' \parallel_A P_2 \right) \lor \left( \mathcal{D}_{P_1}(P_1) \cdot \mathcal{P}_2(P_2') = \text{true} \land P' = P_1 \parallel_A P_2' \right)
\]

by definition of \( \mathcal{D}_{P_1} \) and \( \mathcal{D}_{P_2} \)

\[
\iff \left( \mathcal{P}_1 \parallel_A \mathcal{D}_{P_2}(P_1') \parallel_A P_2 = \text{true} \land P' = P_1' \parallel_A P_2 \right) \lor \left( \mathcal{D}_{P_1} \parallel_A \mathcal{P}_2(P_2') = \text{true} \land P' = P_1 \parallel_A P_2' \right)
\]

by definition of \( \parallel_A \)

\[
\iff \left( \mathcal{P}_1 \parallel_A \mathcal{D}_{P_2}(P_1') \parallel_A P_2 = \text{true} \lor \left( \mathcal{D}_{P_1} \parallel_A \mathcal{P}_2(P_2') \right) = \text{true} \right)
\]

by definition of \( \parallel_A, \mathcal{D}_{P_1} \) and \( \mathcal{D}_{P_2} \)

\[
\iff \left( \mathcal{P}_1 \parallel_A \mathcal{D}_{P_2}(P_1') \right) = \text{true} \lor \left( \mathcal{D}_{P_1} \parallel_A \mathcal{P}_2(P_2') \right) = \text{true}
\]

by definition of \( + \) on \( \text{FS}(\mathcal{P} \text{ml}, \mathcal{B}) \)

\[
\iff \mathcal{P}(P') = \text{true}
\]

The other cases are standard, or similar and easier.

(b) Guarded induction. We treat the cases for \( \mu.P \) and \( P_1 \parallel_A P_2 \).

Case \( \mu.P \). Assume \( \mu.P \xrightarrow{\delta} \mathcal{P} \), then \( \mathcal{P} = [P \mapsto \mu] \). Moreover, it holds that \( \mu.P \) admits a single \( \rightarrow \)-transition, viz. \( \mu.P \xrightarrow{\mu} P \).

Thus we have \( \sum \{ \| \lambda \mid \mu.P \xrightarrow{\lambda} P' \} = \mu = \{P \mapsto \mu\}(P) = \mathcal{P}(P) \).

Case \( P_1 \parallel_A P_2 \). Assume \( P_1 \xrightarrow{\delta} \mathcal{P}_1, P_2 \xrightarrow{\delta} \mathcal{P}_2 \) and \( P_1 \parallel_A P_2 \xrightarrow{\delta} \mathcal{P} \). It holds that \( \mathcal{P} = (\mathcal{P}_1 \parallel_A \mathcal{D}_{P_2}) + (\mathcal{D}_{P_1} \parallel_A \mathcal{P}_2) \). We calculate

\[
\sum \{ \| \lambda \mid P_1 \parallel_A P_2 \xrightarrow{\lambda} P' \} = \sum \{ \| \lambda \mid P_1 \xrightarrow{\lambda} P_1', P' = P_1' \parallel_A P_2 \} + \sum \{ \| \lambda \mid P_2 \xrightarrow{\lambda} P_2', P' = P_1 \parallel_A P_2' \}
\]

by analysis of \( \rightarrow \)

\[
= (\text{if } P' = P_1' \parallel_A P_2 \text{ then } \sum \{ \| \lambda \mid P_1 \xrightarrow{\lambda} P_1' \} \text{ else 0 end }) +
\]

\[
(\text{if } P' = P_1 \parallel_A P_2' \text{ then } \sum \{ \| \lambda \mid P_2 \xrightarrow{\lambda} P_2' \} \text{ else 0 end })
\]
\[
= (\text{if } P' = P'_1 \parallel A P_2 \text{ then } \mathcal{P}_1(P'_1) \text{ else } 0 \text{ end}) + \\
(\text{if } P' = P_1 \parallel A P'_2 \text{ then } \mathcal{P}_2(P'_2) \text{ else } 0 \text{ end})
\]

by induction hypothesis for \( P_1 \) and \( P_2 \)

\[
= (\mathcal{P}_1 \parallel A D P_2)(P') + (D P_1 \parallel A P_2)(P')
\]

by definition of \( \parallel_A, D P_1, D P_2 \) and + on \( \mathcal{FS}(\mathcal{P}_{\text{iml}}, \mathbb{R}_{\geq 0}) \)

The remaining cases are left to the reader.

We are now in a position to relate \( \text{FuTS} \) bisimilarity and standard strong bisimilarity for \( \text{IML} \). In essence, Lemma \[7.6\] is all we need.

**Theorem 7.7.** For any two processes \( P_1, P_2 \in \mathcal{P}_{\text{iml}} \) it holds that \( P_1 \sim_{\text{iml}} P_2 \) iff \( P_1 \sim_{\text{iml}} P_2 \).

**Proof.** Let \( R \) be an equivalence relation on \( \mathcal{P}_{\text{iml}} \). Pick \( P \in \mathcal{P}_{\text{iml}}, a \in A \) and choose any \( Q \in \mathcal{P}_{\text{iml}} \). Suppose \( P \xrightarrow{a} Q \). Thus \( \theta_1(P)(a) = Q \). Then we have

\[
\mathcal{T}(P, a, [Q]_R) \iff \exists Q' \in [Q]_R: P \xrightarrow{a} Q' \text{ by definition of } \mathcal{T}
\]
\[
\iff \exists Q' \in [Q]_R: \mathcal{P}(Q') = \text{true} \text{ by Lemma } [7.6a]
\]
\[
\iff \sum_{Q' \in [Q]_R} \theta_1(P)(a)(Q') = \text{true} \text{ by definition of } \theta_1
\]

Note, summation in \( \mathbb{B} \) is disjunction. Likewise, on the Markovian side, we have

\[
\mathcal{R}(P, [Q]_R) = \sum_{Q' \in [Q]_R} \sum \{ | L | P \xrightarrow{t} Q' \} \text{ by definition of } \mathcal{R}
\]
\[
= \sum_{Q' \in [Q]_R} \mathcal{P}(Q') \text{ by Lemma } [7.6b]
\]
\[
= \sum_{Q' \in [Q]_R} \theta_2(P)(\delta)(Q) \text{ by definition of } \theta_2
\]

We conclude that a strong bisimulation for \( \text{IML} \) is also an \( S_{\text{iml}} \)-bisimulation for the \( p\text{FuTS} \) \( S_{\text{iml}} \), and vice versa. From this the theorem follows.

From the theorem we conclude that also for \( \text{IML} \) the concrete notion of strong bisimilarity \( \sim_{\text{iml}} \) is coalgebraically underpinned, as it coincides with the behavioral equivalence \( \sim_{\text{iml}} \) that comes with the corresponding \( \text{FuTS} \) \( S_{\text{iml}} \).

8. \text{FuTS} Semantics of TPC

In this section we consider a simple language of timed processes for which we provide a combined \( \text{FuTS} \). The language is a relevant fragment of the timed process algebra \( TPC \) presented in [3]. The model of time under consideration is discrete and deterministic. The relevant construct is the time-prefix \((n)P\), with \( n \in \mathbb{N}, n > 0 \), expressing that the process \( P \) is to be executed after \( n \) time steps. We will provide a \( \text{FuTS} \) semantics and compare the induced notion of bisimulation to the notion of timed bisimulation underlying the operational semantics reported in [3].

To the best of our knowledge, this is the first time a deterministically timed model is dealt with in the coalgebraic framework. As we will see, we resort to \( 2^\mathbb{N} \) as co-domain for the time continuations, instead of just \( \mathbb{N} \), as one may expect. In particular, we use the semiring \( 2^\mathbb{N} \) with set union as sum and intersection as multiplication. The reason of this choice is mainly technical and is connected to the proof of the bisimulation correspondence
Definition 8.1. The set $\mathcal{P}_{tpc}$ of TPC processes is given by the grammar below:

\[
\mathcal{P} ::= \text{nil} \mid a.P \mid (n).P \mid P + P \mid P \parallel A P \mid X
\]

where $a$ ranges over the set of actions $A$, $n$ over $\mathbb{N}$ with $n > 0$, $A$ over the set of finite subsets of $A$, and $X$ over the set of constants $\mathcal{X}$.

We assume the same notation and guardedness requirements for constant definition and usage as for PEPA or IML.

Definition 8.2. The formal semantics of $\mathcal{P}_{tpc}$ is given by the FuTS $\mathcal{S}_{tpc} = (\mathcal{P}_{tpc}, \rightarrow_1, \rightarrow_2)$, a combined FuTS over the label sets $A$ and $\Theta$ with $\Theta = \{\sqrt{\phantom{\cdot}}\}$ and the semirings $\mathbb{B}$ and $\mathbb{2}^\mathbb{N}$ with transition relations $\rightarrow_1 \subseteq \mathcal{P}_{tpc} \times A \times \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{B})$ and $\rightarrow_2 \subseteq \mathcal{P}_{tpc} \times \Theta \times \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{2}^\mathbb{N})$ defined as the least relations satisfying the rules of Figure 6.

Also $\mathcal{S}_{tpc}$ is a combined FuTS, having the two state-to-function relations $\rightarrow_1$ and $\rightarrow_2$. Actions $a \in A$ decorate $\rightarrow_1$, the special symbol $\sqrt{\phantom{\cdot}}$ decorates $\rightarrow_2$ (with a similar role as $\delta$ for IML). As for $\rightarrow_2$ the label is always the same, we occasionally suppress it. Note rule (APF3) and rule (TPF1) involve the null-functions of $\mathbb{2}^\mathbb{N}$ and of $\mathbb{B}$, respectively, to express that a process $a.P$ does not trigger a delay and a process $(n).P$ does not execute an action. In Figure 6 and in the rest of this section we use $\mathcal{P}, \mathcal{Q} \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{B})$ as typical action continuations, and $\mathcal{P}, \mathcal{Q} \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{2}^\mathbb{N})$ as typical time continuations.
The second time prefix rule (TPF2) combines a possible evolution over time of the process \(P\) into its continuation \(P\) with the elapse of the prefix. Note, the continuation in the conclusion of rule (TPF2) is a sum of three parts, viz. \([n; P], [P \mapsto \{n\}]\), and \((n + P)\). The auxiliary mappings \([n; P]\) and \((n + P)\), for timed continuations, are given by

\[
[n; P](Q) = \begin{cases} 
\{n\} & \text{if } Q = (n - m).P, \, 0 < m < n \\
\emptyset & \text{otherwise}
\end{cases}
\]

\((n + P)(Q) = \{n + m \mid m \in \mathcal{P}(Q)\}\)

It is easy to see that, for \(n \in \mathbb{N}, Q \in \mathcal{P}_{tpc}\), and \(P \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathcal{P}^N)\), \([n; Q] = [(n - i).Q \mapsto \{i\}]_{i=1}^{n-1}\), and if \(\mathcal{P}(Q) = \emptyset\), then also \((n + P)(Q) = \emptyset\). Time progress taking fewer steps than \(n\) is covered by the continuation \([n; P]\). For \(m\) strictly between 0 and \(n\), after \(m\) time steps there remains \((n - m).P\) to be executed. After exactly \(n\) time steps, \(P\) is to be executed, i.e. the component \([P \mapsto \{n\}]\) is used. After more than \(n\) time steps, say \(n + m\) time steps, process \(Q\) is to be executed if \(m \in \mathcal{P}(Q)\). Thus, if no such \(m\) exist, i.e. if \(\mathcal{P}(Q) = \emptyset\), this yields an empty set too.

The rules for the choice and parallel construct of TPC make use of corresponding operations on \(\mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{B})\) and \(\mathcal{FS}(\mathcal{P}_{tpc}, \mathcal{P}^N)\). For \(\mathcal{P}, \mathcal{Q} \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{B})\), the functions \(\mathcal{P} + \mathcal{Q}\) and \(\mathcal{P} \parallel_A \mathcal{Q}\) are as before. For \(\mathcal{FS}(\mathcal{P}_{tpc}, \mathcal{P}^N)\) the following operators are used:

\[
(\mathcal{P} + \mathcal{Q})(R) = \begin{cases} 
\mathcal{P}(P) \cap \mathcal{Q}(Q) & \text{if } R = P + Q \text{ for } P, Q \in \mathcal{P}_{tpc} \\
\emptyset & \text{otherwise}
\end{cases}
\]

and, likewise

\[
(\mathcal{P} \parallel_A \mathcal{Q})(R) = \begin{cases} 
\mathcal{P}(P) \cap \mathcal{Q}(Q) & \text{if } R = P \parallel_A Q, \text{for } P, Q \in \mathcal{P}_{tpc} \\
\emptyset & \text{otherwise}
\end{cases}
\]

We have that for \(P \in \mathcal{P}_{tpc}\) there exists a unique \(\mathcal{P} \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathcal{P}^N)\) such that \(P \rightarrow_2 \mathcal{P}\). Moreover, given the rules for \(\mathcal{S}_{tpc}\) and the definition of the operators above, it can verified that, for \(P, Q \in \mathcal{P}_{tpc}\) and \(\mathcal{P} \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathcal{P}^N)\) such that \(P \rightarrow_2 \mathcal{P}\) it holds that \(\mathcal{P}(Q)\) is either a singleton or the empty set. See Lemma 8.3 below.

In order to prove the lemma we introduce an auxiliary function \(md : \mathcal{P}_{tpc} \rightarrow \mathbb{N}\), establishing the so-called maximum delay of a process, given by

\[
\begin{align*}
md(\text{nil}) &= 0 \\
md(a.P) &= 0 \\
md((n).P) &= n + md(P) \\
md(P_1 + P_2) &= \min\{md(P_1), md(P_2)\} \\
md(P_1 \parallel_A P_2) &= \min\{md(P_1), md(P_2)\} \\
md((X) = md(P) & \text{ if } X := P
\end{align*}
\]

By guarded induction, one straightforwardly verifies the property that \(md(Q') < md(Q)\) for \(Q, Q' \in \mathcal{P}_{tpc}\) and \(Q \in \mathcal{FS}(\mathcal{P}_{tpc}, \mathbb{N})\) such that \(Q \rightarrow_2 Q\) and \(Q(Q') \neq \emptyset\). From this observation is follows that \([n; P], [P \mapsto \{n\}]\) and \((n + P)\) have disjoint supports: We have that (i) if \([n; P](P') \neq \emptyset\) then \(P' = (n - m).P\) for \(0 < m < n\), hence \(md(P') = (m - n) + md(P) > md(P)\); (ii) if \([P \mapsto \{n\}](P') \neq \emptyset\) then \(P' = P\), hence \(md(P') = md(P)\); (iii) if \((n + P)(P') \neq \emptyset\) then \(P'(P') \neq \emptyset\) hence, using the property above, \(md(P') < md(P)\).

**Lemma 8.3.**

(a) The FuTS \(\mathcal{S}_{tpc}\) is total and deterministic.

(b) If \(P \rightarrow_2 \mathcal{P}\) then either \(\mathcal{P}(Q) = \{n\}\) for some \(n > 0\) or \(\mathcal{P}(Q) = \emptyset\).

**Proof.** Part (a) goes by guarded induction on \(P\), both for \(\rightarrow_1\) and \(\rightarrow_2\). Part (b) follows by guarded induction. For the time prefix \((n).P\) we use that \([n; P], [P \mapsto \{n\}]\) and \((n + P)\)
have disjoint supports, as noted above. For the constructs $P + Q$ and $P \parallel A \cdot Q$ we observe
that the operations $[+]$ and $[\parallel_A]$ preserve the property mentioned, as the intersection of
two singletons holding a positive number is either a singleton with a positive number or the
elementary set.

Below we have $S_{tpc} = (\mathcal{P}_{tpc}, \theta_1, \theta_2)$ and use $\simeq_{tpc}$ to denote the bisimulation equivalence
induced by $S_{tpc}$.

The standard SOS semantics of the TPC fragment of interest is given in Figure 7 involving
the transition relations

$$\rightarrow \subseteq \mathcal{P}_{tpc} \times A \times \mathcal{P}_{tpc} \quad \text{and} \quad \sim \subseteq \mathcal{P}_{tpc} \times \mathbb{N}_{>0} \times \mathcal{P}_{tpc}$$

Note that for timed transitions $P \trans{a} P'$ it is required that $n > 0$. Therefore, regarding
rule (DEC), a process $(n).P$ for example with a timed prefix will not yield a zero-time step
$(n).P \trans{(\emptyset)} (n).P$ for which time does not progress. The case for $(n).P$ where $n$ time step
elapse, is covered by rule (PRE).

The definition of timed bisimilarity for TPC we give below is a bit more concise than
the one originally introduced in [3], but the two notions can be easily proven to coincide. We
will compare timed bisimilarity with the notion of bisimulation associated with the combined
FuTS $S_{tpc}$.

**Definition 8.4.** An equivalence relation $R \subseteq \mathcal{P}_{tpc} \times \mathcal{P}_{tpc}$ is a timed bisimulation for TPC if,
for all $P_1, P_2 \in \mathcal{P}_{tpc}$ such that $R(P_1, P_2)$, it holds that for all $a \in A$ and $n \in \mathbb{N}$

- whenever $P_1 \trans{a} Q_1$, then $P_2 \trans{a} Q_2$ for some $Q_2 \in \mathcal{P}_{tpc}$ with $R(Q_1, Q_2)$;
- whenever $P_1 \trans{\emptyset} Q_1$, then $P_2 \trans{\emptyset} Q_2$ for some $Q_2 \in \mathcal{P}_{tpc}$ with $R(Q_1, Q_2)$.

Two processes $P_1, P_2 \in \mathcal{P}_{tpc}$ are called timed bisimilar, notation $P_1 \simeq_{tpc} P_2$ if $R(P_1, P_2)$ for
some timed bisimulation for $\mathcal{P}_{tpc}$.

To establish the correspondence of FuTS bisimilarity $\simeq_{tpc}$ for $S_{tpc}$ of Definition 8.2 and
timed bisimilarity $\sim_{tpc}$ for TPC of Definition 8.4, we need to connect the state-to-function
relation $\rightarrow_1$ and the transition relation $\rightarrow$ as well as the state-to-function relation $\rightarrow_2$ and
the transition relation $\rightarrow$. The connection is established by Lemma 8.6. First we state an
auxiliary result, which is commonly referred to as time-determinism (cf. [4]) and which can be shown straightforwardly by guarded induction.

**Lemma 8.5.** If \( P \overset{a}{\rightarrow} P' \) and \( P \not\overset{a}{\rightarrow} P'' \), for \( P, P', P'' \in \mathcal{P}_{tpc} \) and \( n > 0 \), then \( P' = P'' \).

We use time-determinism of \( TPC \) in the proof of the following lemma.

**Lemma 8.6.**

(a) Let \( P \in \mathcal{P}_{tpc} \) and \( a \in A \). If \( P \overset{a}{\rightarrow} \mathcal{P} \) then \( P \overset{a}{\netarrow} P' \iff \mathcal{P}(P') = \text{true} \).

(b) Let \( P \in \mathcal{P}_{tpc} \). If \( P \overset{a}{\rightarrow} \mathcal{P} \) then \( P \overset{a}{\rightarrow} P' \iff \mathcal{P}(P') = \{n\} \).

**Proof.** Part (a) is similar to the corresponding part of Lemma 7.6. Part (b) can be shown by guarded induction for which we exhibit two cases (the others being similar or straightforward).

For readability, we suppress the label \( \sqrt{\cdot} \) of \( \rightarrow \).

Case \((m).P\). Suppose \((m).P \rightarrow P \) and \( P \rightarrow P' \). Then, by (TPF2), we have \( \mathcal{P}(P') = \{\ell\} \), for \( 0 < \ell < m \), iff \( P' = (m - \ell).P \), \( \mathcal{P}(P') = \{\ell\} \) iff \( P' = P \), and \( \mathcal{P}(P') = \{\ell\} \), for \( \ell > m \) iff \( P'(P') = \{\ell - m\} \). Now, if \((m).P \not\rightarrow P' \) for \( 0 < n < m \), then \( P' = (m - n).P \), because of rules (PRE) and (DEC) and Lemma 8.5. Therefore, \( \mathcal{P}(P') = \mathcal{P}((m - n).P) = \{n\} \). If \((m).P \not\rightarrow P' \) with \( n = m \), then \( P' = P \), as (PRE) applies (and with an appeal to Lemma 8.5).

Therefore, \( \mathcal{P}(P') = \mathcal{P}(P) = \{m\} = \{n\} \). Finally, if \((m).P \not\rightarrow P' \) for \( n > m \), then we have \( P \overset{n}{\rightarrow} P' \), in view of rule (SUM) and because of time-determinism. By induction hypothesis, we obtain \( \mathcal{P}'(P') = \{n - m\} \) and therefore \( \mathcal{P}(P') = (m + n')(P') = \{m + n \mid n \in \mathcal{P}'(P')\} = \{m + n - m\} = \{n\} \). Reversely, by rules (PRE) and (DEC) we have \((m).P \not\rightarrow (m - \ell).P \), for \( 0 < \ell < m \) and \((m).P \not\rightarrow P \). Moreover, if \( \mathcal{P}(P') = \{\ell\} \), for \( \ell > m \), then \( \mathcal{P}'(P') = \{\ell - m\} \). By induction hypothesis, \( P \overset{\ell}{\rightarrow} P' \). Hence, \((m).P \overset{m + \ell}{\rightarrow} P' \), i.e. \((m).P \not\rightarrow P' \), by (SUM).

Case \( P_1 + P_2 \). Suppose \( P_1 + P_2 \rightarrow P \). Then \( \mathcal{P} = \mathcal{P}_1 \parallel \mathcal{P}_2 \) for \( \mathcal{P}_1, \mathcal{P}_2 \in \mathcal{F}_s(\mathcal{P}_{tpc}, 2^N) \) such that \( P_1 \rightarrow P \) and \( P_2 \rightarrow P \). If \( P_1 + P_2 \not\rightarrow P' \), then exist \( P_1', P_2' \in \mathcal{P}_{tpc} \) such that \( P_1 \not\rightarrow P_1', P_2 \not\rightarrow P_2', P_1' + P_2' \), because (ALT) is the only rule applicable. By induction hypothesis, \( \mathcal{P}_1(P_1') = \{n\} \) and \( \mathcal{P}_2(P_2') = \{n\} \). Hence \( \mathcal{P}(P') = (\mathcal{P}_1 \parallel \mathcal{P}_2)(P_1' + P_2') = \{n\} \).

In the other direction, if \( \mathcal{P}(P') = \{n\} \), then \( P' = P_1' + P_2' \) for processes \( P_1', P_2' \in \mathcal{P}_{tpc} \) such that \( \mathcal{P}_1(P_1') = \{n\} \) and \( \mathcal{P}_2(P_2') = \{n\} \). By induction hypothesis, \( P_1 \not\rightarrow P_1' \) and \( P_2 \not\rightarrow P_2' \), from which it follows that \( P_1 + P_2 \not\rightarrow P_1' + P_2' \), i.e. \( P_1 + P_2 \not\rightarrow P' \), by (SUM).

With Lemma 8.6 in place we are ready to show the correspondence of FuTS bisimilarity and timed bisimilarity for \( TPC \).

**Theorem 8.7.** For any two processes \( P_1, P_2 \in \mathcal{P}_{tpc} \) it holds that \( P_1 \simeq_{tpc} P_2 \) iff \( P_1 \simeq_{tpc} P_2 \).

**Proof.** Suppose \( P_1 \simeq_{tpc} P_2 \), for \( P_1, P_2 \in \mathcal{P}_{tpc} \). Let \( R \subseteq \mathcal{P}_{tpc} \times \mathcal{P}_{tpc} \) be a bisimulation with respect to \( \mathcal{S}_{tpc} \) such that \( R(P_1, P_2) \). We verify that \( R \) meets the two transfer conditions of Definition 8.4.

If \( P_1 \rightarrow Q_1 \), for some \( a \in A \) and \( Q_1 \in \mathcal{P}_{tpc} \), then \( \theta_1(P_1)(a)(Q_1) = \text{true} \) by Lemma 8.6.

From the definition of a FuTS bisimulation we obtain

\[
\sum_{Q' \in [Q]_R} \theta_1(P_1)(a)(Q') = \sum_{Q' \in [Q]_R} \theta_1(P_2)(a)(Q') \tag{8.1}
\]

for all \( Q \in \mathcal{P}_{tpc} \). As we have seen before, we argue that summation of \( B \) is disjunction, and since \( \theta_1(P_1)(a)(Q_1) = \text{true} \), there must exist \( Q_2 \in [Q_1]_R \) such that \( \theta_1(P_2)(Q_2) = \text{true} \).

Hence, \( R(Q_1, Q_2) \) and, by Lemma 8.6 \( P_2 \rightarrow P_2' \).
If $P_1 \sim_{tpc} Q_1$, for some $n > 0$, then, by Lemma 8.6, $\theta_2(P_1)(\sqrt{})(Q_1) = \{n\}$. From the definition of $FuTS$ bisimulation we obtain

$$\sum_{Q' \in [Q]_R} \theta_2(P_1)(\sqrt{})(Q') = \sum_{Q' \in [Q]_R} \theta_2(P_2)(\sqrt{})(Q')$$

(8.2)

for all $Q \in \mathcal{P}_{tpc}$. Note, summation of the semiring $2^\mathbb{N}$ is union of sets. So, by picking $Q = Q_1$ we have $n \in \sum_{Q' \in [Q]_R} \theta_2(P_2)(\sqrt{})(Q')$. Thus, for some $Q_2 \in \mathcal{P}_{tpc}$ with $R(Q_1, Q_2)$ it holds that $n \in \theta_2(P_2)(\sqrt{})(Q_2)$. It follows from Lemma 8.3b that $\theta_2(P_2)(\sqrt{})(Q_2) = \{n\}$, and thus, again by Lemma 8.6, $P_2 \sim_{tpc} Q_2$.

Now suppose $P_1 \sim_{tpe} P_2$, for $P_1, P_2 \in \mathcal{P}_{tpc}$. Let $R \subseteq \mathcal{P}_{tpe} \times \mathcal{P}_{tpe}$ be a timed bisimulation such that $R(P_1, P_2)$. We verify that, with respect to $P_1$ and $P_2$, $R$ meets the two summation conditions of Definition 6.2 for the case of $S_{tpe}$, i.e., equations (8.1) and (8.2), for all $Q \in \mathcal{P}_{tpe}$ and $a \in A$. We have

$$\sum_{Q' \in [Q]_R} \theta_1(P_1)(a)(Q')$$

$$\iff \exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land \theta_1(P_1)(a)(Q') = \text{true} \quad \text{by structure of $\mathbb{B}$}$$

$$\iff \exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land P_1 \xrightarrow{a} Q' \quad \text{by Lemma 8.6}$$

$$\iff \exists Q'' \in \mathcal{P}_{tpe} : R(Q'', Q) \land P_2 \xrightarrow{\sim_{tpe}} Q'' \quad \text{R($P_1, P_2$) and $R$ timed bisimulation}$$

$$\iff \exists Q'' \in \mathcal{P}_{tpe} : R(Q'', Q) \land \theta_1(P_2)(a)(Q'') = \text{true} \quad \text{by Lemma 8.6}$$

$$\iff \sum_{Q'' \in [Q]_R} \theta_1(P_2)(a)(Q'')$$

and also

$$n \in \sum_{Q' \in [Q]_R} \theta_2(P_1)(\sqrt{})(Q')$$

$$\iff \exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land n \in \theta_2(P_1)(\sqrt{})(Q') \quad \text{by structure of } 2^\mathbb{N}$$

$$\iff \exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land \theta_2(P_1)(\sqrt{})(Q') = \{n\} \quad \text{by Lemma 8.3}$$

$$\iff \exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land P_1 \xrightarrow{\sim_{tpe}} Q' \quad \text{by Lemma 8.6}$$

$$\iff \exists Q'' \in \mathcal{P}_{tpe} : R(Q'', Q) \land P_2 \xrightarrow{\sim_{tpe}} Q'' \quad \text{R($P_1, P_2$) and $R$ timed bisimulation}$$

$$\iff \exists Q'' \in \mathcal{P}_{tpe} : R(Q'', Q) \land \theta_2(P_2)(\sqrt{})(Q'') = \{n\} \quad \text{by Lemma 8.6}$$

$$\iff \exists Q'' \in \mathcal{P}_{tpe} : R(Q'', Q) \land n \in \theta_2(P_2)(\sqrt{})(Q'') \quad \text{by Lemma 8.3}$$

$$\iff \sum_{Q'' \in [Q]_R} \theta_2(P_2)(\sqrt{})(Q'')$$

and also

Thus, $R$ satisfies the conditions for a $FuTS$ bisimulation for $S_{tpe}$. 

We conclude that also in the setting of a $FuTS$ for discrete time involving the semiring $2^\mathbb{N}$, we have an example of a correspondence result of $FuTS$-bisimilarity and bisimilarity based on a standard SOS definition. It is worth pointing out that in the proof above, the equivalence of $n \in \sum_{Q' \in [Q]_R} \theta_2(P_1)(\sqrt{})(Q')$ and $\exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land n \in \theta_2(P_1)(\sqrt{})(Q')$, holds because we are working with a semiring of (finite) sets over $\mathbb{N}$ with summation to be interpreted as (finite) union. Was summation to be interpreted as sum over $\mathbb{N}$, as it would have been the case if we would have used the semiring $\mathbb{N}$, i.e. using $\mathcal{FS}(\mathcal{P}_{tpe}, \mathbb{N})$ instead of $\mathcal{FS}(\mathcal{P}_{tpe}, 2^\mathbb{N})$, then, from $n = \sum_{Q' \in [Q]_R} \theta_2(P_1)(\sqrt{})(Q')$ we would not have been able to conclude $\exists Q' \in \mathcal{P}_{tpe} : R(Q', Q) \land n = \theta_2(P_1)(\sqrt{})(Q')$, and vice-versa.
9. Nested FuTS

In this section we extend the applicability of the FuTS framework to more complex models, in particular those in which different aspects of behaviour are integrated in a non-orthogonal way—as it is the case for non-deterministic choice of probabilistic distributions over behaviour in probabilistic and Markov automata. We introduce the notion of a nested FuTS, namely a FuTS where the transition relation involves continuation functions that do not act on the set of states \( S \) directly, but instead on functions acting on \( S \) or, in the general case, on functions over the latter and so on. As mentioned in the introduction, here we restrict our investigation on nested FuTSs with two levels, namely nested FuTSs where the domain of the continuation functions is a set of functions the domain of which is the set \( S \) of states. In the following, we give the formal definition of a simple two-level nested FuTS, i.e. a nested FuTS involving two levels of continuations that has a single transition relation.

**Definition 9.1.** Let \( \mathcal{L} \) be a set of labels and \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \) be two semirings. A (simple) two-level nested FuTS \( S \), over \( \mathcal{L} \) and \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \) is a tuple \( S = (S, \rightarrow) \) with set of states \( S \) and transition relation \( \rightarrow \subseteq S \times \mathcal{L} \times \text{FS}(\text{FS}(S, \mathcal{R}_1), \mathcal{R}_2) \).

A two-level nested FuTS is called total and deterministic if, for all \( s \in S \) and \( \ell \in \mathcal{L} \), there exists exactly one \( \psi \in \text{FS}(\text{FS}(S, \mathcal{R}_1), \mathcal{R}_2) \) such that \( s \rightarrow \psi \). As before, for a total and deterministic nested FuTS we use the notation \((S, \theta)\) where the function \( \theta \) has type \( S \rightarrow \mathcal{L} \rightarrow \text{FS}(\text{FS}(S, \mathcal{R}_1), \mathcal{R}_2) \). Here, for \( s \in S \), \( \ell \in \mathcal{L} \), \( \varphi \in \text{FS}(S, \mathcal{R}_1) \), \( y \in \mathcal{R}_2 \), we have \( \theta(s)(\ell)(\varphi) = y \) iff \( \psi(\varphi) = y \) for the unique \( \psi \in \text{FS}(\text{FS}(S, \mathcal{R}_1), \mathcal{R}_2) \) such that \( s \rightarrow \psi \).

For a set of states \( S \) and a semiring \( \mathcal{R} \), an equivalence relation \( R \) on \( S \) induces an equivalence relation on \( \text{FS}(S, \mathcal{R}) \), referred to as the lifting of \( R \) to \( \text{FS}(S, \mathcal{R}) \) and also denoted as \( R \). The induced relation \( R \) is defined by

\[
R(\varphi_1, \varphi_2) \text{ iff } \sum_{t \in [\varphi_1]} \varphi_1(t') = \sum_{t \in [\varphi_2]} \varphi_2(t') \text{ for all } t \in S
\]

for \( \varphi_1, \varphi_2 \in \text{FS}(S, \mathcal{R}) \). It is easy to see that \( R \) on \( \text{FS}(S, \mathcal{R}) \) is indeed an equivalence relation. Therefore, the notion of a two-level bisimulation for a two-level nested FuTS given below is well-defined.

**Definition 9.2.** Let \( S = (S, \rightarrow) \) be a two-level nested FuTS over the label set \( \mathcal{L} \) and semirings \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \). An equivalence relation \( R \subseteq S \times S \) is a two-level bisimulation for \( S \) if and only if \( R(s_1, s_2) \) implies

\[
\sum_{\varphi' \in [\varphi]} \theta(s_1)(\ell)(\varphi') = \sum_{\varphi' \in [\varphi]} \theta(s_2)(\ell)(\varphi') \quad (9.1)
\]

for all \( \ell \in \mathcal{L} \) and \( \varphi \in \text{FS}(S, \mathcal{R}_1) \). Two elements \( s_1, s_2 \in S \) are called bisimilar for \( S \) if \( R(s_1, s_2) \) for some two-level bisimulation \( R \) for \( S \). Notation \( s_1 \sim_{S} s_2 \).

In Section[10] we will show that, in the setting of Markov Automata, the notion of a two-level bisimulation for a suitable two-level nested FuTS (having \( \mathcal{R}_1 = \mathbb{R}_{\geq 0} \) and \( \mathcal{R}_2 = \mathbb{B} \)) coincides with the notion of strong bisimulation for Markov Automata.

As is to be expected, a total and deterministic two-level FuTS can be considered as a coalgebra of a suitable functor on sets.

**Definition 9.3.** Let \( \mathcal{L} \) be a label set and \( R = (\mathcal{R}_1, \mathcal{R}_2) \) be an pair of semirings. The functor \( \mathcal{W}_R^{\mathcal{L}} : \text{Set} \rightarrow \text{Set} \) assigns to a set \( X \) the function space \( \text{FS}(\text{FS}(X, \mathcal{R}_1), \mathcal{R}_2)^{\mathcal{L}} \) of all
functions \( \psi : \mathcal{L} \to \mathcal{FS}(\mathcal{FS}(X, R_1), R_2) \) and assigns to a mapping \( f : X \to Y \) the mapping \( \mathcal{W}^L_R(f) : \mathcal{FS}(\mathcal{FS}(X, R_1), R_2)^L \to \mathcal{FS}(\mathcal{FS}(Y, R_1), R_2)^L \) where

\[
\mathcal{W}^L_R(f)(\Phi)(\ell)(\psi) = \sum_{\varphi \in \mathcal{FS}(\mathcal{FS}(X, R_1), R_2)^{-1}(\psi)} \Phi(\ell)(\varphi)
\]

for all \( \Phi : \mathcal{L} \to \mathcal{FS}(\mathcal{FS}(X, R_1), R_2), \ell \in \mathcal{L}, \psi \in \mathcal{FS}(Y, R_1) \), where we use the function \( \mathcal{FS}(f, R_1) : \mathcal{FS}(X, R_1) \to \mathcal{FS}(Y, R_1) \) with \( \mathcal{FS}(f, R_1)(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x) \) for \( \varphi \in \mathcal{FS}(X, R_1) \) and \( y \in Y \).

Note that in the definition above the sums exist since \( \Phi \) and \( \varphi \) have finite support.

For readability we use \( \mathcal{W} \) as shorthand for \( \mathcal{W}^L_R \), when the label set \( \mathcal{L} \) and the pair of semirings \( R \) are clear from the context. It is readily checked that each \( \mathcal{W} \) is a functor, in fact an accessible one being a composition of accessible functors. Thus, \( \mathcal{W} \) possesses a final coalggebra. The associated notion of behavioural equivalence is denoted by \( \approx_{\mathcal{W}} \). As before, we have for nested FuTS a correspondence result as well.

**Theorem 9.4.** Let \( S = (S, \theta) \) be a two-level nested FuTS over the label set \( \mathcal{L} \) and the semirings \( R_1 \) and \( R_2 \). Let the functor \( \mathcal{W} \) be as in Definition 9.3. Then \( s_1 \approx_{S} s_2 \iff s_1 \approx_{\mathcal{W}} s_2 \), for all \( s_1, s_2 \in S \).

**Proof.** Let \( s_1, s_2 \in S \). We first prove \( s_1 \approx_{S} s_2 \Rightarrow s_1 \approx_{\mathcal{W}} s_2 \). So, assume \( s_1 \approx_{S} s_2 \). Let \( R \subseteq S \times S \) be a two-level bisimulation with \( R(s_1, s_2) \). We turn the collection of equivalence classes \( S/R \) into a \( \mathcal{W} \)-coalgebra \( S_R = (S/R, \theta_R) \) by putting

\[
\theta_R(\ell)((\bar{s}))(\varphi) = \sum_{\varphi \in \mathcal{FS}(S/R, R_1)^{-1}(\varphi)} \theta(s)(\ell)(\varphi)
\]

for \( s \in S \), \( \ell \in \mathcal{L} \), and \( \varphi \in \mathcal{FS}(S/R, R_1) \) and \( \varepsilon : S \to S/R \) the canonical mapping. This is well-defined since \( R \) is a two-level bisimulation and \( \mathcal{FS}(S/R, R_1)^{-1}(\varphi) \) is an equivalence class of \( R \), for all \( \varphi \in \mathcal{FS}(S/R, R_1) \). For, if \( \varphi_1, \varphi_2 \in \mathcal{FS}(\varepsilon_R, R_1)^{-1}(\bar{\varphi}) \), \( t \in S \) then \( \mathcal{FS}(\varepsilon_R, R_1)(\varphi_1)([t]_R) = \mathcal{FS}(\varepsilon_R, R_1)(\varphi_2)([t]_R) \). Thus \( \sum_{t' \in [t]_R} \varphi_1(t') = \sum_{t' \in [t]_R} \varphi_2(t') \) for all \( t \in S \), hence \( \varepsilon_R(s_1, \varphi_1) = \varepsilon_R(s_2, \varphi_2) \). Therefore, \( \varepsilon_R : S \to S/R \) is a \( \mathcal{W} \)-homomorphism: for \( \ell \in \mathcal{L} \) and \( \varphi \in \mathcal{FS}(S/R, R_1) \), we have

\[
\mathcal{W}(\varepsilon_R)((\theta(s))(\ell))(\varphi) = \sum_{\varphi \in \mathcal{FS}(\varepsilon_R, R_1)^{-1}(\varphi)} \theta(s)(\ell)(\varphi) \quad \text{by definition of } \mathcal{W}
\]

\[
= \theta_R([s]_R)(\ell)(\varphi) \quad \text{by definition of } \theta_R
\]

\[
= \varepsilon_R([s]_R)(\ell)(\varphi) \quad \text{by definition of } \varepsilon_R
\]

Thus, \( \varepsilon_R \circ \theta = \theta_R \circ \varepsilon_R \) and \( \varepsilon_R : S \to S_R \) is a \( \mathcal{W} \)-homomorphism as claimed. Now, by uniqueness of a final morphism, we have \( [\ ]_S = [\ ]_W \circ \varepsilon_R \). In particular, with respect to \( S \), this implies that \( [s_1]_W = [s_2]_W \) since \( \varepsilon_R(s_1) = \varepsilon_R(s_2) \). Thus, \( s_1 \approx_{S} s_2 \) as was to be shown.

For the reverse, \( s_1 \approx_{\mathcal{W}} s_2 \Rightarrow s_1 \approx_{S} s_2 \), assume \( s_1 \approx_{\mathcal{W}} s_2 \), i.e. \( [s_1]_W = [s_2]_W \), for \( s_1, s_2 \in S \). Since the map \( [\ ]_S : (S, \theta) \to (\Omega, \omega) \) is a \( \mathcal{W} \)-homomorphism, the equivalence relation \( R_S \) given by \( R_S(s', s'') \iff [s']_W = [s'']_W \) is a two-level bisimulation: Suppose \( R_S(s', s'') \), i.e. \( s' \approx_{\mathcal{W}} s'' \), for some \( s', s'' \in S \). Pick \( \ell \in \mathcal{L}, t \in S \) and assume \( [t]_W = w \in \Omega \). For \( W \) we have \( \omega \circ [\ ]_W = \mathcal{W}([\ ]_W) \circ \theta \). Hence, for \( s \in S, \ell \in \mathcal{L}, \chi \in \mathcal{FS}(\Omega, R_1) \), it holds that

\[
\omega([s]_W)(\ell)(\chi) = \mathcal{W}([\ ]_W)((\theta(s))(\ell))(\chi) = \sum_{\varphi \in \mathcal{FS}(\mathcal{FS}(\Omega, R_1)^{-1}(\chi))} \theta(s)(\ell)(\varphi)
\]

Moreover, we have, for \( \varphi_1, \varphi_2 \in \mathcal{FS}(S, R_1) \), that

\[
R_S(\varphi_1, \varphi_2) \iff \mathcal{FS}(\mathcal{FS}(\Omega, R_1)(\varphi_1) = \mathcal{FS}(\mathcal{FS}(\Omega, R_1)(\varphi_2)
\]

(9.2)
since we observe that
\[ FS(\llbracket W \rrbracket, R_1)(\varphi_1) = FS(\llbracket W \rrbracket, R_1)(\varphi_2) \]
\[ \iff \forall w \in [S]_W : FS(\llbracket W \rrbracket, R_1)(\varphi_1)(w) = FS(\llbracket W \rrbracket, R_1)(\varphi_2)(w) \]
\[ \text{since both } FS(\llbracket W \rrbracket, R_1)(\varphi_1)(w), FS(\llbracket W \rrbracket, R_1)(\varphi_2)(w) = 0 \text{ if } \llbracket W \rrbracket^{-1}(w) = \emptyset \]
\[ \iff \forall t \in S : FS(\llbracket W \rrbracket, R_1)(\varphi_1)([t]_W) = FS(\llbracket W \rrbracket, R_1)(\varphi_2)([t]_W) \]
\[ \forall t \in S : \sum_{t' \in [t]_W^{-1}([t]_W)} \varphi_1(t') = \sum_{t' \in [t]_W^{-1}([t]_W)} \varphi_2(t') \]
\[ \text{by definition of } FS(\cdot, R_1) \]
\[ \forall t \in S : \sum_{t' \in [t]_R S} \varphi_1(t') = \sum_{t' \in [t]_R S} \varphi_2(t') \]
\[ since t' \in [t]_R S \iff [t']_W = [t]_W \]
\[ \iff R_S(\varphi_1, \varphi_2) \]
\[ \text{by definition of } R_S \text{ on } FS(S, R_1) \]

Therefore,
\[ \varphi' \in [\varphi]_R S \iff \varphi' \in FS(\llbracket W \rrbracket, R_1)^{-1}(\chi) \text{ for } \chi = FS(\llbracket W \rrbracket, R_1)(\varphi) \]

Now, let \( s', s'' \in S \) such that \( R_S(s', s'') \), and choose any \( \ell \in L \) and \( \varphi \in FS(S, R_1) \). Put \( \chi = FS(\llbracket W \rrbracket, R_1)(\varphi) \). Then we have
\[ \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') \]
\[ = \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') \text{ by Equation (9.3) and definition } \chi \]
\[ = \omega([s']_W)(\ell)(\chi) \text{ by Equation (9.2)} \]
\[ = \omega([s'']_W)(\ell)(\chi) \text{ by assumption } s' \approx W s'' \]
\[ = \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') \text{ by Equation (9.2)} \]
\[ = \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') \text{ by Equation (9.3) and definition } \chi \]

Thus, if \( R_S(s', s'') \) then \( \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') = \sum_{\varphi' \in [\varphi]_R S} \theta(s')(\ell)(\varphi') \) for all \( \ell \in L \) and \( \varphi \in FS(S, R_1) \). Therefore, \( R_S \) is a two-level bisimulation according to Definition 9.2. Since \( [s_1]_W = [s_2]_W \), it follows that \( R_S(s_1, s_2) \). Thus \( R_S \) is a two-level bisimulation relating \( s_1 \) and \( s_2 \). Conclusion, it holds that \( s_1 \simeq_S s_2 \).

Above we introduced the notion of a two-level nested \( \text{FuTS} \) and an associated notion of bisimulation. Also in the case of such nested \( \text{FuTS} \), \( \text{FuTS} \)-bisimulation and behavioral equivalence of the corresponding functor coincides. Combination of nested \( \text{FuTS} \), or combination of nested and simple \( \text{FuTS} \), over the same set of states, is a straightforward generalization along the lines of Section 6. We will not pursue unfolding of the details here. In the next section we will encounter an example of such a construction.

10. \text{FuTS} Semantics of a Language for Markov Automata

As a final application of the \( \text{FuTS} \) approach to modeling quantitative behavior we consider so-called Markov automata (MA). A Markov automaton, as proposed in \[22, 23, 53\], combines non-deterministic and probabilistic behavior, on the one hand, with stochastic time behaviour, on the other hand. Therefore, we need a combination of a nested and a simple \( \text{FuTS} \) to model the respective behaviour.
The definition of an MA here follows [53]. We first recall some definitions from [53] with \( \text{Distr}(S) \subseteq \mathcal{FS}(S, \mathbb{R}_{\geq 0}) \) denoting the class of (finitely supported) probability distributions over \( S \).

The superposition of non-deterministic and probabilistic behaviour is provided in Markov automata by means of a combination of a standard choice operator \( \lor \) together with the probabilistic extension of action prefix \( a.\{ P_1 : P_1 \square \cdots \square p_h : P_h \} \) for \( a \in \mathcal{A} \), \( h \geq 0 \), and \( p_1, \ldots, p_h \in (0, 1) \) such that \( p_1 + \cdots + p_h = 1 \). The syntactic construct \( \{ P_1 : P_1 \square \cdots \square p_h : P_h \} \) denotes the distribution \( \mu_{\{ P_1 : P_1 \square \cdots \square p_h : P_h \}} \) over processes defined by

\[
\mu_{\{ p_1 : P_1 \square \cdots \square p_h : P_h \}} = \sum_{i=1}^{h} [P_i \mapsto p_i]
\]

The intuitive meaning is then obvious: process \( a.\{ P_1 : P_1 \square \cdots \square p_h : P_h \} \) performs action \( a \) and then behaves as process \( P \) with probability \( \mu_{\{ p_1 : P_1 \square \cdots \square p_h : P_h \}}(P) \).

A process language for Markov Automata called MAPA (Markov Automata Process Algebra) has been proposed in [53, 54, 52]. MAPA includes a rich data system and is equipped with restrictions to facilitate state space generation of relatively small models. Below, we consider MAL as introduced in [18]. MAL constitutes a simplified fragment of MAPA which highlights how nested non-deterministic and probabilistic behaviour combined with Markovian behaviour can be modeled in the \( \text{FuTS} \) framework.

**Definition 10.1.** The set \( \mathcal{P}_{\text{mal}} \) of MA processes is given by the grammar

\[
P ::= \text{nil} \mid a.\{ P_1 : P_1 \square \cdots \square p_h : P_h \} \mid \lambda P \mid \boxed{P + P} \mid \boxed{P \parallel A} \mid X
\]

where \( a \) ranges over the set of actions \( \mathcal{A} \), \( p_i \) over the interval \((0, 1]\), \( \lambda \) over \( \mathbb{R}_{> 0} \), \( A \) over the set of finite subsets of \( \mathcal{A} \) and \( X \) over the set of constants \( \mathcal{X} \). For an probabilistic action-prefix \( a.\{ p_1 : P_1 \square \cdots \square p_h : P_h \} \) it is required that \( h \geq 0 \) and \( p_1 + \cdots + p_h = 1 \).

We assume the same notation, guardedness requirements and conventions for constant definitions as in Section 5 for PEPA, IML and TPC.

In the setting of \( \mathcal{P}_{\text{mal}} \) we use \( \mathcal{P}, \mathcal{Q} \) to range over \( \mathcal{FS}(\mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathcal{B}) \) and \( \mathcal{P}, \mathcal{Q} \) to range over \( \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \). We use \( \mu, \nu \) to range over \( \text{Distr}(\mathcal{P}_{\text{mal}}) \subseteq \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \). As before, we let \( P_1 + P_2 \) be the pointwise sum of \( P_1 \) and \( P_2 \). (Note, we are adding rates here.) We put \( D_P = [P \mapsto 1] \) in \( \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \) and define \( F_1 \parallel A F_2 : \mathcal{P}_{\text{mal}} \rightarrow \mathbb{R}_{\geq 0} \), for \( F_1, F_2 \in \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \) and \( A \subseteq \mathcal{A} \), by

\[
(F_1 \parallel A F_2)(R) = \begin{cases} F_1(R_1) \cdot F_2(R_2) & \text{if } R = R_1 \parallel A R_2 \text{ for some } R_1, R_2 \in \mathcal{P}_{\text{mal}} \\ 0 & \text{otherwise} \end{cases}
\]

Note \( F_1 \parallel A F_2 \in \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \). Moreover, if \( \mu_1, \mu_2 \in \text{Distr}(\mathcal{P}_{\text{mal}}) \) then \( \mu_1 \parallel A \mu_2 \in \text{Distr}(\mathcal{P}_{\text{mal}}) \) too, since \( \sum_{R \in \mathcal{P}_{\text{mal}}} (\mu_1 \parallel A \mu_2)(R) = \sum_{R_1, R_2 \in \mathcal{P}_{\text{mal}}} \mu_1(R_1) \cdot \mu_2(R_2) \) and the latter summation is equal to \( (\sum_{R_1 \in \mathcal{P}_{\text{mal}}} \mu_1(R_1)) \cdot (\sum_{R_2 \in \mathcal{P}_{\text{mal}}} \mu_2(R_2)) \) while \( \sum_{R_1 \in \mathcal{P}_{\text{mal}}} \mu_1(R_1) \) and \( \sum_{R_2 \in \mathcal{P}_{\text{mal}}} \mu_2(R_2) \) are both equal to 1. For \( F_1, F_2 \in \mathcal{FS}(\mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathcal{B}) \) and \( A \subseteq \mathcal{A} \), we also use pointwise conjunction and pointwise disjunction, and \( F_1 \parallel A F_2 \) is defined by

\[
\sum \mu_1, \mu_2 : F_1(\mu_1) = \text{true} \land F_2(\mu_2) = \text{true} \quad [\mu_1 \parallel A \mu_2 \mapsto \text{true}]
\]

Thus \( (F_1 \parallel A F_2)(\mu) = \text{true} \iff \mu = \mu_1 \parallel A \mu_2 \) for \( \mu_1 \) such that \( F_1(\mu_1) = \text{true} \) and \( \mu_2 \) such that \( F_2(\mu_2) = \text{true} \). We overload \( D_P \) for \( P \in \mathcal{P}_{\text{mal}} \); with respect to \( \mathcal{FS}(\mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathcal{B}) \) we have \( D_P = \{ [P \mapsto 1] \mapsto \text{true} \} \). Because of the contexts no confusion arises whether to interpret \( D_P \) with respect to \( \mathcal{FS}(\mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathcal{B}) \) or with respect to \( \mathcal{FS}(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}) \).
We use for $P \in \text{FS}$

**Lemma 10.3.** For all $P \in \text{FS}$, we have $\mu(P \to_1) = \mu(P \to_2) = 1$.

By guarded induction we obtain that the finitely supported functions involved in the definition of $\to_1$ and $\to_2$ are indeed probability distributions. Ultimately this relies on the restriction on the extended prefix, for the process $a.\{p_1 \cdot P_1 \ldots \cdot p_h \cdot P_h\}$ the finite sum $p_1 + \ldots + p_h$ must be equal to 1.

**Lemma 10.4.** The general FuTS $S_{\text{mal}}$ is total and deterministic.

Below we use $S_{\text{mal}} = (\text{FS}(\mathbb{R}_\geq), \mathbb{R}_\geq)$ where $\mathbb{R}_\geq$ is the set of non-negative real numbers.

With the operators defined above in place, and a combined treatment of actions and probabilities vs. stochastic delays, it is straightforward to capture the semantics of MAL with FuTS, cf. [13].

**Definition 10.2.** The formal semantics of $P_{\text{mal}}$ is given by $S_{\text{mal}} = (P_{\text{mal}}, \to_1, \to_2)$, a general FuTS over the label sets $\mathcal{A}$ and $\Delta = \{\delta\}$ and the semirings $\mathbb{R}_\geq$, $\mathbb{B}$ and $\mathbb{R}_\geq$ again with transition relations $\to_1$ and $\to_2$, where $\to_1 \subseteq P_{\text{mal}} \times \mathcal{A} \times \text{FS}(\text{FS}(P_{\text{mal}}, \mathbb{R}_\geq), \mathbb{B})$ and $\to_2 \subseteq P_{\text{mal}} \times \Delta \times \text{FS}(P_{\text{mal}}, \mathbb{R}_\geq)$, defined as the least relations satisfying the rules of Figure 8.

By guarded induction we obtain that the finitely supported functions involved in the definition of $\to_1$ are indeed probability distributions. Ultimately this relies on the restriction on the extended prefix, for the process $a.\{p_1 \cdot P_1 \ldots \cdot p_h \cdot P_h\}$ the finite sum $p_1 + \ldots + p_h$ must be equal to 1.

**Lemma 10.3.** For all $P \in P_{\text{mal}}, a \in \mathcal{A}$, functions $\mathcal{P} \in \text{FS}(\text{FS}(P_{\text{mal}}, \mathbb{R}_\geq), \mathbb{B})$ and $\mathcal{P} \in \text{FS}(P_{\text{mal}}, \mathbb{R}_\geq)$, if $P \overset{a}{\to} \mathcal{P}$ and $\mathcal{P}(\mathcal{P}) = \text{true}$, then $\mathcal{P} \in \text{Distr}(P_{\text{mal}})$.

It is not difficult either to verify that $S_{\text{mal}}$ is a total and deterministic combined FuTS, i.e. for $P \in P_{\text{mal}}$, $a \in \mathcal{A}$ we have $P \overset{a}{\to} \mathcal{P}$ for exactly one $\mathcal{P} \in \text{FS}(\text{FS}(P_{\text{mal}}, \mathbb{R}_\geq), \mathbb{B})$ and $P \overset{\delta}{\to} \mathcal{P}$ for exactly one $\mathcal{P} \in \text{FS}(P_{\text{mal}}, \mathbb{R}_\geq)$.

**Lemma 10.4.** The general FuTS $S_{\text{mal}}$ is total and deterministic.

Below we use $S_{\text{mal}} = (P_{\text{mal}}, \theta_1, \theta_2)$ with $\theta_1 : P_{\text{mal}} \to \text{FS}(\text{FS}(P_{\text{mal}}, \mathbb{R}), \mathbb{B})$ and $\theta_2 : P_{\text{mal}} \to \text{FS}(P_{\text{mal}}, \mathbb{R})$ induced by $\to_1$ and $\to_2$, respectively. We write $\approx_{\text{mal}}$ for the associated notion.
A standard LTS-based operational semantics of MAL is given by the SOS rules of Figure 9. The semantics is similar to the one reported in [53, 54]. Here, however, the technical overhead of decorations on transitions as used in the above mentioned papers is avoided at the expense of implicit multiplicities, in line with the treatment of PEPA and IML in Sections 5 and 7, respectively. Note, as MAL extends IML, there are separate rules for interactive transitions (ACT, CHO1–2, PAR1–3 and REC1) captured by the transition relation $\rightarrow$, and for Markovian transitions (DELAY, CHO3–4, PAR4–5, REC2) captured by the transition relation $\rightarrow\rightarrow$.

Definition 10.5. An equivalence relation $R \subseteq \mathcal{P}_{mal} \times \mathcal{P}_{mal}$ is an $\mathcal{S}_{mal}$-bisimulation if and only if $R$ is a nested bisimulation with respect to $\theta_1$ and a simple bisimulation with respect to $\theta_2$.

If we unfold the definitions for the two types of FuTS bisimulation we obtain that an equivalence relation $R \subseteq \mathcal{P}_{mal} \times \mathcal{P}_{mal}$ is an $\mathcal{S}_{mal}$-bisimulation, if for all $P_1, P_2 \in \mathcal{P}_{mal}$ such that $R(P_1, P_2)$, it holds that

- for all $a \in A$ and $\mu \in \text{Distr}(\mathcal{P}_{mal})$: $\sum_{\mu' \in [\mu]_R} \theta_1(P_1)(a)(\mu') = \sum_{\mu' \in [\mu]_R} \theta_1(P_2)(a)(\mu')$, and
- for all $Q \in \mathcal{P}_{mal}$: $\sum_{\mu \in Q} \theta_2(P_1)(\delta)(Q') = \sum_{\mu \in Q} \theta_2(P_2)(\delta)(Q')$ with $R$ on $\text{Distr}(\mathcal{P}_{mal})$ induced by $R$ on $\mathcal{P}_{mal}$. Recall that, for $\mu_1, \mu_2 \in \text{Distr}(\mathcal{P}_{mal})$, $R(\mu_1, \mu_2)$ if and only if $\sum_{Q' \in [\mu_2]_R} \mu_1(Q') = \sum_{Q' \in [\mu_1]_R} \mu_2(Q')$ for all $Q \in \mathcal{P}_{mal}$.

A standard LTS-based operational semantics of MAL is given by the SOS rules of Figure 9. The semantics is similar to the one reported in [53, 54]. Here, however, the technical overhead of decorations on transitions as used in the above mentioned papers is avoided at the expense of implicit multiplicities, in line with the treatment of PEPA and IML in Sections 5 and 7, respectively. Note, as MAL extends IML, there are separate rules for interactive transitions (ACT, CHO1–2, PAR1–3 and REC1) captured by the transition relation $\rightarrow$, and for Markovian transitions (DELAY, CHO3–4, PAR4–5, REC2) captured by the transition relation $\rightarrow\rightarrow$.

Definition 10.6. The semantics of the process language MAL is the tuple $(\mathcal{P}_{mal}, A, \rightarrow, \rightarrow\rightarrow)$ where the probabilistic transition relation $\rightarrow \subseteq \mathcal{P}_{mal} \times A \times \text{Distr}(\mathcal{P}_{mal})$ and the standard transition relation $\rightarrow\rightarrow \subseteq \mathcal{P}_{mal} \times \mathbb{R}_{>0} \times \mathcal{P}_{mal}$ are given by the SOS rules of Figure 9.
Similar to our treatment of $P_{\text{mal}}$ in Section 7, we introduce the functions $I$ and $M$ based on the transition relations $\to$ and $\rightarrow$ of Definition 10.6 for $P_{\text{mal}}$. Now, for the interactive part of $\text{MAL}$, we have $I: P_{\text{mal}} \times A \times 2^{\text{Distr}(P_{\text{mal}})} \rightarrow \mathbb{B}$ given by $I(P, a, \mathcal{C}) = \text{true}$ if the set $\{ \mu \in \mathcal{C} \mid P \xrightarrow{a} \mu \}$ is non-empty, for all $P \in P_{\text{mal}}$, $a \in A$ and any subset $\mathcal{C} \subseteq \text{Distr}(P_{\text{mal}})$. The Markovian part of $\text{MAL}$ is similar to that of $\text{IML}$. We define for $\text{MAL}$ the function $M: P_{\text{mal}} \times P_{\text{mal}} \rightarrow \mathbb{R}_{\geq 0}$ by $M(P, P') = \sum \{ \lambda \mid P \xrightarrow{\lambda} P' \}$. Because of the implicit multiplicities of the SOS of Definition 10.6, the comprehension is over the multiset of transitions leading from $P$ to $P'$ with label $\lambda$. We also extend $M$, now to $P_{\text{mal}} \times 2^{P_{\text{mal}}}$, by $M(P, C) = \sum_{P' \in C} \sum \{ \lambda \mid P \xrightarrow{\lambda} P' \}$, for $P \in P_{\text{mal}}$ and $C \subseteq P_{\text{mal}}$. With the adapted functions $I$ and $M$ in place, the notion of strong bisimulation for $\text{MAL}$ is defined as follows.

**Definition 10.7.** An equivalence relation $R \subseteq P_{\text{mal}} \times P_{\text{mal}}$ is called a strong bisimulation for $\text{MAL}$ if, for all $P_1, P_2 \in P_{\text{mal}}$ such that $R(P_1, P_2)$, it holds that

- for all $a \in A$ and $\mu \in \text{Distr}(P_{\text{mal}})$ : $I(P_1, a, [\mu]_R) \iff I(P_2, a, [\mu]_R)$
- for all $Q \in P_{\text{mal}}$: $M(P_1, [Q]_R) = M(P_2, [Q]_R)$

with the relation $R$ on $\text{Distr}(P_{\text{mal}})$ induced by the relation $R$ on $P_{\text{mal}}$. Two processes $P_1, P_2 \in P_{\text{mal}}$ are called strongly bisimilar if it holds that $R(P_1, P_2)$ for a strong bisimulation $R$ for $\text{MAL}$, notation $P_1 \sim_{\text{ma}} P_2$.

Recall, again, that the relation $R \subseteq P_{\text{mal}} \times P_{\text{mal}}$ induces relation $R \subseteq \text{Distr}(P_{\text{mal}}) \times \text{Distr}(P_{\text{mal}})$ by $R(\mu_1, \mu_2)$ if and only if $\sum_{Q \in [Q]_R} \mu_1(Q) = \sum_{Q' \in [Q]_R} \mu_2(Q')$ for all $Q \in P_{\text{mal}}$.

In line with what we have seen in the previous sections, the crux for relating the notion of $\mathcal{S}_{\text{mal}}$-bisimulation and the notion of strong bisimulation of Definition 10.7 is the following result.

**Lemma 10.8.**

(a) Let $P \in P_{\text{mal}}$ and $a \in A$. If $P \xrightarrow{a} \mathcal{P}$ then $P \xrightarrow{a} \mu \iff \mathcal{P}(\mu) = \text{true}$.

(b) Let $P \in P_{\text{mal}}$. If $P \xrightarrow{\delta} \mathcal{P}$ then $\sum \{ \lambda \mid P \xrightarrow{\lambda} P' \} = \mathcal{P}(P')$.

**Proof.**

(a) Guarded induction. Let $a \in A$. We treat the cases $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \}$ and $P_1 \parallel_{A} P_2$ for $a \in A$.

Case $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \}$, $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \} \xrightarrow{a} [\mu_{\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \} \mapsto \text{true}]}$, while $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \} \xrightarrow{a} \mu_{\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \}}$ is the only transition for process $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \}$.

Case $P_1 \parallel_{A} P_2$, $a \in A$. Assume $P_1 \parallel_{A} P_2 \xrightarrow{a} \mathcal{P}$. Then $\mathcal{P} = \mathcal{P}_1 \parallel_{A} \mathcal{P}_2$ for $\mathcal{P}_1, \mathcal{P}_2 : \mathcal{FS}(P_{\text{mal}}, \mathbb{R}_{\geq 0}) \rightarrow \mathbb{B}$ such that $P_1 \xrightarrow{a} \mathcal{P}_1, P_2 \xrightarrow{a} \mathcal{P}_2$. Suppose $P_1 \parallel_{A} P_2 \xrightarrow{a} \mu$. Then there exist $\mu_1, \mu_2 \in \text{Distr}(P_{\text{mal}})$ such that $P_1 \xrightarrow{a} \mu_1, P_2 \xrightarrow{a} \mu_2$ and $\mu = \mu_1 \parallel_{A} \mu_2$, since then rule (PAR3) of Figure 3 applies. By induction hypothesis, $\mathcal{P}_1(\mu_1) = \text{true}$ and $\mathcal{P}_2(\mu_2) = \text{true}$. Hence $\mathcal{P}(\mu) = (\mathcal{P}_1 \parallel_{A} \mathcal{P}_2)(\mu_1 \parallel_{A} \mu_2)$ is true by definition of $\parallel_{A}$ on $\mathcal{FS}(\mathcal{FS}(P_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathbb{B})$. Reversely, suppose $\mathcal{P}(\mu) = \text{true}$. Then $\mu = \mu_1 \parallel_{A} \mu_2$ for $\mu_1, \mu_2 \in \text{Distr}(P_{\text{mal}})$ such that $\mathcal{P}_1(\mu_1) = \text{true}$ and $\mathcal{P}_2(\mu_2) = \text{true}$. Then $\mu_1 \parallel_{A} \mu_2$. By induction hypothesis, $P_1 \xrightarrow{a} \mu_1$ and $P_2 \xrightarrow{a} \mu_2$. Hence $P_1 \parallel_{A} P_2 \xrightarrow{a} \mu_1 \parallel_{A} \mu_2$ by rule (PAR3), i.e. $P_1 \parallel_{A} P_2 \xrightarrow{a} \mu$.

The other cases are left to the reader.

(b) Guarded induction. Compared to the proof of Lemma 7.6 there is only one new case, viz. for processes of the form $a.\{ p_1.P_1 \sqcup \cdots \sqcup p_h.P_h \}$. This case is straightforward, since,
on the one hand, \( a.\{ p_1 \cdot P_1 \square \cdots \square p_h \cdot P_h \} \overset{\delta}{\rightarrow} [\mathbb{R}_{\geq 0}] \) by definition of \( \rightarrow_2 \) and, on the other hand, we have that \( a.\{ p_1 \cdot P_1 \square \cdots \square p_h \cdot P_h \} \overset{\lambda}{\rightarrow} P' \) for no \( P' \in \mathcal{P}_{\text{mal}} \) by definition of \( \rightarrow_\lambda \).

The remaining cases are similar to the proof for the corresponding lemma for IML and left to the reader. \( \square \)

We are now in a position to relate the notions of FuTS bisimilarity \( \simeq_{\text{mal}} \) and standard strong bisimilarity \( \sim_{\text{ma}} \) for MAL.

**Theorem 10.9.** For any two processes \( P_1, P_2 \in \mathcal{P}_{\text{mal}} \) it holds that \( P_1 \simeq_{\text{mal}} P_2 \) iff \( P_1 \sim_{\text{ma}} P_2 \).

**Proof.** Let \( R \) be an equivalence relation on \( \mathcal{P}_{\text{mal}} \). Pick \( P \in \mathcal{P}_{\text{mal}} \), \( a \in A \) and choose any \( \mathcal{P} \in \mathcal{F}S(\mathcal{F}S(\mathcal{P}_{\text{mal}}, \mathbb{R}_{\geq 0}), \mathbb{B}) \). Suppose \( P \overset{a}{\rightarrow} \mathcal{P} \). Thus \( \theta_1(P)(a) = \mathcal{P} \). Then we have, for any \( \mu \in \text{Distr}(\mathcal{P}_{\text{mal}}) \),

\[
I(P, a, [\mu]_R) \iff \exists \mu' \in [\mu]_R: P \overset{a}{\rightarrow} \mu' \quad \text{by definition of } I
\]

\[
\iff \exists \mu' \in [\mu]_R: \mathcal{P}(\mu') = \text{true} \quad \text{by Lemma 10.8a}
\]

\[
\iff \sum_{\mu' \in [\mu]_R} \theta_1(P)(a)(\mu') \quad \text{by definition of } \theta_1
\]

Note, summation in \( \mathbb{B} \) is disjunction. Likewise, on the Markovian side, we have, for any \( Q \in \mathcal{P}_{\text{mal}} \),

\[
M(P, [Q]_R) = \sum_{Q' \in [Q]_R} \sum \{ \lambda \mid P \overset{\lambda}{\rightarrow} Q' \} \quad \text{by definition of } M
\]

\[
= \sum_{Q' \in [Q]_R} \mathcal{P}(Q') \quad \text{by Lemma 10.8},
\]

\[
= \sum_{Q' \in [Q]_R} \theta_2(P)(\delta)(Q) \quad \text{by definition of } \theta_2
\]

Comparing the equations following Definition 10.5 and the equations of Definition 10.7, we conclude that a strong bisimulation for MAL is also an \( S_{\text{mal}} \)-bisimulation for the FuTS \( S_{\text{mal}} \), and vice versa. From this the theorem follows. \( \square \)

As a corollary of the theorem we obtain that also for MAL the concrete notion of strong bisimilarity \( \sim_{\text{ma}} \) is coalgebraically underpinned, as it coincides, with the behavioral equivalence \( \simeq_{\text{mal}} \) that comes with the corresponding FuTS \( S_{\text{mal}} \).

11. Concluding remarks

Total and deterministic state-to-function labeled transition systems, FuTSs, are a convenient instrument to express the operational semantics of both qualitative and quantitative process languages. In this paper we have discussed the notion of bisimilarity that arises from a FuTS, possibly involving multiple transition relations, from a coalgebraic perspective. For FuTS models of prominent process languages based on prominent stochastic process algebras we related the induced notion of bisimulation to the standard equivalences, thus providing these equivalence with a coalgebraic underpinning. The main technical contributions of our paper include correspondence results, viz. Theorem 4.3, Theorem 6.4 and Theorem 9.4 that relate in the simple, combined and the new nested case, bisimilarity of a FuTS \( S \) to behavioural equivalence of the functor associated with \( S \). The result extends to general FuTS as well.

It is noted in [10], in the context of weighted automata, that in general the type of functors \( \mathcal{F}S(\cdot, \mathcal{R}) \) may not preserve weak pullbacks and, therefore, the notions of coalgebraic bisimilarity and of behavioural equivalence may not coincide. A counter example is provided, cf. [10] Section 2.2. Essential for the construction of the counter-example, in their setting,
is the fact that the sum of non-zero weights may add to weight 0. The same phenomenon prevents a general proof, along the lines of [56], for coalgebraic bisimilarity and FuTS bisimilarity to coincide. In the construction of a mediating morphism, going from FuTS bisimulation to coalgebraic bisimulation a denominator may be zero, hence a division undefined, in case the sum over an equivalence class cancels out. In the concrete case for [35], although no detailed proof is provided there, this will not happen with \( \mathbb{R}_{\geq 0} \) as underlying semiring. In [25, Theorem 5.13] for \( FS(\cdot, M) \), with \( M \) a monoid, a characterization is given for weak preservation of pullbacks: \( M \) should be positive and refinable, i.e. (i) \( m_1 + m_2 = 0 \) iff \( m_1, m_2 = 0 \), and (ii) if \( m_1 + m_2 = n_1 + n_2 \) there exist \( p_{ij} \) such that \( p_{i1} + p_{i2} = m_i \) and \( p_{1j} + p_{2j} = n_j \) for \( 1 \leq i, j \leq 2 \). The latter condition is also referred to as the row-column property for \( 2 \times 2 \) matrices over \( M \), a property going back to [43]. In [39] we propose to consider semirings which admit a (right) multiplicative inverse for non-zero elements, and satisfy the so-called zero-sum property, stating that for a sum \( x = x_1 + \cdots + x_n \) it holds that \( x = 0 \) iff \( x_i = 0 \) for all \( i = 1 \ldots n \). The proof follows the set-up of [56], hence is different from [26]: we see that zero-sum coincides with positivity, while the existence of multiplicative inverses guarantees refinability. Thus, for semirings involved enjoying these properties, pullbacks are weakly preserved by \( FS(\cdot, R) \). Therefore, coalgebraic bisimilarity and behavioural equivalence are the same. As a consequence, under conditions which are met by the SPCs discussed in the preceding, we have that concrete bisimulation, FuTS-bisimilarity, behavioural equivalence and coalgebraic bisimilarity coincide.

For typical stochastic process languages based on PEPA and IMC we have shown that the notion of strong equivalence and strong bisimilarity associated with these calculi, coincides with the notion of bisimilarity of the corresponding FuTS. Using these FuTS as a stepping stone, the correspondence results bridge between the concrete notion of bisimulation for PEPA and IML, and the associated coalgebraic notions of behavioural equivalence. Hence, from this perspective, the concrete notions are seen as the natural strong equivalence to consider. Obviously, classical strong bisimilarity [42, 44] and bisimilarity for FuTS over \( \mathbb{B} \) coincide (see [35] or [39] for details). Also, strong bisimulation of [31], an alternative to Hillston’s notion of strong equivalence covered here, involving apart from the usual transfer conditions the comparison of state information, viz. the apparent rates, can be treated with FuTS. Again the two notions of equivalence coincide. Finally, we gave an account of how languages based on discrete deterministic time, TPC, as well as those where stochastic time is integrated with discrete probability and with non-determinism, MAL, can be treated in the FuTS framework. A similar mediating role for FuTS applies to these calculi too: the concrete notion of bisimulation coincides with FuTS bisimulation, hence coincides with the corresponding behavioral equivalence.

As mentioned in Section 1, related work in the area of systematic approaches to frameworks for the semantics of SPC—and quantitative extensions of process calculi in general—includes the study of abstract quantitative GSOS, with its application to Weighted Transition Systems (WTS) [35, 34, 41]. Stochastic GSOS (SGSOS) and Weighted GSOS appear to be a special case of Miculan and Peressotti’s weight function GSOS. In [35, 41] a treatment is given for PEPA, in line with Section 5 of the present paper. The formats mentioned above arise from the abstract theory of SOS. A noteworthy result, shown in [35], is that stochastic bisimilarity of SPC defined using the SGSOS format is guaranteed to be a congruence. The result is generalized to WTS in [41]. We did not address the issue of congruences for FuTS in the present paper. Nevertheless, we note that Rated Transition Systems—the semantic model used in [35]—are very similar to RTS of Latella, Massink et
al. [15, 17, 16], which are the instantiation of simple \(FuTS\) on non-negative real numbers, and that \(WTS\) are very similar to simple \(FuTS\). Thus, it is to be expected that simple \(FuTS\) can be represented as \(WTS\) using the SGSOS, which would extend the congruence result to \(FuTS\). The issue of the relationship with \(WTS\) remains, though, for the richer class of combined, nested, and general \(FuTS\), which we leave for further study.

Acknowledgments The authors are grateful to Rocco De Nicola, Fabio Gadducci, Daniel Gebler, Michele Loreti, Jan Rutten, and Ana Sokolova for fruitful discussions on the subject and useful suggestions. The constructive comments by the reviewers have been of help and are much appreciated. DL and MM acknowledge support by EU Project n. 600708 A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours (QUANTICOL). This research has been partially conducted while EV was spending a sabbatical leave at the CNR/ISTI. EV gratefully acknowledges the hospitality and support during his stay in Pisa.

References


