Stochastic bisimulation and performance evaluation in discrete time stochastic and deterministic Petri box calculus dtsdPBC

Igor V. Tarasyuk
A.P. Ershov Institute of Informatics Systems,
Siberian Branch of the Russian Academy of Sciences,
Acad. Lavrentiev pr. 6, 630090 Novosibirsk, Russian Federation
itar@iis.nsk.su

Abstract

We propose dtsdPBC, an extension with deterministically timed multiactions of discrete time stochastic and immediate Petri box calculus (dtsiPBC), previously presented by I.V. Tarasyuk, H. Macià and V. Valero. In dtsdPBC, non-negative integers specify deterministic multiactions with fixed (including zero) time delays. The step operational semantics is constructed via labeled probabilistic transition systems. The Petri net denotational semantics is defined via dtsd-boxes, a subclass of labeled discrete time stochastic Petri nets with deterministic transitions. We also define step stochastic bisimulation equivalence of the process expressions, which is used to compare the qualitative and quantitative behaviour of the specified processes. The consistency of the operational and denotational semantics of dtsdPBC up to that equivalence is established.

In order to evaluate performance in dtsdPBC, the underlying semi-Markov chains (SMCs) and (reduced) discrete time Markov chains (DTMCs and RDTMCs) of the process expressions are analyzed. We explain how step stochastic bisimulation equivalence of the expressions can be used for quotienting their transition systems and Markov chains, as well as to compare the stationary behaviour and residence time properties. We prove that the equivalence guarantees coincidence of the functional and performance characteristics and therefore can be used to simplify performance analysis of the algebraic processes. In a case study, a method of modeling, performance evaluation and behaviour reduction for concurrent systems with discrete fixed and stochastic delays is applied to the generalized shared memory system with maintenance. We also determine the main advantages of dtsdPBC by comparing it with other well-known or similar SPAs.

Keywords: stochastic process algebra, stochastic Petri net, Petri box calculus, discrete time, stochastic multiaction, deterministic multiaction, transition system, operational semantics, stochastic transition, deterministic transition, dtsd-box, denotational semantics, Markov chain, performance evaluation, reduction, stochastic bisimulation, quotient, shared memory system.

Contents

1 Introduction

Algebraic process calculi, like Communicating Sequential Processes (CSP) [167], Algebra of Communicating Processes (ACP) [24] and Calculus of Communicating Systems (CCS) [223] are well-known formal models for specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In recent decades, stochastic extensions of PAs were proposed, such as Markovian TImed Processes and Performability (Performance and dependability) evaluation (MTIPP) [162], Performance Evaluation Process Algebra (PEPA) [164] and Extended Markovian Process Algebra (EMPA) [36]. Unlike standard PAs, stochastic process algebras (SPAs) do not just specify actions which can occur

(qualitative features), but they associate with the actions the distribution parameters of their random time delays (quantitative characteristics).

1.1 Petri box calculus

PAs specify concurrent systems in a compositional way via an expressive formal syntax. On the other hand, Petri nets (PNs) provide a graphical representation of such systems and capture explicit asynchrony in their behaviour. To combine the advantages of both models, a semantics of algebraic formulas via PNs was defined.

Petri box calculus (PBC) [39, 41, 40] is a flexible and expressive process algebra developed as a tool for specification of the PNs structure and their interrelations. Its goal was also to propose a compositional semantics for high level constructs of concurrent programming languages in terms of elementary PNs. Formulas of PBC are combined not from single (visible or invisible) actions and variables, like in CCS, but from multisets of elementary actions and their conjugates, called multiactions (*basic formulas*). The empty multiset of actions is interpreted as the silent multiaction specifying some invisible activity. In contrast to CCS, synchronization is separated from parallelism (*concurrent constructs*). Synchronization is a unary multi-way stepwise operation, based on communication of actions and their conjugates. This extends the CCS approach with conjugate matching labels. Synchronization in PBC is asynchronous, unlike that in Synchronous CCS (SCCS) [223]. Other operations are sequence and choice (*sequential constructs*). The calculus includes also restriction and relabeling (*abstraction constructs*). To specify infinite processes, refinement, recursion and iteration operations were added (*hierarchical constructs*). Thus, unlike CCS, PBC has an additional iteration operation to specify infinite behaviour when the semantic interpretation in finite PNs is possible. PBC has a step operational semantics in terms of labeled transition systems, based on the rules of structural operational semantics (SOS). The operational semantics of PBC is of step type, since its SOS rules have transitions with (multi)sets of activities, corresponding to simultaneous executions of activities (steps). A denotational semantics of PBC was proposed via a subclass of PNs equipped with an interface and considered up to isomorphism, called Petri boxes. For more detailed comparison of PBC with other process algebras and the reasoning about importance of non-interleaving semantics see [39, 40].

The extensions of PBC with a deterministic, a nondeterministic or a stochastic model of time were presented.

1.2 Time extensions of Petri box calculus

To specify systems with time constraints, deterministic (fixed) or nondeterministic (interval) delays are used.

A time extension of PBC with a nondeterministic time model, called time Petri box calculus (tPBC), was proposed in [183]. In tPBC, timing information is added by associating time intervals (the earliest and the latest firing time) with instantaneous *actions*. tPBC has a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled time Petri nets (LtPNs), based on tPNs [222] and called time Petri boxes (ct-boxes).

Another time enrichment of PBC, called Timed Petri box calculus (TPBC), was defined in [211, 212], it accommodates a deterministic model of time. In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. Additionally, in TPBC there exist no "illegal" multiaction occurrences, unlike tPBC. The complexity of "illegal" occurrences mechanism was one of the main intentions to construct TPBC though this calculus appeared to be more complicated than tPBC. TPBC has a step timed operational semantics in terms of labeled transition systems. The denotational semantics of TPBC was defined in terms of a subclass of labeled Timed Petri nets (LTPNs), based on TPNs [256] and called Timed Petri boxes (T-boxes). tPBC and TPBC differ in ways they capture time information, and they are not in competition but complement each other.

The third time extension of PBC, called arc time Petri box calculus (atPBC), was constructed in [232, 233], and it implements a nondeterministic time. In atPBC, multiactions are associated with time delay intervals. atPBC possesses a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined on a subclass of labeled arc time Petri nets (atPNs), based of those from [47, 150], where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes).

tPBC, TPBC and atPBC, all adapt the discrete time approach, but TPBC has no immediate (multi)actions (those with zero delays).

1.3 Stochastic extensions of Petri box calculus

The set of states for the systems with deterministic or nondeterministic delays often differs drastically from that for the timeless systems, hence, the analysis results for untimed systems may be not valid for the time ones. To solve this problem, stochastic delays are considered, which are the random variables with a (discrete or continuous) probability distribution. If the random variables governing delays have an infinite support then the corresponding SPA can exhibit all the same behaviour as its underlying untimed PA.

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [204, 200]. In sPBC, multiactions have stochastic delays that follow (negative) exponential distribution. Each multiaction is equipped with a rate that is a parameter of the corresponding exponential distribution. The instantaneous execution of a stochastic multiaction is possible only after the corresponding stochastic time delay. The calculus has an interleaving operational semantics defined via transition systems labeled with multiactions and their rates. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs, based on CTSPNs [227, 213, 15, 75, 16] and called stochastic Petri boxes (s-boxes). In sPBC, performance of the processes is evaluated by analyzing their underlying continuous time Markov chains (CTMCs). In [201], a number of new equivalence relations were proposed for regular terms of sPBC to choose later a suitable candidate for a congruence.

sPBC was enriched with immediate multiactions having zero delays in [202, 203]. We call such an extension generalized sPBC (gsPBC). An interleaving operational semantics of gsPBC was constructed via transition systems labeled with stochastic or immediate multiactions together with their rates or probabilities. A denotational semantics of gsPBC was defined via a subclass of labeled generalized stochastic PNs, based on GSPNs [213, 214, 76, 15, 16] and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on the semi-Markov chains (SMCs).

PBC has a step operational semantics, whereas sPBC has an interleaving one. In step semantics, parallel executions of activities (steps) are permitted while in interleaving semantics, we can execute only single activities. Hence, a stochastic extension of PBC with a step semantics was needed to keep the concurrency degree of behavioural analysis at the same level as in PBC. As mentioned in [226, 228], in contrast to continuous time approach (used in sPBC), discrete time approach allows for constructing models of common clock systems and clocked devices. In such models, multiple transition firings (or executions of multiple activities) at time moments (ticks of the central clock) are possible, resulting in a step semantics. Moreover, employment of discrete stochastic time fills the gap between the models with deterministic (fixed) time delays and those with continuous stochastic time delays. As argued in [1], arbitrary delay distributions are much easier to handle in a discrete time domain. In [209, 210, 207], discrete stochastic time was preferred to enable simultaneous expiration of multiple delays.

In [266, 267, 268, 270], we presented an extension of the algebra PBC, called discrete time stochastic PBC (dtsPBC). In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [226, 228] and called discrete time stochastic Petri boxes (dts-boxes). The performance evaluation in dtsPBC is accomplished via the underlying discrete time Markov chains (DTMCs) of the algebraic processes. A variety of stochastic equivalences were proposed to identify stochastic processes with similar behaviour which are differentiated by the semantic equivalence. The interrelations of all the introduced equivalences were studied. Since dtsPBC has a discrete time semantics and geometrically distributed sojourn time in the process states, unlike sPBC with continuous time semantics and exponentially distributed delays, the calculi apply two different approaches to the stochastic extension of PBC, in spite of some similarity of their syntax and semantics inherited from PBC. The main advantage of dtsPBC is that concurrency is treated like in PBC having step semantics, whereas in sPBC parallelism is simulated by interleaving, obliging one to collect the information on causal independence of activities before constructing the semantics.

In [276, 277, 278, 279, 280], an enhanced calculus discrete time stochastic and immediate PBC (dtsiPBC) was proposed as an extension with immediate multiactions of dtsPBC. Immediate multiactions increase the specification capability: they can model logical conditions, probabilistic branching, instantaneous probabilistic choices and activities whose durations are negligible in comparison with those of others. They are also used to specify urgent activities and the ones that are not relevant for performance evaluation. Thus, immediate multiactions can be considered as a kind of instantaneous dynamic state adjustment and, in many cases, they result in a simpler and more clear system representation. The step operational semantics of dtsiPBC was constructed with the use of labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic and immediate PNs (LDTSIPNs), based on the extension of DTSPNs [226, 228] with transition labeling and immediate transitions, called dtsi-boxes. The corresponding stochastic process, the underlying SMC, was constructed and investigated, with the purpose of performance evaluation. In addition, the alternative solution methods were developed, based on the underlying ordinary and reduced DTMCs. Step stochastic bisimulation equivalence of the process expressions was defined to compare and reduce their transition systems and Markov chains, as well as to identify the stationary behaviour.

1.4 Equivalence relations

A notion of equivalence is very important in theory of computing systems. Equivalences are applied both to compare behaviour of systems and reduce their structure. There is a wide diversity of behavioural equivalences, and their interrelations are well explored in the literature. The best-known and widely used one is bisimulation. Typically, the mentioned equivalences take into account only functional (qualitative) but not performance (quantitative) aspects. Additionally, the equivalences are usually interleaving ones, i.e. they interpret concurrency as sequential nondeterminism. Interleaving equivalences permit to imitate parallel execution of actions via all possible occurrence sequences (interleavings) of them. Step equivalences require instead simulating such a parallel execution by simultaneous occurrence (step) of all the involved actions. To respect quantitative features of behaviour, probabilistic equivalences have additional requirement on execution probabilities. Two equivalent processes must be able to execute the same sequences of actions, and for every such sequence, its execution probabilities within both processes should coincide. In case of probabilistic bisimulation equivalence, the states from which similar future behaviours start are grouped into equivalence classes that form elements of the aggregated state space. From every two bisimilar states, the same actions can be executed, and the subsequent states resulting from execution of an action belong to the same equivalence class. In addition, for both states, the cumulative probabilities to move to the same equivalence class by executing the same action coincide. A different kind of quantitative relations is called Markovian equivalences, which take rate (the parameter of exponential distribution that governs time delays) instead of probability. Note that the probabilistic equivalences can be seen as discrete time analogues of the Markovian ones, since the latter are defined as the continuous time relations.

Interleaving probabilistic weak trace equivalence was introduced in [88] on labeled probabilistic transition systems. Interleaving probabilistic strong bisimulation equivalence was proposed in [194] on the same model. Interleaving probabilistic equivalences were defined for probabilistic processes in [178, 138]. Interleaving Markovian strong bisimulation equivalence was constructed in [162] for MTIPP, in [164] for PEPA and in [36, 35, 25] for EMPA. Several variants of interleaving Markovian weak bisimulation equivalence were considered in [73] on Markovian process algebras, in [75] on labeled CTSPNs and in [76] on labeled GSPNs. In [78, 79, 269], interleaving and step probabilistic trace and bisimulation equivalences that abstract from silent actions were defined on labeled DTSPNs (LDTSPNs) with invisible transitions, including the back and back-forth variants of the considered bisimulation relations. In [30, 31], interleaving probabilistic and Markovian trace, testing and bisimulation equivalences on the respective sequential probabilistic (PPC) and Markovian (MPC) process calculi were logically characterized. In [26, 27, 28], a comparison of interleaving Markovian trace, test, strong and weak bisimulation equivalences was carried out on sequential (SMPC or MPC) and concurrent (CMPC) Markovian process calculi. In [102], interleaving strong and branching probabilistic bisimulation equivalences were defined on the calculus of Interactive Probabilistic Chains (IPC).

Further, in [37, 38, 29], a wide range of probabilistic and Markovian trace, testing and bisimulation equivalences were investigated on Uniform Labeled Transition Systems (ULTraS) that capture different models of concurrent processes, such as fully nondeterministic processes (labeled transition systems, LTSs), fully probabilistic processes (labeled DTMCs), fully stochastic processes (labeled continuous time Markov chains, CTMCs), nondeterministic and probabilistic processes (Markov decision processes, MDPs), nondeterministic and stochastic processes (continuous time MDPs, CTMDPs). In [195], the bisimulation equivalences induced by some specific labeled state-to Function Transition Systems (FuTSs) were shown to coincide with the equivalences underlying the fragments of PEPA, Interactive Markov Language (IML) for Interactive Markov Chains (IMC) [158, 159, 33], Timed Process Calculus (TPC) [2] and Markov Automata Language (MAL) for Markov Automata Process Algebra (MAPA) [281, 282, 141, 142]. In [205, 246, 206], ordinary bisimulation (strong), quasi-lumping bisimulation (approximate strong) and proportional bisimulation equivalences on the PEPA components were investigated that induce, respectively, ordinary, quasi- and proportional lumpabilities on the corresponding CTMCs.

Nevertheless, no appropriate equivalence notion was defined for *parallel SPAs*. The non-interleaving bisimulation equivalence in Generalized Semi-Markovian Process Algebra (GSMPA) [66, 64] uses Start-Termination- (ST-) semantics for action particles while in Generalized Stochastic π-calculus (that we call GSπ) [251] it is based on a sophisticated labeling.

1.5 Our contributions

In this paper, we present an extension of dtsiPBC with deterministic multiactions, called *discrete time stochastic and deterministic Petri box calculus* (dtsdPBC), which enhances the expressiveness of dtsiPBC and extends the application area of the associated specification and analysis techniques. In dtsdPBC, besides the probabilities from the real-valued interval $(0; 1)$ that are used to calculate discrete time delays of stochastic multiactions, also non-negative integers are used to specify fixed delays of deterministic multiactions (including zero delay, which is the case of immediate multiactions). To resolve conflicts among deterministic multiactions, they are additionally

equipped with positive real-valued weights. As argued in [309, 305, 306], a combination of deterministic and stochastic delays fits well to model technical systems with constant (fixed) durations of the regular non-random activities and probabilistically distributed (stochastic) durations of the randomly occurring activities.

It should be stressed that dtsdPBC is rather a qualitative than merely a quantitative extension of dtsiPBC. The main reason is that in the former calculus, the probability of transitions between markings (untimed states, represented by overbars and underbars in the process expressions) generally depends both on the current marking and for how long the deterministic multiactions were enabled. Hence, the marking change probabilities in dtsdPBC may not possess the Markov (memoryless) property. Thus, the timer values should be associated with deterministic multiactions to specify the process states and then obtain the (semi-)Markovian state change probabilities as a result of "unfolding" the discrete residence times at the markings. In other words, the longer that one delays at the markings should be splitted into one time units and be allocated with the consecutive process states, in order to obtain a (semi-)Markovian model. Another reason is that, unlike dtsiPBC, the activities of different types can be executed from the the same marking in dtsdPBC, depending on the (decreasing) timer values of the enabled deterministic multiactions. In particular, the enabled stochastic multiactions may preempt (interrupt) the enabled waiting (positively delayed deterministic) ones that cannot be executed at the next time moment from a marking. Otherwise, only enabled waiting multiactions are executed from it. Note that the stochastic multiactions interrupting the waiting ones with the non-expired timers can be used to model failures while data transfer in communication protocols. Immediate multiactions are always executed first and separately from other types of activities. It is supposed that the activities are ordered according to their priorities as follows: immediate (highest priority), waiting (middle priority) and stochastic (lowest priority) multiactions.

Our novel approach was inspired by some ideas on combining deterministic and stochastic discrete time transition delays in DTSPNs [226, 228], discrete time deterministic and stochastic PNs (DTDSPNs) [309, 305, 306], dts-nets [1], non-Markovian SPNs (NMSPNs) [171] and stochastic preemptive time PNs (spTPNs) [71] (all with parallel step semantics), as well as in Defective Discrete Phase SPNs (DDP-SPNs) [89], discrete deterministic and stochastic PNs (DDSPNs) [307, 308] and DTDSPNs from [311, 312, 310] (all featuring interleaving semantics). The key idea was to interpret the waiting multiactions with the timer values (remaining times to execute) one as the (stochastic) transitions of DTSPNs [226, 228] with the conditional probability 1. Then the waiting multiactions with the timer values greater than one are ignored, i.e. when enabled, they are executed with the probability 0 at the next time moment.

The step operational semantics of dtsdPBC is constructed with the use of labeled probabilistic transition systems. Its denotational semantics is defined in terms of a special interface-featured subclass of labeled discrete time stochastic and deterministic Petri nets (LDTSPNs with deterministic transitions, LDTSDPNs), based on the extension of DTSPNs [226, 228] with transition labeling and deterministic transitions, called dtsd-boxes.

With the purpose of performance evaluation in dtsdPBC, the corresponding stochastic process of the process expressions is constructed and investigated, which is a semi-Markov chain (SMC). The obtained stationary probability masses and average sojourn times in the states of the SMC are used to calculate the performance measures (indices) of interest. We call that approach *embedding*, since the SMC is described by the embedded DTMC (EDTMC) specifying the state change probabilities, together with the probability distribution functions (PDFs) of the residence times in the states. In addition, the alternative solution methods are developed, based on the underlying discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states, i.e. those with zero sojourn (residence) times. The approach based on the DTMC allows one to avoid the costly intermediate stages of building the EDTMC, weighting the probability masses in the states by their average sojourn times (rescaling) and final normalization. We call that approach *abstraction*, since we abstract from all vanishing states by taking into account only the (normalized) DTMC-based stationary probabilities of the tangible states, i.e. those with positive sojourn times. The approach based on the RDTMC simplifies performance analysis of large systems due to eliminating the non-stop transit (vanishing) states where only instantaneous activities can be executed, resulting in a smaller model having only tangible states that can be solved directly with less efforts. We call that approach *elimination*, since we eliminate all vanishing states. We formally prove that the reduced SMC (RSMC) coincides with RDTMC.

We propose step stochastic bisimulation equivalence allowing one to identify algebraic processes with similar behaviour that are however differentiated by the semantics of the calculus. It enhances the corresponding relation from dtsiPBC, in that we now have to make difference between the states with positive sojourn times (called tangible states) and those with zero sojourn times (called vanishing states). Therefore, in the definition of step stochastic bisimulation for dtsdPBC, we add a condition stating that vanishing states may only be related with vanishing states. We establish consistency of the operational and denotational semantics of dtsdPBC up to step stochastic bisimulation equivalence, meaning that the transition systems of the process expressions are equivalent to the reachability graphs of their dtsd-boxes. We examine the interrelations of the proposed notion with other equivalences of the algebra. We describe how step stochastic bisimulation equivalence can be used to reduce transition systems and SMCs, DTMCs and RDTMCs of the process expressions while preserving the

qualitative and quantitative characteristics. We demonstrate isomorphism of the quotient (by that equivalence) SMCs (derived from the transition systems) of the process expressions and (derived from the reachability graphs) of their dtsd-boxes. We explore how the quotienting is related to extraction (of Markov chains from transition systems), embedding and reduction, by analyzing the transition probability matrices (TPMs) of the quotient DTMCs, EDTMCs and RDTMCs. In this way, we show that the reduced quotient TPMs coincide with the quotient reduced TPMs for DTMCs of the process expressions. We prove that the mentioned equivalence guarantees identity of the stationary behaviour and residence time properties in the equivalence classes. This implies coincidence of performance indices based on steady-state probabilities of the modeled stochastic systems. The equivalences possessing the property can be used to reduce the state space of a system and thus simplify its performance evaluation, what is usually a complex problem due to the state space explosion.

The theory developed is illustrated with a series of interesting and non-trivial examples that include the travel system model. The examples demonstrate how to construct the transition systems of the expressions with different types of multiactions (stochastic and deterministic, the latter consisting of immediate and waiting) and various operations, as well as the reachability graphs of the corresponding dtsd-boxes. The resulted transition systems and reachability graphs have all 3 possible kinds of states (stochastically tangible, waitingly tangible and vanishing) and all 4 kinds of transitions (that capture executions of the empty multiset, stochastic, waiting or immediate multiactions). From stochastically tangible (s-tangible) states, only the empty set or stochastic multiactions can be executed at the next time moment (after one unit delay). From waitingly tangible (w-tangible) states, only waiting multiactions can be executed at the next time moment. From vanishing states, only immediate multiactions can be executed at the same time moment (after zero delay). The examples show the specification flexibility and expressive power of the calculus, the most important features and peculiarities of its semantics, as well as application of step stochastic bisimulation to the performance analysis methods within dtsdPBC.

We present an extended case study of a system consisting of two processors and a common shared memory with maintenance (service) that explains how to specify, model and analyze performance of concurrent systems with stochastic and deterministic delays within the calculus. The proposed application example also demonstrates how to reduce the systems behaviour while preserving their performance indices and making easier the performance evaluation. We consider a generalized variant of the shared memory system by treating the probabilities and weights from the standard system's specification as variables (parameters) over continuous sets of values. First, we consider a process expression of the concrete system that differentiates among the processors, and then we analyze its functionality and performance using the corresponding transition system, SMC, DTMC and RDTMC. Second, we model the abstract system that abstracts from the names of the processors (by making identical the actions from their specifications), in order to apply reduction by the equivalence. The quotients of the abstract system's behaviour (represented by the transition system, SMC, DTMC and RDTMC) by step stochastic bisimulation equivalence are constructed. Third, the generalized probabilities of the reduced quotient DTMC (coinciding with the quotient RDTMC) are treated as parameters (or variables of the performance index functions) to be adjusted for the performance optimization. The analytical expressions for the performance measures are explored, after which the global and local extrema (maximal and minimal values) of the corresponding rational functions of the parameters are found.

In the enhanced related work overview, strong points of dtsdPBC with respect to other SPAs are detected. In overall, we compare dtsdPBC with about 90 existing SPAs and then classify them according to the time model and concept, parallelism in the (operational) semantics, existence of immediate or positively deterministic (waiting) (multi)actions and (distribution) types of the stochastic delays. If to compare dtsdPBC with the classical SPAs MTIPP, PEPA and EMPA, the first main difference between them comes from PBC, since dtsdPBC is based on this calculus: all algebraic operations and a notion of multiaction are inherited from PBC. The second main difference is discrete probabilities of activities induced by the discrete time approach, whereas action rates are used in the standard SPAs with continuous time. As a consequence, dtsdPBC has a non-interleaving step operational semantics. This is in contrast to the classical SPAs, where concurrency is modeled by interleaving because of the continuous probability distributions of action delays and the race condition applied when several actions can be executed in a state. The third main difference is deterministic (particularly, immediate) multiactions. There are no even instantaneous activities in MTIPP and PEPA while immediate actions in EMPA can have different priority levels. In dtsdPBC, all immediate (zero deterministic) multiactions have the same (highest) priority, and all waiting (positive deterministic) multiactions have the same (medium) priority (leaving the lowest priority to stochastic multiactions). The intention is to simplify the specification and analysis, since weights (assigned also to immediate actions in EMPA) are enough to denote preferences among deterministic multiactions and to produce the conformable probabilistic behaviours. The salient point of dtsdPBC is a combination of deterministic multiactions, discrete stochastic time and step semantics in an SPA.

In the extensive discussion, analytical solution, concurrency interpretation, application area and general advantages of dtsdPBC are explained.

In [271], we have reported the first results on the mentioned topics: the syntax, operational and denotational semantics of dtsdPBC, as well as the methods of performance analysis within the calculus. The present paper is an improvement and substantial (more than three times longer) extension of that work with the new results, (partially) presented earlier in [272, 273, 274, 275]. The novel topics include definitions and interrelations of the stochastic equivalences, quotienting the transition systems and Markov chains by step stochastic bisimulation equivalence, preservation of the stationary behaviour and residence time properties by that equivalence, case study of the generalized shared memory system with maintenance, related work overview and discussion.

Thus, the main contributions of the paper are the following.

- Syntax of new powerful and expressive discrete time SPA with deterministic activities, called dtsdPBC.
- Parallel step operational semantics of dtsdPBC in terms of labeled probabilistic transition systems.
- Petri net denotational semantics of dtsdPBC via discrete time stochastic and deterministic Petri nets.
- Performance analysis via underlying semi-Markov chains and (reduced) discrete time Markov chains.
- Stochastic bisimulation used for behaviour-preserving reduction that simplifies the performance evaluation.
- Extended case study of the shared memory system showing how to apply the theoretical results in practice.

1.6 Structure of the paper

The paper is organized as follows. In Section 2, the syntax of algebra dtsdPBC is presented. In Section 3, we construct the step operational semantics of the calculus in terms of labeled probabilistic transition systems. In Section 4, we propose the Petri net denotational semantics based on dtsd-boxes, a subclass of novel LDTSDPNs. In Section 5, the underlying stochastic process (SMC) is defined and analyzed, then the alternative solution methods are outlined, based on the corresponding DTMCs and RDTMCs. Step stochastic bisimulation equivalence, used to prove consistency of the both semantics, is defined and investigated in Section 6. In Section 7, we explain how to reduce transition systems and underlying SMCs of process expressions modulo the equivalence. In Section 8, the introduced equivalence is applied to the stationary behaviour comparison to verify the performance preservation. In Section 9, the generalized shared memory system with maintenance is presented as a case study. The difference between dtsdPBC and other well-known or similar SPAs is considered in Section 10. The advantages of dtsdPBC are explained in Section 11. Finally, Section 12 summarizes the results obtained and outlines research perspectives in this area. The long and complex proofs are moved to Appendix A.

2 Syntax

In this section, we propose the syntax of dtsdPBC.

2.1 Activities and operations

We recall a definition of multiset that is an extension of the set notion by allowing several identical elements.

Definition 2.1 *Let X be a set. A finite* multiset (bag) *M over X is a mapping $M : X \to \mathbb{N}$ such that $|\{x \in X \mid M(x) > 0\}| < \infty$, i.e. it can contain a finite number of elements only.*

We denote the *set of all finite multisets* over a set X by $\mathbb{N}_{fin}^X$. Let $M, M' \in \mathbb{N}_{fin}^X$. The *cardinality* of M is defined as $|M| = \sum_{x \in X} M(x)$. We write $x \in M$ if $M(x) > 0$ and $M \subseteq M'$ if $\forall x \in X\ M(x) \leq M'(x)$. We define $(M + M')(x) = M(x) + M'(x)$ and $(M - M')(x) = \max\{0, M(x) - M'(x)\}$. When $\forall x \in X,\ M(x) \leq 1$, M can be interpreted as a proper set and denoted by $M \subseteq X$. The *set of all subsets (powerset)* of X is denoted by 2^X.

Let $Act = \{a, b, \ldots\}$ be the set of *elementary actions*. Then $\widehat{Act} = \{\hat{a}, \hat{b}, \ldots\}$ is the set of *conjugated actions (conjugates)* such that $\hat{a} \neq a$ and $\hat{\hat{a}} = a$. Let $\mathcal{A} = Act \cup \widehat{Act}$ be the set of *all actions*, and $\mathcal{L} = \mathbb{N}_{fin}^{\mathcal{A}}$ be the set of *all multiactions*. Note that $\emptyset \in \mathcal{L}$, this corresponds to an internal move, i.e. the execution of a multiaction that contains no visible action names. The *alphabet* of $\alpha \in \mathcal{L}$ is defined as $\mathcal{A}(\alpha) = \{x \in \mathcal{A} \mid \alpha(x) > 0\}$.

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probabilities of stochastic multiactions are required not to be equal to 1 to avoid extra model complexity, since in this case one should assign with them weights, needed to make a choice when several stochastic multiactions with probability 1 can be executed from a state. The difficulty

is that when the stochastic multiactions with probability 1 occur in a step (parallel execution), all other with the less probabilities do not. In this case, the conflicts resolving requires a special attention, as discussed in [226, 228] within SPNs. This decision also allows us to avoid technical difficulties related to conditioning events with probability 0. In [13], formal approaches to causality of probabilistic systems represented by DTMCs were discussed and necessity of the positive probabilities of causes for the correct definition of conditional probabilities was explained. The probability 1 is left for (implicitly assigned to) waiting multiactions (positively delayed deterministic multiactions, to be defined later), which are delayed for at least one time unit before their execution and have weights to resolve conflicts with other waiting multiactions. On the other hand, there is no sense to allow probability 0 of stochastic multiactions, since they would never be performed in this case. Let $\mathcal{SL}$ be the set of *all stochastic multiactions*.

A *deterministic multiaction* is a pair $(\alpha, \natural_l^\theta)$, where $\alpha \in \mathcal{L}$, $\theta \in I\!N$ is the non-negative integer-valued *(fixed) delay* and $l \in I\!R_{>0} = (0; \infty)$ is the positive real-valued *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with execution of the deterministic multiaction at the discrete time moment when the corresponding delay has expired. Such weights are used to calculate the probabilities to execute sets of deterministic multiactions after their time delays. An *immediate multiaction* is a deterministic multiaction with the delay 0 while a *waiting multiaction* is a deterministic multiaction with a positive delay. In case of no conflicts among waiting multiactions, whose remaining times to execute (RTEs, to be explained later in more detail) are equal to one time unit, they are executed with probability 1 at the next time moment. Deterministic multiactions have a priority over stochastic ones while immediate multiactions have a priority over waiting ones. One can assume that all immediate multiactions have (the highest) priority 2 and all waiting multiactions have (the medium) priority 1, whereas all stochastic multiactions have (the lowest) priority 0. This means that in a state where all kinds of multiactions can occur, immediate multiactions always occur before waiting ones that, in turn, are always executed before stochastic ones. Different types of multiactions cannot participate together in some step (parallel execution), i.e. just the steps consisting only of immediate multiactions or waiting ones, or those including only stochastic multiactions, are allowed. Let $\mathcal{DL}$ be the set of *all deterministic multiactions*, $\mathcal{IL}$ be the set of *all immediate multiactions* and $\mathcal{WL}$ be the set of *all waiting multiactions*. Obviously, we have $\mathcal{DL} = \mathcal{IL} \cup \mathcal{WL}$.

Let us note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities, (fixed) delays and weights in the same specification. An *activity* is a stochastic or a deterministic multiaction. Let $\mathcal{SDL} = \mathcal{SL} \cup \mathcal{DL} = \mathcal{SL} \cup \mathcal{IL} \cup \mathcal{WL}$ be the set of *all activities*. The *alphabet* of an activity $(\alpha, \kappa) \in \mathcal{SDL}$ is defined as $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. The *alphabet* of a multiset of activities $\Upsilon \in I\!N_{fin}^{\mathcal{SDL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon}\mathcal{A}(\alpha)$. For an activity $(\alpha, \kappa) \in \mathcal{SDL}$, we define its *multiaction part* as $\mathcal{L}(\alpha, \kappa) = \alpha$ and its *probability* or *weight part* as $\Omega(\alpha, \kappa) = \kappa$ if $\kappa \in (0; 1)$; or $\Omega(\alpha, \kappa) = l$ if $\kappa = \natural_l^\theta$, $\theta \in I\!N$, $l \in I\!R_{>0}$. The *multiaction part* of a multiset of activities $\Upsilon \in I\!N_{fin}^{\mathcal{SIL}}$ is defined as $\mathcal{L}(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon} \alpha$.

Activities are combined into formulas (process expressions) by the following operations: *sequence* ;, *choice* [], *parallelism* $\|$, *relabeling* $[f]$ of actions, *restriction* rs over a single action, *synchronization* sy on an action and its conjugate, and *iteration* $[**]$ with three arguments: initialization, body and termination.

Sequence (sequential composition) and choice (choice composition) have a standard interpretation, like in other process algebras, but parallelism (parallel composition) does not include synchronization, unlike the corresponding operation in CCS [223].

Relabeling functions $f : \mathcal{A} \to \mathcal{A}$ are bijections preserving conjugates, i.e. $\forall x \in \mathcal{A}\ f(\hat{x}) = \widehat{f(x)}$. Relabeling is extended to multiactions in the usual way: for $\alpha \in \mathcal{L}$ we define $f(\alpha) = \sum_{x\in\alpha} f(x)$. Relabeling is extended to activities: for $(\alpha, \kappa) \in \mathcal{SDL}$, we define $f(\alpha, \kappa) = (f(\alpha), \kappa)$. Relabeling is extended to the multisets of activities as follows: for $\Upsilon \in I\!N_{fin}^{\mathcal{SDL}}$ we define $f(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon}(f(\alpha), \kappa)$.

Restriction over an elementary action $a \in Act$ means that, for a given expression, any process behaviour containing a or its conjugate $\hat{a}$ is not allowed.

Let $\alpha, \beta \in \mathcal{L}$ be two multiactions such that for some elementary action $a \in Act$ we have $a \in \alpha$ and $\hat{a} \in \beta$, or $\hat{a} \in \alpha$ and $a \in \beta$. Then, synchronization of α and β by a is defined as $\alpha \oplus_a \beta = \gamma$, where

$$\gamma(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$$

In other words, we require that $\alpha \oplus_a \beta = \alpha + \beta - \{a, \hat{a}\}$, i.e. we remove one exemplar of a and one exemplar of $\hat{a}$ from the multiset sum $\alpha + \beta$, since the synchronization of a and $\hat{a}$ produces $\emptyset$. Activities are synchronized with the use of their multiaction parts, i.e. the synchronization by a of two activities, whose multiaction parts α and β possess the properties mentioned above, results in the activity with the multiaction part $\alpha \oplus_a \beta$. We may synchronize activities of the same type only: either both stochastic multiactions or both deterministic ones *with the same delay*, since stochastic, waiting and immediate multiactions have different priorities, and diverse delays of waiting multiactions would contradict their joint timing. Hence, the multiactions of different types cannot be executed together (note also that the execution of immediate multiactions takes no time, unlike that

of waiting or stochastic ones). Synchronization by a means that, for a given expression with a process behaviour containing two concurrent activities that can be synchronized by a, there exists also the process behaviour that differs from the former only in that the two activities are replaced with the result of their synchronization.

In the iteration, the initialization subprocess is executed first, then the body is performed zero or more times, and finally, the termination subprocess is executed.

2.2 Process expressions

Static expressions specify the structure of processes, i.e. how activities are combined by operations in order to construct the composite process-algebraic formulas. As we shall see, static expressions correspond to unmarked LDTSDPNs (LDTSDPNs are marked by definition). Remember that a marking is the allocation of tokens in the places of a PN. Markings are used to describe dynamic behaviour of PNs in terms of transition firings.

We assume that every waiting multiaction has a countdown timer associated, whose value is the discrete time amount left till the moment when the waiting multiaction can be executed. Therefore, besides standard (unstamped) waiting multiactions in the form of $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, a special case of the *stamped* waiting multiactions should be considered in the definition of static expressions. Each (time) stamped waiting multiaction in the form of $(\alpha, \natural_l^\theta)^\delta$ has an extra superscript $\delta \in \{1, \ldots, \theta\}$ assigned that specifies a time stamp indicating the *latest* value of the countdown timer associated with that multiaction. The standard waiting multiactions have no time stamps, to demonstrate irrelevance of the timer values for them (for example, their timers have not yet started or have already finished their operation). The notions of the alphabet, multiaction part, weight part for (the multisets of) stamped waiting multiactions are defined, respectively, like those for (the multisets of) unstamped waiting multiactions.

By reasons of simplicity, we do not assign the timer value superscripts δ to immediate multiactions, which are a special case of deterministic multiactions $(\alpha, \natural_l^\theta)$ with the delay $\theta = 0$ in the form of $(\alpha, \natural_l^0)$, since their timer values can only be equal to 0. Analogously, the superscript δ might be omitted for the waiting multiactions $(\alpha, \natural_l^\theta)$ with the delay $\theta = 1$ in the form of $(\alpha, \natural_l^1)$, since the corresponding timer can only have a single value 1. Nevertheless, to maintain syntactic uniformity among waiting multiactions, we leave the timer value superscripts for those that are 1-delayed.

Definition 2.2 *Let* $(\alpha, \kappa) \in \mathcal{SDL}$, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$ *and* $a \in Act$. *A* static expression *of dtsdPBC is defined as*

$$E ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \textsf{ rs } a \mid E \textsf{ sy } a \mid [E * E * E].$$

Let *StatExpr* denote the set of *all static expressions* of dtsdPBC.

To make the grammar above unambiguous, one can add parentheses in the productions with binary operations: $(E; E)$, $(E[]E)$, $(E \| E)$. However, here and further we prefer the PBC approach and add them to resolve ambiguities only.

To avoid technical difficulties with the iteration operator, we should not allow any concurrency at the highest level of the second argument of iteration. This is not a severe restriction though, since we can always prefix parallel expressions by an activity with the empty multiaction part. Later on, in Example 4.14, we shall demonstrate that relaxing the restriction can result in LDTSDPNs which are not safe. Alternatively, we can use a different, safe, version of the iteration operator, but its net translation has six arguments. See also [40] for discussion on this subject. Remember that a PN is *n-bounded* ($n \in \mathbb{N}$) if for all its reachable (from the initial marking by the sequences of transition firings) markings there are at most n tokens in every place, and a PN is *safe* if it is 1-bounded.

Definition 2.3 *Let* $(\alpha, \kappa) \in \mathcal{SDL}$, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$ *and* $a \in Act$. *A* regular static expression *of dtsdPBC is defined as*

$$\begin{array}{l} E ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \textsf{ rs } a \mid E \textsf{ sy } a \mid [E * D * E], \\ \textit{where } D ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid D; E \mid D[]D \mid D[f] \mid D \textsf{ rs } a \mid D \textsf{ sy } a \mid [D * D * E]. \end{array}$$

Let *RegStatExpr* denote the set of *all regular static expressions* of dtsdPBC.

Let E be a regular static expression. The *underlying timer-free regular static expression* $\lfloor E$ of E is obtained by removing from it all timer value superscripts.

Further, the set of *all stochastic multiactions (from the syntax) of* E is $\mathcal{SL}(E) = \{(\alpha, \rho) \mid (\alpha, \rho)$ is a subexpression of $E\}$. The set of *all immediate multiactions (from the syntax) of* E is $\mathcal{IL}(E) = \{(\alpha, \natural_l^0) \mid (\alpha, \natural_l^0)$ is a subexpression of $E\}$. The set of *all waiting multiactions (from the syntax) of* E is $\mathcal{WL}(E) = \{(\alpha, \natural_l^\theta) \mid (\alpha, \natural_l^\theta)$ or $(\alpha, \natural_l^\theta)^\delta$ is a subexpression of E for $\delta \in \{1, \ldots, \theta\}\}$. Thus, the set of *all deterministic multiactions*

(from the syntax) of E is $\mathcal{DL}(E) = \mathcal{IL}(E) \cup \mathcal{WL}(E)$ and the set of *all activities (from the syntax) of* E is $\mathcal{SDL}(E) = \mathcal{SL}(E) \cup \mathcal{DL}(E) = \mathcal{SL}(E) \cup \mathcal{IL}(E) \cup \mathcal{WL}(E)$.

The set of *all activities (from the syntax) of* a regular static expression can also be defined in a structural way. Let $(\alpha, \kappa) \in \mathcal{SDL},\ (\alpha, \natural_l^\theta) \in \mathcal{WL},\ \delta \in \{1, \ldots, \theta\},\ E, F, K \in RegStatExpr$ and $a \in Act$. Then

1. $\mathcal{SDL}((\alpha, \kappa)) = \{(\alpha, \kappa)\}$;
2. $\mathcal{SDL}((\alpha, \natural_l^\theta)^\delta) = \{(\alpha, \natural_l^\theta)\}$;
3. $\mathcal{SDL}(E \circ F) = \mathcal{SDL}(E) \cup \mathcal{SDL}(F)\ (\circ \in \{;, [], \|\})$;
4. $\mathcal{SDL}(E[f]) = \mathcal{SDL}(E\ \mathsf{rs}\ a) = \mathcal{SDL}(E\ \mathsf{sy}\ a) = \mathcal{SDL}(E)$;
5. $\mathcal{SDL}([E * F * K]) = \mathcal{SDL}(E) \cup \mathcal{SDL}(F) \cup \mathcal{SDL}(K)$.

The set of *all stochastic multiactions (from the syntax) of* E is $\mathcal{SL}(E) = \mathcal{SDL}(E) \cap \mathcal{SL}$. The set of *all immediate multiactions (from the syntax) of* E is $\mathcal{IL}(E) = \mathcal{SDL}(E) \cap \mathcal{IL}$. The set of *all waiting multiactions (from the syntax) of* E is $\mathcal{WL}(E) = \mathcal{SDL}(E) \cap \mathcal{WL}$. Thus, the set of *all deterministic multiactions (from the syntax) of* E is $\mathcal{DL}(E) = \mathcal{SDL}(E) \cap \mathcal{DL} = \mathcal{IL}(E) \cup \mathcal{WL}(E)$.

Dynamic expressions specify the states of processes, i.e. some particular stages of the process behaviour. As we shall see, dynamic expressions correspond to LDTSDPNs (which are marked by default). Dynamic expressions are obtained from static ones, by annotating them with upper or lower bars which specify the active components of the system at the current moment of time. The dynamic expression with upper bar (the overlined one) $\overline{E}$ denotes the *initial*, and that with lower bar (the underlined one) $\underline{E}$ denotes the *final* state of the process specified by a static expression E.

For every overlined stamped waiting multiaction in the form of $\overline{(\alpha, \natural_l^\theta)^\delta}$, the superscript $\delta \in \{1, \ldots, \theta\}$ specifies the *current* value of the *running* countdown timer associated with the waiting multiaction. That decreasing discrete timer is started with the *initial* value θ (equal to the delay of the waiting multiaction) at the moment when the waiting multiaction becomes overlined. Then such a newly overlined stamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)^\theta}$ may be seen similar to the freshly overlined unstamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)}$. Such similarity will be captured by the structural equivalence, to be defined later.

While the stamped waiting multiaction stays overlined with the specified process execution, the timer decrements by one discrete time unit with each global time tick until the timer value becomes 1. This fact indicates that one unit of time remains till execution of that multiaction (the remaining time to execute, RTE, equals one). Then its execution should follow in the next moment with probability 1, in case there are no conflicting with it immediate multiactions or conflicting waiting multiactions whose RTEs equal to one, and it is not affected by restriction. An activity is said to be affected by restriction, if it is within the scope of a restriction operation with the argument action, such that it or its conjugate is contained in the multiaction part of that activity.

Definition 2.4 *Let* $E \in StatExpr$ *and* $a \in Act$. *A* dynamic expression *of dtsdPBC is defined as*

$$G ::= \overline{E} \mid \underline{E} \mid G; E \mid E; G \mid G[]E \mid E[]G \mid G\|G \mid G[f] \mid G\ \mathsf{rs}\ a \mid G\ \mathsf{sy}\ a \mid [G * E * E] \mid [E * G * E] \mid [E * E * G].$$

Let $DynExpr$ denote the set of *all dynamic expressions* of dtsdPBC.

Let G be a dynamic expression. The *underlying static (line-free) expression* $\lfloor G \rfloor$ of G is obtained by removing from it all upper and lower bars. Note that if the underlying static expression of a dynamic one is not regular, the corresponding LDTSDPN can be non-safe but it is 2-bounded in the worst case, like shown for PNs in [40]).

Definition 2.5 *A dynamic expression* G *is* regular *if its underlying static expression* $\lfloor G \rfloor$ *is regular.*

Let $RegDynExpr$ denote the set of *all regular dynamic expressions* of dtsdPBC.

Let G be a regular dynamic expression. The *underlying timer-free regular dynamic expression* $\lfloor G$ of G is obtained by removing from it all timer value superscripts.

Further, the set of *all stochastic multiactions (from the syntax) of* G is $\mathcal{SL}(G) = \mathcal{SL}(\lfloor G \rfloor)$. The set of *all immediate multiactions (from the syntax) of* G is $\mathcal{IL}(G) = \mathcal{IL}(\lfloor G \rfloor)$. The set of *all waiting multiactions (from the syntax) of* G is $\mathcal{WL}(G) = \mathcal{WL}(\lfloor G \rfloor)$. Thus, the set of *all deterministic multiactions (from the syntax) of* G is $\mathcal{DL}(G) = \mathcal{IL}(G) \cup \mathcal{WL}(G)$ and the set of *all activities (from the syntax) of* G is $\mathcal{SDL}(G) = \mathcal{SL}(G) \cup \mathcal{DL}(G) = \mathcal{SL}(G) \cup \mathcal{IL}(G) \cup \mathcal{WL}(G)$.

3 Operational semantics

In this section, we define the step operational semantics in terms of labeled transition systems.

3.1 Inaction rules

The inaction rules for dynamic expressions describe their structural transformations in the form of $G \Rightarrow \widetilde{G}$ which do not change the states of the specified processes. The goal of those syntactic transformations is to obtain the well-structured resulting expressions called operative ones to which no inaction rules can be further applied. As we shall see, the application of an inaction rule to a dynamic expression does not lead to any discrete time tick or any transition firing in the corresponding LDTSDPN, hence, its current marking stays unchanged.

Thus, an application of every inaction rule does not require any discrete time delay, i.e. the dynamic expression transformation described by the rule is accomplished instantly.

In Table 1, we define inaction rules for regular dynamic expressions being overlined and underlined static ones. In this table, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$, $E, F, K \in RegStatExpr$ and $a \in Act$. The first inaction rule suggests that the timer value of each newly overlined waiting multiaction is set to the delay of that waiting multiaction.

Table 1: Inaction rules for overlined and underlined regular static expressions

$\overline{(\alpha, \natural_l^\theta)} \Rightarrow \overline{(\alpha, \natural_l^\theta)^\theta}$	$\overline{E;F} \Rightarrow \overline{E};F$	$\underline{E};F \Rightarrow E;\overline{F}$	$E;\underline{F} \Rightarrow \underline{E;F}$
$\overline{E[]F} \Rightarrow \overline{E}[]F$	$\overline{E[]F} \Rightarrow E[]\overline{F}$	$\underline{E}[]F \Rightarrow \underline{E[]F}$	$E[]\underline{F} \Rightarrow \underline{E[]F}$
$\overline{E\|F} \Rightarrow \overline{E}\|\overline{F}$	$\underline{E}\|\underline{F} \Rightarrow \underline{E\|F}$	$\overline{E[f]} \Rightarrow \overline{E}[f]$	$\underline{E}[f] \Rightarrow \underline{E[f]}$
$\overline{E \text{ rs } a} \Rightarrow \overline{E} \text{ rs } a$	$\underline{E} \text{ rs } a \Rightarrow \underline{E \text{ rs } a}$	$\overline{E \text{ sy } a} \Rightarrow \overline{E} \text{ sy } a$	$\underline{E} \text{ sy } a \Rightarrow \underline{E \text{ sy } a}$
$\overline{[E * F * K]} \Rightarrow [\overline{E} * F * K]$	$[\underline{E} * F * K] \Rightarrow [E * \overline{F} * K]$	$[E * \underline{F} * K] \Rightarrow [E * \overline{F} * K]$	$[E * \underline{F} * K] \Rightarrow [E * F * \overline{K}]$
$[E * F * \underline{K}] \Rightarrow \underline{[E * F * K]}$			

In Table 2, we introduce inaction rules for regular dynamic expressions in the arbitrary form. In this table, $E, F \in RegStatExpr$, $G, H, \widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$. By reason of brevity, two distinct inaction rules with the same premises are collated in some cases, resulting in the inaction rules with double conclusion.

Table 2: Inaction rules for arbitrary regular dynamic expressions

$$\frac{G \Rightarrow \widetilde{G},\ \circ \in \{;, []\}}{G \circ E \Rightarrow \widetilde{G} \circ E,\ E \circ G \Rightarrow E \circ \widetilde{G}} \qquad \frac{G \Rightarrow \widetilde{G}}{G\|H \Rightarrow \widetilde{G}\|H,\ H\|G \Rightarrow H\|\widetilde{G}} \qquad \frac{G \Rightarrow \widetilde{G}}{G[f] \Rightarrow \widetilde{G}[f]} \qquad \frac{G \Rightarrow \widetilde{G},\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}}{G \circ a \Rightarrow \widetilde{G} \circ a}$$

$$\frac{G \Rightarrow \widetilde{G}}{[G * E * F] \Rightarrow [\widetilde{G} * E * F]} \qquad \frac{G \Rightarrow \widetilde{G}}{[E * G * F] \Rightarrow [E * \widetilde{G} * F],\ [E * F * G] \Rightarrow [E * F * \widetilde{G}]}$$

Example 3.1 *Let* $E = (\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3})$. *The following inferences by the inaction rules are possible from* $\overline{E}$*:*

$$\overline{(\{a\}, \natural_1^3)[](\{b\}, \tfrac{1}{3})} \Rightarrow \overline{(\{a\}, \natural_1^3)}[](\{b\}, \tfrac{1}{3}) \Rightarrow \overline{(\{a\}, \natural_1^3)^3}[](\{b\}, \tfrac{1}{3}), \quad \overline{(\{a\}, \natural_1^3)[](\{b\}, \tfrac{1}{3})} \Rightarrow (\{a\}, \natural_1^3)[]\overline{(\{b\}, \tfrac{1}{3})}.$$

Definition 3.1 *A regular dynamic expression* G *is* operative *if no inaction rule can be applied to it.*

Let *OpRegDynExpr* denote the set of *all operative regular dynamic expressions* of dtsdPBC.

Note that any dynamic expression can be always transformed into a (not necessarily unique) operative one by using the inaction rules.

In the following, we consider regular expressions only and omit the word "regular".

Definition 3.2 *The relation* $\approx\ = (\Rightarrow \cup \Leftarrow)^*$ *is a* structural equivalence *of dynamic expressions in dtsdPBC, where* * *is the reflexive and transitive closure operation. Thus, two dynamic expressions* G *and* G' *are* structurally equivalent*, denoted by* $G \approx G'$*, if they can be reached from each other by applying the inaction rules in a forward or a backward direction.*

Let X be some set. We denote the Cartesian product $X \times X$ by X^2. Let $\mathcal{E} \subseteq X^2$ be an equivalence relation on X. Then the *equivalence class* (with respect to $\mathcal{E}$) of an element $x \in X$ is defined by $[x]_\mathcal{E} = \{y \in X \mid (x, y) \in \mathcal{E}\}$. The equivalence $\mathcal{E}$ partitions X into the *set of equivalence classes* $X/_\mathcal{E} = \{[x]_\mathcal{E} \mid x \in X\}$.

Let G be a dynamic expression. Then $[G]_{\approx} = \{H \mid G \approx H\}$ is the equivalence class of G with respect to the structural equivalence, called the (corresponding) *state*. Next, G is an *initial* dynamic expression, denoted by $init(G)$, if $\exists E \in RegStatExpr\ G \in [\overline{E}]_{\approx}$. Further, G is a *final* dynamic expression, denoted by $final(G)$, if $\exists E \in RegStatExpr\ G \in [\underline{E}]_{\approx}$.

Example 3.2 *Let E be from Example 3.1. We have $init(\overline{E})$ and $[\overline{E}]_{\approx} = \{\overline{(\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)}[](\{b\}, \frac{1}{3}), (\{a\}, \natural_1^3)[]\overline{(\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)^3}[](\{b\}, \frac{1}{3}), (\{a\}, \natural_1^3)^3[]\overline{(\{b\}, \frac{1}{3})}, \{\overline{(\{a\}, \natural_1^3)^3}[](\{b\}, \frac{1}{3})\}$. Then $[\overline{E}]_{\approx} \cap OpRegDynExpr = \{(\{a\}, \natural_1^3)[]\overline{(\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)^3}[](\{b\}, \frac{1}{3}), (\{a\}, \natural_1^3)^3[]\overline{(\{b\}, \frac{1}{3})}\}$.*

Let G be a dynamic expression and $s = [G]_{\approx}$. The set of *all enabled stochastic multiactions of s* is $EnaSto(s) = \{(\alpha, \rho) \in \mathcal{SL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \rho)}$ is a subexpression of $H\}$, i.e. it consists of all stochastic multiactions that, being overlined, are the subexpressions of some operative dynamic expression from the state s. Analogously, the set of *all enabled immediate multiactions of s* is $EnaImm(s) = \{(\alpha, \natural_l^0) \in \mathcal{IL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^0)}$ is a subexpression of $H\}$. The set of *all enabled waiting multiactions of s* is $EnaWait(s) = \{(\alpha, \natural_l^\theta) \in \mathcal{WL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^\theta)^\delta},\ \delta \in \{1, \ldots, \theta\},$ is a subexpression of $H\}$, i.e. it consists of all waiting multiactions that, being superscribed with the values of their timers and overlined, are the subexpressions of some operative dynamic expression from the state s. The set of *all newly enabled waiting multiactions of s* is $EnaWaitNew(s) = \{(\alpha, \natural_l^\theta) \in \mathcal{WL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^\theta)^\theta}$ is a subexpression of $H\}$, i.e. it consists of all waiting multiactions that, being superscribed with the initial values of their timers (delays of those waiting multiactions) and overlined, are the subexpressions of some operative dynamic expression from the state s.

Thus, the set of *all enabled deterministic multiactions of s* is $EnaDet(s) = EnaImm(s) \cup EnaWait(s)$ and the set of *all enabled activities of s* is $Ena(s) = EnaSto(s) \cup EnaDet(s) = EnaSto(s) \cup EnaImm(s) \cup EnaWait(s)$. As we shall see, $Ena(s) = Ena([G]_{\approx})$ is an algebraic analogue of the set of all transitions enabled at the initial marking of the LDTSDPN corresponding to G. Note that the activities, resulted from synchronization, are not present explicitly in the syntax of the dynamic expressions. Nevertheless, their enabledness status can be recovered by observing that of the pair of synchronized activities from the syntax (they both should be enabled for enabling their synchronous product), even if they are affected by restriction after the synchronization.

Example 3.3 *Let E be from Example 3.1. Then we have $EnaSto([\overline{E}]_{\approx}) = \{(\{b\}, \frac{1}{3})\}$, $EnaImm([\overline{E}]_{\approx}) = \emptyset$ and $EnaWait([\overline{E}]_{\approx}) = EnaWaitNew([\overline{E}]_{\approx}) = \{(\{a\}, \natural_1^3)\}$, hence, $Ena([\overline{E}]_{\approx}) = \{(\{a\}, \natural_1^3), (\{b\}, \frac{1}{3})\}$.*

Definition 3.3 *An operative dynamic expression G is* saturated (with the values of timers)*, if each enabled waiting multiaction of $[G]_{\approx}$, being (certainly) superscribed with the value of its timer and possibly overlined, is the subexpression of G.*

Let $SaOpRegDynExpr$ denote the set of *all saturated operative dynamic expressions* of dtsdPBC.

Proposition 3.1 *Any operative dynamic expression can be always transformed into the saturated one by applying the inaction rules in a forward or a backward direction.*

Proof. Let G be a dynamic expression, $(\alpha, \natural_l^\theta) \in EnaWait([G]_{\approx})$ and there exists $H \in [G]_{\approx} \cap OpRegDynExpr$ that contains a subexpression $\overline{(\alpha, \natural_l^\theta)^\delta},\ \delta \in \{1, \ldots, \theta - 1\}$. Then all operative dynamic expressions from $[G]_{\approx} \cap OpRegDynExpr$ contain a subexpression $\overline{(\alpha, \natural_l^\theta)^\delta}$ or $(\alpha, \natural_l^\theta)^\delta$, i.e. the (possibly overlined) enabled waiting multiaction $(\alpha, \natural_l^\theta)$ with the (non-initial) timer value superscript $\delta \leq \theta - 1$. Note that the timer value superscript δ is the same for all such structurally equivalent operative dynamic expressions. Indeed, all inaction rules, besides the first one, do not change the values of timers, but those rules just modify the overlines and underlines of dynamic expressions. The first inaction rule just sets up the timer of each overlined waiting multiaction $\overline{(\alpha, \natural_l^\theta)}$ with the initial value $\delta = \theta$, equal to the delay of that waiting multiaction, as follows: $\overline{(\alpha, \natural_l^\theta)^\theta}$. Then the remaining inaction rules can shift out the overline of that enabled waiting multiaction before setting up its timer, which results in a non-overlined enabled waiting multiaction without timer value superscript $(\alpha, \natural_l^\theta)$. Thus, for $(\alpha, \natural_l^\theta) \in EnaWait([G]_{\approx})$, it may happen that $\overline{(\alpha, \natural_l^\theta)^\theta}$ a subexpression of some $H \in [G]_{\approx} \cap OpRegDynExpr$ and $(\alpha, \natural_l^\theta)$ is a subexpression of a different $H' \in [G]_{\approx} \cap OpRegDynExpr$.

Let now G be an operative dynamic expression that is not saturated. By the arguments above, the saturation can be violated only if G contains as a subexpression at least one newly enabled waiting multiaction $(\alpha, \natural_l^\theta)$ of $[G]_{\approx}$ that is not superscribed with the timer value. By the definition of the new-enabling, there exists $H \in [G]_{\approx} \cap OpRegDynExpr$ such that $\overline{(\alpha, \natural_l^\theta)^\theta}$ is a subexpression of H. Since $G \approx H$, there is a sequence of the inaction rules applications (in a forward or a backward direction) that transforms G into H. Then the reverse sequence transforms H into G. Let us remove from that reverse sequence the following backward

application of the first inaction rule: $\overline{(\alpha,\natural_l^\theta)} \Leftarrow \overline{(\alpha,\natural_l^\theta)^\theta}$. Then such a reduced reverse sequence will turn H into $G_1 \in [G]_\approx \cap \mathit{OpRegDynExpr}$, obtained from G by replacing $(\alpha,\natural_l^\theta)$ with $(\alpha,\natural_l^\theta)^\theta$. Let us start from G_1 and apply the above procedure to the remaining not superscribed with the timer values newly enabled waiting multiactions of $[G]_\approx = [G_1]_\approx$. After repeated application of the mentioned procedure for all $n \geq 1$ non-superscribed newly enabled waiting multiactions of G, we shall get from it the saturated operative dynamic expression $G_n = \widetilde{G} \in [G]_\approx \cap \mathit{OpRegDynExpr}$. □

Thus, any dynamic expression can be always transformed into a (not necessarily unique) saturated operative one by (possibly reverse) applying the inaction rules.

Example 3.4 *Let E be from Example 3.1. We have $[\overline{E}]_\approx \cap \mathit{SaOpRegDynExpr} = \{\overline{(\{a\},\natural_1^3)^3}[](\{b\},\frac{1}{3}),$ $(\{a\},\natural_1^3)^3[]\overline{(\{b\},\frac{1}{3})}\}$. Consider the sequence of inaction rules, applied (in a forward or a backward direction) in the following transformation of a non-saturated $G \in [\overline{E}]_\approx \cap \mathit{OpRegDynExpr}$ with the non-superscribed with the timer value (unstamped) enabled waiting multiaction $(\{a\},\natural_1^3)$ into (a saturated) $H \in [\overline{E}]_\approx \cap \mathit{OpRegDynExpr}$, in which $(\{a\},\natural_1^3)$ is stamped:*

$$G = (\{a\},\natural_1^3)[]\overline{(\{b\},\tfrac{1}{3})} \approx \overline{(\{a\},\natural_1^3)[](\{b\},\tfrac{1}{3})} \approx \overline{(\{a\},\natural_1^3)}[](\{b\},\tfrac{1}{3}) \approx \overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3}) = H.$$

The reduced reverse sequence of inaction rules induces the following transformations of H that result in a saturated $G_1 = \widetilde{G} \in [\overline{E}]_\approx \cap \mathit{OpRegDynExpr}$, in which $(\{a\},\natural_1^3)$ is stamped:

$$H = \overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3}) \approx \overline{(\{a\},\natural_1^3)^3[](\{b\},\tfrac{1}{3})} \approx (\{a\},\natural_1^3)^3[]\overline{(\{b\},\tfrac{1}{3})} = G_1 = \widetilde{G}.$$

Let G be a saturated operative dynamic expression. Then $\circlearrowleft G$ is written for the *timer decrement* operator $\circlearrowleft$, applied to G. It denotes a saturated operative dynamic expression, obtained from G via decrementing by one time unit all greater than 1 values of the timers associated with all (if any) stamped waiting multiactions from the syntax of G. Thus, each such stamped waiting multiaction changes its timer value from δ in G to $\max\{1,\delta-1\}$ in $\circlearrowleft G$, where $\delta \in I\!N_{\geq 1}$. Formally, the timer decrement operator affects the (possibly overlined or underlined) stamped waiting multiactions being the subexpressions of G as follows. The overlined stamped waiting multiaction $\overline{(\alpha,\natural_l^\theta)^\delta}$ is replaced with $\overline{(\alpha,\natural_l^\theta)^{\max\{1,\delta-1\}}}$. The underlined stamped waiting multiaction $\underline{(\alpha,\natural_l^\theta)^\delta}$ is replaced with $\underline{(\alpha,\natural_l^\theta)^{\max\{1,\delta-1\}}}$. The stamped waiting multiaction without overline or underline $(\alpha,\natural_l^\theta)^\delta$ is replaced with $(\alpha,\natural_l^\theta)^{\max\{1,\delta-1\}}$.

Note that when $\delta = 1$, we have $\max\{1,\delta-1\} = \max\{1,0\} = 1$, hence, the timer value $\delta = 1$ may remain unchanged for a stamped waiting multiaction that is not executed by some reason at the next time moment, but stays stamped. For example, that stamped waiting multiaction may be affected by restriction. If the timer values cannot be decremented with a time tick for all stamped waiting multiactions (if any) from G then $\circlearrowleft G = G$ and we obtain so-called *empty loop* transition that will be formally defined later.

Observe that the timer decrement operator keeps stamping of the waiting multiactions, since it may only decrease their timer values, so that the stamped waiting multiactions stay stamped (with their timer values, possibly decremented by one).

Example 3.5 *Let E be from Example 3.1. We have $\mathit{Ena}([\overline{E}]_\approx) = \{(\{a\},\natural_1^3),(\{b\},\frac{1}{3})\}$ and $\mathit{Ena}([\overline{E}]_\approx) \cap \mathcal{WL} = \{(\{a\},\natural_1^3)\}$. The following one time unit timer decrements are possible from the saturated operative dynamic expressions belonging to $[\overline{E}]_\approx$:*

$$\circlearrowleft(\overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3})) = \overline{(\{a\},\natural_1^3)^2}[](\{b\},\tfrac{1}{3}), \quad \circlearrowleft((\{a\},\natural_1^3)^3[]\overline{(\{b\},\tfrac{1}{3})}) = (\{a\},\natural_1^3)^2[]\overline{(\{b\},\tfrac{1}{3})}.$$

Let G be a dynamic expression. Then $I_G : \mathcal{WL}(G) \to I\!N_{\geq 1}$ is the *timer valuation function* of the waiting multiactions of G, defined as follows. For $(\alpha,\natural_l^\theta) \in \mathcal{WL}(G)$, let $I_G((\alpha,\natural_l^\theta)) = \delta \in \{1,\ldots,\theta\}$, if $\exists H \in [G]_\approx \cap$ $\mathit{SatOpRegDynExpr}$ $\overline{(\alpha,\natural_l^\theta)^\delta}$ or $\underline{(\alpha,\natural_l^\theta)^\delta}$ or $(\alpha,\natural_l^\theta)^\delta$ is a subexpression of H. Otherwise, we let $I_G((\alpha,\natural_l^\theta)) = \infty$, where '$\infty$' denotes the undefined value (infinite time till the activity execution). The definition is correct by the argumentation from the proof of Proposition 3.1. Indeed, for each waiting multiaction of G, its timer value superscript (if any) is the same for every $H \in [G]_\approx \cap \mathit{SatOpRegDynExpr}$, in which that waiting multiaction, possibly being superscribed with the value of its timer and overlined or underlined, is a subexpression. Note that we may have $I_G((\alpha,\natural_l^\theta)) < \infty$ for $(\alpha,\natural_l^\theta) \in \mathcal{WL}(G) \setminus \mathit{EnaWait}([G]_\approx)$, i.e. the non-enabled waiting multiactions of $[G]_\approx$ may have finite timer valuations. The latter is allowed only in the "incomplete" specifications by the compositionality reasons. It is assumed that all such non-enabled waiting multiactions have infinite timer values in the "complete" specification, hence, all and only enabled waiting multiactions have finite timer values there.

Let $G \in \mathit{SatOpRegDynExpr}$. Then for all $(\alpha,\natural_l^\theta) \in \mathcal{WL}(G)$, we have $I_{\circlearrowleft G}((\alpha,\natural_l^\theta)) = \max\{1, I_G((\alpha,\natural_l^\theta))-1\}$.

3.2 Action and empty move rules

The action rules are applied when some activities are executed. With these rules we capture the prioritization among different types of multiactions. We also have the empty move rule which is used to capture a delay of one discrete time unit when no immediate or waiting multiactions are executable. In this case, the empty multiset of activities is executed. The action and empty move rules will be used later to determine all multisets of activities which can be executed from the structural equivalence class of every dynamic expression (i.e. from the state of the corresponding process). This information together with that about probabilities or delays and weights of the activities to be executed from the current process state will be used to calculate the probabilities of such executions.

The action rules with stochastic (immediate or waiting, respectively) multiactions describe dynamic expression transformations in the form of $G \stackrel{\Gamma}{\to} \widetilde{G}$ ($G \stackrel{I}{\to} \widetilde{G}$ or $G \stackrel{W}{\to} \widetilde{G}$, respectively) due to execution of non-empty multisets Γ of stochastic (I of immediate or W of waiting, respectively) multiactions. The rules represent possible state changes of the specified processes when some non-empty multisets of stochastic (immediate or waiting, respectively) multiactions are executed. As we shall see, the application of an action rule with stochastic (immediate or waiting, respectively) multiactions to a dynamic expression leads in the corresponding LDTSDPN to a discrete time tick at which some stochastic or waiting transitions fire (or to the instantaneous firing of some immediate transitions) and possible change of the current marking. The current marking stays unchanged only if there is a self-loop produced by the iterative execution of a non-empty multiset, which must be one-element, i.e. a single stochastic (immediate or waiting, respectively) multiaction. The reason is the regularity requirement that allows no concurrency at the highest level of the second argument of iteration.

The empty move rule (applicable only when no immediate or waiting multiactions can be executed from the current state) describes dynamic expression transformations in the form of $G \stackrel{\emptyset}{\to} \circlearrowleft G$, called the *empty moves*, due to execution of the empty multiset of activities at a discrete time tick. When no timer values are decremented within G with the empty multiset execution at the next moment, we have $\circlearrowleft G = G$. For example, this is the case if G contains no stamped waiting multiactions, or all their timers have decreased to value 1 while those stamped waiting multiactions are either affected by restriction or not overlined). In such a case, the empty move from G is in the form of $G \stackrel{\emptyset}{\to} G$, called the *empty loop*. As we shall see, the application of the empty move rule to a dynamic expression leads to a discrete time tick in the corresponding LDTSDPN at which no transitions fire and the current marking is not changed, but the timer values of the waiting transitions enabled at the marking (if any) are decremented by one. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay, but no notion of time exists in PBC. The PBC rule $G \stackrel{\emptyset}{\to} G$ from [41, 40] in our setting would correspond to the rule $G \Rightarrow G$ that describes staying in the current state when no time elapses. Since we do not need the latter rule to transform dynamic expressions into operative ones and it can even destroy the definition of operative expressions, we do not introduce it in dtsdPBC.

Thus, an application of every action rule with stochastic or waiting multiactions or the empty move rule requires one discrete time unit delay, i.e. the execution of a (possibly empty) multiset of stochastic or (non-empty) multiset of waiting multiactions leading to the dynamic expression transformation described by the rule is accomplished instantly after one time unit. An application of every action rule with immediate multiactions does not take any time, i.e. the execution of a (non-empty) multiset of immediate multiactions is accomplished instantly at the current moment of time.

Note that expressions of dtsdPBC can contain identical activities. To avoid technical difficulties, such as the proper calculation of the state change probabilities for multiple transitions, we can always enumerate coinciding activities from left to right in the syntax of expressions. The new activities, resulted from synchronization will be annotated with concatenation of numberings of the activities they come from, hence, the numbering should have a tree structure to reflect the effect of multiple synchronizations. We now define the numbering which encodes a binary tree with the leaves labeled by natural numbers.

Definition 3.4 *The* numbering *of expressions is defined as* $\iota ::= n \mid (\iota)(\iota)$, *where* $n \in I\!N$.

Let Num denote the set of *all numberings* of expressions.

Example 3.6 *The numbering* 1 *encodes the binary tree depicted in Figure 1(a) with the root labeled by* 1. *The numbering* (1)(2) *corresponds to the binary tree depicted in Figure 1(b) without internal nodes and with two leaves labeled by* 1 *and* 2. *The numbering* (1)((2)(3)) *represents the binary tree depicted in Figure 1(c) with one internal node, which is the root for the subtree* (2)(3), *and three leaves labeled by* 1, 2 *and* 3.

The new activities resulting from synchronizations in different orders should be considered up to permutation of their numbering. In this way, we shall recognize different instances of the same activity. If we compare the

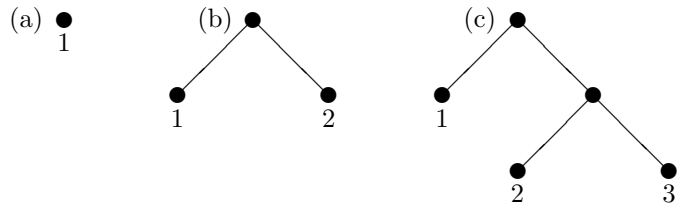

Figure 1: The binary trees encoded with the numberings 1, (1)(2) and (1)((2)(3))

contents of different numberings, i.e. the sets of natural numbers in them, we shall be able to identify the mentioned instances.

The *content* of a numbering $\iota \in Num$ is

$$Cont(\iota) = \begin{cases} \{\iota\}, & \iota \in \mathbb{N}; \\ Cont(\iota_1) \cup Cont(\iota_2), & \iota = (\iota_1)(\iota_2). \end{cases}$$

After the enumeration, the multisets of activities from the expressions will become the proper sets. In the following, we suppose that the identical activities are enumerated when needed to avoid ambiguity. This enumeration is considered to be implicit.

Definition 3.5 *Let $G \in OpRegDynExpr$. We now define the* set of all non-empty multisets of activities which can be potentially executed from G, *denoted by $Can(G)$. Let $(\alpha, \kappa) \in \mathcal{SDL}$, $E, F \in RegStatExpr$, $H \in OpRegDynExpr$ and $a \in Act$.*

1. *If $final(G)$ then $Can(G) = \emptyset$.*
2. *If $G = \overline{(\alpha, \kappa)^\delta}$ and $\kappa = \natural_l^\theta$, $\theta \in \mathbb{N}_{\geq 2}$, $l \in \mathbb{R}_{>0}$, $\delta \in \{2, \ldots, \theta\}$, then $Can(G) = \emptyset$.*
3. *If $G = \overline{(\alpha, \kappa)}$ and $\kappa \in (0;1)$ or $\kappa = \natural_l^0$, $l \in \mathbb{R}_{>0}$, then $Can(G) = \{\{(\alpha, \kappa)\}\}$.*
4. *If $G = \overline{(\alpha, \kappa)^1}$ and $\kappa = \natural_l^\theta$, $\theta \in \mathbb{N}_{\geq 1}$, $l \in \mathbb{R}_{>0}$, then $Can(G) = \{\{(\alpha, \kappa)\}\}$.*
5. *If $\Upsilon \in Can(G)$ then $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$, $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$, $f(\Upsilon) \in Can(G[f])$, $\Upsilon \in Can(G \text{ rs } a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$.*
6. *If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$.*
7. *If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different activities such that $a \in \alpha$, $\hat{a} \in \beta$, then*
   (a) *$\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\} \in Can(G \text{ sy } a)$ if $\kappa, \lambda \in (0;1)$;*
   (b) *$\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \natural_{l+m}^\theta)\} \in Can(G \text{ sy } a)$ if $\kappa = \natural_l^\theta$, $\lambda = \natural_m^\theta$, $\theta \in \mathbb{N}$, $l, m \in \mathbb{R}_{>0}$.*

   When we synchronize the same multiset of activities in different orders, we obtain several activities with the same multiaction and probability or delay and weight parts, but with different numberings having the same content. Then we only consider a single one of the resulting activities to avoid introducing redundant ones.

   For example, the synchronization of stochastic multiactions $(\alpha, \rho)_1$ and $(\beta, \chi)_2$ in different orders generates the activities $(\alpha \oplus_a \beta, \rho \cdot \chi)_{(1)(2)}$ and $(\beta \oplus_a \alpha, \chi \cdot \rho)_{(2)(1)}$. Similarly, the synchronization of deterministic multiactions $(\alpha, \natural_l^\theta)_1$ and $(\beta, \natural_m^\theta)_2$ in different orders generates the activities $(\alpha \oplus_a \beta, \natural_{l+m}^\theta)_{(1)(2)}$ and $(\beta \oplus_a \alpha, \natural_{m+l}^\theta)_{(2)(1)}$. Since $Cont((1)(2)) = \{1, 2\} = Cont((2)(1))$, in both cases, only the first activity (or, symmetrically, the second one) resulting from synchronization will appear in a multiset from $Can(G \text{ sy } a)$.

Note that if $\Upsilon \in Can(G)$ then by definition of $Can(G)$, $\forall \Xi \subseteq \Upsilon$, $\Xi \neq \emptyset$, we have $\Xi \in Can(G)$.

Let $G \in OpRegDynExpr$ and $Can(G) \neq \emptyset$. Obviously, if there are only stochastic (immediate or waiting, respectively) multiactions in the multisets from $Can(G)$ then these stochastic (immediate or waiting, respectively) multiactions can be executed from G. Otherwise, besides stochastic ones, there are also deterministic (immediate and/or waiting) multiactions in the multisets from $Can(G)$. By the note above, there are non-empty multisets of deterministic multiactions in $Can(G)$ as well, i.e. $\exists \Upsilon \in Can(G)$ $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{DL}} \setminus \{\emptyset\}$. In this case, no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty multisets of stochastic

multiactions, since deterministic multiactions have a priority over stochastic ones, and should be executed first. Further, if there are no stochastic, but both waiting and immediate multiactions in the multisets from $Can(G)$, then, analogously, no waiting multiactions can be executed from G, since immediate multiactions have a priority over waiting ones (besides that over stochastic ones).

When there are only waiting and, possibly, stochastic multiactions in the multisets from $Can(G)$ then, from above, only waiting ones can be executed from G. Then just *maximal* non-empty multisets of waiting multiactions can be executed from G, since all non-conflicting waiting multiactions cannot wait anymore and they should occur at the next time moment with probability 1. The next definition formalizes these requirements.

Definition 3.6 *Let $G \in OpRegDynExpr$. The* set of all non-empty multisets of activities which can be executed from G *is*

$$Now(G) = \begin{cases} Can(G) \cap \mathbb{N}_{fin}^{\mathcal{IL}}, & Can(G) \cap \mathbb{N}_{fin}^{\mathcal{IL}} \neq \emptyset; \\ \{W \in Can(G) \cap \mathbb{N}_{fin}^{\mathcal{WL}} \mid \forall V \in Can(G) \cap \mathbb{N}_{fin}^{\mathcal{WL}}\ W \subseteq V \ \Rightarrow\ V = W\}, & (Can(G) \cap \mathbb{N}_{fin}^{\mathcal{IL}} = \emptyset) \wedge \\ & (Can(G) \cap \mathbb{N}_{fin}^{\mathcal{WL}} \neq \emptyset); \\ Can(G), & otherwise. \end{cases}$$

Consider an operative dynamic expression $G \in OpRegDynExpr$. The expression G is *s-tangible (stochastically tangible)*, denoted by $stang(G)$, if $Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}$. In particular, we have $stang(G)$, if $Now(G) = \emptyset$. The expression G is *w-tangible (waitingly tangible)*, denoted by $wtang(G)$, if $\emptyset \neq Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{WL}} \setminus \{\emptyset\}$. The expression G is *tangible*, denoted by $tang(G)$, if $stang(G)$ or $wtang(G)$, i.e. $Now(G) \subseteq (\mathbb{N}_{fin}^{\mathcal{SL}} \cup \mathbb{N}_{fin}^{\mathcal{WL}}) \setminus \{\emptyset\}$. Again, we particularly have $tang(G)$, if $Now(G) = \emptyset$. Otherwise, the expression G is *vanishing*, denoted by $vanish(G)$, and in this case $\emptyset \neq Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. Note that the operative dynamic expressions from $[G]_{\approx}$ may have different types in general. The following example demonstrates two operative dynamic expressions H and H' with $H \approx H'$, such that $vanish(H)$, but $stang(H')$.

Example 3.7 *Let $G = (\overline{(\{a\}, \natural_1^0)}[](\{b\}, \natural_2^0))\|\overline{(\{c\}, \frac{1}{2})}$ and $G' = ((\{a\}, \natural_1^0)[]\overline{(\{b\}, \natural_2^0)})\|\overline{(\{c\}, \frac{1}{2})}$. Then $G \approx G'$, since $G \Leftarrow G'' \Rightarrow G'$ for $G'' = \overline{((\{a\}, \natural_1^0)[](\{b\}, \natural_2^0))\|(\{c\}, \frac{1}{2})}$, but $Can(G) = \{\{(\{a\}, \natural_1^0)\}, \{(\{c\}, \frac{1}{2})\}, \{(\{a\}, \natural_1^0), (\{c\}, \frac{1}{2})\}\}$, $Can(G') = \{\{(\{b\}, \natural_2^0)\}, \{(\{c\}, \frac{1}{2})\}, \{(\{b\}, \natural_2^0), (\{c\}, \frac{1}{2})\}\}$ and $Now(G) = \{\{(\{a\}, \natural_1^0)\}\}$, $Now(G') = \{\{(\{b\}, \natural_2^0)\}\}$. Clearly, we have $vanish(G)$ and $vanish(G')$. The executions like that of $\{(\{c\}, \frac{1}{2})\}$ (and all multisets including it) from G and G' must be disabled using preconditions in the action rules, since immediate multiactions have a priority over stochastic ones, hence, the former are always executed first.*

Let $H = \overline{(\{a\}, \natural_1^0)[](\{b\}, \frac{1}{2})}$ and $H' = (\{a\}, \natural_1^0)[]\overline{(\{b\}, \frac{1}{2})}$. Then $H \approx H'$, since $H \Leftarrow H'' \Rightarrow H'$ for $H'' = \overline{(\{a\}, \natural_1^0)[](\{b\}, \frac{1}{2})}$, but $Can(H) = Now(H) = \{\{(\{a\}, \natural_1^0)\}\}$ and $Can(H') = Now(H') = \{\{(\{b\}, \frac{1}{2})\}\}$. We have $vanish(H)$, but $stang(H')$. To make the action rules correct under structural equivalence, the executions like that of $\{(\{b\}, \frac{1}{2})\}$ from H' must be disabled using preconditions in the action rules, since immediate multiactions have a priority over stochastic ones, hence, the choices between them are always resolved in favour of the former.

Let $G \in RegDynExpr$. We write $stang([G]_{\approx})$, if $\forall H \in [G]_{\approx} \cap OpRegDynExpr\ stang(H)$. We write $wtang([G]_{\approx})$, if $\exists H \in [G]_{\approx} \cap OpRegDynExpr\ wtang(H)$ and $\forall H' \in [G]_{\approx} \cap OpRegDynExpr\ tang(H')$. We write $tang([G]_{\approx})$, if $stang([G]_{\approx})$ or $wtang([G]_{\approx})$. Otherwise, we write $vanish([G]_{\approx})$, and in this case $\exists H \in [G]_{\approx} \cap OpRegDynExpr\ vanish(H)$.

In Table 3, we define the action and empty move rules. In the table, $(\alpha, \rho), (\beta, \chi) \in \mathcal{SL}$, $(\alpha, \natural_l^0), (\beta, \natural_m^0) \in \mathcal{IL}$ and $(\alpha, \natural_l^\theta), (\beta, \natural_m^\theta) \in \mathcal{WL}$. Further, $E, F \in RegStatExpr$, $G, H \in SatOpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, $\Gamma, \Delta \in \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}$, $\Gamma' \in \mathbb{N}_{fin}^{\mathcal{SL}}$, $I, J \in \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$, $I' \in \mathbb{N}_{fin}^{\mathcal{IL}}$, $V, W \in \mathbb{N}_{fin}^{\mathcal{WL}} \setminus \{\emptyset\}$, $V' \in \mathbb{N}_{fin}^{\mathcal{WL}}$ and $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SDL}} \setminus \{\emptyset\}$. We denote $\Upsilon_a = \{(\alpha, \kappa) \in \Upsilon \mid (a \in \alpha) \vee (\hat{a} \in \alpha)\}$.

We use the following abbreviations in the names of the rules from the table: "**E**" for "**E**mpty move", "**B**" for "**B**asis case", "**S**" for "**S**equence", "**C**" for "**C**hoice", "**P**" for "**P**arallel", "**L**" for "re**L**abeling", "**R**" for "**R**estriction", "**I**" for "**I**teraton" and "**Sy**" for "**Sy**nchronization". The first rule in the table is the empty move rule **E**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge the rules with stochastic, immediate ans waiting multiactions in one rule for some operation then we get the coupled action rules. In such cases, the names of the action rules with stochastic multiactions have a suffix '**s**', those with immediate multiactions have a suffix '**i**', and those with waiting multiactions have a suffix '**w**'. To make presentation more compact, the action rules with double conclusion are combined from two distinct action rules with the same premises.

Almost all the rules in Table 3 (excepting **E**, **Bw**, **P2s**, **P2i**, **P2w**, **R**, **Sy2s**, **Sy2i** and **Sy2w**) resemble those of gsPBC, but the former correspond to execution of multisets of activities, not of single activities, as in

Table 3: Action and empty move rules

$$\mathbf{E}\ \frac{stang([G]_{\approx})}{G \xrightarrow{\emptyset} \circlearrowleft G} \qquad \mathbf{Bs}\ \frac{}{\overline{(\alpha,\rho)}^{\{(\alpha,\rho)\}} \underline{(\alpha,\rho)}} \qquad \mathbf{Bi}\ \frac{}{\overline{(\alpha,\natural_l^0)}^{\{(\alpha,\natural_l^0)\}} \underline{(\alpha,\natural_l^0)}} \qquad \mathbf{Bw}\ \frac{}{\overline{(\alpha,\natural_l^\theta)^1}^{\{(\alpha,\natural_l^\theta)\}} \underline{(\alpha,\natural_l^\theta)}}$$

$$\mathbf{S}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G;E \xrightarrow{\Upsilon} \widetilde{G};E,\ E;G \xrightarrow{\Upsilon} E;\widetilde{G}} \qquad \mathbf{Cs}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge stang([\overline{E}]_{\approx}))}{G[]E \xrightarrow{\Gamma} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{\Gamma} \downarrow E[]\widetilde{G}}$$

$$\mathbf{Ci}\ \frac{G \xrightarrow{I} \widetilde{G}}{G[]E \xrightarrow{I} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{I} \downarrow E[]\widetilde{G}} \qquad \mathbf{Cw}\ \frac{G \xrightarrow{V} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang([\overline{E}]_{\approx}))}{G[]E \xrightarrow{V} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{V} \downarrow E[]\widetilde{G}}$$

$$\mathbf{P1s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ stang([H]_{\approx})}{G\|H \xrightarrow{\Gamma} \widetilde{G}\|\circlearrowleft H,\ H\|G \xrightarrow{\Gamma} \circlearrowleft H\|\widetilde{G}} \qquad \mathbf{P1i}\ \frac{G \xrightarrow{I} \widetilde{G}}{G\|H \xrightarrow{I} \widetilde{G}\|H,\ H\|G \xrightarrow{I} H\|\widetilde{G}}$$

$$\mathbf{P1w}\ \frac{G \xrightarrow{V} \widetilde{G},\ stang([H]_{\approx})}{G\|H \xrightarrow{V} \widetilde{G}\|\circlearrowleft H,\ H\|G \xrightarrow{V} \circlearrowleft H\|\widetilde{G}} \qquad \mathbf{P2s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ H \xrightarrow{\Delta} \widetilde{H}}{G\|H \xrightarrow{\Gamma+\Delta} \widetilde{G}\|\widetilde{H}}$$

$$\mathbf{P2i}\ \frac{G \xrightarrow{I} \widetilde{G},\ H \xrightarrow{J} \widetilde{H}}{G\|H \xrightarrow{I+J} \widetilde{G}\|\widetilde{H}} \qquad \mathbf{P2w}\ \frac{G \xrightarrow{V} \widetilde{G},\ H \xrightarrow{W} \widetilde{H}}{G\|H \xrightarrow{V+W} \widetilde{G}\|\widetilde{H}}$$

$$\mathbf{L}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \widetilde{G}[f]} \qquad \mathbf{R}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G\ \mathsf{rs}\ a \xrightarrow{\Upsilon-\Upsilon_a} \widetilde{G}\ \mathsf{rs}\ a}$$

$$\mathbf{I1}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{[G*E*F] \xrightarrow{\Upsilon} [\widetilde{G}*E*F]} \qquad \mathbf{I2s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge stang([\overline{F}]_{\approx}))}{[E*G*F] \xrightarrow{\Gamma} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{\Gamma} [E*\downarrow F*\widetilde{G}]}$$

$$\mathbf{I2i}\ \frac{G \xrightarrow{I} \widetilde{G}}{[E*G*F] \xrightarrow{I} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{I} [E*\downarrow F*\widetilde{G}]} \qquad \mathbf{I2w}\ \frac{G \xrightarrow{V} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang([\overline{F}]_{\approx}))}{[E*G*F] \xrightarrow{V} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{V} [E*\downarrow F*\widetilde{G}]}$$

$$\mathbf{Sy1}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G\ \mathsf{sy}\ a \xrightarrow{\Upsilon} \widetilde{G}\ \mathsf{sy}\ a} \qquad \mathbf{Sy2s}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha,\rho)\}+\{(\beta,\chi)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha \oplus_a \beta, \rho \cdot \chi)\}} \widetilde{G}\ \mathsf{sy}\ a}$$

$$\mathbf{Sy2i}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha,\natural_l^0)\}+\{(\beta,\natural_m^0)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha \oplus_a \beta, \natural_{l+m}^0)\}} \widetilde{G}\ \mathsf{sy}\ a} \qquad \mathbf{Sy2w}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{V'+\{(\alpha,\natural_l^\theta)\}+\{(\beta,\natural_m^\theta)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{V'+\{(\alpha \oplus_a \beta, \natural_{l+m}^\theta)\}} \widetilde{G}\ \mathsf{sy}\ a}$$

the latter, and our rules have simpler preconditions (if any), since all immediate multiactions in dtsdPBC have the same priority level, unlike those of gsPBC.

The preconditions in rules **E**, **Cs**, **P1s**, and **I2s** are needed to ensure that (possibly empty) multisets of stochastic multiactions are executed only from *s-tangible* saturated operative dynamic expressions, such that all dynamic expressions structurally equivalent to them are s-tangible as well. For example, assuming that $stang([G]_{\approx})$ in rule **Cs**, if $init(G)$ then $G \approx \overline{F}$ for some static expression F and $G[]E \approx \overline{F}[]E \approx \overline{F[]E} \approx F[]\overline{E}$. Hence, it should be guaranteed that $stang(F[]\overline{E}]_{\approx})$, which holds iff $stang([\overline{E}]_{\approx})$. The case $E[]G$ is treated similarly. Assuming that $stang([G]_{\approx})$ in rule **P1s**, it should be guaranteed that $stang([G\|H]_{\approx})$ and $stang([H\|G]_{\approx})$, which holds iff $stang([H]_{\approx})$. The precondition in rule **I2s** is analogous to that in rule **Cs**.

Analogously, the preconditions in rules **Cw**, and **I2w** are needed to ensure that non-empty multisets of waiting multiactions are executed only from *w-tangible* saturated operative dynamic expressions, such that all dynamic expressions structurally equivalent to them are tangible. This requirement (about tangible expressions) means that only (possibly empty) multisets of stochastic multiactions or non-empty multisets of waiting multiactions, and no immediate multiactions, can be executed from the subprocess that is composed *alternatively (in choice)* with the subprocess G. Hence, the multiset W of waiting multiactions, executed from G, can also be executed from the composition of G and that alternative subprocess, since immediate multiactions cannot occur from the latter. Otherwise, it would prevent the execution of W from G in the composite process, by disregarding the alternative choice of the branch specified by G, due to the zero delays and priority (captured by all action rules) of immediate multiactions over all other multiaction types.

The precondition in rule **P1w** is an exception from the above. It also ensures that non-empty multisets of waiting multiactions are executed only from *w-tangible* saturated operative dynamic expressions, such that all dynamic expressions structurally equivalent to them are tangible, but all the expressions structurally equivalent to H specifying parallel with G subprocess should be s-tangible. This stricter requirement (about s-tangible, instead of just tangible, expressions) means that only (possibly empty) multisets of stochastic multiactions, and no immediate or waiting multiactions, can be executed from the subprocess H that is composed *concurrently (in parallel)* with the subprocess G. Hence, the multiset W of waiting multiactions, executed from G, is also a maximal (by the inclusion relation) multiset that can be executed from the parallel composition of G and H.

The reason is that only the timers decrement by one time unit (by applying rule **E**) is actually possible in H while executing W from G, due to priority (captured by all action rules) of waiting multiactions over stochastic ones. Thus, taking the rule precondition $stang([H]_{\approx})$ instead of $tang([H]_{\approx})$ preserves maximality of the steps consisting of waiting multiactions while applying parallel composition.

In rules **P1s** and **P1w**, the timer value decrementing by one $\circlearrowleft H$, applied to the s-tangible saturated operative dynamic expression H that is composed in parallel with G, from which stochastic multiactions are executed at the next time tick, is used to maintain the time progress uniformity in the composite expression. Although rules **P1s** and **P1w** can be merged, we have not done it, aiming to emphasize the exceptional precondition in rule **P1w**.

In rules **Cs**, **Ci** and **Cw**, the timer values discarding $\lfloor E$, applied to the static expression E that is composed in choice with G, from which activities are executed, signifies that the timer values of the non-chosen subexpression (branch) become irrelevant in the composite expression and thus may be removed. Analogously, in rules **I2s**, **I2i** and **I2w**, the timer values discarding $\lfloor F$ is applied to the static expression F that is an alternative to G, from which activities are executed, since the choice is always made between the body and termination subexpressions of the composite iteration expression (between the second and third arguments of iteration).

Rule **E** corresponds to one discrete time unit delay (passage of one unit of time) while executing no activities and therefore it has no analogues among the rules of gsPBC that adapts the continuous time model. Rule **E** is a *global* one, i.e. it is applied only to the whole (topmost level of) expressions, rather than to their parts. The reason is that all other action rules describe dynamic expressions transformations due to execution of *non-empty* multisets of activities. Hence, the actionless time move described by rule **E** cannot "penetrate" with action rules through the expressions structure. This guarantees that time progresses uniformly in all their subexpressions.

Rule **Bw** differs from the more standard ones **Bs** and **Bi** that both resemble rule **B** in gsPBC. The reason is that in **Bw**, the overlined waiting multiaction has an extra superscript '1', indicating that one time unit is remained until the multiaction's execution (RTE equals one) that should follow in the next moment.

Rules **P2s**, **P2i** and **P2w** have no similar rules in gsPBC, since interleaving semantics of the algebra allows no simultaneous execution of activities. On the other hand, **P2s**, **P2i** and **P2w** have in PBC the analogous rule **PAR** that is used to construct step semantics of the calculus, but the former two rules correspond to execution of multisets of activities, unlike that of multisets of multiactions in the latter rule. Rules **P2s**, **P2i** and **P2w** cannot be merged, since otherwise simultaneous execution of different types of multiactions would be allowed.

Rule **R** differs from the corresponding restriction rule in gsPBC, since it respects the maximality of the steps of waiting multiactions that is maintained by all action rules, in particular, by **P1w** and **P2w**. In case Υ is a multiset of waiting multiactions, no proper subset of Υ (such as that consisting of the activities not affected by restriction over a) can be executed from G. Since we prefer to have the same restriction rule for stochastic, immediate ans waiting multiactions, rule **R** should explicitly allow execution of the subset $\Upsilon - \Upsilon_a$ from G rs a.

Rules **Sy2s**, **Sy2i** and **Sy2w** differ from the corresponding synchronization rules in gsPBC, since the probability or the weight of synchronization in the former rules and the rate or the weight of synchronization in the latter rules are calculated in two distinct ways. Rules **Sy2i** and **Sy2w** cannot be merged, since otherwise synchronous execution of immediate and waiting multiactions would be allowed.

Rule **Sy2s** establishes that the synchronization of two stochastic multiactions is made by taking the product of their probabilities, since we are considering that both must occur for the synchronization to happen, so this corresponds, in some sense, to the probability of the independent event intersection, but the real situation is more complex, since these stochastic multiactions can also be executed in parallel. Nevertheless, when scoping (the combined operation consisting of synchronization followed by restriction over the same action [40]) is applied over a parallel execution, we get as final result just the simple product of the probabilities, since no normalization is needed there. Multiplication is an associative and commutative binary operation that is distributive over addition, i.e. it fulfills all practical conditions imposed on the synchronization operator in [163]. Further, if both arguments of multiplication are from $(0;1)$ then the result belongs to the same interval, hence, multiplication naturally maintains probabilistic compositionality in our model. Our approach is similar to the multiplication of rates of the synchronized actions in MTIPP [162] in the case when the rates are less than 1. Moreover, for the probabilities ρ and χ of two stochastic multiactions to be synchronized we have $\rho \cdot \chi < \min\{\rho, \chi\}$, i.e. multiplication meets the performance requirement stating that the probability of the resulting synchronized stochastic multiaction should be less than the probabilities of the two ones to be synchronized. While performance evaluation, it is usually supposed that the execution of two components together require more system resources and time than the execution of each single one. This resembles the *bounded capacity* assumption from [163]. Thus, multiplication is easy to handle with and it satisfies the algebraic, probabilistic, time and performance requirements. Therefore, we have chosen the product of the probabilities for the synchronization. See also [69, 68] for a discussion about binary operations producing the rates of synchronization in the continuous time setting.

In rules **Sy2i** and **Sy2w**, we sum the weights of two synchronized immediate (waiting, respectively) multi-

actions, since the weights can be interpreted as the rewards [260], thus, we collect the rewards. Moreover, we express that the synchronized execution of immediate (waiting) multiactions has more importance than that of every single one. The weights of immediate and waiting (i.e. deterministic) multiactions can also be seen as bonus rewards associated with transitions [32]. The rewards are summed during synchronized execution of immediate (waiting) multiactions, since in that case all the synchronized activities can be seen as participated in the execution. We prefer to collect more rewards, thus, the transitions providing greater rewards will have a preference and they will be executed with a greater probability. In particular, since execution of immediate multiactions takes no time, we prefer to collect in a step (parallel execution) as many synchronized immediate multiactions as possible to get more significant progress in behaviour. Under behavioural progress we understand an advance in executing activities, which does not always imply a progress in time, as in the case when the activities are immediate multiactions. This aspect will be used later, while evaluating performance via analysis of the embedded discrete time Markov chains (EDTMCs) of expressions. Since every state change in EDTMC takes one unit of (its local) time, greater advance in operation of the EDTMC allows one to calculate quicker many performance indices. As for waiting multiactions, only the maximal multisets of them, executable from a state, occur with a time tick. The reason is that each waiting multiaction has a probability 1 to occur in the next moment, when the remaining time of its timer (RTE) equals one and there exist no conflicting waiting multiactions. Hence, all waiting multiactions with the RTE being one that are executable together from a state must participate in a step from that state. Since there may exist different such maximal multisets of waiting multiactions, a probabilistic choice among all possible steps is made, imposed by the weights of those multiactions. Thus, the steps of waiting multiactions always produce maximal overall weights, but they are mainly used to calculate the probabilities of alternative maximal steps rather than the cumulative bonus rewards.

We do not have self-synchronization, i.e. synchronization of an activity with itself, since all the (enumerated) activities executed together are considered to be different. This allows us to avoid rather cumbersome and unexpected behaviour, as well as many technical difficulties [40].

Notice that the timers of all enabled waiting multiactions that lose their enabledness when a state change occurs become inactive (turned off) and their values become irrelevant while the timers of all those preserving their enabledness continue running with their stored values decreased by one. Hence, we adapt the *enabling memory* policy [214, 1, 15, 16] when the process states are changed and the enabledness of deterministic multiactions is possibly modified (remember that immediate multiactions may be seen as those with the timers displaying a single value 0, so we do not need to store their values). Then the timer values of waiting multiactions are taken as the enabling memory variables.

Similar to [183], we are mainly interested in the dynamic expressions, inferred by applying the inaction rules (also in the reverse direction) and action rules from the overlined static expressions, such that no stamped (i.e. superscribed with the timer values) waiting multiaction is a subexpression of them. The reason is to ensure that time proceeds uniformly and only enabled waiting multiactions are stamped. We call such dynamic expressions reachable, by analogy with the reachable states of LDTSDPNs, to be presented later. Formally, a dynamic expression G is *reachable*, if there exists a static expression E without timer value superscripts, such that $\overline{E} \approx G$ or $\overline{E} \approx G_0 \stackrel{\Upsilon_1}{\to} H_1 \approx G_1 \stackrel{\Upsilon_2}{\to} \ldots \stackrel{\Upsilon_n}{\to} H_n \approx G$ for some $\Upsilon_1, \ldots, \Upsilon_n \in I\!N_{fin}^{\mathcal{SDL}}$.

Therefore, we consider a dynamic expression $G = \overline{(\{a\}, \natural_1^2)^1}[](\{b\}, \natural_2^3)^1$ as "illegal" and that $H = \overline{(\{a\}, \natural_1^2)^1}[](\{b\}, \natural_2^3)^2$ as "legal", since the latter is obtained from the overlined static expression without timer value superscripts $\overline{E} = \overline{(\{a\}, \natural_1^2)[](\{b\}, \natural_2^3)}$ after one time tick. On the other hand, G is "illegal" only when it is intended to specify a complete process, but it may become "legal" as a part of some complete specification, like G sr a, since after two time ticks from $\overline{E \text{ sr } a}$, the timer values cannot be decreased further when the value 1 is approached. Thus, we should allow the dynamic expressions like G, by assuming that they are incomplete specifications, to be further composed. Further, a dynamic expression $G = \overline{(\{a\}, \frac{1}{2})}; (\{b\}, \natural_1^2)^1$ is "illegal", since the waiting multiaction $(\{b\}, \natural_1^2)$ is not enabled in $[G]_{\approx}$ and its timer cannot start before the stochastic multiaction $(\{a\}, \frac{1}{2})$ is executed. Enabledness of the stamped waiting multiactions is considered in the next proposition.

Proposition 3.2 *Let G be a reachable dynamic expression. Then only waiting multiactions from $EnaWait([G]_{\approx})$ are stamped in G.*

Proof. By the definition of reachability, there exists $E \in StatExpr$ without stamped waiting multiactions, such that G is derived from $\overline{E}$ by applying the inaction rules (also those reversed) and action rules.

In that derivation, only the first *inaction* rule can add timer value superscripts to the waiting multiactions from $\mathcal{WL}(G) = \mathcal{WL}(E)$ that are overlined. The other inaction rules (also reversed) can just "shift" the upper bars from / to those stamped waiting multiactions while preserving the enabledness of all waiting multiactions from $\mathcal{WL}(G)$. Thus, just the waiting multiactions from $EnaWait([G]_{\approx})$ *become* stamped in the subexpressions of G, such as $\overline{(\alpha, \natural_l^\theta)^\theta}$ or $(\alpha, \natural_l^\theta)^\theta$.

Table 4: Comparison of inaction, action and empty move rules

Rules	State change	Time progress	Activities execution
Inaction rules	$-$	$-$	$-$
Action rules (stochastic or waiting multiactions)	$\pm$	$+$	$+$
Action rules (immediate multiactions)	$\pm$	$-$	$+$
Empty move rule	$\pm$	$+$	$-$

Further, in the derivation, the *action* rules cannot add timer value superscripts to the waiting multiactions from $\mathcal{WL}(G)$. Instead, the action rules can make such waiting multiactions non-enabled (disabled), i.e. belonging to $\mathcal{WL}(G) \setminus EnaWait([G]_{\approx})$. Such "disabling" action rules correspond either to the executing an overlined stamped (with the value 1) waiting multiaction (rule **Bw**) or to the choice of some alternative process branch (rules **Cs**, **Ci**, **Cw**, **I2s**, **I2i**, **I2w**). In the both cases, all the disabled waiting multiactions loose their timer value superscripts. Thus, only the waiting multiactions from $EnaWait([G]_{\approx})$ *remain* stamped in G.

Hence, $\overline{E}$ does not contain stamped waiting multiactions and in the derivation of G from it, only the waiting multiactions from $EnaWait([G]_{\approx})$ *become* and *remain* stamped in G. Therefore, only waiting multiactions from $EnaWait([G]_{\approx})$ are stamped in G. □

In Table 4, inaction rules, action rules (with stochastic or immediate, or waiting multiactions) and empty move rule are compared according to the three questions about their application: whether it changes the current state, whether it leads to a time progress, and whether it results in execution of some activities. Positive answers to the questions are denoted by the plus sign while negative ones are specified by the minus sign. If both positive and negative answers can be given to some of the questions in different cases then the plus-minus sign is written. Notice that the process states are considered up to structural equivalence of the corresponding expressions, and time progress is not regarded as a state change.

3.3 Transition systems

We now construct labeled probabilistic transition systems associated with dynamic expressions. The transition systems are used to define the operational semantics of dynamic expressions.

Let G be a dynamic expression and $s = [G]_{\approx}$. The set of *all multisets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. Here $H \xrightarrow{\Upsilon} \widetilde{H}$ is an inference by the rules from Table 3.

It can be proved by induction on the structure of expressions that $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ implies $\exists H \in s\ \Upsilon \in Now(H)$. The reverse statement does not hold in general, since the preconditions in the action rules disable executions of the activities with the lower-priority types from every $H \in s$, as the next example shows.

Example 3.8 *Let H, H' be from Example 3.7 and $s = [H]_{\approx} = [H']_{\approx}$. We have $Now(H) = \{\{(\{a\}, \natural_1^0)\}\}$ and $Now(H') = \{\{(\{b\}, \frac{1}{2})\}\}$. Since only rules* **Ci** *and* **Bi** *can be applied to H while no action rule can be applied to H', we get $Exec(s) = \{\{(\{a\}, \natural_1^0)\}\}$. Then, for $H' \in s$ and $\Upsilon = \{(\{b\}, \frac{1}{2})\} \in Now(H')$, we obtain $\Upsilon \notin Exec(s)$.*

The state s is *s-tangible (stochastically tangible)*, denoted by $stang(s)$, if $Exec(s) \subseteq I\!N_{fin}^{\mathcal{SL}}$. For an s-tangible state s we always have $\emptyset \in Exec(s)$ by rule **E**, hence, we may have $Exec(s) = \{\emptyset\}$. The state s is *w-tangible (waitingly tangible)*, denoted by $wtang(s)$, if $Exec(s) \subseteq I\!N_{fin}^{\mathcal{WL}} \setminus \{\emptyset\}$. The state s is *tangible*, denoted by $tang(s)$, if $stang(s)$ or $wtang(s)$, i.e. $Exec(s) \subseteq I\!N_{fin}^{\mathcal{SL}} \cup I\!N_{fin}^{\mathcal{WL}}$. Again, for a tangible state s we may have $\emptyset \in Exec(s)$ and $Exec(s) = \{\emptyset\}$. Otherwise, the state s is *vanishing*, denoted by $vanish(s)$, and in this case $Exec(s) \subseteq I\!N_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$.

Since for every $H \in s$, $Now(H)$ containing the multisets of activities with the lower-priority types is not included into $Exec(s)$, and the types of states are determined from the highest-priority types of the executable activities, the state type definitions based on $Now(H)$, $H \in s$, and on $Exec(s)$ are consistent.

Note that if $\Upsilon \in Exec(s)$ and $\Upsilon \in I\!N_{fin}^{\mathcal{SL}} \cup I\!N_{fin}^{\mathcal{IL}}$ then by rules **P2s**, **P2i**, **Sy2s**, **Sy2i** and definition of $Exec(s)$ $\forall \Xi \subseteq \Upsilon,\ \Xi \neq \emptyset$, we have $\Xi \in Exec(s)$, i.e. $2^{\Upsilon} \setminus \{\emptyset\} \subseteq Exec(s)$.

Since the inaction rules only set the initial timer values or distribute and move upper and lower bars along the syntax of dynamic expressions, all $H \in s$ have the same timer-free underlying static expression F. Process expressions always have a finite length, hence, the number of all (enumerated) activities and the number of all operations in the syntax of F are finite as well. The action rules **Sy2s**, **Sy2i** and **Sy2w** are the only ones that generate new activities. They result from the handshake synchronization of actions and their conjugates

belonging to the multiaction parts of the first and second constituent activity, respectively. Since we have a finite number of operators sy in F and all the multiaction parts of the activities are finite multisets, the number of the new synchronized activities is also finite. The action rules contribute to $Exec(s)$ (in addition to the empty set, if rule **E** is applicable) only the sets consisting both of activities from F and the new activities, produced by **Sy2s**, **Sy2i** and **Sy2w**. Since we have a finite number n of all such activities, we get $|Exec(s)| \leq 2^n < \infty$. Thus, summation and multiplication by elements from the finite set $Exec(s)$ are well-defined. Similar reasoning can be used to demonstrate that for all dynamic expressions H (not just for those from s), $Now(H)$ is a finite set.

Definition 3.7 *The* derivation set *of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that*

- $[G]_{\approx} \in DR(G)$;
- *if* $[H]_{\approx} \in DR(G)$ *and* $\exists \Upsilon\ H \xrightarrow{\Upsilon} \widetilde{H}$ *then* $[\widetilde{H}]_{\approx} \in DR(G)$.

The set of *all s-tangible states from* $DR(G)$ is denoted by $DR_{ST}(G)$, and the set of *all w-tangible states from* $DR(G)$ is denoted by $DR_{WT}(G)$. The set of *all tangible states from* $DR(G)$ is denoted by $DR_T(G) = DR_{ST}(G) \cup DR_{WT}(G)$. The set of *all vanishing states from* $DR(G)$ is denoted by $DR_V(G)$. Obviously, $DR(G) = DR_T(G) \uplus DR_V(G) = DR_{ST}(G) \uplus DR_{WT}(G) \uplus DR_V(G)$, where $\uplus$ denotes disjoint union.

Let now G be a dynamic expression and $s, \tilde{s} \in DR(G)$.

Let $\Upsilon \in Exec(s) \setminus \{\emptyset\}$. The *probability that the multiset of stochastic multiactions Υ is ready for execution in s* or the *weight of the multiset of deterministic multiactions Υ which is ready for execution in s* is

$$PF(\Upsilon, s) = \begin{cases} \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}}(1-\chi), & s \in DR_{ST}(G); \\ \sum_{(\alpha,\natural_l^\theta)\in\Upsilon} l, & s \in DR_{WT}(G) \cup DR_V(G). \end{cases}$$

In the case $\Upsilon = \emptyset$ and $s \in DR_{ST}(G)$ we define

$$PF(\emptyset, s) = \begin{cases} \prod_{\{(\beta,\chi)\}\in Exec(s)}(1-\chi), & Exec(s) \neq \{\emptyset\}; \\ 1, & Exec(s) = \{\emptyset\}. \end{cases}$$

If $s \in DR_{ST}(G)$ and $Exec(s) \neq \{\emptyset\}$ then $PF(\Upsilon, s)$ can be interpreted as a *joint* probability of independent events (in a probability sense, i.e. the probability of intersection of these events is equal to the product of their probabilities). Each such an event consists in the positive or the negative decision to be executed of a particular stochastic multiaction. Every executable stochastic multiaction decides probabilistically (using its probabilistic part) and independently (from others), if it wants to be executed in s. If Υ is a multiset of all executable stochastic multiactions which have decided to be executed in s and $\Upsilon \in Exec(s)$ then Υ is ready for execution in s. The multiplication in the definition is used because it reflects the probability of the independent event intersection. Alternatively, when $\Upsilon \neq \emptyset$, $PF(\Upsilon, s)$ can be interpreted as the probability to execute *exclusively* the multiset of stochastic multiactions Υ in s, i.e. the probability of *intersection* of two events calculated using the conditional probability formula in the form of $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y)$. The event X consists in the execution of Υ in s. The event Y consists in the non-execution in s of all the executable stochastic multiactions not belonging to Υ. Since the mentioned non-executions are obviously independent events, the probability of Y is a product of the probabilities of the non-executions: $\mathsf{P}(Y) = \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}}(1-\chi)$. The conditioning of X by Y makes the executions of the stochastic multiactions from Υ independent, since all of them can be executed in parallel in s by definition of $Exec(s)$. Hence, the probability to execute Υ *under condition* that no executable stochastic multiactions not belonging to Υ are executed in s is a product of probabilities of these stochastic multiactions: $\mathsf{P}(X|Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho$. Thus, the probability that Υ is executed *and* no executable stochastic multiactions not belonging to Υ are executed in s is the probability of X conditioned by Y multiplied by the probability of Y: $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}}(1-\chi)$. When $\Upsilon = \emptyset$, $PF(\Upsilon, s)$ can be interpreted as the probability not to execute in s any executable stochastic multiactions, thus, $PF(\emptyset, s) = \prod_{\{(\beta,\chi)\}\in Exec(s)}(1-\chi)$. When only the empty multiset of activities can be executed in s, i.e. $Exec(s) = \{\emptyset\}$, we take $PF(\emptyset, s) = 1$, since nothing more can be executed in s in this case. Since the probabilities of all stochastic multiactions are strictly less than 1, for $s \in DR_{ST}(G)$ we have $PF(\emptyset, s) \in (0; 1]$. Hence, we always execute the empty multiset of activities in s at the next time moment with a certain positive probability.

If $s \in DR_{WT}(G) \cup DR_V(G)$ then $PF(\Upsilon, s)$ could be interpreted as the *overall (cumulative)* weight of the deterministic multiactions from Υ, i.e. the sum of all their weights. The summation here is used since the weights can be seen as the rewards which are collected [260]. This means that concurrent execution of the deterministic multiactions has more importance than that of every single one. The weights of deterministic multiactions can also be interpreted as bonus rewards of transitions [32]. The rewards are summed when deterministic multiactions are executed in parallel, because all of them participated in the execution. In particular, since

execution of immediate multiactions takes no time, we prefer to collect in a step (parallel execution of activities) as many parallel immediate multiactions as possible to get more progress in behaviour. This aspect will be used later, while while evaluating performance on the basis of the EDTMCs of expressions. Concerning waiting multiactions, only the maximal multisets of them executable from a state occur in the next moment. Therefore, the steps of waiting multiactions produce maximal overall weights, which are used to calculate probabilities of alternative maximal steps rather than the cumulative bonuses. Note that this reasoning is the same as that used to define the weight of synchronized immediate (waiting, respectively) multiactions in the rules **Sy2i** and **Sy2w**.

Note that the definition of $PF(\Upsilon, s)$ (as well as the definitions of other probability functions which we shall present) is based on the enumeration of activities which is considered implicit.

Let $\Upsilon \in Exec(s)$. Besides Υ, some other multisets of activities may be ready for execution in s, hence, a kind of conditioning or normalization is needed to calculate the execution probability. The *probability to execute the multiset of activities* Υ *in* s is

$$PT(\Upsilon, s) = \frac{PF(\Upsilon, s)}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)}.$$

If $s \in DR_{ST}(G)$ then $PT(\Upsilon, s)$ can be interpreted as the *conditional* probability to execute Υ in s calculated using the conditional probability formula in the form of $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z \cap W)}{\mathsf{P}(W)}$. The event Z consists in the exclusive execution of Υ in s, hence, $\mathsf{P}(Z) = PF(\Upsilon, s)$. The event W consists in the exclusive execution of any set (including the empty one) $\Xi \in Exec(s)$ in s. Thus, $W = \cup_j Z_j$, where $\forall j,\ Z_j$ are mutually exclusive events (in a probability sense, i.e. intersection of these events is the empty event) and $\exists i,\ Z = Z_i$. We have $\mathsf{P}(W) = \sum_j \mathsf{P}(Z_j) = \sum_{\Xi \in Exec(s)} PF(\Xi, s)$, because summation reflects the probability of the mutually exclusive event union. Since $Z \cap W = Z_i \cap (\cup_j Z_j) = Z_i = Z$, we have $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z)}{\mathsf{P}(W)} = \frac{PF(\Upsilon, s)}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)}$. One can also treat $PT(\Upsilon, s)$ and $PF(\Upsilon, s)$ as the *actual* and *potential* probabilities to execute Υ in s, respectively, since we have $PT(\Upsilon, s) = PF(\Upsilon, s)$ only when *all* sets (including the empty one) consisting of the executable stochastic multiactions can be executed in s. In this case, all the mentioned stochastic multiactions can be executed in parallel in s and we have $\sum_{\Xi \in Exec(s)} PF(\Xi, s) = 1$, since this sum collects the products of *all* combinations of the probability parts of the stochastic multiactions and the negations of these parts. But in general, for example, for two stochastic multiactions (α, ρ) and (β, χ) executable in s, it may happen that they cannot be executed in s together, in parallel, i.e. $\emptyset, \{(\alpha, \rho)\}, \{(\beta, \chi)\} \in Exec(s)$, but $\{(\alpha, \rho), (\beta, \chi)\} \notin Exec(s)$. Note that for $s \in DR_{ST}(G)$ we have $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to execute the empty multiset of activities in s at the next time moment.

If $s \in DR_{WT}(G) \cup DR_V(G)$ then $PT(\Upsilon, s)$ can be interpreted as the weight of the set of deterministic multiactions Υ which is ready for execution in s *normalized* by the weights of *all* the sets executable in s. This approach is analogous to that used in the EMPA definition of the probabilities of immediate actions executable from the same process state [36] (inspired by way in which the probabilities of conflicting immediate transitions in GSPNs are calculated [16]). The only difference is that we have a step semantics and, for every set of deterministic multiactions executed in parallel, we should use its cumulative weight. To get the analogy with EMPA possessing interleaving semantics, we should interpret the weights of immediate actions of EMPA as the cumulative weights of the sets of deterministic multiactions of dtsdPBC.

The advantage of our two-stage approach to definition of the probability to execute a set of activities is that the probability formula $PT(\Upsilon, s)$ is valid both for (sets of) stochastic and deterministic multiactions. It allows one to unify the notation used later while constructing the operational semantics and analyzing performance.

Note that the sum of outgoing probabilities for the expressions belonging to the derivations of G is equal to 1. More formally, $\forall s \in DR(G)\ \sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1$. This, obviously, follows from the definition of $PT(\Upsilon, s)$, and guarantees that it defines a probability distribution.

The *probability to move from* s *to* $\tilde{s}$ *by executing any multiset of activities* is

$$PM(s, \tilde{s}) = \sum_{\{\Upsilon | \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PT(\Upsilon, s).$$

The summation in the definition above reflects the probability of the mutually exclusive event union, since $\sum_{\{\Upsilon | \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PT(\Upsilon, s) = \frac{1}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)} \cdot \sum_{\{\Upsilon | \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PF(\Upsilon, s)$, where for each $\Upsilon,\ PF(\Upsilon, s)$ is the probability of the exclusive execution of Υ in s. Note that $\forall s \in DR(G)$ $\sum_{\{\tilde{s} | \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ \exists \Upsilon\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PM(s, \tilde{s}) = \sum_{\{\tilde{s} | \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ \exists \Upsilon\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} \sum_{\{\Upsilon | \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PT(\Upsilon, s) =$ $\sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1$.

Example 3.9 *Let* $E = (\{a\}, \rho)[](\{a\}, \chi)$*, where* $\rho, \chi \in (0; 1)$*.* $DR(\overline{E})$ *consists of the equivalence classes* $s_1 = [\overline{E}]_{\approx}$ *and* $s_2 = [\underline{E}]_{\approx}$*. We have* $DR_T(\overline{E}) = \{s_1, s_2\}$*. The execution probabilities are calculated as follows. Since*

Table 5: Calculation of the probability functions $PF,\ PT,\ PM$ for $s_1 \in DR(\overline{E})$ and $E = (\{a\},\rho)[](\{a\},\chi)$

$s_1\backslash\Upsilon$	$\emptyset$	$\{(\{a\},\rho)\}$	$\{(\{a\},\chi)\}$	Σ
PF	$(1-\rho)(1-\chi)$	$\rho(1-\chi)$	$\chi(1-\rho)$	$1-\rho\chi$
PT	$\frac{(1-\rho)(1-\chi)}{1-\rho\chi}$	$\frac{\rho(1-\chi)}{1-\rho\chi}$	$\frac{\chi(1-\rho)}{1-\rho\chi}$	1
PM	$\frac{(1-\rho)(1-\chi)}{1-\rho\chi}$ (s_1)	$\frac{\rho+\chi-2\rho\chi}{1-\rho\chi}$ (s_2)		1

Table 6: Calculation of the probability functions $PF,\ PT,\ PM$ for $s'_1 \in DR(\overline{E}')$ and $E' = (\{a\},\natural_l^0)[](\{a\},\natural_m^0)$

$s'_1\backslash\Upsilon$	$\{(\{a\},\natural_l^0)\}$	$\{(\{a\},\natural_m^0)\}$	Σ
PF	l	m	$l+m$
PT	$\frac{l}{l+m}$	$\frac{m}{l+m}$	1
PM	1 (s'_2)		1

$Exec(s_1) = \{\emptyset, \{(\{a\},\rho)\}, \{(\{a\},\chi)\}\}$, we get $PF(\{(\{a\},\rho)\}, s_1) = \rho(1-\chi),\ PF(\{(\{a\},\chi)\}, s_1) = \chi(1-\rho)$ and $PF(\emptyset, s_1) = (1-\rho)(1-\chi)$. Then $\sum_{\Xi\in Exec(s_1)} PF(\Xi, s_1) = \rho(1-\chi) + \chi(1-\rho) + (1-\rho)(1-\chi) = 1-\rho\chi$. Thus, $PT(\{(\{a\},\rho)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi},\ PT(\{(\{a\},\chi)\}, s_1) = \frac{\chi(1-\rho)}{1-\rho\chi}$ and $PT(\emptyset, s_1) = PM(s_1, s_1) = \frac{(1-\rho)(1-\chi)}{1-\rho\chi}$. Further, $Exec(s_2) = \{\emptyset\}$, hence, $\sum_{\Xi\in Exec(s_2)} PF(\Xi, s_2) = PF(\emptyset, s_2) = 1$ and $PT(\emptyset, s_2) = PM(s_2, s_2) = \frac{1}{1} = 1$. Finally, $PM(s_1, s_2) = PT(\{(\{a\},\rho)\}, s_1) + PT(\{(\{a\},\chi)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi} + \frac{\chi(1-\rho)}{1-\rho\chi} = \frac{\rho+\chi-2\rho\chi}{1-\rho\chi}$. In Table 5, the calculation of the probability functions $PF(\Upsilon, s_1),\ PT(\Upsilon, s_1),\ PM(s_1, s)$ is explained, where $\Upsilon \in Exec(s_1),\ s \in \{s_1, s_2\}$ (the value of s is depicted in the parentheses near the value of $PM(s_1, s)$) and $\Sigma = \sum_{\Xi\in Exec(s_1)} PX(\Xi, s_1),\ PX \in \{PF, PT, PM\}$.

Let $E' = (\{a\},\natural_l^0)[](\{a\},\natural_m^0)$, where $l, m \in \mathbb{R}_{>0}$. $DR(\overline{E'})$ consists of the equivalence classes $s'_1 = [\overline{E'}]_\approx$ and $s'_2 = [\underline{E'}]_\approx$. We have $DR_T(\overline{E'}) = \{s'_2\}$ and $DR_V(\overline{E'}) = \{s'_1\}$. The execution probabilities are calculated as follows. Since $Exec(s'_1) = \{\{(\{a\},\natural_l^0)\}, \{(\{a\},\natural_m^0)\}\}$, we get $PF(\{(\{a\},\natural_l^0)\}, s'_1) = l$ and $PF(\{(\{a\},\natural_m^0)\}, s'_1) = m$. Then $\sum_{\Xi\in Exec(s'_1)} PF(\Xi, s'_1) = l + m$. Thus, $PT(\{(\{a\},\natural_l^0)\}, s'_1) = \frac{l}{l+m}$ and $PT(\{(\{a\},\natural_m^0)\}, s'_1) = \frac{m}{l+m}$. Further, $Exec(s'_2) = \{\emptyset\}$, hence, $\sum_{\Xi\in Exec(s'_2)} PF(\Xi, s'_2) = PF(\emptyset, s'_2) = 1$ and $PT(\emptyset, s'_2) = PM(s'_2, s'_2) = \frac{1}{1} = 1$. Finally, $PM(s'_1, s'_2) = PT(\{(\{a\},\natural_l^0)\}, s'_1) + PT(\{(\{a\},\natural_m^0)\}, s'_1) = \frac{l}{l+m} + \frac{m}{l+m} = 1$. In Table 6, the calculation of the probability functions $PF(\Upsilon, s'_1),\ PT(\Upsilon, s'_1),\ PM(s'_1, s')$ is explained, where $\Upsilon \in Exec(s'_1),\ s' \in \{s'_2\}$ (the value of s' is depicted in the parentheses near the value of $PM(s'_1, s')$) and $\Sigma = \sum_{\Xi\in Exec(s'_1)} PX(\Xi, s'_1),\ PX \in \{PF, PT, PM\}$.

Definition 3.8 *Let G be a dynamic expression. The* (labeled probabilistic) transition system *of G is a quadruple $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, where*

- *the set of* states *is $S_G = DR(G)$;*
- *the set of* labels *is $L_G = \mathbb{N}_{fin}^{\mathcal{SDL}} \times (0;1]$;*
- *the set of* transitions *is $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s, \tilde{s} \in DR(G),\ \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$;*
- *the* initial state *is $s_G = [G]_\approx$.*

Example 3.10 *Let E be from Example 3.1. The next inferences by rule* **E** *are possible from the elements of $[\overline{E}]_\approx$:*

$$\overline{(\{a\},\natural_1^3)[](\{b\},\tfrac{1}{3})} \approx \overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3}) \xrightarrow{\emptyset} \overline{(\{a\},\natural_1^3)^2}[](\{b\},\tfrac{1}{3}),$$
$$\overline{(\{a\},\natural_1^3)[](\{b\},\tfrac{1}{3})} \approx (\{a\},\natural_1^3)^3[]\overline{(\{b\},\tfrac{1}{3})} \xrightarrow{\emptyset} (\{a\},\natural_1^3)^2[]\overline{(\{b\},\tfrac{1}{3})}.$$

The first and second inferences suggest the empty move transition $[\overline{E}]_\approx \xrightarrow{\emptyset} [\overline{(\{a\},\natural_1^3)^2[](\{b\},\frac{1}{3})}]_\approx \neq [\overline{E}]_\approx$. The intuition is that the timer of the enabled waiting multiaction $(\{a\},\natural_1^3)$ is decremented by one time unit in the both cases, whenever it is overlined or not. Later we shall see that in the both cases, the respective waiting transition of the LDTSDPN corresponding to $\overline{E}$ will be enabled at a "common" marking (that also enables a stochastic transition, matched up to $(\{b\},\frac{1}{3})$), so its timer should be decreased by one with a time tick while staying at the same marking*, and such a time move will lead to a* different state *of the LDTSDPN.*

The definition of $TS(G)$ is correct, i.e. for every state, the sum of the probabilities of all the transitions starting from it is 1. This is guaranteed by the note after the definition of $PT(\Upsilon, s)$. Thus, we have defined a *generative* model of probabilistic processes, according to the classification from [138]. The reason is that the sum of the probabilities of the transitions with all possible labels should be equal to 1, not only of those with the same labels (up to enumeration of activities they include) as in the *reactive* models, and we do not have a nested probabilistic choice as in the *stratified* models.

The transition system $TS(G)$ associated with a dynamic expression G describes all the steps (parallel executions) that occur at discrete time moments with some (one-step) probability and consist of multisets of activities. Every step consisting of stochastic (waiting, respectively) multiactions or the empty step (i.e. that consisting of the empty multiset of activities) occurs instantly after one discrete time unit delay. Each step consisting of immediate multiactions occurs instantly without any delay. The step can change the current state to a different one. The states are the structural equivalence classes of dynamic expressions obtained by application of action rules starting from the expressions belonging to $[G]_{\approx}$. A transition $(s, (\Upsilon, \mathcal{P}), \tilde{s}) \in \mathcal{T}_G$ will be written as $s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$. It is interpreted as follows: the probability to change from state s to $\tilde{s}$ as a result of executing Υ is $\mathcal{P}$.

Note that from every s-tangible state the empty multiset of activities can always be executed by rule **E**. Hence, for s-tangible states, Υ may be the empty multiset, and its execution only decrements by one the timer values (if any) of the current state (i.e. the equivalence class). Then we may have a transition $s \xrightarrow{\emptyset}_{\mathcal{P}} \circlearrowleft s$ from an s-tangible state s to the tangible (i.e. s-tangible or w-tangible) state $\circlearrowleft s = [\circlearrowleft H]_{\approx}$ for $H \in s \cap SatOpRegDynExpr$. Since structurally equivalent saturated operative dynamic expressions remain so after decreasing by one their timer values, $\circlearrowleft s$ is unique for each s and the definition is correct. Thus, $\circlearrowleft s$ is the structural equivalence class of an arbitrary saturated operative dynamic expression from s, where timer values have been decrements by one, prior to taking the equivalence class of that expression. We cannot simply collect all the timer-decremented (by one) saturated operative dynamic expressions from s, since $\circlearrowleft s$ should be a *state* itself, i.e. it must contain *all* structurally equivalent expressions.

Example 3.11 *Let E be from Example 3.1 and $s = [\overline{E}]_{\approx}$. Then* $\circlearrowleft s = [\overline{(\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3})}]_{\approx} = [(\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3})]_{\approx} = \{\overline{(\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3})}, (\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3}), \overline{(\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3})}\} = [\overline{(\{a\}, \natural_1^3)^2[](\{b\}, \frac{1}{3})}]_{\approx}$.

The construction of $\circlearrowleft s$ corresponds to applying the empty move rule to an arbitrary saturated operative dynamic expression from s, followed by taking the structural equivalence class of the resulting expression. We have to keep track of the executions like $s \xrightarrow{\emptyset}_{\mathcal{P}} \circlearrowleft s$, called the *empty moves*, since they affect the timers and have non-zero probabilities. The latter follows from the definition of $PF(\emptyset, s)$ and the fact that the probabilities of stochastic multiactions cannot be equal to 1 as they belong to the interval $(0; 1)$.

When it holds $\circlearrowleft H = H$ for $H \in s \cap SatOpRegDynExpr$, we obtain $\circlearrowleft s = s$ by definition of $\circlearrowleft s$. Then the empty move from s is in the form of $s \xrightarrow{\emptyset}_{\mathcal{P}} s$, called the *empty loop*. For w-tangible and vanishing states Υ cannot be the empty multiset, since we must execute some immediate (waiting, respectively) multiactions from them at the current (next, respectively) time moment.

The step probabilities belong to the interval $(0; 1]$, being 1 in the case when we cannot leave an s-tangible state s and the only transition leaving it is the empty move one $s \xrightarrow{\emptyset}_1 \circlearrowleft s$, or if there is just a single transition from a w-tangible or a vanishing state to any other one.

We write $s \xrightarrow{\Upsilon} \tilde{s}$ if $\exists \mathcal{P}\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$ and $s \to \tilde{s}$ if $\exists \Upsilon\ s \xrightarrow{\Upsilon} \tilde{s}$.

The first equivalence we are going to introduce is isomorphism which is a coincidence of systems up to renaming of their components or states.

Definition 3.9 *Let G, G' be dynamic expressions and $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G), TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$ be their transition systems. A mapping $\beta : S_G \to S_{G'}$ is an* isomorphism *between $TS(G)$ and $TS(G')$, denoted by $\beta : TS(G) \simeq TS(G')$, if*

1. *β is a bijection such that $\beta(s_G) = s_{G'}$;*
2. *$\forall s, \tilde{s} \in S_G\ \forall \Upsilon\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s} \Leftrightarrow \beta(s) \xrightarrow{\Upsilon}_{\mathcal{P}} \beta(\tilde{s})$.*

Two transition systems $TS(G)$ and $TS(G')$ are isomorphic*, denoted by $TS(G) \simeq TS(G')$, if $\exists \beta : TS(G) \simeq TS(G')$.*

Transition systems of static expressions can be defined as well. For $E \in RegStatExpr$, let $TS(E) = TS(\overline{E})$.

Definition 3.10 *Two dynamic expressions G and G' are* equivalent with respect to transition systems*, denoted by $G =_{ts} G'$, if $TS(G) \simeq TS(G')$.*

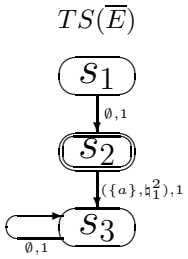

Figure 2: The transition system of $\overline{E}$ for $E = (\{a\},\natural_1^2)[](\{b\},\natural_2^3)$

3.4 Examples of transition systems

We now present a series of examples that demonstrate how to construct the transition systems of the dynamic expressions that include various compositions of stochastic, waiting and immediate multiactions. In the transition systems, the s-tangible and w-tangible states are depicted in ordinary and double ovals, respectively, and the vanishing ones are depicted in boxes. To simplify the graphical representation, the singleton multisets of activities are written without outer braces.

Example 3.12 *Let $E = (\{a\},\natural_1^2)[](\{b\},\natural_2^3)$. $DR(\overline{E})$ consists of the equivalence classes*

$$s_1 = [\overline{(\{a\},\natural_1^2)^2}[](\{b\},\natural_2^3)^3]_{\approx} = [(\{a\},\natural_1^2)^2[]\overline{(\{b\},\natural_2^3)^3}]_{\approx}, \quad s_2 = [\overline{(\{a\},\natural_1^2)^1}[](\{b\},\natural_2^3)^2]_{\approx} = [(\{a\},\natural_1^2)^1[]\overline{(\{b\},\natural_2^3)^2}]_{\approx},$$
$$s_3 = [\underline{(\{a\},\natural_1^2)[](\{b\},\natural_2^3)}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_3\}$, $DR_{WT}(\overline{E}) = \{s_2\}$ and $DR_V(\overline{E}) = \emptyset$. In Figure 2, the transition system $TS(\overline{E})$ is shown.

This example demonstrates a choice between two waiting multiactions with different delays. It shows that the waiting multiaction $(\{a\},\natural_1^2)$ with a less delay 2 is always executed first, hence, the choice is resolved in favour of it in any case and an absorbing state is then reached, so that the waiting multiaction $(\{b\},\natural_2^3)$ with a greater delay 3 is never executed.

Example 3.13 *Let $E = (\{a\},\natural_1^3)[](\{b\},\frac{1}{3})$ (E is from Example 3.1). $DR(\overline{E})$ consists of the equivalence classes*

$$s_1 = [\overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3})]_{\approx} = [(\{a\},\natural_1^3)^3[]\overline{(\{b\},\tfrac{1}{3})}]_{\approx}, \quad s_2 = [\overline{(\{a\},\natural_1^3)^2}[](\{b\},\tfrac{1}{3})]_{\approx} = [(\{a\},\natural_1^3)^2[]\overline{(\{b\},\tfrac{1}{3})}]_{\approx},$$
$$s_3 = [\overline{(\{a\},\natural_1^3)^1}[](\{b\},\tfrac{1}{3})]_{\approx} = [(\{a\},\natural_1^3)^1[]\overline{(\{b\},\tfrac{1}{3})}]_{\approx}, \quad s_4 = [\underline{(\{a\},\natural_1^3)[](\{b\},\tfrac{1}{3})}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_2, s_4\}$, $DR_{WT}(\overline{E}) = \{s_3\}$ and $DR_V(\overline{E}) = \emptyset$. In Figure 3, the transition system $TS(\overline{E})$ is shown.

This example demonstrates a choice between waiting and stochastic multiactions. It shows that the stochastic multiaction $(\{b\},\frac{1}{3})$ can be executed until the timer value of the waiting multiaction $(\{a\},\natural_1^3)$ becomes 1, after which only the waiting multiaction can be executed in the next moment, leading to an absorbing state. Thus, in our setting, a waiting multiaction that cannot be executed in the next time moment and whose timer is still running may be interrupted (preempted) by executing a stochastic multiaction.

Example 3.14 *Let $E = ((\{a\},\natural_1^3)[](\{b\},\frac{1}{3}))$ rs a. $DR(\overline{E})$ consists of the equivalence classes*

$$s_1 = [(\overline{(\{a\},\natural_1^3)^3}[](\{b\},\tfrac{1}{3}))\ \mathsf{rs}\ a]_{\approx} = [((\{a\},\natural_1^3)^3[]\overline{(\{b\},\tfrac{1}{3})})\ \mathsf{rs}\ a]_{\approx},$$
$$s_2 = [(\overline{(\{a\},\natural_1^3)^2}[](\{b\},\tfrac{1}{3}))\ \mathsf{rs}\ a]_{\approx} = [((\{a\},\natural_1^3)^2[]\overline{(\{b\},\tfrac{1}{3})})\ \mathsf{rs}\ a]_{\approx},$$
$$s_3 = [(\overline{(\{a\},\natural_1^3)^1}[](\{b\},\tfrac{1}{3}))\ \mathsf{rs}\ a]_{\approx} = [((\{a\},\natural_1^3)^1[]\overline{(\{b\},\tfrac{1}{3})})\ \mathsf{rs}\ a]_{\approx},$$
$$s_4 = [\underline{((\{a\},\natural_1^3)[](\{b\},\tfrac{1}{3}))\ \mathsf{rs}\ a}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_2, s_3, s_4\}$ and $DR_{WT}(\overline{E}) = \emptyset = DR_V(\overline{E})$. In Figure 4, the transition system $TS(\overline{E})$ is shown.

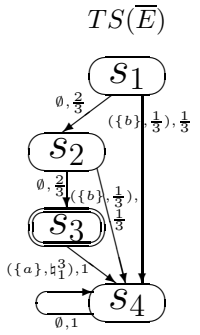

Figure 3: The transition system of $\overline{E}$ for $E = (\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3})$

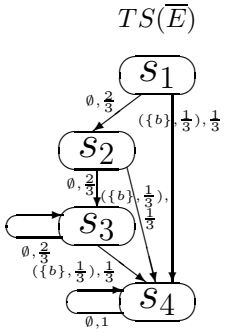

Figure 4: The transition system of $\overline{E}$ for $E = ((\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3}))$ rs a

This example is a modification of the previous Example 3.13 by applying a restriction operation by action a to the whole expression. The present example shows that the stochastic multiaction $(\{b\}, \frac{1}{3})$ can be executed until the timer value of the "restricted" waiting multiaction $(\{a\}, \natural_1^3)$ becomes 1, after which the waiting multiaction also cannot be executed in the next moment, since it is affected by the restriction. Instead, the stochastic multiaction $(\{b\}, \frac{1}{3})$ can be executed again, leading to an absorbing state, or we return to the current state after one time tick (the empty loop in that state). Thus, a waiting multiaction that cannot be executed because of the restriction and whose timer runs until reaching its final value 1 may always be preempted by executing a stochastic multiaction. To verify that the timer value 1 remains unchanged with the time progress, recall the empty move rule **E** *from Table 3 and the definition of $\circlearrowleft G$ with $\max\{1, \delta - 1\} = \max\{1, 0\} = 1$ when $\delta = 1$.*

Note that the timer decrement of the "restricted" waiting multiaction $(\{a\}, \natural_1^3)$ induces a partial (for the first 2 time ticks) unfolding of the behaviour consisting in a choice between executing and non-executing the stochastic multiaction $(\{b\}, \frac{1}{3})$. In our setting, the timer values are kept even for the waiting multiactions that cannot be executed because of the restriction, since they can potentially participate in a synchronization, but the activities resulted from synchronization do not appear explicitly in the syntax of the process expressions, and their timer values can be detected only by observing those of the both synchronized waiting multiactions. Later we shall see an importance of such a construction, particularly, in Examples 3.18 and 3.22.

Example 3.15 *Let $E = [(\{a\}, \frac{1}{2}) * (\{b\}, \natural_1^3) * (\{c\}, \frac{1}{3})]$. $DR(\overline{E})$ consists of the equivalence classes*

$$
\begin{aligned}
s_1 &= [[\overline{(\{a\}, \tfrac{1}{2})} * (\{b\}, \natural_1^3) * (\{c\}, \tfrac{1}{3})]]_{\approx}, \\
s_2 &= [[(\{a\}, \tfrac{1}{2}) * \overline{(\{b\}, \natural_1^3)^3} * (\{c\}, \tfrac{1}{3})]]_{\approx} = [[(\{a\}, \tfrac{1}{2}) * (\{b\}, \natural_1^3)^3 * \overline{(\{c\}, \tfrac{1}{3})}]]_{\approx}, \\
s_3 &= [[(\{a\}, \tfrac{1}{2}) * \overline{(\{b\}, \natural_1^3)^2} * (\{c\}, \tfrac{1}{3})]]_{\approx} = [[(\{a\}, \tfrac{1}{2}) * (\{b\}, \natural_1^3)^2 * \overline{(\{c\}, \tfrac{1}{3})}]]_{\approx}, \\
s_4 &= [[(\{a\}, \tfrac{1}{2}) * \overline{(\{b\}, \natural_1^3)^1} * (\{c\}, \tfrac{1}{3})]]_{\approx} = [[(\{a\}, \tfrac{1}{2}) * (\{b\}, \natural_1^3)^1 * \overline{(\{c\}, \tfrac{1}{3})}]]_{\approx}, \\
s_5 &= [\underline{[(\{a\}, \tfrac{1}{2}) * (\{b\}, \natural_1^3) * (\{c\}, \tfrac{1}{3})]}]_{\approx}.
\end{aligned}
$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_2, s_3, s_5\}$, $DR_{WT}(\overline{E}) = \{s_4\}$ and $DR_V(\overline{E}) = \emptyset$. In Figure 5, the transition system $TS(\overline{E})$ is shown.

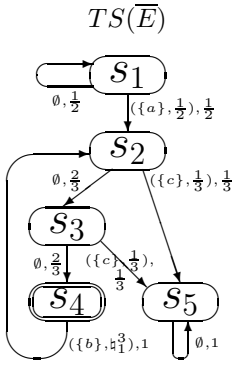

Figure 5: The transition system of $\overline{E}$ for $E = [(\{a\}, \frac{1}{2}) * (\{b\}, \natural_1^3) * (\{c\}, \frac{1}{3})]$

This example demonstrates an iteration loop with a waiting multiaction. The iteration initiation is modeled by a (initiating) stochastic multiaction $(\{a\}, \frac{1}{2})$*. The iteration body that corresponds to the loop consists of a (looping) waiting multiaction* $(\{b\}, \natural_1^3)$*. The iteration termination is represented by a (terminating) stochastic multiaction* $(\{c\}, \frac{1}{3})$*. The terminating stochastic multiaction can be executed until the timer value of the waiting multiaction becomes* 1*, after which only the waiting multiaction can be executed in the next moment. Thus, the iteration termination can either complete the repeated execution of the iteration body or break its execution when the waiting multiaction timer shows some intermediate value (that is less than the initial value, being the multiaction delay, but greater than* 1*). The execution of the waiting multiaction* $(\{b\}, \natural_1^3)$ *leads to the repeated start of the iteration body. The execution of the terminating stochastic multiaction* $(\{c\}, \frac{1}{3})$ *brings to the final absorbing state of the iteration construction.*

Example 3.16 *Let* $E = (\{a\}, \natural_1^0) \| (\{b\}, \natural_2^2) \| (\{c\}, \natural_3^3)$*.* $DR(\overline{E})$ *consists of the equivalence classes*

$$s_1 = [\overline{(\{a\}, \natural_1^0)} \| \overline{(\{b\}, \natural_2^2)^2} \| \overline{(\{c\}, \natural_3^3)^3}]_{\approx}, \quad s_2 = [\underline{(\{a\}, \natural_1^0)} \| \overline{(\{b\}, \natural_2^2)^2} \| \overline{(\{c\}, \natural_3^3)^3}]_{\approx},$$
$$s_3 = [\underline{(\{a\}, \natural_1^0)} \| \overline{(\{b\}, \natural_2^2)^1} \| \overline{(\{c\}, \natural_3^3)^2}]_{\approx}, \quad s_4 = [\underline{(\{a\}, \natural_1^0)} \| \underline{(\{b\}, \natural_2^2)} \| \overline{(\{c\}, \natural_3^3)^1}]_{\approx},$$
$$s_5 = [\underline{(\{a\}, \natural_1^0)} \| (\{b\}, \natural_2^2) \| (\{c\}, \natural_3^3)]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_2, s_5\}$*,* $DR_{WT}(\overline{E}) = \{s_3, s_4\}$ *and* $DR_V(\overline{E}) = \{s_1\}$*. In Figure 6, the transition system* $TS(\overline{E})$ *is shown.*

This example demonstrates a parallel composition of an immediate and two waiting multiactions with different delays. It shows that the immediate multiaction $(\{a\}, \natural_1^0)$ *is always executed before any parallel with it waiting multiaction. Next, from the two parallel waiting multiactions, that* $(\{b\}, \natural_2^2)$ *with a less delay* 2 *executed first in any case. Finally, the execution of the waiting multiaction* $(\{c\}, \natural_3^3)$ *with a greater delay* 3 *leads to an absorbing state. Thus, in spite of parallelism of those three deterministic multiactions, they are executed sequentially in fact, in the increasing order of their (different) delays. That sequence also includes the empty set, executed after the immediate multiaction* $(\{a\}, \natural_1^0)$*, since the waiting multiaction* $(\{b\}, \natural_2^2)$ *with a less delay will then need a passage of one time unit (one time tick) for its timer value (RTE) to become* 1 *and it can be executed itself. Though the example is not complex, it shows a transition system with all three types of states: s-tangible, w-tangible and vanishing.*

Example 3.17 *Let* $E = (\{a\}, \natural_1^3) \| (\{b\}, \frac{1}{3})$*.* $DR(\overline{E})$ *consists of the equivalence classes*

$$s_1 = [\overline{(\{a\}, \natural_1^3)^3} \| \overline{(\{b\}, \tfrac{1}{3})}]_{\approx}, \quad s_2 = [\overline{(\{a\}, \natural_1^3)^2} \| \overline{(\{b\}, \tfrac{1}{3})}]_{\approx}, \quad s_3 = [\overline{(\{a\}, \natural_1^3)^2} \| \underline{(\{b\}, \tfrac{1}{3})}]_{\approx},$$
$$s_4 = [\overline{(\{a\}, \natural_1^3)^1} \| \overline{(\{b\}, \tfrac{1}{3})}]_{\approx}, \quad s_5 = [\overline{(\{a\}, \natural_1^3)^1} \| \underline{(\{b\}, \tfrac{1}{3})}]_{\approx}, \quad s_6 = [\underline{(\{a\}, \natural_1^3)} \| \overline{(\{b\}, \tfrac{1}{3})}]_{\approx},$$
$$s_7 = [\underline{(\{a\}, \natural_1^3) \| (\{b\}, \tfrac{1}{3})}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_2, s_3, s_6, s_7\}$*,* $DR_{WT}(\overline{E}) = \{s_4, s_5\}$ *and* $DR_V(\overline{E}) = \emptyset$*. In Figure 7, the transition system* $TS(\overline{E})$ *is shown.*

This example demonstrates a parallel composition of waiting and stochastic multiactions. It shows that the stochastic multiaction $(\{b\}, \frac{1}{3})$ *can be executed until the timer value of the waiting multiaction* $(\{a\}, \natural_1^3)$ *becomes* 1*, after which only the waiting multiaction can be executed in the next moment. The execution of the latter leads*

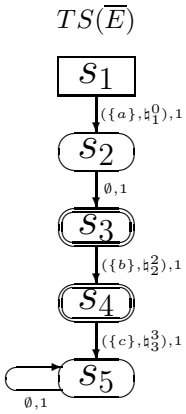

Figure 6: The transition system of $\overline{E}$ for $E = (\{a\},\natural_1^0)\|(\{b\},\natural_2^2)\|(\{c\},\natural_3^3)$

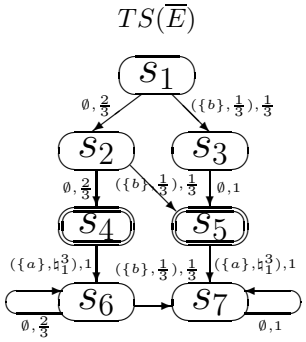

Figure 7: The transition system of $\overline{E}$ for $E = (\{a\},\natural_1^3)\|(\{b\},\frac{1}{3})$

to an absorbing state either directly or indirectly, via executing a possible empty loop, followed (via sequential composition) by the stochastic multiaction $(\{b\},\frac{1}{3})$ *that has not been executed in the preceding states.*

Example 3.18 *Let* $E = ((\{a\},\natural_1^2)\|(\{\hat{a}\},\natural_2^2))$ sy a rs a. $DR(\overline{E})$ *consists of the equivalence classes*

$$s_1 = [(\overline{(\{a\},\natural_1^2)^2}\|\overline{(\{\hat{a}\},\natural_2^2)^2})\ \mathsf{sy}\ a\ \mathsf{rs}\ a]_{\approx},\quad s_2 = [(\overline{(\{a\},\natural_1^2)^1}\|\overline{(\{\hat{a}\},\natural_2^2)^1})\ \mathsf{sy}\ a\ \mathsf{rs}\ a]_{\approx},$$
$$s_3 = [\underline{((\{a\},\natural_1^2)\|(\{\hat{a}\},\natural_2^2))\ \mathsf{sy}\ a\ \mathsf{rs}\ a}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_3\}$, $DR_{WT}(\overline{E}) = \{s_2\}$ *and* $DR_V(\overline{E}) = \emptyset$. *In Figure 8, the transition system* $TS(\overline{E})$ *is shown.*

This example demonstrates a parallel composition of two waiting multiactions $(\{a\},\natural_1^2)$ *and* $(\{\hat{a}\},\natural_2^2)$, *whose multiaction parts are singleton multisets with an action* a *and its conjugate* $\hat{a}$, *respectively. The resulting composition is synchronized and then restricted by that action, which (and its conjugate) therefore "disappears" from the composite process behaviour. From the initial state, only the empty multiset of activities is executed that decrements by one the values of the timers. That evolution follows by the execution of a new waiting multiaction* $(\emptyset,\natural_3^2)$ *with the empty multiaction part, resulted from synchronization of the two waiting multiactions, which leads to an absorbing state.*

Note that the timer values of the two waiting multiactions and that of the new waiting multiaction $(\emptyset,\natural_3^2)$ *(being their synchronous product) coincide until all of them remain enabled with the time progress. Thus, it is very useful that the expression syntax preserves such two enabled synchronized waiting multiactions, removed by restriction from the behaviour, since their timer values suggest that of their synchronous product, which is not explicit in the syntax. Thus, the timer values of those two "virtual" enabled waiting multiactions cannot just be marked as undefined in the syntax, provided that one keeps track of the timer value of their synchronous product being only implicit in the syntax.*

If both synchronized waiting multiactions lose their enabledness with the time progress then their synchronous product $(\emptyset,\natural_3^2)$ *also loses its enabledness and all of them obviously loose their timer value annotations. It may happen that one of the synchronized waiting multiactions loses its enabledness (for example, when a conflicting*

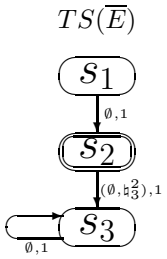

Figure 8: The transition system of $\overline{E}$ for $E = (((\{a\},\natural_1^2)\|(\{\hat{a}\},\natural_2^2))\ \mathsf{sy}\ a\ \mathsf{rs}\ a$

waiting multiaction is executed) while the other one keeps its enabledness. Then their synchronous product also loses its enabledness, together with its timer value annotation. In such a case, the timer value of the enabled synchronized waiting multiaction does not suggest anymore that of the synchronous product. That "saved" timer value merely decrements with every time tick unless it becomes equal to 1*, after which either the enabled synchronized waiting multiaction is executed or it cannot be executed by some reason (for example, when affected by restriction) and then the timer value* 1 *remains unchanged with the time progress. To verify this, recall the empty move rule* **E** *from Table 3 and the definition of* $\circlearrowleft G$ *with* $\max\{1,\delta-1\} = \max\{1,0\} = 1$ *when* $\delta = 1$*.*

Example 3.19 *Let* $E = ((((\{a\},\natural_1^1);(\{b\},\natural_2^3))\|(\{\hat{b}\},\natural_3^3))\ \mathsf{sy}\ b$*.* $DR(\overline{E})$ *consists of the equivalence classes*

$$s_1 = [(((\overline{(\{a\},\natural_1^1)^1};(\{b\},\natural_2^3))\|\overline{(\{\hat{b}\},\natural_3^3)^3})\ \mathsf{sy}\ b]_{\approx},\quad s_2 = [((((\{a\},\natural_1^1);\overline{(\{b\},\natural_2^3)^3})\|\overline{(\{\hat{b}\},\natural_3^3)^2})\ \mathsf{sy}\ b]_{\approx},$$
$$s_3 = [((((\{a\},\natural_1^1);\overline{(\{b\},\natural_2^3)^2})\|\overline{(\{\hat{b}\},\natural_3^3)^1})\ \mathsf{sy}\ b]_{\approx},\quad s_4 = [((((\{a\},\natural_1^1);\overline{(\{b\},\natural_2^3)^1})\|\underline{(\{\hat{b}\},\natural_3^3)})\ \mathsf{sy}\ b]_{\approx},$$
$$s_5 = [\underline{((((\{a\},\natural_1^1);(\{b\},\natural_2^3))\|(\{\hat{b}\},\natural_3^3))\ \mathsf{sy}\ b}]_{\approx}.$$

We have $DR_{ST}(\overline{E}) = \{s_2, s_5\}$*,* $DR_{WT}(\overline{E}) = \{s_1, s_3, s_4\}$ *and* $DR_V(\overline{E}) = \emptyset$*. In Figure 9, the transition system* $TS(\overline{E})$ *is shown.*

This example demonstrates a parallel composition of two subprocesses. The first subprocess is a sequential composition of two waiting multiactions $(\{a\},\natural_1^1)$ *and* $(\{b\},\natural_2^3)$*. The second subprocess consists of a single waiting multiaction* $(\{\hat{b}\},\natural_3^3)$*. The resulting composition is synchronized by the action* b*, which (and its conjugate) therefore "disappears" from the behaviour of their synchronous product. From the initial state, only the waiting multiaction* $(\{a\},\natural_1^1)$ *is executed and the timer of the newly enabled waiting multiaction* $(\{b\},\natural_2^3)$ *starts with the value* 3 *while the timer value* 3 *of* $(\{\hat{b}\},\natural_3^3)$ *is decreased by one and becomes* 2*. That evolution follows by the execution of the empty multiset of activities that further decrements the values of those timers that become* 2 *and* 1*, respectively. Then the waiting multiaction* $(\{\hat{b}\},\natural_3^3)$ *is executed and its timer value annotation disappears while the timer value of* $(\{b\},\natural_2^3)$ *becomes* 1*. Then the execution of the waiting multiaction* $(\{b\},\natural_2^3)$ *finally leads to an absorbing state.*

Thus, the new waiting multiaction $(\emptyset,\natural_5^3)$*, resulted from synchronization of* $(\{b\},\natural_2^3)$ *and* $(\{\hat{b}\},\natural_3^3)$*, cannot be executed, since those synchronized waiting multiactions cannot be executed together (in parallel) in any reachable state. Note that a synchronous product cannot be executed even if one (the latest, in case the timers are disbalanced) of the synchronized activities cannot be executed. Then only the* maximum *timer value of the two synchronized waiting multiactions suggests the timer value of their synchronous product* $(\emptyset,\natural_5^3)$*, until all of them remain enabled with the time progress. The enabledness keeps the corresponding timer value annotations present in the syntax and those values* defined*. Each defined timer value of* $(\{b\},\natural_2^3)$ *is always less by one than that of* $(\{\hat{b}\},\natural_3^3)$*, since the execution of the former waiting multiaction is delayed for one time unit due to the execution of the preceding* $(\{a\},\natural_1^1)$*. Then simultaneous starting the timers of the two synchronized waiting multiactions is prevented, resulting in the disbalanced timers. If just one timer value of the two synchronized waiting multiactions is undefined then that of their synchronous product is undefined too, since it is not enabled in that case.*

Example 3.20 *Let* $E = ((((\{a\},\natural_1^1);(\{b,\hat{x}\},\natural_2^0))\|(((\{x\},\natural_3^0)[](\{c\},\natural_4^1)))\ \mathsf{sy}\ x\ \mathsf{rs}\ x$*.* $DR(\overline{E})$ *consists of the equivalence classes*

$$s_1 = [(((\overline{(\{a\},\natural_1^1)^1};(\{b,\hat{x}\},\natural_2^0))\|(((\{x\},\natural_3^0)[]\overline{(\{c\},\natural_4^1)^1}))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx} =$$
$$[(((\overline{(\{a\},\natural_1^1)^1};(\{b,\hat{x}\},\natural_2^0))\|(\overline{(\{x\},\natural_3^0)}[](\{c\},\natural_4^1)^1))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx},$$
$$s_2 = [((((\{a\},\natural_1^1);\overline{(\{b,\hat{x}\},\natural_2^0)})\|\underline{(((\{x\},\natural_3^0)[](\{c\},\natural_4^1)))}\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx}.$$

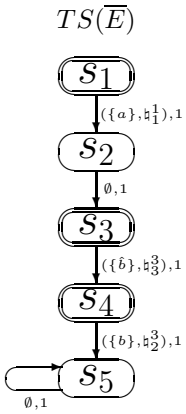

Figure 9: The transition system of $\overline{E}$ for $E = ((((\{a\},\natural_1^1);(\{b\},\natural_2^3))\|(\{\hat{b}\},\natural_3^3))\ \mathsf{sy}\ b$

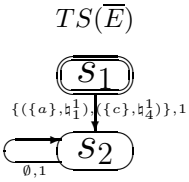

Figure 10: The transition system of $\overline{E}$ for $E = ((((\{a\},\natural_1^1);(\{b,\hat{x}\},\natural_2^0))\|(((\{x\},\natural_3^0)[](\{c\},\natural_4^1)))\ \mathsf{sy}\ x\ \mathsf{rs}\ x$

We have $DR_{ST}(\overline{E}) = \{s_2\}$, $DR_{WT}(\overline{E}) = \{s_1\}$ and $DR_V(\overline{E}) = \emptyset$. In Figure 10, the transition system $TS(\overline{E})$ is shown.

This example demonstrates a parallel composition of two subprocesses, synchronized and then restricted by an auxiliary action that (and its conjugate) hereupon "disappears" from the composite process behaviour. The first subprocess is a sequential composition of the waiting $(\{a\},\natural_1^1)$ and immediate $(\{b,\hat{x}\},\natural_2^0)$ multiactions. The second subprocess is a choice between the immediate $(\{x\},\natural_3^0)$ and waiting $(\{c\},\natural_4^1)$ multiactions. The immediate multiactions $(\{b,\hat{x}\},\natural_2^0)$ and $(\{x\},\natural_3^0)$ in the first and second subprocesses are synchronized via an auxiliary action x that (and its conjugate) is then removed from the behaviour by the restriction operation. Since those immediate multiactions are within coverage of restriction by the auxiliary action, they cannot be executed. The new immediate multiaction $(\{b\},\natural_5^0)$, resulted from that synchronization can only be executed if the waiting multiaction $(\{a\},\natural_1^1)$ (preceding it via sequential composition) in the first subprocess has occurred and the waiting multiaction $(\{c\},\natural_4^1)$ (conflicting with it via the choice composition) in the second subprocess has not occurred. Since only maximal multisets of parallel waiting multiactions may be executed, waiting multiactions in both the subprocesses must occur, thus preventing execution of the new immediate multiaction $(\{b\},\natural_5^0)$, generated by synchronization.

Example 3.21 *Let $E = ((((\{a\},\natural_1^2);(\{b,\hat{x}\},\natural_2^2))\|(((\{x\},\natural_3^2)[](\{c\},\natural_4^2)))\ \mathsf{sy}\ x\ \mathsf{rs}\ x$. $DR(\overline{E})$ consists of the equivalence classes*

$$
\begin{aligned}
s_1 &= [(((\overline{(\{a\},\natural_1^2)^2};(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)^2[]\overline{(\{c\},\natural_4^2)^2}))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx} = \\
&\quad [(((\overline{(\{a\},\natural_1^2)^2};(\{b,\hat{x}\},\natural_2^2))\|(\overline{(\{x\},\natural_3^2)^2}[](\{c\},\natural_4^2)^2))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx}, \\
s_2 &= [(((\overline{(\{a\},\natural_1^2)^1};(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)^1[]\overline{(\{c\},\natural_4^2)^1}))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx} = \\
&\quad [(((\overline{(\{a\},\natural_1^2)^1};(\{b,\hat{x}\},\natural_2^2))\|(\overline{(\{x\},\natural_3^2)^1}[](\{c\},\natural_4^2)^1))\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx}, \\
s_3 &= [((((\{a\},\natural_1^2);\overline{(\{b,\hat{x}\},\natural_2^2)^2})\|\underline{((\{x\},\natural_3^2)[](\{c\},\natural_4^2))})\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx}, \\
s_4 &= [((((\{a\},\natural_1^2);\overline{(\{b,\hat{x}\},\natural_2^2)^1})\|\underline{((\{x\},\natural_3^2)[](\{c\},\natural_4^2))})\ \mathsf{sy}\ x\ \mathsf{rs}\ x]_{\approx}.
\end{aligned}
$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_3, s_4\}$, $DR_{WT}(\overline{E}) = \{s_2\}$ and $DR_V(\overline{E}) = \emptyset$. In Figure 11, the transition system $TS(\overline{E})$ is shown.

This example is a modification of the previous Example 3.20 by replacing all the immediate multiactions with the waiting ones and by setting to 2 the delays of all the waiting multiactions from the syntax. Thus, we

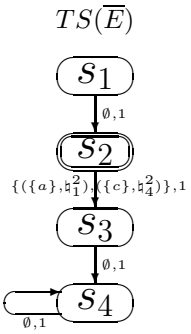

Figure 11: The transition system of $\overline{E}$ for $E = (((\{a\},\natural_1^2);(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)[](\{c\},\natural_4^2)))$ sy x rs x

examine a compound process, constructed with parallelism, synchronization and restriction operations from the following two subprocesses. The first subprocess is a sequential composition of two waiting multiactions $(\{a\},\natural_1^2)$ *and* $(\{b,\hat{x}\},\natural_2^2)$*. The second subprocess is a choice between other two waiting multiactions* $(\{x\},\natural_3^2)$ *and* $(\{c\},\natural_4^2)$*. The second waiting multiaction* $(\{b,\hat{x}\},\natural_2^2)$ *in the first subprocess and the first waiting multiaction* $(\{x\},\natural_3^2)$ *in the second subprocess are synchronized via an auxiliary action* x *that (and its conjugate) is then removed from the behaviour by the restriction operation. The new waiting multiaction* $(\{b\},\natural_5^2)$*, resulted from that synchronization has the same delay* 2 *as the two synchronized waiting multiactions. It can only be executed if the first waiting multiaction* $(\{a\},\natural_1^2)$ *(preceding it via sequential composition) in the first subprocess has occurred and the second waiting multiaction* $(\{c\},\natural_4^2)$ *(conflicting with it via the choice composition) in the second subprocess has not occurred. Since only maximal multisets of parallel waiting multiactions may be executed, the mentioned ("first in first" and "second in second") waiting multiactions in both the subprocesses must occur, thus preventing execution of the new waiting multiaction* $(\{b\},\natural_5^2)$*, generated by synchronization.*

Note that the overlined second waiting multiaction in the first subprocess is within coverage of restriction by the auxiliary action. Consider the state, reached from the initial state by execution of the empty multiset of activities, followed by the parallel execution of the mentioned ('first in first" and "second in second") waiting multiactions. After the empty multiset execution from the considered state, the associated timer value of that overlined waiting multiaction is decremented to 1*. Then an absorbing state is reached, from which only the empty loop is possible, which leaves that timer value* 1 *unchanged though. To verify this, recall the empty move rule* **E** *from Table 3 and the definition of* $\circlearrowleft G$ *with* $\max\{1,\delta-1\} = \max\{1,0\} = 1$ *when* $\delta = 1$*.*

Example 3.22 *Let* $E = (((\{a\},\natural_1^2);(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)[](\{c\},\natural_4^2)))$ sy x*. $DR(\overline{E})$ consists of the equivalence classes*

$$
\begin{aligned}
s_1 &= [(((\overline{(\{a\},\natural_1^2)^2};(\{b,\hat{x}\},\natural_2^2))\|(\overline{(\{x\},\natural_3^2)^2}[](\{c\},\natural_4^2)^2))\ \mathsf{sy}\ x]_{\approx} = \\
&\quad [(((\overline{(\{a\},\natural_1^2)^2};(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)^2[]\overline{(\{c\},\natural_4^2)^2}))\ \mathsf{sy}\ x]_{\approx}, \\
s_2 &= [(((\overline{(\{a\},\natural_1^2)^1};(\{b,\hat{x}\},\natural_2^2))\|(\overline{(\{x\},\natural_3^2)^1}[](\{c\},\natural_4^2)^1))\ \mathsf{sy}\ x]_{\approx} = \\
&\quad [(((\overline{(\{a\},\natural_1^2)^1};(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)^1[]\overline{(\{c\},\natural_4^2)^1}))\ \mathsf{sy}\ x]_{\approx}, \\
s_3 &= [((\{a\},\natural_1^2);\overline{(\{b,\hat{x}\},\natural_2^2)^2})\|\underline{((\{x\},\natural_3^2)[](\{c\},\natural_4^2))}\ \mathsf{sy}\ x]_{\approx}, \\
s_4 &= [((\{a\},\natural_1^2);\overline{(\{b,\hat{x}\},\natural_2^2)^1})\|\underline{((\{x\},\natural_3^2)[](\{c\},\natural_4^2))}\ \mathsf{sy}\ x]_{\approx}, \\
s_5 &= [\underline{(((\{a\},\natural_1^2);(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)[](\{c\},\natural_4^2)))\ \mathsf{sy}\ x}]_{\approx}.
\end{aligned}
$$

We have $DR_{ST}(\overline{E}) = \{s_1,s_3,s_5\}$*,* $DR_{WT}(\overline{E}) = \{s_2,s_4\}$ *and* $DR_V(\overline{E}) = \emptyset$*. In Figure 12, the transition system* $TS(\overline{E})$ *is shown.*

This example is a modification of the previous Example 3.21 by removing restriction from the syntax. Thus, we examine a compound process, constructed with parallelism and synchronization operations from the two subprocesses being a sequential composition of two waiting multiactions $(\{a\},\natural_1^2)$ *and* $(\{b,\hat{x}\},\natural_2^2)$ *and a choice between other two waiting multiactions* $(\{x\},\natural_3^2)$ *and* $(\{c\},\natural_4^2)$*, respectively. All the four waiting multiactions have the same delay* 2*. The second waiting multiaction* $(\{b,\hat{x}\},\natural_2^2)$ *in the first subprocess and the first waiting multiaction* $(\{x\},\natural_3^2)$ *in the second subprocess are synchronized via an auxiliary action* x*. The new waiting multiaction* $(\{b\},\natural_5^2)$*, resulted from that synchronization has the same delay* 2 *as the two synchronized waiting multiactions. It can only be executed if the first waiting multiaction* $(\{a\},\natural_1^2)$ *(preceding it via sequential composition) in the first subprocess has occurred and the second waiting multiaction* $(\{c\},\natural_4^2)$ *(conflicting with it via the*

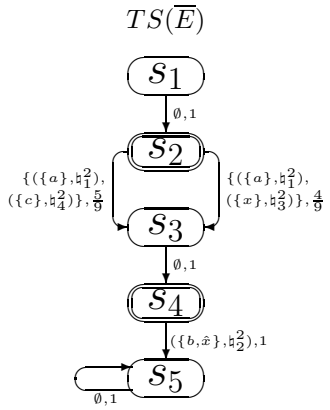

Figure 12: The transition system of $\overline{E}$ for $E = ((((\{a\},\natural_1^2);(\{b,\hat{x}\},\natural_2^2))\|((\{x\},\natural_3^2)[](\{c\},\natural_4^2)))$ sy x

choice composition) in the second subprocess has not occurred. Since only maximal multisets of parallel waiting multiactions may be executed, the mentioned ("first in first" and "second in second") waiting multiactions in the subprocesses must occur, thus preventing execution of the new waiting multiaction $(\{b\},\natural_5^2)$*, generated by synchronization. The alternative maximal multiset of parallel waiting multiactions that may be executed from the same state consists of the "first in first"* $(\{a\},\natural_1^2)$ *and "first in second"* $(\{x\},\natural_3^2)$ *waiting multiactions in the subprocesses, but the 'first in second" waiting multiaction* $(\{x\},\natural_3^2)$ *is the second of the two synchronized waiting multiactions, and its occurrence also prevents execution of their synchronous product* $(\{b\},\natural_5^2)$*.*

Example 3.23 *Consider the expression* $\mathsf{Stop} = (\{g\},\frac{1}{2})$ rs g *specifying the special process that is only able to perform empty loops with probability* 1 *and never terminates. We could actually use any arbitrary action from* $\mathcal{A}$ *and any probability belonging to the interval* $(0;1)$ *in the definition of* Stop*. Note that* Stop *is analogous to the one used in the examples within sPBC. The latter is a continuous time stochastic analogue of the* stop *process proposed in [40].* Stop *is a discrete time stochastic analogue of the* stop*.*

Let $E = [(\{a\},\frac{1}{2}) * ((\{b\},\natural_1^1)[](((\{c\},\natural_2^1);(\{d\},\frac{1}{3}))) * \mathsf{Stop}]$*.* $DR(\overline{E})$ *consists of the equivalence classes*

$$
\begin{aligned}
s_1 &= [[\overline{(\{a\},\tfrac{1}{2})} * ((\{b\},\natural_1^1)[](((\{c\},\natural_2^1);(\{d\},\tfrac{1}{3}))) * \mathsf{Stop}]]_{\approx},\\
s_2 &= [[(\{a\},\tfrac{1}{2}) * \overline{((\{b\},\natural_1^1)^1[](((\{c\},\natural_2^1)^1;(\{d\},\tfrac{1}{3}))) * \mathsf{Stop}]]_{\approx} =\\
&\quad [[(\{a\},\tfrac{1}{2}) * ((\{b\},\natural_1^1)^1[](\overline{((\{c\},\natural_2^1)^1;(\{d\},\tfrac{1}{3}))) * \mathsf{Stop}]]_{\approx},\\
s_3 &= [[(\{a\},\tfrac{1}{2}) * ((\{b\},\natural_1^1)[](((\{c\},\natural_2^1);\overline{(\{d\},\tfrac{1}{3})})) * \mathsf{Stop}]]_{\approx}.
\end{aligned}
$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_3\}$*,* $DR_{WT}(\overline{E}) = \{s_2\}$ *and* $DR_V(\overline{E}) = \emptyset$*. In Figure 13, the transition system* $TS(\overline{E})$ *is presented.*

This example demonstrates an infinite iteration loop. The loop is preceded with the iteration initiation, modeled by a (first) stochastic multiaction $(\{a\},\frac{1}{2})$*. The iteration body that corresponds to the loop consists of the choice between two conflicting waiting multiactions* $(\{b\},\natural_1^1)$ *and* $(\{c\},\natural_2^1)$ *with the same delay* 1*, the second of them followed (via sequential composition) by a (second) stochastic multiaction* $(\{d\},\frac{1}{3})$*. Hence, the iteration loop actually consists of the two alternative subloops, such that the first one is a self-loop (one-state loop from a state to itself) with the first waiting multiaction* $(\{b\},\natural_1^1)$*, and the second one* $(\{c\},\natural_2^1)$ *is a two-state loop with an intermediate state, reached after the second waiting multiaction has been executed, and from which the second stochastic multiaction* $(\{d\},\frac{1}{3})$ *is then started. Thus, the iteration generates the self-loop with probability less than one (since the two-state loop from the same state has a non-zero probability) from the state in which only waiting multiactions are executed. The iteration termination* Stop *demonstrates an empty behaviour, assuring that the iteration does not reach its final state after any number of repeated executions of its body.*

Example 3.24 *Let* $E = [(\{a\},\rho) * ((\{b\},\natural_k^1);((((\{c\},\natural_l^0);(\{d\},\theta))[](((\{e\},\natural_m^0);(\{f\},\phi))))) * \mathsf{Stop}]$*, where* $\rho,\theta,\phi \in (0;1)$ *and* $k,l,m \in \mathbb{N}_{>0}$*.* $DR(\overline{E})$ *consists of the equivalence classes*

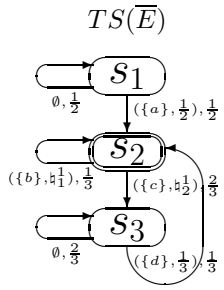

Figure 13: The transition system of $\overline{E}$ for $E = [(\{a\}, \frac{1}{2}) * ((\{b\}, \natural_1^1)[](\{c\}, \natural_2^1); (\{d\}, \frac{1}{3}))) * \mathsf{Stop}]$

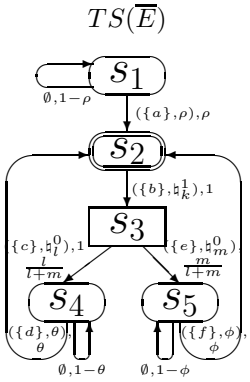

Figure 14: The transition system of $\overline{E}$ for E=$[(\{a\}, \rho)*((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))))*\mathsf{Stop}]$

$$
\begin{aligned}
s_1 &= [[\overline{(\{a\}, \rho)} * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_2 &= [[(\{a\}, \rho) * (\overline{((\{b\}, \natural_k^1)^1}; ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_3 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); (\overline{(((\{c\}, \natural_l^0)}; (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx} =\\
&\quad [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]\overline{((\{e\}, \natural_m^0)}; (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_4 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); \overline{(\{d\}, \theta))}[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_5 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); \overline{(\{f\}, \phi)}))) * \mathsf{Stop}]]_{\approx}.
\end{aligned}
$$

We have $DR_{ST}(\overline{E}) = \{s_1, s_4, s_5\}$, $DR_{WT}(\overline{E}) = \{s_2\}$ *and* $DR_V(\overline{E}) = \{s_3\}$. *In Figure 14, the transition system* $TS(\overline{E})$ *is presented.*

This example demonstrates an infinite iteration loop. The loop is preceded with the iteration initiation, modeled by a stochastic multiaction $(\{a\}, \rho)$. *The iteration body that corresponds to the loop consists of a waiting multiaction* $(\{b\}, \natural_k^1)$, *followed (via sequential composition) by the probabilistic choice, modeled via two conflicting immediate multiactions* $(\{c\}, \natural_l^0)$ *and* $(\{e\}, \natural_m^0)$, *followed by different stochastic multiactions* $(\{d\}, \theta)$ *and* $(\{f\}, \phi)$. *The iteration termination* Stop *demonstrates an empty behaviour, assuring that the iteration does not reach its final state after any number of repeated executions of its body.*

Note that, due to the time constraints and since waiting multiactions may be preempted by stochastic ones, some simple dynamic expressions can have complex transition systems (Examples 3.12–3.17, 3.19, 3.22), or vice versa (Examples 3.18, 3.20, 3.21, 3.23, 3.24).

4 Denotational semantics

In this section, we construct the denotational semantics in terms of a subclass of labeled discrete time stochastic and deterministic PNs (LDTSDPNs), called discrete time stochastic and immediate Petri boxes (dtsd-boxes).

4.1 Labeled DTSDPNs

Let us introduce a class of labeled discrete time stochastic and deterministic PNs (LDTSDPNs), which are essentially a subclass of DTSPNs [226, 228] (since we do not allow the stochastic transition probabilities to be equal to 1) extended with transition labeling and deterministic transitions. LDTSDPNs resemble in part discrete time deterministic and stochastic PNs (DTDSPNs) [309, 305, 306, 311, 312, 310], as well as discrete deterministic and stochastic PNs (DDSPNs) [307, 308]. DTDSPNs and DDSPNs are the extensions of DTSPNs with deterministic transitions (having fixed delay that can be zero), inhibitor arcs, priorities and guards. In addition, while stochastic transitions of DTDSPNs, like those of DTSPNs, have geometrically distributed delays, stochastic transitions of DDSPNs have discrete time phase-type [231, 265, 170, 286, 188, 172] distributed delays. At the same time, LDTSDPNs are not subsumed by DTDSPNs or DDSPNs, by the following reasons. First, in DTDSPNs from [309, 305, 306], both stochastic and deterministic (including immediate) transitions have probabilities and weights associated, but in LDTSDPNs only stochastic transitions have probabilities and only immediate ones have weights, hence, the state change probabilities of the underlying Markov chains for those PN classes are calculated in two different ways. Second, LDTSDPNs have a step semantics while DTDSPNs from [311, 312, 310] and DDSPNs have interleaving one, since in in the first PN class simultaneous transition firings are possible while in the second and third PN classes only firings of single transitions are allowed. LDTSDPNs are somewhat similar to labeled weighted DTSPNs (LWDTSPNs) from [78], but in LWDTSPNs there are no deterministic transitions, all (stochastic) transitions have weights, the transition probabilities may be equal to 1 and only maximal fireable subsets of the enabled transitions are fired.

Stochastic preemptive time PNs (spTPNs) [71] is a discrete time model with a maximal step semantics, where both time ticks and instantaneous parallel firings of maximal transition sets are possible, but the transition steps in LDTSDPNs are not obliged to be maximal (excepting the steps of waiting transitions). The transition delays in spTPNs are governed by static general discrete distributions, associated with the transitions, while the transitions of LDTSDPNs are only associated with probabilities (or delays and weights), used later to calculate the step probabilities after one unit (from tangible states) or zero (from vanishing states) delay. Further, LDTSDPNs have just geometrically distributed or deterministic zero delays at the states. Moreover, the discrete time tick and concurrent transition firing are treated in spTPNs as different events while firing every (possibly empty) set of stochastic or waiting transitions in LDTSDPNs requires one unit time delay. spTPNs are essentially a modification and extension of unlabeled LWDTSPNs with additional facilities, such as inhibitor arcs, priorities, resources, preemptions, schedulers etc. However, the price of such an expressiveness of spTPNs is that the model is rather intricate and difficult to analyze.

Note also that guards in DTDSPNs and DDSPNs, inhibitor arcs and priorities in DTDSPNs, DDSPNs and spTPNs, as well as the maximal step semantics of LWDTSPNs and spTPNs make all these models Turing powerful, resulting in undecidability of many important behavioural properties.

First, we present a formal definition (construction, syntax) of LDTSDPNs. The set of *all row vectors of* $n \in \mathbb{N}_{\geq 1}$ *elements from a set* X is defined as $X^n = \{(x_1, \ldots, x_n) \mid x_i \in X \ (1 \leq i \leq n)\}$.

Definition 4.1 *A* labeled discrete time stochastic and deterministic PN (LDTSDPN) *is a tuple* $N = (P_N, T_N, W_N, D_N, \Omega_N, \mathcal{L}_N, Q_N)$*, where*

- P_N *and* $T_N = Ts_N \uplus Td_N$ *are finite sets of* places *and* stochastic and deterministic transitions*, respectively, such that* $P_N \cup T_N \neq \emptyset$ *and* $P_N \cap T_N = \emptyset$*;*
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ *is a function providing the* weights of arcs *between places and transitions;*
- $D_N : Td_N \to \mathbb{N}$ *is the* transition delay *function imposing delays to deterministic transitions;*

  An immediate transition *is a deterministic transition with the delay* 0 *while a* waiting transition *is that with a positive delay. Then* $Td_N = Ti_N \uplus Tw_N$ *consists of the sets of* immediate and waiting transitions*.*
- Ω_N *is the* transition probability and weight *function such that*
  - $\Omega_N|_{Ts_N} : Ts_N \to (0; 1)$ *(it associates stochastic transitions with probabilities);*
  - $\Omega_N|_{Td_N} : Td_N \to \mathbb{R}_{>0}$ *(it associates deterministic transitions with weights);*
- $\mathcal{L}_N : T_N \to \mathcal{L}$ *is the* transition labeling *function assigning multiactions to transitions;*
- $Q_N = (M_N, V_N)$ *is the* initial state*, where* $M_N \in \mathbb{N}_{fin}^{P_N}$ *is the* initial marking *(distribution of tokens in the places) and* $V_N : Tw_N \to \mathbb{N}_{\geq 1} \cup \{\infty\}$ *is the* initial timer valuation function *of the waiting transitions (in the vector notation,* $V_N \in (\mathbb{N}_{\geq 1} \cup \{\infty\})^{|Tw_N|}$*), where '*$\infty$*' denotes the undefined value of inactive timers (infinite time till the transition firing); we define* $\forall t \in Tw_N \cap Ena(M_N)\ V_N(t) = D_N(t)$ *(each enabled waiting transition is initially valuated with its transition delay) and* $\forall t \in Tw_N \setminus Ena(M_N)\ V_N(t) = \infty$

(each non-enabled waiting transition is initially valuated with the undefined value), where $Ena(M)$ denotes the set of transitions enabled at the marking M, to be defined later.

The graphical representation of LDTSDPNs is like that for standard labeled PNs, but with probabilities or delays and weights written near the corresponding transitions. Square boxes of normal thickness depict stochastic transitions, and those with thick borders represent deterministic transitions. In the case the probabilities or the delays and weights are not given in the picture, they are considered to be of no importance in the corresponding examples. The weights of arcs are depicted with them. The names of places and transitions are depicted near them when needed.

We now consider the semantics of LDTSDPNs.

Let N be an LDTSDPN and $t \in T_N$, $U \in I\!N_{fin}^{T_N}$. The *precondition* ${}^\bullet t$ and the *postcondition* $t^\bullet$ of t are the multisets of places defined as $({}^\bullet t)(p) = W_N(p,t)$ and $(t^\bullet)(p) = W_N(t,p)$. The *precondition* ${}^\bullet U$ and the *postcondition* $U^\bullet$ of U are the multisets of places defined as ${}^\bullet U = \sum_{t \in U} {}^\bullet t$ and $U^\bullet = \sum_{t \in U} t^\bullet$. Note that for $U = \emptyset$ we have ${}^\bullet\emptyset = \emptyset = \emptyset^\bullet$.

Let N be an LDTSDPN and $Q = (M, V), \widetilde{Q} = (\widetilde{M}, \widetilde{V}) \in I\!N_{fin}^{P_N} \times (I\!N_{\geq 1} \cup \{\infty\})^{|Tw_N|}$ be its states.

Deterministic transitions have a priority over stochastic ones, and there is also difference in priorities between immediate and waiting transitions. One can assume that all immediate transitions have (the highest) priority 2 and all waiting transitions have (the medium) priority 1, whereas all stochastic transitions have (the lowest) priority 0. This means that at a marking where all kinds of transitions can occur, immediate transitions always occur before waiting ones that, in turn, are always executed before stochastic ones.

A transition $t \in T_N$ is *enabled* at a marking $M \in I\!N_{fin}^{P_N}$, if ${}^\bullet t \subseteq M$. In other words, a transition is enabled at a marking if it has enough tokens in its input places (i.e. in the places from its precondition) at the marking. Let $Ena(M)$ be the set of *all transitions enabled at* M.

Firings of transitions are atomic operations, and transitions can fire in parallel by taking part in steps. We assume that all transitions participating in a step should differ, hence, only the sets (not multisets) of transitions may fire. Thus, we do not allow self-concurrency, i.e. firing of transitions in parallel to themselves. This restriction is introduced to avoid some technical difficulties while calculating probabilities for multisets of transitions as we shall see after the following formal definitions. Moreover, we do not need to consider self-concurrency, since denotational semantics of expressions will be defined via dtsd-boxes which are safe LDTSDPNs (hence, no self-concurrency is possible).

In terms of timed Petri nets (tPNs) [222], we apply the *single server* policy (semantics) [59], since we do not allow multiple firings of the same transition. We also adapt the *strong time* policy (semantics) [264, 60], since every deterministic transition must fire or become disabled when the right border of its firing interval is approached (in our case, the zero length interval with the left and right closed borders, both equal to the transition delay).

The following definition of fireability respects the prioritization among different types of transitions. A set of transitions $U \subseteq Ena(M)$ is *fireable* in a state $Q = (M, V)$, if ${}^\bullet U \subseteq M$ and one of the following holds:

1. $\emptyset \neq U \subseteq Ti_N$; or
2. $\emptyset \neq U \subseteq Tw_N$ and
   - $\forall t \in U\ V(t) = 1$,
   - $Ena(M - {}^\bullet U) \cap \{u \in Tw_N \setminus U \mid V(u) = 1\} = \emptyset$,
   - $Ena(M) \subseteq Tw_N \cup Ts_N$; or
3. $U \subseteq Ts_N$ and
   - $Ena(M) \subseteq Ts_N$.

In other words, a set of transitions U is fireable in a state, if it has enough tokens in its input places at the substituent marking M of the state and the following holds. If U consists of *immediate* transitions then it is enabled, since no additional condition is needed for its fireability. If U consists of *waiting* transitions then the countdown timer value (called remaining time to fire or RTF) of each transition from U equals one, U is a maximal (by the inclusion relation) set of the enabled at M waiting transitions with the RTF equal to one and enough tokens in its input places at M, and there exist no immediate transitions enabled at M. If U is empty or it consists of *stochastic* transitions then there exist no immediate or waiting transitions enabled at M. Note that the second condition of item 2 of the above definition means that no waiting transition (from $Ena(M)$) with the RTF being one can be added to U so that the resulting transition set will still have enough tokens in its input places at M. This condition is equivalent to the following maximality requirement (informally mentioned above): $\forall T \subseteq Ena(M),\ (\forall u \in T\ V(u) = 1) \wedge ({}^\bullet T \subseteq M) \wedge (U \subseteq T)\ \Rightarrow\ T = U$. Let $Fire(Q)$ be the set of *all transition sets fireable in* Q.

Thus, concerning the LDTSDPNs transitions fireable in a state, the enabled waiting transitions with the RTF greater than one are ignored while those with the RTF being one are treated like (stochastic) transitions of DTSPNs [226, 228] with the conditional probability 1, which have a priority in firing over the (stochastic) transitions with the conditional probability less than 1.

By the definition of fireability, it follows that $Fire(Q) \subseteq 2^{Ti_N} \setminus \{\emptyset\}$ or $Fire(Q) \subseteq 2^{Tw_N} \setminus \{\emptyset\}$, or $Fire(Q) \subseteq 2^{Ts_N}$ (to be convinced of it, check the definition's items in the reverse order). The state Q is *s-tangible (stochastically tangible)*, denoted by $stang(Q)$, if $Fire(Q) \subseteq 2^{Ts_N}$. For an s-tangible state Q we always have $\emptyset \in Fire(Q)$ by the definition of fireability (item 3), hence, we may have $Fire(Q) = \{\emptyset\}$. The state Q is *w-tangible (waitingly tangible)*, denoted by $wtang(Q)$, if $Fire(Q) \subseteq 2^{Tw_N} \setminus \{\emptyset\}$. The state Q is *tangible*, denoted by $tang(Q)$, if $stang(Q)$ or $wtang(Q)$, i.e. $Fire(Q) \subseteq 2^{Ts_N} \cup 2^{Tw_N}$. Again, for a tangible state Q we may have $\emptyset \in Fire(Q)$ and $Fire(Q) = \{\emptyset\}$. Otherwise, the state Q is *vanishing*, denoted by $vanish(Q)$, and in this case $Fire(Q) \subseteq 2^{Ti_N} \setminus \{\emptyset\}$. A transition $t \in Ena(M)$ is *fireable* in a state Q, denoted by $t \in Fire(Q)$, if $\{t\} \in Fire(Q)$. If $stang(Q)$ then a stochastic transition $t \in Fire(Q)$ fires with probability $\Omega_N(t)$ when no different stochastic transition is fireable in Q, i.e. $Fire(Q) = \{\emptyset, \{t\}\}$. By the definition of fireability, if $stang(Q)$ or $vanish(Q)$ then $\forall U \in Fire(Q)\ 2^U \setminus \{\emptyset\} \subseteq Fire(Q)$.

Let $U \in Fire(Q)$ and $U \neq \emptyset$. The *probability that the set of stochastic transitions U is ready for firing in Q* or the *weight of the set of deterministic transitions U which is ready for firing in Q* is

$$PF(U,Q) = \begin{cases} \prod_{t \in U} \Omega_N(t) \cdot \prod_{\{u \in Fire(Q) | u \notin U\}} (1 - \Omega_N(u)), & stang(Q); \\ \sum_{t \in U} \Omega_N(t), & wtang(Q) \vee vanish(Q). \end{cases}$$

In the case $U = \emptyset$ and $stang(Q)$ we define

$$PF(\emptyset,Q) = \begin{cases} \prod_{u \in Fire(Q)} (1 - \Omega_N(u)), & Fire(Q) \neq \{\emptyset\}; \\ 1, & Fire(Q) = \{\emptyset\}. \end{cases}$$

Let $U \in Fire(Q)$. Besides U, some other sets of transitions may be ready for firing in Q, hence, a kind of conditioning or normalization is needed to calculate the firing probability. The parallel firing of the transitions from U changes the state $Q = (M, V)$ to another state $\widetilde{Q} = (\widetilde{M}, \widetilde{V})$, denoted by $Q \stackrel{U}{\rightarrow}_{\mathcal{P}} \widetilde{Q}$, where

1. $\widetilde{M} = M - {}^\bullet U + U^\bullet$;
2. $\forall u \in Tw_N\ \widetilde{V}(u) = \begin{cases} \infty, & u \notin Ena(\widetilde{M}); \\ V_N(u), & (u \in Ena(\widetilde{M}) \setminus Ena(M - {}^\bullet U)) \vee (u \in Ena(\widetilde{M}) \cap U); \\ V(u), & (u \in Ena(M - {}^\bullet U)) \wedge (U \subseteq Ti_N); \\ V(u) - 1, & \text{otherwise}; \end{cases}$
3. $\mathcal{P} = PT(U,Q)$ is the *probability that the set of transitions U fires in Q* defined as

$$PT(U,Q) = \frac{PF(U,Q)}{\sum_{V \in Fire(Q)} PF(V,Q)}.$$

Let us explain the definition above in more detail. The first case of the item 2 demonstrates a waiting transition u that is not enabled at the marking $\widetilde{M}$, regardless of whether it was enabled at the "intermediate" marking $M - {}^\bullet U$ (obtained by removing from M the input places of all transitions belonging to U, and that should be examined, especially when N has structural loops), and therefore the transition timer becomes inactive (turned off) and it is set to the undefined value ∞. The second case of the item 2 describes a waiting transition u that was not enabled at $M - {}^\bullet U$ or was fired within U ($u \in U$) and is (first or again, after leaving M) enabled at $\widetilde{M}$, hence, its timer is restored to the initial value $V_N(u)$, which is the delay of that transition. The third case of the item 2 explains a waiting transition u that was enabled at $M - {}^\bullet U$ and, hence, is still enabled at $\widetilde{M}$, resulted from firing a set of immediate transitions U instantly (in zero time), so the transition timer does not decrement and its value stays equal to $V(u)$. The fourth case of the item 2 corresponds to the remaining option, i.e. a waiting transition u that was enabled at $M - {}^\bullet U$ and, hence, is still enabled at $\widetilde{M}$, resulted from firing a set of stochastic (waiting) transitions U at a time tick (in one time unit), so the transition timer decrements by one and its value becomes $V(u) - 1$.

We do not have to worry that for $u \in Tw_N \setminus U$, such that $u \in Ena(M - {}^\bullet U)$, where $U \subseteq Ts_N \cup Tw_N$, the value of $\widetilde{V}(u) = V(u) - 1$ could become zero or negative, by the following reasons. Note that by the definition of fireability, we have $Ena(M) \subseteq Tw_N \cup Ts_N$. If $V(u) = 1$ then u must fire in the next time moment within some maximal (by the inclusion relation) set of the enabled at M waiting transitions with the RTF equal to one and enough tokens in the set's input places at M. Then we get $U \in Fire(Q) \subseteq 2^{Tw_N} \setminus \{\emptyset\}$, hence, $\emptyset \neq U \subseteq Tw_N$.

Therefore, $\forall t \in U\ V(t) = 1$ and $Ena(M - {}^\bullet U) \cap \{w \in Tw_N \setminus U \mid V(w) = 1\} = \emptyset$, which contradicts to $u \in Ena(M - {}^\bullet U) \cap \{w \in Tw_N \setminus U \mid V(w) = 1\}$. Thus, there exists no transition $u \in Tw_N \setminus U$, such that $u \in Ena(M - {}^\bullet U)$ and $V(u) = 1$. In regard to the transitions $t \in U \subseteq Tw_N$ with $V(t) = 1$, we have $\widetilde{V}(t) = \infty$, if $t \notin Ena(\widetilde{M})$, or $\widetilde{V}(t) = V_N(t)$, if $t \in Ena(\widetilde{M})$.

Note that when $U = \emptyset$ and $stang(Q)$, we get $M = \widetilde{M}$ and $\forall u \in Tw_N\ \widetilde{V}(u) = \begin{cases} \infty, & u \notin Ena(M); \\ V(u) - 1, & u \in Ena(M). \end{cases}$

Note that the timers of all enabled waiting transitions that are disabled when a marking change occurs become inactive (turned off) and their values become undefined while the timers of all those staying enabled (also at the intermediate marking) continue running with their stored values decreased by one. Hence, we adapt the *enabling memory* policy [214, 1, 15, 16] when the markings are changed and the enabling of deterministic transitions is possibly modified (remember that immediate transitions may be seen as those with the timers displaying a single value 0, so we do not need to store their values). Then the timer values of waiting transitions are taken as the enabling memory variables.

In terms of timed Petri nets (tPNs) [222], we apply the *intermediate memory* policy (semantics) [23, 259, 59], since for all waiting transitions that are disabled at the intermediate marking or fired and then are (first or again, after leaving the starting marking) enabled at the resulting (successor) marking, the timers are restored to their initial values (the delays of those transitions).

The advantage of our two-stage approach to definition of the probability that a set of transitions fires is that the probability formula $PT(U, Q)$ is valid both for (sets of) stochastic and deterministic transitions. It allows one to unify the notation used later while constructing the denotational semantics and analyzing performance.

Note that for all states of an LDTSDPN N, the sum of outgoing probabilities is equal to 1. More formally, $\forall Q = (M, V) \in I\!N_{fin}^{P_N} \times (I\!N_{\geq 1} \cup \{\infty\})^{|Tw_N|}\ \sum_{U \in Fire(Q)} PT(U, Q) = 1$. This obviously follows from the definition of $PT(U, Q)$ and guarantees that it defines a probability distribution.

We write $Q \xrightarrow{U} \widetilde{Q}$ if $\exists \mathcal{P}\ Q \xrightarrow{U}_{\mathcal{P}} \widetilde{Q}$ and $Q \to \widetilde{Q}$ if $\exists U\ Q \xrightarrow{U} \widetilde{Q}$.

The *probability to move from Q to $\widetilde{Q}$ by firing any set of transitions* is

$$PM(Q, \widetilde{Q}) = \sum_{\{U \mid Q \xrightarrow{U} \widetilde{Q}\}} PT(U, Q).$$

Since $PM(Q, \widetilde{Q})$ is the probability for *any* (including the empty one) transition set to change from state Q to $\widetilde{Q}$, we use summation in the definition. Note that $\forall Q = (M, V) \in I\!N_{fin}^{P_N} \times (I\!N_{\geq 1} \cup \{\infty\})^{|Tw_N|}\ \sum_{\{\widetilde{Q} \mid Q \to \widetilde{Q}\}} PM(Q, \widetilde{Q}) = \sum_{\{\widetilde{Q} \mid Q \to \widetilde{Q}\}} \sum_{\{U \mid Q \xrightarrow{U} \widetilde{Q}\}} PT(U, Q) = \sum_{U \in Fire(Q)} PT(U, Q) = 1$.

Definition 4.2 *Let N be an LDTSDPN. The* reachability set *of N, denoted by $RS(N)$, is the minimal set of states such that*

- $Q_N \in RS(N)$;
- *if* $Q \in RS(N)$ *and* $Q \to \widetilde{Q}$ *then* $\widetilde{Q} \in RS(N)$.

Definition 4.3 *Let N be an LDTSDPN. The* reachability graph *of N is a (labeled probabilistic) transition system $RG(N) = (S_N, L_N, \mathcal{T}_N, s_N)$, where*

- *the set of* states *is* $S_N = RS(N)$;
- *the set of* labels *is* $L_N = 2^{T_N} \times (0; 1]$;
- *the set of* transitions *is* $\mathcal{T}_N = \{(Q, (U, \mathcal{P}), \widetilde{Q}) \mid Q, \widetilde{Q} \in RS(N),\ Q \xrightarrow{U}_{\mathcal{P}} \widetilde{Q}\}$;
- *the* initial state *is* $s_N = Q_N$.

The set of *all s-tangible states from* $RS(N)$ is denoted by $RS_{ST}(N)$, and the set of *all w-tangible states from* $RS(N)$ is denoted by $RS_{WT}(N)$. The set of *all tangible states from* $RS(N)$ is denoted by $RS_T(N) = RS_{ST}(N) \cup RS_{WT}(N)$. The set of *all vanishing states from* $RS(N)$ is denoted by $RS_V(N)$. Obviously, $RS(N) = RS_T(N) \uplus RS_V(N) = RS_{ST}(N) \uplus RS_{WT}(N) \uplus RS_V(N)$.

4.2 Algebra of dtsd-boxes

We now introduce discrete time stochastic and deterministic Petri boxes and the algebraic operations to define a net representation of dtsdPBC expressions.

Definition 4.4 *A* discrete time stochastic and deterministic Petri box (dtsd-box) *is a tuple* $N = (P_N, T_N, W_N, \Lambda_N)$*, where*

- P_N *and* T_N *are finite sets of* places *and* transitions*, respectively, such that* $P_N \cup T_N \neq \emptyset$ *and* $P_N \cap T_N = \emptyset$*;*
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ *is a function providing the* weights of arcs *between places and transitions;*
- Λ_N *is the* place and transition labeling *function such that*
  - $\Lambda_N|_{P_N} : P_N \to \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$ *(it specifies* entry, internal *and* exit *places, respectively);*
  - $\Lambda_N|_{T_N} : T_N \to \{\varrho \mid \varrho \subseteq \mathbb{N}_{fin}^{\mathcal{SDL}} \times \mathcal{SDL}\}$ *(it associates transitions with* relabeling relations *on activities).*

Moreover, $\forall t \in T_N \ {}^\bullet t \neq \emptyset \neq t^\bullet$*. In addition, for the set of* entry *places of* N*, defined as* ${}^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathsf{e}\}$*, and for the set of* exit *places of* N*, defined as* $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathsf{x}\}$*, the following conditions hold:* ${}^\circ N \neq \emptyset \neq N^\circ$ *and* ${}^\bullet({}^\circ N) = \emptyset = ({}^\circ N)^\bullet$*.*

A dtsd-box is *plain* if $\forall t \in T_N \ \exists (\alpha, \kappa) \in \mathcal{SDL} \ \Lambda_N(t) = \varrho_{(\alpha,\kappa)}$, where $\varrho_{(\alpha,\kappa)} = \{(\emptyset, (\alpha, \kappa))\}$ is a constant relabeling that can be identified with the activity (α, κ). The set of *waiting transitions* of a plain dtsd-box N is defined as $Tw_N = \{t \in T_N \mid \Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)},\ \theta \in \mathbb{N}_{\geq 1},\ l \in \mathbb{R}_{>0}\}$.

A *(timer-)clocked plain dtsd-box* is a pair (N, V), where $N = (P_N, T_N, W_N, \Lambda_N)$ is a plain dtsd-box and $V : Tw_N \to \mathbb{N}_{\geq 1} \cup \{\infty\}$ is a *timer valuation function* of the waiting transitions of N, such that $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$ (we say that the transition t *corresponds* to the activity (α, κ) in such a case) it holds $V(t) \in \{1, \ldots, \theta\} \cup \{\infty\}$.

A *marked and (timer-)clocked plain dtsd-box* is a pair (N, Q), where N is a plain dtsd-box and $Q = (M, V)$ is its *state*. Here $M \in \mathbb{N}_{fin}^{P_N}$ is a *marking* of N and $V : Tw_N \to \mathbb{N}_{\geq 1} \cup \{\infty\}$ is a *timer valuation function* of the waiting transitions of N, such that $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$ it holds $V(t) \in \{1, \ldots, \theta\} \cup \{\infty\}$ and $V(t) < \infty$, if $t \in Tw_N \cap Ena(M)$.

Let (N, Q) be a marked and clocked plain dtsd-box. By the definition above, $\forall t \in Tw_N \cap Ena(M)\ V(t) < \infty$, i.e. all enabled at M waiting transitions have finite timer values. Note that for some $t \in Tw_N \setminus Ena(M)$ we may have $V(t) < \infty$, which is allowed in the "incomplete" box specifications for the reason of compositionality, by assuming that t will be enabled at an "extended" marking of the "complete" box specification. The state $Q = (M, V)$ is *consistent*, if $\forall t \in Tw_N \setminus Ena(M)\ V(t) = \infty$, i.e. all non-enabled at M waiting transitions have infinite timer values. It is assumed that the "complete" box specification always has consistent states, i.e. that the underlying markings of those states are "large" enough to make enabled all waiting transitions with finite timer values, thus leaving the infinite timer values just for the non-enabled waiting transitions. A plain dtsd-box $N = (P_N, T_N, W_N, \Lambda_N)$ can be seen as a clocked plain dtsd-box (N, V^∞), where $\forall t \in Tw_N\ V^\infty(t) = \infty$, i.e. $V^\infty \equiv \infty$. Next, a clocked plain dtsd-box (N, V) can be treated as a marked and clocked plain dtsd-box $(N, (\emptyset, V))$. Thus, a plain dtsd-box N can be interpreted as a marked and clocked plain dtsd-box $(N, (\emptyset, V^\infty))$.

Let (N, V) be a clocked plain dtsd-box. We denote $\overline{(N, V)} = (N, Q_{\overline{(N,V)}})$, where $Q_{\overline{(N,V)}} = ({}^\circ N, V_{\overline{(N,V)}})$ and $V_{\overline{(N,V)}} : Tw_N \to \mathbb{N}_{\geq 1} \cup \{\infty\}$ is such that $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

$$V_{\overline{(N,V)}}(t) = \begin{cases} \min\{V(t), \theta\}, & t \in Tw_N \cap Ena({}^\circ N); \\ V(t), & t \in Tw_N \setminus Ena({}^\circ N). \end{cases}$$

By definition of the timer valuation function, $\forall t \in Tw_N\ (V(t) \leq \theta) \vee (V(t) = \infty)$. Hence, we may have $V(t) > \theta$ only in case $V(t) = \infty$. The definition above implies $V_{\overline{(N,V)}}(t) < \infty$ for every $t \in Tw_N \cap Ena({}^\circ N)$. Thus, $\overline{(N, V)}$ is a marked and clocked plain dtsd-box.

We also denote $\underline{(N, V)} = (N, Q_{\underline{(N,V)}})$, where $Q_{\underline{(N,V)}} = (N^\circ, V^\infty)$. Since $Ena(N^\circ) = \emptyset$, one can see that $\underline{(N, V)}$ is a marked and clocked plain dtsd-box. We call ${}^\circ N$ and N° the *entry* and *exit markings* of N, respectively.

Note that a marked and clocked plain dtsd-box $(P_N, T_N, W_N, \Lambda_N, Q)$ with the consistent state Q can be interpreted as the LDTSDPN $(P_N, T_N, W_N, D_N, \Omega_N, \mathcal{L}_N, Q)$, where the functions D_N, Ω_N and $\mathcal{L}_N$ are defined as follows: $\forall t \in T_N$ with $\Lambda_N(t) = \varrho_{(\alpha,\kappa)}$ it holds $\Omega_N(t) = \kappa$ if $\kappa \in (0; 1)$; or $D_N(t) = \theta$, $\Omega_N(t) = l$ if $\kappa = \natural_l^\theta$, $\theta \in \mathbb{N}$, $l \in \mathbb{R}_{>0}$; and $\mathcal{L}_N(t) = \alpha$. Behaviour of the marked and clocked dtsd-boxes with consistent states follows from the firing rule of LDTSDPNs. A plain dtsd-box N is *n-bounded* ($n \in \mathbb{N}$) if $\overline{N}$ is so, i.e. $\forall Q = (M, V) \in RS(\overline{N})\ \forall p \in P_N\ M(p) \leq n$, and it is *safe* if it is 1-bounded. A plain dtsd-box N is *clean* if $\forall Q = (M, V) \in RS(\overline{N})\ {}^\circ N \subseteq M \ \Rightarrow\ M = {}^\circ N$ and $N^\circ \subseteq M \ \Rightarrow\ M = N^\circ$, i.e. if there are tokens in all its entry (exit) places then no other places have tokens.

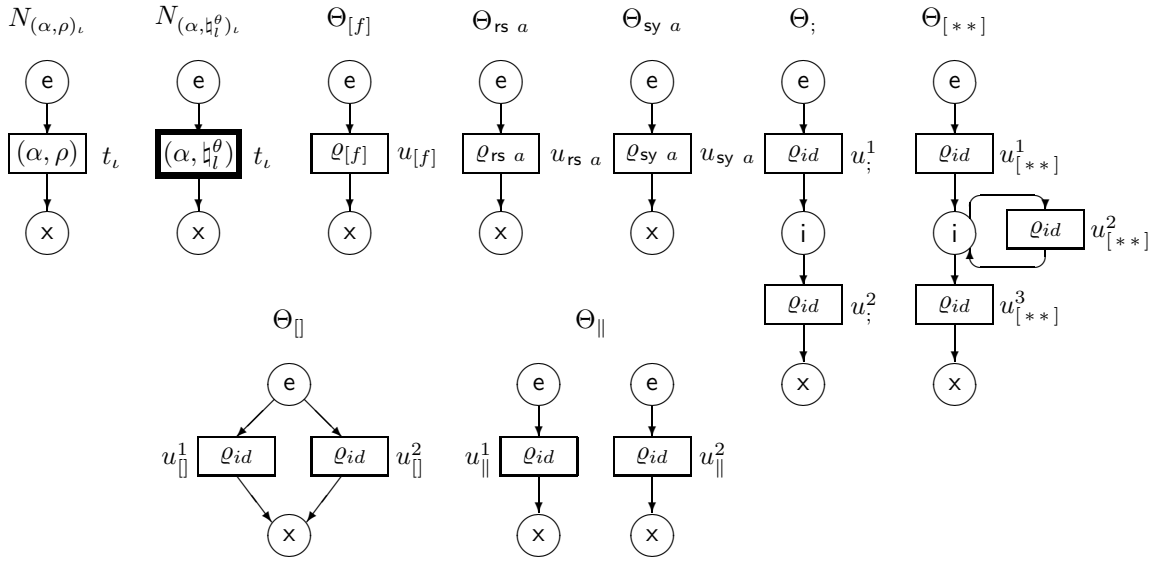

Figure 15: The plain and operator dtsd-boxes

The structure of the plain dtsd-box corresponding to a static expression without timer value superscripts is constructed like in PBC [41, 40], i.e. we use simultaneous refinement and relabeling meta-operator (net refinement) in addition to the *operator dtsd-boxes* corresponding to the algebraic operations of dtsdPBC and featuring transformational transition relabelings. Operator dtsd-boxes specify n-ary functions from plain dtsd-boxes to plain dtsd-boxes (we have $1 \leq n \leq 3$ in dtsdPBC). Thus, as we shall see in Theorem 4.1, the resulting plain dtsd-boxes are safe and clean. In the definition of the denotational semantics, we shall apply standard constructions used for PBC. Let Θ denote *operator box* and u denote *transition name* from the PBC setting.

The relabeling relations $\varrho \subseteq I\!N_{fin}^{\mathcal{SDL}} \times \mathcal{SDL}$ are defined as follows:

- $\varrho_{id} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SDL}\}$ is the *identity relabeling* keeping the interface as it is;
- $\varrho_{(\alpha,\kappa)} = \{(\emptyset,(\alpha,\kappa))\}$ is the *constant relabeling* that can be identified with $(\alpha,\kappa) \in \mathcal{SDL}$ itself;
- $\varrho_{[f]} = \{(\{(\alpha,\kappa)\},(f(\alpha),\kappa)) \mid (\alpha,\kappa) \in \mathcal{SDL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SDL},\ a,\hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling relation containing ϱ_{id} such that if $(\Upsilon,(\alpha,\kappa)),(\Xi,(\beta,\lambda)) \in \varrho_{\mathsf{sy}\ a}$ and $a \in \alpha,\ \hat{a} \in \beta$ then
  - $(\Upsilon + \Xi,(\alpha \oplus_a \beta,\kappa \cdot \lambda)) \in \varrho_{\mathsf{sy}\ a}$ if $\kappa,\lambda \in (0;1)$;
  - $(\Upsilon + \Xi,(\alpha \oplus_a \beta,\natural_{l+m}^\theta)) \in \varrho_{\mathsf{sy}\ a}$ if $\kappa = \natural_l^\theta,\ \lambda = \natural_m^\theta,\ \theta \in I\!N,\ l,m \in I\!R_{>0}$.

The plain dtsd-boxes $N_{(\alpha,\rho)_\iota},\ N_{(\alpha,\natural_l^\theta)_\iota}$, where $\rho \in (0;1)$, $\theta \in I\!N$, $l \in I\!R_{>0}$, and operator dtsd-boxes are presented in Figure 15. Note that the label i of internal places is usually omitted.

In the case of the iteration, a decision that we must take is the selection of the operator box that we shall use for it, since we have two proposals in plain PBC for that purpose [40]. One of them provides us with a safe version with six transitions in the operator box, but there is also a simpler version, which has only three transitions. In general, in PBC, with the latter version we may generate 2-bounded nets, which only occurs when a parallel behavior appears at the highest level of the body of the iteration. Nevertheless, in our case, and due to the syntactical restriction introduced for regular terms, this particular situation cannot occur, so that the net obtained will be always safe.

Let $(N_i,V_i) = (P_{N_i},T_{N_i},W_{N_i},\Lambda_{N_i},V_i)$ $(1 \leq i \leq 3)$ be clocked plain dtsd-boxes. The operator dtsd-boxes are extended so that they will specify the n-ary functions from/to *clocked* plain dtsd-boxes, as follows.

- $\Theta_\circ((N_1,V_1),(N_2,V_2)) = (\Theta_\circ(N_1,N_2),V),\ \circ \in \{;,[],\|\}$, where

$$V(t) = \begin{cases} V_1(t), & t \in T_{N_1}; \\ V_2(t), & t \in T_{N_2}. \end{cases}$$

- $\Theta_{[f]}(N_1, V_1) = (\Theta_{[f]}(N_1), V)$, where

$$V(t) = V_1(t),\ t \in T_{N_1}.$$

- $\Theta_{\mathsf{rs}\ a}(N_1, V_1) = (\Theta_{\mathsf{rs}\ a}(N_1), V)$, where

$$V(t) = V_1(t),\ t \in T_{N_1},\ a, \hat{a} \notin \alpha,\ \Lambda_{N_1}(t) = \varrho_{(\alpha,\kappa)}.$$

- $\Theta_{\mathsf{sy}\ a}(N_1, V_1) = (\Theta_{\mathsf{sy}\ a}(N_1), V)$, where

$$V(t) = \begin{cases} V_1(t), & t \in Tw_{N_1}; \\ \max\{V_1(v), V_1(w)\}, & t \text{ results from synchronization of } v, w \in Tw_{N_1}. \end{cases}$$

- $\Theta_{[**]}((N_1, V_1), (N_2, V_1), (N_3, V_1)) = (\Theta_{[**]}(N_1, N_2, N_3), V)$, where

$$V(t) = \begin{cases} V_1(t), & t \in T_{N_1}; \\ V_2(t), & t \in T_{N_2}; \\ V_3(t), & t \in T_{N_3}. \end{cases}$$

To define a semantic function that assigns a clocked plain dtsd-box to every static expression of dtsdPBC, we introduce the *enumeration* function $Enu : T \to Num$, which associates the numberings with transitions of a clocked plain dtsd-box $N = (P, T, W, \Lambda, V)$ in accordance with those of activities. In the case of synchronization, the function associates with the resulting new transition a concatenation of the parenthesized numberings of the transitions it comes from.

We now define the enumeration function Enu for every operator of dtsdPBC. Let $Box_{dtsd}(E) = (N_E, V_E) = (P_E, T_E, W_E, \Lambda_E, V_E)$ be the clocked plain dtsd-box corresponding to a static expression E, and $Enu_E : T_E \to Num$ be the enumeration function for (N_E, V_E). We use the analogous notation for static expressions F and K.

- $Box_{dtsd}((\alpha, \rho)_\iota) = (N_{(\alpha,\rho)_\iota}, \emptyset)$. Since a single transition t_ι corresponds to the activity $(\alpha, \rho)_\iota \in \mathcal{SL}$, their numberings coincide:

$$Enu(t_\iota) = \iota.$$

- $Box_{dtsd}((\alpha, \natural_l^0)_\iota) = (N_{(\alpha,\natural_l^0)_\iota}, \emptyset)$. Since a single transition t_ι corresponds to the activity $(\alpha, \natural_l^0)_\iota \in \mathcal{IL}$, their numberings coincide:

$$Enu(t_\iota) = \iota.$$

- $Box_{dtsd}((\alpha, \natural_l^\theta)_\iota) = (N_{(\alpha,\natural_l^\theta)_\iota}, (t_\iota, \infty))$. Since a single transition t_ι corresponds to the activity $(\alpha, \natural_l^\theta)_\iota \in \mathcal{WL}$, their numberings coincide:

$$Enu(t_\iota) = \iota.$$

- $Box_{dtsd}((\alpha, \natural_l^\theta)_\iota^\delta) = (N_{(\alpha,\natural_l^\theta)_\iota}, (t_\iota, \delta))$. Since a single transition t_ι corresponds to the activity $(\alpha, \natural_l^\theta)_\iota \in \mathcal{WL}$, their numberings coincide:

$$Enu(t_\iota) = \iota.$$

- $Box_{dtsd}(E \circ F) = \Theta_\circ(Box_{dtsd}(E), Box_{dtsd}(F)),\ \circ \in \{;, [], \|\}$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F. \end{cases}$$

- $Box_{dtsd}(E[f]) = \Theta_{[f]}(Box_{dtsd}(E))$. Since we only replace the labels of some multiactions by a bijection, we preserve the initial numbering:

$$Enu(t) = Enu_E(t),\ t \in T_E.$$

- $Box_{dtsd}(E \text{ rs } a) = \Theta_{\text{rs } a}(Box_{dtsd}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, this does not change the numbering of the remaining transitions:

$$Enu(t) = Enu_E(t),\ t \in T_E,\ a, \hat{a} \notin \alpha,\ \Lambda_E(t) = \varrho_{(\alpha,\kappa)}.$$

- $Box_{dtsd}(E \text{ sy } a) = \Theta_{\text{sy } a}(Box_{dtsd}(E))$. Note that $\forall v, w \in T_E$ such that $\Lambda_E(v) = \varrho_{(\alpha,\kappa)},\ \Lambda_E(w) = \varrho_{(\beta,\lambda)}$ and $a \in \alpha,\ \hat{a} \in \beta$, the new transition t resulting from synchronization of v and w has the label $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \kappa \cdot \lambda)}$ if t is a stochastic transition ($\kappa, \lambda \in (0;1)$); or $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \natural^\theta_{l+m})}$ if t is a deterministic one ($\kappa = \natural^\theta_l,\ \lambda = \natural^\theta_m,\ \theta \in \mathbb{N},\ l, m \in \mathbb{R}_{>0}$); and the numbering $Enu(t) = (Enu_E(v))(Enu_E(w))$.

  Thus, the enumeration function is defined as

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ (Enu_E(v))(Enu_E(w)), & t \text{ results from synchronization of } v \text{ and } w. \end{cases}$$

  According to the definition of $\varrho_{\text{sy } a}$, the synchronization is only possible when all the transitions in the set are stochastic (immediate or waiting, respectively). If we synchronize the same set of transitions in different orders, we obtain several resulting transitions with the same label and probability or weight, but with the different numberings having the same content. Then, we only consider a single transition from the resulting ones in the clocked plain dtsd-box to avoid introducing redundant transitions.

  For example, if the transitions t and u are generated by synchronizing v and w in different orders, we have $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \kappa \cdot \lambda)} = \Lambda(u)$ for stochastic transitions ($\kappa, \lambda \in (0;1)$) or $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \natural^\theta_{l+m})} = \Lambda(u)$ for deterministic ones ($\kappa = \natural^\theta_l,\ \lambda = \natural^\theta_m,\ \theta \in \mathbb{N},\ l, m \in \mathbb{R}_{>0}$), but $Enu(t) = (Enu_E(v))(Enu_E(w)) \neq (Enu_E(w))(Enu_E(v)) = Enu(u)$, whereas $Cont(Enu(t)) = Cont(Enu(v)) \cup Cont(Enu(w)) = Cont(Enu(u))$. Then only one transition t (or u, symmetrically) will appear in $Box_{dtsd}(E \text{ sy } a)$.

- $Box_{dtsd}([E * F * K]) = \Theta_{[**]}(Box_{dtsd}(E), Box_{dtsd}(F), Box_{dtsd}(K))$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F; \\ Enu_K(t), & t \in T_K. \end{cases}$$

We now can formally define the denotational semantics as a homomorphism.

Definition 4.5 *Let* $(\alpha, \rho) \in \mathcal{SL},\ (\alpha, \natural^0_l) \in \mathcal{IL},\ (\alpha, \natural^\theta_l) \in \mathcal{WL},\ \delta \in \{1, \ldots, \theta\},\ a \in Act$ *and* $E, F, K \in RegStatExpr$. *The* denotational semantics *of dtsdPBC is a mapping* Box_{dtsd} *from* $RegStatExpr$ *into the domain of clocked plain dtsd-boxes, defined as follows:*

1. $Box_{dtsd}((\alpha, \rho)_\iota) = (N_{(\alpha,\rho)_\iota}, \emptyset)$;
2. $Box_{dtsd}((\alpha, \natural^0_l)_\iota) = (N_{(\alpha,\natural^0_l)_\iota}, \emptyset)$;
3. $Box_{dtsd}((\alpha, \natural^\theta_l)_\iota) = (N_{(\alpha,\natural^\theta_l)_\iota}, (t_\iota, \infty))$;
4. $Box_{dtsd}((\alpha, \natural^\theta_l)^\delta_\iota) = (N_{(\alpha,\natural^\theta_l)_\iota}, (t_\iota, \delta))$;
5. $Box_{dtsd}(E \circ F) = \Theta_\circ(Box_{dtsd}(E), Box_{dtsd}(F)),\ \circ \in \{;, [], \|\}$;
6. $Box_{dtsd}(E[f]) = \Theta_{[f]}(Box_{dtsd}(E))$;
7. $Box_{dtsd}(E \circ a) = \Theta_{\circ a}(Box_{dtsd}(E)),\ \circ \in \{\text{rs}, \text{sy}\}$;
8. $Box_{dtsd}([E * F * K]) = \Theta_{[***]}(Box_{dtsd}(E), Box_{dtsd}(F), Box_{dtsd}(K))$.

The marked and clocked dtsd-boxes of dynamic expressions can be defined as well. For $E \in RegStatExpr$, let $Box_{dtsd}(\overline{E}) = \overline{Box_{dtsd}(E)}$ and $Box_{dtsd}(\underline{E}) = \underline{Box_{dtsd}(E)}$. Note that this definition is compositional in the sense that, for any arbitrary dynamic expression, we may decompose it in some inner dynamic and static expressions, for which we may apply the definition, thus obtaining the corresponding clocked plain dtsd-boxes, which can be joined according to the term structure (by definition of Box_{dtsd}), the resulting clocked plain box being marked in the places that were marked in the argument nets.

Importantly, when composing marked and clocked dtsd-boxes of arbitrary dynamic expressions, we should guarantee that the operations correctly propagate the timer values from the clocked to non-clocked operands. For that, we have to respect the time spent in the entry markings and delays of the waiting transitions, which become enabled at them when composing. The main idea is that the timer values in the composite marked and clocked dtsd-boxes should be as close as possible to those in the substituent marked and clocked dtsd-boxes, whose waiting transition timers should sometimes be decreased to maintain the time progress uniformity in the resulting composition.

Let $E, F \in RegStatExpr,\ G, H \in RegDynExpr$ and $a \in Act$. Then $Box_{dtsd}(E) = (P_E, T_E, W_E, \Lambda_E, V_E) = (N_E, V_E)$ is the clocked plain dtsd-box of E, and analogously for F. The marked and clocked plain dtsd-box of G is $Box_{dtsd}(G) = (N_G, (M_G, V_G))$ (defined by induction on the structure of G, as will be descried below), and similarly for H. Next, $Box_{dtsd}(\overline{E}) = \overline{(N_E, V_E)} = (N_E, (^\circ N_E, V_{\overline{E}}))$ is the marked and clocked plain dtsd-box of $\overline{E}$, and analogously for $\overline{F}$. Thus, $\forall t \in Tw_E$ with $\Lambda_E(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

$$V_{\overline{E}}(t) = \begin{cases} \min\{V_E(t), \theta\}, & t \in Tw_E \cap Ena(^\circ N_E); \\ V_E(t), & t \in Tw_E \setminus Ena(^\circ N_E). \end{cases}$$

Also, $Box_{dtsd}(\underline{E}) = \underline{(N_E, V_E)} = (N_E, (N_E^\circ, V^\infty))$ is the marked and clocked plain dtsd-box of $\underline{E}$, and similarly for $\underline{F}$.

Let N, N' be two plain dtsd-boxes and $p \in {}^\circ N \cup N^\circ,\ p' \in {}^\circ N' \cup N'^\circ$ be their respective entry or exit places. Then $(p, p') \in ({}^\circ N \cup N^\circ) \times ({}^\circ N' \cup N'^\circ)$ denotes the merging of p and p' in the composed plain dtsd-box such that (p, p') inherits all their connectivities from the net structures of N and N'.

Let $(N, (M, V))$ be a marked and clocked plain dtsd-box, where $T = Ts \uplus Ti \uplus Tw$ consists of stochastic, immediate and waiting transitions. The *marking age of the state* (M, V) is defined as

$$\square(M, V) = \max\{\eta - V(u) \mid u \in Tw \cap Ena(M),\ \Lambda(u) = \varrho_{(\beta, \natural_m^\eta)}\}.$$

We now inductively define the dtsd-boxes of arbitrary dynamic expressions.

- $Box_{dtsd}(\overline{E}) = \overline{Box_{dtsd}(E)}$ and $Box_{dtsd}(\underline{E}) = \underline{Box_{dtsd}(E)}$.
- $Box_{dtsd}(G; E) = (Box_{dtsd}(\lfloor G \rfloor; E), (M, V))$, where $M = \begin{cases} M_G, & M_G \neq N_G^\circ; \\ N_G^\circ \times {}^\circ N_E, & M_G = N_G^\circ; \end{cases}$ and $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

  $$V(t) = \begin{cases} V_G(t), & t \in Tw_G; \\ \min\{V_E(t), \theta\}, & t \in Tw_E \cap Ena(M); \\ V_E(t), & t \in Tw_E \setminus Ena(M). \end{cases}$$

  Thus, each waiting transition of N_E enabled at the entry marking of it has set its timer to $\min\{V_E(t), \theta\}$.
- $Box_{dtsd}(E; G) = (Box_{dtsd}(E; \lfloor G \rfloor), (M, V))$, where $M = \begin{cases} M_G, & M_G \neq {}^\circ N_G; \\ N_E^\circ \times {}^\circ N_G, & M_G = {}^\circ N_G; \end{cases}$ and $\forall t \in Tw_N$:

  $$V(t) = \begin{cases} V_E(t), & t \in Tw_E; \\ V_G(t), & t \in Tw_G. \end{cases}$$
- $Box_{dtsd}(G[]E) = (Box_{dtsd}(\lfloor G \rfloor [] E), (M, V))$, where $M = \begin{cases} M_G, & (M_G \neq {}^\circ N_G) \wedge (M_G \neq N_G^\circ); \\ {}^\circ N_G \times {}^\circ N_E, & M_G = {}^\circ N_G; \\ N_G^\circ \times N_E^\circ, & M_G = N_G^\circ; \end{cases}$

  and $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

  $$V(t) = \begin{cases} \theta - \min\{\square(M_G, V_G), \square(^\circ N_E, V_{\overline{E}})\}, & ((t \in Tw_G \cap Ena(M)) \wedge (M_G = {}^\circ N_G)) \vee \\ & (t \in Tw_E \cap Ena(M)); \\ V_G(t), & ((t \in Tw_G \cap Ena(M)) \wedge (M_G \neq {}^\circ N_G)) \vee \\ & (t \in Tw_G \setminus Ena(M)); \\ V_E(t), & t \in Tw_E \setminus Ena(M). \end{cases}$$

  Thus, if ζ is the minimum of the times spent at the markings of the states (M_G, V_G), such that $M_G = {}^\circ N_G$, and $(^\circ N_E, V_{\overline{E}})$ then each waiting transition, enabled at the marking M, has set its timer to $\theta - \zeta$, where θ is the delay of that transition. The idea is to ensure that the time progresses uniformly, for which the timer decrements of all waiting transitions, enabled at M, should be synchronized (equalized). Hence, the subnet with the more time spent in its local marking should "wait" for the other subnet by modifying appropriately (via increasing by the difference between residence times at M_G and ${}^\circ N_E$) the timer values of its waiting transitions, enabled at M.

Note that $\Box(M_G,V_G)\neq\Box(^\circ N_E,V_{\overline{E}})$ cannot hold for any dynamic expression, obtained by applying action rules, starting from an overlined static expression without timer value superscripts. The reason is that all the action rules maintain the time progress uniformity, hence, $\zeta=\Box(M_G,V_G)=\Box(^\circ N_E,V_{\overline{E}})$ in that case. Further, the inequality $\eta-V_G(u)<\Box(M_G,V_G)$ may only happen when the $(\beta,\natural^\eta_m)\in\mathcal{WL}(G)$, corresponding to $u\in Tw_G\cap Ena(M_G)$, is later affected by restriction, so that the timer of that waiting multiaction stops with the value 1 while the waiting multiaction can never be executed. The same holds for $\Box(^\circ N_E,V_{\overline{E}})$. Thus, if we start from an overlined static expression without time stamps and the waiting multiaction corresponding to t is not subsequently affected by restriction then $V(t)=\theta-\Box(M_G,V_G)=V_G(t)$ for $(t\in Tw_G\cap Ena(M))\wedge(M_G=\,^\circ N_G)$ and $V(t)=\theta-\Box(^\circ N_E,V_{\overline{E}})=\min\{V_E(t),\theta\}$ for $t\in Tw_E\cap Ena(M)$, i.e. $V(t)$ is defined like that for the case $Box_{dtsd}(G;E)$.

The definition of $Box_{dtsd}(E[]G)$ is similar.

- $Box_{dtsd}(G\|H)=(Box_{dtsd}(\lfloor G\rfloor\|\lfloor H\rfloor),(M,V))$, where $M=M_G\cup M_H$, and $\forall t\in Tw_N$ with $\Lambda_N(t)=\varrho_{(\alpha,\natural^\theta_l)}$:

$$V(t)=\begin{cases}\theta-\min\{\Box(M_G,V_G),\Box(M_H,V_H)\}, & t\in(Tw_G\cup Tw_H)\cap Ena(M);\\ V_G(t), & t\in Tw_G\setminus Ena(M);\\ V_H(t), & t\in Tw_H\setminus Ena(M).\end{cases}$$

Thus, if ζ is the minimum of the times spent at the markings of the states (M_G,V_G) and (M_H,V_H) then each waiting transition, enabled at the marking M, has set its timer to $\theta-\zeta$, where θ is the delay of that transition. The idea is to ensure that the time progresses uniformly, for which the timer decrements of all waiting transitions, enabled at M, should be synchronized (equalized). Hence, the subnet with the more time spent in its local marking should "wait" for the other subnet by modifying appropriately (via increasing by the difference between residence times at M_G and M_H) the timer values of its waiting transitions, enabled at M.

Note that $\Box(M_G,V_G)\neq\Box(M_H,V_H)$ cannot hold for any dynamic expression, obtained by applying action rules, starting from an overlined static expression without timer value superscripts. The reason is that all the action rules maintain the time progress uniformity, hence, $\zeta=\Box(M_G,V_G)=\Box(M_H,V_H)$ in that case. Further, the inequality $\eta-V_G(u)<\Box(M_G,V_G)$ may only happen when the $(\beta,\natural^\eta_m)\in\mathcal{WL}(G)$, corresponding to $u\in Tw_G\cap Ena(M_G)$, is later affected by restriction, so that the timer of that waiting multiaction stops with the value 1 while the waiting multiaction can never be executed. The same holds for $\Box(M_H,V_H)$. Thus, if we start from an overlined static expression without time stamps and the waiting multiaction corresponding to t is not subsequently affected by restriction then $V(t)=\theta-\Box(M_G,V_G)=V_G(t)$ for $t\in Tw_G\cap Ena(M)$ and $V(t)=\theta-\Box(M_H,V_H)=V_H(t)$ for $t\in Tw_H\cap Ena(M)$, i.e. $V(t)$ is defined like that for the case $Box_{dtsd}(E;G)$, if to replace E with H in the syntax of that definition.

- $Box_{dtsd}(G[f])=(Box_{dtsd}(\lfloor G\rfloor[f]),(M,V))$, where $M=M_G$, and $\forall t\in Tw_N$:

$$V(t)=\ V_G(t),\quad t\in Tw_G.$$

- $Box_{dtsd}(G\ \mathsf{rs}\ a)=(Box_{dtsd}(\lfloor G\rfloor\ \mathsf{rs}\ a),(M,V))$, where $M=M_G$, and $\forall t\in Tw_N$:

$$V(t)=\ V_G(t),\quad t\in Tw_G,\ a,\hat{a}\notin\alpha.$$

- $Box_{dtsd}(G\ \mathsf{sy}\ a)=(Box_{dtsd}(\lfloor G\rfloor\ \mathsf{sy}\ a),(M,V))$, where $M=M_G$, and $\forall t\in Tw_N$:

$$V(t)=\begin{cases}V_G(t), & t\in Tw_G;\\ \max\{V_G(v),V_G(w)\}, & t\text{ results from synchronization of }v,w\in Tw_G.\end{cases}$$

Thus, the timer for the synchronous product of the waiting transitions v and w from N_G is set to maximum of their timer values. This means that we wait for the latest (being delayed for some reason) of the two synchronized transitions, since their synchronous product cannot fire until they both can fire. If at least one of the timers of v and w has the undefined value ∞ (i.e. the corresponding transition is not enabled at M_G) then the result of their synchronization also has the timer value ∞, since both the synchronized transitions must be enabled at M_G in order to enable their synchronous product.

- $Box_{dtsd}([G*E*F])=(Box_{dtsd}(\lfloor G\rfloor*E*F),(M,V))$, where $M=\begin{cases}M_G, & M_G\neq N^\circ_G;\\ N^\circ_G\times(^\circ N_E\times N^\circ_E)\times{}^\circ N_F), & M_G=N^\circ_G;\end{cases}$ and $\forall t\in Tw_N$ with $\Lambda_N(t)=\varrho_{(\alpha,\natural^\theta_l)}$:

$$V(t) = \begin{cases} V_G(t), & t \in Tw_G; \\ \theta - \min\{\Box(M_E, V_E), \Box(M_F, V_F)\}, & t \in (Tw_E \cup Tw_F) \cap Ena(M); \\ V_E(t), & t \in Tw_E \setminus Ena(M); \\ V_F(t), & t \in Tw_F \setminus Ena(M). \end{cases}$$

Thus, if ζ is the minimum of the times spent at the markings of the states (M_E, V_E) and (M_F, V_F) then each waiting transition, enabled at the marking M, has set its timer to $\theta - \zeta$, where θ is the delay of that transition. The idea is to ensure that the time progresses uniformly, for which the timer decrements of all waiting transitions, enabled at M, should be synchronized (equalized). Hence, the subnet with the more time spent in its local marking should "wait" for the other subnet by modifying appropriately (via increasing by the difference between residence times at M_E and M_F) the timer values of its waiting transitions, enabled at M.

- $Box_{dtsd}([E * G * F]) = (Box_{dtsd}(E * \lfloor G \rfloor * F), (M, V))$, where $M = \begin{cases} M_G, & (M_G \neq {}^\circ N_G) \wedge (M_G \neq N_G^\circ); \\ N_E^\circ \times (({}^\circ N_G \times N_G^\circ) \times {}^\circ N_F), & (M_G = {}^\circ N_G) \vee (M_G = N_G^\circ); \end{cases}$ and $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

$$V(t) = \begin{cases} \theta - \min\{\Box(M_G, V_G), \Box({}^\circ N_F, V_{\overline{F}})\}, & ((t \in Tw_G \cap Ena(M)) \wedge ((M_G = {}^\circ N_G) \vee (M_G = N_G^\circ))) \vee \\ & (t \in Tw_F \cap Ena(M)); \\ V_G(t), & ((t \in Tw_G \cap Ena(M)) \wedge (M_G \neq {}^\circ N_G) \wedge (M_G \neq N_G^\circ)) \vee \\ & (t \in Tw_G \setminus Ena(M)); \\ V_F(t), & t \in Tw_F \setminus Ena(M). \end{cases}$$

Thus, if ζ is the minimum of the times spent at the markings of the states (M_G, V_G), such that $(M_G = {}^\circ N_G) \vee (M_G = N_G^\circ)$, and $({}^\circ N_F, V_{\overline{F}})$ then each waiting transition, enabled at the marking M, has set its timer to $\theta - \zeta$, where θ is the delay of that transition. The idea is to ensure that the time progresses uniformly, for which the timer decrements of all waiting transitions, enabled at M, should be synchronized (equalized). Hence, the subnet with the more time spent in its local marking should "wait" for the other subnet by modifying appropriately (via increasing by the difference between residence times at M_G and ${}^\circ N_F$) the timer values of its waiting transitions, enabled at M.

- $Box_{dtsd}([E{*}F{*}G]) = (Box_{dtsd}(E{*}F{*}\lfloor G \rfloor), (M, V))$, where $M = \begin{cases} M_G, & M_G \neq {}^\circ N_G; \\ N_E^\circ \times (({}^\circ N_F \times N_F^\circ) \times {}^\circ N_G), & M_G = {}^\circ N_G; \end{cases}$ and $\forall t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$:

$$V(t) = \begin{cases} \theta - \min\{\Box({}^\circ N_F, V_{\overline{F}}), \Box(M_G, V_G)\}, & (t \in Tw_F \cap Ena(M)) \vee \\ & ((t \in Tw_G \cap Ena(M)) \wedge (M_G = {}^\circ N_G)); \\ V_F(t), & t \in Tw_F \setminus Ena(M); \\ V_G(t), & ((t \in Tw_G \cap Ena(M)) \wedge (M_G \neq {}^\circ N_G)) \vee \\ & (t \in Tw_G \setminus Ena(M)). \end{cases}$$

Thus, if ζ is the minimum of the times spent at the markings of the states $({}^\circ N_F, V_{\overline{F}})$ and (M_G, V_G), such that $M_G = {}^\circ N_G$, then each waiting transition, enabled at the marking M, has set its timer to $\theta - \zeta$, where θ is the delay of that transition. The idea is to ensure that the time progresses uniformly, for which the timer decrements of all waiting transitions, enabled at M, should be synchronized (equalized). Hence, the subnet with the more time spent in its local marking should "wait" for the other subnet by modifying appropriately (via increasing by the difference between residence times at ${}^\circ N_F$ and M_G) the timer values of its waiting transitions, enabled at M.

Remember that for any $H \in SatOpRegDynExpr$, all waiting multiactions from $EnaWait([H]_\approx)$ have (finite) timer value superscripts. Then for $Box_{dtsd}(H) = (N, (M, V))$ we have $\forall t \in Tw_N \cap Ena(M)\ V(t) < \infty$. Hence, if $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$ then $\min\{V(t), \theta\} = V(t)$. Suppose that H is also obtained by applying action rules, starting from an overlined static expression without timer value superscripts and the waiting multiactions corresponding to each $t \in Tw_N$ are not affected by restriction (note that the waiting multiactions affected by restriction in H have no corresponding transitions in $Box_{dtsd}(H)$). In such a case, by the remarks on the $\Box(M, V)$ function simplification in the constructions above, the timer valuation function V is obtained simply by combining those of the subformulas of G. For example, if $H = [G * E * F]$ then $V(t) = \begin{cases} V_G(t), & t \in Tw_G; \\ V_E(t), & t \in Tw_E; \\ V_F(t), & t \in Tw_F. \end{cases}$

Theorem 4.1 *For any static expression E, $Box_{dtsd}(\overline{E})$ is safe and clean.*

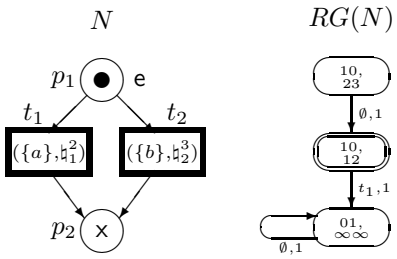

Figure 16: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = (\{a\}, \natural_1^2)[](\{b\}, \natural_2^3)$ and its reachability graph

Proof. The structure of the net is obtained as in PBC [41, 40], combining both refinement and relabeling. Consequently, the dtsd-boxes thus obtained will be safe and clean. □

Proposition 4.1 *For any static expression E without timer value superscripts, all states of $RG(Box_{dtsd}(\overline{E}))$ (i.e. those from $RS(Box_{dtsd}(\overline{E}))$) are consistent.*

Proof. Let $Box_{dtsd}(E) = (N_E, V_E)$. Since E is without timer value superscripts, $V_E = V^\infty$ and $\underline{Box_{dtsd}(E)} = (N_E, V^\infty)$. By construction of marked and clocked dtsd-boxes, we get $Box_{dtsd}(\overline{E}) = \overline{Box_{dtsd}(E)} = \overline{(N_E, V^\infty)} = (N_E, (^\circ N_E, V_{\overline{E}}))$, where $V_{\overline{E}}(t) = \begin{cases} \min\{V^\infty(t), \theta\} = \min\{\infty, \theta\} = \theta, & t \in Tw_E \cap Ena(^\circ N_E); \\ V^\infty(t) = \infty, & t \in Tw_E \setminus Ena(^\circ N_E). \end{cases}$ Thus, the initial state $(^\circ N_E, V_{\overline{E}})$ of $RG(Box_{dtsd}(\overline{E}))$ is consistent and $(N_E, (^\circ N_E, V_{\overline{E}}))$ is an LDTSDPN.

By definition of the firing rule for LDTSDPNs, the waiting transitions that are not enabled in the next state get (or keep) infinite timer values (item 2, case 1: the infinity value) while those enabled in the next state get (or keep) finite timer values (item 2, cases 2–4: the new, old or decreased by one value). Thus, the firing rule always transforms consistent states into consistent ones. Since the initial state of $RG(Box_{dtsd}(\overline{E}))$ is consistent and the subsequent states are added according to the firing rule, all states of $RG(Box_{dtsd}(\overline{E}))$ are consistent. □

4.3 Examples of dtsd-boxes

We now present a series of examples that demonstrate how to construct the dtsd-boxes of the dynamic expressions that include various compositions of stochastic, waiting and immediate multiactions. In the reachability graphs of the dtsd-boxes, the s-tangible and w-tangible states are depicted in ordinary and double ovals, respectively, and the vanishing ones are depicted in boxes. To simplify the graphical representation, the singleton sets of transitions are written without outer braces.

Example 4.1 *Let E be from Example 3.12. In Figure 16, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_1), V(t_2))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.2 *Let E be from Example 3.13. In Figure 17, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the one-element vector (scalar) $V = V(t_1)$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.3 *Let E be from Example 3.14. In Figure 18, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. Since N has no waiting transitions (a single waiting multiaction in E is affected by restriction), we may consider the substituent markings M as the whole states $Q = (M, \varepsilon) \in RS(N)$, where ε is the zero-element vector (empty sequence). Note that $TS(\overline{E})$ and $RG(N)$ are not isomorphic, but bisimilar (i.e. related by step stochastic bisimulation equivalence, to be defined later).*

Example 4.4 *Let E be from Example 3.15. In Figure 19, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the one-element vector (scalar) $V = V(t_2)$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

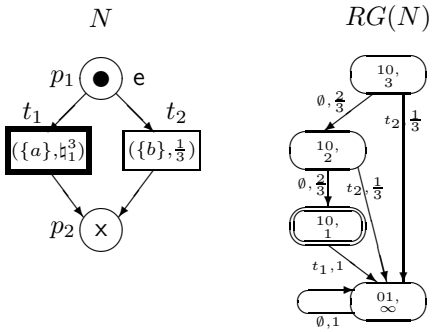

Figure 17: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = (\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3})$ and its reachability graph

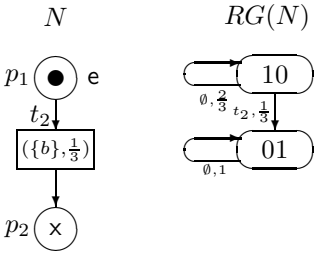

Figure 18: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = ((\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3}))$ rs a and its reachability graph

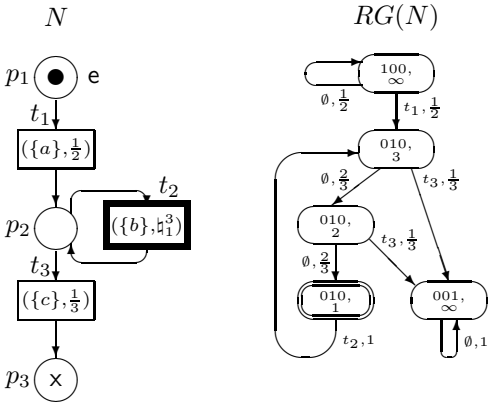

Figure 19: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = [(\{a\}, \frac{1}{2}) * (\{b\}, \natural_1^3) * (\{c\}, \frac{1}{3})]$ and its reachability graph

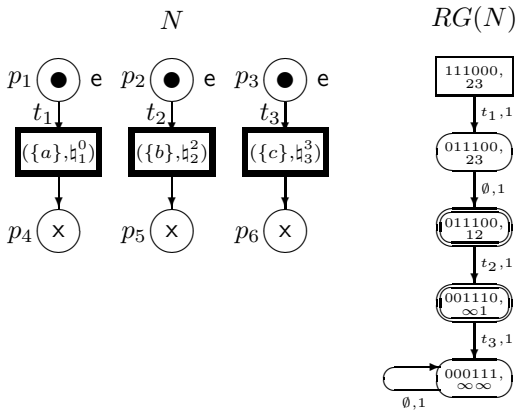

Figure 20: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = (\{a\}, \natural_1^0) \| (\{b\}, \natural_2^2) \| (\{c\}, \natural_3^3)$ and its reachability graph

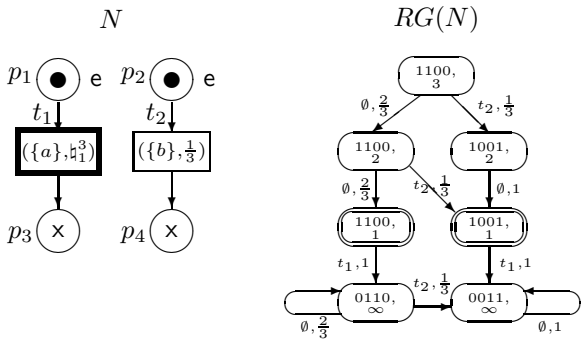

Figure 21: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = (\{a\}, \natural_1^3) \| (\{b\}, \frac{1}{3})$ and its reachability graph

Example 4.5 *Let E be from Example 3.16. In Figure 20, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_2), V(t_3))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.6 *Let E be from Example 3.17. In Figure 21, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the one-element vector (scalar) $V = V(t_1)$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.7 *Let E be from Example 3.18. In Figure 22, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the one-element vector (scalar) $V = V(t_{(1)(2)})$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.8 *Let E be from Example 3.19. In Figure 23, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_1), V(t_2), V(t_{(2)(3)}), V(t_3))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.9 *Let E be from Example 3.20. In Figure 24, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_1), V(t_4))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

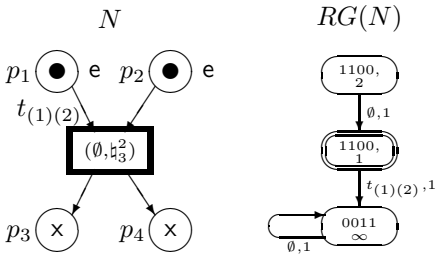

Figure 22: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = (\!|\{a\},\lightning_1^2|\!)\|(\!|\{\hat{a}\},\lightning_2^2|\!))$ sy a rs a and its reachability graph

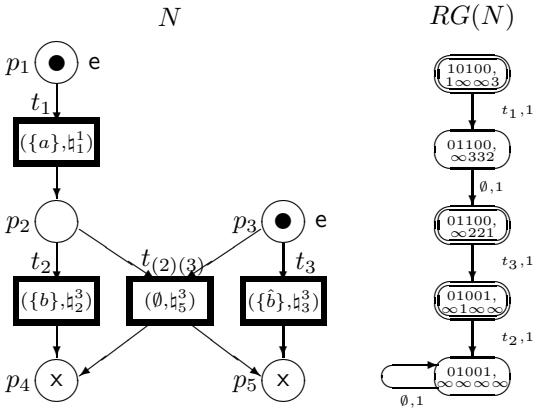

Figure 23: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = ((\!|\{a\},\lightning_1^1|\!);(\!|\{b\},\lightning_2^3|\!))\|(\!|\{\hat{b}\},\lightning_3^3|\!))$ sy b and its reachability graph

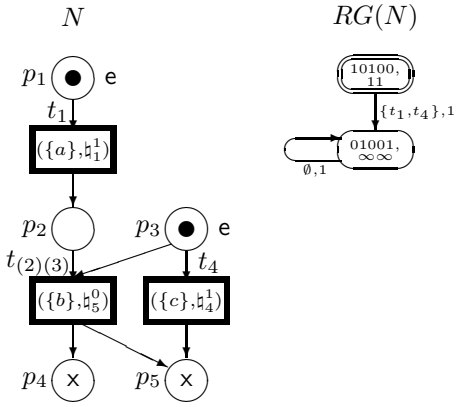

Figure 24: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = ((\!|\{a\},\lightning_1^1|\!);(\!|\{b,\hat{x}\},\lightning_2^0|\!))\|((\!|\{x\},\lightning_3^0|\!)[]$ $(\!|\{c\},\lightning_4^1|\!)))$ sy x rs x and its reachability graph

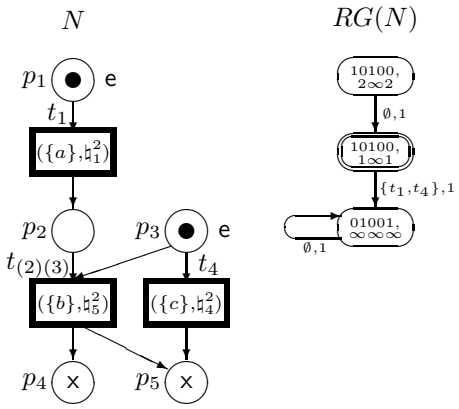

Figure 25: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = ((((\{a\}, \natural_1^2); (\{b, \hat{x}\}, \natural_2^2)) \| ((\{x\}, \natural_3^2) [] (\{c\}, \natural_4^2)))$ sy x rs x and its reachability graph

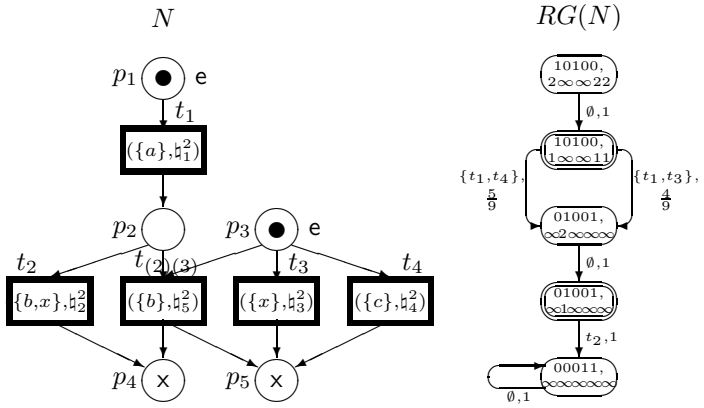

Figure 26: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = ((((\{a\}, \natural_1^2); (\{b, \hat{x}\}, \natural_2^2)) \| ((\{x\}, \natural_3^2) [] (\{c\}, \natural_4^2)))$ sy x and its reachability graph

Example 4.10 *Let E be from Example 3.21. In Figure 25, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_1), V(t_{(2)(3)}), V(t_4))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are not isomorphic, but bisimilar (i.e. related by step stochastic bisimulation equivalence, to be defined later).*

Example 4.11 *Let E be from Example 3.22. In Figure 26, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_1), V(t_2), V(t_{(2)(3)}), V(t_3), V(t_4))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.12 *Let E be from Example 3.23. In Figure 27, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the vector $V = (V(t_2), V(t_3))$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

Example 4.13 *Let E be from Example 3.24. In Figure 28, the marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ and its reachability graph $RG(N)$ are presented. For each state $Q = (M, V) \in RS(N)$, the timer valuation function is described by the one-element vector (scalar) $V = V(t_2)$, placed under the corresponding marking M. Note that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

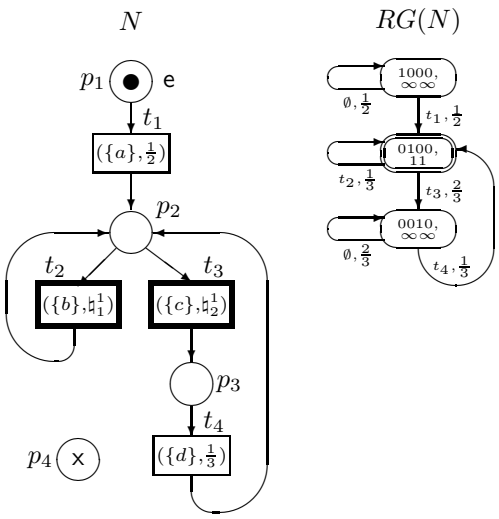

Figure 27: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = [(\{a\}, \frac{1}{2}) * ((\{b\}, \lightning_1^1)[](\{c\}, \lightning_2^1); (\{d\}, \frac{1}{3}))) * \mathsf{Stop}]$ and its reachability graph

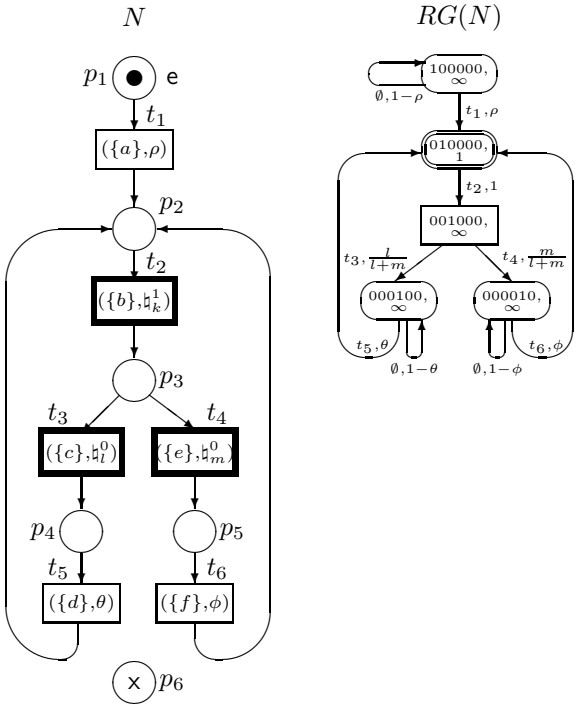

Figure 28: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \lightning_k^1); ((((\{c\}, \lightning_l^0); (\{d\}, \theta))[] (((\{e\}, \lightning_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$ and its reachability graph

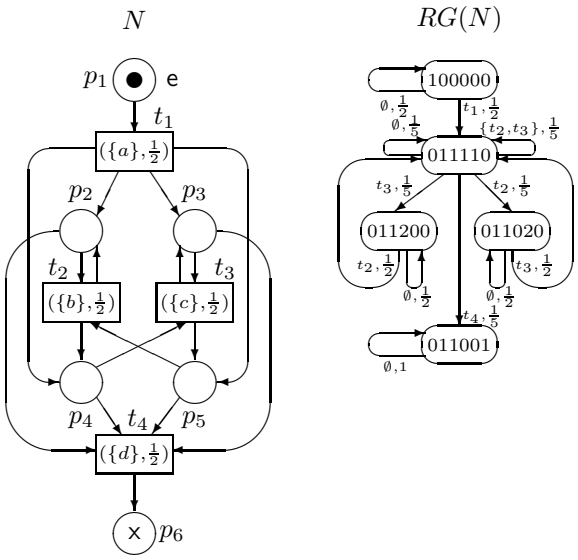

Figure 29: The marked and clocked dtsd-box $N = Box_{dtsd}(\overline{E})$ for $E = [((\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}) \| (\{c\}, \frac{1}{2})) * (\{d\}, \frac{1}{2})]$ and its reachability graph

In Examples 4.1–4.13, the marked and clocked dtsd-boxes $N = Box_{dtsd}(\overline{E})$ are presented for E from Examples 3.12–3.24. Note that, due to the time constraints and since waiting multiactions may be preempted by stochastic ones, some dynamic expressions can have complex transition systems (reachability graphs) and simple marked and clocked dtsd-boxes (Examples 4.1–4.7), or vice versa (Examples 4.8–4.13).

The following example demonstrates that without the syntactic restriction on regularity of expressions the corresponding marked and clocked dtsd-boxes may be not safe.

Example 4.14 *Let* $E = [((\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}) \| (\{c\}, \frac{1}{2})) * (\{d\}, \frac{1}{2})]$. *In Figure 29, the marked and clocked dtsd-box* $N = Box_{dtsd}(\overline{E})$ *and its reachability graph* $RG(N)$ *are presented. Since* N *has no waiting transitions, we may consider the substituent markings* M *as the whole states* $Q = (M, \varepsilon) \in RS(N)$, *where* ε *is the zero-element vector (empty sequence). At the marking* $(0, 1, 1, 2, 0, 0)$ *there are* 2 *tokens in the place* p_4. *Symmetrically, at the marking* $(0, 1, 1, 0, 2, 0)$ *there are* 2 *tokens in the place* p_5. *Thus, allowing concurrency in the second argument of iteration in the expression* $\overline{E}$ *can lead to non-safeness of the corresponding marked and clocked dtsd-box* N, *though, it is* 2*-bounded in the worst case [40]. The origin of the problem is that* N *has as a self-loop with two subnets which can function independently. Therefore, we have decided to consider regular expressions only, since the alternative, which is a safe version of the iteration operator with six arguments in the corresponding dtsd-box, like that from [40], is rather cumbersome and has too intricate PN interpretation. Our motivation was to keep the algebraic and PN specifications as simple as possible.*

5 Performance evaluation

In this section we demonstrate how Markov chains corresponding to the expressions and dtsd-boxes can be constructed and then used for performance evaluation.

5.1 Analysis of the underlying SMC (embedding)

For a dynamic expression G, a discrete random variable $\xi(s)$ is associated with every tangible state $s \in DR_T(G)$. The variable captures the residence (sojourn) time in the state. One can interpret staying in a state at the next discrete time moment as a failure and leaving it as a success in some trial series. It is easy to see that $\xi(s)$ is geometrically distributed with the parameter $1 - PM(s, s)$, since the probability to stay in s for $k - 1$ time moments and leave it at the moment $k \geq 1$, called the probability mass function (PMF) of the residence time in s, is $p_{\xi(s)}(k) = \mathsf{P}(\xi(s) = k) = PM(s, s)^{k-1}(1 - PM(s, s))$ $(k \in \mathbb{N}_{\geq 1})$ (the residence time in s is k in this case). Hence, the probability distribution function (PDF) of the residence time in s is $F_{\xi(s)}(k) = \mathsf{P}(\xi(s) < k) = 1 - PM(s, s)^{k-1}$ $(k \in \mathbb{N}_{\geq 1})$ (the probability that the residence time in s is less than k).

Note that the deterministic residence time 1 in a tangible state s can be interpreted as a random variable $\xi(s)$ that is geometrically distributed with the parameter $1 = 1 - PM(s,s)$. In that case, $PM(s,s) = 0$ and $k = 1$ is the only residence time value with a positive probability. Hence, $p_{\xi(s)}(1) = PM(s,s)^{1-1}(1 - PM(s,s)) = 0^0 \cdot 1 = 1$, i.e. the probability that the residence time is 1 equals 1.

Further, the residence time ∞ in an absorbing tangible state s can be interpreted as a random variable $\xi(s)$ that is geometrically distributed with the parameter $0 = 1 - PM(s,s)$. In that case, $PM(s,s) = 1$ and there exists no finite residence time value with a positive probability. Hence, $p_{\xi(s)}(k) = PM(s,s)^{k-1}(1-PM(s,s)) = 1^{k-1} \cdot 0 = 0$ ($k \in I\!N_{\geq 1}$), i.e. the probability that the residence time is k equals 0 for every $k \geq 1$. Then we cannot leave s for a different state after any number of time ticks and we stay in s for infinite time.

The mean value formula for the geometrical distribution allows us to calculate the average sojourn time in $s \in DR_T(G)$ as $SJ(s) = \frac{1}{1-PM(s,s)}$. The average sojourn time in each vanishing state $s \in DR_V(G)$ is $SJ(s) = 0$. Let $s \in DR(G)$.

The *average sojourn time in the state s* is

$$SJ(s) = \begin{cases} \frac{1}{1-PM(s,s)}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

The *average sojourn time vector* of G, denoted by SJ, has the elements $SJ(s),\ s \in DR(G)$.

The *sojourn time variance in the state s* is

$$VAR(s) = \begin{cases} \frac{PM(s,s)}{(1-PM(s,s))^2}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

The *sojourn time variance vector* of G, denoted by VAR, has the elements $VAR(s),\ s \in DR(G)$.

To evaluate performance of the system specified by a dynamic expression G, we should investigate the stochastic process associated with it. The process is the underlying semi-Markov chain (SMC) [260, 265, 186, 49, 285, 188, 261, 263], denoted by $SMC(G)$, which can be analyzed by extracting from it the embedded (absorbing) discrete time Markov chain (EDTMC) corresponding to G, denoted by $EDTMC(G)$. The construction of the latter is analogous to that applied in the context of generalized stochastic PNs (GSPNs) in [213, 214, 15, 16], and also in the framework of discrete time deterministic and stochastic PNs (DTDSPNs) in [309, 305, 306, 311, 312, 310], as well as within discrete deterministic and stochastic PNs (DDSPNs) [307, 308]. $EDTMC(G)$ only describes the state changes of $SMC(G)$ while ignoring its time characteristics. Thus, to construct the EDTMC, we should abstract from all time aspects of behaviour of the SMC, i.e. from the sojourn time in its states. The (local) sojourn time in every state of the EDTMC is deterministic and it is equal to one discrete time unit. It is well-known that every SMC is fully described by the EDTMC and the state sojourn time distributions (the latter can be specified by the vector of PDFs of residence time in the states) [155, 265, 285, 188].

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$. The transition system $TS(G)$ can have self-loops going from a state to itself which have a non-zero probability. Clearly, the current state remains unchanged in this case.

Let $s \to s$. The *probability to stay in s due to k* ($k \geq 1$) *self-loops* is

$$PM(s,s)^k.$$

The *self-loops abstraction factor in the state s* is

$$SL(s) = \begin{cases} \frac{1}{1-PM(s,s)}, & s \to s; \\ 1, & \text{otherwise.} \end{cases}$$

The *self-loops abstraction vector* of G, denoted by SL, has the elements $SL(s),\ s \in DR(G)$.

Let $s \to \tilde{s}$ and $s \neq \tilde{s}$, i.e. $PM(s,s) < 1$. The *probability to move from s to $\tilde{s}$ by executing any multiset of activities after possible self-loops* is

$$PM^*(s,\tilde{s}) = \left\{ \begin{array}{ll} PM(s,\tilde{s}) \sum_{k=0}^{\infty} PM(s,s)^k = \frac{PM(s,\tilde{s})}{1-PM(s,s)}, & s \to s; \\ PM(s,\tilde{s}), & \text{otherwise;} \end{array} \right\} = SL(s)PM(s,\tilde{s}).$$

The value $k = 0$ in the summation above corresponds to the case when no self-loops occur.

Let $s \in DR_T(G)$. If there exist self-loops from s (i.e. if $s \to s$) then $PM(s,s) > 0$ and $SL(s) = \frac{1}{1-PM(s,s)} = SJ(s)$. Otherwise, if there exist no self-loops from s then $PM(s,s) = 0$ and $SL(s) = 1 = \frac{1}{1-PM(s,s)} = SJ(s)$. Thus, $\forall s \in DR_T(G)\ SL(s) = SJ(s)$, hence, $\forall s \in DR_T(G)$ with $PM(s,s) < 1$ it holds $PM^*(s,\tilde{s}) = SJ(s)PM(s,\tilde{s})$. Note that the self-loops from tangible states are of the empty or non-empty type, the latter produced by iteration, since empty loops are not possible from w-tangible states, but they are possible from s-tangible states, while non-empty loops are possible from both s-tangible and w-tangible states.

Let $s \in DR_V(G)$. We have $\forall s \in DR_V(G)\ SL(s) \neq SJ(s) = 0$ and $\forall s \in DR_V(G)$ with $PM(s,s) < 1$ it holds $PM^*(s,\tilde{s}) = SL(s)PM(s,\tilde{s})$. If there exist self-loops from s then $PM^*(s,\tilde{s}) = \frac{PM(s,\tilde{s})}{1-PM(s,s)}$ when $PM(s,s) < 1$. Otherwise, if there exist no self-loops from s then $PM^*(s,\tilde{s}) = PM(s,\tilde{s})$. Note that the self-loops from vanishing states are always of the non-empty type, produced by iteration, since empty loops are not possible from vanishing states. Further, we suppose that all (if any) loops among vanishing states are "transient" rather than "absorbing", as in [214, 16]. Then for each s with $PM(s,s) = 1$ (absorbing state) we have $s \in DR_T(G)$, since there exist no absorbing vanishing states, hence, $\forall s \in DR_V(G)\ PM(s,s) < 1$.

Note that after abstraction from the probabilities of transitions which do not change the states, the remaining transition probabilities are normalized. In order to calculate transition probabilities $PT(\Upsilon, s)$, we had to normalize $PF(\Upsilon, s)$. Then, to obtain transition probabilities of the state-changing steps $PM^*(s,\tilde{s})$, we now have to normalize $PM(s,\tilde{s})$. Thus, we have a two-stage normalization as a result.

Notice that $PM^*(s,\tilde{s})$ defines a probability distribution, since $\forall s \in DR(G)$ such that s is not an absorbing state (i.e. $PM(s,s) < 1$, hence, there are transitions to different states after possible self-loops from it) we have $\sum_{\{\tilde{s}|s\to\tilde{s},\ s\neq\tilde{s}\}} PM^*(s,\tilde{s}) = \frac{1}{1-PM(s,s)} \sum_{\{\tilde{s}|s\to\tilde{s},\ s\neq\tilde{s}\}} PM(s,\tilde{s}) = \frac{1}{1-PM(s,s)}(1 - PM(s,s)) = 1$.

We decided to consider self-loops followed only by a state-changing step just for convenience. Alternatively, we could take a state-changing step followed by self-loops or a state-changing step preceded and followed by self-loops. In all these three cases our sequence begins or/and ends with the loops which do not change states. At the same time, the overall probabilities of the evolutions can differ, since self-loops have positive probabilities. To avoid inconsistency of definitions and too complex description, we consider sequences ending with a state-changing step. It resembles in some sense a construction of branching bisimulation [137] taking self-loops instead of silent transitions. Further, we shall not abstract from self-loops with probability 1 while constructing EDTMCs, in order to maintain a probability distribution among transitions (actually, a single transition to the same state) from every state with such a self-loop.

Definition 5.1 *Let G be a dynamic expression. The* embedded (absorbing) discrete time Markov chain (EDTMC) *of G, denoted by $EDTMC(G)$, has the state space $DR(G)$, the initial state $[G]_{\approx}$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$, if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s,\tilde{s})$; or $s \twoheadrightarrow_1 s$, if $PM(s,s) = 1$.*

The underlying SMC *of G, denoted by $SMC(G)$, has the EDTMC $EDTMC(G)$ and the sojourn time in every $s \in DR_T(G)$ is geometrically distributed with the parameter $1 - PM(s,s)$ (in particular, the sojourn time is 1 when $PM(s,s) = 0$, and ∞ when $PM(s,s) = 1$) while the sojourn time in every $s \in DR_V(G)$ is equal to 0.*

EDTMCs and underlying SMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $EDTMC(E) = EDTMC(\overline{E})$ and $SMC(E) = SMC(\overline{E})$.

Let G be a dynamic expression. The elements $\mathcal{P}^*_{ij}$ $(1 \leq i,j \leq n = |DR(G)|)$ of the (one-step) transition probability matrix (TPM) $\mathbf{P}^*$ for $EDTMC(G)$ are defined as

$$\mathcal{P}^*_{ij} = \begin{cases} PM^*(s_i,s_j), & s_i \to s_j,\ i \neq j; \\ 1, & PM(s_i,s_i) = 1,\ i = j; \\ 0, & \text{otherwise.} \end{cases}$$

The transient (k-step, $k \in I\!N$) PMF $\psi^*[k] = (\psi^*[k](s_1), \ldots, \psi^*[k](s_n))$ for $EDTMC(G)$ is calculated as

$$\psi^*[k] = \psi^*[0](\mathbf{P}^*)^k,$$

where $\psi^*[0] = (\psi^*[0](s_1), \ldots, \psi^*[0](s_n))$ is the initial PMF defined as

$$\psi^*[0](s_i) = \begin{cases} 1, & s_i = [G]_{\approx}; \\ 0, & \text{otherwise.} \end{cases}$$

Note also that $\psi^*[k+1] = \psi^*[k]\mathbf{P}^*$ $(k \in I\!N)$.

The steady-state PMF $\psi^* = (\psi^*(s_1), \ldots, \psi^*(s_n))$ for $EDTMC(G)$ is a solution of the equation system

$$\begin{cases} \psi^*(\mathbf{P}^* - \mathbf{I}) = \mathbf{0} \\ \psi^*\mathbf{1}^T = 1 \end{cases},$$

where $\mathbf{I}$ is the identity matrix of order n and $\mathbf{0}$ is a row vector of n values 0, $\mathbf{1}$ is that of n values 1.

Note that the vector ψ^* exists and is unique if $EDTMC(G)$ is ergodic. Then $EDTMC(G)$ has a single steady state, and we have $\psi^* = \lim_{k\to\infty} \psi^*[k]$. We shall consider only Markov chains with at most one steady state.

The steady-state PMF for the underlying semi-Markov chain $SMC(G)$ is calculated via multiplication of every $\psi^*(s_i)$ $(1 \leq i \leq n)$ by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Remember that for each tangible state $s \in DR_T(G)$ we have $SJ(s) \geq 1$, and for each vanishing state $s \in DR_V(G)$ we have $SJ(s) = 0$.

Thus, the steady-state PMF $\varphi = (\varphi(s_1), \ldots, \varphi(s_n))$ for $SMC(G)$ is

$$\varphi(s_i) = \begin{cases} \frac{\psi^*(s_i)SJ(s_i)}{\sum_{j=1}^{n} \psi^*(s_j)SJ(s_j)}, & s_i \in DR_T(G); \\ 0, & s_i \in DR_V(G). \end{cases}$$

Thus, to calculate φ, we apply abstraction from self-loops with probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. We call that technique *embedding*, since the embedded DTMC (EDTMC) is used to specify the SMC state change probabilities. $EDTMC(G)$ has no self-loops with probability less than 1, unlike $SMC(G)$, hence, the behaviour of $EDTMC(G)$ may stabilize quicker than that of $SMC(G)$ (if each of them has a single steady state), since $\mathbf{P}^*$ has only zero (excepting the states having self-loops with probability 1) elements at the main diagonal.

Example 5.1 *Let E be from Example 3.24. In Figure 30, the underlying SMC $SMC(\overline{E})$ is presented. The average sojourn times in the states of the underlying SMC are written next to them in bold font.*

The average sojourn time vector of $\overline{E}$ is

$$SJ = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}, \frac{1}{\phi}\right).$$

The sojourn time variance vector of $\overline{E}$ is

$$VAR = \left(\frac{1-\rho}{\rho^2}, 0, 0, \frac{1-\theta}{\theta^2}, \frac{1-\phi}{\phi^2}\right).$$

The TPM for $EDTMC(\overline{E})$ is

$$\mathbf{P}^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The steady-state PMF for $EDTMC(\overline{E})$ is

$$\psi^* = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{l}{3(l+m)}, \frac{m}{3(l+m)}\right).$$

The steady-state PMF ψ^* weighted by SJ is

$$\left(0, \frac{1}{3}, 0, \frac{l}{3\theta(l+m)}, \frac{m}{3\phi(l+m)}\right).$$

It remains to normalize the steady-state weighted PMF by dividing it by the sum of its components

$$\psi^* SJ^T = \frac{\theta\phi(l+m) + \phi l + \theta m}{3\theta\phi(l+m)}.$$

Thus, the steady-state PMF for $SMC(\overline{E})$ is

$$\varphi = \frac{1}{\theta\phi(l+m) + \phi l + \theta m}(0, \theta\phi(l+m), 0, \phi l, \theta m).$$

In the case $l = m$ and $\theta = \phi$ we have

$$\varphi = \frac{1}{2(1+\theta)}(0, 2\theta, 0, 1, 1).$$

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$, $S, \widetilde{S} \subseteq DR(G)$. The following standard *performance indices (measures)* can be calculated based on the steady-state PMF φ for $SMC(G)$ and the average sojourn time vector SJ of G [230, 179].

- The *average recurrence (return) time in the state s* (i.e. the number of discrete time units or steps required for this) is $ReturnTime(s) = \frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $TimeFract(s) = \varphi(s)$.

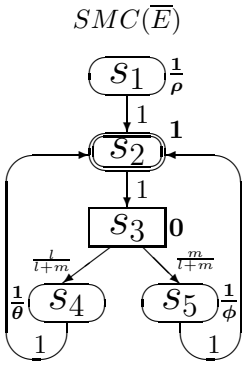

Figure 30: The underlying SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

- The *fraction of residence time in the set of states* S or the *probability of the event determined by a condition that is true for all states from* S is $TimeFract(S) = \sum_{s \in S} \varphi(s)$.
- The *relative fraction of residence time in the set of states* S *with respect to that in* $\widetilde{S}$ is $RltTimeFract(S, \widetilde{S}) = \frac{\sum_{s \in S} \varphi(s)}{\sum_{\tilde{s} \in \widetilde{S}} \varphi(\tilde{s})}$.
- The *exit/entrance frequency (rate of leaving/entering, average number of exits/entrances per unit of time) the state* s is $ExitFreq(s) = \frac{\varphi(s)}{SJ(s)}$.
- The *steady-state probability to perform a step with a multiset of activities* Ξ is $ActsProb(\Xi) = \sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon | \Xi \subseteq \Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function* r *on the states* is $Prob(r) = \sum_{s \in DR(G)} \varphi(s) r(s)$, where $\forall s \in DR(G)\ 0 \leq r(s) \leq 1$.

Example 5.2 *Let us interpret* E *from Example 3.24 as a specification of the travel system. A tourist visits regularly new cities. After seeing the sights of the current city, he goes to the next city by the nearest train or bus available at the city station. Buses depart less frequently than trains, but the next city is quicker reached by bus than by train. We suppose that the stay duration in every city (being a constant), the departure numbers of trains and buses, as well as their speeds* do not depend *on a particular city, bus or train. The travel route has been planned so that the distances between successive cities* coincide.

The meaning of actions and activities from the syntax of E *is as follows. The action* a *corresponds to the system activation after planning the travel route that takes a time, geometrically distributed with a parameter* ρ*, the probability of the corresponding stochastic multiaction* $(\{a\}, \rho)$*. The action* b *represents coming to the city station after completion of looking round the current city that takes (for every city) a fixed time equal to* 1 *(say, one hour), the time delay of the corresponding waiting multiaction* $(\{b\}, \natural_k^1)$ *with (resolving no choice) weight* k*. The actions* c *and* e *correspond to the urgent (in zero time) getting on bus and train, respectively, and thus model the choice between these two transport facilities. The weights of the two corresponding immediate multiactions* $(\{c\}, \natural_l^0)$ *and* $(\{e\}, \natural_m^0)$ *suggest that every* l *departures of buses take the same time as* m *departures of trains* $(l < m)$*, hence, a bus departs with the probability* $\frac{l}{l+m}$ *while a train departs with the probability* $\frac{m}{l+m}$*. The actions* d *and* f *correspond to coming in a city by bus and train, respectively, that takes a time, geometrically distributed with the parameters* θ *and* ϕ*, respectively* $(\theta > \phi)$*, the probabilities of the corresponding stochastic multiactions* $(\{d\}, \theta)$ *and* $(\{f\}, \phi)$*.*

The meaning of states from $DR(\overline{E})$ *is the following. The s-tangible state* s_1 *corresponds to staying at home and planning the future travel. The w-tangible state* s_2 *means residence in a city for exactly one time unit (hour). The vanishing state* s_3 *with zero residence time represents instantaneous stay at the city station, signifying that the tourist does not wait there for departure of the transport. The s-tangible states* s_4 *and* s_5 *correspond to going by bus and train, respectively.*

Using Example 5.1, we now calculate the performance indices, based on the steady-state PMF for $SMC(\overline{E})$ $\varphi = \frac{1}{\theta\phi(l+m)+\phi l+\theta m}(0, \theta\phi(l+m), 0, \phi l, \theta m)$ *and the average sojourn time vector of* $\overline{E}$ $SJ = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}, \frac{1}{\phi}\right)$.

- *The average time between comings to the successive cities (mean sightseeing and travel time) is* $ReturnTime(s_2) = \frac{1}{\varphi(s_2)} = 1 + \frac{\phi l + \theta m}{\theta\phi(l+m)}$.

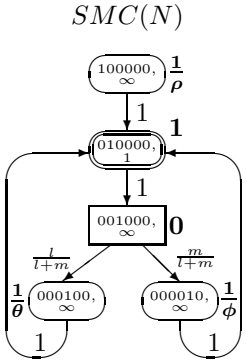

Figure 31: The underlying SMC of $N = Box_{dtsd}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

- *The fraction of time spent in a city (sightseeing time fraction) is* $TimeFract(s_2) = \varphi(s_2) = \frac{\theta\phi(l+m)}{\theta\phi(l+m)+\phi l+\theta m}$.
- *The fraction of time spent in a transport (travel time fraction) is* $TimeFract(\{s_4, s_5\}) = \varphi(s_4) + \varphi(s_5) = \frac{\phi l+\theta m}{\theta\phi(l+m)+\phi l+\theta m}$.
- *The relative fraction of time spent in a city with respect to that spent in transport (sightseeing relative to travel time fraction) is* $RltTimeFract(\{s_2\}, \{s_4, s_5\}) = \frac{\varphi(s_2)}{\varphi(s_4)+\varphi(s_5)} = \frac{\theta\phi(l+m)}{\phi l+\theta m}$.
- *The rate of leaving/entering a city (departure/arrival rate) is* $ExitFreq(s_2) = \frac{\varphi(s_2)}{SJ(s_2)} = \frac{\theta\phi(l+m)}{\theta\phi(l+m)+\phi l+\theta m}$.

Let $N = (P_N, T_N, W_N, D_N, \Omega_N, \mathcal{L}_N, Q_N)$ be a LDTSDPN and $Q, \widetilde{Q}$ be its states. Then the average sojourn time $SJ(Q)$, sojourn time variance $VAR(Q)$, probabilities $PM^*(Q, \widetilde{Q})$, transition relation $Q \twoheadrightarrow_{\mathcal{P}} \widetilde{Q}$, EDTMC $EDTMC(N)$, underlying SMC $SMC(N)$ and steady-state PMF for it are defined like the corresponding notions for dynamic expressions.

As we have mentioned earlier, every marked and clocked plain dtsd-box can be interpreted as an LDTSDPN. Therefore, we can evaluate performance with the LDTSDPNs corresponding to dtsd-boxes and then transfer the results to the latter.

Example 5.3 *Let E be from Example 3.24 and $N = Box_{dtsd}(\overline{E})$. In Figure 31, the underlying SMC $SMC(N)$ is presented. Note that $SMC(\overline{E})$ and $SMC(N)$ are isomorphic. Thus, both the transient and steady-state PMFs for $SMC(N)$ and $SMC(\overline{E})$ coincide.*

As mentioned in [309, 305, 306], if is useful to consider performance measures over only the markings of DTDSPNs, instead of their whole states, whose second components are the remaining firing time vectors. In the context of dtsdPBC, such markings correspond to those of the dtsd-boxes of dynamic expressions, i.e. to the markings of the respective LDTSDPNs, obtained from their states by abstracting from the second components, which are the timer valuation functions.

Let G be a dynamic expression. The *underlying timer-free state* of a state $s \in DR(G)$ is defined as $\lfloor s = [\lfloor H]_{\approx}$ for $H \in s$. Note that only in the first inaction rule in Table 1 the left and right parts have different timer value annotations. After removing all timer values, the both parts will coincide. Since structurally equivalent dynamic expressions remain so after removing their timer value annotations, $\lfloor s$ is unique for each s and the definition is correct. Thus, $\lfloor s$ is the structural equivalence class of an arbitrary dynamic expression from s, where timer values have been removed, prior to taking the equivalence class of that expression. We cannot simply collect all the timer-discarded dynamic expressions from s, since $\lfloor s$ should be a *state* itself, i.e. it must contain *all* structurally equivalent expressions.

Example 5.4 *Let E be from Example 3.1 and $s = [\overline{E}]_{\approx}$, $\tilde{s} = \circlearrowleft s$. Then* $\lfloor s = \lfloor \tilde{s} = [\overline{(\{a\}, \natural_1^3)}[](\{b\}, \frac{1}{3})]_{\approx} = [(\{a\}, \natural_1^3)[]\overline{(\{b\}, \frac{1}{3})}]_{\approx} = [\overline{(\{a\}, \natural_1^3)}[]\overline{(\{b\}, \frac{1}{3})}]_{\approx} = \{\overline{(\{a\}, \natural_1^3)}[](\{b\}, \frac{1}{3}), (\{a\}, \natural_1^3)[]\overline{(\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)[](\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)^3}[](\{b\}, \frac{1}{3}), (\{a\}, \natural_1^3)^3[]\overline{(\{b\}, \frac{1}{3})}, \overline{(\{a\}, \natural_1^3)^3[](\{b\}, \frac{1}{3})}\}$.

Hence, some *enabled* waiting multiactions of s may have the *initial* timer value superscripts. Actually, this does not provide $\lfloor s$ with an extra timing information, since those superscripts are determined only by the delays of the corresponding waiting multiactions and their enabling status. The elements of the *set of all timer-free states of* G, defined as $\lfloor DR(G) = \{\lfloor s \mid s \in DR(G)\}$), correspond to the reachable markings of the LDTSDPN $N = Box_{dtsd}(G)$.

Let $s \in DR(G)$ and $\bar{s} = \lfloor s$. The steady-state PMF for $SMC(G)$ over the timer-free states of G is defined as follows: $\varphi(\bar{s}) = \sum_{\{s \in DR(G) | \lfloor s = \bar{s}\}} \varphi(s)$. Then $\varphi(\bar{s})$ can be used to calculate the standard *performance indices over the timer-free states* of G (hence, over the markings of N), by analogy with the standard performance indices, defined over the arbitrary states of G. Then also the performance measures that are specific for LDTSDPNs can be derived, based on the numbers of tokens in the places of N.

5.2 Analysis of the DTMC (abstraction)

Let us consider an alternative solution method, studying the DTMCs of expressions based on the state change probabilities $PM(s, \tilde{s})$.

Definition 5.2 *Let G be a dynamic expression. The* discrete time Markov chain (DTMC) *of G, denoted by $DTMC(G)$, has the state space $DR(G)$, the initial state $[G]_{\approx}$ and the transitions $s \to_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM(s, \tilde{s})$.*

DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $DTMC(E) = DTMC(\overline{E})$.

One can see that $EDTMC(G)$ is constructed from $DTMC(G)$ as follows. For each state of $DTMC(G)$, we remove a possible self-loop with probability less than 1, associated with it and then normalize the probabilities of the remaining transitions from the state. Thus, $EDTMC(G)$ and $DTMC(G)$ differ only by existence of self-loops with probability less than 1 and magnitudes of the probabilities of the remaining transitions. Hence, $EDTMC(G)$ and $DTMC(G)$ have the same communication classes of states and $EDTMC(G)$ is irreducible iff $DTMC(G)$ is so. Since both $EDTMC(G)$ and $DTMC(G)$ are finite, they are positive recurrent. Thus, in case of irreducibility, each of them has a single stationary PMF. Note that both $EDTMC(G)$ and $DTMC(G)$ or just one of them may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that $EDTMC(G)$ is periodic while $DTMC(G)$ is aperiodic due to self-loops associated with some states of the latter. The states of $SMC(G)$ are classified using $EDTMC(G)$, hence, $SMC(G)$ is irreducible (positive recurrent, aperiodic) iff $EDTMC(G)$ is so.

Let G be a dynamic expression. The elements $\mathcal{P}_{ij}$ $(1 \le i, j \le n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ are defined as

$$\mathcal{P}_{ij} = \begin{cases} PM(s_i, s_j), & s_i \to s_j; \\ 0, & \text{otherwise.} \end{cases}$$

The steady-state PMF ψ for $DTMC(G)$ is defined like the corresponding notion ψ^* for $EDTMC(G)$.

Let us determine a relationship between steady-state PMFs for $DTMC(G)$ and $EDTMC(G)$. The following theorem proposes the equation that relates the mentioned steady-state PMFs.

First, we introduce some helpful notation. For a vector $v = (v_1, \ldots, v_n)$, let $Diag(v)$ be a diagonal matrix of order n with the elements $Diag_{ij}(v)$ $(1 \le i, j \le n)$ defined as

$$Diag_{ij}(v) = \begin{cases} v_i, & i = j; \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 5.1 *Let G be a dynamic expression and SL be its self-loops abstraction vector. Then the steady-state PMFs ψ for $DTMC(G)$ and ψ^* for $EDTMC(G)$ are related as follows: $\forall s \in DR(G)$*

$$\psi(s) = \frac{\psi^*(s) SL(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s}) SL(\tilde{s})}.$$

Proof. Let there is an absorbing state $s_i \in DR(G)$ $(1 \le i \le n)$, i.e. $PM(s_i, s_i) = 1$. Then $\mathcal{P}_{ii} = 1$ in the TPM $\mathbf{P}$ of $DTMC(G)$ and $\mathcal{P}^*_{ii} = 1$ in the TPM $\mathbf{P}^*$ of $EDTMC(G)$, by definitions of those TPMs. We have earlier supposed that there exist no absorbing vanishing states, hence, $s_i \in DR_T(G)$ and $SL(s_i) = \infty = SJ(s_i)$. We have also supposed at most one single steady state in the considered Markov chains, hence, $\{s_i\}$ is a single communication (and ergodic) class of states in both $DTMC(G)$ and $EDTMC(G)$. Then $\psi(s_i) = 1 = \psi^*(s_i)$, whereas $\forall s \in DR(G) \setminus \{s_i\}$ $SL(s) < \infty$, $SJ(s) < \infty$ and $\psi(s) = 0 = \psi^*(s)$. We thus get for s_i

$$\frac{\psi^*(s_i) SL(s_i)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s}) SL(\tilde{s})} = \frac{1 \cdot SL(s_i)}{1 \cdot SL(s_i)} = \frac{1}{1} = 1 = \psi(s_i),$$

whereas $\forall s \in DR(G) \setminus \{s_i\}$

$$\frac{\psi^*(s)SL(s)}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{0 \cdot SL(s)}{1 \cdot SL(s_i)} = \frac{0}{\infty} = 0 = \psi(s).$$

Let there are no absorbing states, i.e. $\forall s \in DR(G)\ PM(s,s) < 1$. Let PSL be a vector with the elements

$$PSL(s) = \left\{ \begin{array}{ll} PM(s,s), & s \to s; \\ 0, & \text{otherwise} \end{array} \right\} = 1 - \frac{1}{SL(s)}.$$

By definition of $PM^*(s,\tilde{s})$, we have $\mathbf{P}^* = Diag(SL)(\mathbf{P} - Diag(PSL)) = Diag(SL)(\mathbf{P} - \mathbf{I} + Diag(SL)^{-1}) = Diag(SL)(\mathbf{P} - \mathbf{I}) + \mathbf{I}$. Hence,

$$\mathbf{P}^* - \mathbf{I} = Diag(SL)(\mathbf{P} - \mathbf{I}).$$

Then $\psi^*(\mathbf{P}^* - \mathbf{I}) = \mathbf{0}$ implies

$$\psi^* Diag(SL)(\mathbf{P} - \mathbf{I}) = \mathbf{0}.$$

For $v = \psi^* Diag(SL)$, we get

$$v(\mathbf{P} - \mathbf{I}) = \mathbf{0}.$$

In order to calculate ψ on the basis of v, we must normalize it by dividing its elements by their sum, since we should have $\psi \mathbf{1}^T = 1$ as a result:

$$\psi = \frac{1}{v\mathbf{1}^T} v = \frac{1}{\psi^* Diag(SL)\mathbf{1}^T} \psi^* Diag(SL).$$

Thus, the elements of ψ are calculated as follows: $\forall s \in DR(G)$

$$\psi(s) = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}.$$

It is easy to check that ψ is a solution of the equation system

$$\left\{ \begin{array}{l} \psi(\mathbf{P} - \mathbf{I}) = \mathbf{0} \\ \psi \mathbf{1}^T = 1 \end{array} \right. ,$$

hence, it is indeed the steady-state PMF for $DTMC(G)$. □

The following proposition relates the steady-state PMFs for $SMC(G)$ and $DTMC(G)$.

Proposition 5.1 *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$. Then $\forall s \in DR(G)$*

$$\varphi(s) = \left\{ \begin{array}{ll} \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{array} \right.$$

Proof. Let $s \in DR_T(G)$. Remember that $\forall s \in DR_T(G)\ SL(s) = SJ(s)$ and $\forall s \in DR_V(G)\ SJ(s) = 0$.

Then, by Theorem 5.1, we have $\frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})} = \frac{\frac{\psi^*(s)SL(s)}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}}{\sum_{\tilde{s}\in DR_T(G)} \left(\frac{\psi^*(\tilde{s})SL(\tilde{s})}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})} \right)} = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}$.

$\frac{\sum_{\check{s}\in DR(G)} \psi^*(\check{s})SL(\check{s})}{\sum_{\tilde{s}\in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{\psi^*(s)SJ(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi^*(\tilde{s})SJ(\tilde{s})} = \frac{\psi^*(s)SJ(s)}{\sum_{\tilde{s}\in DR(G)} \psi^*(\tilde{s})SJ(\tilde{s})} = \varphi(s).$ □

Thus, to calculate φ, one can only apply normalization to some elements of ψ (corresponding to the tangible states), instead of abstracting from self-loops with probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. We call that technique *abstraction*, since we abstract from the vanishing states and consider only the (normalized) DTMC-based stationary probabilities of the tangible states. Hence, using $DTMC(G)$ instead of $EDTMC(G)$ allows one to avoid multistage analysis, but the payment for it is more time-consuming numerical and more complex analytical calculation of ψ with respect to ψ^*. The reason is that $DTMC(G)$ may have self-loops with probability less than 1, unlike $EDTMC(G)$, hence, the behaviour of $DTMC(G)$ may stabilize slower than that of $EDTMC(G)$ (if each of them has a single steady state) and $\mathbf{P}$ is potentially more dense matrix than $\mathbf{P}^*$, since $\mathbf{P}$ may have additional non-zero elements at the main diagonal. Nevertheless, Proposition 5.1 is very important, since the relationship between φ and ψ it discovers will be used in Proposition 5.2 to relate the steady-state PMFs for $SMC(G)$ and the reduced $DTMC(G)$.

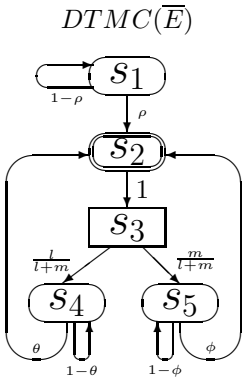

Figure 32: The DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

Example 5.5 *Let E be from Example 3.24. In Figure 32, the DTMC $DTMC(\overline{E})$ is presented. The TPM for $DTMC(\overline{E})$ is*

$$\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \phi & 0 & 0 & 1-\phi \end{pmatrix}.$$

The steady-state PMF for $DTMC(\overline{E})$ is

$$\psi = \frac{1}{2\theta\phi(l+m) + \phi l + \theta m}(0, \theta\phi(l+m), \theta\phi(l+m), \phi l, \theta m).$$

Remember that $DR_T(\overline{E}) = DR_{ST}(\overline{E}) \cup DR_{WT}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{E}) = \{s_3\}$. Hence,

$$\sum_{s \in DR_T(\overline{E})} \psi(s) = \psi(s_1) + \psi(s_2) + \psi(s_4) + \psi(s_5) = \frac{\theta\phi(l+m) + \phi l + \theta m}{2\theta\phi(l+m) + \phi l + \theta m}.$$

By Proposition 5.1, we have

$$\begin{aligned}
&\varphi(s_1) = 0 \cdot \tfrac{2\theta\phi(l+m)+\phi l+\theta m}{\theta\phi(l+m)+\phi l+\theta m} = 0, \\
&\varphi(s_2) = \tfrac{\theta\phi(l+m)}{2\theta\phi(l+m)+\phi l+\theta m} \cdot \tfrac{2\theta\phi(l+m)+\phi l+\theta m}{\theta\phi(l+m)+\phi l+\theta m} = \tfrac{\theta\phi(l+m)}{\theta\phi(l+m)+\phi l+\theta m}, \\
&\varphi(s_3) = 0, \\
&\varphi(s_4) = \tfrac{\phi l}{2\theta\phi(l+m)+\phi l+\theta m} \cdot \tfrac{2\theta\phi(l+m)+\phi l+\theta m}{\theta\phi(l+m)+\phi l+\theta m} = \tfrac{\phi l}{\theta\phi(l+m)+\phi l+\theta m}, \\
&\varphi(s_5) = \tfrac{\theta m}{2\theta\phi(l+m)+\phi l+\theta m} \cdot \tfrac{2\theta\phi(l+m)+\phi l+\theta m}{\theta\phi(l+m)+\phi l+\theta m} = \tfrac{\theta m}{\theta\phi(l+m)+\phi l+\theta m}.
\end{aligned}$$

Thus, the steady-state PMF for $SMC(\overline{E})$ is

$$\varphi = \frac{1}{\theta\phi(l+m) + \phi l + \theta m}(0, \theta\phi(l+m), 0, \phi l, \theta m).$$

This coincides with the result obtained in Example 5.1 with the use of ψ^* and SJ.

5.3 Analysis of the reduced DTMC (elimination)

Let us now consider the method from [87, 90, 91, 214, 15, 22, 16] that eliminates vanishing states from the EMC (EDTMC, in our terminology) corresponding to the underlying SMC of every GSPN N. The TPM for the resulting *reduced* EDTMC (REDTMC) has smaller size than that for the EDTMC. The method demonstrates that there exists a transformation of the underlying SMC of N into a CTMC, whose states are the tangible markings of N. This CTMC, which is essentially the *reduced* underlying SMC (RSMC) of N, is constructed on the basis of the REDTMC. The CTMC can then be directly solved to get both the transient and the steady-state PMFs over the tangible markings of N. In [91], the program and computational complexities of such

an *elimination* method, based on the REDTMC, were evaluated and compared with those of the *preservation* method that does not eliminate vanishing states and based on the EDTMC. The preservation method for GSPNs corresponds in dtsdPBC to the analysis of the underlying SMCs of expressions, called the *embedding* approach.

The elimination method for GSPNs can be easily transferred to dtsdPBC, hence, for every dynamic expression G, we can find a DTMC (since the sojourn time in the tangible states from $DR(G)$ is discrete and geometrically distributed) with the states from $DR_T(G)$, which can be directly solved to find the transient and the steady-state PMFs over the tangible states. We shall demonstrate that such a *reduced* DTMC (RDTMC) of G, denoted by $RDTMC(G)$, can be constructed from $DTMC(G)$, using the method analogous to that designed in [214, 15, 22, 16] in the framework of GSPNs to transform EDTMC into REDTMC. Since the sojourn time in the vanishing states is zero, the state changes of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, unlike those of $EDTMC(G)$, which happen only when the current state changes to some *different* one, irrespective of the global time. Therefore, in our case, we can skip the stages of constructing the REDTMC of G, denoted by $REDTMC(G)$, from $EDTMC(G)$, and recovering RSMC of G, denoted by $RSMC(G)$, (which is the sought-for DTMC) from $REDTMC(G)$, since we shall have $RSMC(G) = RDTMC(G)$.

Let G be a dynamic expression and $\mathbf{P}$ be the TPM for $DTMC(G)$. We reorder the states from $DR(G)$ so that the first rows and columns of $\mathbf{P}$ will correspond to the states from $DR_V(G)$ and the last ones will correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The resulting matrix can be decomposed as follows:

$$\mathbf{P} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}.$$

The elements of the $(n-m) \times (n-m)$ submatrix $\mathbf{C}$ are the probabilities to move from vanishing to vanishing states, and those of the $(n-m) \times m$ submatrix $\mathbf{D}$ are the probabilities to move from vanishing to tangible states. The elements of the $m \times (n-m)$ submatrix $\mathbf{E}$ are the probabilities to move from tangible to vanishing states, and those of the $m \times m$ submatrix $\mathbf{F}$ are the probabilities to move from tangible to tangible states.

The TPM $\mathbf{P}^\diamond$ for $RDTMC(G)$ is the $m \times m$ matrix, calculated as

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD},$$

where the elements of the matrix $\mathbf{G}$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states.

If there are no loops among vanishing states then for any vanishing state there exists a value $l \in I\!N$ such that every sequence of state changes that starts in a vanishing state and is longer than l should reach a tangible state. Thus, $\exists l \in I\!N\ \forall k > l\ \mathbf{C}^k = \mathbf{0}$ and $\sum_{k=0}^{\infty} \mathbf{C}^k = \sum_{k=0}^{l} \mathbf{C}^k$. If there are loops among vanishing states then all such loops are supposed to be of "transient" rather than "absorbing" type, since the latter is treated as a specification error to be corrected, like in [214, 16]. We have earlier required that $SMC(G)$ has a single closed communication (which is also ergodic) class of states. Remember that a communication class of states is their equivalence class with respect to communication relation, i.e. a maximal subset of communicating states. A communication class of states is closed if only the states belonging to it are accessible from every its state. The ergodic class cannot consist of vanishing states only to avoid "absorbing" loops among them, hence, it contains tangible states as well. Thus, any sequence of vanishing state changes that starts in the ergodic class will reach a tangible state at some time moment. All the states that do not belong to the ergodic class should be transient. Hence, any sequence of vanishing state changes that starts in a transient vanishing state will some time reach either a transient tangible state or a state from the ergodic class [265, 186, 49, 285, 188, 261, 263]. In the latter case, a tangible state will be reached as well, as argued above. Thus, every sequence of vanishing state changes in $SMC(G)$ that starts in a vanishing state will exit the set of all vanishing states in the future. This implies that the probabilities to move from vanishing to vanishing states in $k \in I\!N$ state changes, without traversal of tangible states, will lead to 0 when k tends to ∞. Then we have $\lim_{k\to\infty} \mathbf{C}^k = \lim_{k\to\infty} (\mathbf{I} - (\mathbf{I} - \mathbf{C}))^k = \mathbf{0}$, hence, $\mathbf{I} - \mathbf{C}$ is a non-singular matrix, i.e. its determinant is not equal to zero. Thus, the inverse matrix of $\mathbf{I} - \mathbf{C}$ exists and may be expressed by a Neumann series as $\sum_{k=0}^{\infty} (\mathbf{I} - (\mathbf{I} - \mathbf{C}))^k = \sum_{k=0}^{\infty} \mathbf{C}^k = (\mathbf{I} - \mathbf{C})^{-1}$. Therefore,

$$\mathbf{G} = \sum_{k=0}^{\infty} \mathbf{C}^k = \begin{cases} \sum_{k=0}^{l} \mathbf{C}^k, & \exists l \in I\!N\ \forall k > l\ \mathbf{C}^k = \mathbf{0}, \quad \text{no loops among vanishing states;} \\ (\mathbf{I} - \mathbf{C})^{-1}, & \lim_{k\to\infty} \mathbf{C}^k = \mathbf{0}, \quad \text{loops among vanishing states;} \end{cases}$$

where $\mathbf{0}$ is the square matrix consisting only of zeros and $\mathbf{I}$ is the identity matrix, both of order $n - m$.

For $1 \le i, j \le m$ and $1 \le k, l \le n - m$, let $\mathcal{F}_{ij}$ be the elements of the matrix $\mathbf{F}$, $\mathcal{E}_{ik}$ be those of $\mathbf{E}$, $\mathcal{G}_{kl}$ be those of $\mathbf{G}$ and $\mathcal{D}_{lj}$ be those of $\mathbf{D}$. By definition, the elements $\mathcal{P}^\diamond_{ij}$ of the matrix $\mathbf{P}^\diamond$ are calculated as

$$\mathcal{P}^{\diamond}_{ij} = \mathcal{F}_{ij} + \sum_{k=1}^{n-m}\sum_{l=1}^{n-m} \mathcal{E}_{ik}\mathcal{G}_{kl}\mathcal{D}_{lj} = \mathcal{F}_{ij} + \sum_{k=1}^{n-m} \mathcal{E}_{ik} \sum_{l=1}^{n-m} \mathcal{G}_{kl}\mathcal{D}_{lj} = \mathcal{F}_{ij} + \sum_{l=1}^{n-m} \mathcal{D}_{lj} \sum_{k=1}^{n-m} \mathcal{E}_{ik}\mathcal{G}_{kl},$$

i.e. $\mathcal{P}^{\diamond}_{ij}$ $(1 \le i, j \le m)$ is the total probability to move from the tangible state s_i to the tangible state s_j in any number of steps, without traversal of tangible states, but possibly going through vanishing states.

Let $s, \tilde{s} \in DR_T(G)$ such that $s = s_i,\ \tilde{s} = s_j$. The *probability to move from s to $\tilde{s}$ in any number of steps, without traversal of tangible states* (if such a movement is possible, i.e. its probability is positive) is

$$PM^{\diamond}(s, \tilde{s}) = \mathcal{P}^{\diamond}_{ij}.$$

Definition 5.3 *Let G be a dynamic expression and $[G]_{\approx} \in DR_T(G)$. The* reduced discrete time Markov chain (RDTMC) *of G, denoted by $RDTMC(G)$, has the state space $DR_T(G)$, the initial state $[G]_{\approx}$ and the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^{\diamond}(s, \tilde{s})$.*

RDTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $RDTMC(E) = RDTMC(\overline{E})$.

Let us now try to define $RSMC(G)$ as a "restriction" of $SMC(G)$ to its tangible states. Since the sojourn time in the tangible states of $SMC(G)$ is discrete and geometrically distributed, we can see that $RSMC(G)$ is a DTMC with the state space $DR_T(G)$, the initial state $[G]_{\approx}$ and the transitions whose probabilities collect all those in $SMC(G)$ to move from the tangible to the tangible states, directly or indirectly, namely, by going through its vanishing states only. Thus, $RSMC(G)$ has the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^{\diamond}(s, \tilde{s})$, hence, we get $RSMC(G) = RDTMC(G)$.

One can see that $RDTMC(G)$ is constructed from $DTMC(G)$ as follows. All vanishing states and all transitions to, from and between them are removed. All transitions between tangible states are preserved. The probabilities of transitions between tangible states may become greater and new transitions between tangible states may be added, both iff there exist moves between these tangible states in any number of steps, going through vanishing states only. Thus, for each sequence of transitions between two tangible states in $DTMC(G)$ there exists a (possibly shorter, since the eventual passed through vanishing states are removed) sequence between the same states in $RDTMC(G)$ and vice versa. If $DTMC(G)$ is irreducible then all its states (including tangible ones) communicate, hence, all states of $RDTMC(G)$ communicate as well and it is irreducible. Since both $DTMC(G)$ and $RDTMC(G)$ are finite, they are positive recurrent. Thus, in case of irreducibility of $DTMC(G)$, each of them has a single stationary PMF. Note that $DTMC(G)$ and/or $RDTMC(G)$ may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that $DTMC(G)$ is aperiodic while $RDTMC(G)$ is periodic due to removing vanishing states from the former.

Let $DR_T(G) = \{s_1, \ldots, s_m\}$ and $[G]_{\approx} \in DR_T(G)$. Then the transient ($k$-step, $k \in \mathbb{N}$) PMF $\psi^{\diamond}[k] = (\psi^{\diamond}[k](s_1), \ldots, \psi^{\diamond}[k](s_m))$ for $RDTMC(G)$ is calculated as

$$\psi^{\diamond}[k] = \psi^{\diamond}[0](\mathbf{P}^{\diamond})^k,$$

where $\psi^{\diamond}[0] = (\psi^{\diamond}[0](s_1), \ldots, \psi^{\diamond}[0](s_m))$ is the initial PMF defined as

$$\psi^{\diamond}[0](s_i) = \begin{cases} 1, & s_i = [G]_{\approx}; \\ 0, & \text{otherwise.} \end{cases}$$

Note also that $\psi^{\diamond}[k+1] = \psi^{\diamond}[k]\mathbf{P}^{\diamond}$ $(k \in \mathbb{N})$.

The steady-state PMF $\psi^{\diamond} = (\psi^{\diamond}(s_1), \ldots, \psi^{\diamond}(s_m))$ for $RDTMC(G)$ is a solution of the equation system

$$\begin{cases} \psi^{\diamond}(\mathbf{P}^{\diamond} - \mathbf{I}) = \mathbf{0} \\ \psi^{\diamond}\mathbf{1}^T = 1 \end{cases},$$

where $\mathbf{I}$ is the identity matrix of order m and $\mathbf{0}$ is a row vector of m values 0, $\mathbf{1}$ is that of m values 1.

Note that the vector $\psi^{\diamond}$ exists and is unique if $RDTMC(G)$ is ergodic. Then $RDTMC(G)$ has a single steady state, and we have $\psi^{\diamond} = \lim_{k\to\infty} \psi^{\diamond}[k]$.

The zero sojourn times in the vanishing states guarantee that the state changes of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, i.e. every such state change occurs after one time unit delay. Hence, the sojourn time in the tangible states is the same for $RDTMC(G)$ and $SMC(G)$. The state change probabilities of $RDTMC(G)$ are those to move from tangible to tangible states in any number of steps, without traversal of tangible states. Thus, $RDTMC(G)$ and $SMC(G)$ have the same transient behaviour over the tangible states, thus, the transient analysis of $SMC(G)$ is possible to accomplish using $RDTMC(G)$.

The following proposition relates the steady-state PMFs for $SMC(G)$ and $RDTMC(G)$. It proves that the steady-state probabilities of the tangible states coincide for them.

Proposition 5.2 *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and $\psi^\diamond$ be the steady-state PMF for $RDTMC(G)$. Then $\forall s \in DR(G)$*

$$\varphi(s) = \begin{cases} \psi^\diamond(s), & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

Proof. To make the proof more clear, we use the following unified notation. $\mathbf{I}$ denotes the identity matrices of the appropriate size. $\mathbf{0}$ denotes square matrices and row vectors of the appropriate size and length of values 0. $\mathbf{1}$ denotes square matrices and row vectors of the appropriate size and length of values 1.

Let $\mathbf{P}$ be the reordered TPM for $DTMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$, i.e. ψ is a solution of the equation system

$$\begin{cases} \psi(\mathbf{P} - \mathbf{I}) = \mathbf{0} \\ \psi \mathbf{1}^T = 1 \end{cases}.$$

Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The decomposed $\mathbf{P}$, $\mathbf{P} - \mathbf{I}$ and ψ are

$$\mathbf{P} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}, \ \mathbf{P} - \mathbf{I} = \begin{pmatrix} \mathbf{C} - \mathbf{I} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} - \mathbf{I} \end{pmatrix} \text{ and } \psi = (\psi_V, \psi_T),$$

where $\psi_V = (\psi_1, \ldots, \psi_{n-m})$ is the subvector of ψ with the steady-state probabilities of vanishing states and $\psi_T = (\psi_{n-m+1}, \ldots, \psi_n)$ is that with the steady-state probabilities of tangible states.

Then the equation system for ψ is decomposed as follows:

$$\begin{cases} \psi_V(\mathbf{C} - \mathbf{I}) + \psi_T \mathbf{E} = \mathbf{0} \\ \psi_V \mathbf{D} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \\ \psi_V \mathbf{1}^T + \psi_T \mathbf{1}^T = 1 \end{cases}.$$

Further, let $\mathbf{P}^\diamond$ be the TPM for $RDTMC(G)$. Then $\psi^\diamond$ is a solution of the equation system

$$\begin{cases} \psi^\diamond(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0} \\ \psi^\diamond \mathbf{1}^T = 1 \end{cases}.$$

We have

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD},$$

where the matrix $\mathbf{G}$ can have two different forms, depending on whether the loops among vanishing states exist, hence, we consider the two following cases.

1. There exist *no loops among vanishing states*. We have $\exists l \in I\!N \ \forall k > l \ \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^k$.

   Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

   $$\psi_V(\mathbf{CG} - \mathbf{G}) + \psi_T \mathbf{EG} = \mathbf{0}.$$

   Taking into account that $\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^k$, we get

   $$\psi_V \left(\sum_{k=1}^{l} \mathbf{C}^k + \mathbf{C}^{l+1} - \mathbf{C}^0 - \sum_{k=1}^{l} \mathbf{C}^k \right) + \psi_T \mathbf{EG} = \mathbf{0}.$$

   Since $\mathbf{C}^0 = \mathbf{I}$ and $\mathbf{C}^{l+1} = \mathbf{0}$, we obtain

   $$-\psi_V + \psi_T \mathbf{EG} = \mathbf{0} \text{ and } \psi_V = \psi_T \mathbf{EG}.$$

   Let us substitute ψ_V with $\psi_T \mathbf{EG}$ in the second equation of the decomposed equation system for ψ:

   $$\psi_T \mathbf{EGD} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \text{ and } \psi_T(\mathbf{F} + \mathbf{EGD} - \mathbf{I}) = \mathbf{0}.$$

   Since $\mathbf{F} + \mathbf{EGD} = \mathbf{P}^\diamond$, we have

   $$\psi_T(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0}.$$

2. There exist *loops among vanishing states*. We have $\lim_{k\to\infty} \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = (\mathbf{I} - \mathbf{C})^{-1}$.

   Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

$$-\psi_V(\mathbf{I} - \mathbf{C})\mathbf{G} + \psi_T\mathbf{E}\mathbf{G} = \mathbf{0}.$$

   Taking into account that $\mathbf{G} = (\mathbf{I} - \mathbf{C})^{-1}$, we get

$$-\psi_V + \psi_T\mathbf{E}\mathbf{G} = \mathbf{0} \text{ and } \psi_V = \psi_T\mathbf{E}\mathbf{G}.$$

   Let us substitute ψ_V with $\psi_T\mathbf{E}\mathbf{G}$ in the second equation of the decomposed equation system for ψ:

$$\psi_T\mathbf{E}\mathbf{G}\mathbf{D} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \text{ and } \psi_T(\mathbf{F} + \mathbf{E}\mathbf{G}\mathbf{D} - \mathbf{I}) = \mathbf{0}.$$

   Since $\mathbf{F} + \mathbf{E}\mathbf{G}\mathbf{D} = \mathbf{P}^\diamond$, we have

$$\psi_T(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0}.$$

The third equation $\psi_V\mathbf{1}^T + \psi_T\mathbf{1}^T = 1$ of the decomposed equation system for ψ implies that if ψ_V has non-zero elements then the sum of the elements of ψ_T is less than one. We normalize ψ_T by dividing its elements by their sum:

$$v = \frac{1}{\psi_T\mathbf{1}^T}\psi_T.$$

It is easy to check that v is a solution of the equation system

$$\begin{cases} v(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0} \\ v\mathbf{1}^T = 1 \end{cases},$$

hence, it is the steady-state PMF for $RDTMC(G)$ and we have

$$\psi^\diamond = v = \frac{1}{\psi_T\mathbf{1}^T}\psi_T.$$

Note that $\forall s \in DR_T(G)\ \psi_T(s) = \psi(s)$. Then the elements of $\psi^\diamond$ are calculated as follows: $\forall s \in DR_T(G)$

$$\psi^\diamond(s) = \frac{\psi_T(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi_T(\tilde{s})} = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}.$$

By Proposition 5.1, $\forall s \in DR_T(G)\ \varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}$.

Therefore, $\forall s \in DR_T(G)$

$$\varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})} = \psi^\diamond(s).$$

□

Thus, to calculate φ, one can just take all the elements of $\psi^\diamond$ as the steady-state probabilities of the tangible states, instead of abstracting from self-loops with probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. We call that technique *elimination*, since we eliminate the vanishing states. Hence, using $RDTMC(G)$ instead of $EDTMC(G)$ allows one to avoid such a multistage analysis, but constructing $\mathbf{P}^\diamond$ also requires some efforts, including calculating matrix powers or inverse matrices. Note that $RDTMC(G)$ may have self-loops with probability less than 1, unlike $EDTMC(G)$, hence, the behaviour of $RDTMC(G)$ may stabilize slower than that of $EDTMC(G)$ (if each of them has a single steady state). On the other hand, $\mathbf{P}^\diamond$ is generally smaller and denser matrix than $\mathbf{P}^*$, since $\mathbf{P}^\diamond$ may have additional non-zero elements not only at the main diagonal, but also many of them outside it. Therefore, in most cases, we have less time-consuming numerical calculation of $\psi^\diamond$ with respect to ψ^*. At the same time, the complexity of the analytical calculation of $\psi^\diamond$ with respect to ψ^* depends on the model structure, such as the number of vanishing states and loops among them, but usually it is lower, since the matrix size reduction plays an important role in many cases. Hence, for the system models with many immediate activities, we normally have a significant simplification of the solution. At the abstraction level of SMCs, the elimination of vanishing states decreases their impact to the solution complexity while allowing immediate activities to specify a comprehensible logical structure of systems at the higher level of transition systems.

Example 5.6 *Let E be from Example 3.24. Remember that $DR_T(\overline{E}) = DR_{ST}(\overline{E}) \cup DR_{WT}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{E}) = \{s_3\}$. We reorder the states from $DR(\overline{E})$, by moving vanishing states to the first positions: s_3, s_1, s_2, s_4, s_5.*

The reordered TPM for $DTMC(\overline{E})$ is

$$\mathbf{P}_r = \begin{pmatrix} 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1-\rho & \rho & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \theta & 1-\theta & 0 \\ 0 & 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

The result of the decomposing $\mathbf{P}_r$ are the matrices

$$\mathbf{C} = 0,\ \mathbf{D} = \left(0, 0, \frac{l}{l+m}, \frac{m}{l+m}\right),\ \mathbf{E} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ \mathbf{F} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

Since $\mathbf{C}^1 = 0$, we have $\forall k > 0\ \mathbf{C}^k = 0$, hence, $l = 0$ and there are no loops among vanishing states. Then

$$\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^k = \mathbf{C}^0 = \mathbf{I}.$$

Further, the TPM for $RDTMC(\overline{E})$ is

$$\mathbf{P}^{\diamond} = \mathbf{F} + \mathbf{EGD} = \mathbf{F} + \mathbf{EID} = \mathbf{F} + \mathbf{ED} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

In Figure 33, the reduced DTMC $RDTMC(\overline{E})$ is presented. The steady-state PMF for $RDTMC(\overline{E})$ is

$$\psi^{\diamond} = \frac{1}{\theta\phi(l+m) + \phi l + \theta m}(0, \theta\phi(l+m), \phi l, \theta m).$$

Note that $\psi^{\diamond} = (\psi^{\diamond}(s_1), \psi^{\diamond}(s_2), \psi^{\diamond}(s_4), \psi^{\diamond}(s_5))$. By Proposition 5.2, we have

$$\begin{aligned} \varphi(s_1) &= 0, \\ \varphi(s_2) &= \tfrac{\theta\phi(l+m)}{\theta\phi(l+m)+\phi l+\theta m}, \\ \varphi(s_3) &= 0, \\ \varphi(s_4) &= \tfrac{\phi l}{\theta\phi(l+m)+\phi l+\theta m}, \\ \varphi(s_5) &= \tfrac{\theta m}{\theta\phi(l+m)+\phi l+\theta m}. \end{aligned}$$

Thus, the steady-state PMF for $SMC(\overline{E})$ is

$$\varphi = \frac{1}{\theta\phi(l+m) + \phi l + \theta m}(0, \theta\phi(l+m), 0, \phi l, \theta m).$$

This coincides with the result obtained in Example 5.1 with the use of ψ^* and SJ.

Example 5.7 *Let E be from Example 3.24. In Figure 34, the reduced underlying SMC $RSMC(\overline{E})$ is depicted. The average sojourn times in the states of the reduced underlying SMC are written next to them in bold font. In spite of the equality $RSMC(\overline{E}) = RDTMC(\overline{E})$, the graphical representation of $RSMC(\overline{E})$ differs from that of $RDTMC(\overline{E})$, since the former is based on the $REDTMC(\overline{E})$, where each state is decorated with the* positive *average sojourn time of $RSMC(\overline{E})$ in it. $REDTMC(\overline{E})$ is constructed from $EDTMC(\overline{E})$ in the similar way as $RDTMC(\overline{E})$ is obtained from $DTMC(\overline{E})$. By construction, the residence time in each state of $RSMC(\overline{E})$ is geometrically distributed. Hence, the associated parameter of geometrical distribution is uniquely recovered from the average sojourn time in the state.*

$RDTMC(\overline{E})$

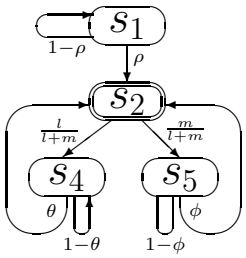

Figure 33: The reduced DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

$RSMC(\overline{E})$

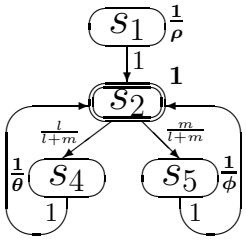

Figure 34: The reduced SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

Let us now formally prove that RSMC coincides with RDTMC. Although this assertion is very intuitive, its proof is rather involved. The relation between $DTMC$ and $RDTMC$ is obtained using the transition function $PM^\diamond(s, \tilde{s})$, based on $PM(s, \tilde{s})$. The relation between $RDTMC$ and the embedded $RDTMC$ ($ERDTMC$) is obtained using the transition function $(PM^\diamond)^*(s, \tilde{s})$, based on $PM^\diamond(s, \tilde{s})$. The relation between $EDTMC$ and the reduced $EDTMC$ ($REDTMC$) is obtained using the transition function $(PM^*)^\diamond(s, \tilde{s})$, based on $PM^*(s, \tilde{s})$. Let G be a dynamic expression. We shall prove that the TPM $(\mathbf{P}^\diamond)^*$ for the embedded $RDTMC(G)$ ($ERDTMC(G)$), (forwardly) constructed by reduction (eliminating vanishing states) of $DTMC(G)$, followed by embedding $ERDTMC(G)$ into $RDTMC(G)$, *coincides with* the (finally) embedded TPM $((\mathbf{P}^*)^\diamond)^*$, (reversely) constructed by embedding $EDTMC(G)$ into $SMC(G)$, followed by reduction $REDTMC(G)$ of $EDTMC(G)$, and final embedding $EREDTMC(G)$ into $RSMC(G)$. The final embedding in the reverse construction is needed, since new self-loops may arise after reducing $EDTMC(G)$, i.e. $REDTMC(G)$ may become not an EDTMC, but a DTMC featuring self-loops with probability less than 1. Note that for $s, \tilde{s} \in DR_T(G)$, we have $(PM^\diamond)^*(s, \tilde{s}) = SL^\diamond(s)PM^\diamond(s, \tilde{s})$ in $ERDTMC(G)$. Here $SL^\diamond(s)$ is the self-loops abstraction factor in s in $RDTMC(G)$. This corresponds to a different expression $(PM^*)^\diamond(s, \tilde{s}) = (SL \cdot PM)^\diamond(s, \tilde{s})$ in $REDTMC(G)$. In particular, $SL^\diamond(s) > SL(s)$ when $PM^\diamond(s, s) > PM(s, s)$, which is the reason for a new self-loop associated with s in $RDTMC(G)$. As we shall see, in that case $(PM^\diamond)^*(s, \tilde{s}) > (PM^*)^\diamond(s, \tilde{s})$. The following theorem relates those finally embedded reduced embedded TPM $((\mathbf{P}^*)^\diamond)^*$ (i.e. the TPM for $EREDTMC(G)$) and embedded reduced TPM $(\mathbf{P}^\diamond)^*$ (the TPM for $ERDTMC(G)$).

Theorem 5.2 *Let G be a dynamic expression, $(\mathbf{P}^\diamond)^*$ results from embedding the TPM $\mathbf{P}^\diamond$ for $RDTMC(G)$, and $((\mathbf{P}^*)^\diamond)^*$ results from reduction and final embedding the TPM $\mathbf{P}^*$ for $EDTMC(G)$. Then*

$$((\mathbf{P}^*)^\diamond)^* = (\mathbf{P}^\diamond)^*.$$

Proof. See Appendix A.1. □

Thus, reduction before embedding is more optimal computationally for DTMCs of the process expressions.

By Theorem 5.2, $EREDTMC(G) = ERDTMC(G)$. The sojourn time in every $s \in DR_T(G)$ is geometrically distributed with the parameter $\frac{1}{SL_F(s)SL_{H'}(s)} = \frac{1}{SL^\diamond(s)}$, where $SL_{H'}(s) = \frac{1}{1-SL_F(s)PM_H(s,s)}$, while the sojourn time in every $s \in DR_V(G)$ is equal to 0. Here $SL_F(s)$ is the self-loops abstraction factor in s for the submatrix $\mathbf{F}$ (see Example 5.6), $PM_H(s, s)$ is the self-loop probability in s for the matrix $\mathbf{H} = \mathbf{EGD}$ (see Example 5.6) and $SL_{H'}(s)$ is the self-loops abstraction factor in s in $REDTMC(G)$ (for the matrix $\mathbf{H}'$, whose elements

are the probabilities to move from tangible to tangible states, via any *positive* number of vanishing states, without traversal of tangible states, in $EDTMC(G)$). Hence, $RSMC(G) = RDTMC(G)$, where $RSMC(G)$ is the SMC with the EDTMC $EREDTMC(G)$, such that $\frac{1}{SL_F(s)SL_{H'}(s)}$ is the geometrical distribution parameter of the sojourn time in every $s \in DR_T(G)$ while the sojourn time is zero in every $s \in DR_V(G)$.

Example 5.8 *Let E be from Example 3.24. The TPMs for $RDTMC(\overline{E})$ and $ERDTMC(\overline{E})$ are*

$$\mathbf{P}^{\diamond} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}, \ (\mathbf{P}^{\diamond})^* = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The TPMs for $REDTMC(\overline{E})$ and $EREDTMC(\overline{E})$ are

$$(\mathbf{P}^*)^{\diamond} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \ ((\mathbf{P}^*)^{\diamond})^* = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The self-loops abstraction subvector of $\overline{E}$ for the submatrix $\mathbf{F}$ (see Example 5.6) is $SL_F = \left(\frac{1}{\rho}, 1, \frac{1}{\theta}, \frac{1}{\phi}\right)$. The self-loops abstraction vector of $\overline{E}$ in $REDTMC(\overline{E})$ (for the matrix $\mathbf{H}'$, see below) is $(SL^*)^{\diamond} = SL_{H'} = (1,1,1,1)$. The self-loops abstraction vector of $\overline{E}$ in $RDTMC(\overline{E})$ is $SL^{\diamond} = \mathbf{1}Diag(SL_F)Diag(SL_{H'}) = \left(\frac{1}{\rho}, 1, \frac{1}{\theta}, \frac{1}{\phi}\right)$, where $\mathbf{1}$ is a row vector of n values 1.

The elements of the matrix $\mathbf{H}'$ are the probabilities to move from tangible to tangible states, via any positive *number of vanishing states, without traversal of tangible states, in $EDTMC(G)$. We have $\mathbf{H}' = Diag(SL_F)\mathbf{H}$, where elements of the matrix $\mathbf{H} = \mathbf{EGD}$ (see Example 5.6) are the probabilities to move from tangible to tangible states, via any* positive *number of vanishing states, without traversal of tangible states, in $DTMC(G)$. The matrices $\mathbf{H}$ and $\mathbf{H}'$ are*

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathbf{H}' = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then it is easy to check that

$$((\mathbf{P}^*)^{\diamond})^* = Diag(SL^{\diamond})(\mathbf{P}^{\diamond} - \mathbf{I}) + \mathbf{I} = Diag(SL_{H'})Diag(SL_F)(\mathbf{P}^{\diamond} - \mathbf{I}) + \mathbf{I} = (\mathbf{P}^{\diamond})^*.$$

Note that our reduction of the underlying SMC by eliminating its vanishing states, resulting in the reduced DTMC, partially resembles the hierarchical aggregation method from [100] for singularly perturbed finite state Markov processes with rare transitions. The method constructs a sequence of increasingly simplified (with consecutively reduced order) models and then combines them to approximate asymptotically the original process.

Our reduction technique also resembles the method from [208] that removes instantaneous states of stochastically discontinuous Markov reward chains. The latter are "limits" of continuous time Markov chains with state rewards and fast transitions when the rates (speeds) of these transitions tend to infinity, making them immediate. By analogy with this work, we could consider DTMCs extended with instantaneous states instead of SMCs with geometrically distributed or zero sojourn times in the states. However, within dtsdPBC, we have decided to take SMCs as the underlying stochastic process to be able to consider not only geometrically distributed and zero residence time in the states, but arbitrary fixed discrete time delays as well.

6 Stochastic equivalences

Consider the expressions $E = (\{a\}, \frac{1}{2})$ and $E' = (\{a\}, \frac{1}{3})_1[](\{a\}, \frac{1}{3})_2$, for which $\overline{E} \neq_{ts} \overline{E'}$, since $TS(\overline{E})$ has only one transition from the initial to the final state (with probability $\frac{1}{2}$) while $TS(\overline{E'})$ has two such ones (with probabilities $\frac{1}{4}$). On the other hand, all the mentioned transitions are labeled by activities with the same multiaction part $\{a\}$. Moreover, the overall probabilities of the mentioned transitions of $TS(\overline{E})$ and $TS(\overline{E'})$ coincide: $\frac{1}{2} = \frac{1}{4} + \frac{1}{4}$. Further, $TS(\overline{E})$ (as well as $TS(\overline{E'})$) has one empty loop transition from the initial state to itself with probability $\frac{1}{2}$ and one empty loop transition from the final state to itself with probability 1. The empty loop transitions are labeled by the empty multiset of activities. For calculating the transition probabilities of

$TS(\overline{E'})$, take $\rho = \chi = \frac{1}{3}$ in Example 3.9. Then you will see that the probability parts $\frac{1}{3}$ and $\frac{1}{3}$ of the activities $(\{a\}, \frac{1}{3})_1$ and $(\{a\}, \frac{1}{3})_2$ are "splitted" among probabilities $\frac{1}{4}$ and $\frac{1}{4}$ of the corresponding transitions and the probability $\frac{1}{2}$ of the empty loop transition. Unlike $=_{ts}$, most of the probabilistic and stochastic equivalences proposed in the literature do not differentiate between the processes such as those specified by E and E'. In Figure 36(a), the marked dtsd-boxes corresponding to the dynamic expressions $\overline{E}$ and $\overline{E'}$ are presented, i.e. $N = Box_{dtsd}(\overline{E})$ and $N' = Box_{dtsd}(\overline{E'})$.

Since the semantic equivalence $=_{ts}$ is too discriminating in many cases, we need weaker equivalence notions. These equivalences should possess the following necessary properties. First, any two equivalent processes must have the same sequences of multisets of multiactions, which are the multiaction parts of the activities executed in steps starting from the initial states of the processes. Second, for every such sequence, its execution probabilities within both processes must coincide. Third, the desired equivalence should preserve the branching structure of computations, i.e. the points of choice of an external observer between several extensions of a particular computation should be taken into account. In this section, we define one such notion: step stochastic bisimulation equivalence.

6.1 Step stochastic bisimulation equivalence

Bisimulation equivalences respect the particular points of choice in the behavior of a system. To define stochastic bisimulation equivalences, we have to consider a bisimulation as an *equivalence* relation that partitions the states of the *union* of the transition systems $TS(G)$ and $TS(G')$ of two dynamic expressions G and G' to be compared. For G and G' to be bisimulation equivalent, the initial states $[G]_{\approx}$ and $[G']_{\approx}$ of their transition systems should be related by a bisimulation having the following transfer property: if two states are related then in each of them the same multisets of multiactions can occur, leading with the identical overall probability from each of the two states to *the same equivalence class* for every such multiset.

Thus, we follow the approaches of [178, 194, 162, 164, 36, 26, 27], but we implement step semantics instead of interleaving one considered in these papers. Recall also that we use the generative probabilistic transition systems, like in [178], in contrast to the reactive model, treated in [194], and we take transition probabilities instead of transition rates from [162, 164, 36, 26, 27]. Thus, step stochastic bisimulation equivalence that we define further is (in the probabilistic sense) comparable only with interleaving probabilistic bisimulation equivalence from [178], and our equivalence is obviously stronger.

In the definition below, we consider $\mathcal{L}(\Upsilon) \in I\!N^{\mathcal{L}}_{fin}$ for $\Upsilon \in I\!N^{\mathcal{SIL}}_{fin}$, i.e. (possibly empty) multisets of multiactions. The multiactions can be empty as well. In this case, $\mathcal{L}(\Upsilon)$ contains the elements $\emptyset$, but it is not empty itself.

Let G be a dynamic expression and $\mathcal{H} \subseteq DR(G)$. Then, for any $s \in DR(G)$ and $A \in I\!N^{\mathcal{L}}_{fin}$, we write $s \stackrel{A}{\rightarrow}_{\mathcal{P}} \mathcal{H}$, where $\mathcal{P} = PM_A(s, \mathcal{H})$ is the *overall probability to move from s into the set of states $\mathcal{H}$ via steps with the multiaction part A* defined as

$$PM_A(s, \mathcal{H}) = \sum_{\{\Upsilon \mid \exists \tilde{s} \in \mathcal{H} \ s \stackrel{\Upsilon}{\rightarrow} \tilde{s}, \ \mathcal{L}(\Upsilon) = A\}} PT(\Upsilon, s).$$

We write $s \stackrel{A}{\rightarrow} \mathcal{H}$ if $\exists \mathcal{P} \ s \stackrel{A}{\rightarrow}_{\mathcal{P}} \mathcal{H}$. Further, we write $s \rightarrow_{\mathcal{P}} \mathcal{H}$ if $\exists A \ s \stackrel{A}{\rightarrow} \mathcal{H}$, where $\mathcal{P} = PM(s, \mathcal{H})$ is the *overall probability to move from s into the set of states $\mathcal{H}$ via any steps* defined as

$$PM(s, \mathcal{H}) = \sum_{\{\Upsilon \mid \exists \tilde{s} \in \mathcal{H} \ s \stackrel{\Upsilon}{\rightarrow} \tilde{s}\}} PT(\Upsilon, s).$$

For $\tilde{s} \in DR(G)$, we write $s \stackrel{A}{\rightarrow}_{\mathcal{P}} \tilde{s}$ if $s \stackrel{A}{\rightarrow}_{\mathcal{P}} \{\tilde{s}\}$ and $s \stackrel{A}{\rightarrow} \tilde{s}$ if $\exists \mathcal{P} \ s \stackrel{A}{\rightarrow}_{\mathcal{P}} \tilde{s}$.

To introduce a stochastic bisimulation between dynamic expressions G and G', we should consider the "composite" set of states $DR(G) \cup DR(G')$, since we have to identify the probabilities to come from any two equivalent states into the same "composite" equivalence class (with respect to the stochastic bisimulation). Note that, for $G \neq G'$, transitions starting from the states of $DR(G)$ (or $DR(G')$) always lead to those from the same set, since $DR(G) \cap DR(G') = \emptyset$, and this allows us to "mix" the sets of states in the definition of stochastic bisimulation.

Definition 6.1 *Let G and G' be dynamic expressions. An* equivalence *relation $\mathcal{R} \subseteq (DR(G) \cup DR(G'))^2$ is a* step stochastic bisimulation *between G and G', denoted by $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$, if:*

1. $([G]_{\approx}, [G']_{\approx}) \in \mathcal{R}$.
2. $(s_1, s_2) \in \mathcal{R}$ *implies* $SJ(s_1) = 0 \ \Leftrightarrow \ SJ(s_2) = 0$ *and* $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}} \ \forall A \in I\!N^{\mathcal{L}}_{fin}$

$$s_1 \stackrel{A}{\rightarrow}_{\mathcal{P}} \mathcal{H} \;\Leftrightarrow\; s_2 \stackrel{A}{\rightarrow}_{\mathcal{P}} \mathcal{H}.$$

Two dynamic expressions G and G' are step stochastic bisimulation equivalent*, denoted by $G \underline{\leftrightarrow}_{ss} G'$, if* $\exists \mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$.

Note that the condition $SJ(s_1) = 0 \;\Leftrightarrow\; SJ(s_2) = 0$ in item 2 of the definition above is needed to make difference between w-tangible states (all having at least one time unit sojourn times) and vanishing states (all having zero sojourn times). The reason is that both from w-tangible and vanishing states, no empty moves can be made, unlike s-tangible states, from which empty moves are always possible. When comparing dynamic expressions for step stochastic bisimulation equivalence, we can use empty moves only to make difference between s-tangible and other (w-tangible or vanishing) states. Without the mentioned condition, w-tangible and vanishing states could be related by the bisimulation. We intend to avoid such the relationships, since vanishing states are a special case that should be specifically treated in the proofs of our forthcoming results.

We now define the multiaction transition systems, whose transitions are labeled with the multisets of multiactions, extracted from the corresponding activities.

Definition 6.2 *Let G be a dynamic expression. The* (labeled probabilistic) multiaction transition system *of G is a quadruple $TS_{\mathcal{L}}(G) = (S_{\mathcal{L}}, L_{\mathcal{L}}, \mathcal{T}_{\mathcal{L}}, s_{\mathcal{L}})$, where*

- $S_{\mathcal{L}} = DR(G)$;
- $L_{\mathcal{L}} = \mathbb{N}_{fin}^{\mathcal{L}} \times (0; 1]$;
- $\mathcal{T}_{\mathcal{L}} = \{(s, (A, PM_A(s, \{\tilde{s}\})), \tilde{s}) \mid s, \tilde{s} \in DR(G),\ s \stackrel{A}{\rightarrow} \tilde{s}\}$;
- $s_{\mathcal{L}} = [G]_{\approx}$.

The transition $(s, (A, \mathcal{P}), \tilde{s}) \in \mathcal{T}_{\mathcal{L}}$ will be written as $s \stackrel{A}{\rightarrow}_{\mathcal{P}} \tilde{s}$.

The multiaction transition systems of static expressions can be defined as well. For $E \in RegStatExpr$ let $TS_{\mathcal{L}}(E) = TS_{\mathcal{L}}(\overline{E})$.

Let G and G' be dynamic expressions and $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. Then the relation $\mathcal{R}$ can be interpreted as a step stochastic bisimulation between the transition systems $TS_{\mathcal{L}}(G)$ and $TS_{\mathcal{L}}(G')$, denoted by $\mathcal{R} : TS_{\mathcal{L}}(G) \underline{\leftrightarrow}_{ss} TS_{\mathcal{L}}(G')$, which is defined by analogy (excepting step semantics) with interleaving probabilistic bisimulation on generative probabilistic transition systems from [178].

Example 6.1 *Let us consider an abstraction F of the static expression E from Example 3.24, such that $c = e,\ d = f,\ \theta = \phi$, i.e.*

$$F = [(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[]((\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}].$$

Then $DR(\overline{F}) = \{s'_1, s'_2, s'_3, s'_4, s'_5\}$ is obtained from $DR(\overline{E})$ via substitution of the symbols $e,\ f,\ \phi$ by $c,\ d,\ \theta$, respectively, in the specifications of the corresponding states from the latter set. We have $DR_{ST}(\overline{F}) = \{s'_1, s'_4, s'_5\}$, $DR_{WT}(\overline{F}) = \{s'_2\}$ and $DR_V(\overline{F}) = \{s'_3\}$. In Figure 35, the multiaction transition system $TS_{\mathcal{L}}(\overline{F})$ is presented. To simplify the graphical representation, the singleton multisets of multiactions are written without outer braces.

Example 6.2 *Let us interpret F from Example 6.1 as an abstraction of the travel system from Example 5.2. In such an* abstract *travel system, we do not differentiate between the transport facilities (trains or buses) that always have the same speed, but every l departures of the transport from the first platform take the same time as m departures of the transport from the second platform, and the traveler can choose between the two platforms.*

By taking $\theta = \phi$ in Example 5.2, we now calculate following the performance indices, based on the steady-state PMF for $SMC(\overline{F})$ $\varphi = \frac{1}{1+\theta}\left(0, \theta, 0, \frac{l}{l+m}, \frac{m}{l+m}\right)$ and the average sojourn time vector of $\overline{F}$ $SJ = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}, \frac{1}{\theta}\right)$.

- *The average time between comings to the successive cities (mean sightseeing and travel time) is $ReturnTime(s'_2) = \frac{1}{\varphi(s'_2)} = 1 + \frac{\theta l + \theta m}{\theta^2(l+m)} = 1 + \frac{1}{\theta}$.*
- *The fraction of time spent in a city (sightseeing time fraction) is $TimeFract(s'_2) = \varphi(s'_2) = \frac{\theta^2(l+m)}{\theta^2(l+m)+\theta l+\theta m} = \frac{\theta}{1+\theta}$.*
- *The fraction of time spent in a transport (travel time fraction) is $TimeFract(\{s'_4, s'_5\}) = \varphi(s'_4) + \varphi(s'_5) = \frac{\theta l+\theta m}{\theta^2(l+m)+\theta l+\theta m} = \frac{1}{1+\theta}$.*

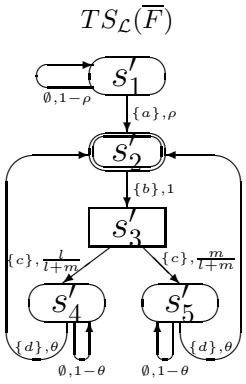

Figure 35: The multiaction transition system of $\overline{F}$ for $F = [(\{a\},\rho) * ((\{b\},\natural_k^1); ((((\{c\},\natural_l^0); (\{d\},\theta)_1)[]((\{c\},\natural_m^0); (\{d\},\theta)_2))) * \mathsf{Stop}]$

- *The relative fraction of time spent in a city with respect to that spent in transport (sightseeing relative to travel time fraction) is* $RltTimeFract(\{s'_2\},\{s'_4,s'_5\}) = \frac{\varphi(s'_2)}{\varphi(s'_4)+\varphi(s'_5)} = \frac{\theta^2(l+m)}{\theta l+\theta m} = \theta$.
- *The rate of leaving/entering a city (departure/arrival rate) is* $ExitFreq(s'_2) = \frac{\varphi(s'_2)}{SJ(s'_2)} = \frac{\theta^2(l+m)}{\theta^2(l+m)+\theta l+\theta m} = \frac{\theta}{1+\theta}$.

The following proposition states that every step stochastic bisimulation binds s-tangible states only with s-tangible ones, and the same is valid for w-tangible states, as well as for vanishing states.

Proposition 6.1 *Let G and G' be dynamic expressions and $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. Then*

$$\mathcal{R} \subseteq (DR_{ST}(G) \cup DR_{ST}(G'))^2 \uplus (DR_{WT}(G) \cup DR_{WT}(G'))^2 \uplus (DR_V(G) \cup DR_V(G'))^2.$$

Proof. By definition of transition systems of expressions, for every s-tangible state, there is an empty move from it, and no empty move transitions are possible from w-tangible or vanishing states. Further, $\mathcal{R}$ preserves empty moves. To verify this fact, first take $A = \emptyset$ in its definition to get $\forall (s_1,s_2) \in \mathcal{R}\ \forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ s_1 \stackrel{\emptyset}{\rightarrow}_{\mathcal{P}} \mathcal{H} \ \Leftrightarrow\ s_2 \stackrel{\emptyset}{\rightarrow}_{\mathcal{P}} \mathcal{H}$. Thus, $\mathcal{R}$ makes difference between s-tangible and all other (i.e. w-tangible or vanishing) states.

To verify that $\mathcal{R}$ also makes difference between w-tangible and vanishing states, we first notice that $\mathcal{R}$ preserves zero sojourn times, since $\forall (s_1,s_2) \in \mathcal{R}\ SJ(s_1) = 0 \ \Leftrightarrow\ SJ(s_2) = 0$. Then remember that the sojourn time in each vanishing state is equal to 0 while that in each w-tangible state is greater or equal to 1. □

Note that Proposition 6.1 implies $\mathcal{R} \subseteq (DR_T(G) \cup DR_T(G'))^2 \uplus (DR_V(G) \cup DR_V(G'))^2$, since $DR_T(G) = DR_{ST}(G) \uplus DR_{WT}(G)$ and $DR_T(G') = DR_{ST}(G') \uplus DR_{WT}(G')$. This fact will be used in (and is enough for) the proofs of the results from Section 8 on the stationary behaviour preservation.

Let $\mathcal{R}_{ss}(G,G') = \bigcup\{\mathcal{R} \mid \mathcal{R} : G \underline{\leftrightarrow}_{ss} G'\}$ be the *union of all step stochastic bisimulations* between G and G'. The following proposition proves that $\mathcal{R}_{ss}(G,G')$ is also an *equivalence* and $\mathcal{R}_{ss}(G,G') : G \underline{\leftrightarrow}_{ss} G'$.

Proposition 6.2 *Let G and G' be dynamic expressions and $G \underline{\leftrightarrow}_{ss} G'$. Then $\mathcal{R}_{ss}(G,G')$ is the largest step stochastic bisimulation between G and G'.*

Proof. See Appendix A.2. □

In [12], an algorithm for strong probabilistic bisimulation on labeled probabilistic transition systems (a reformulation of probabilistic automata) was proposed with time complexity $O(n^2 m)$, where n is the number of states and m is the number of transitions. In [14], a decision algorithm for strong probabilistic bisimulation on generative labeled probabilistic transition systems was constructed with time complexity $O(m \log n)$ and space complexity $O(m+n)$. In [85], a polynomial algorithm for strong probabilistic bisimulation on probabilistic automata was presented. The mentioned algorithms for interleaving probabilistic bisimulation equivalence can be adapted for $\underline{\leftrightarrow}_{ss}$ using the method from [175], applied to get the decidability results for step bisimulation equivalence. The method takes into account that transition systems in interleaving and step semantics differ only by availability of the additional transitions corresponding to parallel execution of activities in the latter (which is our case).

We now can establish a connection between operational and denotational semantics of dtsdPBC. Unlike the situation in dtsiPBC, we do not have an isomorphism between the two semantics in dtsdPBC. In particular, for an overlined static expression, multiple states of its transition system may be related to a single state of the reachability graph of its dtsd-box. The reason is that the decreasing timer values of each enabled "restricted" waiting multiaction from the the derived dynamic expressions generate different states in the transition system while there exists no corresponding waiting transition (and the associated timer) in the dtsd-box, hence, its respective state may stay the same with the time ticks. Thus, that reachability graph state relates to all such "generic" transition system states that differ only by their timer values. In Example 3.14, three states s_1, s_2, s_3 of $TS(\overline{E})$, such that $s_1 \stackrel{\emptyset}{\rightarrow}_{\frac{2}{3}} s_2 \stackrel{\emptyset}{\rightarrow}_{\frac{2}{3}} s_3 \stackrel{\emptyset}{\rightarrow}_{\frac{2}{3}} s_3$, are all related to the initial state Q_1 of $RG(Box_{dtsd}(\overline{E}))$. Thus, in dtsdPBC, like in tPBC [183], the deadlocked states are treated differently by the process expressions-based operational semantics and Petri net-based denotational semantics.

The following theorem shows that both the semantics are step stochastic bisimulation equivalent.

Theorem 6.1 *For any static expression E,*

$$TS(\overline{E}) \underline{\leftrightarrow}_{ss} RG(Box_{dtsd}(\overline{E})).$$

Proof. See Appendix A.3. □

6.2 Interrelations of the stochastic equivalences

We now compare the discrimination power of the stochastic equivalences.

Theorem 6.2 *For dynamic expressions G and G', the following* strict *implications hold:*

$$G \approx G' \;\Rightarrow\; G =_{ts} G' \;\Rightarrow\; G \underline{\leftrightarrow}_{ss} G'.$$

Proof. Let us check the validity of the implications.

- The implication $=_{ts} \rightarrow \underline{\leftrightarrow}_{ss}$ is proved as follows. Let $\beta : G =_{ts} G'$. Then it is easy to see that $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$, where $\mathcal{R} = \{(s, \beta(s)) \mid s \in DR(G)\}$.
- The implication $\approx \rightarrow =_{ts}$ is valid, since the transition system of a dynamic formula is defined based on its structural equivalence class.

Let us see that that the implications are strict, i.e. the reverse ones do not work, by the following counterexamples.

(a) Let $E = (\{a\}, \frac{1}{2})$ and $E' = (\{a\}, \frac{1}{3})_1 [] (\{a\}, \frac{1}{3})_2$. Then $\overline{E} \underline{\leftrightarrow}_{ss} \overline{E'}$, but $\overline{E} \neq_{ts} \overline{E'}$, since $TS(\overline{E})$ has only one transition from the initial to the final state while $TS(\overline{E'})$ has two such ones.

(b) Let $E = (\{a\}, \frac{1}{2}); (\{\hat{a}\}, \frac{1}{2})$ and $E' = ((\{a\}, \frac{1}{2}); (\{\hat{a}\}, \frac{1}{2}))$ sy a. Then $\overline{E} =_{ts} \overline{E'}$, but $\overline{E} \not\approx \overline{E'}$, since $\overline{E}$ and $\overline{E'}$ cannot be reached from each other by applying inaction rules. □

Example 6.3 *In Figure 36, the marked dtsd-boxes corresponding to the dynamic expressions from equivalence examples of Theorem 6.2 are presented, i.e. $N = Box_{dtsd}(\overline{E})$ and $N' = Box_{dtsd}(\overline{E'})$ for each picture (a)–(b).*

7 Reduction modulo equivalences

The equivalences which we proposed can be used to reduce transition systems and SMCs of expressions (reachability graphs and SMCs of dtsd-boxes). Reductions of graph-based models, like transition systems, reachability graphs and SMCs, result in those with less states (the graph nodes). The goal of the reduction is to decrease the number of states in the semantic representation of the modeled system while preserving its important qualitative and quantitative behavioural properties. Thus, the reduction allows one to simplify the functional and performance analysis of systems.

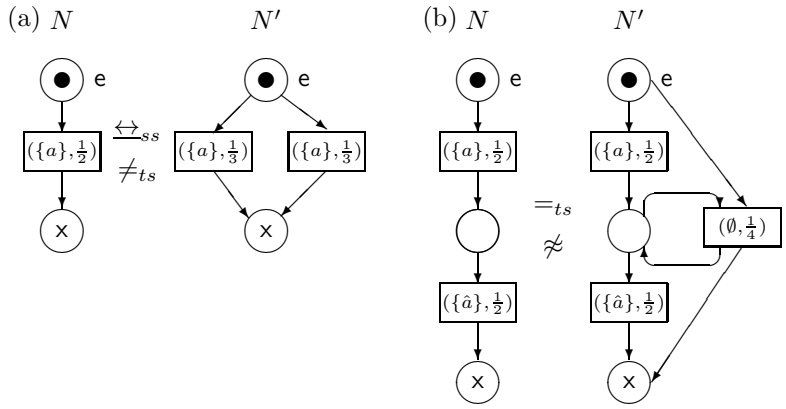

Figure 36: Dtsd-boxes of the dynamic expressions from equivalence examples of Theorem 6.2

7.1 Quotients of the transition systems and Markov chains

We now construct the quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems and Markov chains (SMCs, DTMCs and RDTMCs).

An *autobisimulation* is a bisimulation between an expression and itself. For a dynamic expression G and a step stochastic autobisimulation on it $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G$, let $\mathcal{K} \in DR(G)/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{K}$. We have $\forall \widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}}\ \forall A \in I\!N^{\mathcal{L}}_{fin}\ s_1 \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}} \Leftrightarrow s_2 \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$. The previous equality is valid for all $s_1, s_2 \in \mathcal{K}$, hence, we can rewrite it as $\mathcal{K} \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM_A(\mathcal{K}, \widetilde{\mathcal{K}}) = PM_A(s_1, \widetilde{\mathcal{K}}) = PM_A(s_2, \widetilde{\mathcal{K}})$.

We write $\mathcal{K} \stackrel{A}{\rightarrow} \widetilde{\mathcal{K}}$ if $\exists \mathcal{P}\ \mathcal{K} \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$ and $\mathcal{K} \rightarrow \widetilde{\mathcal{K}}$ if $\exists A\ \mathcal{K} \stackrel{A}{\rightarrow} \widetilde{\mathcal{K}}$. The similar arguments allow us to write $\mathcal{K} \rightarrow_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K}, \widetilde{\mathcal{K}}) = PM(s_1, \widetilde{\mathcal{K}}) = PM(s_2, \widetilde{\mathcal{K}})$.

By the note after Proposition 6.1, $\mathcal{R} \subseteq (DR_T(G))^2 \uplus (DR_V(G))^2$. Hence, $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}}$, all states from $\mathcal{K}$ are tangible, when $\mathcal{K} \in DR_T(G)/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{K} \in DR_V(G)/_{\mathcal{R}}$.

The *average sojourn time in the equivalence class (with respect to $\mathcal{R}$) of states* $\mathcal{K}$ is

$$SJ_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{1}{1-PM(\mathcal{K},\mathcal{K})}, & \mathcal{K} \in DR_T(G)/_{\mathcal{R}}; \\ 0, & \mathcal{K} \in DR_V(G)/_{\mathcal{R}}. \end{cases}$$

The *average sojourn time vector for the equivalence classes (with respect to $\mathcal{R}$) of states* of G, denoted by $SJ_{\mathcal{R}}$, has the elements $SJ_{\mathcal{R}}(\mathcal{K}),\ \mathcal{K} \in DR(G)/_{\mathcal{R}}$.

The *sojourn time variance in the equivalence class (with respect to $\mathcal{R}$) of states* $\mathcal{K}$ is

$$VAR_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{PM(\mathcal{K},\mathcal{K})}{(1-PM(\mathcal{K},\mathcal{K}))^2}, & \mathcal{K} \in DR_T(G)/_{\mathcal{R}}; \\ 0, & \mathcal{K} \in DR_V(G)/_{\mathcal{R}}. \end{cases}$$

The *sojourn time variance vector for the equivalence classes (with respect to $\mathcal{R}$) of states* of G, denoted by $VAR_{\mathcal{R}}$, has the elements $VAR_{\mathcal{R}}(\mathcal{K}),\ \mathcal{K} \in DR(G)/_{\mathcal{R}}$.

Let $\mathcal{R}_{ss}(G) = \bigcup\{\mathcal{R} \mid \mathcal{R} : G\underline{\leftrightarrow}_{ss}G\}$ be the *union of all step stochastic autobisimulations* on G. By Proposition 6.2, $\mathcal{R}_{ss}(G)$ is the largest step stochastic autobisimulation on G. Based on the equivalence classes with respect to $\mathcal{R}_{ss}(G)$, the quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems and the quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMCs of expressions can be defined. The mentioned equivalence classes become the quotient states. The average sojourn time in a quotient state is that in the corresponding equivalence class. Every quotient transition between two such composite states represents all steps (having the same multiaction part in case of the transition system quotient) from the first state to the second one.

Definition 7.1 *Let G be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) (labeled probabilistic) transition system *of G is a quadruple $TS_{\underline{\leftrightarrow}_{ss}}(G) = (S_{\underline{\leftrightarrow}_{ss}}, L_{\underline{\leftrightarrow}_{ss}}, \mathcal{T}_{\underline{\leftrightarrow}_{ss}}, s_{\underline{\leftrightarrow}_{ss}})$, where*

- $S_{\underline{\leftrightarrow}_{ss}} = DR(G)/_{\mathcal{R}_{ss}(G)}$;
- $L_{\underline{\leftrightarrow}_{ss}} = I\!N^{\mathcal{L}}_{fin} \times (0;1]$;
- $\mathcal{T}_{\underline{\leftrightarrow}_{ss}} = \{(\mathcal{K}, (A, PM_A(\mathcal{K}, \widetilde{\mathcal{K}})), \widetilde{\mathcal{K}}) \mid \mathcal{K}, \widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}_{ss}(G)},\ \mathcal{K} \stackrel{A}{\rightarrow} \widetilde{\mathcal{K}}\}$;
- $s_{\underline{\leftrightarrow}_{ss}} = [[G]_{\approx}]_{\mathcal{R}_{ss}(G)}$.

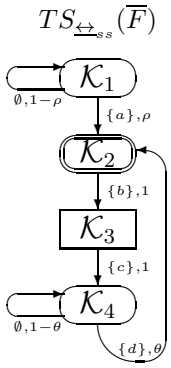

Figure 37: The quotient transition system of $\overline{F}$ for $F = [(\{a\},\rho) * ((\{b\},\natural_k^1); ((((\{c\},\natural_l^0); (\{d\},\theta)_1)[]((\{c\},\natural_m^0); (\{d\},\theta)_2))) * \mathsf{Stop}]$

The transition $(\mathcal{K},(A,\mathcal{P}),\widetilde{\mathcal{K}}) \in \mathcal{T}_{\underline{\leftrightarrow}_{ss}}$ will be written as $\mathcal{K} \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$.

The quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems of static expressions can be defined as well. For $E \in RegStatExpr$, let $TS_{\underline{\leftrightarrow}_{ss}}(E) = TS_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

Let G be a dynamic expression. We define the relation $\mathcal{R}_{\mathcal{L}ss}(G) = \{(s,\mathcal{K}),(\mathcal{K},s) \mid s \in \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}\}^+$, where $^+$ is the transitive closure operation. One can see that $\mathcal{R}_{\mathcal{L}ss}(G) \subseteq (DR(G) \cup DR(G)/_{\mathcal{R}_{ss}(G)})^2$ is an equivalence relation that partitions the set $DR(G) \cup DR(G)/_{\mathcal{R}_{ss}(G)}$ to the equivalence classes $\mathcal{L}_1,\ldots,\mathcal{L}_n$, defined as $\mathcal{L}_i = \mathcal{K}_i \cup \{\mathcal{K}_i\}$ $(1 \leq i \leq n)$, where $DR(G)/_{\mathcal{R}_{ss}(G)} = \{\mathcal{K}_1,\ldots,\mathcal{K}_n\}$. The relation $\mathcal{R}_{\mathcal{L}ss}(G)$ can be interpreted as a step stochastic bisimulation between the transition systems $TS_{\mathcal{L}}(G)$ and $TS_{\underline{\leftrightarrow}_{ss}}(G)$, denoted by $\mathcal{R}_{\mathcal{L}ss}(G) : TS_{\mathcal{L}}(G) \underline{\leftrightarrow}_{ss} TS_{\underline{\leftrightarrow}_{ss}}(G)$, which is defined by analogy (excepting step semantics) with interleaving probabilistic bisimulation on generative probabilistic transition systems from [178]. It is clear that from this viewpoint, $\mathcal{R}_{\mathcal{L}ss}(G)$ is also the union of all step stochastic bisimulations and largest step stochastic bisimulation between $TS_{\mathcal{L}}(G)$ and $TS_{\underline{\leftrightarrow}_{ss}}(G)$.

Example 7.1 *Let F be from Example 6.1. Then $DR(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1,\mathcal{K}_2,\mathcal{K}_3,\mathcal{K}_4\}$, where $\mathcal{K}_1 = \{s_1'\}$, $\mathcal{K}_2 = \{s_2'\}$, $\mathcal{K}_3 = \{s_3'\}$, $\mathcal{K}_4 = \{s_4',s_5'\}$. We have $DR_{ST}(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1,\mathcal{K}_4\}$, $DR_{WT}(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_2\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_3\}$. Thus, $\mathcal{R}_{ss}$ merges the states with the same "futures" from the different branches.*

In Figure 37, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.

The quotient (by $\underline{\leftrightarrow}_{ss}$) reachability graphs are defined similarly to the quotient transition systems. Let $\simeq$ denote isomorphism between quotient transition systems and quotient reachability graphs that binds their initial states. The following proposition establishes a connection between quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems of the overlined static expressions and quotient reachability graphs of their dtsd-boxes.

Proposition 7.1 *For any static expression E,*

$$TS_{\underline{\leftrightarrow}_{ss}}(\overline{E}) \simeq RG_{\underline{\leftrightarrow}_{ss}}(Box_{dtsd}(\overline{E})).$$

Proof. By definitions of the quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems and quotient reachability graphs, their uniqueness up to isomorphism and Theorem 6.1. □

Example 7.2 *Let F be from Example 6.1 and $N' = Box_{dtsd}(\overline{F})$. Then $RS(N')/_{\mathcal{R}_{ss}(N')} = \{\mathcal{L}_1,\mathcal{L}_2,\mathcal{L}_3,\mathcal{L}_4\}$, where $\mathcal{L}_1 = \{Q_1'\}$, $\mathcal{L}_2 = \{Q_2'\}$, $\mathcal{L}_3 = \{Q_3'\}$, $\mathcal{L}_4 = \{Q_4',Q_5'\}$ for $Q_1' = ((1,0,0,0,0,0),\infty)$, $Q_2' = ((0,1,0,0,0,0),1)$, $Q_3' = ((0,0,1,0,0,0),\infty)$, $Q_4' = ((0,0,0,1,0,0),\infty)$, $Q_5' = ((0,0,0,0,1,0),\infty)$. We have $RS_{ST}(N')/_{\mathcal{R}_{ss}(N')} = \{\mathcal{L}_1,\mathcal{L}_4\}$, $RS_{WT}(N')/_{\mathcal{R}_{ss}(N')} = \{\mathcal{L}_2\}$ and $RS_V(N')/_{\mathcal{R}_{ss}(N')} = \{\mathcal{L}_3\}$. In Figure 38, the quotient reachability graph $RG_{\underline{\leftrightarrow}_{ss}}(N')$ is presented. Note that $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ and $RG_{\underline{\leftrightarrow}_{ss}}(N')$ are isomorphic.*

The *quotient (by $\underline{\leftrightarrow}_{ss}$) average sojourn time vector* of G is defined as $SJ_{\underline{\leftrightarrow}_{ss}} = SJ_{\mathcal{R}_{ss}(G)}$.

The *quotient (by $\underline{\leftrightarrow}_{ss}$) sojourn time variance vector* of G is defined as $VAR_{\underline{\leftrightarrow}_{ss}} = VAR_{\mathcal{R}_{ss}(G)}$.

Let G be a dynamic expression and $\mathcal{K},\widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}_{ss}(G)}$. The transition system $TS_{\underline{\leftrightarrow}_{ss}}(G)$ can have self-loops going from an equivalence class to itself which have a non-zero probability. Clearly, the current equivalence class remains unchanged in this case.

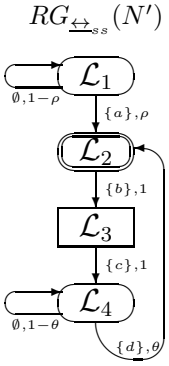

Figure 38: The quotient reachability graph of $N' = Box_{dtsd}(\overline{F})$ for $F = [(\{a\},\rho) * ((\{b\},\natural_k^1); ((((\{c\},\natural_l^0); (\{d\},\theta)_1)[]((\{c\},\natural_m^0); (\{d\},\theta)_2))) * \mathsf{Stop}]$

Let $\mathcal{K} \to \mathcal{K}$. The *probability to stay in* $\mathcal{K}$ *due to* k $(k \geq 1)$ *self-loops* is

$$PM(\mathcal{K},\mathcal{K})^k.$$

The *quotient (by* $\underline{\leftrightarrow}_{ss}$*) self-loops abstraction factor in the equivalence class* $\mathcal{K}$ is

$$SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = \begin{cases} \frac{1}{1-PM(\mathcal{K},\mathcal{K})}, & \mathcal{K} \to \mathcal{K}; \\ 1, & \text{otherwise.} \end{cases}$$

The *quotient (by* $\underline{\leftrightarrow}_{ss}$*) self-loops abstraction vector* of G, denoted by $SL_{\underline{\leftrightarrow}_{ss}}$, has the elements $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})$, $\mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$.

Let $\mathcal{K} \to \widetilde{\mathcal{K}}$ and $\mathcal{K} \neq \widetilde{\mathcal{K}}$, i.e. $PM(\mathcal{K},\mathcal{K}) < 1$. The *probability to move from* $\mathcal{K}$ *to* $\widetilde{\mathcal{K}}$ *by executing any multiset of activities after possible self-loops* is

$$PM^*(\mathcal{K},\widetilde{\mathcal{K}}) = \left\{ \begin{array}{ll} PM(\mathcal{K},\widetilde{\mathcal{K}}) \sum_{k=0}^{\infty} PM(\mathcal{K},\mathcal{K})^k = \frac{PM(\mathcal{K},\widetilde{\mathcal{K}})}{1-PM(\mathcal{K},\mathcal{K})}, & \mathcal{K} \to \mathcal{K}; \\ PM(\mathcal{K},\widetilde{\mathcal{K}}), & \text{otherwise;} \end{array} \right\} = SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) PM(\mathcal{K},\widetilde{\mathcal{K}}).$$

The value $k = 0$ in the summation above corresponds to the case when no self-loops occur.

Let $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$. If there exist self-loops from $\mathcal{K}$ (i.e. if $\mathcal{K} \to \mathcal{K}$) then $PM(\mathcal{K},\mathcal{K}) > 0$ and $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = \frac{1}{1-PM(\mathcal{K},\mathcal{K})} = SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})$. Otherwise, if there exist no self-loops from $\mathcal{K}$ then $PM(\mathcal{K},\mathcal{K}) = 0$ and $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = 1 = \frac{1}{1-PM(\mathcal{K},\mathcal{K})} = SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})$. Thus, $\forall \mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})$, hence, $\forall \mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ with $PM(\mathcal{K},\mathcal{K}) < 1$ it holds $PM^*(\mathcal{K},\widetilde{\mathcal{K}}) = SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) PM(\mathcal{K},\widetilde{\mathcal{K}})$. Note that the self-loops from the equivalence classes of tangible states are of the empty or non-empty type, the latter produced by iteration, since empty loops are not possible from the equivalence classes of w-tangible states, but they are possible from the equivalence classes of s-tangible states, while non-empty loops are possible from the equivalence classes of both s-tangible and w-tangible states.

Let $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$. We have $\forall \mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) \neq SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = 0$ and $\forall \mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ with $PM(\mathcal{K},\mathcal{K}) < 1$ it holds $PM^*(\mathcal{K},\widetilde{\mathcal{K}}) = SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) PM(\mathcal{K},\widetilde{\mathcal{K}})$. If there exist self-loops from $\mathcal{K}$ then $PM^*(\mathcal{K},\widetilde{\mathcal{K}}) = \frac{PM(\mathcal{K},\widetilde{\mathcal{K}})}{1-PM(\mathcal{K},\mathcal{K})}$ when $PM(\mathcal{K},\mathcal{K}) < 1$. Otherwise, if there exist no self-loops from $\mathcal{K}$ then $PM^*(\mathcal{K},\widetilde{\mathcal{K}}) = PM(\mathcal{K},\widetilde{\mathcal{K}})$. Note that the self-loops from the equivalence classes of vanishing states are always of the non-empty type, produced by iteration, since empty loops are not possible from the equivalence classes of vanishing states.

Definition 7.2 *Let* G *be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) EDTMC *of* G*, denoted by* $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$*, has the state space* $DR(G)/_{\mathcal{R}_{ss}(G)}$*, the initial state* $[[G]_{\approx}]_{\mathcal{R}_{ss}(G)}$ *and the transitions* $\mathcal{K} \twoheadrightarrow_{\mathcal{P}} \widetilde{\mathcal{K}}$*, if* $\mathcal{K} \to \widetilde{\mathcal{K}}$ *and* $\mathcal{K} \neq \widetilde{\mathcal{K}}$*, where* $\mathcal{P} = PM^*(\mathcal{K},\widetilde{\mathcal{K}})$*; or* $\mathcal{K} \twoheadrightarrow_1 \mathcal{K}$*, if* $PM(\mathcal{K},\mathcal{K}) = 1$*.*

The quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMC *of* G*, denoted by* $SMC_{\underline{\leftrightarrow}_{ss}}(G)$*, has the EDTMC* $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ *and the sojourn time in every* $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ *is geometrically distributed with the parameter* $1-PM(\mathcal{K},\mathcal{K})$ *while the sojourn time in every* $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ *is equal to zero.*

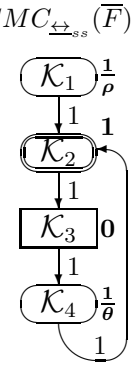

Figure 39: The quotient underlying SMC of $\overline{F}$ for $F = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[]((\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

The quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $SMC_{\underline{\leftrightarrow}_{ss}}(E) = SMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

The steady-state PMFs $\psi^*_{\underline{\leftrightarrow}_{ss}}$ for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $\varphi_{\underline{\leftrightarrow}_{ss}}$ for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ are defined like the corresponding notions ψ^* for $EDTMC(G)$ and φ for $SMC(G)$, respectively.

Example 7.3 *Let F be from Example 6.1. In Figure 39, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented. The average sojourn times in the states of the quotient underlying SMC are written next to them in bold font.*

The quotient average sojourn time vector of $\overline{F}$ is

$$SJ_{\underline{\leftrightarrow}_{ss}} = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}\right).$$

The quotient sojourn time variance vector of $\overline{F}$ is

$$VAR_{\underline{\leftrightarrow}_{ss}} = \left(\frac{1-\rho}{\rho^2}, 0, 0, \frac{1-\theta}{\theta^2}\right).$$

The TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The steady-state PMF for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\psi^*_{\underline{\leftrightarrow}_{ss}} = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right).$$

The steady-state PMF $\psi^*_{\underline{\leftrightarrow}_{ss}}$ weighted by $SJ_{\underline{\leftrightarrow}_{ss}}$ is

$$\left(0, \frac{1}{3}, 0, \frac{l}{3\theta}\right).$$

It remains to normalize the steady-state weighted PMF by dividing it by the sum of its components

$$\psi^*_{\underline{\leftrightarrow}_{ss}} SJ^T_{\underline{\leftrightarrow}_{ss}} = \frac{1+\theta}{3\theta}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\varphi_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+\theta}(0, \theta, 0, 1).$$

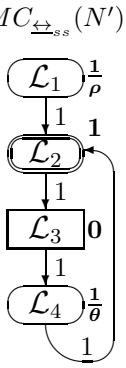

Figure 40: The quotient underlying SMC of $N' = Box_{dtsd}(\overline{F})$ for $F = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[] (\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

Example 7.4 *Let F be from Example 6.1. We now calculate the following performance indices, based on the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ $\varphi_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+\theta}(0, \theta, 0, 1)$ and the quotient average sojourn time vector of $\overline{F}$ $SJ_{\underline{\leftrightarrow}_{ss}} = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}\right)$.*

- *The average time between comings to the successive cities (mean sightseeing and travel time) is $ReturnTime(\mathcal{K}_2) = \frac{1}{\varphi(\mathcal{K}_2)} = 1 + \frac{1}{\theta}$.*
- *The fraction of time spent in a city (sightseeing time fraction) is $TimeFract(\mathcal{K}_2) = \varphi(\mathcal{K}_2) = \frac{\theta}{1+\theta}$.*
- *The fraction of time spent in a transport (travel time fraction) is $TimeFract(\mathcal{K}_4) = \varphi(\mathcal{K}_4) = \frac{1}{1+\theta}$.*
- *The relative fraction of time spent in a city with respect to that spent in transport (sightseeing relative to travel time fraction) is $RltTimeFract(\{\mathcal{K}_2\}, \{\mathcal{K}_4\}) = \frac{\varphi(\mathcal{K}_2)}{\varphi(\mathcal{K}_4)} = \theta$.*
- *The rate of leaving/entering a city (departure/arrival rate) is $ExitFreq(\mathcal{K}_2) = \frac{\varphi(\mathcal{K}_2)}{SJ(\mathcal{K}_2)} = \frac{\theta}{1+\theta}$.*

One can see that the performance indices are the same for the "complete" and the "quotient" abstract travel systems. The coincidence of the indices will illustrate the results of the forthcoming Proposition 8.1 and Proposition 8.2 (both modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{F})$).

Let $\simeq$ denote isomorphism between SMCs that binds their initial states, where two SMCs are isomorphic if their EDTMCs are so and the sojourn times in the isomorphic states of the EDTMCs are identically distributed. The following proposition establishes a connection between quotient (by $\underline{\leftrightarrow}_{ss}$) SMCs of the overlined static expressions and quotient SMCs of their dtsd-boxes.

Proposition 7.2 *For any static expression E*

$$SMC_{\underline{\leftrightarrow}_{ss}}(\overline{E}) \simeq SMC_{\underline{\leftrightarrow}_{ss}}(Box_{dtsd}(\overline{E})).$$

Proof. By definitions of the quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMCs for dynamic expressions and LDTSDPNs and Proposition 7.1, taking into account the following. First, for the associated SMCs, the average sojourn time in the states is the same, since it is defined via the analogous probability functions. Second, the transition probabilities of the associated SMCs are the sums of those belonging to the quotient transition systems or the quotient reachability graphs.

For instance, observe that the probability functions $PM(\mathcal{K}, \widetilde{\mathcal{K}})$ and $PM^*(\mathcal{K}, \widetilde{\mathcal{K}})$ can be respectively defined in the same way as $PM(\mathcal{L}, \widetilde{\mathcal{L}})$ and $PM^*(\mathcal{L}, \widetilde{\mathcal{L}})$, for the corresponding equivalence classes of the process states and net states $\mathcal{K}$ and $\mathcal{L}$, as well as $\widetilde{\mathcal{K}}$ and $\widetilde{\mathcal{L}}$. □

Example 7.5 *Let F be from Example 6.1 and $N' = Box_{dtsd}(\overline{F})$. In Figure 40, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(N')$ is presented. Note that $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ and $SMC_{\underline{\leftrightarrow}_{ss}}(N')$ are isomorphic. Thus, both the transient and steady-state PMFs for $SMC_{\underline{\leftrightarrow}_{ss}}(N')$ and $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ coincide.*

The quotients of both transition systems and underlying SMCs are the minimal reductions of the mentioned objects modulo step stochastic bisimulations. The quotients can be used to simplify analysis of system properties which are preserved by $\underline{\leftrightarrow}_{ss}$, since potentially less states should be examined for it. Such reduction method resembles that from [11] based on place bisimulation equivalence for PNs, excepting that the former method merges states, while the latter one merges places.

Moreover, the algorithms exist to construct the quotients of transition systems by an equivalence (like bisimulation one) [238] and those of (discrete or continuous time) Markov chains by ordinary lumping [119]. The algorithms have time complexity $O(m \log n)$ and space complexity $O(m + n)$, where n is the number of states and m is the number of transitions. As mentioned in [299], the algorithm from [119] can be easily adjusted to produce quotients of labeled probabilistic transition systems by the probabilistic bisimulation equivalence. In [299], the symbolic partition refinement algorithm on state space of CTMCs was proposed. The algorithm can be straightforwardly accommodated to DTMCs, interactive Markov chains (IMCs), Markov reward models, Markov decision processes (MDPs), Kripke structures and labeled probabilistic transition systems. Such a symbolic lumping uses memory efficiently due to compact representation of the state space partition. The symbolic lumping is time efficient, since fast algorithm of the partition representation and refinement is applied. In [120], a polynomial-time algorithm for minimizing behaviour of probabilistic automata by probabilistic bisimulation equivalence was outlined that results in the canonical quotient structures. One can adapt the above algorithms for our framework of transition systems, (reduced) DTMCs and SMCs.

Let us consider quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of expressions based on the state change probabilities $PM(\mathcal{K}, \widetilde{\mathcal{K}})$.

Definition 7.3 *Let G be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) DTMC *of G, denoted by $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, has the state space $DR(G)/_{\mathcal{R}_{ss}(G)}$, the initial state $[[G]_{\approx}]_{\mathcal{R}_{ss}(G)}$ and the transitions $\mathcal{K} \rightarrow_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K}, \widetilde{\mathcal{K}})$.*

The quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $DTMC_{\underline{\leftrightarrow}_{ss}}(E) = DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

The steady-state PMF $\psi_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ is defined like the corresponding notion ψ for $DTMC(G)$.

Example 7.6 *Let F be from Example 6.1. In Figure 41, the quotient DTMC $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.*

The TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \theta & 0 & 1-\theta \end{pmatrix}.$$

The steady-state PMF for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\psi_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+2\theta}(0, \theta, \theta, 1).$$

Remember that $DR_T(\overline{F})/_{\mathcal{R}_{ss}(F)} = DR_{ST}(\overline{F})/_{\mathcal{R}_{ss}(F)} \cup DR_{WT}(\overline{F})/_{\mathcal{R}_{ss}(F)} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(F)} = \{\mathcal{K}_3\}$. Hence,

$$\sum_{\mathcal{K} \in DR_T(\overline{F})/_{\mathcal{R}_{ss}(F)}} \psi(\mathcal{K}) = \psi(\mathcal{K}_1) + \psi(\mathcal{K}_2) + \psi(\mathcal{K}_4) = \frac{1+\theta}{1+2\theta}.$$

By the "quotient" analogue of Proposition 5.1, we have

$$\begin{aligned} &\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_1) = 0 \cdot \tfrac{1+2\theta}{1+\theta} = 0, \\ &\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_2) = \tfrac{\theta}{1+2\theta} \cdot \tfrac{1+2\theta}{1+\theta} = \tfrac{\theta}{1+\theta}, \\ &\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_3) = 0, \\ &\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4) = \tfrac{1}{1+2\theta} \cdot \tfrac{1+2\theta}{1+\theta} = \tfrac{1}{1+\theta}. \end{aligned}$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\varphi_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+\theta}(0, \theta, 0, 1).$$

This coincides with the result obtained in Example 7.3 with the use of $\psi^*_{\underline{\leftrightarrow}_{ss}}$ and $SJ_{\underline{\leftrightarrow}_{ss}}$.

Eliminating equivalence classes (with respect to $\mathcal{R}_{ss}(G)$) of vanishing states from the quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of expressions results in the reductions of such DTMCs.

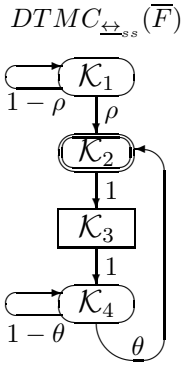

Figure 41: The quotient DTMC of $\overline{F}$ for $F = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[]((\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

Definition 7.4 *The* reduced quotient (by $\underline{\leftrightarrow}_{ss}$) DTMC *of G, denoted by $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, is defined like $RDTMC(G)$ in Section 5, but it is constructed from $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ instead of $DTMC(G)$.*

The reduced quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $RDTMC_{\underline{\leftrightarrow}_{ss}}(E) = RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.
The steady-state PMF $\psi^{\diamond}_{\underline{\leftrightarrow}_{ss}}$ for $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ is defined like the corresponding notion $\psi^{\diamond}$ for $RDTMC(G)$.

Example 7.7 *Let F be from Example 6.1. Remember that $DR_T(\overline{F})/_{\mathcal{R}_{ss}(F)} = DR_{ST}(\overline{F})/_{\mathcal{R}_{ss}(F)} \cup DR_{WT}(\overline{F})/_{\mathcal{R}_{ss}(F)} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(F)} = \{\mathcal{K}_3\}$. We reorder the states from $DR(\overline{F})/_{\mathcal{R}_{ss}(F)}$, by moving vanishing states to the first positions: $\mathcal{K}_3, \mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4$.*

The reordered TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}_{r\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1-\rho & \rho & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \theta & 1-\theta \end{pmatrix}.$$

The result of the decomposing $\mathbf{P}_{r\underline{\leftrightarrow}_{ss}}$ are the matrices

$$\mathbf{C}_{\underline{\leftrightarrow}_{ss}} = 0,\ \mathbf{D}_{\underline{\leftrightarrow}_{ss}} = (0,0,1),\ \mathbf{E}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},\ \mathbf{F}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 \\ 0 & 0 & 0 \\ 0 & \theta & 1-\theta \end{pmatrix}.$$

Since $\mathbf{C}^1_{\underline{\leftrightarrow}_{ss}} = 0$, we have $\forall k > 0\ \mathbf{C}^k_{\underline{\leftrightarrow}_{ss}} = 0$, hence, $l = 0$ and there are no loops among vanishing states. Then

$$\mathbf{G}_{\underline{\leftrightarrow}_{ss}} = \sum_{k=0}^{l} \mathbf{C}^k_{\underline{\leftrightarrow}_{ss}} = \mathbf{C}^0_{\underline{\leftrightarrow}_{ss}} = \mathbf{I}.$$

Further, the TPM for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}} = \mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}_{\underline{\leftrightarrow}_{ss}}\mathbf{G}_{\underline{\leftrightarrow}_{ss}}\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}_{\underline{\leftrightarrow}_{ss}}\mathbf{I}\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}_{\underline{\leftrightarrow}_{ss}}\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 \\ 0 & 0 & 1 \\ 0 & \theta & 1-\theta \end{pmatrix}.$$

In Figure 42, the reduced quotient DTMC $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented. The steady-state PMF for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\psi^{\diamond}_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+\theta}(0, \theta, 1).$$

Note that $\psi^{\diamond}_{\underline{\leftrightarrow}_{ss}} = (\psi^{\diamond}_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_1), \psi^{\diamond}_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_2), \psi^{\diamond}_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4))$. By the "quotient" analogue of Proposition 5.2, we have

$RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$

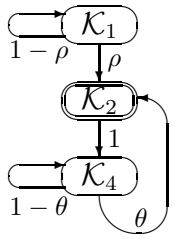

Figure 42: The reduced quotient DTMC of $\overline{F}$ for $F = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[]((\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

$RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$

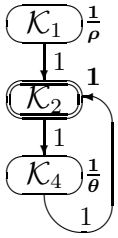

Figure 43: The reduced quotient SMC of $\overline{F}$ for $F = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta)_1)[]((\{c\}, \natural_m^0); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_1) = 0,$$
$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_2) = \tfrac{\theta}{1+\theta},$$
$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_3) = 0,$$
$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4) = \tfrac{1}{1+\theta}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ *is*

$$\varphi_{\underline{\leftrightarrow}_{ss}} = \frac{1}{1+\theta}(0, \theta, 0, 1).$$

This coincides with the result obtained in Example 7.3 with the use of $\psi^*_{\underline{\leftrightarrow}_{ss}}$ *and* $SJ_{\underline{\leftrightarrow}_{ss}}$.

Example 7.8 *Let F be from Example 6.1. In Figure 43, the reduced quotient SMC $RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is depicted. The average sojourn times in the states of the reduced quotient SMC are written next to them in bold font. In spite of the equality $RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F}) = RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$, the graphical representation of $RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ differs from that of $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$, since the former is based on the $REDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$, where each state is decorated with the* positive *average sojourn time of $RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ in it. $REDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ can be constructed from $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ in the similar way as $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ can be obtained from $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$. By construction, the residence time in each state of $RSMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is geometrically distributed. Hence, the associated parameter of geometrical distribution is uniquely recovered from the average sojourn time in the state.*

Let G be a dynamic expression. The coincidence of $RSMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ is formally proved like that of their "non-quotient" versions in Theorem 5.2 and the subsequent explanations.

Obviously, the relationships between the steady-state PMFs $\psi_{\underline{\leftrightarrow}_{ss}}$ and $\psi^*_{\underline{\leftrightarrow}_{ss}}$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ and $\psi_{\underline{\leftrightarrow}_{ss}}$, as well as $\varphi_{\underline{\leftrightarrow}_{ss}}$ and $\psi^{\diamond}_{\underline{\leftrightarrow}_{ss}}$, are the same as those determined between their "non-quotient" versions in Theorem 5.1, Proposition 5.1 and Proposition 5.2, respectively.

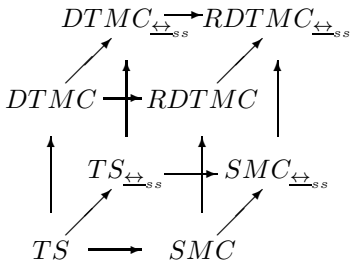

Figure 44: The cube of interrelations for the standard and quotient transition systems and Markov chains of the process expressions

7.2 Interrelations of the standard and quotient behavioural structures

In Figure 44, the cube of interconnections by the relation "constructed from" is depicted for both the standard and quotient transition systems and Markov chains (SMCs, DTMCs and RDTMCs) of the process expressions. The relations between SMC and $SMC_{\underline{\leftrightarrow}_{ss}}$, between $DTMC$ and $DTMC_{\underline{\leftrightarrow}_{ss}}$, as well as between $RDTMC$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}$, can be obtained using the following corresponding transition functions, defined by analogy with those already introduced: $PM^*(\mathcal{K},\widetilde{\mathcal{K}})$, based on $PM^*(s,\tilde{s})$, then $PM(\mathcal{K},\widetilde{\mathcal{K}})$, based on $PM(s,\tilde{s})$, as well as $PM^\diamond(\mathcal{K},\widetilde{\mathcal{K}})$, based on $PM^\diamond(s,\tilde{s})$ (all that to be proved below).

The relations between SMC and $RDTMC$, as well as between $SMC_{\underline{\leftrightarrow}_{ss}}$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}$, can be obtained using the next corresponding transition functions: $PM^\diamond(s,\tilde{s})$, based on $PM^*(s,\tilde{s})$, through $(PM^\diamond)^*(s,\tilde{s})$, as well as $PM^\diamond(\mathcal{K},\widetilde{\mathcal{K}})$, based on $PM^*(\mathcal{K},\widetilde{\mathcal{K}})$, through $(PM^\diamond)^*(\mathcal{K},\widetilde{\mathcal{K}})$ (by Theorem 5.2 and its "quotient" analogue).

In Figure 44, the relation (depicted by arrow) between $DTMC$ and $DTMC_{\underline{\leftrightarrow}_{ss}}$ is obtained using the transition function $PM(\mathcal{K},\widetilde{\mathcal{K}})$, based on $PM(s,\tilde{s})$. Let G be a dynamic expression. We shall prove that the (quotient) TPM $\mathbf{P}_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, (forwardly) constructed by quotienting (by $\underline{\leftrightarrow}_{ss}$) $TS(G)$, followed by extracting $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ from $TS_{\underline{\leftrightarrow}_{ss}}(G)$, *coincides with* the TPM $(\mathbf{P})_{\underline{\leftrightarrow}_{ss}}$, (reversely) constructed by extracting $DTMC(G)$ from $TS(G)$, followed by quotienting $DTMC(G)$. The following proposition relates those quotient extracted TPM $(\mathbf{P})_{\underline{\leftrightarrow}_{ss}}$ and extracted quotient TPM $\mathbf{P}_{\underline{\leftrightarrow}_{ss}}$.

Proposition 7.3 *Let G be a dynamic expression, $\mathbf{P}_{\underline{\leftrightarrow}_{ss}}$ be the TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $(\mathbf{P})_{\underline{\leftrightarrow}_{ss}}$ results from quotienting (by $\underline{\leftrightarrow}_{ss}$) the TPM $\mathbf{P}$ for $DTMC(G)$. Then*

$$(\mathbf{P})_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}.$$

Proof. Let $\mathcal{K},\widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}_{ss}(G)}$ and $s \in \mathcal{K}$.

In $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, we have $\sum_{A\in\mathbb{N}^{\mathcal{L}}_{fin}} PM_A(\mathcal{K},\widetilde{\mathcal{K}}) = \sum_{A\in\mathbb{N}^{\mathcal{L}}_{fin}} PM_A(s,\widetilde{\mathcal{K}}) =$
$\sum_{A\in\mathbb{N}^{\mathcal{L}}_{fin}} \sum_{\{\Upsilon\mid\exists\tilde{s}\in\widetilde{\mathcal{K}}\ s\xrightarrow{\Upsilon}\tilde{s},\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon,s) = \sum_{\{\Upsilon\mid\exists\tilde{s}\in\widetilde{\mathcal{K}}\ s\xrightarrow{\Upsilon}\tilde{s}\}} PT(\Upsilon,s) = PM(s,\widetilde{\mathcal{K}}) = PM(\mathcal{K},\widetilde{\mathcal{K}})$.

In the quotient of $DTMC(G)$, we have $\sum_{\tilde{s}\in\widetilde{\mathcal{K}}} PM(s,\tilde{s}) = \sum_{\tilde{s}\in\widetilde{\mathcal{K}}} \sum_{\{\Upsilon\mid s\xrightarrow{\Upsilon}\tilde{s}\}} PT(\Upsilon,s) =$
$\sum_{\{\Upsilon\mid\exists\tilde{s}\in\widetilde{\mathcal{K}}\ s\xrightarrow{\Upsilon}\tilde{s}\}} PT(\Upsilon,s) = PM(s,\widetilde{\mathcal{K}}) = PM(\mathcal{K},\widetilde{\mathcal{K}})$.

Thus, $(\mathbf{P})_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}$. □

Hence, the quotienting and extraction are permutable for transition systems of the process expressions. Applying extraction before the quotienting is useful to start from the level of Markov chains in the proofs.

Example 7.9 *Let F be from Example 6.1. The TPMs for $DTMC(\overline{F})$ and $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ are*

$$\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \theta & 0 & 0 & 1-\theta \end{pmatrix},\ \mathbf{P}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \theta & 0 & 1-\theta \end{pmatrix}.$$

The TPM for the quotient of $DTMC(\overline{F})$ is

$$(\mathbf{P})_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \theta & 0 & 1-\theta \end{pmatrix}.$$

Then it is clear that

$$(\mathbf{P})_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}.$$

In Figure 44, the relation (depicted by arrow) between SMC and $SMC_{\underline{\leftrightarrow}_{ss}}$ is obtained using the transition function $PM^*(\mathcal{K}, \widetilde{\mathcal{K}})$, based on $PM^*(s, \tilde{s})$. Let G be a dynamic expression. We shall prove that the (quotient) TPM $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$ for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, (forwardly) constructed by quotienting (by $\underline{\leftrightarrow}_{ss}$) $DTMC(G)$, followed by embedding $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ into $SMC_{\underline{\leftrightarrow}_{ss}}(G)$, *coincides with* the (finally) embedded TPM $(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}}$, (reversely) constructed by embedding $EDTMC(G)$ into $SMC(G)$, followed by quotienting $EDTMC(G)$, and final embedding a new EDTMC $EDTMC'(G)$ into the quotient of $EDTMC(G)$. The final embedding in the reverse construction is needed, since new self-loops may arise after quotienting $EDTMC(G)$, i.e. it may become not an EDTMC, but a DTMC featuring self-loops with probability less than 1. Note that for $\mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$ and $s \in \mathcal{K}$, we have $PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})PM(\mathcal{K}, \widetilde{\mathcal{K}}) = SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})PM(s, \widetilde{\mathcal{K}})$ in $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$. This corresponds to a different expression $\sum_{\tilde{s}\in\widetilde{\mathcal{K}}} PM^*(s, \tilde{s}) = \sum_{\tilde{s}\in\widetilde{\mathcal{K}}} SL(s)PM(s, \tilde{s}) = SL(s)\sum_{\tilde{s}\in\widetilde{\mathcal{K}}} PM(s, \tilde{s}) = SL(s)PM(s, \widetilde{\mathcal{K}})$ in the quotient of $EDTMC(G)$. In particular, $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) > SL(s)$ when $PM(s, \mathcal{K} \setminus \{s\}) > 0$, which is the reason for a new self-loop associated with s in the quotient of $EDTMC(G)$. The following proposition relates those finally embedded quotient embedded TPM $(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}}$ (i.e. the TPM for $EDTMC'(G)$) and embedded quotient TPM $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$.

Proposition 7.4 *Let G be a dynamic expression, $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$ be the TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}}$ results from quotienting (by $\underline{\leftrightarrow}_{ss}$) and final embedding the TPM $\mathbf{P}^*$ for $EDTMC(G)$. Then*

$$(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}.$$

Proof. See Appendix A.4. □

Thus, the quotienting before embedding is more optimal computationally for DTMCs of the process expressions.

By Proposition 7.4, $EDTMC'(G) = EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$. The sojourn time in every $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ is geometrically distributed with the parameter $\frac{1}{SL(s)SL'(s,\mathcal{K})} = \frac{1}{SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})}$, where $SL'(s, \mathcal{K}) = \frac{1}{1-SL(s)PM(s,\mathcal{K}\setminus\{s\})}$ while the sojourn time in every $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ is equal to 0. Here $SL'(s, \mathcal{K})$ is the self-loops abstraction factor in the equivalence class $\mathcal{K}$ with respect to the state $s \in \mathcal{K}$ for the quotient of $EDTMC(G)$. Hence, $SMC'(G) = SMC_{\underline{\leftrightarrow}_{ss}}(G)$, where $SMC'(G)$ is the SMC with the EDTMC $EDTMC'(G)$, such that $\frac{1}{SL(s)SL'(s,\mathcal{K})}$ is the geometrical distribution parameter of the sojourn time in every $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ while the sojourn time is zero in every $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$.

Example 7.10 *Let F be from Example 6.1. The TPMs for $EDTMC(\overline{F})$ and $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ are*

$$\mathbf{P}^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{P}^*_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The TPMs for the quotient of $EDTMC(\overline{F})$ and EDTMC of the quotient of $EDTMC(\overline{F})$ ($EDTMC'(\overline{F})$) are

$$(\mathbf{P}^*)_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad (\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Then it is clear that

$$(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}.$$

Let G be a dynamic expression. We now construct the quotient (by $\underline{\leftrightarrow}_{ss}$) of the TPM for $DTMC(G)$ using special collector and distributor matrices. Let $DR(G) = \{s_1, \ldots, s_n\}$ and $DR(G)/_{\mathcal{R}_{ss}(G)} = \{\mathcal{K}_1, \ldots, \mathcal{K}_l\}$.

The elements $(\mathcal{P}_{\underline{\leftrightarrow}_{ss}})_{rs}$ $(1 \leq r, s \leq l)$ of the TPM $\mathbf{P}_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ are defined as

$$(\mathcal{P}_{\underline{\leftrightarrow}_{ss}})_{rs} = \begin{cases} PM(\mathcal{K}_r, \mathcal{K}_s), & \mathcal{K}_r \to \mathcal{K}_s; \\ 0, & \text{otherwise.} \end{cases}$$

Like it has been done for strong performance bisimulation on labeled CTSPNs in [75], the $l \times l$ TPM $\mathbf{P}_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ can be constructed from the $n \times n$ TPM $\mathbf{P}$ for $DTMC(G)$ using the $n \times l$ *collector matrix* $\mathbf{V}$ for the largest step stochastic autobisimulation $\mathcal{R}_{ss}(G)$ on G and the $l \times n$ *distributor matrix* $\mathbf{W}$ for $\mathbf{V}$. Then $\mathbf{W}$ should be a non-negative matrix (i.e. all its elements must be non-negative) with the elements of each its row summed to one, such that $\mathbf{WV} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of order l, i.e. $\mathbf{W}$ is a *left-inverse matrix* for $\mathbf{V}$. It is known that for each collector matrix there is at least one distributor matrix, in particular, the matrix obtained by transposing $\mathbf{V}$ and subsequent normalizing its rows, to guarantee that the elements of each row of the transposed matrix are summed to one. We now present the formal definitions.

The elements $\mathcal{V}_{ir}$ $(1 \leq i \leq n,\ 1 \leq r \leq l)$ of the *collector matrix* $\mathbf{V}$ for the largest step stochastic autobisimulation $\mathcal{R}_{ss}(G)$ on G are defined as

$$\mathcal{V}_{ir} = \begin{cases} 1, & s_i \in \mathcal{K}_r; \\ 0, & \text{otherwise.} \end{cases}$$

Thus, all the elements of $\mathbf{V}$ are non-negative, as required. The row elements of $\mathbf{V}$ are summed to one, since for each s_i $(1 \leq i \leq n)$ there exists exactly one $\mathcal{K}_r$ $(1 \leq r \leq l)$ such that $s_i \in \mathcal{K}_r$. Hence,

$$\mathbf{V}\mathbf{1}^T = \mathbf{1}^T,$$

where $\mathbf{1}$ on the left side is the row vector of l values 1 while $\mathbf{1}$ on the right side is the row vector of n values 1.

The *distributor matrix* $\mathbf{W}$ for the collector matrix $\mathbf{V}$ is defined as

$$\mathbf{W} = (Diag(\mathbf{V}^T\mathbf{1}^T))^{-1}\mathbf{V}^T,$$

where $\mathbf{1}$ is the row vector of n values 1. One can check that $\mathbf{WV} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of order l.

The elements $(\mathcal{PV})_{is}$ $(1 \leq i \leq n,\ 1 \leq s \leq l)$ of the matrix $\mathbf{PV}$ are

$$(\mathcal{PV})_{is} = \sum_{j=1}^{n} \mathcal{P}_{ij}\mathcal{V}_{js} = \sum_{\{j \mid 1 \leq j \leq n,\ s_j \in \mathcal{K}_s\}} PM(s_i, s_j) = PM(s_i, \mathcal{K}_s).$$

As we know, for each s_i $(1 \leq i \leq n)$ there exists exactly one $\mathcal{K}_r$ $(1 \leq r \leq l)$ such that $s_i \in \mathcal{K}_r$. For all $s_i \in \mathcal{K}_r$ we have $PM(\mathcal{K}_r, \mathcal{K}_s) = PM(s_i, \mathcal{K}_s)$ $(1 \leq i \leq n,\ 1 \leq r, s \leq l)$. Then the elements $(\mathcal{VP}_{\underline{\leftrightarrow}_{ss}})_{is}$ $(1 \leq i \leq n,\ 1 \leq s \leq l)$ of the matrix $\mathbf{VP}_{\underline{\leftrightarrow}_{ss}}$ are

$$(\mathcal{VP}_{\underline{\leftrightarrow}_{ss}})_{is} = \sum_{r=1}^{l} \mathcal{V}_{ir}(\mathcal{P}_{\underline{\leftrightarrow}_{ss}})_{rs} = \sum_{\{r \mid 1 \leq r \leq l,\ s_i \in \mathcal{K}_r\}} PM(\mathcal{K}_r, \mathcal{K}_s) = PM(s_i, \mathcal{K}_s).$$

Therefore, we have

$$\mathbf{PV} = \mathbf{VP}_{\underline{\leftrightarrow}_{ss}},\ \mathbf{WPV} = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}.$$

Example 7.11 *Let F be from Example 6.1. The TPMs for $DTMC(\overline{F})$ and $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ are*

$$\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \theta & 0 & 0 & 1-\theta \end{pmatrix},\ \mathbf{P}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \theta & 0 & 1-\theta \end{pmatrix}.$$

The collector matrix $\mathbf{V}$ for $\mathcal{R}_{ss}(\overline{F})$ and the distributor matrix $\mathbf{W}$ for $\mathbf{V}$ are

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ \mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Then it is easy to check that

$$\mathbf{WPV} = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}.$$

In Figure 44, the relation (depicted by arrow) between $RDTMC$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}$ is obtained using the transition function $PM^{\diamond}(\mathcal{K}, \widetilde{\mathcal{K}})$, based on $PM^{\diamond}(s, \tilde{s})$. Let G be a dynamic expression. We shall prove that the TPM $\mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}$, (forwardly) constructed by quotienting (by $\underline{\leftrightarrow}_{ss}$) $DTMC(G)$, followed by reduction (eliminating vanishing states) of $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, *coincides with* the TPM $(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}}$, (reversely) constructed by reduction of $DTMC(G)$, followed by quotienting $RDTMC(G)$. The following proposition relates those quotient reduced TPM $(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}}$ and reduced quotient TPM $\mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}$.

Proposition 7.5 *Let G be a dynamic expression, $\mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}$ be the TPM for $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}}$ results from quotienting (by $\underline{\leftrightarrow}_{ss}$) the TPM $\mathbf{P}^{\diamond}$ for $RDTMC(G)$. Then*

$$(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}.$$

Proof. See Appendix A.5. □

Thus, the quotienting and reduction are permutable for DTMCs of the process expressions. This may simplify the performance evaluation when eliminating vanishing states makes the subsequent quotienting more efficient. The reverse construction (reduction first) is particularly preferable in case of small equivalence classes of vanishing states when quotienting does not merge many of them before eliminating.

Example 7.12 *Let F be from Example 6.1. The reordered TPMs for $DTMC(\overline{F})$ and $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ are*

$$\mathbf{P}_r = \begin{pmatrix} 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1-\rho & \rho & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \theta & 1-\theta & 0 \\ 0 & 0 & \theta & 0 & 1-\theta \end{pmatrix}, \ \mathbf{P}_{r\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1-\rho & \rho & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \theta & 1-\theta \end{pmatrix}.$$

The reordered collector matrix $\mathbf{V}_r$ for $\mathcal{R}_{ss}(\overline{F})$ and the reordered distributor matrix $\mathbf{W}_r$ for $\mathbf{V}_r$ are

$$\mathbf{V}_r = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \ \mathbf{W}_r = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Then it is easy to check that

$$\mathbf{W}_r\mathbf{P}_r\mathbf{V}_r = \mathbf{P}_{r\underline{\leftrightarrow}_{ss}}.$$

Example 7.13 *Let F be from Example 6.1. The TPMs for $RDTMC(\overline{F})$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ are*

$$\mathbf{P}^{\diamond} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 1-\theta & 0 \\ 0 & \theta & 0 & 1-\theta \end{pmatrix}, \ \mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} 1-\rho & \rho & 0 \\ 0 & 0 & 1 \\ 0 & \theta & 1-\theta \end{pmatrix}.$$

The result of the decomposing the reordered collector matrix $\mathbf{V}_r$ for $\mathcal{R}_{ss}(\overline{F})$ and the reordered distributor matrix $\mathbf{W}_r$ for $\mathbf{V}_r$ are the matrices

$$\mathbf{V}_C = 1, \ \mathbf{V}_F = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{W}_C = 1, \ \mathbf{W}_F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Then it is easy to check that

$$(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}} = \mathbf{W}_F\mathbf{P}^{\diamond}\mathbf{V}_F = \mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}.$$

In [74], the ordinary, exact and strict lumpability relations on finite DTMCs are explored. It is investigated which properties of transient and stationary behaviour of DTMCs are preserved by aggregation with respect to the three mentioned kinds of lumping and their approximate "nearly" versions. It is proved that irreducibility is preserved by aggregation with respect to any partition (or equivalence relation) on the states of DTMCs. Since only finite irreducible DTMCs are considered (with a finite number of states), these all are positive recurrent. Aggregation can only decrease the number of states, hence, the aggregated DTMCs are also finite and positive recurrence is preserved by every aggregation. It is known [260, 265, 186, 49, 285, 188, 261, 263] that irreducible and positive recurrent DTMCs have a single stationary PMF. Note that the original and aggregated DTMCs may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that the original DTMC is aperiodic while the aggregated DTMC is periodic due to merging some states of the former. Thus, both finite irreducible DTMCs and their arbitrary aggregates have a single stationary PMF. Then the relationship between stationary probabilities of DTMCs and their aggregates with respect to ordinary, exact and strict lumpability is established in [74]. In particular, it is shown that for every DTMC aggregated by ordinary lumpability, the stationary probability of each aggregate state is a sum of the stationary probabilities of all its constituent states from the original DTMC. The information about individual stationary probabilities of the original DTMC is lost after such a summation, but in many cases, the stationary probabilities of the aggregated DTMC are enough to calculate performance measures of the high-level model, from which the original DTMC is extracted. As mentioned in [74], in some practical applications, the aggregated DTMC can be extracted directly from the high-level model. Thus, the aggregation techniques based on lumping are of practical importance, since they allow one to reduce the state space of the modeled systems, hence, the computational costs for evaluating their performance.

Let G be a dynamic expression. By definition of $\underline{\leftrightarrow}_{ss}$, the relation $\mathcal{R}_{ss}(G)$ on $TS(G)$ induces ordinary lumping on $SMC(G)$, i.e. if the states of $TS(G)$ are related by $\mathcal{R}_{ss}(G)$ then the same states in $SMC(G)$ are related by ordinary lumping. The quotient (maximal aggregate) of $SMC(G)$ by such an induced ordinary lumping is $SMC_{\underline{\leftrightarrow}_{ss}}(G)$. Since we consider only finite SMCs, irreducibility of $SMC(G)$ will imply irreducibility of $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ and they both are positive recurrent. Then a unique quotient stationary PMF of $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ can be calculated from a unique original stationary PMF of $SMC(G)$ by summing some elements of the latter, as described in [74]. Similar arguments demonstrate that the same results hold for $DTMC(G)$ and $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, as well as for $RDTMC(G)$ and $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$.

8 Stationary behaviour

Let us examine how the proposed equivalences can be used to compare the behaviour of stochastic processes in their steady states. We shall consider only formulas specifying stochastic processes with infinite behavior, i.e. expressions with the iteration operator. Note that the iteration operator does not guarantee infiniteness of behaviour, since there can exist a deadlock (blocking) within the body (the second argument) of iteration when the corresponding subprocess does not reach its final state by some reasons. In particular, if the body of iteration contains the Stop expression then the iteration will be "broken". On the other hand, the iteration body can be left after a finite number of its repeated executions and then the iteration termination is started. To avoid executing any activities after the iteration body, we take Stop as the termination argument of iteration.

Like in the framework of SMCs, in LDTSDPNs the most common systems for performance analysis are *ergodic* (irreducible, positive recurrent and aperiodic) ones. For ergodic LDTSDPNs, the steady-state marking probabilities exist and can be determined. In [226, 228], the following sufficient (but not necessary) conditions for ergodicity of DTSPNs are stated: *liveness* (for each transition and any reachable marking there exists a sequence of markings from it leading to the marking enabling that transition), *boundedness* (for any reachable marking the number of tokens in every place is not greater than some fixed number) and *nondeterminism* (the transition probabilities are strictly less than 1). However, it has been shown in [22] that even live, safe and nondeterministic DTSPNs (as well as live and safe CTSPNs and GSPNs) may be non-ergodic.

In this section, we consider only the process expressions such that their underlying SMCs contain exactly one closed communication class of states, and this class should also be ergodic to ensure uniqueness of the stationary distribution, which is also the limiting one. The states not belonging to that class do not disturb the uniqueness, since the closed communication class is single, hence, they all are transient. Then, for each transient state, the steady-state probability to be in it is zero while the steady-state probability to enter into the ergodic class starting from that state is equal to one.

8.1 Steady state, residence time and equivalences

The following proposition demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the steady-state probabilities to enter into an equivalence class coincide. One can also interpret the result stating that the mean

recurrence time for an equivalence class is the same for both expressions.

Proposition 8.1 *Let G, G' be dynamic expressions with $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$ and φ be the steady-state PMF for $SMC(G)$, φ' be the steady-state PMF for $SMC(G')$. Then $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$*

$$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s').$$

Proof. See Appendix A.6. □

Let G be a dynamic expression and φ be the steady-state PMF for $SMC(G)$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ be the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$. By Proposition 8.1 (modified for $\mathcal{R}_{\mathcal{L}ss}(G)$), we have $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$

$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = \sum_{s \in \mathcal{K}} \varphi(s).$$

Thus, for every equivalence class $\mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$, the value of $\varphi_{\underline{\leftrightarrow}_{ss}}$ corresponding to $\mathcal{K}$ is the sum of all values of φ corresponding to the states from $\mathcal{K}$.

Let $\mathbf{V}$ be the collector matrix for $\mathcal{R}_{ss}(G)$. One can see that

$$\varphi \mathbf{V} = \varphi_{\underline{\leftrightarrow}_{ss}}.$$

Hence, using $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ instead of $SMC(G)$ may simplify the analytical solution, since we may have less states, but constructing the TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, denoted by $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$, also requires some efforts, including determining $\mathcal{R}_{ss}(G)$ and calculating the probabilities to move from one equivalence class to other. The behaviour of $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ may stabilize quicker than that of $EDTMC(G)$ (if each of them has a single steady state), since $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$ is generally denser matrix than $\mathbf{P}^*$ (the TPM for $EDTMC(G)$) due to the fact that the former matrix is usually smaller and the transitions between the equivalence classes "include" all the transitions between the states belonging to these equivalence classes.

By Proposition 8.1, $\underline{\leftrightarrow}_{ss}$ preserves the quantitative properties of the stationary behaviour (the level of SMCs). We now intend to demonstrate that the qualitative properties of the stationary behaviour based on the multiaction labels are preserved as well (the level of transition systems).

Definition 8.1 *A* derived step trace *of a dynamic expression G is a chain $\Sigma = A_1 \cdots A_n \in (\mathbb{N}^{\mathcal{L}}_{fin})^*$, where $\exists s \in DR(G)$ $s \xrightarrow{\Upsilon_1} s_1 \xrightarrow{\Upsilon_2} \cdots \xrightarrow{\Upsilon_n} s_n$, $\mathcal{L}(\Upsilon_i) = A_i$ $(1 \leq i \leq n)$. The* probability to execute the derived step trace Σ in s *is*

$$PT(\Sigma, s) = \sum_{\{\Upsilon_1, \ldots, \Upsilon_n \mid s = s_0 \xrightarrow{\Upsilon_1} s_1 \xrightarrow{\Upsilon_2} \cdots \xrightarrow{\Upsilon_n} s_n,\ \mathcal{L}(\Upsilon_i) = A_i\ (1 \leq i \leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}).$$

The following theorem demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the steady-state probabilities to enter into an equivalence class and start a derived step trace from it coincide.

Theorem 8.1 *Let G, G' be dynamic expressions with $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$ and φ be the steady-state PMF for $SMC(G)$, φ' be the steady-state PMF for $SMC(G')$ and Σ be a derived step trace of G and G'. Then $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$*

$$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) PT(\Sigma, s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') PT(\Sigma, s').$$

Proof. See Appendix A.7. □

Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ be the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ and Σ be a derived step trace of G. By Theorem 8.1 (modified for $\mathcal{R}_{\mathcal{L}ss}(G)$), we have $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$

$$\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) PT(\Sigma, \mathcal{K}) = \sum_{s \in \mathcal{K}} \varphi(s) PT(\Sigma, s),$$

where $\forall s \in \mathcal{K}$ $PT(\Sigma, \mathcal{K}) = PT(\Sigma, s)$.

Let $DR(G) = \{s_1, \ldots, s_n\}$ and $DR(G)/_{\mathcal{R}_{ss}(G)} = \{\mathcal{K}_1, \ldots, \mathcal{K}_l\}$ while $\mathbf{V}$ be the collector matrix for $\mathcal{R}_{ss}(G)$ and $\mathbf{W}$ be the distributor matrix for $\mathbf{V}$. We denote $PT(\Sigma) = (PT(\Sigma, s_1), \ldots, PT(\Sigma, s_n))$ and $PT_{\underline{\leftrightarrow}_{ss}}(\Sigma) = (PT(\Sigma, \mathcal{K}_1), \ldots, PT(\Sigma, \mathcal{K}_l))$. One can see that $Diag(PT(\Sigma))\mathbf{V} = \mathbf{V} Diag(PT_{\underline{\leftrightarrow}_{ss}}(\Sigma))$ and $\mathbf{W} Diag(PT(\Sigma))\mathbf{V} = Diag(PT_{\underline{\leftrightarrow}_{ss}}(\Sigma))$. Then we have

$$\varphi Diag(PT(\Sigma))\mathbf{V} = \varphi \mathbf{V} Diag(PT_{\underline{\leftrightarrow}_{ss}}(\Sigma)) = \varphi_{\underline{\leftrightarrow}_{ss}} Diag(PT_{\underline{\leftrightarrow}_{ss}}(\Sigma)).$$

We now present a result that does not concern the steady-state probabilities, but it reveals two very important properties of residence time in the equivalence classes. The following proposition demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the sojourn time averages in an equivalence class coincide, as well as the sojourn time variances in it.

Proposition 8.2 *Let G, G' be dynamic expressions with $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. Then $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$*

$$SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G')),$$
$$VAR_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = VAR_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G')).$$

Proof. See Appendix A.8. □

Example 8.1 *Let*

$$E = [(\{a\}, \tfrac{1}{2}) * ((\{b\}, \tfrac{1}{2}); ((\{c\}, \tfrac{1}{3})_1 [] (\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}],$$
$$E' = [(\{a\}, \tfrac{1}{2}) * ((((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1) [] ((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}].$$

We have $\overline{E} \underline{\leftrightarrow}_{ss} \overline{E'}$.

$DR(\overline{E})$ consists of the equivalence classes

$$s_1 = [[\overline{(\{a\}, \tfrac{1}{2})} * ((\{b\}, \tfrac{1}{2}); ((\{c\}, \tfrac{1}{3})_1 [] (\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_2 = [[(\{a\}, \tfrac{1}{2}) * \overline{(\{b\}, \tfrac{1}{2})}; ((\{c\}, \tfrac{1}{3})_1 [] (\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_3 = [[(\{a\}, \tfrac{1}{2}) * ((\{b\}, \tfrac{1}{2}); \overline{((\{c\}, \tfrac{1}{3})_1 [] (\{c\}, \tfrac{1}{3})_2)}) * \mathsf{Stop}]]_{\approx}.$$

$DR(\overline{E'})$ consists of the equivalence classes

$$s'_1 = [[\overline{(\{a\}, \tfrac{1}{2})} * ((((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1) [] ((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s'_2 = [[(\{a\}, \tfrac{1}{2}) * \overline{((((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1) [] ((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2))} * \mathsf{Stop}]]_{\approx},$$
$$s'_3 = [[(\{a\}, \tfrac{1}{2}) * ((((\{b\}, \tfrac{1}{3})_1; \overline{(\{c\}, \tfrac{1}{2})_1}) [] ((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s'_4 = [[(\{a\}, \tfrac{1}{2}) * ((((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1) [] ((\{b\}, \tfrac{1}{3})_2; \overline{(\{c\}, \tfrac{1}{2})_2})) * \mathsf{Stop}]]_{\approx}.$$

The steady-state PMFs φ for $SMC(\overline{E})$ and φ' for $SMC(\overline{E'})$ are

$$\varphi = \left(0, \frac{1}{2}, \frac{1}{2}\right), \ \varphi' = \left(0, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right).$$

Consider the equivalence class (with respect to $\mathcal{R}_{ss}(\overline{E}, \overline{E'})$) $\mathcal{H} = \{s_3, s'_3, s'_4\}$. One can see that the steady-state probabilities for $\mathcal{H}$ coincide: $\sum_{s\in\mathcal{H}\cap DR(\overline{E})} \varphi(s) = \varphi(s_3) = \frac{1}{2} = \frac{1}{4} + \frac{1}{4} = \varphi'(s'_3) + \varphi'(s'_4) = \sum_{s'\in\mathcal{H}\cap DR(\overline{E'})} \varphi'(s')$.

Let $\Sigma = \{\{c\}\}$. The steady-state probabilities to enter into the equivalence class $\mathcal{H}$ and start the derived step trace Σ from it coincide as well: $\varphi(s_3)(PT(\{(\{c\}, \frac{1}{3})_1\}, s_3) + PT(\{(\{c\}, \frac{1}{3})_2\}, s_3)) = \frac{1}{2}\left(\frac{1}{4} + \frac{1}{4}\right) = \frac{1}{4} = \frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} = \varphi'(s'_3)PT(\{(\{c\}, \frac{1}{2})_1\}, s'_3) + \varphi'(s'_4)PT(\{(\{c\}, \frac{1}{2})_2\}, s'_4)$.

Further, the sojourn time averages in the equivalence class $\mathcal{H}$ coincide: $SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\{s_3\}) = \frac{1}{1-PM(\{s_3\},\{s_3\})} = \frac{1}{1-PM(s_3,s_3)} = \frac{1}{1-\frac{1}{2}} = 2 = \frac{1}{1-\frac{1}{2}} = \frac{1}{1-PM(s'_3,s'_3)} = \frac{1}{1-PM(s'_4,s'_4)} = \frac{1}{1-PM(\{s'_3,s'_4\},\{s'_3,s'_4\})} = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\{s'_3, s'_4\}) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\mathcal{H} \cap DR(G'))$.

Next, the sojourn time variances in the equivalence class $\mathcal{H}$ coincide: $VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\mathcal{H} \cap DR(G)) = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\{s_3\}) = \frac{PM(\{s_3\},\{s_3\})}{(1-PM(\{s_3\},\{s_3\}))^2} = \frac{PM(s_3,s_3)}{(1-PM(s_3,s_3))^2} = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = 2 = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = \frac{PM(s'_3,s'_3)}{(1-PM(s'_3,s'_3))^2} = \frac{PM(s'_4,s'_4)}{(1-PM(s'_4,s'_4))^2} = \frac{PM(\{s'_3,s'_4\},\{s'_3,s'_4\})}{(1-PM(\{s'_3,s'_4\},\{s'_3,s'_4\}))^2} = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\{s'_3, s'_4\}) = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\mathcal{H} \cap DR(G'))$.

In Figure 45, the marked dtsd-boxes corresponding to the dynamic expressions above are presented, i.e. $N = Box_{dtsd}(\overline{E})$ and $N' = Box_{dtsd}(\overline{E'})$.

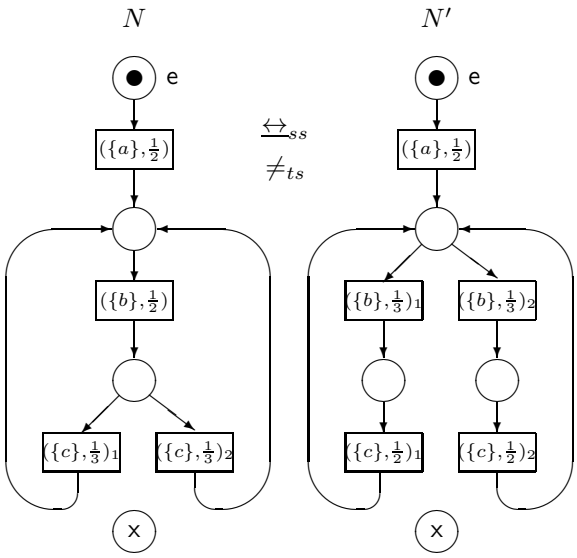

Figure 45: $\underline{\leftrightarrow}_{ss}$ preserves steady-state behaviour and sojourn time properties in the equivalence classes

Example 8.2 *Let F be from Example 6.1. Consider the equivalence class (with respect to $\mathcal{R}_{ss}(\overline{F})$) $\mathcal{K}_4 = \{s_4, s_5\}$. Then the value of $\varphi_{\underline{\leftrightarrow}_{ss}}$ corresponding to $\mathcal{K}_4$ is the sum of all values of φ corresponding to the states from $\mathcal{K}_4$:* $\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4) = \frac{1}{1+\theta} = \frac{l}{(l+m)(1+\theta)} + \frac{m}{(l+m)(1+\theta)} = \varphi(s_4) + \varphi(s_5) = \sum_{s\in\mathcal{K}_4}\varphi(s)$.

Let $\Sigma = \{\{d\}\}$. Then we have $\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4)PT(\Sigma,\mathcal{K}_4) = \frac{1}{1+\theta}\cdot\theta = \frac{\theta}{1+\theta} = \frac{l}{(l+m)(1+\theta)}\cdot\theta + \frac{m}{(l+m)(1+\theta)}\cdot\theta = \varphi(s_4)PT(\{(\{d\},\theta)_1\},s_4)+\varphi(s_5)PT(\{(\{d\},\theta)_2\},s_5) = \varphi(s_4)PT(\Sigma,s_4)+\varphi(s_5)PT(\Sigma,s_5) = \sum_{s\in\mathcal{K}_4}\varphi(s)PT(\Sigma,s)$, *where* $PT(\Sigma,\mathcal{K}_4) = PT(\Sigma,s_4) = PT(\Sigma,s_5) = \theta$.

The sojourn time average in $\mathcal{K}_4$ is $SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4) = \frac{1}{1-PM(\mathcal{K}_4,\mathcal{K}_4)} = \frac{1}{1-(1-\theta)} = \frac{1}{\theta} = \frac{1}{1-(1-\theta)} = \frac{1}{1-PM(s_4,s_4)} = \frac{1}{1-PM(s_5,s_5)} = \frac{1}{1-PM(\{s_4,s_5\},\{s_4,s_5\})} = SJ_{\underline{\leftrightarrow}_{ss}}(\{s_4,s_5\})$.

The sojourn time variance in $\mathcal{K}_4$ is $VAR_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}_4) = \frac{PM(\mathcal{K}_4,\mathcal{K}_4)}{(1-PM(\mathcal{K}_4,\mathcal{K}_4))^2} = \frac{1-\theta}{(1-(1-\theta))^2} = \frac{1-\theta}{\theta^2} = \frac{1-\theta}{(1-(1-\theta))^2} = \frac{PM(s_4,s_4)}{(1-PM(s_4,s_4))^2} = \frac{PM(s_5,s_5)}{(1-PM(s_5,s_5))^2} = \frac{PM(\{s_4,s_5\}}{(1-PM(\{s_4,s_5\},\{s_4,s_5\}))^2} = VAR_{\underline{\leftrightarrow}_{ss}}(\{s_4,s_5\})$.

8.2 Preservation of performance and simplification of its analysis

Many performance indices are based on the steady-state probabilities to enter into a set of similar states or, after coming in it, to start a derived step trace from this set. Some of the indices are calculated using the average or the variance of sojourn time in a set of similar states. The similarity of states is usually captured by an equivalence relation, hence, the sets are often the equivalence classes. Proposition 8.1, Theorem 8.1 and Proposition 8.2 guarantee coincidence of the mentioned indices for the expressions related by $\underline{\leftrightarrow}_{ss}$. Thus, $\underline{\leftrightarrow}_{ss}$ (hence, all the stronger equivalences we have considered) preserves performance of stochastic systems modeled by expressions of dtsdPBC.

In addition, it is easier to evaluate performance using an SMC with less states, since in this case the size of the transition probability matrix will be smaller, and we shall solve systems of less equations to calculate steady-state probabilities. The reasoning above validates the following method of performance analysis simplification.

1. The investigated system is specified by a static expression of dtsdPBC.
2. The transition system of the expression is constructed.
3. After treating the transition system for self-similarity, a step stochastic autobisimulation equivalence for the expression is determined.
4. The quotient underlying SMC is constructed from the quotient transition system.
5. Stationary probabilities and performance indices are calculated using the SMC.

The limitation of the method above is its applicability only to the expressions such that their underlying SMCs contain exactly one closed communication class of states, and this class should also be ergodic to ensure uniqueness of the stationary distribution. If an SMC contains several closed communication classes of states

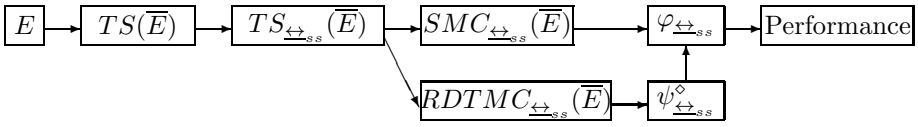

Figure 46: Equivalence-based simplification of performance evaluation

that are all ergodic then several stationary distributions may exist, which depend on the initial PMF. There is an analytical method to determine stationary probabilities for SMCs of this kind as well [186]. Note that the underlying SMC of every process expression has only one initial PMF (that at the time moment 0), hence, the stationary distribution will be unique in this case too. The general steady-state probabilities are then calculated as the sum of the stationary probabilities of all the ergodic classes of states, weighted by the probabilities to enter into these classes, starting from the initial state and passing through some transient states. In addition, it is worth applying the method only to the systems with similar subprocesses.

Before calculating stationary probabilities, we can further reduce the quotient underlying SMC, using an analogue of the deterministic barrier partitioning method described in [144] for semi-Markov processes (SMPs), which allows one to perform quicker the first passage-time analysis. Another option is the method of stochastic state classes proposed in [168, 169] for generalized SMPs (GSMPs) [67, 5] reduction, which allows one to simplify transient performance analysis (the analysis based on the transient probabilities of being in the states of GSMPs).

Alternatively, the results at the end of Section 7 allow us to simplify the steps 4 and 5 of the method above by constructing the reduced quotient DTMC (instead of the quotient underlying SMC) from the quotient transition system, followed by calculating the stationary probabilities of the quotient underlying SMC using that DTMC, and then obtaining the performance indices. In more detail, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{E})$ provides the information both about the probabilities to move between the equivalence classes of states $PM(\mathcal{K}, \widetilde{\mathcal{K}})$ and about the equivalence classes of vanishing states $DR_V(\overline{E})/_{\mathcal{R}_{ss}(\overline{E})}$. That information is used to construct the reordered quotient TPM $\mathbf{P}_{r\underline{\leftrightarrow}_{ss}}$, from which the TPM $\mathbf{P}^{\diamond}_{\underline{\leftrightarrow}_{ss}}$ for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$ is further obtained.

We first merge the equivalent states in transition systems and only then eliminate the vanishing states in Markov chains. The reason is that transition systems, being a higher-level formalism than Markov chains, describe both functional (qualitative) and performance (quantitative) aspects of behaviour while Markov chains represent only performance ones. Thus, eliminating vanishing states first would destroy the functional behaviour (which is respected by the equivalence used for quotienting), since the steps with different multiaction parts may lead to or start from different vanishing states.

Figure 46 presents the main stages of the standard and alternative equivalence-based simplification of performance evaluation described above.

9 Generalized shared memory system with maintenance

Let us consider a model of two processors accessing a common shared memory described in [214, 15, 16] in the continuous time setting on GSPNs. We shall analyze this shared memory system in the discrete time stochastic setting of dtsdPBC, where concurrent execution of activities is possible, while no two transitions of a GSPN may fire simultaneously (in parallel). We also add to the system a feature of the memory maintenance. Our *generalized* model parameterizes the *standard* shared memory system by treating the probabilities and weights from its specification as variables (parameters). The model behaves as follows. After activation of the system (turning the computer on), two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made, if the memory is available. When the decision is made in favour of a processor, it starts acquisition of the memory and the other processor should wait until the former one ends its memory operations, and the system returns to the state with both active processors and available common memory. If the memory is available and not required then its maintenance can be initiated, followed by the short memory service works (for example, the checksum test) during a fixed period of time, after which the memory becomes available again. If the memory requirement and its maintenance initiation happen at the same time then the service works start and no decision on the memory allocation is made while the memory is maintained. The diagram of the system is depicted in Figure 47.

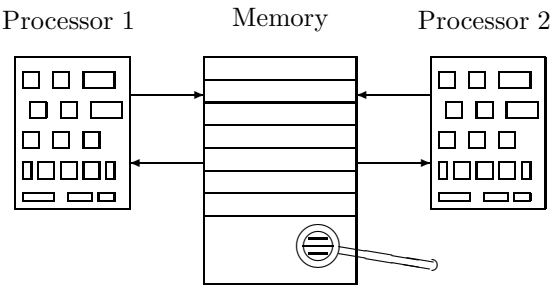

Figure 47: The diagram of the shared memory system with maintenance

9.1 The concrete system

The meaning of actions from the dtsdPBC expressions which will specify the system modules is as follows. The action a corresponds to the system activation. The action c specifies the memory maintenance initiation. The action e means the short memory service (taking a fixed time of 1 unit). The actions r_i $(1 \leq i \leq 2)$ represent the common memory request (whose probability is 10 times greater than that of the maintenance initiation) of processor i. The actions d_i correspond to the (instantaneous) decision on the memory allocation in favour of the processor i. The actions m_i represent the common memory access of processor i. The other actions are used for communication purposes only via synchronization, and we abstract from them later using restriction. For $a_1, \ldots, a_n \in Act$ $(n \in \mathbb{N})$, we shall abbreviate $\mathsf{sy}\ a_1 \cdots \mathsf{sy}\ a_n\ \mathsf{rs}\ a_1 \cdots \mathsf{rs}\ a_n$ to $\mathsf{sr}\ (a_1, \ldots, a_n)$.

We take general values for all multiaction probabilities and weights in the specification. Let all stochastic multiactions have the same generalized probability $\rho \in (0;1)$ and all deterministic ones have the same generalized weight $l \in \mathbb{R}_{>0}$. The resulting specification K of the generalized shared memory system with maintenance is as follows.

The static expression of the first processor is

$$K_1 = [(\{x_1\}, \rho) * ((\{r_1\}, \rho); (\{d_1, y_1\}, \natural_l^0); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}].$$

The static expression of the second processor is

$$K_2 = [(\{x_2\}, \rho) * ((\{r_2\}, \rho); (\{d_2, y_2\}, \natural_l^0); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$K_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{c\}, \frac{\rho}{10}); (\{e\}, \natural_l^1))[]((\{\widehat{y_1}\}, \natural_l^0); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, \natural_l^0); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}].$$

The static expression of the generalized shared memory system with maintenance is

$$K = (K_1 \| K_2 \| K_3)\ \mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2).$$

Let us illustrate an effect of synchronization. As a result of the synchronization of immediate multiactions $(\{d_i, y_i\}, \natural_l^0)$ and $(\{\widehat{y_i}\}, \natural_l^0)$ we get $(\{d_i\}, \natural_{2l})$ $(1 \leq i \leq 2)$. The synchronization of stochastic multiactions $(\{m_i, z_i\}, \rho)$ and $(\{\widehat{z_i}\}, \rho)$ produces $(\{m_i\}, \rho^2)$ $(1 \leq i \leq 2)$. The result of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_1\}, \rho)$ is $(\{a, \widehat{x_2}\}, \rho^2)$, and that of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_2\}, \rho)$ is $(\{a, \widehat{x_1}\}, \rho^2)$. After applying synchronization to $(\{a, \widehat{x_2}\}, \rho^2)$ and $(\{x_2\}, \rho)$, as well as to $(\{a, \widehat{x_1}\}, \rho^2)$ and $(\{x_1\}, \rho)$, we get the same activity $(\{a\}, \rho^3)$.

$DR(\overline{K})$ consists of the equivalence classes

$$\tilde{s}_1 = [(\overline{[(\{x_1\}, \rho)} * ((\{r_1\}, \rho); (\{d_1, y_1\}, \natural_l^0); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$\overline{[(\{x_2\}, \rho)} * ((\{r_2\}, \rho); (\{d_2, y_2\}, \natural_l^0); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$\overline{[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)} * ((((\{c\}, \tfrac{\rho}{10}); (\{e\}, \natural_l^1))[]((\{\widehat{y_1}\}, \natural_l^0); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, \natural_l^0); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2)]_{\approx},$$

$$\tilde{s}_2 = [([(\{x_1\}, \rho) * \overline{((\{r_1\}, \rho)}; (\{d_1, y_1\}, \natural_l^0); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * \overline{(\{r_2\}, \rho)}; (\{d_2, y_2\}, \natural_l^0); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * \overline{((((\{c\}, \tfrac{\rho}{10}); (\{e\}, \natural_l^1))[]((\{\widehat{y_1}\}, \natural_l^0); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, \natural_l^0); (\{\widehat{z_2}\}, \rho)))} * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2)]_{\approx},$$

$$\tilde{s}_3 = [([(\{x_1\},\rho) * \overline{(\{r_1\},\rho)}; (\{d_1,y_1\},\natural_l^0); (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * \overline{(\{r_2\},\rho)}; (\{d_2,y_2\},\natural_l^0); (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); \overline{(\{e\},\natural_l^1)^1})[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_4 = [([(\{x_1\},\rho) * ((\{r_1\},\rho); \overline{(\{d_1,y_1\},\natural_l^0)}; (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * \overline{(\{r_2\},\rho)}; (\{d_2,y_2\},\natural_l^0); (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_5 = [([(\{x_1\},\rho) * (\overline{(\{r_1\},\rho)}; (\{d_1,y_1\},\natural_l^0); (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); \overline{(\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * \overline{((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho)))} * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_6 = [([(\{x_1\},\rho) * ((\{r_1\},\rho); \overline{(\{d_1,y_1\},\natural_l^0)}; (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * \overline{((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); \overline{(\{e\},\natural_l^1)^1})[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_7 = [([(\{x_1\},\rho) * ((\{r_1\},\rho); \overline{(\{d_1,y_1\},\natural_l^0)}; (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * \overline{((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); \overline{(\{e\},\natural_l^1)^1})[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_8 = [([(\{x_1\},\rho) * (\overline{(\{r_1\},\rho)}; (\{d_1,y_1\},\natural_l^0); (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * \overline{((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); \overline{(\{e\},\natural_l^1)^1})[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_9 = [([(\{x_1\},\rho) * ((\{r_1\},\rho); \overline{(\{d_1,y_1\},\natural_l^0)}; (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); \overline{(\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * \overline{((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho)))} * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_{10} = [([(\{x_1\},\rho) * ((\{r_1\},\rho); (\{d_1,y_1\},\natural_l^0); \overline{(\{m_1,z_1\},\rho)}) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0); (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); \overline{(\{\widehat{z_1}\},\rho)})[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_{11} = [([(\{x_1\},\rho) * (\overline{(\{r_1\},\rho)}; (\{d_1,y_1\},\natural_l^0); (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0); \overline{(\{m_2,z_2\},\rho)}) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); \overline{(\{\widehat{z_2}\},\rho)})) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_{12} = [([(\{x_1\},\rho) * ((\{r_1\},\rho); (\{d_1,y_1\},\natural_l^0); \overline{(\{m_1,z_1\},\rho)}) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); \overline{(\{d_2,y_2\},\natural_l^0)}; (\{m_2,z_2\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); \overline{(\{\widehat{z_1}\},\rho)})[]((\{\widehat{y_2}\},\natural_l^0); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

$$\tilde{s}_{13} = [([(\{x_1\},\rho) * ((\{r_1\},\rho); \overline{(\{d_1,y_1\},\natural_l^0)}; (\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho); (\{d_2,y_2\},\natural_l^0); \overline{(\{m_2,z_2\},\rho)}) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\tfrac{\rho}{10}); (\{e\},\natural_l^1))[]((\{\widehat{y_1}\},\natural_l^0); (\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},\natural_l^0); \overline{(\{\widehat{z_2}\},\rho)})) * \mathsf{Stop}])$$
$$\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},$$

We have $DR_{ST}(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_{10}, \tilde{s}_{11}, \tilde{s}_{12}, \tilde{s}_{13}\}$, $DR_{WT}(\overline{K}) = \{\tilde{s}_3, \tilde{s}_6, \tilde{s}_7, \tilde{s}_8\}$ and $DR_V(\overline{K}) = \{\tilde{s}_4, \tilde{s}_5, \tilde{s}_9\}$.

The states are interpreted as follows: $\tilde{s}_1$ is the initial state in which the system is not activated; $\tilde{s}_2$: the system is activated and the memory is not requested and its maintenance is not initiated; $\tilde{s}_3$: the memory maintenance is initiated; $\tilde{s}_4$: the memory is requested by the first processor; $\tilde{s}_5$: the memory is requested by

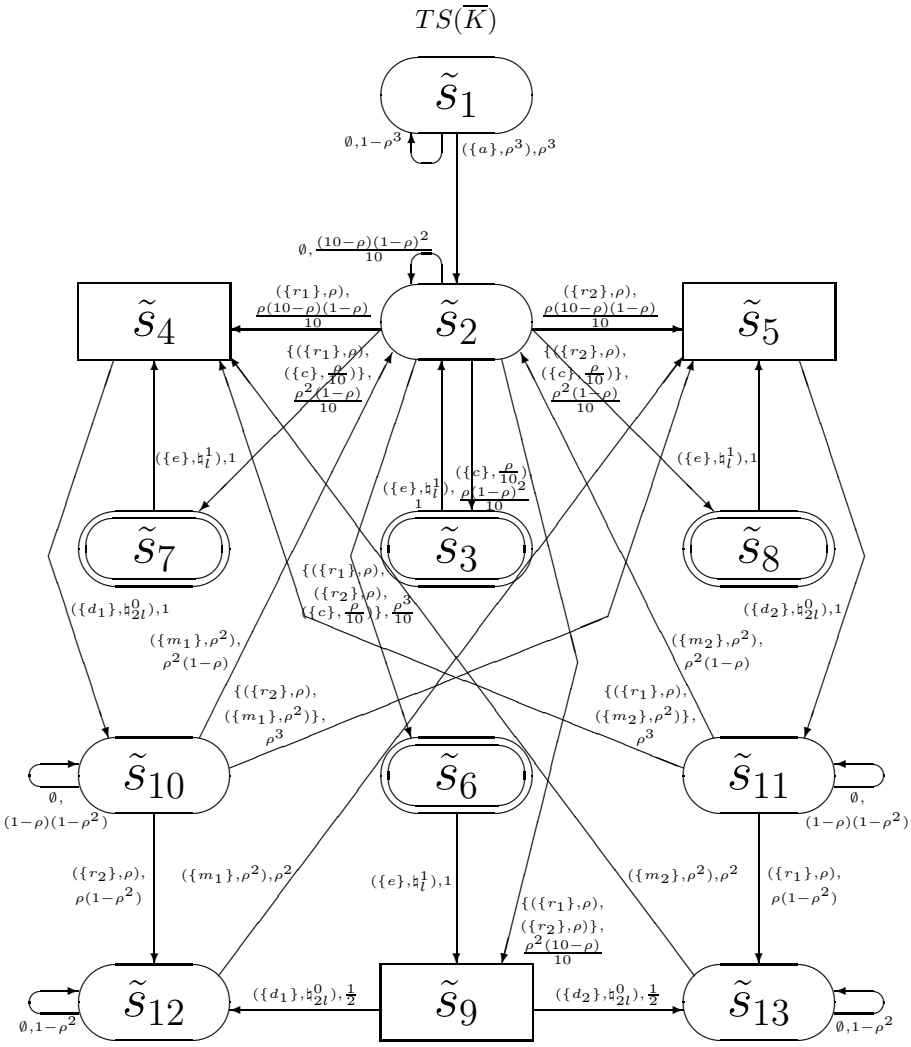

Figure 48: The transition system of the generalized shared memory system with maintenance

the second processor; $\tilde{s}_6$: the memory maintenance is initiated and the memory is requested by two processors; $\tilde{s}_7$: the memory maintenance is initiated and the memory is requested by the first processor; $\tilde{s}_8$: the memory maintenance is initiated and the memory is requested by the second processor; $\tilde{s}_9$: the memory is requested by two processors; $\tilde{s}_{10}$: the memory is allocated to the first processor; $\tilde{s}_{11}$: the memory is allocated to the second processor; $\tilde{s}_{12}$: the memory is allocated to the first processor and requested by the second processor; $\tilde{s}_{13}$: the memory is allocated to the second processor and requested by the first processor.

In Figure 48, the transition system $TS(\overline{K})$ is presented. In Figure 49, the underlying SMC $SMC(\overline{K})$ is depicted. Note that, in step semantics, we can execute the following activities in parallel: $(\{r_1\},\rho),(\{r_2\},\rho)$, as well as $(\{r_1\},\rho),(\{m_2\},\rho^2)$, and $(\{r_2\},\rho),(\{m_1\},\rho^2)$. We can also execute in parallel $(\{r_1\},\rho),(\{c\},\frac{\rho}{10})$, as well as $(\{r_2\},\rho),(\{c\},\frac{\rho}{10})$, and even $(\{r_1\},\rho),(\{r_2\},\rho),(\{c\},\frac{\rho}{10})$. The states $\tilde{s}_6,\tilde{s}_7,\tilde{s}_8,\tilde{s}_9$ only exist in step semantics, since they are reachable exclusively by executing all three activities $(\{r_1\},\rho),(\{r_2\},\rho),(\{c\},\frac{\rho}{10})$ or any pair of them in parallel.

The average sojourn time vector of $\overline{K}$ is

$$\widetilde{SJ} = \left(\frac{1}{\rho^3}, \frac{10}{\rho(21-12\rho+\rho^2)}, 1, 0, 0, 1, 1, 1, 0, \frac{1}{\rho(1+\rho-\rho^2)}, \frac{1}{\rho(1+\rho-\rho^2)}, \frac{1}{\rho^2}, \frac{1}{\rho^2}\right).$$

The sojourn time variance vector of $\overline{K}$ is

$$\widetilde{VAR} = \left(\tfrac{1-\rho^3}{\rho^6}, \tfrac{10(10-\rho)(1-\rho)^2}{\rho^2(21-12\rho+\rho^2)^2}, 0, 0, 0, 0, 0, 0, 0, \tfrac{(1-\rho^2)(1-\rho)}{\rho^2(1+\rho-\rho^2)^2}, \tfrac{(1-\rho^2)(1-\rho)}{\rho^2(1+\rho-\rho^2)^2}, \tfrac{1-\rho^2}{\rho^4}, \tfrac{1-\rho^2}{\rho^4}\right).$$

Let us denote $\chi = 21-12\rho+\rho^2$ and $\theta = 1+\rho-\rho^2$. The TPM for $EDTMC(\overline{K})$ is

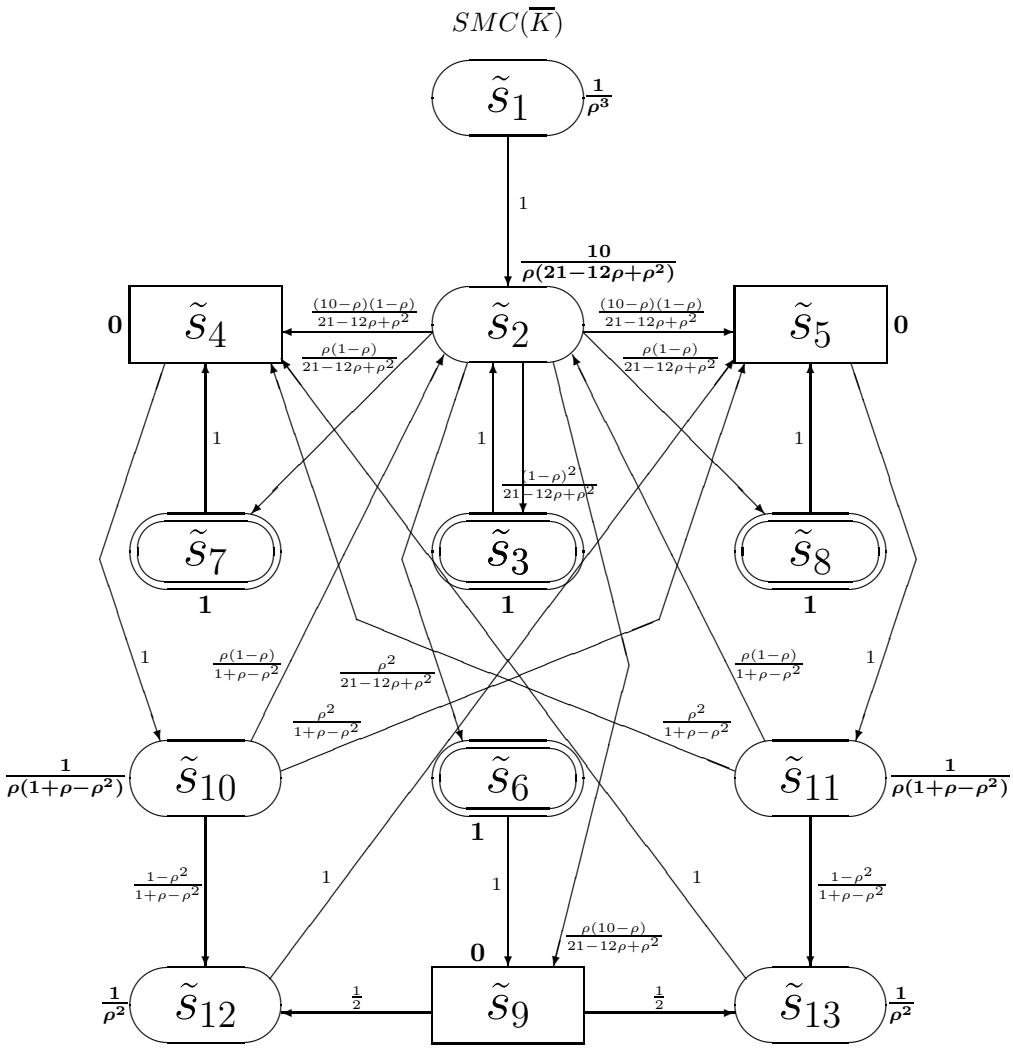

Figure 49: The underlying SMC of the generalized shared memory system with maintenance

$$\widetilde{\mathbf{P}}^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{(1-\rho)^2}{\chi} & \frac{(10-\rho)(1-\rho)}{\chi} & \frac{(10-\rho)(1-\rho)}{\chi} & \frac{\rho^2}{\chi} & \frac{\rho(1-\rho)}{\chi} & \frac{\rho(1-\rho)}{\chi} & \frac{\rho(10-\rho)}{\chi} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\rho(1-\rho)}{\theta} & 0 & 0 & \frac{\rho^2}{\theta} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-\rho^2}{\theta} & 0 \\ 0 & \frac{\rho(1-\rho)}{\theta} & 0 & \frac{\rho^2}{\theta} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1-\rho^2}{\theta} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The steady-state PMF for $EDTMC(\overline{K})$ is

$$\tilde{\psi}^* = \tfrac{1}{60+32\rho-94\rho^2+23\rho^3-\rho^4}(0, \rho(1-\rho)(21-12\rho+\rho^2), \rho(1-\rho)^3, 5(2-\rho)(1+\rho-\rho^2), 5(2-\rho)(1+\rho-\rho^2), \rho^3(1-\rho), \rho^2(1-\rho)^2, \rho^2(1-\rho)^2, 10\rho^2(1-\rho), 5(2-\rho)(1+\rho-\rho^2), 5(2-\rho)(1+\rho-\rho^2), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

The steady-state PMF $\tilde{\psi}^*$ weighted by $\widetilde{SJ}$ is

$$\tfrac{1}{\rho^2(60+32\rho-94\rho^2+23\rho^3-\rho^4)}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, 0, \rho^5(1-\rho), \rho^4(1-\rho)^2, \rho^4(1-\rho)^2, 0, 5\rho(2-\rho), 5\rho(2-\rho), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

It remains to normalize the steady-state weighted PMF by dividing it by the sum of its components

$$\tilde{\psi}^* \widetilde{SJ}^T = \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{\rho^2(60+32\rho-94\rho^2+23\rho^3-\rho^4)}.$$

Thus, the steady-state PMF for $SMC(\overline{K})$ is

$$\tilde{\varphi} = \tfrac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, 0, \rho^5(1-\rho), \rho^4(1-\rho)^2, \rho^4(1-\rho)^2, 0, 5\rho(2-\rho), 5\rho(2-\rho), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

Otherwise, from $TS(\overline{K})$, we can construct the DTMC of $\overline{K}$, $DTMC(\overline{K})$, and then calculate $\tilde{\varphi}$ using it.
In Figure 50, the DTMC $DTMC(\overline{K})$ is depicted.
The TPM for $DTMC(\overline{K})$ is

$$\widetilde{\mathbf{P}} = \begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{(10-\rho)(1-\rho)^2}{10} & \frac{\rho(1-\rho)^2}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{10} & \frac{\rho^2(1-\rho)}{10} & \frac{\rho^2(10-\rho)}{10} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \rho^2(1-\rho) & 0 & 0 & \rho^3 & 0 & 0 & 0 & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) & 0 \\ 0 & \rho^2(1-\rho) & 0 & \rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\ 0 & 0 & 0 & 0 & \rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 & 0 \\ 0 & 0 & 0 & \rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

The steady-state PMF for $DTMC(\overline{K})$ is

$$\tilde{\psi} = \tfrac{1}{20+10\rho+10\rho^2+\rho^3-21\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 5\rho^2(2-\rho)(1+\rho-\rho^2), 5\rho^2(2-\rho)(1+\rho-\rho^2), \rho^5(1-\rho), \rho^4(1-\rho)^2, 10\rho^4(1-\rho), 5\rho(2-\rho), 5\rho(2-\rho), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

Remember that $DR_T(\overline{K}) = DR_{ST}(\overline{K}) \cup DR_{WT}(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_6, \tilde{s}_7, \tilde{s}_8, \tilde{s}_{10}, \tilde{s}_{11}, \tilde{s}_{12}, \tilde{s}_{13}\}$ and $DR_V(\overline{K}) = \{\tilde{s}_4, \tilde{s}_5, \tilde{s}_9\}$. Hence,

$$\textstyle\sum_{\tilde{s} \in DR_T(\overline{K})} \tilde{\psi}(\tilde{s}) = \tilde{\psi}(\tilde{s}_1) + \tilde{\psi}(\tilde{s}_2) + \tilde{\psi}(\tilde{s}_3) + \tilde{\psi}(\tilde{s}_6) + \tilde{\psi}(\tilde{s}_7) + \tilde{\psi}(\tilde{s}_8) + \tilde{\psi}(\tilde{s}_{10}) + \tilde{\psi}(\tilde{s}_{11}) + \tilde{\psi}(\tilde{s}_{12}) + \tilde{\psi}(\tilde{s}_{13}) = \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{20+10\rho+10\rho^2+\rho^3-21\rho^4}.$$

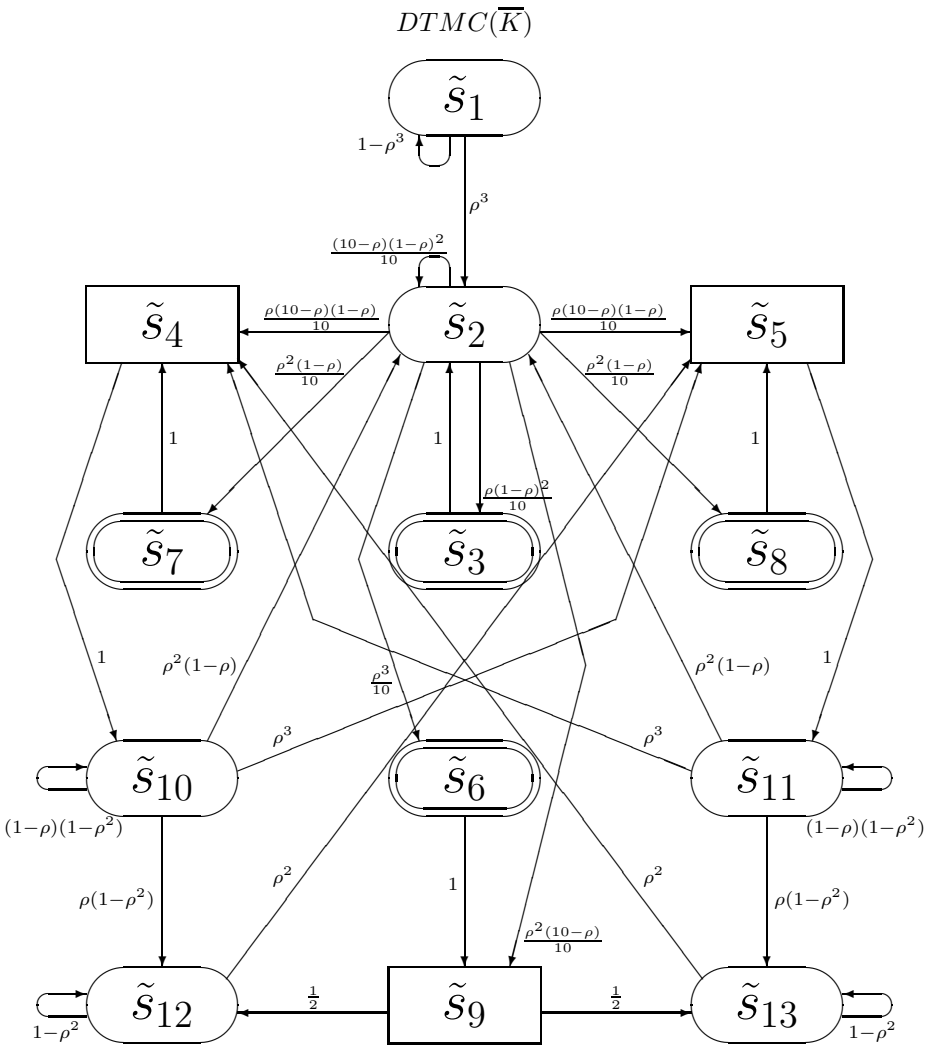

Figure 50: The DTMC of the generalized shared memory system with maintenance

By Proposition 5.1, we have

$$\begin{aligned}
\tilde{\varphi}(\tilde{s}_1) &= 0 \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = 0,\\
\tilde{\varphi}(\tilde{s}_2) &= \tfrac{10\rho^2(1-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_3) &= \tfrac{\rho^3(1-\rho)^3}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_4) &= 0,\\
\tilde{\varphi}(\tilde{s}_5) &= 0,\\
\tilde{\varphi}(\tilde{s}_6) &= \tfrac{\rho^5(1-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_7) &= \tfrac{\rho^4(1-\rho)^2}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_8) &= \tfrac{\rho^4(1-\rho)^2}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_9) &= 0,\\
\tilde{\varphi}(\tilde{s}_{10}) &= \tfrac{5\rho(2-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_{11}) &= \tfrac{5\rho(2-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_{12}) &= \tfrac{5(1-\rho)(2+\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{5(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}(\tilde{s}_{13}) &= \tfrac{5(1-\rho)(2+\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{5(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}.
\end{aligned}$$

Thus, the steady-state PMF for $SMC(\overline{K})$ is

$$\tilde{\varphi} = \tfrac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, 0, \rho^5(1-\rho), \rho^4(1-\rho)^2, \rho^4(1-\rho)^2, 0, 5\rho(2-\rho),\\ 5\rho(2-\rho), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

This coincides with the result obtained with the use of $\tilde{\psi}^*$ and $\widetilde{SJ}$.

Alternatively, from $TS(\overline{K})$, we can construct the reduced DTMC of $\overline{K}$, $RDTMC(\overline{K})$, and then calculate $\tilde{\varphi}$ using it.

Remember that $DR_{ST}(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_{10}, \tilde{s}_{11}, \tilde{s}_{12}, \tilde{s}_{13}\}$, $DR_{WT}(\overline{K}) = \{\tilde{s}_3, \tilde{s}_6, \tilde{s}_7, \tilde{s}_8\}$, $DR_V(\overline{K}) = \{\tilde{s}_4, \tilde{s}_5, \tilde{s}_9\}$. We reorder the elements of $DR(\overline{K})$, by moving vanishing states to the first positions and s-tangible states to the last positions: $\tilde{s}_4, \tilde{s}_5, \tilde{s}_9, \tilde{s}_3, \tilde{s}_6, \tilde{s}_7, \tilde{s}_8, \tilde{s}_1, \tilde{s}_2, \tilde{s}_{10}, \tilde{s}_{11}, \tilde{s}_{12}, \tilde{s}_{13}$.

The reordered TPM for $DTMC(\overline{K})$ is

$$\widetilde{\mathbf{P}}_r = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0\\
\frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho^2(10-\rho)}{10} & \frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{10} & \frac{\rho^2(1-\rho)}{10} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & 0 & 0 & 0 & 0\\
0 & \rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) & 0\\
\rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2)\\
0 & \rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 & 0\\
\rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2
\end{pmatrix}.$$

The result of the decomposing $\widetilde{\mathbf{P}}_r$ are the matrices

$$\widetilde{\mathbf{C}} = \begin{pmatrix} 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0 \end{pmatrix},\ \widetilde{\mathbf{D}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix},\ \widetilde{\mathbf{E}} = \begin{pmatrix}
0 & 0 & 0\\
0 & 0 & 1\\
1 & 0 & 0\\
0 & 1 & 0\\
0 & 0 & 0\\
\frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho^2(10-\rho)}{10}\\
0 & \rho^3 & 0\\
\rho^3 & 0 & 0\\
0 & \rho^2 & 0\\
\rho^2 & 0 & 0
\end{pmatrix},$$

$$\widetilde{\mathbf{F}} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0\\
\frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{10} & \frac{\rho^2(1-\rho)}{10} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) & 0\\
0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2)\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2
\end{pmatrix}.$$

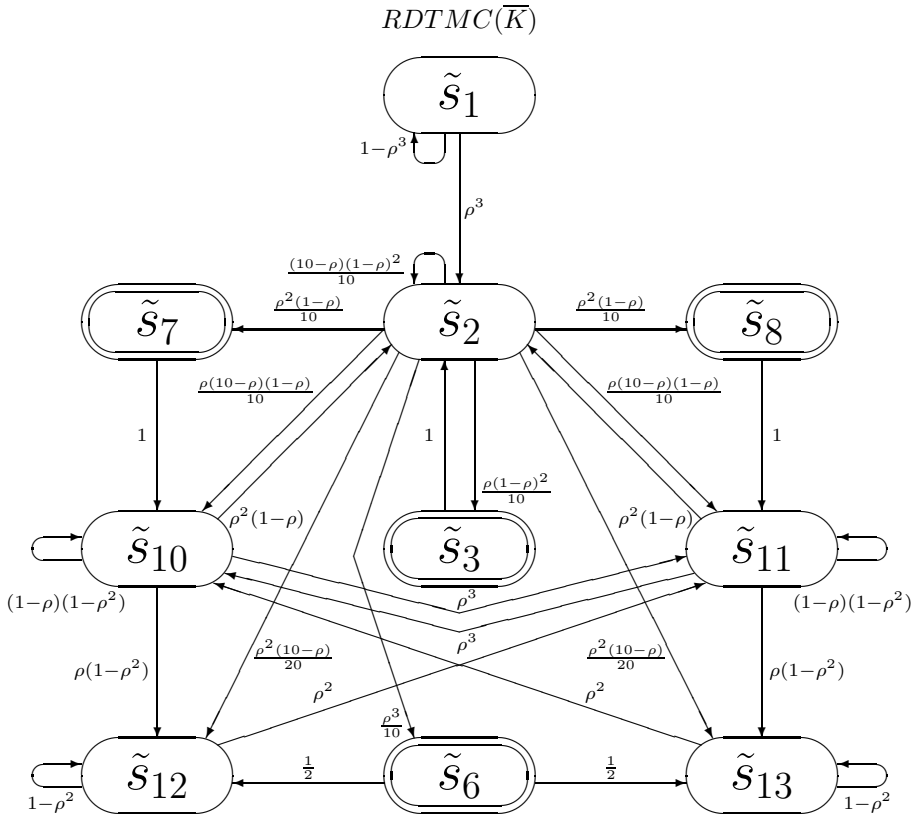

Figure 51: The reduced DTMC of the generalized shared memory system with maintenance

Since $\widetilde{\mathbf{C}}^1 = \mathbf{0}$, we have $\forall k > 0,\ \widetilde{\mathbf{C}}^k = \mathbf{0}$, hence, $l = 0$ and there are no loops among vanishing states. Then

$$\widetilde{\mathbf{G}} = \sum_{k=0}^{l} \widetilde{\mathbf{C}}^k = \widetilde{\mathbf{C}}^0 = \mathbf{I}.$$

Further, the TPM for $RDTMC(\overline{K})$ is

$$\widetilde{\mathbf{P}}^{\diamond} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\widetilde{\mathbf{G}}\widetilde{\mathbf{D}} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\mathbf{I}\widetilde{\mathbf{D}} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\widetilde{\mathbf{D}} =$$

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 \\
\frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{10} & \frac{\rho^2(1-\rho)}{10} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho(10-\rho)(1-\rho)}{10} & \frac{\rho^2(10-\rho)}{20} & \frac{\rho^2(10-\rho)}{20} \\
0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & \rho^3 & \rho(1-\rho^2) & 0 \\
0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & \rho^3 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho^2 & 1-\rho^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \rho^2 & 0 & 0 & 1-\rho^2
\end{pmatrix}.$$

In Figure 51, the reduced DTMC $RDTMC(\overline{K})$ is presented.
Then the steady-state PMF for $RDTMC(\overline{K})$ is

$$\tilde{\psi}^{\diamond} = \tfrac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho^3(1-\rho)^3, \rho^5(1-\rho), \rho^4(1-\rho)^2, \rho^4(1-\rho)^2, 0, 10\rho^2(1-\rho), 5\rho(2-\rho), 5\rho(2-\rho),$$
$$5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

Note that $\tilde{\psi}^{\diamond} = (\tilde{\psi}^{\diamond}(\tilde{s}_3), \tilde{\psi}^{\diamond}(\tilde{s}_6), \tilde{\psi}^{\diamond}(\tilde{s}_7), \tilde{\psi}^{\diamond}(\tilde{s}_8), \tilde{\psi}^{\diamond}(\tilde{s}_1), \tilde{\psi}^{\diamond}(\tilde{s}_2), \tilde{\psi}^{\diamond}(\tilde{s}_{10}), \tilde{\psi}^{\diamond}(\tilde{s}_{11}), \tilde{\psi}^{\diamond}(\tilde{s}_{12}), \tilde{\psi}^{\diamond}(\tilde{s}_{13}))$. By Proposition 5.2, we have

$\tilde{\varphi}(\tilde{s}_1) = 0$, $\tilde{\varphi}(\tilde{s}_2) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}(\tilde{s}_3) = \frac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$,
$\tilde{\varphi}(\tilde{s}_4) = 0$, $\tilde{\varphi}(\tilde{s}_5) = 0$, $\tilde{\varphi}(\tilde{s}_6) = \frac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$,
$\tilde{\varphi}(\tilde{s}_7) = \frac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}(\tilde{s}_8) = \frac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}(\tilde{s}_9) = 0$,
$\tilde{\varphi}(\tilde{s}_{10}) = \frac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}(\tilde{s}_{11}) = \frac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}(\tilde{s}_{12}) = \frac{(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$,
$\tilde{\varphi}(\tilde{s}_{13}) = \frac{(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

Thus, the steady-state PMF for $SMC(\overline{K})$ is

$$\tilde{\varphi} = \tfrac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, 0, \rho^5(1-\rho), \rho^4(1-\rho)^2, \rho^4(1-\rho)^2, 0, 5\rho(2-\rho),$$
$$5\rho(2-\rho), 5(1-\rho)(2+\rho), 5(1-\rho)(2+\rho)).$$

This coincides with the result obtained with the use of $\tilde{\psi}^*$ and $\widetilde{SJ}$.
We can now calculate the main performance indices.

- The average recurrence time in the state $\tilde{s}_2$, where no processor requests the memory and its maintenance is not initiated, called the *average system run-through*, is $\frac{1}{\tilde{\varphi}_2} = \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{10\rho^2(1-\rho)}$.
- The system is not activated only in the state $\tilde{s}_1$. Then the steady-state probability that the system is activated is $1 - \tilde{\varphi}_1 = 1 - 0 = 1$. The common memory is only available in the states $\tilde{s}_2, \tilde{s}_4, \tilde{s}_5, \tilde{s}_9$. Then the steady-state probability that the memory is available is $\tilde{\varphi}_2 + \tilde{\varphi}_4 + \tilde{\varphi}_5 + \tilde{\varphi}_9 = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + 0 + 0 + 0 = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$. The common memory is maintained only in the states $\tilde{s}_3, \tilde{s}_6, \tilde{s}_7, \tilde{s}_8$. Then the steady-state probability that the memory is maintained is $\tilde{\varphi}_3 + \tilde{\varphi}_6 + \tilde{\varphi}_7 + \tilde{\varphi}_8 = \frac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + \frac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + \frac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + \frac{\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \frac{\rho^3(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$. Thus, the steady-state probability that the memory is used (i.e. neither available nor maintained), called the *shared memory utilization*, is $1 - \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} - \frac{\rho^3(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \frac{10(2+\rho-2\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.
- After activation of the system, we leave the state $\tilde{s}_1$ for ever, and the common memory is either requested or allocated or maintained in every remaining state, with exception of $\tilde{s}_2$. Thus, the *rate with which the necessity (also for maintenance) of shared memory emerges* coincides with the rate of leaving $\tilde{s}_2$, calculated as $\frac{\tilde{\varphi}_2}{SJ_2} = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} \cdot \frac{\rho(21-12\rho+\rho^2)}{10} = \frac{\rho^3(1-\rho)(21-12\rho+\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.
- The parallel common memory request of two processors $\{(\{r_1\}, \rho), (\{r_2\}, \rho)\}$ is only possible from the state $\tilde{s}_2$. In this state, the request probability is the sum of the execution probabilities for all multisets of activities containing both $(\{r_1\}, \rho)$ and $(\{r_2\}, \rho)$. The *steady-state probability of the shared memory request from two processors* is $\tilde{\varphi}_2 \sum_{\{\Upsilon | (\{(\{r_1\},\rho),(\{r_2\},\rho)\} \subseteq \Upsilon\}} PT(\Upsilon, \tilde{s}_2) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} \left(\frac{\rho^2(10-\rho)}{10} + \frac{\rho^3}{10}\right) = \frac{10\rho^4(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.
- The common memory request of the first processor $(\{r_1\}, \rho)$ is only possible from the states $\tilde{s}_2, \tilde{s}_{11}$. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r_1\}, \rho)$. The *steady-state probability of the shared memory request from the first processor* is $\tilde{\varphi}_2 \sum_{\{\Upsilon | (\{r_1\},\rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_2) + \tilde{\varphi}_{11} \sum_{\{\Upsilon | (\{r_1\},\rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_{11}) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} \left(\frac{\rho(10-\rho)(1-\rho)}{10} + \frac{\rho^2(1-\rho)}{10} + \frac{\rho^2(10-\rho)}{10} + \frac{\rho^3}{10}\right) + \frac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho(1-\rho^2)+\rho^3) = \frac{5\rho^2(2+\rho-2\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

In Figure 52, the marked dtsd-boxes corresponding to the dynamic expressions of the generalized two processors, shared memory and shared memory system with maintenance are presented, i.e. $N_i = Box_{dtsd}(\overline{K_i})$ $(1 \leq i \leq 3)$ and $N = Box_{dtsd}(\overline{K})$.

9.2 The abstract system and its reduction

Let us consider a modification of the generalized shared memory system with maintenance via abstraction from identifiers of the processors, i.e. such that the processors are indistinguishable. For example, we can just see that a processor requires memory or the memory is allocated to it but cannot observe which processor is it. We call this system the abstract generalized shared memory system with maintenance. To implement the abstraction, we replace the actions r_i, d_i, m_i $(1 \leq i \leq 2)$ in the system specification by r, d, m, respectively.

The static expression of the first processor is

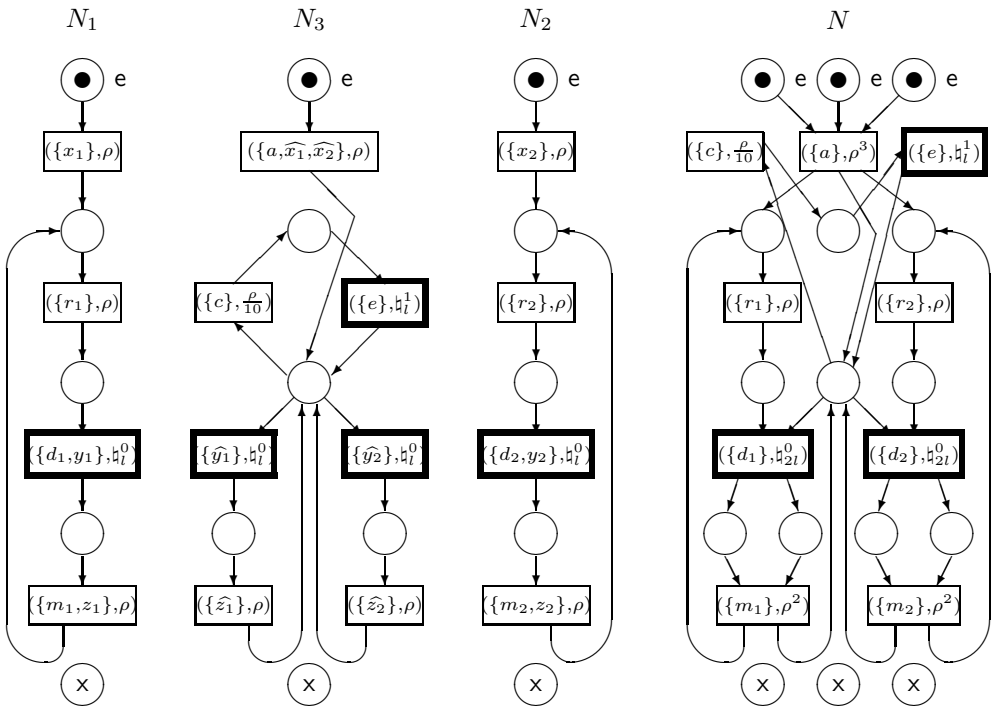

Figure 52: The marked dtsd-boxes of the generalized two processors, shared memory and shared memory system with maintenance

$$L_1 = [(\{x_1\},\rho) * ((\{r\},\rho);(\{d,y_1\},\natural_l^0);(\{m,z_1\},\rho)) * \mathsf{Stop}].$$

The static expression of the second processor is

$$L_2 = [(\{x_2\},\rho) * ((\{r\},\rho);(\{d,y_2\},\natural_l^0);(\{m,z_2\},\rho)) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$L_3 = [(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{c\},\frac{\rho}{10});(\{e\},\natural_l^1))[](((\{\widehat{y_1}\},\natural_l^0);(\{\widehat{z_1}\},\rho))[](((\{\widehat{y_2}\},\natural_l^0);(\{\widehat{z_2}\},\rho))) * \mathsf{Stop}].$$

The static expression of the abstract generalized shared memory system with maintenance is

$$L = (L_1 \| L_2 \| L_3)\ \mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2).$$

$DR(\overline{L}) = \{\tilde{s}'_1,\ldots,\tilde{s}'_{13}\}$ resembles $DR(\overline{K})$, and $TS(\overline{L})$ is similar to $TS(\overline{K})$. We have $SMC(\overline{L}) \simeq SMC(\overline{K})$. Thus, the average sojourn time vectors of $\overline{L}$ and $\overline{K}$, as well as the TPMs and the steady-state PMFs for $EDTMC(\overline{L})$ and $EDTMC(\overline{K})$, coincide.

The first, second, third and fourth performance indices are the same for the generalized system and its abstract modification. Let us consider the following performance index which is specific to the abstract system.

- The common memory request of a processor $(\{r\},\rho)$ is only possible from the states $\tilde{s}'_2,\tilde{s}'_{10},\tilde{s}'_{11}$. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r\},\rho)$. The *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}_2 \sum_{\{\Upsilon|(\{r\},\rho)\in\Upsilon\}} PT(\Upsilon,\tilde{s}'_2) + \tilde{\varphi}_{10} \sum_{\{\Upsilon|(\{r\},\rho)\in\Upsilon\}} PT(\Upsilon,\tilde{s}'_{10}) + \tilde{\varphi}_{11} \sum_{\{\Upsilon|(\{r\},\rho)\in\Upsilon\}} PT(\Upsilon,\tilde{s}'_{11}) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}\left(\frac{\rho(10-\rho)(1-\rho)}{10} + \frac{\rho(10-\rho)(1-\rho)}{10} + \frac{\rho^2(1-\rho)}{10} + \frac{\rho^2(1-\rho)}{10} + \frac{\rho^2(10-\rho)}{10} + \frac{\rho^3}{10}\right) + \frac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho(1-\rho^2)+\rho^3) + \frac{5\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho(1-\rho^2)+\rho^3) = \frac{10\rho^2(2-\rho)(1+\rho-\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}.$

We have $DR(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1,\widetilde{\mathcal{K}}_2,\widetilde{\mathcal{K}}_3,\widetilde{\mathcal{K}}_4,\widetilde{\mathcal{K}}_5,\widetilde{\mathcal{K}}_6,\widetilde{\mathcal{K}}_7,\widetilde{\mathcal{K}}_8,\widetilde{\mathcal{K}}_9\}$, where $\widetilde{\mathcal{K}}_1 = \{\tilde{s}'_1\}$ (the initial state in which the system is not activated), $\widetilde{\mathcal{K}}_2 = \{\tilde{s}'_2\}$ (the system is activated and the memory is not requested and its maintenance is not initiated), $\widetilde{\mathcal{K}}_3 = \{\tilde{s}'_3\}$ (the memory maintenance is initiated), $\widetilde{\mathcal{K}}_4 = \{\tilde{s}'_4,\tilde{s}'_5\}$ (the memory is requested by a processor), $\widetilde{\mathcal{K}}_5 = \{\tilde{s}'_6\}$ (the memory maintenance is initiated and the memory is requested

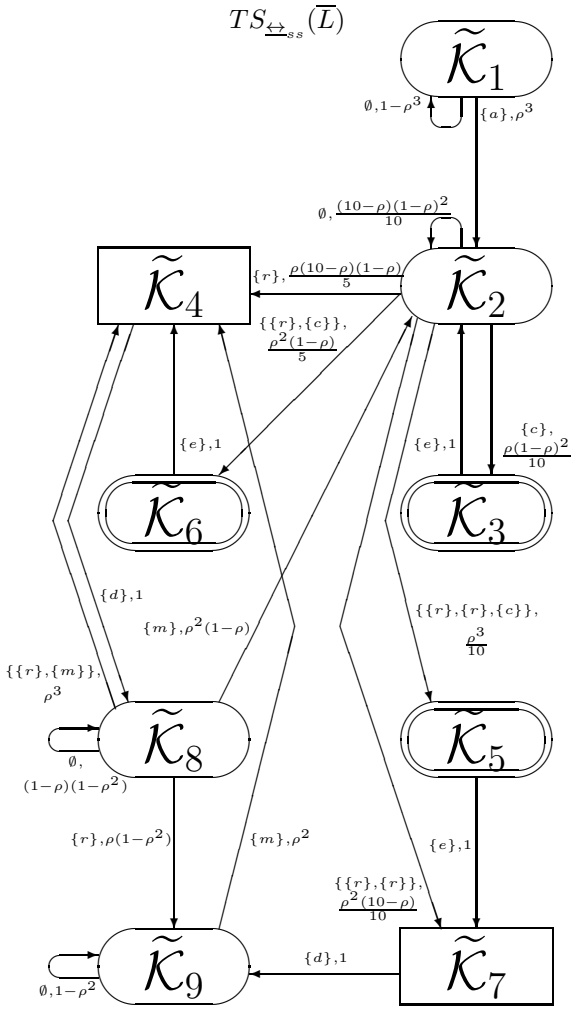

Figure 53: The quotient transition system of the abstract generalized shared memory system with maintenance

by two processors), $\widetilde{\mathcal{K}}_6 = \{\tilde{s}'_7, \tilde{s}'_8\}$ (the memory maintenance is initiated and the memory is requested by a processor), $\widetilde{\mathcal{K}}_7 = \{\tilde{s}'_9\}$ (the memory is requested by two processors), $\widetilde{\mathcal{K}}_8 = \{\tilde{s}'_{10}, \tilde{s}'_{11}\}$ (the memory is allocated to a processor), $\widetilde{\mathcal{K}}_9 = \{\tilde{s}'_{12}, \tilde{s}'_{13}\}$ (the memory is allocated to a processor and requested by another processor).

We have $DR_{ST}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_8, \widetilde{\mathcal{K}}_9\}$, $DR_{WT}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6\}$, $DR_V(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_7\}$.

In Figure 53, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is presented. In Figure 54, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is depicted. Note that, in step semantics, we may execute the following multiactions in parallel: $\{r\}, \{r\}$, as well as $\{r\}, \{m\}$. We can also execute in parallel $\{r\}, \{c\}$, and even $\{r\}, \{r\}, \{c\}$. The states $\widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6, \widetilde{\mathcal{K}}_7$ only exist in step semantics, since they are reachable exclusively by executing all three multiactions $\{r\}, \{r\}, \{c\}$ or any pair of them in parallel.

The quotient average sojourn time vector of $\overline{F}$ is

$$\widetilde{SJ}' = \left(\frac{1}{\rho^3}, \frac{10}{\rho(21-12\rho+\rho^2)}, 1, 0, 1, 1, 0, \frac{1}{\rho(1+\rho-\rho^2)}, \frac{1}{\rho^2}\right).$$

The quotient sojourn time variance vector of $\overline{F}$ is

$$\widetilde{VAR}' = \left(\frac{1-\rho^3}{\rho^6}, \frac{10(10-\rho)(1-\rho)^2}{\rho^2(21-12\rho+\rho^2)^2}, 0, 0, 0, 0, 0, \frac{(1-\rho^2)(1-\rho)}{\rho^2(1+\rho-\rho^2)^2}, \frac{1-\rho^2}{\rho^4}\right).$$

The TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

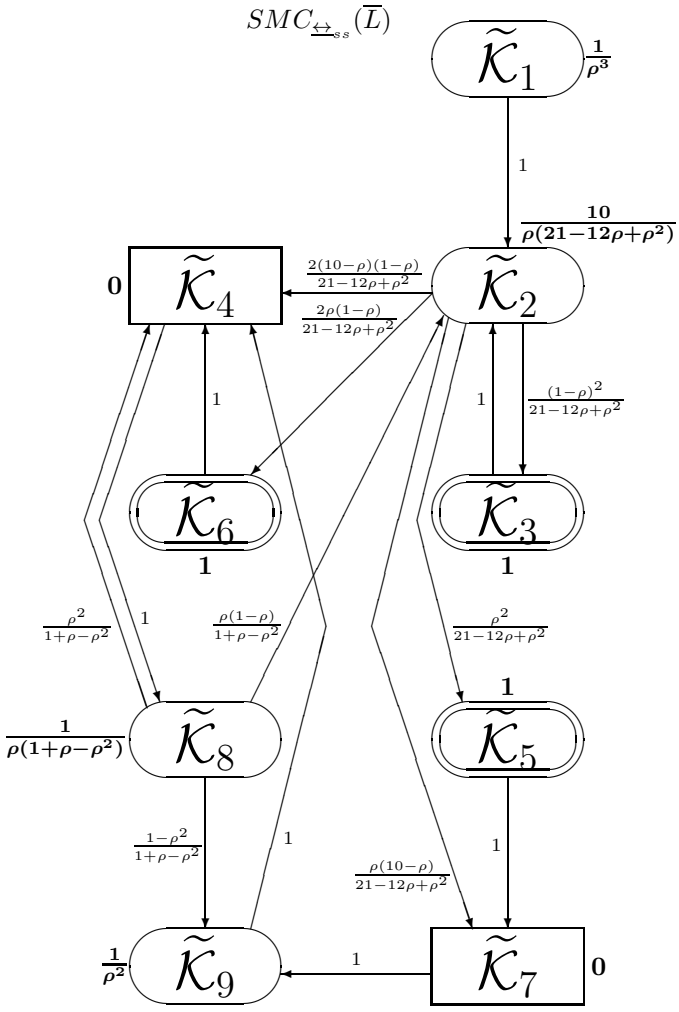

Figure 54: The quotient underlying SMC of the abstract generalized shared memory system with maintenance

$$\widetilde{\mathbf{P}}'^{*} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{(1-\rho)^2}{21-12\rho+\rho^2} & \frac{2(10-\rho)(1-\rho)}{21-12\rho+\rho^2} & \frac{\rho^2}{21-12\rho+\rho^2} & \frac{2\rho(1-\rho)}{21-12\rho+\rho^2} & \frac{\rho(10-\rho)}{21-12\rho+\rho^2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{\rho(1-\rho)}{1+\rho-\rho^2} & 0 & \frac{\rho^2}{1+\rho-\rho^2} & 0 & 0 & 0 & 0 & \frac{1-\rho^2}{1+\rho-\rho^2} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The steady-state PMF for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\psi}'^{*} = \tfrac{1}{60+32\rho-94\rho^2+23\rho^3-\rho^4}(0, \rho(1-\rho)(21-12\rho+\rho^2), \rho(1-\rho)^3, 10(2-\rho)(1+\rho-\rho^2), \rho^3(1-\rho), 2\rho^2(1-\rho)^2, \\ 10\rho^2(1-\rho), 10(2-\rho)(1+\rho-\rho^2), 10(1-\rho)(2+\rho)).$$

The steady-state PMF $\tilde{\psi}'^{*}$ weighted by $\widetilde{SJ}'$ is

$$\frac{1}{60+32\rho-94\rho^2+23\rho^3-\rho^4}(0, 10(1-\rho), \rho(1-\rho)^3, 0, \rho^3(1-\rho), 2\rho^2(1-\rho)^2, 0, 10(2-\rho), 10(1-\rho)(2+\rho)).$$

It remains to normalize the steady-state weighted PMF by dividing it by the sum of its components

$$\tilde{\psi}'^{*}\widetilde{SJ}'^{T} = \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{\rho^2(60+32\rho-94\rho^2+23\rho^3-\rho^4)}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\varphi}' = \frac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, \rho^5(1-\rho), 2\rho^4(1-\rho)^2, 0, 10\rho(2-\rho), 10(1-\rho)(2+\rho)).$$

Otherwise, from $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, we can construct the quotient DTMC of $\overline{L}$, $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, and then calculate $\tilde{\varphi}'$ using it.

In Figure 55, the quotient DTMC $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is depicted.

The TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\widetilde{\mathbf{P}}' = \begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{(10-\rho)(1-\rho)^2}{10} & \frac{\rho(1-\rho)^2}{10} & \frac{\rho(10-\rho)(1-\rho)}{5} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{5} & \frac{\rho^2(10-\rho)}{10} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \rho^2(1-\rho) & 0 & \rho^3 & 0 & 0 & 0 & (1-\rho)(1-\rho^2) & \rho(1-\rho^2) \\ 0 & 0 & 0 & \rho^2 & 0 & 0 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

The steady-state PMF for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\psi}' = \tfrac{1}{20+10\rho+10\rho^2+\rho^3-21\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 10\rho^2(2-\rho)(1+\rho-\rho^2), \rho^5(1-\rho), 2\rho^4(1-\rho)^2, \\ 10\rho^4(1-\rho), 10\rho(2-\rho), 10(1-\rho)(2+\rho)).$$

Remember that $DR_T(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = DR_{ST}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} \cup DR_{WT}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6, \widetilde{\mathcal{K}}_8, \widetilde{\mathcal{K}}_9\}$ and $DR_V(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_7\}$. Hence,

$$\textstyle\sum_{\widetilde{\mathcal{K}} \in DR_T(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})}} \tilde{\psi}'(\widetilde{\mathcal{K}}) = \tilde{\psi}'(\widetilde{\mathcal{K}}_1) + \tilde{\psi}'(\widetilde{\mathcal{K}}_2) + \tilde{\psi}'(\widetilde{\mathcal{K}}_3) + \tilde{\psi}'(\widetilde{\mathcal{K}}_5) + \tilde{\psi}'(\widetilde{\mathcal{K}}_6) + \tilde{\psi}'(\widetilde{\mathcal{K}}_8) + \tilde{\psi}'(\widetilde{\mathcal{K}}_9) = \\ \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{20+10\rho+10\rho^2+\rho^3-21\rho^4}.$$

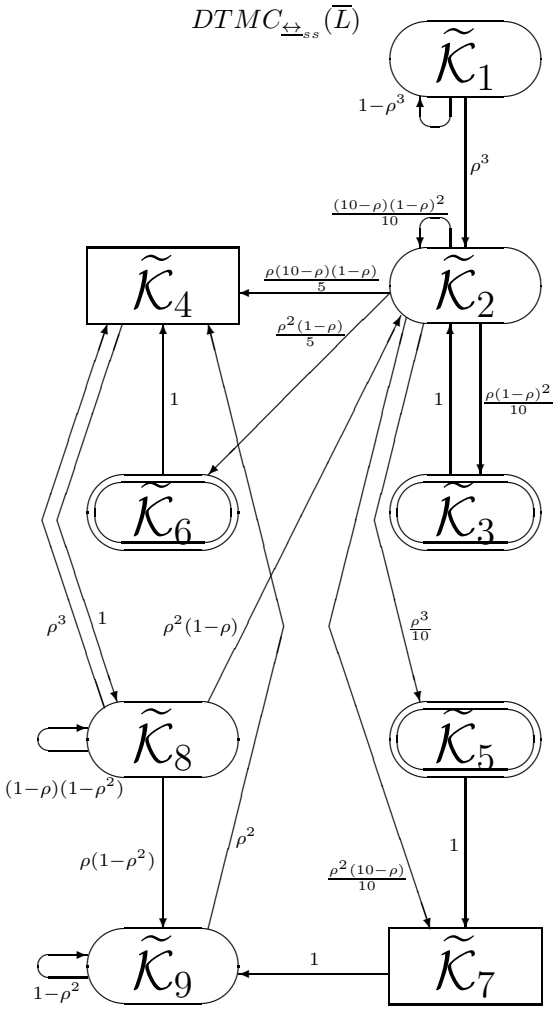

Figure 55: The quotient DTMC of the abstract generalized shared memory system with maintenance

By the "quotient" analogue of Proposition 5.1, we have

$$\begin{aligned}
\tilde{\varphi}'(\widetilde{\mathcal{K}}_1) &= 0 \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = 0,\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_2) &= \tfrac{10\rho^2(1-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_3) &= \tfrac{\rho^3(1-\rho)^3}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_4) &= 0,\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_5) &= \tfrac{\rho^5(1-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_6) &= \tfrac{2\rho^4(1-\rho)^2}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{2\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_7) &= 0,\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_8) &= \tfrac{10\rho(2-\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{10\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4},\\
\tilde{\varphi}'(\widetilde{\mathcal{K}}_9) &= \tfrac{10(1-\rho)(2+\rho)}{20+10\rho+10\rho^2+\rho^3-21\rho^4} \cdot \tfrac{20+10\rho+10\rho^2+\rho^3-21\rho^4}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \tfrac{10(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}.
\end{aligned}$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\varphi}' = \frac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, \rho^5(1-\rho), 2\rho^4(1-\rho)^2, 0, 10\rho(2-\rho), 10(1-\rho)(2+\rho)).$$

This coincides with the result obtained with the use of $\tilde{\psi}'^*$ and $\widetilde{SJ}'$.

Alternatively, from $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, we can construct the reduced quotient DTMC of $\overline{L}$, $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, and then calculate $\tilde{\varphi}'$ using it. By Proposition 7.5, it coincides with the quotient reduced DTMC of $\overline{L}$, the quotient of $RDTMC(\overline{L})$.

Remember that $DR_{ST}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_8, \widetilde{\mathcal{K}}_9\}$, $DR_{WT}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6\}$, $DR_V(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_7\}$. We reorder the elements of $DR(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})}$, by moving the equivalence classes of vanishing states to the first positions and those of s-tangible states to the last positions: $\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_7, \widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6, \widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_8, \widetilde{\mathcal{K}}_9$.

The reordered TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\widetilde{\mathbf{P}}'_r = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0\\
\frac{\rho(10-\rho)(1-\rho)}{5} & \frac{\rho^2(10-\rho)}{10} & \frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{5} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & 0 & 0\\
\rho^3 & 0 & 0 & 0 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & \rho(1-\rho^2)\\
\rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2
\end{pmatrix}.$$

The result of the decomposing $\widetilde{\mathbf{P}}'_r$ are the matrices

$$\widetilde{\mathbf{C}}' = \begin{pmatrix} 0 & 0\\ 0 & 0 \end{pmatrix},\ \widetilde{\mathbf{D}}' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},\ \widetilde{\mathbf{E}}' = \begin{pmatrix}
0 & 0\\
0 & 1\\
1 & 0\\
0 & 0\\
\frac{\rho(10-\rho)(1-\rho)}{5} & \frac{\rho^2(10-\rho)}{10}\\
\rho^3 & 0\\
\rho^2 & 0
\end{pmatrix},$$

$$\widetilde{\mathbf{F}}' = \begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0\\
\frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{5} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & 0 & 0\\
0 & 0 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & \rho(1-\rho^2)\\
0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2
\end{pmatrix}.$$

Since $\widetilde{\mathbf{C}}'^1 = \mathbf{0}$, we have $\forall k > 0,\ \widetilde{\mathbf{C}}'^k = \mathbf{0}$, hence, $l = 0$ and there are no loops among vanishing states. Then

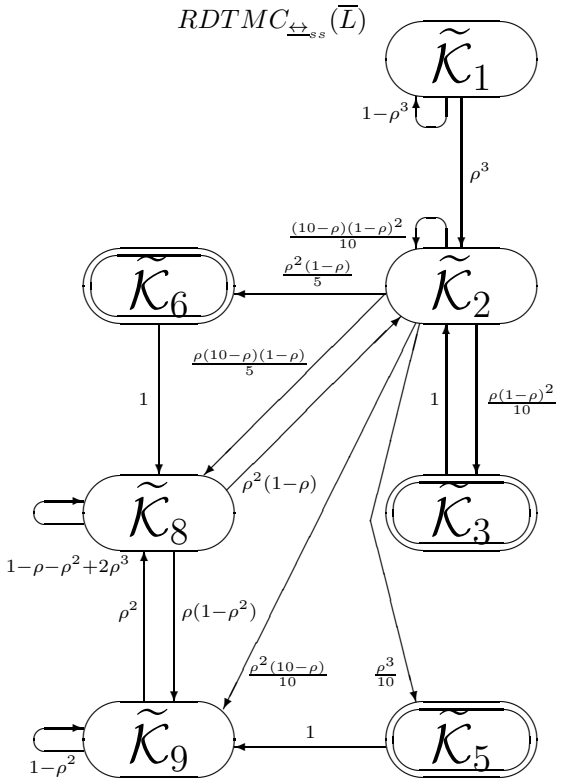

Figure 56: The reduced quotient DTMC of the abstract generalized shared memory system with maintenance

$$\widetilde{\mathbf{G}}' = \sum_{k=0}^{l} \widetilde{\mathbf{C}}'^{l} = \widetilde{\mathbf{C}}'^{0} = \mathbf{I}.$$

Further, the TPM for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\widetilde{\mathbf{P}}'^{\diamond} = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\widetilde{\mathbf{G}}'\widetilde{\mathbf{D}}' = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\mathbf{I}\widetilde{\mathbf{D}}' = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\widetilde{\mathbf{D}}' =$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0 \\ \frac{\rho(1-\rho)^2}{10} & \frac{\rho^3}{10} & \frac{\rho^2(1-\rho)}{5} & 0 & \frac{(10-\rho)(1-\rho)^2}{10} & \frac{\rho(10-\rho)(1-\rho)}{5} & \frac{\rho^2(10-\rho)}{10} \\ 0 & 0 & 0 & 0 & \rho^2(1-\rho) & 1-\rho-\rho^2+2\rho^3 & \rho(1-\rho^2) \\ 0 & 0 & 0 & 0 & 0 & \rho^2 & 1-\rho^2 \end{pmatrix}.$$

In Figure 56, the reduced quotient DTMC $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is presented.
Then the steady-state PMF for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\psi}'^{\diamond} = \frac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho^3(1-\rho)^3, \rho^5(1-\rho), 2\rho^4(1-\rho)^2, 0, 10\rho^2(1-\rho), 10\rho(2-\rho), 10(1-\rho)(2+\rho)).$$

Note that $\tilde{\psi}'^{\diamond} = (\tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_3), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_5), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_6), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_1), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_2), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_8), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_9))$. By the "quotient" analogue of Proposition 5.2, we have

$$\begin{array}{lll} \tilde{\varphi}'(\widetilde{\mathcal{K}}_1) = 0, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_2) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_3) = \frac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4}, \\ \tilde{\varphi}'(\widetilde{\mathcal{K}}_4) = 0, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_5) = \frac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_6) = \frac{2\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4}, \\ \tilde{\varphi}'(\widetilde{\mathcal{K}}_7) = 0, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_8) = \frac{10\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}, & \tilde{\varphi}'(\widetilde{\mathcal{K}}_9) = \frac{10(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}. \end{array}$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\tilde{\varphi}' = \frac{1}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(0, 10\rho^2(1-\rho), \rho^3(1-\rho)^3, 0, \rho^5(1-\rho), 2\rho^4(1-\rho)^2, 0, 10\rho(2-\rho), 10(1-\rho)(2+\rho)).$$

This coincides with the result obtained with the use of $\tilde{\psi}'^*$ and $\widetilde{SJ}'$.
We can now calculate the main performance indices.

- The average recurrence time in the state $\widetilde{\mathcal{K}}_2$, where no processor requests the memory and its maintenance is not initiated, called the *average system run-through*, is $\frac{1}{\tilde{\varphi}'_2} = \frac{20+10\rho-10\rho^2-9\rho^3-\rho^4}{10\rho^2(1-\rho)}$.

- The system is not activated only in the state $\widetilde{\mathcal{K}}_1$. Then the steady-state probability that the system is activated is $1 - \tilde{\varphi}'_1 = 1 - 0 = 1$. The common memory is only available in the states $\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_7$. Then the steady-state probability that the memory is available is $\tilde{\varphi}'_2 + \tilde{\varphi}'_4 + \tilde{\varphi}'_7 = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + 0 + 0 = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$. The common memory is maintained only in the states $\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6$. Then the steady-state probability that the memory is maintained is $\tilde{\varphi}'_3 + \tilde{\varphi}'_5 + \tilde{\varphi}'_6 = \frac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + \frac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} + \frac{2\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \frac{\rho^3(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$. Thus, the steady-state probability that the memory is used (i.e. neither available nor maintained), called the *shared memory utilization*, is $1 - \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} - \frac{\rho^3(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} = \frac{10(2+\rho-2\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

- After activation of the system, we leave the state $\widetilde{\mathcal{K}}_1$ for ever, and the common memory is either requested or allocated or maintained in every remaining state, with exception of $\widetilde{\mathcal{K}}_2$. Thus, the *rate with which the necessity (also for maintenance) of shared memory emerges* coincides with the rate of leaving $\widetilde{\mathcal{K}}_2$, calculated as $\frac{\tilde{\varphi}'_2}{\widetilde{SJ}'_2} = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} \cdot \frac{\rho(21-12\rho+\rho^2)}{10} = \frac{\rho^3(1-\rho)(21-12\rho+\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

- The parallel common memory request of two processors $\{\{r\},\{r\}\}$ is only possible from the state $\widetilde{\mathcal{K}}_2$. In this state, the request probability is the sum of the execution probabilities for all multisets of multiactions containing $\{r\}$ twice. The *steady-state probability of the shared memory request from two processors* is $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}}|\{\{r\},\{r\}\}\subseteq A,\ \widetilde{\mathcal{K}}_2 \stackrel{A}{\to} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}\left(\frac{\rho^2(10-\rho)}{10} + \frac{\rho^3}{10}\right) = \frac{10\rho^4(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

- The common memory request of a processor $\{r\}$ is only possible from the states $\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_8$. In each of the states, the request probability is the sum of the execution probabilities for all multisets of multiactions containing $\{r\}$. The *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_2 \stackrel{A}{\to} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) + \tilde{\varphi}'_8 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_8 \stackrel{A}{\to} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_8, \widetilde{\mathcal{K}}) = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4} \cdot \left(\frac{\rho(10-\rho)(1-\rho)}{5} + \frac{\rho^2(1-\rho)}{5} + \frac{\rho^2(10-\rho)}{10} + \frac{\rho^3}{10}\right) + \frac{10\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}(\rho(1-\rho^2)+\rho^3) = \frac{10\rho^2(2-\rho)(1+\rho-\rho^2)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$.

One can see that the performance indices are the same for the "complete" and the "quotient" abstract generalized shared memory systems with maintenance. The coincidence of the first, second and third performance indices obviously illustrates the results of Proposition 8.1 and Proposition 8.2 (both modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$). The coincidence of the fourth performance index is due to Theorem 8.1 (modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$): one should just apply its result to the derived step traces $\{\{r\},\{r\}\}$ and $\{\{r\},\{r\},\{c\}\}$ of the expression $\overline{L}$ and itself, and then sum the left and right parts of the two resulting equalities. The coincidence of the fifth performance index is due to Theorem 8.1 (modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$): one should just apply its result to the derived step traces $\{\{r\}\}$, $\{\{r\},\{c\}\}$, $\{\{r\},\{r\}\}$, $\{\{r\},\{r\},\{c\}\}$, $\{\{r\},\{m\}\}$ of the expression $\overline{L}$ and itself, and then sum the left and right parts of the five resulting equalities.

Let us consider what is the effect of quantitative changes of the parameter ρ upon performance of the "quotient" abstract generalized shared memory system with maintenance in its steady state. Remember that $\rho \in (0;1)$ is the probability of every stochastic multiaction in the specification of the system. The closer is ρ to 0, the less is the probability to execute some activities at every discrete time tick, hence, the system will most probably *stand idle*. The closer is ρ to 1, the greater is the probability to execute some activities at every discrete time tick, hence, the system will most probably *operate*.

Since $\tilde{\varphi}'_1 = \tilde{\varphi}'_4 = \tilde{\varphi}'_7 = 0$, only $\tilde{\varphi}'_2 = \frac{10\rho^2(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}'_3 = \frac{\rho^3(1-\rho)^3}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}'_5 = \frac{\rho^5(1-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}'_6 = \frac{2\rho^4(1-\rho)^2}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}'_8 = \frac{10\rho(2-\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$, $\tilde{\varphi}'_9 = \frac{10(1-\rho)(2+\rho)}{20+10\rho-10\rho^2-9\rho^3-\rho^4}$ depend on ρ. In Figure 57, the plots of $\tilde{\varphi}'_2$, $\tilde{\varphi}'_8$, $\tilde{\varphi}'_9$ (large probability masses) as functions of ρ are depicted. In Figure 58, the plots of $\tilde{\varphi}'_3$, $\tilde{\varphi}'_5$, $\tilde{\varphi}'_6$ (small probability masses) as functions of ρ are drawn. Notice that, however, we do not allow $\rho = 0$ or $\rho = 1$.

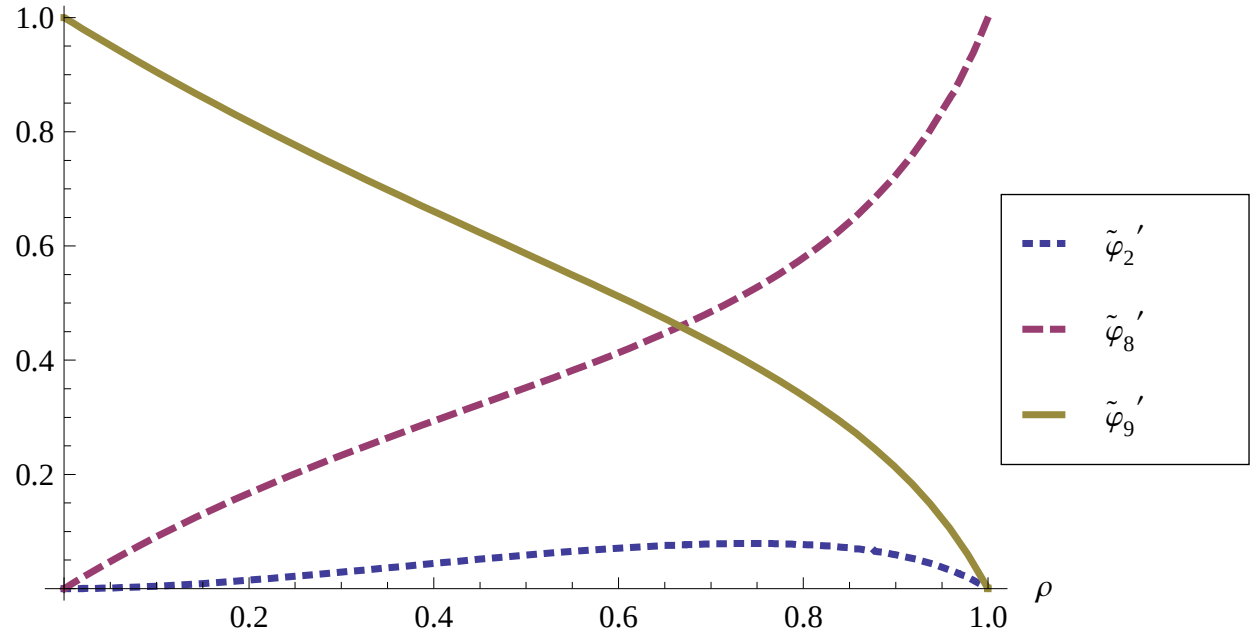

Figure 57: Steady-state probabilities $\tilde{\varphi}'_2,\ \tilde{\varphi}'_8,\ \tilde{\varphi}'_9$ (large probability masses) as functions of the parameter ρ

One can see that $\tilde{\varphi}'_2,\ \tilde{\varphi}'_3,\ \tilde{\varphi}'_5,\ \tilde{\varphi}'_6,\ \tilde{\varphi}'_8$ tend to 0 and $\tilde{\varphi}'_9$ tends to 1 when ρ approaches 0. Thus, when ρ is closer to 0, the probability that the memory is allocated to a processor and requested by another processor increases, hence, we have *more unsatisfied memory requests*.

Further, $\tilde{\varphi}'_2,\ \tilde{\varphi}'_3,\ \tilde{\varphi}'_5,\ \tilde{\varphi}'_6,\ \tilde{\varphi}'_9$ tend to 0 and $\tilde{\varphi}'_8$ tends to 1 when ρ approaches 1. Thus, when ρ is closer to 1, the probability that the memory is allocated to a processor (and not requested by another processor) increases, hence, we have *less unsatisfied memory requests*.

The maximal value 0.0792 of $\tilde{\varphi}'_2$ is reached when $\rho \approx 0.7427$. In this case, the probability that the system is activated and the memory is not requested and its maintenance is not initiated is maximal, i.e. the *maximal shared memory availability* is about 8%.

The maximal value 0.0007 of $\tilde{\varphi}'_3$ is reached when $\rho \approx 0.5158$. In this case, the probability that the memory maintenance is initiated is maximal, i.e. the *maximal shared memory maintenance necessity* is about 0.1%.

The maximal value 0.0044 of $\tilde{\varphi}'_5$ is reached when $\rho \approx 0.8724$. In this case, the probability that the memory maintenance is initiated and the memory is requested by two processors is maximal, i.e. the *maximal double (parallel) demand of shared memory during its maintenance* is about 0.4%.

The maximal value 0.0023 of $\tilde{\varphi}'_6$ is reached when $\rho \approx 0.7015$. In this case, the probability that the memory maintenance is initiated and the memory is requested by a (single) processor is maximal, i.e. the *maximal single demand of shared memory during its maintenance* is about 0.2%.

In Figure 59, the plot of the average system run-through, calculated as $\frac{1}{\tilde{\varphi}'_2}$, as a function of ρ is depicted. One can see that the run-through tends to ∞ when ρ approaches 0 or 1. Its minimal value 12.6259 is reached when $\rho \approx 0.7427$. To speed up operation of the system, one should take the parameter ρ closer to 0.7427.

The first curve in Figure 60 represents the shared memory utilization, calculated as $1 - \tilde{\varphi}'_{2-7}$, where $\tilde{\varphi}'_{2-7} = \tilde{\varphi}'_2 + \tilde{\varphi}'_3 + \tilde{\varphi}'_4 + \tilde{\varphi}'_5 + \tilde{\varphi}'_6 + \tilde{\varphi}'_7$, as a function of ρ. One can see that the utilization tends to 1 both when ρ approaches 0 and when ρ approaches 1. The minimal value 0.9149 of the utilization is reached when $\rho \approx 0.7494$. Thus, the *minimal shared memory utilization* is about 91%. To increase the utilization, one should take the parameter ρ closer to 0 or 1.

The second curve in Figure 60 represents the rate with which the necessity of shared memory emerges, calculated as $\frac{\tilde{\varphi}'_2}{\widetilde{SJ}_2}$, as a function of ρ. One can see that the rate tends to 0 both when ρ approaches 0 and when ρ approaches 1. The maximal value 0.0749 of the rate is reached when $\rho \approx 0.7723$. Thus, the *maximal rate with which the necessity of shared memory emerges* is about $\frac{1}{13}$. To decrease the mentioned rate, one should take the parameter ρ closer to 0 or 1.

The third curve in Figure 60 represents the steady-state probability of the shared memory request from two processors, calculated as $\tilde{\varphi}'_2\widetilde{\mathcal{S}}'_2$, where $\widetilde{\mathcal{S}}'_2 = \sum_{\{A,\widetilde{\mathcal{K}}|\{\{r\},\{r\}\}\subseteq A,\ \widetilde{\mathcal{K}}_2 \overset{A}{\to} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}})$, as function of ρ. One can see that the probability tends to 0 both when ρ approaches 0 and when ρ approaches 1. The maximal value 0.0514 of the probability is reached when $\rho \approx 0.8486$. To decrease the mentioned probability, one should take the parameter ρ closer to 0 or 1.

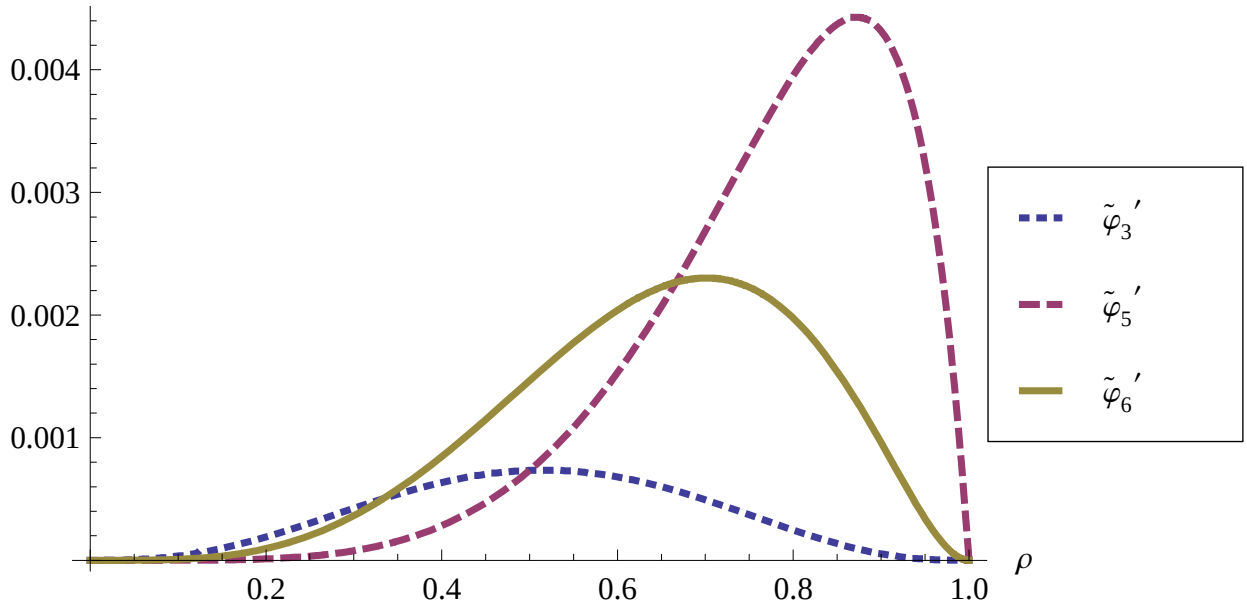

Figure 58: Steady-state probabilities $\tilde{\varphi}_3'$, $\tilde{\varphi}_5'$, $\tilde{\varphi}_6'$ (small probability masses) as functions of the parameter ρ

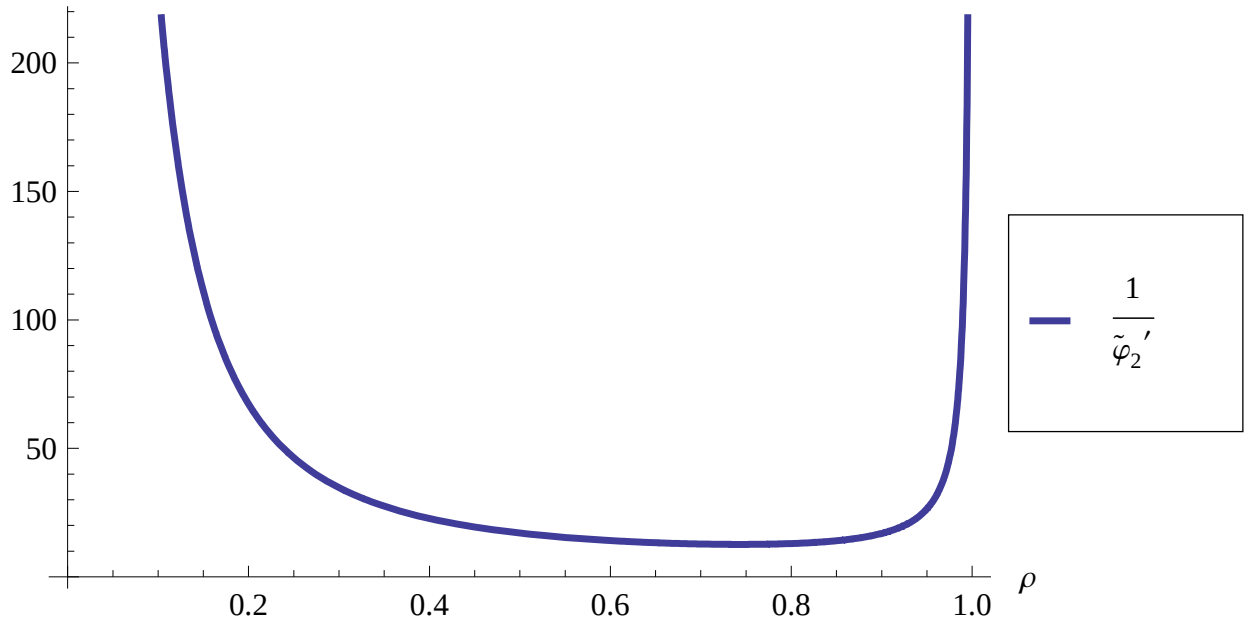

Figure 59: Average system run-through $\frac{1}{\tilde{\varphi}_2'}$ as a function of the parameter ρ

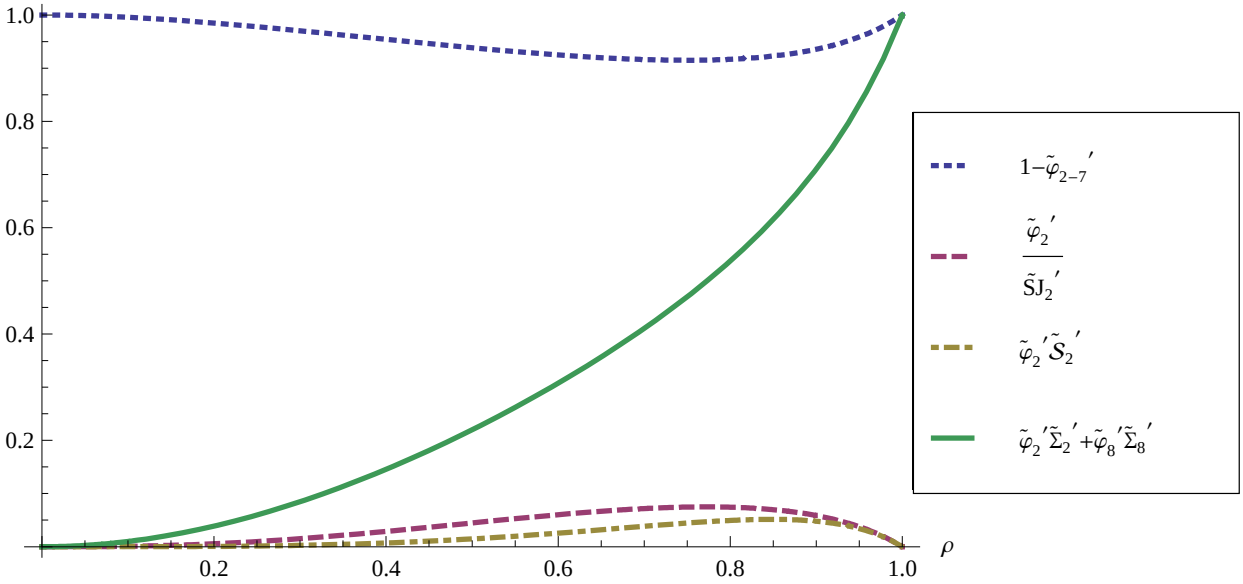

Figure 60: Some performance indices as functions of the parameter ρ

The fourth curve in Figure 60 represents the steady-state probability of the shared memory request from a processor, calculated as $\tilde{\varphi}'_2\widetilde{\Sigma}'_2 + \tilde{\varphi}'_8\widetilde{\Sigma}'_8$, where $\widetilde{\Sigma}'_i = \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_i \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_i, \widetilde{\mathcal{K}}),\ i \in \{2, 8\}$, as a function of ρ. One can see that the probability tends to 0 when ρ approaches 0 and it tends to 1 when ρ approaches 1. To increase the mentioned probability, one should take the parameter ρ closer to 1.

10 Related work

In this section, we consider in detail differences and similarities between dtsdPBC and other well-known or similar SPAs for the purpose of subsequent determining the specific advantages of dtsdPBC.

10.1 Continuous time and interleaving semantics

Let us compare dtsdPBC with classical SPAs: Markovian TImed Processes and Performability (Performance and dependability) evaluation (MTIPP) [162], Performance Evaluation Process Algebra (PEPA) [164] and Extended Markovian Process Algebra (EMPA) [36].

In MTIPP, every activity is a pair consisting of the action name (including the symbol τ for the *internal*, invisible action) and the parameter of exponential distribution of the action delay (the *rate*). The operations are *prefix*, *choice*, *parallel* composition including *synchronization* on the specified action set and *recursion*. It is possible to specify processes by recursive equations as well. The interleaving semantics is defined on the basis of Markovian (i.e. extended with the specification of rates) labeled transition systems. Note that we have the interleaving behaviour here because the exponential PDF is a continuous one, and a simultaneous execution of any two activities has zero probability according to the properties of continuous distributions. CTMCs can be derived from the mentioned transition systems to analyze performance.

In PEPA, activities are the pairs consisting of action types (including the *unknown*, unimportant type τ) and activity rates. The rate is either the parameter of exponential distribution of the activity duration or it is *unspecified*, denoted by $\top$. An activity with unspecified rate is *passive* by its action type. The set of operations includes *prefix*, *choice*, *cooperation*, *hiding* and constants whose meaning is given by the defining equations including the *recursive* ones. The cooperation is accomplished on the set of action types (the cooperation set) on which the components must *synchronize* or cooperate. If the cooperation set is empty, the cooperation operator turns into the *parallel* combinator. The semantics is interleaving, it is defined via the extension of labeled transition systems with a possibility to specify activity rates. Based on the transition systems, the continuous time Markov processes (CTMPs) are generated which are used for performance evaluation with the help of the embedded continuous time Markov chains (ECTMCs).

In EMPA, each action is a pair consisting of its type and rate. Actions can be *external* or *internal* (denoted by τ) according to types. There are three kinds of actions according to rates: *timed* ones with exponentially distributed durations (essentially, the actions from MTIPP and PEPA), *immediate* ones with priorities and weights (the actions analogous to immediate transitions of GSPNs) and *passive* ones (similar to passive actions

of PEPA). Timed actions specify activities that are relevant for performance analysis. Immediate actions model logical events and the activities that are irrelevant from the performance viewpoint or much faster than others. Passive actions model activities waiting for the synchronization with timed or immediate ones, and express nondeterministic choice. The set of operators consist of *prefix*, functional *abstraction*, functional *relabeling*, *alternative* composition and *parallel* composition ones. Parallel composition includes *synchronization* on the set of action types like in TCSP [167]. The syntax also includes *recursive* definitions given by means of constants. The semantics is interleaving and based on the labeled transition systems enriched with the information about action rates. For the exponentially timed kernel of the algebra (the sublanguage including only exponentially timed and passive actions), it is possible to construct CTMCs from the transition systems of the process terms to analyze the performance.

In dtsdPBC, every activity is a pair consisting of the multiaction (not just an action, as in the classical SPAs) as a first element. The second element is either the probability (not the rate, as in the classical SPAs) to execute the multiaction independently (the activity is called a stochastic multiaction in this case) or a combined specification of the (fixed) delay and weight expressing how important is the execution of this multiaction (the activity is called a deterministic multiaction in this case). Immediate (zero delay deterministic) multiactions in dtsdPBC are similar to immediate actions in EMPA, but all the immediate multiactions in dtsdPBC have the same (implicit) priority 2. The purpose is to execute them always before waiting (positive delay deterministic) multiactions with the same (implicit) priority 1, and stochastic multiactions with the same (implicit) priority 0. The immediate actions in EMPA can have different priority levels. Associating the same priority with all immediate (or waiting) multiactions in dtsdPBC results in the simplified specification and analysis, and such a decision is also appropriate to the calculus. The reason is that, as mentioned in [153], weights (assigned also to immediate actions in EMPA) are enough to denote preferences among immediate multiactions (designating their advantages or prescribing sub-priorities to them) and to produce the conformable probabilistic behaviours when one has to make a choice among several immediate multiactions executable in some state. There are no deterministic actions in MTIPP and PEPA. Immediate actions are only available in immediate PEPA (iPEPA) [157], where they are analogous to immediate multiactions in dtsdPBC, and in a variant of TIPP [140] discussed while constructing the calculus Probabilistic Markovian TIPP (PM-TIPP) in [257, 258], but there immediate activities are used just to specify probabilistic branching and they cannot be synchronized.

dtsdPBC has the sequence operation, in contrast to the prefix one in the classical SPAs. One can combine arbitrary expressions with the sequence operator, i.e. it is more flexible than the prefix one, where the first argument should be a single activity. The choice operation in dtsdPBC is analogous to that in MTIPP and PEPA, as well as to the alternative composition in EMPA, in the sense that the choice is probabilistic, but a discrete probability function is used in dtsdPBC, unlike continuous ones in the classical calculi. Concurrency and synchronization in dtsdPBC are different operations (this feature is inherited from PBC), unlike the situation in the classical SPAs where parallel composition (combinator) has a synchronization capability. Relabeling in dtsdPBC is analogous to that in EMPA, but it is additionally extended to conjugated actions. The restriction operation in dtsdPBC differs from hiding in PEPA and functional abstraction in EMPA, where the hidden actions are labeled with a symbol of "silent" action τ. In dtsdPBC, restriction by an action means that, for a given expression, any process behaviour containing the action or its conjugate is not allowed. The synchronization on an elementary action in dtsdPBC collects all the pairs consisting of this elementary action and its conjugate which are contained in the multiactions from the synchronized activities. The operation produces new activities such that the first element of every resulting activity is the union of the multiactions from which all the mentioned pairs of conjugated actions are removed. The second element is either the product of the probabilities of the synchronized stochastic multiactions or a specification of the joint delay and the sum of the weights of the synchronized deterministic multiactions with the same delay. This differs from the way synchronization is applied in the classical SPAs where it is accomplished over identical action names, and every resulting activity consists of the same action name and the rate calculated via some expression (including sums, minimums and products) on the rates of the initial activities, such as the apparent rate in PEPA. dtsdPBC has no recursion operation or recursive definitions, but it includes the iteration operation to specify infinite looping behaviour with the explicitly defined start and termination.

dtsdPBC has a discrete time semantics, and residence time in the tangible states is geometrically distributed, unlike the classical SPAs with continuous time semantics and exponentially distributed activity delays. As a consequence, the semantics of dtsdPBC is the step one, in contrast to the interleaving semantics of the classical SPAs. The performance is investigated via the underlying SMCs and (reduced) DTMCs extracted from the labeled probabilistic transition systems associated with expressions of dtsdPBC. In the classical SPAs, CTMCs are usually used for performance evaluation. In [136], a denotational semantics of PEPA has been proposed via PEPA nets that are high-level CTSPNs with coloured tokens (coloured CTSPNs), from which the underlying CTMCs can be retrieved. In [35, 25], a denotational semantics of EMPA based on GSPNs has been defined, from which one can also extract the underlying SMCs and CTMCs (when both immediate and timed transitions are

present) or DTMCs (but when there are only immediate transitions). dtsdPBC has a denotational semantics in terms of LDTSIPNs from which the underlying SMCs and (reduced) DTMCs can be derived.

Let us briefly describe **other SPAs with continuous time and interleaving semantics**. Such SPAs without immediate (and without positive deterministic) actions belong to the (general) classification group of MTIPP and PEPA. Such SPAs with immediate (and without positive deterministic) actions are (generally) classified as belonging to the group of EMPA.

Continuous time interleaving SPAs without immediate actions Stochastic Process Algebra (PA_S) and Generalized Stochastic Process Algebra (PA_{GS}) [179] extend LOTOS [47] with exponential and generally distributed continuous delays, respectively. PA_S has operations of inaction, prefix with (both) the rate (of an exponential delay) and action, choice, parallel composition (with synchronization by a set of actions), relabeling (with a function) and hiding (of a set of actions). In PA_{GS}, rate and action prefix is replaced with PDF (of the general delay) and action prefix. The remaining operations of PA_S are supplemented in PA_{GS} by successful termination, enabling and disrupt. PA_S and PA_{GS} have the operational semantics on labeled transition systems and denotational semantics on stochastic event structures. Immediate actions can be easily added to the two SPAs. Continuous phase type [231, 265, 170, 286, 188, 172] delays can be defined in PA_{GS}.

PEPA with phase type distributions ($PEPA^{\infty}_{ph}$) [122] extends PEPA to specify and analyze particular types of queues with potentially infinite number of clients. The activities (with visible actions or internal one) of the $PEPA^{\infty}_{ph}$ components have phase type distributed durations. The $PEPA^{\infty}_{ph}$ operators are: (action and duration or passive symbol) prefix, action choice, probabilistic choice, synchronization (by the actions set), hiding (of the actions set) and constant (for recursive definition). The operational semantics of the $PEPA^{\infty}_{ph}$ components is defined on labeled transition systems (multi-graphs). The stationary probabilities for the processes from a $PEPA^{\infty}_{ph}$ fragment are calculated with the matrix-geometric method, to overcome the state explosion problem.

Generalized (General) Process Algebra (GPA) [77] implements generalized cost operations from the semi-ring structures. GPA demonstrates a novel approach to process algebras (PAs) with measurable transitions that permits to construct different classes of PAs, such as untimed, probabilistic and stochastic ones. The mathematical structure that generally represents the transition costs operations in such PAs is a semi-ring. The GPA actions can be visible or invisible. The GPA operations are: terminal agent, (action and cost) prefix, choice, parallel composition (with synchronization by a set of actions), hiding (of a set of actions) and (recursive) definition. Operational semantics of GPA is based on multi-labeled transition systems, where each transition is labeled by (possibly invisible) action and cost (of the transition execution).

Markov Chains (MC) and Markov Action-labeled Chains (MAC) [159] are the SPAs constructed on the basis of CTMCs. In MC, the processes describe (unlabeled) CTMCs as compositions of the transition rates by the operations of (finite) sum of the prefixed (with the rates) processes, recursion (over variables) and (simple) parallel composition. In MAC (also called pure Markovian process algebra), the processes describe labeled (with visible and invisible actions) CTMCs as compositions of the pairs of actions and the transition rates by the operations of (finite) sum of the prefixed (with such pairs) processes, parallel composition (with synchronization by a set of actions), renaming and recursion (over variables).

Stochastic Probes (SP) [9] specify over SPAs the performance requirements to software systems (beginning and end of a measurement by the model developer). The types of measures are: steady-state, transient and passage-time. The SP specifications are based on the regular expressions syntax describing the behaviour that a software model must demonstrate before starting or stopping the performance measurement. Stochastic probes are themselves transformed into SPA components before a software model is explored with the process composition. SP has operators of sequence, choice, zero-or-one, iteration, range, positive closure and (standard) closure.

BioNetGen [44, 123, 124] is a rule-based language allowing one to construct a computational model of dynamics for the biochemical process of cellular signal transduction. The language can respect completely and exactly the specified enzymatic activities, potential modifications and interactions in signalling molecules. Binding and enzymatic biomolecular reactions are described by the rate-assigned reaction rules for transforming reactants into products. BioNetGen provides a graphical representation for the signal transduction networks in biology.

Grouped PEPA (GPEPA) [165, 156, 145, 126] stems from the PEPA [164] performance analysis technique for the large-scaled systems with many replicated components. The technique is based on the ODE systems, instead of the traditional CTMCs. GPEPA is a PEPA conservative extension, to which the fluid-flow analysis method is applied for approximating the mean number of the component types. The method takes as continuous the discrete state space of a process and transforms the discrete model into a coupled ODE system. The GPEPA operations over component groups (purely concurrent groups of standard PEPA components) are cooperation (over a set of synchronized actions), hiding and labeling. Operational semantics of GPEPA is used to construct population CTMCs, being the aggregated CTMCs whose states represent the sets of population members.

Stochastic Kernel language for agents interaction and mobility (StoKlaim) [235, 234, 237] extends programming and modeling language Klaim by adding exponential action delays to describe random phenomena. The

StoKlaim operations are: null process, prefixing (by action and its rate), choice, parallel composition and process instantiation. StoKlaim has operational semantics based on the rate (that extend the labeled) transition systems and transition-labeled CTMCs. The underlying stochastic process of StoKlaim is CTMC, for which the transient or stationary probabilities are calculated.

Stochastic Pi-Machine (SPiM) [245, 243, 298] is a graphical calculus for Sπ [249, 250, 251, 189], aiming to specify biological processes. SPiM is reduction equivalent to Sπ, hence, they have the same expressive power. Such a graphical representation permits to detect cycles and to animate interactions of the system components for the dynamics visualization, as well as to serve as simulator of Sπ for visual modeling and simulation of biological systems by non-specialists. SPiM is a syntactic subset of Sπ, where choice of actions is allowed only on the highest level of definitions. Stochastic behaviour is embedded into the system by assigning channels with interaction rates and delays with extinction rates, both being the parameters of exponential distribution.

κ-calculus (κ) [106, 189, 184] is a formal proteins language, whose biological interaction rules on the set of agents have rates. The rules prevent combinatory explosion when describing the dynamics with ODEs, have intuitive graphical representation based on biological knowledge, and become a natural part of building, changing and discussing the model. κ specifies well biological signal and control processes, being massively distributed systems. It formalizes directly and transparently molecular agents and their interactions in signalling networks.

Stochastic BioAmbients (SBioA) [70, 189] provides calculus of Biological Ambients BioAmbients (BioA) with a stochastic operational semantics to respect quantitative information. BioAmbients was intended to specify, simulate and analyze biological entities. The semantics of SBioA is based on the stochastic simulation algorithm that calculates the real rates (parameters of exponential distribution governing the delays) of transitions. That semantics represents an influence of chemical and physical parameters (such as molecules concentration) to dynamics of living matter. The semantics constructs stochastic transition systems, from which CTMCs are extracted. The stationary probability distributions of those CTMCs are calculated, aiming to explore behaviour of biological systems in their steady state with the reward techniques for computing performance measures.

Markovian Process Calculus (MPC) [31] describes simple Markov (with stochastic delays governed by exponential distribution) processes, constructed with the operators of null term, Markovian action prefix (with an exponentially timed action being a pair of the action name and the rate of its exponential delay), alternative composition and process constant (specified by the equation with a recursion possibility, i.e. by potentially recursive specification). The operational semantics of MPC is defined on labeled (multi)transition systems.

$PEPA + \Pi$ [131] extends PEPA, aiming to model biological systems, by applying mass action law and bounded capacity law cooperations. In $PEPA + \Pi$, cooperation operator of $PEPA$ is supplemented by the cooperation set with mass action kinetics (in addition to the standard one with bounded capacity kinetics). A special notation is also proposed for parallel composition of large numbers of independent (non-cooperating) identical processes. The relationship is established between two semantics of $PEPA + \Pi$: that in terms of CTMCs (with large state space) and that on coupled ODE systems (to handle massive quantities of processes).

Stochastic Bigraphs (SBG) [185] calculus offers a stochastic semantics for Bigraphical Reactive Systems (BRSs), a unifying framework for designing models of concurrent and mobile systems. Such reactive systems are described by rewriting rules with an initial bigraph, to which the rules are applied. Bigraphs are the algebraic terms, represented by special graphs that represent communication of agents and their spatial configuration, so that some nodes can contain others. SBG provides BRSs with a uniform stochastic interpretation, where abstract rules of biomolecular reactions have positive rates assigned, used to calculate reaction rates. Stochastic transition systems, obtained from the SBG semantics, are taken to derive CTMCs that are analyzed with simulation.

Stochastic Păun- (P-) Systems (SPS) [239] is a class of computational models for cell biology and membrane computing. The main ingredients of a P-system are membrane structure (that delimits compartments), multisets of objects and biochemical reaction rules. The rules are endowed with the rates reflecting propensity of the corresponding reactions and can handle both objects and membranes. The known types of P-systems are cellular, tissular and neural ones. The analysis of P-systems consists in applying symbolic probabilistic model checking.

Chemical Ground Form (CGF) [82, 83, 303, 139] is a calculus for modeling biochemical reactions, a modification for biological systems of Sπ [249, 250, 251, 189] without communication. The actions in CGF have associated stochastic rates (positive real numbers, the exponential distribution parameters). Invisible action expresses unary reaction while complementary identically named visible actions specify two reactants in a binary reaction with the same name. The operators of CGF are: (successful) termination, prefix and parallel composition. The probabilistic semantics of CGF is based on DTMCs, its stochastic discrete-state semantics is constructed on CTMCs and its continuous-state semantics is defined on ODEs. The abstract probabilistic semantics of CGF is built by extracting labeled interval DTMCs from abstract labeled transition systems, based on abstract multisets, with intervals of integers used instead of single multiplicities. CGF corresponds to basic chemistry.

Chemical Parametric Form (CPF) [83] enriches CGF [82, 83, 303, 139] with parametrization, communication and reuse, being more general subset of Sπ [249, 250, 251, 189]. The stochastic processes in CPF can be converted to chemical reactions (interrelated with CGF). The mapping of CPF to chemistry results in the parametric and

compositional indirect (two-step) mapping of CPF to ODEs that is easier to define and understand than a direct (one-step) mapping. That indirect mapping can be interpreted as the ODE semantics of the CPF processes.

Biochemical Ground Form (BGF) [84, 303] extends CGF [82, 83, 303, 139] with the capabilities of complexation (joining) and splitting molecules, through association and dissociation. Calculus BGF adds to the syntax of CGF two pairs of complementary actions, intended to specify association and dissociation. The operators of BGF additionally include trailing of the association histories. The discrete-state semantics of BGF (like that of CGF) is based on CTMCs, extracted from labeled transition graphs. Differently from CGF, calculus BGF is Turing powerful and corresponds to biochemistry.

Stochastic Calculus of Communicating Systems (StoCCS) [182, 236] is a CCS [223] stochastic extension being a fragment of Sπ [249, 250, 251, 189]. The StoCCS operators are: null process, prefixing (by action label and its rate), stochastic choice and parallel composition. Labeled state-to Function Transition Systems (FuTSs) are used to unify definitions of the (S)PAs semantics with a goal to compare the calculi. Based on FuTSs, two stochastic enhancements of CCS with binary synchronization are proposed: $StoCCS_{AA}$ with active input and output actions of the channel (along which the synchronization signal is transmitted from the input to output action) and $StoCCS_{AP}$ with passive input and active output actions of the channel.

Stochastic Calculus of Looping Sequences (SCLS) [17, 20] extends stochastically Calculus of Looping Sequences (CLS). SCLS is a quantitative term rewriting formalism for describing evolution of the microbiological systems (such as cellular pathways) while taking into account the activities speed, represented by stochastic rates (the exponential distribution parameters). SCLS has operators of sequencing, looping, containment and parallel composition. The looping operator connects the ends of sequence, resulting in the circular (looping) sequence that can specify membrane. CTMCs are extracted from the semantics of the SCLS systems, with a goal of simulating and verifying their properties with stochastic model checking.

Language for Biochemical Systems (LBS) [240, 241] combines modeling (with rewrite rules) and modularity. LBS is based on Calculus of Biochemical Systems (CBS), intended for modular specification of metabolic, signalling and regulatory networks, as reactions between modified complexes that occur concurrently in the hierarchy of compartments, with possible interactions and transport across compartments. LBS has the species expressions, parametrized modules with subtypes, nondeterminism, as well as nested declarations of species and compartments. Formal specification of the language is given by abstract syntax and general semantics, being parametric on the structure of the target semantic objects, such as PNs, coloured PNs (CPNs), ODEs and CTMCs.

Biochemical Performance Evaluation Process Algebra (Bio-PEPA) [95, 96, 97, 129, 146, 218, 219, 217, 166] is constructed to model and analyze biological networks. For that, PEPA [164] is extended with stoichiometry (quantitative interrelations of reactants in biochemical reactions), the species roles in reactions and functional rates for different kinetic rules types of the reaction dynamics. The processes in Bio-PEPA are seen as species rather than molecules, like in Sπ [249, 250, 251, 189]. The Bio-PEPA operators are: prefix combinator (with the pair of action type and its stoichiometry coefficient, in the role of reactant, product, activator, inhibitor or generic modifier), choice, constant, cooperation (by the activities set) and concentration level. The operational semantics of Bio-PEPA is defined on stochastic labeled transition systems, based on the discrete concentration levels. Bio-PEPA maintains several analysis methods: Stochastic Simulation Algorithm (SSA) [135], numerical solution for the steady-state analysis of the CTMC (with discrete concentration levels) underlying the model semantics, translation into the equivalent deterministic model of ODEs, as well as stochastic model checking.

Spatial Calculus of Looping Sequences (that we call SpCLS) [18, 19] is a spatial extension of Calculus of Looping Sequences (CLS) that observes the position and taken space of biological elements with time passage in a continuous two- or three-dimensional space. The movement of elements in the space can be exactly described, and they can interact when constraints on their positions are satisfied. Both deterministic and stochastic movements of the elements can be specified. Like in SCLS [17, 20], rewrite rules for reactions in SpCLS are endowed with kinetic parameters defining their stochastic propensity rates. The reaction rates are the parameters of exponential distribution that models the expected duration of a reaction with a specific combination of reactants.

Stochastic Calculus of Wrapped Compartments (SCWC) [101, 262] naturally describes a wide class of biological systems via direct representation of membranes and compartments. SCWC is a variant of SCLS [17, 20] without sequencing operator and with multisets (instead of ordered sequences) of atomic elements, to specify membranes. SCWC is intended to simplify the development of automatic analysis tools while preserving the SCLS expressiveness. Every reaction rule in SCWC has an assigned rate function of the context. SCWC has a stochastic operational semantics, from which a CTMC is extracted to verify the system properties. To identify kinetic parameters of biological systems in SCWC, an effective stochastic simulator is applied (instead of standard ODE-based methods), thereby extending the class of investigated systems.

Markovian Agent Spatial Stochastic Process Algebra (MASSPA) [143] formally describes behaviour of Markovian Agent Models (MAMs), a spatial stochastic modeling framework. A Markovian agent in a MAM is a simple sequential component that can have local transitions (with exponential rates), possibly sends messages and can have message-induced transitions. The MASSPA operations are: (exponentially rated) prefix, choice,

(Poisson distributed number of) message sending, (probabilistic) message reception, constant, null process and parallel (with the message exchange). The underlying CTMC of a lumped process is approximated with special techniques. The CTMC describes the density evolution of an agent type at current moment for a given location. The ODE-based analysis of higher moments (such as variation) is proposed in the performance evaluation of discrete spatial stochastic models. Stochastic simulation is used to verify the ODE-based approximation of mean and standard deviations for counting the model components.

Process Algebra with Hooks (PAH) [110, 111, 112] is intended to model biological systems at multiple levels of detail (scales). The processes of PAH describe different scales, such as biochemistry, cells and tissue. The operators of PAH include the deadlock process, agent definition, sequential execution and nondeterministic choice. In addition, two symmetric operators of synchronization (on the set of actions) are used to compose processes within one level of detail (horizontal cooperation) and between the levels (vertical cooperation). Stochastic semantics of PAH is based on functional rates of reactions. Continuous time and exponential delays are applied.

Typed Stochastic Calculus of Looping Sequences (TSCLS) [43] is an extension of SCLS [17, 20] with the types of the elements that speed up or slow down reactions, such as positive of negative catalyzers. The operational semantics of TSCLS that respects the types of the species is applied to derive the stochastic evolution of a system, where the activities speeds can be modified by catalyzers. The types offer an abstraction that can represent the interactions of elements without exact specification of their positions. The rewrite rules have the rates that permit the evolutions of the rules follow different probability distributions, what is useful for high-level simulation. The typed stochastic semantics generates the transition systems, producing the CTMCs that are applied in the simulation procedure.

Fluid Process Algebra (FPA) [287, 288] is a subalgebra of GPEPA [165, 156, 145, 126] being a conservative extension of PEPA [164] with fluid semantics, intended to simplify solving the systems of coupled ODEs. FPA has the expressive power of GPEPA without hiding operator. The FPA operations over the PEPA-like model components, specified using operations of (action and rate) prefix, choice, recursive definition with constant, and cooperation, are: cooperation (over a set of synchronized actions) and labeling. The labels are used to distinguish the representative components, which are replicated. A fluid atom is an occurrence of some labeled component in a process. FPA has a fluid semantics based on the underlying systems of ODEs that are used for the analysis. Each ODE system approximates the evolution in time of the processes population representing a local state.

Simple Stochastic Process Algebra (SSPA) [296] describes CTMCs with a product-form solution, implying that their stationary distributions are effectively solvable. The proofs of important properties for SSPA are simpler than for labeled Markov Automata (LMAs) that have a direct relation with CTMCs, but do not permit to use the inductive structure of the language. SSPA preserves semantics of the cooperation operator of LMAs, what is important for correctness of the product-form solution. The operators of SSPA are the empty process, identifier, choice (from a set of processes, prefixed with actions and rates), closure (replacing variable by a real-valued rate in each pair of an action and a variable) and interaction (among many processes, by a set of actions).

Calculus of Chemical Systems (that we call CChS) [248] is proposed for modular description of chemical reaction systems and modeling with rules in systems biology. CChS is based on CCS [223], but with communication replaced by chemical reactions. The operations of (quantitative version of) CChS include rule (with a positive real-valued rate), parallel composition, the empty process, local definition and process identifier. Different compositional semantics of (quantitative) CChS are given, based on quantitative PNs (CTSPNs), ODEs and stochastic transition matrices. Complete axiomatizations and normal forms are presented for all the semantics.

Fluid Extended Process Algebra (FEPA) [289, 290, 173] explores the models, specified with large systems of ordinary differential equations (ODEs). The sequential process components, called fluid atoms, can have a multiplicity (the number of copies in the model specification). There are two variants of synchronization: with the minimum of the rates of the synchronized processes (to model computer systems, as in PEPA [164]) or with their product (to represent chemical reactions and biological networks with the rule of large numbers, as in Bio-PEPA [95, 96, 97, 129, 146, 218, 219, 217, 166]). The FEPA processes are described by the ODE systems with the derivatives of the population functions that define the multiplicities (numbers of replicas in a population) of fluid atoms by one variable (time). The typical FEPA multiplicity values are rather large and interpreted as non-negative real (instead of natural) numbers, defined by the population functions of time, whose values can be found for every particular moment. The FEPA expressiveness is restricted to the processes being a parallel composition (with the embedded synchronization by the cooperation actions) of the fluid atoms denoting a large number of copies of simple sequential components, specified with the operations of (action and rate) prefix, choice and recursive definition with constants. The FEPA fluid atoms are considered uniformly, without dividing into "discrete" atoms with small multiplicities and "continuous" ones with large multiplicities.

Probabilistic Programming Process Algebra (ProPPA) [132, 133, 166] is an extension of Bio-PEPA [95, 96, 97, 129, 146, 218, 219, 217, 166]. ProPPA permits uncertain description of models and application of the machine learning techniques, aiming to include observational information in the modeling. The semantics of ProPPA is defined on probabilistic constraint Markov chains (PCMCs), an extension of constraint Markov chains (CMCs).

CMCs generalize DTMCs so that the state change probabilities become not fixed, but satisfy some constraints or belong to a set of acceptable values. Markov decision processes (MDPs) or uncertain Markov chains (UMCs) are used to simulate CMCs. Analogously, PCMCs generalize CTMCs, but they associate a probability distribution with the constraint satisfaction set of values. The stochastic relation defines the rate of the transition from one complete system to another. The rate is generalized in ProPPA to a distribution over possible rates.

Collective Adaptive Resource-sharing Markovian Agents (CARMA) [58, 199, 130] is used to specify and analyze collective adaptive systems. CARMA has linguistic constructs for modeling and programming systems that work in openended and unpredictable environments. A model is a collective of components, each expressing a set of attributes. To model dynamic aggregations (ensembles), CARMA has communication primitives based on predicates (over the expressed attributes), to select the communication participants. There are multicast- and unicast communications. The CARMA operations are: empty process, component destroy, action prefix, choice, parallel composition, predicate guarding and recursive definition. The operational semantics is defined on labeled state-to Function Transition Systems (FuTSs) [236], from which the action-labeled CTMCs are derived.

Cox and Convenience Calculus (CCC) [254, 42] is constructed to generate of and manipulate continuous acyclic phase type (APH) distributions for compositional representation of process delays. The delays correspond to the completion times of activities. Basic delays are described by exponential distributions. Complex delays are obtained by composing basic delays with stochastic operations on continuous probability distributions: summation (convolution), minimum and maximum. CCC generates representations of APH distributions in Cox forms. The stochastic operations have the respective ones among CCC operators: rate (of exponential delay), disabling (race between two exponential distributions), sequential composition (corresponds to convolution), choice (corresponds to minimum) and parallel composition (corresponds to maximum). The operational semantics maps the CCC expressions onto Markov (decorated) transition systems and then interprets them as absorbing CTMCs.

Modelling in Ecology with Location Attributes (MELA) [297] is designed to model ecological systems while respecting location in space and influence of environment. MELA is a high-level language for formal description of the ecological concurrent systems of agents that can evolve simultaneously and interact. It specifies population models with single or multiple species. Its actions have rates being the exponential distribution parameters. The MELA operators are: no-influence action, influence action, probabilistic effect of action, choice, constant, null component and parallel. Operational semantics of MELA is defined on the labeled transition systems with qualitative and quantitative information about actions. MELA supports stochastic simulation and direct analysis of the underlying CTMCs, as well as numerical solution of the fluid approximation with the ODE systems.

Network of Broadcasting Agents (NBA) [57] is a fragment of CARMA [58, 199, 130] without attributes. NBA supports both unicast and broadcast communication to model quantitative aspects of the systems of broadcasting processes. Within the agents, actions have rates (the exponential distribution parameters), broadcast messages have probabilities while unicast messages have weights, used to calculate probabilities. Stochastic operational semantics of NBA is defined on labeled state-to Function Transition Systems (FuTSs) [236]. Fluid approximation theorem is proved for the NBA population semantics, based on population CTMCs.

Continuous time interleaving SPAs with immediate actions Markovian Process Algebra of M. Bernardo, L. Donatiello and R. Gorrieri (MPA) [34] is used to model functionality and performance. The MPA action is a pair of its type and rate. According to type, the actions can be external (observable) and internal (invisible). According to rate, the actions can be passive (with zero rate) and active (with a positive rate). The active actions can be timed (with a finite positive rate) and immediate (with an infinite positive rate). The immediate actions have priorities ans weights. The operators of MPA include null term, prefix (with an action), functional abstraction (hiding, by the actions types set), temporal restriction (by the passive actions types set), relabeling (with a function), alternative composition (choice), parallel composition (with the synchronization set), constant (for recursive definition). The operational semantics of MPA is based on labeled transitions systems, from which homogenous CTMCs are extracted, being a performance evaluation formalism (stochastic model, Markovian semantics). The net semantics (distributed model) of MPA is defined on GSPNs that clearly represent parallelism and causality.

Markovian Process Algebra of P. Buchholz (the extended version that we call MPA-B) [72, 73] is based on CCS and CSP. Every action in MPA-B has a basic transition rate. Activities are the pairs from an action and the value parameter, expressing the action speed or its invocations number. There is also a distinguished invisible action. For each activity, action is executed after exponential delay with the rate equal to the product of the activity value parameter and basic transition rate of the action. This enables compositional analysis with parallel composition of process expressions. MPA-B has the operators of termination, prefix, choice, parallel composition (with synchronization on the actions set), hiding and recursion. The operational semantics of MPA-B is defined on finite multi-labeled transition systems (MLTSs), whose transitions are labeled by the pairs from an action and the value parameter. The MLTSs, constructed with the rules of structural operational semantics, are used to extract the underlying CTMCs. The extensions of MPA-B with immediate activities and

non-exponential activities durations (such as phase type distributed) are proposed.

Probabilistic Markovian TImed Processes and Performability evaluation (PM-TIPP) [257, 258] is an extension of the Markovian variant of TIPP [140] by *probabilistic branching*. It can be specified by weighted immediate activities, also describing management actions, to test for resources availability. Instead, PM-TIPP is MTIPP [162] with an additional *probabilistic choice* operator. PM-TIPP has hiding and relabeling operators as well. In PM-TIPP, Markovian transitions (with exponential delay) are merged at the semantic level with their direct followers, corresponding to the probabilistic choice. The operational semantics of PM-TIPP is based on labeled transition systems with two transition relations denoting Markovian transitions (labeled with actions, rates and additional words) and probabilistic transitions (labeled with probabilities and additional words).

Spectral Expansion TImed Processes and Performability evaluation (SE-TIPP) [225] is a modification of TIPP [140] enabling solution with the spectral expansion method (SE). SE-TIPP can model processes with infinite state space, resulting in a significant modeling power increase. The systems with infinite number of states are modeled intuitively while the specification scheme for such systems gives a compact representation of infinite Markov processes, to be solved with SE. In SE-TIPP, actions have the associated rates (parameters of exponential distribution) The operations of SE-TIPP are: (successful) termination, variable, prefix (with an activity), choice, parallelism (with the synchronized actions set), hiding and recursion. The underlying Markov processes may be two-dimensional: finite in one direction and infinite in the other. The model description is transformed into a matrices set, used in the SE solution procedure, followed by the performance measures calculation.

Stochastic Process Algebra for Discrete Event Simulation of P.G. Harrison and B. Strulo (that we call SPADES-HS) [151] extends Timed CCS [301, 302] to formally describe discrete event simulation. SPADES-HS specifies time progress and probabilistic choice (discrete or continuous): selecting from a countable processes number or taking a random waiting time. The SPA describes infinite (as a rule) semantic objects, has immediate and delayed prefixing, can specify separately random timer starts, timer completions and current activities. Time delays may be non-exponential, resulting in more generality. SPADES-HS has visible actions, their conjugates and the (self-conjugate) invisible action. The operators of SPADES-HS are: deadlocked (terminated) process, time prefix with fixed delay, (impatient) prefix with action, patient prefix with action, nondeterministic choice, probabilistic choice, time prefix with random delay (with the density), parallel, relabeling (with the renaming function), restriction (on the actions set) and recursion. The operational semantics is defined on labeled transitions systems with the labeled (with actions), probabilistic and (time) evolution transitions.

Stochastic Timed Calculus (STC) [161] extends CCS [223] with stochastic time under the maximal progress assumption. The (visible and a special invisible) immediate actions of STC are separated from delays, governed by an exponential distribution with the parameters called rates. The STC operations include delay (with a rate) prefix, action prefix, choice and recursion (by variables). The operational semantics of STC is defined on labeled transition systems with the action (the action name labels) and timed (the rate labels) transition relations.

Stochastic Process Algebra for Discrete Event Simulation of P.R. D'Argenio, J.-P. Katoen and E. Brinksma (SPADES) [7, 8, 4, 159, 6] is a non-Markovian stochastic calculus. The actions in SPADES are separated from continuous time generally distributed stochastic delays. The semantics of SPADES is based on stochastic automata (SAs) [5] that can be executed using discrete event simulation. The semantics of SAs themselves is defined via probabilistic (labeled) transition systems with general distributions (discrete, continuous and singular). The operations of SPADES include stop (inaction) process, (action) prefixing, triggering condition, choice, clock setting, parallel composition (with the synchronization set of actions), left merge (with set of actions), communication merge (with set of actions), renaming (with the function) and process instantiation (for recursive definition).

Stochastic Process Algebra ($\mathcal{SPA}$) [46] is an extension of TCSP [167] with anonymous (unnamed) timed actions, to model continuous time stochastic delays between visible actions. All named (visible and invisible) actions are immediate (of zero duration) and have no delays assigned. Timed actions are specified by positive real numbers being the exponential distribution parameters of delays (rates) of those (Markovian) actions. Functional and temporal behaviour is treated separately. The $\mathcal{SPA}$ operators are: empty (stop) process, prefixing with immediate named actions, prefixing with timed anonymous actions, choice, parallel composition (with the synchronization set of actions), restriction (on the actions set) and recursion. Synchronization may occur only between immediate named actions and termed timeless [163]. The structural operational semantics of $\mathcal{SPA}$ is based on the rules for timed anonymous (Markovian) actions and for immediate named actions. It is defined on labeled transition systems with two transition relations: implementation of unnamed time delay or instantaneous execution of a named action. The compositional structure of the $\mathcal{SPA}$ specifications is used for a novel solution of the underlying stochastic process by reformulation of the underlying CTMC as semi-Markov processes.

Non-Markovian Stochastic Process Algebra (NMSPA) [196] permits general (not only exponential) probability distributions of delays, resulting in increase of the expressive power. Some practically important distribution types are allowed, such as uniform, discrete and Poisson. This fact allows one to specify passive, urgent and immediate actions. Besides visible actions, NMSPA has the (urgent) invisible action. The operators of NMSPA include deadlocked process (STOP), choice of the fastest action (with the random variable of delay) from those

prefixing processes, parallel composition (with the synchronization set), restriction (on the actions set), renaming (with the function) and recursion. The operational semantics of NMSPA is based on labeled transition systems.

Stochastic Basic Language Of Temporal Ordering Specification (that we call SB-LOTOS) [160] is an extension of Basic (data-absent) LOTOS [47] with the continuous phase type distributed delays [231, 265, 170, 286, 188, 172]. The actions can be visible ones, the invisible (internal, unobservable) one and the successful termination one (denoted by δ). The SB-LOTOS operators are: inaction (stop), successful termination (exit), action prefix, rate (of the exponential delay) prefix, choice, sequential composition (enabling), disabling, parallel composition (by a set of actions), hiding (of a set of actions), relabeling (with a function), process instantiation (for recursive definitions) and elapse (describing continuous phase type delay via its absorbing CTMC using the actions start, delay and break). The SB-LOTOS operational semantics is defined on the generalization of Interactive Markov Chains (IMC) [158, 159], with the transitions on the time being continuously phase type distributed.

Biochemical Stochastic π-calculus (BioSpi) [253] extends the name-passing SPA Sπ [249, 250, 251, 189] with the goal of investigating biomolecular systems and interactions. BioSpi describes the structure and dynamics of biochemical networks. The actions have the rates assigned (parameters of the exponential distribution of delays), corresponding to the basal reaction rates. The channel requests (send or/and receive, or withdraw) may have the infinite rate, i.e. be instantaneous. BioSpi inherits all operations of Sπ, but the prefixing actions are replaced by the pairs of actions and rates. Reduction semantics of BioSpi respects the time and probability of biochemical reactions. The quantitative analysis is based on the stochastic discrete simulation with support of mobility.

Immediate Markov Action-labeled Chains (IMAC) [159] enriches MAC [159] with immediate actions that are executed without any delay. The operations of MAC are supplemented with immediate (action) prefix, such that the prefixing immediate action is executed instantly before evolving into the prefixed process. The operational semantics of IMAC is based on labeled transition systems with two transition relations, describing executions of durational actions (with rates) and those of immediate actions, respectively. The choice between durational actions is probabilistic while that between immediate actions is nondeterministic.

Interactive Markov Chains (IMC) and their Interactive Markov Language (IML) [158, 159, 33] is a compositional continuous time behavioural model. Immediate actions in IMC are added to MC, i.e. time transitions (with rates) and action (interactive, immediate) transitions are separated. The processes describe interactive Markov chains (IMCs) with visible nd invisible actions as compositions of the actions and transition rates by the operations of (finite) sum of the prefixed (with actions or rates) processes, parallel composition (with the synchronization set of actions), renaming and recursion (over variables). The operational semantics of IMC is based on the union of labeled transition systems and CTMCs. The semantic transitions are Markovian, with the rates-defined exponentially distributed delays, or interactive, corresponding to instant execution of (possibly invisible) actions.

Interactive Generalized Semi-Markov Processes (IGSMP) [67] is a calculus with interleaving semantics for (visible and invisible) actions and ST- (Start-Termination-) semantics for (non-Markovian) delays. The actions are all immediate and separated from delays, specified as a pair of distribution function (of the duration probability) and weight. IGSMP specifies the probabilistic timed delays with general continuous distributions and synchronized actions with zero duration, as well as probabilistic (under preselection policy), nondeterministic and prioritized choice. The operators are: empty process, delay prefix, action prefix, choice, hiding (the set of actions turned into invisible ones), relabeling (with the function), parallel (with the synchronization set of actions), recursion (over variables). The operational semantics of IGSMP constructs generalized semi-Markov processes (GSMPs) being the probabilistic systems with generally distributed time, which are extended with the action transitions describing interactions among the system components. The concurrent execution of delays is expressed by a variant of ST-semantics, based on dynamic names. For performance evaluation, GSMPs are extracted from IGSMPs and analyzed with mathematical and simulative methods to obtain the performance measures.

Value Passing Stochastic Process Algebra (VPSPA) [197, 198] extends NMSPA [196] with the value passing feature. VPSPA permits generally distributed delays. The properties of the VPSPA specifications are studied by translating them into the programs of concurrent functional programming language Eden, where parallel processes are executed and their quantitative properties are investigated. To analyze the specified generally delayed systems, simulation is used instead of model checking. The performance of the system implementation is simulated, in order to obtain the real estimates of its theoretical performance. The communication actions are input (message receiveing) and output (message transmitting) ones. There is also the invisible action (urgent, immediate). There are data transmission channels with the values. Stochastic actions describe delays specified by random variables with generally distributed (continuous time) delays. Choice has no probability assigned while parallelism has the associated set of indexed starts and terminations of delays (like in the ST-semantics of IGSMP [67]). Further operators are termination, delay prefix, channel receiving, channel transmission, conditional, hiding and recursive definition. The operational semantics of VPSPA is based on labeled transition systems.

Semi-Markov PEPA (SM-PEPA) [61, 62, 63, 10] extends PEPA with the action delays distributions that ensure the underlying stochastic model to be an SMC. In SM-PEPA, actions are associated with symbolic priorities and parameters of general delays. The parameters are the rates of exponential delays or the pairs of a weight and

a generally distributed delay, defined by Laplace transform. The SM-PEPA operators are: prefix, cooperation, hiding and constant. At each priority level, only one type of actions is allowed: Markovian or semi-Markovian. The semi-Markov synchronization is specified by the user-defined functions of the combined weight and delay. Operational semantics of SM-PEPA is constructed with the rules for Markovian and semi-Markovian actions. For the model analysis, the SM-PEPA specifications are automatically transformed into semi-Markov SPNs.

Stochastic Beta-binders (that we call SBB) [108] is a stochastic version of Beta-binders [252, 220] with typed interaction sites for accurate description of biological entities. In SBB, quantitative measures on biological phenomena are studied. The quantitative parameters are extracted from typed interaction sites, resulting in the affinity concept. The SPA has exponential or zero action delays. The quantitative information is given by the action rates (exponential distribution parameters) that represent stochastic behaviour and define reaction speeds. The operators are: inactive process, (input, output, hiding, unhiding and exposing) prefixing (with the pair of action and rate), parallelism, static binding and multiple instances of prefixed process. The operational (stochastic reductional) semantics is based on labeled transition systems. The underlying stochastic process is CTMC.

MOdeling and DEscription language for Stochastic Timed systems (MODEST) [45, 147, 80, 152] is a formalism for modular description of reactive systems behaviour respecting functional and nonfunctional aspects (timing or service quality) of systems in a single specification. The actions are separated from (random) delays, there are simple and structured data types, structuring mechanisms (like parallel composition and abstraction), means to control the assignments granularity, exception handling, nondeterministic and random branching, timing. There are patient and impatient actions, exception names, the unhandled error action, the break action and the unobservable (silent) action. The operations incluse stop (no activity), abort (unhandled error), break (action with no restriction), act (action with no restriction), condition (when), urgency (of the first activity), process instantiation, call by value, choice, sequential composition, loop, relabeling, alphabet extension, exception handling, probabilistic prefix and parallel composition (with the multiway synchronization set). The operational semantics is defined on stochastic timed automata, a union of timed and stochastic automata [5], interpreted over (infinite) timed probabilistic transition systems (with immediate action transitions to discrete probability distributions over successor states, and timed transitions with positive real-valued delays). MODEST describes a wide spectrum of models: labeled transition systems, timed and hybrid automata (and probabilistic variants of them), stochastic processes like (discrete and continuous time, generalized semi-) Markov chains and (discrete and continuous time) Markov decision processes, Markov and stochastic (timed and hybrid) automata.

Stochastic Concurrent Constraint Programming (sCCP) [50, 52, 51, 53, 54, 55, 56] is a stochastic extension of CCP [247]. sCCP is proposed for modeling and analysis of biological systems. In sCCP, communication is asynchronous, species are described by variables, reactions are represented by constraints on the variables and rates are specified by functions. The sCCP operations are: tell prefix (with rate), ask prefix (with rate), choice, empty process, (recursive) procedure call (with rate), hiding and parallel composition. Two operational semantics of sCCP are defined: the continuous time one (standard) and discrete time one (with the rates interpreted as weights to calculate probabilities; we call the latter SPA dsCCP), resulting in CTMCs and DTMCs, respectively, as the performance analysis model. The traditional semantics (the standard one on CTMCs and differential one on ODEs with fluid flow approximation) are supplemented with hybrid semantics on (Non-)Deterministic Hybrid Automata (DHA, NDA) and Simple Stochastic Hybrid Automata (SSHA). The analysis consists in stochastic simulation or in translations into the stochastic verification programming system, ODEs and hybrid systems.

Nano-κ-calculus (nanoκ) [103, 192], based on κ [106, 189, 184], is intended for modeling, analysis and prediction of the molecular devices properties. Biochemical systems are modeled in nanoκ by defining their reaction sets. The semantics of nanoκ is based on reaction rules, where reactions (creations, destructions or exchanges) have finite or infinite rates. The stochastic model of nanoκ is based on the stochastic transition system with finite (for markovian transitions) and infinite (for invisible, silent, interactive transitions) rates, thus resulting in the markovian and transient states, respectively. From that transition system (with only silent actions), interactive Markov chain (IMC) [158, 159] is extracted that can be downgraded to CTMC, if all invisible interactive transitions are partitioned into the confluent directed acyclic graphs of finite depth. The nanoκ implementation into SPiM [245] takes molecules as processes and derives the overall (stochastic) behavior by communication rules.

Stochastic Pi-Calculus for Concurrent Objects (SPiCO) [187, 220] is a modeling and simulation language for systems biology, based on Sπ [249, 250, 251, 189]. SPiCO supports high-level modeling by using the multi-profile concurrent objects with static inheritance that correctly represent interacting molecules. The SPiCO operators are: empty process, parallel composition, channel creation, sum, application, pattern input (receive), tuple output (send) and (recursive) definition. The SPiCO stochastic semantics of is defined on CTMCs. The transitions can be timed (exponential delays with finite rates) or immediate (zero delay with infinite rate and probabilities). To construct CTMC, immediate transitions (corresponding to instantaneous reactions) are eliminated and their probability effects are respected. SPiCO can be encoded back into BioSpi [253] while preserving the semantics.

Stochastic π-calculus with polyadic synchronisation (Sπ@) [292, 293, 294] is an enrichment of the language Sπ [249, 250, 251, 189]. The calculus Sπ@ is a stochastic extension of the process algebra π@ [295]. In Sπ@,

finite and infinite rates are allowed, hence, there exist immediate actions (reaction types), taken as possessing higher priority (from two possible, defined by types of the rates) than standard ones. The operators of Sπ@ are: null process, guarded (action prefixed) choice, parallel, guarded (action prefixed) replication and scope restriction (of a name). The language Sπ@ can flexibly model multiple compartments with dynamic structure and provides enhanced biological faithfulness. The biological systems specified in Sπ@ are used in the extension of Stochastic Simulation Algorithm (SSA) [135] that handles multiple compartments with varying volumes.

Attributed π-calculus ($\pi(\mathcal{L})$) [176] extends π-calculus [224] with attributed processes and attribute dependent synchronization, for application in systems biology. $\pi(\mathcal{L})$ is parametrized with the language $\mathcal{L}$ defining the attribute values and expresses polyadic synchronization with different compartment organizations. The $\pi(\mathcal{L})$ operators are: defined process, parallel composition, channel creation, summation of (receiver of sender prefixed) alternative choices, empty solution and parametric process definition. The nondeterministic (small step reduction) and stochastic (CTMCs extraction) semantics are proposed, with the rates possibly dependent on the attribute values. Finite and infinite (immediate transition) rates are allowed in the stochastic semantics, for which a simulation algorithm is developed.

EXtended Stochastic Probes (XSP) [10, 98] is an enrichment of SP [9] for the state-aware performance analysis with the queries combining instantaneous observations of the model states and finite sequences of the model activities observations. The queries are realized in XSP by composing the observers with the described by an SPA model that has a discrete time representation via CTMC. The communicating local probes have immediate actions for instantaneous communication among components of the probes and transferral the states information without affecting the behaviour and without perturbing the performance analysis. The XSP novelty is a combination of the state and activity specifications with local and global observations. The XSP activity probes operators are: (activity) observation, sequence, choice, labeling, (upper bounded) iteration, range (lower and upper bounded) iteration, one-or-more, zero-or-more, zero-or-one, resetting and bracketing.

Phase Type Processes (PTP) [300] allow for probabilistic and nondeterministic choices, as well as continuous phase type (generalizing exponential) [231, 265, 170, 286, 188, 172] and zero delays. The (visible or invisible) action transitions, used to react on the external stimuli, are separated from the phase type transitions. The PTP semantics is constructed via the path probabilities with respect to schedulers resolving the nondeterministic choices in the timed process history. Parallel composition is studied in the context of the partial memoryless property. A mapping from PTP to a subclass of the single phase processes with exponentially distributed delays is defined.

BlenX [114, 115, 116] is a language used with Beta-binders calculus [252, 220] as a basis for scaled structure for modeling, simulation and analysis of biological systems. With that goal, a programming system Beta Workbench (BWB), based on BlenX, is described that simplifies development of the bio-systems models at different abstraction levels, can simulate their dynamic behaviour, check and ask the simulation results. BWB has three tools that jointly use the compiler and runtime environment of BlenX: stochastic simulator, CTMCs generator and reactions generator. The actions in BlenX have the rates (parameters of the exponential distribution of delay) or executed without delay. The operators over the BlenX processes are: deadlocked process, parallel (logical *and*), (guarded) choice (logical *or*), conditional (if-then), (guarded) replication and (action sequence) prefixing.

DNA Strand Displacement language (DSD) [244, 190, 191, 189] is intended for designing, modeling and simulation of the DNA circuits that make computations via strand displacement. The examples of applying that computational mechanism are the digital logic circuits and catalytic signal amplification circuits, functioning as efficient molecular detectors. The DSD syntax describes molecules, their segments and three types of sequences: concatenation, left and right overhangings. DSD can model slow reactions with finite rates and fast reactions that occur instantly or much faster than others. After exploring all trajectories (interleavings of reactions, reduction paths) of the specified system, CTMCs are generated, used to analyze quantitative properties of its behaviour.

Markovian Calculus of Communicating Systems (mCCS) [117, 118] is a Markovian extension of CCS [223] with the interpretation on Markov automata (MAs) that describe systems behaviour via nondeterministic, probabilistic and timed events. Markov labeled transition systems are extracted from MAs in order to study their behaviour. Analogously to [158, 121], external actions are taken as immediate, time progresses when no internal activity is possible, and timed actions only demonstrate Markovian behaviuor. The operations of mCCS include (successful) termination, indefinite (imprecise) and definite rate prefixing, insistent prefixing with an action, choice, parallelism, process constant (associated with definition) and probabilistic choice (with a finite index set).

Markov Automata Process Algebra (MAPA) [281, 282, 141, 142] is proposed with a goal of effective compositional specification, generation and modeling of Markov automata (MAs), whose events can be nondeterministic or happen probabilistically, or have exponential timed delay. The operations of MAPA include process instantiation (allowing recursion), conditional, nondeterministic choice, (possibly infinite) nondeterministic choice over data type, probabilistic choice over data type and rate (of the exponential delay) prefixing. For the modular construction of large systems, the top-level operations can be added: parallelism, encapsulation (analogous to restriction), hiding and renaming. The operational semantics of MAPA is defined in terms of MAs.

Immediate PEPA (iPEPA) [157] adds to PEPA [164] immediate actions with weights. The weights are trans-

formed into probabilities while each parallel composition application and the resulted probabilities are recalculated at all composition levels. Immediate actions supplement standard timed actions having rates and are used to represent communication between the measurement processes, intended for specification of stationary and transient passage time measures. The iPEPA operations are: standard and immediate (high-priority) prefix, choice, constant and cooperation (on the action types from a set). After removal of vanishing states (in which only immediate actions are executed) from the transition systems of the well-behaved (without immediate cycles and with deterministic initial behaviour) iPEPA components, the derived transition systems (with only timed transitions, i.e. those accomplished by timed actions) are obtained, translated into CTMCs for performance computation.

Immediate GPEPA (iGPEPA) [157] is an enrichment of GPEPA [165, 156, 145, 126] with immediate actions having weights. The actions in iGPEPA are timed (a pair of timed action type and rate) or immediate (a pair of immediate action type and weight). A component of iGPEPA is a component group with the group labeling or a cooperative composition of the iGPEPA components. The component group is an f- (fluid-) component or an unsynchronized parallel composition of f-components. The iGPEPA operations over component groups (purely concurrent groups of standard components of iPEPA [157]) are cooperation (over a set of synchronized actions) and labeling. The iGPEPA models have the associated systems of coupled first-order ODEs, used to calculate stationary and transient fluid passage times. The vanishing states (that enable immediate actions) are eliminated at the level of f-components when two regularity conditions are satisfied: absence of immediate cycles and deterministic initial behaviour. The operational semantics of iGPEPA is defined on the underlying CTMCs.

PHASE [93, 94, 92] is designed to model non-Markovian systems by implementing phase type distributed [231, 265, 170, 286, 188, 172] action delays. PHASE has sequential, choice and parallel operators. The elementary process is a phase type transition (a pair of action and infinitesimal generator matrix of its phase type delay). The parallel processes are synchronization by actions. The PHASE operational semantics represents phase type distributions through their generating CTMCs. It defines Markovian transitions (expressing exponentially distributed delays) and action transitions (corresponding to the instantaneous executions of actions). PHASE advantageously models and more accurately analyzes performance of non-Markovian systems with phase type distributions. PHASE is applied in the general analysis method for such systems, to obtain the processes translated into a probabilistic model checker for studying quantitative properties of the Markovian approximations.

Process Algebra for LOcated MArkovian agents (PALOMA) [125, 126] describes the systems of populations consisting from the agents distributed over space, where the relative positions of agents influence their interaction, and comprises Markovian Multi-class, Multi-message Markovian Agent Models (M^2MAM). PALOMA can construct formal models of large collective adaptive systems, with the agents distributed over the named locations. Each action in PALOMA is either spontaneous (durational with a rate of the exponential occurrence time when it can emit a broadcast or unicast message of the same type) or induced (immediate with a probability to receive a broadcast or unicast message of the same type). The PALOMA operators over agents (parameterized by locations) are: spontaneous (with broadcast, unicast or without any emission) action prefixing, induced (by broadcast or unicast) action prefixing, choice and parallel composition. PALOMA has a discrete operational semantics, based on the labeled transition systems with delay and probabilistic transitions, from which semi-Markov chains (SMCs) can be extracted. The calculus also has a differential, population, operational semantics, from which population CTMC (pCTMCs) can be obtained, used while deriving ODEs for the mean-field model.

Stochastic Hybrid Communicating Sequential Processes (SHCSP) [242] extends Hybrid Communicating Sequential Processes (HCSP) calculus [304] with probability and stochasticity. In SHCSP, nondeterministic choice is replaced by probabilistic one and ODEs are generalized by stochastic differential equations (SDEs) that describe stochastic continuous evolution, including Brownian motion. In addition to the HCSP operations of null process, assignment, receiving or sending value along channel, sequential composition, alternative statement and repetition, the SHCSP operations include probabilistic choice and SDE-governed evolution, can specify preemption, weights, communication and concurrency, aiming to construct stochastic hybrid processes in a modular way.

Continuous time interleaving SPAs with positive deterministic actions Bio-PEPA with delays (Bio-PEPAd) [81] is an enrichment of Bio-PEPA [95, 96, 97, 129, 146, 218, 219, 217, 166], by adding non-Markovian action delays. The syntax of Bio-PEPAd is inherited from Bio-PEPA and endowed with the action delays functions. Bio-PEPAd has a Start-Termination- (ST-) operational semantics, where the beginning and end of each action execution are taken as separate events, defined by different action delays: exponentially distributed and positively deterministically timed, respectively. Distinguishing the starts and completions of actions is similar to the idea of ST-semantics for GSMPA [66, 64]. The processes of Bio-PEPAd are translated into generalized semi-Markov processes (GSMPs) [67, 5], used in the Delay Stochastic Simulation Algorithm (DSSA) and in Delay Differential Equations (DDEs) extending the deterministic formalism of ODEs to model biological systems with delays.

Table 7 (specifically) classifies the continuous time interleaving SPAs surveyed above (including MTIPP, PEPA and EMPA) and in Section 1 (the calculi sPBC and gsPBC) according to whether the time delays are

Table 7: Classification of the continuous time interleaving stochastic process algebras

Time	Deterministic (multi)actions	Exponential delays	Phase type delays	General delays
Integrated	Non-exist	MTIPP, PA_S, **PEPA**, **sPBC**, MAC, SP, BioNetGen, GPEPA, StoKlaim, SPiM, κ, SBioA, MPC, $PEPA$+II, SBG, SPS, CGF, CPF, BGF, StoCCS, SCLS, **LBS**, Bio-PEPA, SpCLS, SCWC, PAH, TSCLS, FPA, SSPA, **CChS**, FEPA, ProPPA, CARMA, MELA, NBA	$PEPA^{\infty}_{ph}$	PA_{GS}, GPA
	Immediate	MPA, MPA-B, PM-TIPP, SE-TIPP, **EMPA**, BioSpi, IMAC, SBB, nanoκ, SPiCO, Sπ@, $\pi(\mathcal{L})$, XSP, BlenX, **gsPBC**, DSD, iPEPA, iGPEPA, PALOMA	PHASE	NMSPA, **SM-PEPA**
	Positive	Bio-PEPAd	—	—
Orthogonal	Non-exist	MC, MASSPA	CCC	—
	Immediate	STC, $\mathcal{SPA}$, IMC, IML, sCCP, mCCS, MAPA	SB-LOTOS, PTP	SPADES-HS, SPADES, IGSMP, VPSPA, MODEST, SHCSP

associated with (multi)actions (integrated or orthogonal time [99, 166]), the presence of (positive) deterministic or (only) immediate (multi)actions, and the type of stochastic delays (exponentially or phase type, or generally distributed). The names of SPAs with the denotational semantics based on SPNs are printed in bold font.

10.2 Continuous time and non-interleaving semantics

Only a few non-interleaving SPAs were considered among non-Markovian ones [180, 65]. The semantics of all Markovian calculi is interleaving and their action delays have exponential distribution, which is the only continuous probability distribution with memoryless (Markovian) property.

In [69], Generalized Stochastic Process Algebra (GSPA) was introduced. It has a true-concurrent denotational semantics in terms of generalized stochastic event structures (GSESs) with non-Markovian stochastic delays of events. In that paper, no operational semantics or performance evaluation methods for GSPA were presented. In [181], generalized semi-Markov processes (GSMPs) [67, 5] were extracted from GSESs to analyze performance.

In [250, 251, 189], Generalized Stochastic π-calculus (that we call GSπ) with general continuous distributions of activity delays was defined. It has a proved operational semantics with transitions labeled by encodings of their deduction trees. No well-established underlying performance model for this version of GSπ was described.

In [66, 64], Generalized Semi-Markovian Process Algebra (GSMPA) was developed with an ST-operational semantics and non-Markovian action delays. The performance analysis in GSMPA is accomplished via GSMPs.

Again, the first fundamental difference between dtsdPBC and the calculi GSPA, GSπ and GSMPA is that dtsdPBC is based on PBC, whereas GSPA is an extension of simple Process Algebra (PA) from [69], GSπ extends π-calculus [224] and GSMPA is an enrichment of EMPA. Therefore, both GSPA and GSMPA have *prefixing*, *choice* (*alternative* composition), *parallel* composition, *renaming* (*relabeling*) and *hiding* (*abstraction*) operations, but only GSMPA permits *constants*. Unlike dtsdPBC, GSPA has neither iteration or recursion, GSMPA allows only *recursive* definitions, whereas GSπ additionally has operations to specify *mobility*. Note also that GSPA, GSπ and GSMPA do not specify even instantaneous events or activities while dtsdPBC has deterministic multiactions.

The second significant difference is that geometrically distributed or zero delays are associated with process states in dtsdPBC, unlike generally distributed delays assigned to events in GSPA or to activities in GSπ and GSMPA. As a consequence, dtsdPBC has a discrete time operational semantics allowing for concurrent execution of activities in steps. GSPA has no operational semantics while GSπ and GSMPA have continuous time ones. In continuous time semantics, concurrency is simulated by interleaving, since simultaneous occurrence of any two events has zero probability according to the properties of continuous probability distributions. Therefore, interleaving transitions are often annotated with an additional information to keep concurrency data. The transition labels in the operational semantics of GSπ encode the action causality information and allow one to derive the enabling relations and the firing distributions of concurrent transitions from the transition sequences. At the

same time, abstracting from stochastic delays leads to the classical early interleaving semantics of π-calculus [224]. The ST-operational semantics of GSMPA is based on decorated transition systems governed by transition rules with rather complex preconditions. There are two types of transitions: the choice (action beginning) and the termination (action ending) ones. The choice transitions are labeled by weights of single actions chosen for execution while the termination transitions have no labels. Only single actions can begin, but several actions can end in parallel. Thus, the choice transitions happen just sequentially while the termination transitions can happen simultaneously. As a result, the decorated interleaving / step transition systems are obtained. dtsdPBC has an SPN-based denotational semantics. In comparison with event structures, PNs are more expressive and visually tractable formalism, capable of finitely specifying an infinite behaviour. Recursion in GSPA produces infinite GSESs while dtsdPBC has iteration operation with a finite SPN semantics. Identification of infinite GSESs that can be finitely represented in GSPA was left for a future research.

10.3 Discrete time

In [1], a class of compositional DTSPNs with generally distributed discrete time transition delays was proposed, called dts-nets. The denotational semantics of a stochastic extension (that we call stochastic ACP or sACP) of a subset of Algebra of Communicating Processes (ACP) [24] can be constructed via dts-nets. There are two types of transitions in dts-nets: immediate (timeless) ones, with zero delays, and time ones, whose delays are random variables having general discrete distributions. The top-down synthesis of dts-nets consists in the substitution of their transitions by blocks (dts-subnets) corresponding to the sequence, choice, parallelism and iteration operators. It was explained how to calculate the throughput time of dts-nets using the service time (defined as holding time or delay) of their transitions. For this, the notions of service distribution for the transitions and throughput distribution for the building blocks were defined. Since the throughput time of the parallelism block was calculated as the maximal service time for its two constituting transitions, the analogue of the step semantics approach was implemented.

In [209, 210], an SPA called Theory of Communicating Processes with discrete stochastic time (TCP^{dst}) was introduced, later in [207] called Theory of Communicating Processes with discrete real and stochastic time (TCP^{drst}). It has discrete real time (deterministic) delays (including zero delays) and discrete stochastic time delays. The algebra generalizes real time processes to discrete stochastic time ones by applying real time properties to stochastic time and imposing race condition to real time semantics. TCP^{dst} has an interleaving operational semantics in terms of stochastic transition systems. The performance is analyzed via discrete time probabilistic reward graphs which are essentially the reward transition systems with probabilistic states having finite number of outgoing probabilistic transitions and timed states having a single outgoing timed transition. The mentioned graphs can be transformed by unfolding or geometrization into discrete time Markov reward chains (DTMRCs) appropriate for transient or stationary analysis.

The first difference between dtsdPBC and the algebras sACP and TCP^{dst} is that dtsdPBC is based on PBC, but sACP and TCP^{dst} are the extensions of ACP [24]. sACP has taken from ACP only *sequence*, *choice*, *parallelism* and *iteration* operations, whereas dtsdPBC has additionally relabeling, restriction and synchronization ones, inherited from PBC. In TCP^{dst}, besides standard action *prefixing*, *alternative* composition, *parallel* composition, *encapsulation* (similar to *restriction*) and *recursive* variables, there are also *timed delay prefixing*, *dependent delays scope* and the *maximal time progress* operators, which are new both for ACP and dtsdPBC.

The second difference is that dtsdPBC, sACP and TCP^{dst}, all have zero delays, however, discrete time delays in dtsdPBC are zeros or geometrically distributed (being 1 or ∞ as special cases) and associated with process states. The zero delays are possible just in vanishing states while geometrically distributed delays are possible only in tangible states. For each s-tangible (w-tangible) state, the parameter of geometric distribution governing the delay in the state is completely determined by the probabilities (weights) of all stochastic (waiting) multiactions executable from it. In sACP and TCP^{dst}, delays are generally distributed, but they are assigned to transitions in sACP and separated from actions (excepting zero delays) in TCP^{dst}. Moreover, a special attention is given to zero delays in sACP and deterministic delays in TCP^{dst}. In sACP, immediate (timeless) transitions with zero delays serve as source and sink transitions of the dts-subnets corresponding to the choice, parallelism and iteration operators. In TCP^{dst}, zero delays of actions are specified by undelayable action prefixes while positive deterministic delays of processes are specified with timed delay prefixes. Neither formal syntax nor operational semantics for sACP are defined and it is not explained how to derive Markov chains from the algebraic expressions or the corresponding dts-nets to analyze performance. It is not stated explicitly, which type of semantics (interleaving or step) is accommodated in sACP. In spite of the discrete time approach, operational semantics of TCP^{dst} is still interleaving, unlike that of dtsdPBC. In addition, no denotational semantics was defined for TCP^{dst}.

Let us briefly describe **other SPAs with discrete time and interleaving semantics**.

Discrete time interleaving SPAs without immediate actions Weighted Synchronous Calculus of Communicating Systems (WSCCS) [283, 284, 221] is an extension of SCCS [223] with weights. WSCCS is a calculus of probabilistic processes, where probabilities are not directly assigned to the choice operation. Instead, weights are interpreted as the probabilistic specifications using the relative frequency concept and corresponding equality criterium. There exist special weights expressing priorities. The weights and actions in WSCCS are separated. The actions in WSCCS form an Abelian group with the identity action and inverse of each action. The WSCCS operators are: empty (null) process, (action) prefix, weighted choice (with a finite number of weights), (synchronous) parallel composition, permit (only actions in a set), prioritized parts (taking only), relabeling and recursion (or recursive definition). The discrete model of time is applied in WSCCS and processes are executed at time ticks. Either weighted choice or named action execution occur at each tick, resulting in the interleaving semantics and *stratified* model [138]. WSCCS is also used to calculate upper bounds on the performance of mutually affected systems, in which action delays are specified symbolically as random values with general discrete phase type distributions [231, 265, 170, 286, 188, 172].

Discrete time variant (that we call dsCCP) of stochastic Concurrent Constraint Programming (sCCP) was proposed in [50]. The calculus sCCP is constructed to model and analyze biological systems. In sCCP, communication is asynchronous, species are described by variables, reactions are seen as constraints on the variables and rates are defined using functions. The operations of sCCP include ask prefix (with rate), tell prefix (with rate), choice, empty process, procedure call (with rate), hiding and parallel composition. The analysis consists in stochastic simulation, as well as in translation into a probabilistic verification tool, ODEs and hybrid systems. A discrete time version (with rates interpreted as weights, used to calculate probabilities) of the sCCP operational semantic results in the algebra dsCCP, with DTMCs being the performance analysis model.

Discrete time interleaving SPAs with immediate actions Interactive Probabilistic Chains (IPC) [102, 154] calculus unifies in a single probabilistic discrete time model the capabilities of compositional modeling, functional verification and performance analysis (through translation into DTMCs) for industrial systems and networks on chips. The operators of IPC are termination, sequential composition, probabilistic choice (with a set of probabilities), nondeterministic choice, parallel composition (with synchronization set), hiding (of actions set), process call and (possibly recursive) process definition. Being a discrete time analogue of the algebra IMC [158, 159], IPC has an interleaving operational semantics on the unification of labeled transition systems and DTMCs. The transitions in that semantics are either probabilistic (that occur with particular probabilities during exactly one discrete time tick) or interactive (corresponding to the instantaneous execution of some actions, possibly invisible). The performance model of IPC is DTMCs, obtained from interactive probabilistic chains using schedulers to resolve a nondeterministic choice by replacing it with with a probabilistic choice.

The three SPAs are rather specific: unlike standard approach, weights in WSCCS, rates (weights) in dsCCP and probabilities in IPC are not associated with actions. In dsCCP, probabilities are calculated using rates (weights) that are assigned to operations. In IPC, actions are executed instantaneously while probabilistic choices take one unit time. In the common SPAs with the *integrated time* concept, the time parameters are combined with actions into pairs called activities. In dsCCP and IPC, the *orthogonal time* concept is applied, where time progress is separated from actions, assumed to be immediate and to specify logical progress [99, 166].

Table 8 summarizes the SPAs comparison above and that from Section 1 (the calculi sPBC, gsPBC, dtsPBC and dtsiPBC), by (generally) classifying the SPAs according to the concept of time, the presence of (positive or arbitrary) deterministic or (only) immediate (multi)actions, and the operational semantics type. The names of SPAs with the denotational semantics based on SPNs are printed in bold font. The underlying stochastic process (if defined) for each presented SPA is specified in parentheses near its name.

11 Discussion

Let us now discuss which advantages has dtsdPBC in comparison with the SPAs described in Section 10.

11.1 Analytical solution

An important aspect is the analytical tractability of the underlying stochastic process, used for performance evaluation in SPAs. The underlying CTMCs in MTIPP and PEPA, as well as SMCs in EMPA, are treated analytically, but these continuous time SPAs have interleaving semantics. GSPA, GSπ and GSMPA are the continuous time models, for which a non-interleaving semantics is constructed, but for the underlying GSMPs in GSPA and GSMPA, only simulation and numerical methods are applied, whereas no performance model for GSπ is defined. sACP and TCP^{dst} are the discrete time models with the associated analytical methods for

Table 8: Classification of stochastic process algebras

Time	Deterministic (multi)actions	Interleaving semantics	Non-interleaving semantics
Continuous	Non-exist	MTIPP (CTMC), **PEPA** (CTMP), **sPBC** (CTMC)	GSPA (GSMP), GSπ, GSMPA (GSMP)
	Immediate	**EMPA** (SMC, CTMC), **gsPBC** (SMC)	—
	Positive	Bio-PEPAd (GSMP)	—
Discrete	Non-exist	WSCCS (DTMC), dsCCP (DTMC)	**dtsPBC** (DTMC)
	Immediate	IPC (DTMC)	**dtsiPBC** (SMC, DTMC)
	Arbitrary	TCP^{dst} (DTMRC)	**sACP**, **dtsdPBC** (SMC, DTMC)

the throughput calculation in sACP or for the performance evaluation based on the underlying DTMRCs in TCP^{dst}, but both models have interleaving semantics. dtsdPBC is a discrete time model with a non-interleaving semantics, where analytical methods are applied to the underlying SMCs. Hence, if an interleaving model is appropriate as a framework for the analytical solution towards performance evaluation then one has a choice between the continuous time SPAs MTIPP, PEPA, EMPA and the discrete time ones sACP, TCP^{dst}. Otherwise, if one needs a non-interleaving model with the associated analytical methods for performance evaluation and the discrete time approach is feasible then dtsdPBC is the right choice.

The existence of an analytical solution also permits to interpret quantitative values (rates, probabilities, weights etc.) from the system specifications as parameters, which can be adjusted to optimize the system performance, like in dtsPBC, dtsiPBC and dtsdPBC. The DTMCs whose transition probabilities are parameters were introduced in [107]. The parameters can also be adjusted in parametric probabilistic transition systems (PTSs) [193], i.e. in the DTMCs whose transition probabilities may be real-valued parameters. Parametric CTMCs with the transition rates treated as parameters were investigated in [149]. Parametric probabilistic timed automata (PTAs) were defined in [86]. Parametric DTMCs with the transition probabilities being polynomials over real-valued parameters were investigated in [148]. In [174], a new method of computing the reachability probabilities was proposed for parametric DTMCs whose state change probabilities are the fractions of polynomials over the set of parameters. The parameter value synthesis problem was studied in [113] for parametric interval DTMCs (IDTMCs), in which the parameters are the borders of the transition probability intervals. In [255], a new parameter synthesis technique called lifting was proposed for three parametric models: stochastic games (SGs), Markov decision processes (MDPs) and DTMCs. Parametric verification for concurrent systems modeled by parametric versions of timed automata (TAs), interval (DT)MCs (IMCs), PNs and logic Action-Restricted CTL (ARCTL) was surveyed in [3]. For parametric verification with logic PCTL in [104], uncertain MDPs (UMDPs) were applied whose parameters may be either controlled (as in the standard parametric MDPs) or uncontrolled (being random values with the probability distributions), aiming to specify uncertainty of the transition probabilities and reward functions.

On the other hand, no parameters in formulas of SPAs were considered in the literature so far. In dtsdPBC we can easily construct examples with more parameters than we did in our case study. The performance indices will be then interpreted as functions of several variables. The advantage of our approach is that, unlike of the method from [193] and other works, we should not impose to the parameters any special conditions needed to guarantee that the real values, interpreted as the transition probabilities, always lie in the interval $[0;1]$. To be convinced of this fact, just remember that, as we have demonstrated, the positive probability functions PF, PT, PM, PM^*, $PM^\diamond$ define probability distributions, hence, they always return values belonging to $(0;1]$ for any probability parameters from $(0;1)$ and weight parameters from $\mathbb{R}_{>0}$. In addition, the transition constraints (their probabilities, rates and guards), calculated using the parameters, in our case should not always be polynomials over variables-parameters, as often required in the mentioned papers, but they may also be fractions of polynomials, like in our case study.

11.2 Concurrency interpretation

One can see that the stochastic process calculi proposed in the literature are based on interleaving, as a rule, and parallelism is simulated by synchronous or asynchronous execution. As a semantic domain, the interleaving formalism of transition systems is often used. However, to properly support intuition of the behaviour of concurrent and distributed systems, their semantics should treat parallelism as a primitive concept that cannot be reduced to nondeterminism. Moreover, in interleaving semantics, some important properties of these systems

cannot be expressed, such as simultaneous occurrence of concurrent transitions [109] or local deadlock in the spatially distributed processes [229]. Therefore, investigation of stochastic extensions for more expressive and powerful algebraic calculi is an important issue. The development of step or "true concurrency" (such that parallelism is considered as a causal independence) SPAs is an interesting and nontrivial problem, which has attracted special attention last years. Nevertheless, not so many formal stochastic models of parallel systems were defined whose underlying stochastic processes are based on DTMCs. As mentioned in [127], such models are more difficult to analyze, since several events can occur simultaneously in discrete time systems (the models have a step semantics) and the probability of a set of events cannot be easily related to the probability of the single ones. Therefore, parallel executions of actions are often not considered also in the discrete time SPAs, such as TCP^{dst}, whose underlying stochastic process is DTMCs with rewards (DTMRCs). As observed in [169], even for stochastic models with generally distributed time delays, some restrictions on the concurrency degree were imposed to simplify their analysis techniques. In particular, the enabling restriction requires that no two generally distributed transitions are enabled in any reachable marking. Hence, their activity periods do not intersect and no two such transitions can fire simultaneously, this results in interleaving semantics of the model.

Stochastic models with discrete time and step semantics have the following important advantage over those having just an interleaving semantics. The underlying Markov chains of parallel stochastically timed processes have the additional transitions corresponding to the simultaneous execution of concurrent (i.e. non-synchronized) activities. The transitions of that kind allow one to bypass a lot of intermediate states, which otherwise should be visited when interleaving semantics is accommodated. When step semantics is used, the intermediate states can also be visited with some probability (this is an advantage, since some alternative system's behaviour may start from these states), but this probability is not greater than the corresponding one in case of interleaving semantics. While in interleaving semantics, only the empty or singleton (multi)sets of activities can be executed, in step semantics, generally, the (multi)sets of activities with more than one element can be executed as well. Hence, in step semantics, there are more variants of execution from each state than in the interleaving case and the executions probabilities, whose sum should be equal to 1, are distributed among more possibilities. Therefore, the systems with parallel stochastic processes usually have smaller average run-through. In case the underlying Markov chains of the processes are ergodic, they will generally take less discrete time units to stabilize the behaviour, since their TPMs will be usually denser because of additional non-zero elements outside the main diagonal. Hence, both the first passage-time performance indices based on the transient probabilities and the steady-state performance indices based on the stationary probabilities can be potentially computed quicker, resulting in mostly faster quantitative analysis of the systems. On the other hand, step semantics, induced by simultaneous firing several transitions at each step, is natural for Petri nets and allows one to exploit full power of the model. Therefore, it is important to respect the probabilities of parallel executions of activities in discrete time SPAs, especially in those with a Petri net denotational semantics.

The speed (rate) of converging the transient PMF for a DTMC to its stationary PMF was investigated in [188] (the quantitative estimate via the TPM's second eigenvalue, ordered by the absolute value descendance) and in [128] (the equivalent qualitative conditions in terms of geometric ergodicity, i.e. exponentially fast approaching the stationary distribution with time progress).

11.3 Application area

From the application viewpoint, one considers what kind of systems are more appropriate to be modeled and analyzed within SPAs. MTIPP and PEPA are well-suited for the interleaving continuous time systems such that the activity rates or the average sojourn time in the states are known in advance and exponential distribution approximates well the activity delay distributions, whereas EMPA can be used to model the mentioned systems with the activity delays of different duration order or the extended systems, in which purely probabilistic choices or urgent activities must be implemented. GSPA and GSMPA fit well for modeling the continuous time systems with a capability to keep the activity causality information, and with the known activity delay distributions, which cannot be approximated accurately by exponential distribution, while GSπ can additionally model mobility in such systems. TCP^{dst} is a good choice for interleaving discrete time systems with deterministic (fixed) and generalized stochastic delays, whereas sACP is capable to model non-interleaving systems as well, but it offers not enough performance analysis methods.

dtsdPBC is consistent for the step discrete time systems such that the independent execution probabilities of activities are known and geometrical distribution approximates well the state residence time distributions. These include Dirac distribution of the positive deterministic sojourn time, which is then splitted into one time units and allocated with the consecutive process states. In addition, dtsdPBC can model the mentioned systems featuring very scattered activity delays, or even more complex systems with instantaneous probabilistic choice or urgency. Hence, dtsdPBC can be taken as a non-interleaving discrete time counterpart of TCP^{dst}.

Table 9: Classification of stochastic Petri nets

Time	Stochastic transitions	Stochastic and immediate transitions	Stochastic and deterministic transitions
Continuous	(L)CTSPNs (PEPA, sPBC)	(L)GSPNs (EMPA, gsPBC)	DSPNs (—)
Discrete	**(L)WDTSPNs**, **(L)DTSPNs** (—), (**dtsPBC**)	spTPNs, **(L)DTSIPNs** (—), (**dtsiPBC**)	DTDSPNs, **(L)DTSDPNs** (—), (**dtsdPBC**)

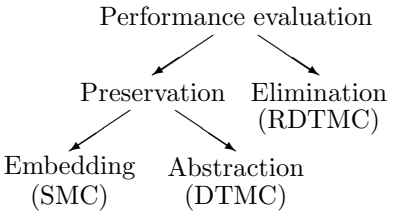

Figure 61: Performance evaluation methods in dtsdPBC

11.4 Advantages of our approach

Table 9 contains a classification of the (labeled) SPNs classes mentioned in this paper, according to the model of time (continuous or discrete) and presence of (besides stochastic) immediate or deterministic (i.e. immediate and waiting) transitions. We consider (labeled) CTSPNs [227, 213, 75, 15, 22, 16], GSPNs [87, 90, 213, 91, 214, 76, 15, 22, 16], WDTSPNs [78], DTSPNs [226, 228, 266, 267, 268, 269, 270], spTPNs [71], DTSIPNs [276, 277, 278, 279, 280], DTDSPNs [309, 305, 306] and DTSDPNs [271, 272, 273, 274, 275]. For completeness, we also consider a continuous time model of deterministic stochastic Petri nets (DSPNs) [215, 216] with stochastic (exponential) and deterministic transitions. In the parentheses under the SPNs classes, the names of the SPAs discussed here are written whose denotational semantics is based on the corresponding types of SPNs. For example, denotational semantics of PEPA is constructed using (labeled) CTSPNs while that of dtsiPBC is defined via dtsi-boxes, a special subclass of LDTSIPNs. The names of the SPNs and SPAs, defined by us, are printed in bold font. In the table, all the SPNs with continuous time have interleaving semantics whereas those with discrete time have non-interleaving (step) semantics.

Thus, the main advantages of dtsdPBC are the flexible multiaction labels, stochastic and deterministic multiactions, powerful operations, as well as a step operational and a Petri net denotational semantics allowing for concurrent execution of activities (transitions), together with an ability for analytical and parametric performance evaluation. The uniqueness of our approach consists in applying a parallel semantics for the process expressions and preserving the concurrency level in the extracted performance models (SMC, DTMC and RDTMC) through their state changes corresponding to the simultaneous executions.

12 Conclusion

In this paper, we have proposed a discrete time stochastic extension dtsdPBC of PBC, enriched with deterministic multiactions. The calculus has a parallel step operational semantics, based on labeled probabilistic transition systems and a denotational semantics in terms of a subclass of LDTSDPNs. A technique of performance evaluation in the framework of the calculus has been presented (*embedding*) that explores the corresponding stochastic process, which is a semi-Markov chain (SMC). In such an SMC, the sojourn time in every tangible state is geometrically distributed (with the special cases of being one or infinity) while the sojourn time in every vanishing state is zero. It has been proved that the underlying discrete time Markov chain (DTMC) or its reduction (RDTMC) by eliminating vanishing states may alternatively and suitably be studied for that purpose (*abstraction* and *elimination*, respectively). Since vanishing states are preserved by both the *embedding* and *abstraction*, the latter can be seen as an alternative (to the former) *preservation* method [91]. We have also established coincidence of the reduced SMC (RSMC) and RDTMC. In Figure 61, a classification of the techniques for performance analysis within dtsdPBC is presented.

Further, step stochastic bisimulation equivalence of process expressions has been defined, used to establish the consistency of the operational and denotational semantics, and its interrelations with other equivalences of the calculus have been investigated. We have explained how to reduce transition systems and Markov chains (SMCs, DTMCs and RDTMCs) by building their quotients with respect to the introduced equivalence. We

have studied an effect of the quotienting to extraction, embedding and reduction, in terms of the transition probability matrices (TPMs) of the quotient DTMCs, EDTMCs and RDTMCs. We have demonstrated that for DTMCs of the process expressions, the quotienting is permutable (commute) with both extraction and reduction, whereas an additional embedding of the quotient embedded DTMC is needed to coincide with the embedded quotient DTMC. Thus, making extraction before the quotienting permits to start reasoning from the Markov chain level. Applying reduction before the quotienting simplifies quantitative analysis in case of many non-equivalent vanishing states. The quotienting before embedding requires less computations. We have proved that the mentioned equivalence guarantees identity of the stationary behaviour and the sojourn time properties, and thus preserves performance measures. The theory presented has been illustrated with an extensive series of examples, among which is the travel system application demonstrating performance analysis within dtsdPBC. A case study of a generalization of the shared memory system with maintenance, by allowing for variable probabilities in its specification, has been presented. The case study is an example of modeling, performance evaluation and performance preserving reduction in the calculus. We have also determined the advantages of dtsdPBC by comparing it with other SPAs. We have discussed the SPAs approaches to the analytical solution, concurrency interpretation and application area.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, whereas this is not possible in classical Markov chains. As argued in [285], (stochastic) PNs represent the systems structure more concisely and can be an intermediate formalism for their more intuitive translation into Markov chains. Second, algebraic formulas represent processes in a more compact way than PNs and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature demonstrates that it is not straightforward to construct such PNs directly from the system specifications. dtsdPBC is well suited for the discrete time applications, whose discrete states change with a global time tick, such as business processes, neural and transportation networks, computer and communication systems, timed web services [291], as well as for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis, such as genetic regulatory and cellular signalling networks (featuring maximal parallelism) in biology [106, 48, 21] (remember that, in step semantics, we have additional transitions due to concurrent executions). In [134], biological networks were jointly modeled by (standard, qualitative) PNs, CTSPNs and continuous PNs (CPNs), to demonstrate their complementarity that makes necessary adding deterministic time to stochastic models, as well as combining stochastic and continuous (deterministic) aspects into one model (such as stochastic rates of reactions and continuous amounts of species). dtsdPBC is also capable to model and analyze parallel systems with fixed durations of the typical activities (loading, processing, transfer, repair, low-level events, message delivery) and stochastic durations of the randomly occurring activities (arrival, departure, failure, packet loss, message collision), including industrial, manufacturing, queueing, computing and network systems.

In particular, we have adapted for dtsdPBC all examples of the expressions, ct-boxes and inferences by the transition rules from tPBC [183]. Whereas the examples from that paper explore only some selected state-transition sequences (paths), we always construct the complete transition systems of the expressions. We have observed that in our framework we have no difficulties like those in tPBC, which have forced to allow illegal transition sequences. In tPBC, the increasing timers are associated with the overlines and underlines of multiactions and suggest the ages of the corresponding markings in the respective boxes. In dtsdPBC, the decreasing (up to the value 1) timers are associated with the enabled waiting multiactions and specify their remaining times to execute (RTEs), like the timers of the enabled deterministic transitions in DTDSPNs from [311, 312, 310]. Besides such a PNs intuition, making difference between markings (overlines and underlines) and timers of (waiting) multiactions offers us more syntactical flexibility to express their progress in time. The decreasing timers allow us to avoid problems with infinitely growing timer values in the deadlocked and final (absorbing) states. Each decreasing timer should start with a particular value that cannot be suggested by the current marking, but such an initial value is the delay of the waiting multiaction the timer is associated with.

It is known that the attempts to combine time restrictions, parallelism and compositionality usually lead to many technical difficulties, so that the formal models possessing all the mentioned properties have almost not been proposed in the literature, in spite of the investigations in the related areas (for example, discrete time, generally distributed delays, non-interleaving functional semantics in the SPA framework). To solve the mentioned problem, some new (not existing in dtsiPBC) notions and constructions have been introduced in dtsdPBC, such as deterministic multiactions, decreasing timers of waiting multiactions, enabledness of activities, saturation with the timer values, timers discarding and decreasing operations, extended Can and Now functions, s-tangible and w-tangible dynamic expressions and states, inaction and action rules respecting waiting multiactions, empty moves, reachability of dynamic expressions, transition systems with 3 types of states and 4 types of transitions (unlike 2 types of states and 3 types of transitions in dtsiPBC). Thus, the main advantages

of dtsdPBC are the flexible multiaction labels, deterministic multiactions, powerful operations, as well as a step operational and a Petri net denotational semantics allowing for parallel executions of activities (firings of the PNs transitions), together with an ability for analytical and parametric performance evaluation. The uniqueness of our approach consists in applying a parallel semantics for the process expressions and preserving the concurrency level in the extracted performance models (SMC, DTMC and RDTMC) through their state changes corresponding to the simultaneous executions.

Future work could consist in constructing a congruence relation for dtsdPBC, i.e. the equivalence that withstands application of all operations of the algebra. The first possible candidate is a stronger version of step stochastic bisimulation equivalence, defined via transition systems equipped with two extra transitions skip and redo, like those from sPBC [201]. Moreover, recursion operation could be added to dtsdPBC to increase further specification power of the algebra. It would be very interesting to implement the class of DTSDPNs, to be able to specify them and then model their behaviour by constructing the reahability graphs. Note that even DTSPNs of M.K. Molloy [226, 228] have never been implemented. Mostly interleaving and continuous time variants of stochastic or timed PNs have been implemented so far.

We also plan to extend dtsdPBC with discrete phase type action delays that are described by arbitrary finite absorbing DTMCs and include both geometric and non-Markovian (such as deterministic) ones as special cases. Discrete phase type probability distributions can with any required precision approximate general discrete distributions over the positive integers basis and are closed under minimum (alternative composition, conflict), maximum (parallel composition, parallelism), finite convolution (sequential composition, precedence), finite weighted and infinite geometric summations [231, 265, 170, 286, 188, 172]. Some known SPNs with with phase type transition delays are: SPNs with phase-type distributed transition times (PTDTT-SPNs) [105] and phased delay PNs (PDPNs) [177] (the both SPN classes with discrete and continuous time), as well as defective discrete phase SPNs (DDP-SPNs) [89], discrete deterministic and stochastic PNs (DDSPNs) [307, 308] and non-Markovian SPNs (NMSPNs) [171] (the three SPN classes with discrete time). Among all those SPN classes, only NMSPNs have a non-interleaving transition firing semantics, but it is too complex and technical. Some existing SPAs with phase type action delays are: a modification of PA_{GS} [179], $PEPA^{\infty}_{ph}$ [122], SB-LOTOS [160], PTP [300], PHASE [93, 94, 92] and CCC [254, 42] (the six SPAs with continuous time), as well as a variant of WSCCS [284] (with discrete time). All those SPAs have only interleaving operational and no SPN-based denotational semantics. Those interleaving phase SPAs are rather specialized or theoretically-oriented and hardly applicable in practice or with restricted specification capabilities. In detail, PA_{GS} is rather theoretical, $PEPA^{\infty}_{ph}$ describes very special subclasses of non-Markovian systems, SB-LOTOS separates actions and delays, WSCCS has technically complex and non-sufficiently intuitive syntax and semantics, PTP has cooperating processes that cannot be synchronized by the shared activities, PHASE offers just a few operators, whereas CCC does not have actions, synchronization and recursion. Unlike PA_{GS}, $PEPA^{\infty}_{ph}$, SB-LOTOS, PTP, PHASE and CCC with continuous time, WSCCS adapts a discrete time model, but the semantics of WSCCS is still interleaving. Thus, it is actual to construct a discrete time SPA with phase type delays and non-interleaving semantics: operational one (on the labeled transition systems with parallel executions of activities) and denotational one (on the SPNs with phase delays and parallel firings of transitions).

References

[1] van der Aalst W.M.P., van Hee K.M., Reijers H.A. *Analysis of discrete-time stochastic Petri nets. Statistica Neerlandica* **54(2)**, p. 237–255, The Netherlands Society for Statistics and Operations Research (VVSOR), Wiley Blackwell Publishers, July 2000, `http://tmitwww.tm.tue.nl/staff/hreijers/H.A. Reijers Bestanden/Statistica.pdf`, `http://onlinelibrary.wiley.com/doi/epdf/10.1111/1467-9574.00139`.

[2] Aldini A., Bernardo M., Corradini F. *A process algebraic approach to software architecture design.* 304 p., Springer, London, UK, 2010 (ISBN 978-1-84800-222-7).

[3] André É., Knapik M., Lime D., Penczek W., Petrucci L. *Parametric verification: an introduction. Lecture Notes in Computer Science* **11790**, p. 64–100, 2019.

[4] D'Argenio P.R. *Algebras and automata for timed and stochastic systems. Ph.D. thesis*, *IPA Dissertation Series* **1999-10**, *CTIT PhD-Thesis Series* **99-25**, 342 p., Department of Computer Science, University of Twente, Enschede, The Netherlands, Print Partners Imskamp, Enschede, The Netherlands, November 1999, `http://www.cs.famaf.unc.edu.ar/~dargenio/Publications/dissertation/dissertation.pdf`.

[5] D'Argenio P.R., Katoen J.-P. *A theory of stochastic systems. Part I: Stochastic automata. Information and Computation* **203(1)**, p. 1–38, November 2005.

[6] D'Argenio P.R., Katoen J.-P. *A theory of stochastic systems. Part II: Process algebra. Information and Computation* **203(1)**, p. 39–74, November 2005.

[7] D'Argenio P.R., Katoen J.-P., Brinksma E. *An algebraic approach to the specification of stochastic systems (extended abstract). Proceedings of the IFIP Working Conference on Programming Concepts and Methods - 98 (PROCOMET'98)*, p. 126–147 (D. Gries, W.-P. de Roever, eds.), Shelter Island, New York, USA, Chapman & Hall, London, UK, 1998, `http://cs.famaf.unc.edu.ar/~dargenio/Publications/papers/procomet98.ps.gz`.

[8] D'Argenio P.R., Katoen J.-P., Brinksma E. *A compositional approach to generalised semi-Markov processes. Proceedings of* 4^{th} *International Workshop on Discrete Event Systems - 98 (WODES'98)*, p. 391–397 (A. Guia, M. Spathopoulos, R. Smedinga, eds.), Cagliary, Italy, IEEE Computer Society Press, London, UK, 1998, `http://cs.famaf.unc.edu.ar/~dargenio/Publications/papers/wodes98.ps.gz`.

[9] Argent-Katwala A., Bradley J.T., Dingle N.J. *Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models. Proceedings of* 4^{th} *International Workshop on Software and Performance - 04 (WOSP'04)*, Redwood Shores, California, USA, January 2004, p. 49–58, ACM Press, 2004.

[10] Argent-Katwala A., Bradley J.T., Clark A., Gilmore S. *Location-aware quality of service measurements for service-level agreements. Lecture Notes in Computer Science* **4912**, p. 222–239, 2008.

[11] Autant C., Schnoebelen Ph. *Place bisimulations in Petri nets. Lecture Notes in Computer Science* **616**, p. 45–61, June 1992.

[12] Baier C. *Polynomial time algorithms for testing probabilistic bisimulation and simulation. Lecture Notes in Computer Science* **1102**, p. 50–61, 1996, `http://www.inf.tu-dresden.de/content/institutes/thi/algi/publikationen/texte/27_00_old.pdf`.

[13] Baier C., Dubslaff C., Funke F., Jantsch S., Majumdar R., Piribauer J. *From verification to causality-based explications. Proceedings of* 48^{th} *International Colloquium on Automata, Languages, and Programming - 21 (ICALP'21)* (N. Bansal, E. Merelli, J. Worrell, eds.), 2021, *Leibniz International Proceedings in Informatics (LIPIcs)*, p. 1:1–1:20 (article 1), Dagstuhl Publishing, Leibniz-Zentrum für Informatik, Schloss Dagstuhl, Germany, 2021.

[14] Baier C., Engelen B., Majster-Cederbaum M. *Deciding bisimilarity and similarity for probabilistic processes. Journal of Computer and System Sciences* **60(1)**, p. 187–231, Elsevier, February 2000.

[15] Balbo G. *Introduction to stochastic Petri nets. Lecture Notes in Computer Science* **2090**, p. 84–155, 2001.

[16] Balbo G. *Introduction to generalized stochastic Petri nets. Lecture Notes in Computer Science* **4486**, p. 83–131, 2007.

[17] Barbuti R., Caravagna G., Maggiolo-Schettini A., Milazzo P., Pardini G. *The calculus of looping sequences. Lecture Notes in Computer Science* **5016**, p. 387–423, 2008.

[18] Barbuti R., Maggiolo-Schettini A., Milazzo P., Pardini G. *Spatial calculus of looping sequences. Proceedings of International Workshop From Biology to Concurrency and Back - 08 (FBTC'08)*, 2008, *Electronic Notes in Theoretical Computer Science* **229(1)**, p. 21–39, 2009.

[19] Barbuti R., Maggiolo-Schettini A., Milazzo P., Pardini G. *Spatial calculus of looping sequences. Theoretical Computer Science* **412**, p. 5976–6001, 2011.

[20] Barbuti R., Maggiolo-Schettini A., Milazzo P., Tiberi P., Troina P. *Stochastic calculus of looping sequences for the modelling and simulation of cellular pathways. Lecture Notes in Computer Science* **5121** p. 86–113, 2008.

[21] Bartocci E., Lió P. *Computational modeling, formal analysis, and tools for systems biology. PLoS Computational Biology* **12(1)**, p. 1–22 (e1004591), Public Library of Science, Cambridge, UK, January 2016.

[22] Bause F., Kritzinger P.S. *Stochastic Petri nets: an introduction to the theory.* Vieweg Verlag, 2^{nd} edition, 218 p., 2002, `http://ls4-www.cs.tu-dortmund.de/cms/de/home/bause/bause_kritzinger_spn_book_print.pdf`.

[23] Bérard B., Cassez F., Haddad S., Lime D., Roux O.H. *Comparison of different semantics for time Petri nets. Lecture Notes in Computer Science* **3707**, p. 293–307, 2005.

[24] Bergstra J.A., Klop J.W. *Algebra of communicating processes with abstraction. Theoretical Computer Science* **37**, p. 77–121, 1985.

[25] Bernardo M. *Theory and application of extended Markovian process algebra. Ph.D. thesis*, 276 p., University of Bologna, Italy, February 1999, `http://www.sti.uniurb.it/bernardo/documents/phdthesis.pdf`.

[26] Bernardo M. *A survey of Markovian behavioral equivalences. Formal Methods for Performance Evaluation* (M. Bernardo, J. Hillston, eds.), *Lecture Notes in Computer Science* **4486**, p. 180–219, June 2007, `http://www.sti.uniurb.it/bernardo/documents/sfm07pe.tuto.pdf`.

[27] Bernardo M. *Non-bisimulation-based Markovian behavioral equivalences. Journal of Logic and Algebraic Programming* **72**, p. 3–49, May 2007, `http://www.sti.uniurb.it/bernardo/documents/jlap72.pdf`.

[28] Bernardo M. *On the tradeoff between compositionality and exactness in weak bisimilarity for integrated-time Markovian process calculi. Theoretical Computer Science* **563**, p. 99–143, January 2015, `http://www.sti.uniurb.it/bernardo/documents/tcs563.pdf`.

[29] Bernardo M. *ULTraS at work: compositionality metaresults for bisimulation and trace semantics. Journal of Logical and Algebraic Methods in Programming* **94**, p. 150–182, January 2018, `http://www.sti.uniurb.it/bernardo/documents/jlamp94.pdf`.

[30] Bernardo M., Botta S. *Modal logic characterization of Markovian testing and trace equivalences. Proceedings of* 1^{st} *International Workshop on Logic, Models and Computer Science - 06 (LMCS'06)* (F. Corradini, C. Toffalori, eds.), Camerino, Italy, April 2006, *Electronic Notes in Theoretical Computer Science* **169**, p. 7–18, 2006, `http://www.sti.uniurb.it/bernardo/documents/lmcs2006.pdf`.

[31] Bernardo M., Botta S. *A survey of modal logics characterizing behavioural equivalences for non-deterministic and stochastic systems. Mathematical Structures in Computer Science* **18**, p. 29–55, Cambridge University Press, Cambridge, UK, February 2008, `http://www.sti.uniurb.it/bernardo/documents/mscs18.pdf`.

[32] Bernardo M., Bravetti M. *Reward based congruences: can we aggregate more? Lecture Notes in Computer Science* **2165**, p. 136–151, 2001, `http://www.cs.unibo.it/~bravetti/papers/papm01b.ps`.

[33] Bernardo M., Corradini F., Tesei L. *Timed process calculi with deterministic or stochastic delays: commuting between durational and durationless actions. Theoretical Computer Science* **629**, p. 2–39, May 2016, `http://www.sti.uniurb.it/bernardo/documents/tcs629.pdf`.

[34] Bernardo M., Donatiello L., Gorrieri R. *Modeling and analyzing concurrent systems with MPA. Proceedings of* 2^{nd} *International Workshop on Process Algebra and Performance Modelling - 94 (PAPM'94)* (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, July 1994, *Arbeitsberichte des IMMD* **27(4)**, p. 71–88, Universität Erlangen-Nürnberg, Germany, 1994, `http://www.sti.uniurb.it/bernardo/documents/papm1994.pdf`.

[35] Bernardo M., Donatiello L., Gorrieri R. *A formal approach to the integration of performance aspects in the modeling and analysis of concurrent systems. Information and Computation* **144(2)**, p. 83–154, August 1998, `http://www.sti.uniurb.it/bernardo/documents/ic144.pdf`.

[36] Bernardo M., Gorrieri R. *A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theoretical Computer Science* **202**, p. 1–54, July 1998, `http://www.sti.uniurb.it/bernardo/documents/tcs202.pdf`.

[37] Bernardo M., De Nicola R., Loreti M. *A uniform framework for modeling nondeterministic, probabilistic, stochastic, or mixed processes and their behavioral equivalences. Information and Computation* **225**, p. 29–82, April 2013, `http://www.sti.uniurb.it/bernardo/documents/ic225.pdf`.

[38] Bernardo M., Tesei L. *Encoding timed models as uniform labeled transition systems. Proceedings of* 10^{th} *European Performance Engineering Workshop - 13 (EPEW'13)* (M.S. Balsamo, W.J. Knottenbelt, A. Marin, eds.), Venice, Italy, September 2013, *Lecture Notes in Computer Science* **8168**, p. 104–118, 2013, `http://www.sti.uniurb.it/bernardo/documents/epew2013.pdf`.

[39] Best E., Devillers R., Hall J.G. *The box calculus: a new causal algebra with multi-label communication. Lecture Notes in Computer Science* **609**, p. 21–69, 1992.

[40] Best E., Devillers R., Koutny M. *Petri net algebra. EATCS Monographs on Theoretical Computer Science*, 378 p., Springer Verlag, 2001.

[41] Best E., Koutny M. *A refined view of the box algebra. Lecture Notes in Computer Science* **935**, p. 1–20, 1995, `http://parsys.informatik.uni-oldenburg.de/~best/publications/pn95.ps.gz`.

[42] Bies A., Hermanns H., Köhl M.A., Schmidt A. *Matching distributions under structural constraints. Lecture Notes in Computer Science* **14287**, p. 221–237, 2023.

[43] Bioglio L., Dezani-Ciancaglini M., Giannini P., Troina A. *Typed stochastic semantics for the calculus of looping sequences. Theoretical Computer Science* **431**, p. 165–180, May 2012.

[44] Blinov M.L., Faeder J.R., Goldstein B., Hlavacek W.S. *BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics* **20(17)**, p. 3289–3291, Oxford University Press, Oxford, UK, November 2004.

[45] Bohnenkamp H.C., D'Argenio P.R., Hermanns H., Katoen J.-P. *MODEST: a compositional modeling formalism for hard and softly timed systems. IEEE Transactions on Software Engineering* **32(10)**, p. 812–830, IEEE Computer Society Press, October 2006.

[46] Bohnenkamp H.C., Haverkort B.R. *Semi-numerical solution of stochastic process algebra models. Lecture Notes in Computer Science* **1601**, p. 228–243, 1999.

[47] Bolognesi T., Lucidi F., Trigila S. *From timed Petri nets to timed LOTOS. Proceedings of the IFIP WG 6.1* 10^{th} *International Symposium on Protocol Specification, Testing and Verification - 90*, Ottawa, Canada, p. 395–408, North-Holland, Amsterdam, The Netherlands, 1990.

[48] Bonzanni N., Feenstra K.A., Fokkink W., Krepska E. *What can formal methods bring to systems biology? Lecture Notes in Computer Science* **5850**, p. 16–22, 2009, `https://www.cs.vu.nl/~wanf/pubs/position-fm4sb.pdf`.

[49] Borovkov A.A. *Probability theory.* Translation from the 5^{th} edition, Universitext (UTX) series, 733 p., Springer London, 2013 (ISBN 978-1-4471-5200-2).

[50] Bortolussi L. *Stochastic concurrent constraint programming. Electronic Notes in Theoretical Computer Science* **164**, p. 65–80, 2006.

[51] Bortolussi L. *Constraint-based approaches to stochastic dynamics of biological systems. Ph.D. thesis, Ph.D. Thesis Series* **CS2007/1**, University of Udine, Italy, 2007, `http://www.dmi.units.it/~bortolu/files/reps/Bortolussi-PhDThesis.pdf`.

[52] Bortolussi L., Policriti A. *Modeling biological systems in concurrent constraint programming. Proceedings of* 2^{nd} *International Workshop on Constraint Based Methods in Bioinformatics - 06 (WCB'06)* (A. Dovier, A. Dal Palù and S. Will, eds.), Nantes, France, 2006, p. 6–29, 2006, `https://dmi.units.it/~bortolu/files/conferences/bortolussi_WCB06.pdf`.

[53] Bortolussi L., Policriti A. *Stochastic concurrent constraint programming and differential equations. Electronic Notes in Theoretical Computer Science* **190(3)**, p. 27–42, September 2007.

[54] Bortolussi L., Policriti A. *Modeling biological systems in stochastic concurrent constraint programming. Constraints* **13(1–2)**, p. 66–90, Springer, June 2008.

[55] Bortolussi L., Policriti A. *Hybrid systems and biology: continuous and discrete modeling for systems biology. Lecture Notes in Computer Science* **5016**, p. 424–448, 2008.

[56] Bortolussi L., Policriti A. *Hybrid dynamics of stochastic programs. Theoretical Computer Science* **411**, p. 2052–2077, 2010.

[57] Bortolussi L., Hillston J., Loreti M. *Fluid approximation of broadcasting systems. Theoretical Computer Science* **816**, p. 221–248, 2020.

[58] Bortolussi L., De Nicola R., Galpin V., Gilmore S., Hillston J., Latella D., Loreti M., Massink M. *CARMA: Collective Adaptive Resource-sharing Markovian Agents. Proceedings of* 13^{th} *Workshop on Quantitative Aspects of Programming Languages and Systems - 15 (QAPL'15)* (N. Bertrand and M. Tribastone, eds.), London, UK, April 2015, *Electronic Proceedings in Theoretical Computer Science* **194**, p. 16–31, Open Publishing Association, 2015.

[59] Boucheneb H., Lime D., Roux O.H. *On multi-enabledness in time Petri nets. Lecture Notes in Computer Science* **7927**, p. 130–149, 2013.

[60] Boyer M., Roux O.H. *Comparison of the expressiveness of arc, place and tranition time Petri nets. Lecture Notes in Computer Science* **4546**, p. 63–82, 2007.

[61] Bradley J.T. *Semi-Markov PEPA: a contradiction in terms? Proceedings of* 2^{nd} *Workshop on Process Algebras and Stochastically Timed Activities - 03 (PASTA'03)* (S.T. Gilmore, ed.), p. 1–6, LFCS, Edinburgh, UK, June 2003, `http://www.doc.ic.ac.uk/~jb/reports/ps/2003-bradley-5.ps.gz`.

[62] Bradley J.T. *Semi-Markov PEPA: compositional modelling and analysis with generally distributed actions. Proceedings of* 20^{th} *Annual UK Performance Engineering Workshop - 04 (UKPEW'04)* (I. Awan, ed.), p. 266–275, University of Bradford, UK, July 2004, `http://www.doc.ic.ac.uk/~jb/reports/ps/2004-bradley-2.ps.gz`.

[63] Bradley J.T. *Semi-Markov PEPA: modelling with generally distributed actions. International Journal of Simulation: Systems, Science and Technology* **6(3–4)**, p. 43–51, UK Society for Modelling and Simulation, The Nottingham Trent University, UK, February 2005, `http://pubs.doc.ic.ac.uk/semi-markov-pepa/semi-markov-pepa.pdf`, `http://www.doc.ic.ac.uk/~jb/reports/ps/2005-bradley-0.ps.gz`, `http://ijssst.info/Vol-06/No-3&4/CRC-Jeremy.pdf`.

[64] Bravetti M. *Specification and analysis of stochastic real-time systems. Ph.D. thesis*, 432 p., University of Bologna, Italy, February 2002, `http://www.cs.unibo.it/~bravetti/papers/phdthesis.ps.gz`.

[65] Bravetti M., D'Argenio P.R. *Tutte le algebre insieme: concepts, discussions and relations of stochastic process algebras with general distributions. Lecture Notes in Computer Science* **2925**, p. 44–88, 2004, `http://www.cs.unibo.it/~bravetti/papers/voss03.ps`.

[66] Bravetti M., Bernardo M., Gorrieri R. *Towards performance evaluation with general distributions in process algebras. Proceedings of* 9^{th} *International Conference on Concurrency Theory - 98 (CONCUR'98)* (D. Sangiorgi, R. de Simone, eds.), Nice, France, September 1998, *Lecture Notes in Computer Science* **1466**, p. 405–422, 1998, `http://www.cs.unibo.it/~bravetti/papers/concur98.ps`, `http://www.sti.uniurb.it/bernardo/documents/concur1998.pdf`.

[67] Bravetti M., Gorrieri R. *The theory of interactive generalized semi-Markov processes. Theoretical Computer Science* **282(1)**, p. 5–32, 2002.

[68] Brinksma E., Hermanns H. *Process algebra and Markov chains. Lecture Notes in Computer Science* **2090**, p. 183–231, 2001.

[69] Brinksma E., Katoen J.-P., Langerak R., Latella D. *A stochastic causality-based process algebra. The Computer Journal* **38 (7)**, p. 552–565, 1995, `http://eprints.eemcs.utwente.nl/6387/01/552.pdf`.

[70] Brodo L., Degano P., Priami C. *A stochastic semantics for BioAmbients. Lecture Notes in Computer Science* **4671**, p. 22–34, 2007.

[71] Bucci G., Sassoli L., Vicario E. *Correctness verification and performance analysis of real-time systems using stochastic preemptive time Petri nets. IEEE Transactions on Software Engineering* **31(11)**, p. 913–927, November 2005, `http://www.dsi.unifi.it/~vicario/Research/TSE05.pdf`.

[72] Buchholz P. *On a Markovian process algebra. Forschungsbericht* **500**, 37 p., Fachbereich Informatik, Technische Universität Dortmund, Germany, January 1994.

[73] Buchholz P. *Markovian process algebra: composition and equivalence. Proceedings of* 2^{nd} *International Workshop on Process Algebras and Performance Modelling - 94 (PAPM'94)* (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, July 1994, *Arbeitsberichte des IMMD* **27(4)**, p. 11–30, Universität Erlangen-Nürnberg, Germany, November 1994.

[74] Buchholz P. *Exact and ordinary lumpability in finite Markov chains. Journal of Applied Probability* **31(1)**, p. 59–75, Applied Probability Trust, March 1994.

[75] Buchholz P. *A notion of equivalence for stochastic Petri nets. Lecture Notes in Computer Science* **935**, p. 161–180, 1995.

[76] Buchholz P. *Iterative decomposition and aggregation of labeled GSPNs. Lecture Notes in Computer Science* **1420**, p. 226–245, 1998.

[77] Buchholz P., Kemper P. *Quantifying the dynamic behavior of process algebras. Lecture Notes in Computer Science* **2165**, p. 184–199, 2001.

[78] Buchholz P., Tarasyuk I.V. *Net and algebraic approaches to probabilistic modeling. Joint Novosibirsk Computing Center and Institute of Informatics Systems Bulletin, Series Computer Science* **15**, p. 31–64, Novosibirsk, 2001, `http://itar.iis.nsk.su/files/itar/pages/spnpancc.pdf`.

[79] Buchholz P., Tarasyuk I.V. *Equivalences for stochastic Petri nets and stochastic process algebras. Vestnik, Quartal Journal of Novosibirsk State University, Series: Mathematics, Mechanics and Informatics* **6(1)**, p. 14–42, Novosibirsk State University, Novosibirsk, 2006 (in Russian), `http://itar.iis.nsk.su/files/itar/pages/vestnik06.pdf`.

[80] Butkova Y., Hartmanns A., Hermanns H. *A Modest approach to modelling and checking Markov automata. Lecture Notes in Computer Science* **11785**, p. 52–69, 2019.

[81] Caravagna G., Hillston J. *Bio-PEPAd: a non-Markovian extension of Bio-PEPA. Theoretical Computer Science* **419**, p. 26–49, February 2012.

[82] Cardelli L. *On process rate semantics. Theoretical Computer Science* **391(3)**, p. 190–215, February 2008.

[83] Cardelli L. *From processes to ODEs by chemistry. Proceedings of* 5^{th} *IFIP International Conference on Theoretical Computer Science - 08 (TCS'08), IFIP* 20^{th} *World Computer Congress - 08 (WCC'08), TC 1, Foundations of Computer Science* (G. Ausiello, J. Karhumäki, G. Mauri, L. Ong, eds.), Milan, Italy, September 2008, *IFIP International Federation for Information Processing (IFIPAICT)* **273**, p. 261–281, Springer, Boston, USA, 2008.

[84] Cardelli L., Zavattaro G. *On the computational power of biochemistry. Lecture Notes in Computer Science* **5147**, p. 65–80, 2008.

[85] Cattani S., Segala R. *Decision algorithms for probabilistic bisimulation. Lecture Notes in Computer Science* **2421**, p. 371–385, 2002, `http://www.cs.bham.ac.uk/~dxp/papers/CS02.pdf`, `http://qav.cs.ox.ac.uk/papers/CS02.pdf`.

[86] Chamseddine N., Duflot M., Fribourg L., Picaronny C., Sproston J. *Computing expected absorption times for parametric determinate probabilistic timed automata. Proceedings of* 5^{th} *International Conference on Quantitative Evaluation of Systems - 08 (QEST'08)*, St. Malo, France, September 2008, p. 254–263, IEEE Computer Society Press, 2008.

[87] Chiola G. *A software package for the analysis of generalized stochastic Petri net models. Proceedings of* 1^{st} *International Workshop on Timed Petri Nets - 85*, Turin, Italy, IEEE Computer Society Press, July 1985.

[88] Christoff I. *Testing equivalence and fully abstract models of probabilistic processes. Lecture Notes in Computer Science* **458**, p. 126–140, 1990.

[89] Ciardo G. *Discrete-time Markovian stochastic Petri nets. Computations with Markov Chains: Proceedings of* 2^{nd} *International Workshop on the Numerical Solution of Markov Chains - 95 (NSMC'95)* (W.J. Stewart, ed.), Raleigh, NC, USA, January 1995, Chapter **20**, p. 339–358, Kluwer Academic Publishers, Boston, MA, USA, 1995, `http://www.cs.ucr.edu/~ciardo/pubs/1995NSMC-Discrete.pdf`, `http://www.cs.iastate.edu/~ciardo/pubs/1995NSMC-Discrete.pdf`.

[90] Ciardo G., Muppala J.K., Trivedi K.S. *SPNP: stochastic Petri net package. Proceedings of* 3^{rd} *International Workshop on Petri Nets and Performance Models - 89 (PNPM'89)*, Kyoto, Japan, p. 142–151, IEEE Computer Society Press, December 1989, `http://www.cs.iastate.edu/~ciardo/pubs/1989PNPM-SPNP.pdf`, `http://people.ee.duke.edu/~kst/spn_papers/package.ps`.

[91] Ciardo G., Muppala J.K., Trivedi K.S. *On the solution of GSPN reward models. Performance Evaluation* **12(4)**, p. 237–253, July 1991, `http://people.ee.duke.edu/~kst/spn_papers/gspnrew.ps`.

[92] Ciobanu G. *Analyzing non-Markovian systems by using a stochastic process calculus and a probabilistic model checker. Mathematics* **11**, Article 302, 17 p., Multidisciplinary Digital Publishing Institute (MDPI), Basel, Switzerland, January 2023.

[93] Ciobanu G., Rotaru A.S. *PHASE: a stochastic formalism for phase-type distributions. Lecture Notes in Computer Science* **8829**, p. 91–106, 2014.

[94] Ciobanu G., Rotaru A.S. *Phase-type approximations for non-Markovian systems: a case study. Lecture Notes in Computer Science* **8938**, p. 323–334, 2015.

[95] Ciocchetta F., Hillston J. *Bio-PEPA: An extension of the process algebra PEPA for biochemical networks. Electronic Notes in Theoretical Computer Science* **194(3)**, p. 103–117, 2008.

[96] Ciocchetta F., Hillston J. *Process algebras in systems biology. Lecture Notes in Computer Science* **5016**, p. 265–312, 2008.

[97] Ciocchetta F., Hillston J. *Bio-PEPA: a framework for the modelling and analysis of biochemical networks. Theoretical Computer Science* **410(33–34)**, p. 3065–3084, August 2009.

[98] Clark G., Gilmore S. *State-aware performance analysis with eXtended Stochastic Probes. Lecture Notes in Computer Science* **5261**, p. 125–140, 2008.

[99] Clark A., Gilmore S., Hillston J., Tribastone M. *Stochastic process algebras. Lecture Notes in Computer Science* **4486**, p. 132–179, 2007, `http://homepages.inf.ed.ac.uk/jeh/bertinoro_tutorial_2007.pdf`.

[100] Coderch M., Willsky A.S., Sastry S.S., Castanon D.A. *Hierarchical aggregation of singularly perturbed finite state Markov processes. Stochastics* **8(4)**, p. 259–289, Gordon and Breach Science Publishers Inc., New York, NY, USA, 1983, `http://ssg.mit.edu/~willsky/publ_pdfs/41_pub_STOC.pdf`.

[101] Coppo M., Damiani F., Drocco M., Grassi E., Troina A. *Stochastic calculus of wrapped compatnents. Proceedings of* 8^{th} *International Workshop on Quantitative Aspects of Programming Languages - 10 (QAPL'10)* (A. Di Pierro, G. Norman, eds.), 2010, *Electronic Proceedings in Theoretical Computer Science* **28**, p. 82–98, Open Publishing Association, 2010.

[102] Coste N., Hermanns H., Lantreibecq E., Serwe W. *Towards performance prediction of compositional models in industrial GALS designs. Lecture Notes in Computer Science* **5643**, p. 204–218, 2009.

[103] Credi A., Garavelli M., Laneve C., Pradalier S., Silvi S., Zavattaro G. *Modelization and simulation of nano devices in nanoκ calculus. Lecture Notes in Computer Science* **4695**, p. 168–183, 2007.

[104] Cubuktepe M., Jansen N., Junges S., Katoen J.-P., Topcu U. *Scenario-based verification of uncertain MDPs. Lecture Notes in Computer Science* **12078**, p. 287–305, 2020.

[105] Cumani A. *ESP — a package for the evaluation of stochastic Petri nets with phase-type distributed transition times. Proceedings of* 1^{st} *International Workshop on Timed Petri Nets - 85*, Turin, Italy, July 1985, p. 144–151, IEEE Computer Society Press, 1985.

[106] Danos V., Feret J., Fontana W., Harmer R., Krivine J. *Rule-based modelling of cellular signalling. Lecture Notes in Computer Science* **4703**, p. 17–41, 2007.

[107] Daws C. *Symbolic and parametric model checking of discrete-time Markov chains. Lecture Notes in Computer Science* **3407**, p. 280–294, 2005.

[108] Degano P., Prandi D., Priami C., Quaglia P. *Beta-binders for biological quantitative experiments. Electronic Notes in Theoretical Computer Science* **164**, p. 101–117, 2006.

[109] Degano P., Priami C. *Non-interleaving semantics for mobile processes. Theoretical Computer Science* **216(1–2)**, p. 237–270, March 1999.

[110] Degasperi A. *Multi-scale modelling of biological systems in process algebra. Ph.D. thesis*, 198 p., School of Computing Science, College of Science and Engineering, University of Glasgow, UK, July 2011.

[111] Degasperi A., Calder M. *Multi-scale modelling of biological systems in process algebra with multiway synchronisation. Proceedings of* 9^{th} *International Conference on Computational Methods in Systems Biology - 11 (CMSB'11)*, Paris, France, September 2011, p. 195–208, ACM Press, 2011.

[112] Degasperi A., Calder M. *A process algebra framework for multi-scale modelling of biological systems. Theoretical Computer Science* **488**, p. 15–45, June 2013.

[113] Delahaye B., Lime D., Petrucci L. *Parameter synthesis for parametric interval Markov chains. Lecture Notes in Computer Science* **9583**, p. 372–390, 2016.

[114] Dematté L., Priami C., Romanel A. *Modelling and simulation of biological processes in BlenX. ACM SIGMETRICS Performance Evaluation Review* **35(4)**, p. 32–39, ACM Press, 2008.

[115] Dematté L., Priami C., Romanel A. *The BetaWorkbench: a computational tool to study the dynamics of biological systems. Briefings In Bioinformatics* **9(5)**, p. 437–449, Oxford University Press, Oxford, UK, May 2008.

[116] Dematté L., Priami C., Romanel A. *The BlenX language: a tutorial. Lecture Notes in Computer Science* **5016**, p. 313–365, 2008.

[117] Deng Y., Hennessy M.C.B. *On the semantics of Markov automata. Lecture Notes in Computer Science* **6756**, p. 307–318, 2011.

[118] Deng Y., Hennessy M.C.B. *On the semantics of Markov automata. Information and Computation* **222**, p. 139–168, January 2013.

[119] Derisavi S., Hermanns H., Sanders W.H. *Optimal state-space lumping of Markov chains. Information Processing Letters* **87(6)**, p. 309–315, Elsevier, 2003, `http://www.perform.illinois.edu/Papers/USAN_papers/02DER01.pdf`.

[120] Eisentraut Ch., Hermanns H., Schuster J., Turrini A., Zhang L. *The quest for minimal quotients for probabilistic automata. Lecture Notes in Computer Science* **7795**, p. 16–31, 2013.

[121] Eisentraut Ch., Hermanns H., Zhang L. *On probabilistic automata in continuous time. Proceedings of* 25^{th} *Annual IEEE Symposium on Logic in Computer Science - 10 (LICS'10)*, Edinburgh, UK, July 2010, p. 342–351, IEEE Computer Society Press, 2010.

[122] El-Rayes A., Kwiatkowska M., Norman G. *Solving infinite stochastic process algebra models through matrix-geometric methods. Proceedings of* 7^{th} *International Workshop on Process Algebra and Performance Modelling - 99 (PAPM'99)*, (J. Hillston, M. Silva, eds.), Zaragoza, Spain, p. 41–62, Prensas Universitarias de Zaragoza, Spain, September 1999, `http://qav.comlab.ox.ac.uk/papers/papm99.pdf`.

[123] Faeder J.R., Blinov M.L., Goldstein B., Hlavacek W.S. *Combinatorial complexity and dynamical restriction of network flows in signal transduction. Systems Biology* **2(1)**, p. 5–15, The Institution of Engineering and Technology (IET), UK, March 2005.

[124] Faeder J.R., Blinov M.L., Hlavacek W.S. *Graphical rule-based representation of signal-transduction networks. Proceedings of ACM Symposium on Applied Computing - 05 (SAC'05)*, March 2005, Santa Fe, New Mexico, USA, p. 133–140, ACM Press, 2005.

[125] Feng Ch., Hillston J. *PALOMA: a process algebra for located Markovian agents. Lecture Notes in Computer Science* **8657**, p. 265–280, 2014.

[126] Feng Ch., Hillston J. *Speed-up of stochastic simulation of PCTMC models by statistical model reduction. Lecture Notes in Computer Science* **9272**, p. 291–305, 2015.

[127] Fourneau J.M. *Collaboration of discrete-time Markov chains: tensor and product form. Performance Evaluation* **67**, p. 779–796, 2010.

[128] Gallegos-Herrada M.A., Ledvinka D., Rosenthal J.S. *Equivalences of geometric ergodicity of Markov chains. Journal of Theoretical Probability*, 27 p., Springer, May 2023, DOI: 10.1007/s10959-023-01240-1.

[129] Galpin V., Hillston J. *Equivalence and discretisation in Bio-PEPA. Lecture Notes in Bioinformatics* **5688**, p. 189–204, 2009.

[130] Galpin V., Zoń N., Wilsdorf P., Gilmore S. *Mesoscopic modelling of pedestrian movement using Carma and its tools. ACM Transactions on Modeling and Computer Simulation (TOMACS)* **28(2)**, Article 11, p. 11:2–11:26, ACM Press, March 2018.

[131] Geisweiller N., Hillston J., Stenico M. *Relating continuous and discrete PEPA models of signalling pathways. Theoretical Computer Science* **404(1–2)**, p. 97–111, September 2008.

[132] Georgoulas A., Hillston J., Milios D., Sanguinetti G. *Probabilistic programming process algebra. Lecture Notes in Computer Science* **8657**, p. 249–264, 2014.

[133] Georgoulas A., Hillston J., Sanguinetti G. *ProPPA: probabilistic programming for stochastic dynamical systems. ACM Transactions on Modeling and Computer Simulation (TOMACS)* **28(1)**, Article 3, p. 3:1–3:23, ACM Press, January 2018.

[134] Gilbert D., Heiner M., Lehrack S. *A unifying framework for modelling and analysing biochemical pathways using Petri nets. Lecture Notes in Computer Science* **4695**, p. 200–216, 2007.

[135] Gillespie D.T. *Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry* **81(25)**, p. 2340–2361, 1977.

[136] Gilmore S., Hillston J., Kloul L., Ribaudo M. *PEPA nets: a structured performance modelling formalism. Performance Evaluation* **54**, p. 79–104, 2003, `http://www.dcs.ed.ac.uk/pepa/pepanetsJournal.pdf`.

[137] van Glabbeek R.J. *The linear time – branching time spectrum II: the semantics of sequential systems with silent moves. Extended abstract. Lecture Notes in Computer Science* **715**, p. 66–81, 1993.

[138] van Glabbeek R.J., Smolka S.A., Steffen B. *Reactive, generative, and stratified models of probabilistic processes. Information and Computation* **121(1)**, p. 59–80, 1995, `http://boole.stanford.edu/pub/prob.ps.gz`.

[139] Gori R., Levi F. *An analysis for proving probabilistic termination of biological systems. Theoretical Computer Science* **471**, p. 27–73, February 2013.

[140] Götz N., Herzog U., Rettelbach M. *Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras. Lecture Notes in Computer Science* **729**, p. 121–146, 1993.

[141] Guck D., Hatefi H., Hermanns H., Katoen J.-P., Timmer M. *Modelling, reduction and analysis of Markov automata. Lecture Notes in Computer Science* **8054**, p. 55–71, 2013.

[142] Guck D., Hatefi H., Hermanns H., Katoen J.-P., Timmer M. *Analysis of timed and long-run objectives for Markov automata. Logical Methods in Computer Science* **10(3:17)**, p. 1–29, 2014, `http://arxiv.org/pdf/1407.7356.pdf`.

[143] Guenther M.C., Bradley J.T. *Higher moment analysis of a spatial stochastic process algebra. Lecture Notes in Computer Science* **6977**, p. 87–101, 2011.

[144] Guenther M.C., Dingle N.J., Bradley J.T., Knottenbelt W.J. *Passage-time computation and aggregation strategies for large semi-Markov processes. Performance Evaluation* **68**, p. 221–236, 2011.

[145] Guenther M.C., Stefanek A., Bradley J.T. *Moment closures for performance models with highly non-linear rates. Lecture Notes in Computer Science* **7587**, p. 32–47, 2013.

[146] Guerriero M.L., Pokhilko A., Fernández A.P., Halliday K.J., Millar A.J., Hillston J. *Stochastic properties of the plant circadian clock. Journal of the Royal Society Interface* **9(69)**, p. 744–756, The Royal Society Publishing, UK, April 2012.

[147] Hahn E.M., Hartmanns A., Hermanns H., Katoen J.-P. *A compositional modelling and analysis framework for stochastic hybrid systems. Formal Methods in System Design* **43(2)**, p. 191–232, 2013.

[148] Hahn E.M., Hermanns H., Zhang L. *Probabilistic reachability for parametric Markov models. International Journal on Software Tools for Technology Tranfer* **13(1)**, p. 3–19, Springer, 2011.

[149] Han T., Katoen J.-P., Mereacre A. *Approximate parameter synthesis for probabilistic time-bounded reachability. Proceedings of* 29^{th} *IEEE Real-Time Systems Symposium - 08 (RTSS'08)*, New York, USA, p. 173–182, IEEE Computer Society Press, 2008.

[150] Hanish H.M. *Analysis of place/transition nets with timed-arcs and its application to batch process control. Lecture Notes in Computer Science* **691**, p. 282–299, 1993.

[151] Harrison P.G., Strulo B. *Stochastic process algebra for discrete event simulation. Quantitative Methods in Parallel Systems*, Esprit Basic Research Series, p. 18–37 (F. Bacelli, A. Jean-Marie, I. Mitrani, eds.), Springer, Berlin, Germany, 1995.

[152] Hartmanns A. *An overview of Modest models and tools for real stochastic timed systems. Proceedings of 5^{th} Workshop on Models for Formal Analysis of Real Systems - 22 (MARS'22)* (C. Dubslaff, B. Luttik, eds.), 2022, *Electronic Proceedings in Theoretical Computer Science* **355**, p. 1–12, Open Publishing Association, 2022.

[153] Hashemi V., Hermanns H., Song L. *Reward-bounded reachability probability for uncertain weighted MDPs. Lecture Notes in Computer Science* **9583**, p. 351–371, 2016.

[154] Hatefi H., Hermanns H. *Improving time bounded reachability computations in interactive Markov chains. Lecture Notes in Computer Science* **8161**, p. 250–266, 2013.

[155] Haverkort B.R. *Markovian models for performance and dependability evaluation. Lecture Notes in Computer Science* **2090**, p. 38–83, 2001, `http://www-i2.informatik.rwth-aachen.de/Teaching/Seminar/VOSS2005/have01.pdf`.

[156] Hayden R.A., Bradley J.T. *A fluid analysis framework for a Markovian process algebra. Theoretical Computer Science* **411**, p. 2260–2297, May 2010.

[157] Hayden R.A., Bradley J.T., Clark A. *Performance specification and evaluation with unified stochastic probes and fluid analysis. IEEE Transactions on Software Engineering* **39(1)**, p. 97–118, IEEE Computer Society Press, January 2013, `http://pubs.doc.ic.ac.uk/fluid-unified-stochastic-probes/fluid-unified-stochastic-probes.pdf`.

[158] Hermanns H. *Interactive Markov chains: the quest for quantified quality. Lecture Notes in Computer Science* **2428**, 217 p., 2002.

[159] Hermanns H., Herzog U., Katoen J.-P. *Process algebra for performance evaluation. Theoretical Computer Science* **274(1–2)**, p. 43–87, 2002, `http://wwwhome.cs.utwente.nl/~hermanns/papers/tcs00.pdf`.

[160] Hermanns H., Katoen J.-P. *Automated compositional Markov chain generation for a plain-old telephone system. Science of Computer Programming* **36(1)**, p. 97–127, January 2000.

[161] Hermanns H., Lohrey M. *Observational congruence in stochastic timed calculus with maximal progress. Technischer Bericht* **IMMD VII-7/97**, 37 p., Institut für Matematische Maschinen und Datenverarbeitung (IMMD), Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), Lehrstuhl für Informatik VII (Rechnerarchitektur und Verkehrstheorie), Erlangen, Germany, October 1997.

[162] Hermanns H., Rettelbach M. *Syntax, semantics, equivalences and axioms for MTIPP. Proceedings of 2^{nd} Workshop on Process Algebras and Performance Modelling*, Regensberg / Erlangen (Herzog U., Rettelbach M., eds.), *Arbeitsberichte des IMMD* **27**, p. 71–88, University of Erlangen, Germany, 1994, `http://ftp.informatik.uni-erlangen.de/local/inf7/papers/Hermanns/syntax_semantics_equivalences_axioms_for_MTIPP.ps.gz`.

[163] Hillston J. *The nature of synchronisation. Proceedings of the 2^{nd} International Workshop on Process Algebra and Performance Modelling - 94 (PAPM'94)*, Regensberg / Erlangen (U. Herzog, M. Rettelbach, eds.), *Arbeitsberichte des IMMD* **27**, p. 51–70, University of Erlangen, Germany, November 1994, `http://www.dcs.ed.ac.uk/pepa/synchronisation.pdf`.

[164] Hillston J. *A compositional approach to performance modelling.* 158 p., Cambridge University Press, UK, 1996, `http://www.dcs.ed.ac.uk/pepa/book.pdf`.

[165] Hillston J. *Fluid flow approximation of PEPA models. Proceedings of 2^{nd} International Conference on the Quantitative Evaluation of Systems - 05 (QEST'05)*, Turin, Italy, September 2005, p. 33–43, IEEE Computer Society Press, 2005, `http://www.dcs.ed.ac.uk/pepa/fluidflow.pdf`.

[166] Hillston J. *Stochastic process algebras and their Markovian semantics. ACM SIGLOG News* **5(2)**, p. 20–35, April 2018.

[167] Hoare C.A.R. *Communicating sequential processes.* Prentice-Hall, London, UK, 1985, `http://www.usingcsp.com/cspbook.pdf`.

[168] Horváth A., Paolieri M., Ridi L., Vicario E. *Probabilistic model checking of non-Markovian models with concurrent generally distributed timers. Proceedings of 8^{th} International Conference on Quantitative Evaluation of Systems - 11 (QEST'11)*, Aachen, Germany, September 2011, p. 131–140, IEEE Computer Society Press, 2011, `http://www.di.unito.it/~horvath/publications/papers/HoPaRiVi11.pdf`, `http://www.dsi.unifi.it/~vicario/Research/milton.pdf`.

[169] Horváth A., Paolieri M., Ridi L., Vicario E. *Transient analysis of non-Markovian models using stochastic state classes. Performance Evaluation* **69(7–8)**, p. 315–335, 2012, `http://www.dsi.unifi.it/~vicario/Research/HPRV_PEva12_Ola.pdf`.

[170] Horváth A., Paolieri M., Vicario E. *Approximating distributions and transient probabilities by matrix exponential distributions and functions. Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, Chapter **5**, p. 107–127 (D. Bruneo, S. Distefano, eds.), *Performability Engineering Series*, Scrivener Publishing LLC, Beverly, MA, USA, April 2015.

[171] Horváth A., Puliafito A., Scarpa M., Telek M. *Analysis and evaluation of non-Markovian stochastic Petri nets. Lecture Notes in Computer Science* **1786**, p. 171–187, March 2000, `http://webspn.hit.bme.hu/~telek/cikkek/horv00b.ps.gz`.

[172] Horváth A., Vicario E. *Construction of phase type distributions by Bernstein exponentials. Lecture Notes in Computer Science* **14231**, p. 201–215, 2023.

[173] Iacobelli G., Tribastone M., Vandin A. *Differential bisimulation for a Markovian process algebra. Lecture Notes in Computer Science* **9234**, p. 293–306, 2015.

[174] Jansen N., Corzilius F., Volk M., Wimmer R., Ábráham E., Katoen J.-P., Becker B. *Accelerating parametric probabilistic verification. Lecture Notes in Computer Science* **8657**, p. 404–420, 2014.

[175] Jategaonkar L., Meyer A.R. *Deciding true concurrency equivalences on safe, finite nets. Theoretical Computer Science* **154(1)**, p. 107–143, 1996.

[176] John M., Lhoussaine C., Niehren J., Uhrmacher A.M. *The attributed pi calculus. Lecture Notes in Computer Science* **5307**, p. 83–102, 2008.

[177] Jones R.L., Ciardo G. *On phased delay stochastic Petri nets. Proceedings of 9^{th} IEEE International Workshop on Petri Nets and Performance Models - 01 (PNPM'01)*, p. 165–174, Aachen, Germany, IEEE Computer Society Press, September 2001.

[178] Jou C.-C., Smolka S.A. *Equivalences, congruences and complete axiomatizations for probabilistic processes. Lecture Notes in Computer Science* **458**, p. 367–383, 1990.

[179] Katoen J.-P. *Quantinative and qualitative extensions of event structures. Ph. D. thesis, CTIT Ph. D.-thesis series* **96-09**, 303 p., Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 1996.

[180] Katoen J.-P., D'Argenio P.R. *General distributions in process algebra. Lecture Notes in Computer Science* **2090**, p. 375–429, 2001.

[181] Katoen J.-P., Brinksma E., Latella D., Langerak R. *Stochastic simulation of event structures. Proceedings of 4^{th} International Workshop on Process Algebra and Performance Modelling - 96 (PAPM'96)* (M. Ribaudo, ed.), p. 21–40, CLUT Press, Torino, Italy, July 1996, `http://eprints.eemcs.utwente.nl/6487/01/263_KLLB96b.pdf`.

[182] Klin B., Sassone V. *Structural operational semantics for stochastic process calculi. Lecture Notes in Computer Science* **4962**, p. 428–442, 2008.

[183] Koutny M. *A compositional model of time Petri nets. Lecture Notes in Computer Science* **1825**, p. 303–322, 2000.

[184] Krivine J. *Systems biology. ACM SIGLOG News* **4(3)**, p. 43–61, ACM Press, July 2017.

[185] Krivine J., Milner R., Troina A. *Stochastic bigraphs. Electronic Notes in Theoretical Computer Science* **218**, p. 73–96, 2008.

[186] Kulkarni V.G. *Modeling and analysis of stochastic systems. Texts in Statistical Science*, 542 p., Chapman and Hall / CRC Press, 2010.

[187] Kuttler C., Lhoussaine C., Niehren J. *A stochastic Pi calculus for concurrent objects. Lecture Notes in Computer Science* **4545**, p. 232–246, 2007.

[188] Lakatos L., Szeidl L., Telek M. *Introduction to queueing systems with telecommunication applications.* 2^{nd} edition, 559 p., Springer Nature, Cham, Switzerland, 2019 (ISBN 978-3-030-15141-6).

[189] Lakin M.R., Paulevé L., Phillips A. *Stochastic simulation of multiple process calculi for biology. Theoretical Computer Science* **431**, p. 181–206, May 2012.

[190] Lakin M.R., Phillips A. *Modelling, simulating and verifying Turing-powerful strand displacement systems. Lecture Notes in Computer Science* **6937**, p. 130–144, 2011.

[191] Lakin M.R., Youssef S., Cardelli L., Phillips A. *Abstractions for DNA circuit design. Journal of the Royal Society Interface* **9(68)**, p. 470–486, The Royal Society Publishing, UK, March 2012.

[192] Laneve C., Pradalier S., Zavattaro G. *From biochemistry to stochastic processes. Electronic Notes in Theoretical Computer Science* **253**, p. 167–185, 2009.

[193] Lanotte R., Maggiolo-Schettini A., Troina A. *Parametric probabilistic transition systems for system design and analysis. Formal Aspects of Computing* **19(1)**, p. 93–109, March 2007.

[194] Larsen K.G., Skou A. *Bisimulation through probabilistic testing. Information and Computation* **94(1)**, p. 1–28, September 1991.

[195] Latella D., Massink M., de Vink E.P. *Bisimulation of labelled state-to-function transition systems coalgebraically. Logical Methods in Computer Science* **11(4:16)**, p. 1–40, 2015.

[196] López B.N., Núñez G.M. *NMSPA: a non-Markovian model for stochastic processes. Proceedings of International Workshop on Distributed System Validation and Verification - 00 (DSVV'00)*, p. 33–40, 2000, `http://dalila.sip.uclm.es/membros/manolo/papers/dsvv2000.ps.gz`.

[197] López B.N., Núñez G.M., Rubio F. *Stochastic process algebras meet Eden. Lecture Notes in Computer Science* **2335**, p. 29–48, 2002, `http://dalila.sip.uclm.es/membros/manolo/papers/ifm02.zip`.

[198] López B.N., Núñez G.M., Rubio F. *An integrated framework for the performance analysis of asynchronous communicating stochastic processes. Formal Aspects of Computing* **16(3)**, p. 238–262, August 2004.

[199] Loreti M., Hillston J. *Modelling and analysis of collective adaptive systems with CARMA and its tools. Lecture Notes in Computer Science* **9700**, p. 83–119, 2016.

[200] Macià S.H., Valero R.V., Cazorla L.D.C., Cuartero G.F. *Introducing the iteration in sPBC. Proceedings of* 24^{th} *International Conference on Formal Techniques for Networked and Distributed Systems - 04 (FORTE'04)*, Madrid, Spain, *Lecture Notes in Computer Science* **3235**, p. 292–308, October 2004, `http://www.info-ab.uclm.es/retics/publications/2004/forte04.pdf`.

[201] Macià S.H., Valero R.V., Cuartero G.F., de Frutos E.D. *A congruence relation for sPBC. Formal Methods in System Design* **32(2)**, p. 85–128, Springer, The Netherlands, April 2008.

[202] Macià S.H., Valero R.V., Cuartero G.F., Ruiz M.C. *sPBC: a Markovian extension of Petri box calculus with immediate multiactions. Fundamenta Informaticae* **87(3–4)**, p. 367–406, IOS Press, Amsterdam, The Netherlands, 2008.

[203] Macià S.H., Valero R.V., Cuartero G.F., Ruiz M.C., Tarasyuk I.V. *Modelling a video conference system with sPBC. Applied Mathematics and Information Sciences* **10(2)**, p. 475–493, Natural Sciences Publishing, New York, NY, USA, March 2016 (ISSN 1935-0090), DOI: 10.18576/amis/100210, `http://itar.iis.nsk.su/files/itar/pages/amis16.pdf`.

[204] Macià S.H., Valero R.V., de Frutos E.D. *sPBC: a Markovian extension of finite Petri box calculus. Proceedings of* 9^{th} *IEEE International Workshop on Petri Nets and Performance Models - 01 (PNPM'01)*, Aachen, Germany, p. 207–216, IEEE Computer Society Press, September 2001, `http://www.info-ab.uclm.es/retics/publications/2001/pnpm01.ps`.

[205] Marin A., Piazza C., Rossi S. *Proportional lumpability. Lecture Notes in Computer Science* **11750**, p. 265–281, 2019.

[206] Marin A., Piazza C., Rossi S. *Proportional lumpability and proportional bisimilarity. Acta Informatica* **59(2–3)**, p. 211–244, June 2022.

[207] Markovski J., D'Argenio P.R., Baeten J.C.M., de Vink E.P. *Reconciling real and stochastic time: the need for probabilistic refinement. Formal Aspects of Computing* **24(4–6)**, p. 497–518, July 2012.

[208] Markovski J., Sokolova A., Trčka N., de Vink E.P. *Compositionality for Markov reward chains with fast and silent transitions. Performance Evaluation* **66**, p. 435–452, 2009.

[209] Markovski J., de Vink E.P. *Extending timed process algebra with discrete stochastic time. Proceedings of* 12^{th} *International Conference on Algebraic Methodology and Software Technology - 08 (AMAST'08)*, Urbana, IL, USA, *Lecture Notes of Computer Science* **5140**, p. 268–283, 2008.

[210] Markovski J., de Vink E.P. *Performance evaluation of distributed systems based on a discrete real- and stochastic-time process algebra. Fundamenta Informaticae* **95(1)**, p. 157–186, IOS Press, Amsterdam, The Netherlands, 2009.

[211] Marroquín A.O., de Frutos E.D. *TPBC: timed Petri box calculus. Technical Report*, Departamento de Sistemas Infofmáticos y Programación, Universidad Complutense de Madrid, Madrid, Spain, 2000 (in Spanish).

[212] Marroquín A.O., de Frutos E.D. *Extending the Petri box calculus with time. Lecture Notes in Computer Science* **2075**, p. 303–322, 2001.

[213] Marsan M.A. *Stochastic Petri nets: an elementary introduction. Lecture Notes in Computer Science* **424**, p. 1–29, 1990.

[214] Marsan M.A., Balbo G., Conte G., Donatelli S., Franceschinis G. *Modelling with generalized stochastic Petri nets. Wiley Series in Parallel Computing*, John Wiley and Sons, 316 p., 1995, `http://www.di.unito.it/~greatspn/GSPN-Wiley`.

[215] Marsan M.A., Chiola G. *On Petri nets with deterministic and exponentially distributed firing times. Lecture Notes in Computer Science* **266**, p. 132–145, 1987.

[216] Marsan M.A., Chiola G., Fumagalli A. *Improving the efficiency of the analysis of DSPN models. Lecture Notes in Computer Science* **424**, p. 30–50, 1990.

[217] Massink M., Brambilla M., Latella D., Dorigo M., Birattari M. *On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics. Swarm Intelligence* **7**, p. 201–228, Springer, April 2013.

[218] Massink M., Latella D. *Fluid analysis of foraging ants. Lecture Notes in Computer Science* **7274**, p. 152–165, 2012.

[219] Massink M., Latella D., Bracciali A., Harrison M.D., Hillston J. *Scalable context-dependent analysis of emergency egress models. Formal Aspects of Computing* **24(2)**, p. 267–302, British Computer Society (BCS), March 2012.

[220] Maus C., John M., Röhl M., Uhrmacher A.M. *Hierarchical modeling for computational biology. Lecture Notes in Computer Science* **5016**, p. 381–124, 2008.

[221] McCaig Ch., Norman R., Shankland C. *Process algebra models of population dynamics. Lecture Notes in Computer Science* **5147**, p. 139–155, 2008.

[222] Merlin Ph.M., Farber D.J. *Recoverability of communication protocols: implications of a theoretical study. IEEE Transactions on Communications* **24(9)**, p. 1036–1043, IEEE Computer Society Press, September 1976.

[223] Milner R.A.J. *Communication and concurrency.* Prentice-Hall, 260 p., Upper Saddle River, NJ, USA, 1989.

[224] Milner R.A.J., Parrow J.G., Walker D.J. *A calculus of mobile processes (I and II). Information and Computation* **100(1)**, p. 1–77, 1992.

[225] Mitrani I., Ost A., Rettelbach M. *TIPP and the spectral expansion method. Quantitative Methods in Parallel Systems*, Esprit Basic Research Series, p. 99–113 (F. Bacelli, A. Jean-Marie, I. Mitrani, eds.), Springer, Berlin, Germany, 1995.

[226] Molloy M.K. *On the integration of the throughput and delay measures in distributed processing models. Ph.D. thesis*, *Report* **CSD-810-921**, 108 p., University of California, Los Angeles, CA, USA, 1981.

[227] Molloy M.K. *Performance analysis using stochastic Petri nets. IEEE Transactions on Computing* **31(9)**, p. 913–917, IEEE Computer Society Press, 1982.

[228] Molloy M.K. *Discrete time stochastic Petri nets. IEEE Transactions on Software Engineering* **11(4)**, p. 417–423, 1985.

[229] Montanari U., Pistore M., Yankelevich D. *Efficient minimization up to location equivalence. Lecture Notes in Computer Science* **1058**, p. 265–279, 1996.

[230] Mudge T.N., Al-Sadoun H.B. *A semi-Markov model for the performance of multiple-bus systems. IEEE Transactions on Computers* **C-34(10)**, p. 934–942, October 1985, `http://www.eecs.umich.edu/~tnm/papers/SemiMarkov.pdf`.

[231] Neuts M.F. *Matrix-geometric solutions in stochastic models: an algorithmic approach. Johns Hopkins Series in the Mathematical Sciences* **2**, 352 p., Johns Hopkins University Press, Baltimore, MD, USA, July 1981.

[232] Niaouris A. *An algebra of Petri nets with arc-based time restrictions. Lecture Notes in Computer Science* **3407**, p. 447–462, 2005.

[233] Niaouris A., Koutny M. *An algebra of timed-arc Petri nets. Technical Report* **CS-TR-895**, 60 p., School of Computer Science, University of Newcastle upon Tyne, UK, March 2005, `http://www.cs.ncl.ac.uk/publications/trs/papers/895.pdf`.

[234] De Nicola R., Katoen J.-P., Latella D., Loreti M., Massink M. *Model checking mobile stochastic logic. Theoretical Computer Science* **382(1)**, p. 42–70, August 2007.

[235] De Nicola R., Katoen J.-P., Latella D., Massink M. *StoKlaim: a stochastic extension of Klaim. Technical Report* **2006-TR-01**, Consiglio Nazionale delle Ricerche, Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', Italy, 2006.

[236] De Nicola R., Latella D., Loreti M., Massink M. *A uniform definition of stochastic process calculi. ACM Computing Surveys* **46(1)**, Article 5, 56 p., ACM Press, October 2013.

[237] De Nicola R., Latella D., Loreti M., Massink M. *Quantitative analysis of distributed systems in StoKlaim: a tutorial. Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, Chapter **2**, p. 27–55 (D. Bruneo, S. Distefano, eds.), *Performability Engineering Series*, Scrivener Publishing LLC, Beverly, MA, USA, April 2015.

[238] Paige R., Tarjan R.E. *Three partition refinement algorithms. SIAM Journal of Computing* **16(6)**, p. 973–989, Society for Industrial and Applied Mathematics, 1987.

[239] Păun Gh., Romero-Campero F.J. *Membrane computing as a modeling framework. Cellular systems case studies. Lecture Notes in Computer Science* **5016**, p. 168–214, 2008.

[240] Pedersen M.R., Plotkin G.D. *A language for biochemical systems. Lecture Notes in Computer Science* **5307**, p. 63–82, 2008.

[241] Pedersen M.R., Plotkin G.D. *A language for biochemical systems: design and formal specifications. Lecture Notes in Computer Science* **5945**, p. 77–145, 2010.

[242] Peng Y., Wang Sh., Zhan N., Zhang L. *Extending hybrid CSP with probability and stochasticity. Proceedings of* 1^{st} *Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA'15)*, Nanjing, China, November 2015, *Lecture Notes in Computer Science* **9409**, p. 87–102, 2015.

[243] Phillips A., Cardelli L. *Efficient, correct simulation of biological processes in the stochastic pi-calculus. Lecture Notes in Computer Science* **4695**, p. 184–199, 2007.

[244] Phillips A., Cardelli L. *A programming language for composable DNA circuits. Journal of the Royal Society Interface* **6(Suppl4)**, p. S419–S436, The Royal Society Publishing, UK, August 2009.

[245] Phillips A., Cardelli L., Castagna G. *A graphical representation for biological processes in the stochastic pi-calculus. Lecture Notes in Computer Science* **4230**, p. 123–152, 2006.

[246] Piazza C., Rossi S. *Reasoning about proportional lumpability. Lecture Notes in Computer Science* **12846**, p. 372–390, 2021.

[247] Pino L.F., Boncho F., Valencia F.D. *A behavioural congruence for concurrent constraint programming with nondeterministic choice. Lecture Notes in Computer Science* **8687**, p. 351–368, 2014.

[248] Plotkin G.D. *A calculus of chemical systems. Lecture Notes in Computer Science* **8000**, p. 445–465, 2013.

[249] Priami C. *Stochastic π-calculus. The Computer Journal* **38(7)**, p. 578–589, Oxford University Press, Oxford, UK, 1995.

[250] Priami C. *Stochastic π-calculus with general distributions. Proceedings of 4^{th} International Workshop on Process Algebra and Performance Modelling - 96 (PAPM'96)* (M. Ribaudo, ed.), p. 41–57, CLUT Press, Torino, Italy, 1996.

[251] Priami C. *Language-based performance prediction for distributed and mobile systems. Information and Computation* **175(2)**, p. 119–145, June 2002.

[252] Priami C., Quaglia P. *Beta-binders for biological interactions. Lecture Notes in Computer Science* **3082**, p. 21–34, 2005.

[253] Priami C., Regev A., Silverman W., Shapiro E. *Application of a stochastic namepassing calculus to representation and simulation of molecular processes. Information Processing Letters* **80(1)**, p. 25–31, Elsevier, 2001.

[254] Pulungan R., Hermanns H. *A construction and minimization service for continuous probability distributions. International Journal on Software Tools for Technology Tranfer* **17(1)**, p. 77–90, Springer, February 2015.

[255] Quatmann T., Dehnert Ch., Jansen N., Junges S., Katoen J.-P. *Parameter synthesis for Markov models: faster than ever. Lecture Notes in Computer Science* **9938**, p. 50–67, 2016.

[256] Ramchandani C. *Performance evaluation of asynchronous concurrent systems by timed Petri nets. Ph.D. thesis*, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 1973.

[257] Rettelbach M. *Immediate transitions in stochastic process algebras — theory and application. Ph.D. thesis*, Universität Erlangen-Nürnberg, Germany, 1995 (in German).

[258] Rettelbach M. *Probabilistic branching in Markovian process algebras. The Computer Journal* **38(7)**, p. 590–599, Oxford University Press, Oxford, UK, 1995.

[259] Reynier P.-A., Sangnier A. *Weak time Petri nets strike back! Lecture Notes in Computer Science* **5710**, p. 557–571, 2009.

[260] Ross S.M. *Stochastic processes.* 2^{nd} edition, John Wiley and Sons, 528 p., New York, USA, April 1996.

[261] Ross S.M. *Introduction to probability models.* 12^{th} edition, 826 p., Academic Press, Elsevier, UK, 2019 (ISBN 978-0-12-814346-9).

[262] Sciacca E., Spinella S., Calcagno C., Damiani F., Coppo M. *Parameter identification and assessment of nutrient transporters in AM symbiosis through stochastic simulations. Proceedings of CS2Bio'12*, 2012, *Electronic Notes in Theoretical Computer Science* **293**, p. 83–96, 2013.

[263] Shiryaev A.N. *Probability-2.* 3^{rd} edition, Graduate Texts in Mathematics (GTM) series, 348 p., Springer, New York, 2019 (ISBN 978-0-387-72207-8).

[264] Sifakis J., Yovine S. *Compositional specification of timed systems. Lecture Notes in Computer Science* **1046**, p. 345–359, 1996.

[265] Stewart W.J. *Probability, Markov chains, queues, and simulation. The mathematical basis of performance modeling.* 758 p., Princeton University Press, Princeton, NJ, USA, 2009.

[266] Tarasyuk I.V. *Discrete time stochastic Petri box calculus. Berichte aus dem Department für Informatik* **3/05**, 25 p., Carl von Ossietzky Universität Oldenburg, Germany, November 2005 (ISSN 1867-9218), `http://itar.iis.nsk.su/files/itar/pages/dtspbcib_cov.pdf`.

[267] Tarasyuk I.V. *Iteration in discrete time stochastic Petri box calculus. Bulletin of the Novosibirsk Computing Center, Series Computer Science, IIS Special Issue* **24**, p. 129–148, NCC Publisher, Novosibirsk, 2006 (ISSN 1680-6972), http://itar.iis.nsk.su/files/itar/pages/dtsitncc.pdf.

[268] Tarasyuk I.V. *Stochastic Petri box calculus with discrete time. Fundamenta Informaticae* **76(1–2)**, p. 189–218, IOS Press, Amsterdam, The Netherlands, February 2007 (ISSN 0169-2968), http://itar.iis.nsk.su/files/itar/pages/dtspbcfi.pdf.

[269] Tarasyuk I.V. *Equivalences for behavioural analysis of concurrent and distributed computing systems.* 321 p., Geo Academic Publisher, Novosibirsk, Russia, 2007 (ISBN 978-5-9747-0098-9, in Russian).

[270] Tarasyuk I.V. *Equivalence relations for modular performance evaluation in dtsPBC. Mathematical Structures in Computer Science* **24(1)**, p. 78–154 (e240103), Cambridge University Press, Cambridge, UK, February 2014 (ISSN 0960-1295), DOI: 10.1017/S0960129513000029, http://itar.iis.nsk.su/files/itar/pages/dtsdphms.pdf.

[271] Tarasyuk I.V. *Discrete time stochastic and deterministic Petri box calculus. CoRR* **abs/1905.00456** (*arXiv*:**1905.00456**), 57 p., Computing Research Repository, Cornell University Library, Ithaca, NY, USA, May 2019, http://itar.iis.nsk.su/files/itar/pages/dtsdpbcarxiv.pdf, http://arxiv.org/pdf/1905.00456.pdf.

[272] Tarasyuk I.V. *Stochastic bisimulation and performance evaluation in discrete time stochastic and deterministic Petri box calculus dtsdPBC. HAL Open Archives* **hal-02573419v1**, 113 p., France, May 2020, http://itar.iis.nsk.su/files/itar/pages/dtsdpbchal.pdf, https://hal.science/hal-02573419v1/file/dtsdpbchal.pdf.

[273] Tarasyuk I.V. *Discrete time stochastic and deterministic Petri box calculus dtsdPBC. Siberian Electronic Mathematical Reports* **17**, p. 1598–1679, S.L. Sobolev Institute of Mathematics, Novosibirsk, October 2020 (ISSN 1813-3304), DOI: 10.33048/semi.2020.17.112, http://itar.iis.nsk.su/files/itar/pages/dtsdpbcodesemr.pdf, http://semr.math.nsc.ru/v17/p1598-1679.pdf.

[274] Tarasyuk I.V. *Performance evaluation in stochastic process algebra dtsdPBC. Siberian Electronic Mathematical Reports* **18(2)**, p. 1105–1145, S.L. Sobolev Institute of Mathematics, Novosibirsk, October 2021 (ISSN 1813-3304), DOI: 10.33048/semi.2021.18.085, http://itar.iis.nsk.su/files/itar/pages/dtsdpbcpevsemr.pdf, http://semr.math.nsc.ru/v18/n2/p1105-1145.pdf.

[275] Tarasyuk I.V. *Performance preserving equivalence for stochastic process algebra dtsdPBC. Siberian Electronic Mathematical Reports* **20(2)**, p. 646–699, S.L. Sobolev Institute of Mathematics, Novosibirsk, July 2023 (ISSN 1813-3304), DOI: 10.33048/semi.2023.20.039, http://itar.iis.nsk.su/files/itar/pages/dtsdpbceqsemr.pdf, http://semr.math.nsc.ru/v20/n2/p646-699.pdf.

[276] Tarasyuk I.V., Macià S.H., Valero R.V. *Discrete time stochastic Petri box calculus with immediate multiactions. Technical Report* **DIAB-10-03-1**, 25 p., Department of Computer Systems, High School of Computer Science Engineering, University of Castilla - La Mancha, Albacete, Spain, March 2010, http://itar.iis.nsk.su/files/itar/pages/dtsipbc.pdf, http://www.dsi.uclm.es/descargas/technicalreports/DIAB-10-03-1/dtsipbc.pdf.

[277] Tarasyuk I.V., Macià S.H., Valero R.V. *Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC. Proc.* 6^{th} *International Workshop on Practical Applications of Stochastic Modelling - 12 (PASM'12) and* 11^{th} *International Workshop on Parallel and Distributed Methods in Verification - 12 (PDMC'12), Electronic Notes in Theoretical Computer Science* **296**, p. 229–252, Elsevier, 2013, http://itar.iis.nsk.su/files/itar/pages/dtsipbcentcs.pdf.

[278] Tarasyuk I.V., Macià S.H., Valero R.V. *Performance analysis of concurrent systems in algebra dtsiPBC. Programming and Computer Software* **40(5)**, p. 229–249, Pleiades Publishing, Ltd., September 2014 (ISSN 0361-7688), DOI: 10.1134/S0361768814050089, http://www.maik.rssi.ru/journals/procom.htm.

[279] Tarasyuk I.V., Macià S.H., Valero R.V. *Stochastic process reduction for performance evaluation in dtsiPBC. Siberian Electronic Mathematical Reports* **12**, p. 513–551, S.L. Sobolev Institute of Mathematics, Novosibirsk, September 2015 (ISSN 1813-3304), DOI: 10.17377/semi.2015.12.044, http://itar.iis.nsk.su/files/itar/pages/dtsipbcsemr.pdf, http://semr.math.nsc.ru/v12/p513-551.pdf.

[280] Tarasyuk I.V., Macià S.H., Valero R.V. *Stochastic equivalence for performance analysis of concurrent systems in dtsiPBC. Siberian Electronic Mathematical Reports* **15**, p. 1743–1812, S.L. Sobolev Institute of Mathematics, Novosibirsk, December 2018 (ISSN 1813-3304), DOI: 10.33048/semi.2018.15.144, `http://itar.iis.nsk.su/files/itar/pages/dtsipbceqsemr.pdf`, `http://semr.math.nsc.ru/v15/p1743-1812.pdf`.

[281] Timmer M., Katoen J.-P., van de Pol J., Stoelinga M.I.A. *Efficient modelling and generation of Markov automata. Lecture Notes in Computer Science* **7454**, p. 364–379, 2012.

[282] Timmer M., van de Pol J., Stoelinga M.I.A. *Confluence reduction for Markov automata. Lecture Notes in Computer Science* **8053**, p. 243–257, 2013.

[283] Tofts C. *Processes with probabilities, priority and time. Formal Aspects of Computing* **6(5)**, p. 536–564, September 1994.

[284] Tofts C. *Symbolic approaches to probability distributions in process algebra. Formal Aspects of Computing* **12(5)**, p. 392–415, December 2000.

[285] Trivedi K.S. *Probability and statistics with reliability, queuing, and computer science applications.* 2^{nd} edition, 857 p., John Wiley and Sons, Hoboken, NJ, USA, 2016 (ISBN 978-1-119-28542-7).

[286] Trivedi K.S., Bobbio A. *Reliability and availability engineering: modeling, analysis and applications.* 1^{st} edition, 712 p., Cambridge University Press, Cambridge, UK, 2017 (ISBN 978-1-107-09950-0).

[287] Tschaikowski M., Tribastone M. *Exact fluid lumpability for Markovian process algebra. Lecture Notes in Computer Science* **7454**, p. 380–394, 2012.

[288] Tschaikowski M., Tribastone M. *Exact fluid lumpability in Markovian process algebra. Theoretical Computer Science* **538**, p. 140–166, 2014.

[289] Tschaikowski M., Tribastone M. *Extended differential aggregations in process algebra for performance and biology. Proceedings of* 12^{th} *International Workshop on Quantitative Aspects of Programming Languages and Systems - 14 (QAPL'14)*, Grenoble, France, April 2014, *Electronic Proceedings in Theoretical Computer Science* **154**, p. 34–47, Open Publishing Association, 2014, `http://eprints.imtlucca.it/2591/1/EPTCS_Tribastone_Tschaikowski.pdf`, `http://arxiv.org/pdf/1406.2067v1.pdf`.

[290] Tschaikowski M., Tribastone M. *A unified framework for differential aggregations in Markovian process algebra. Journal of Logical and Algebraic Methods in Programming* **84**, p. 238–258, 2015.

[291] Valero R.V., Cambronero P.M.E. *Using unified modelling language to model the publish/subscribe paradigm in the context of timed Web services with distributed resources. Mathematical and Computer Modelling of Dynamical Systems* **23(6)**, p. 570–594, Tailor and Francis, 2017.

[292] Versari C., Busi N. *Stochastic simulation of biological systems with dynamical compartment structure. Lecture Notes in Computer Science* **4695**, p. 80–95, 2007.

[293] Versari C., Busi N. *Efficient stochastic simulation of biological systems with multiple variable volumes. Proceedings of* 1^{st} *Workshop From Biology To Concurrency and back - 07 (FBTC'07)* (N. Cannata, E. Merelli, eds.), Lisbon, Portugal, September 2007, *Electronic Notes in Theoretical Computer Science* **194(3)**, p. 165–180, Elsevier, January 2008.

[294] Versari C., Busi N. *Stochastic biological modelling in the presence of multiple compartments. Theoretical Computer Science* **410**, p. 3039–3064, 2009.

[295] Versari C., Gorrieri R. *π@: a π-based process calculus for the implementation of compartmentalised bio-inspired calculi. Lecture Notes in Computer Science* **5016**, p. 449–506, 2008.

[296] Vigliotti M.G. *Operational semantics for product-form solution. Lecture Notes in Computer Science* **7587**, p. 16–31, 2013.

[297] Vissat L.L., Hillston J., Marion G., Smith M.J. *MELA: modelling in ecology with location attributes. Proceedings of* 14^{th} *International Workshop on Quantitative Aspects of Programming Languages and Systems - 16 (QAPL'16)* (M. Tribastone, H. Wiklicky, eds.), Eindhoven, The Netherlands, April 2016, *Electronic Proceedings in Theoretical Computer Science* **227**, p. 82–97, Open Publishing Association, 2016.

[298] Wang D.Y.Q., Cardelli L., Phillips A., Piterman N., Fisher J. *Computational modeling of the EGFR network elucidates control mechanisms regulating signal dynamics. BMC Systems Biology* **3**, Article 118, 17 p., BioMed Central Ltd., December 2009.

[299] Wimmer R., Derisavi S., Hermanns H. *Symbolic partition refinement with automatic balancing of time and space. Performance Evaluation* **67**, p. 816–836, 2010.

[300] Wolf V. *Equivalences on phase type processes. Ph.D. thesis*, 202 p., University of Mannheim, Mannheim, Germany, April 2008.

[301] Yi W. *Real-time behaviour of asynchronous agents. Lecture Notes in Computer Science* **458**, p. 502–520, 1990.

[302] Yi W. *CCS + time = an interleaving model for real time systems. Lecture Notes in Computer Science* **510**, p. 217–228, 1991.

[303] Zavattaro G. *A gentle introduction to Stochastic (Poly)Automata Collectives and the (Bio)Chemical Ground Form. Lecture Notes in Computer Science* **5016**, p. 507–523, 2008.

[304] Zhou C., Wang J., Ravn A.P. *A formal description of hybrid systems. Lecture Notes in Computer Science* **1066**, p. 511–530, 1996.

[305] Zijal R. *Discrete time deterministic and stochastic Petri nets. Proceedings of the International Workshop on Quality of Communication-Based Systems - 94* (G. Hommel, ed.), Technical University of Berlin, Germany, September 1994, p. 123–136, Kluwer Academic Publishers, 1995.

[306] Zijal R. *Analysis of discrete time deterministic and stochastic Petri nets. Ph.D. thesis*, Technical University of Berlin, Germany, October 1997.

[307] Zijal R., Ciardo G. *Discrete deterministic and stochastic Petri nets. ICASE Report* **96-72**, Institute for Computer Applications in Science and Engineering (ICASE), NASA, Langley Research Centre, Hampton, VA, USA, i+23 p., 1996, `http://www.cs.odu.edu/~mln/ltrs-pdfs/icase-1996-72.pdf`, `http://www.dtic.mil/dtic/tr/fulltext/u2/a322409.pdf`.

[308] Zijal R., Ciardo G., Hommel G. *Discrete deterministic and stochastic Petri nets. Proceedings of* 9^{th} *ITG/GI Professional Meeting on Measuring, Modeling and Evaluation of Computer and Communication Systems - 97 (MMB'97)* (K. Irmscher, Ch. Mittasch, K. Richter, eds.), Freiberg, Germany, September 1997, Volume **1**, p. 103–117, VDE-Verlag, Berlin, Germany, 1997, `http://www.cs.ucr.edu/~ciardo/pubs/1997MMB-DDSPN.pdf`.

[309] Zijal R., German R. *A new approach to discrete time stochastic Petri nets. Proceedings of* 11^{th} *International Conference on Analysis and Optimization of Systems, Discrete Event Systems - 94 (DES'94)* (G. Cohen, J.-P. Quadrat, eds.), Sophia-Antipolis, France, June 1994, *Lecture Notes in Control and Information Sciences* **199**, p. 198–204, Springer, 1994.

[310] Zimmermann A. *Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. Proceedings of* 6^{th} *International ICST Conference on Performance Evaluation Methodologies and Tools - 12 (VALUETOOLS'12)* (B. Gaujal, A. Jean-Marie, E. Jorswieck, A. Seuret, eds.), Cargèse, France, October 2012, 10 p., IEEE Computer Society Press, November 2012, `https://www.tu-ilmenau.de/fileadmin/public/sse/Veroeffentlichungen/2012/VALUETOOLS2012.pdf`.

[311] Zimmermann A., Freiheit J., German R., Hommel G. *Petri net modelling and performability evaluation with TimeNET 3.0. Lecture Notes in Computer Science* **1786**, p. 188–202, 2000.

[312] Zimmermann A., Freiheit J., Hommel G. *Discrete time stochastic Petri nets for modeling and evaluation of real-time systems. Proceedings of Workshop on Parallel and Distributed Real Time Systems*, San Francisco, USA, 6 p., 2001, `http://pdv.cs.tu-berlin.de/~azi/texte/WPDRTS01.pdf`.

A Proofs

A.1 Proof of Theorem 5.2

Let $\mathbf{P}_r$ be the reordered (by moving vanishing states to the first positions) TPM for $DTMC(G)$. Like in Section 5, we reorder the states from $DR(G)$ so that the first rows and columns of $\mathbf{P}_r$ will correspond to the states from

$DR_V(G)$ and the last ones will correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. Then the reordered TPM for $DTMC(G)$ can be decomposed as

$$\mathbf{P}_r = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}.$$

The elements of the $(n-m) \times (n-m)$ submatrix $\mathbf{C}$ are the probabilities to move from vanishing to vanishing states, and those of the $(n-m) \times m$ submatrix $\mathbf{D}$ are the probabilities to move from vanishing to tangible states. The elements of the $m \times (n-m)$ submatrix $\mathbf{E}$ are the probabilities to move from tangible to vanishing states, and those of the $m \times m$ submatrix $\mathbf{F}$ are the probabilities to move from tangible to tangible states.

The TPM $\mathbf{P}^\diamond$ for $RDTMC(G)$ is the $m \times m$ matrix, calculated as

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD},$$

where the elements of the matrix $\mathbf{G} = \sum_{k=0}^{\infty} \mathbf{C}^k$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states, in $DTMC(G)$. We define the matrix $\mathbf{H} = \mathbf{EGD}$. For $s, \tilde{s} \in DR_T(G)$, let $PM_F(s, \tilde{s})$ and $PM_H(s, \tilde{s})$ be the probabilities to change from s to $\tilde{s}$ for the submatrix $\mathbf{F}$ and matrix $\mathbf{H}$, respectively.

In a similar way, the reordered TPM for $EDTMC(G)$ can be decomposed as

$$\mathbf{P}_r^* = \begin{pmatrix} \mathbf{C}^* & \mathbf{D}^* \\ \mathbf{E}^* & \mathbf{F}^* \end{pmatrix}.$$

The elements of the submatrices of $\mathbf{P}_r^*$ are described like those of the submatrices of $\mathbf{P}_r$.

The TPM $(\mathbf{P}^*)^\diamond$ for $REDTMC(G)$ is the $m \times m$ matrix, calculated as

$$(\mathbf{P}^*)^\diamond = \mathbf{F}^* + \mathbf{E}^*\mathbf{G}'\mathbf{D}^*,$$

where the elements of the matrix $\mathbf{G}' = \sum_{k=0}^{\infty} (\mathbf{C}^*)^k$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states, in $EDTMC(G)$. We define the matrix $\mathbf{H}' = \mathbf{E}^*\mathbf{G}'\mathbf{D}^*$. For $s, \tilde{s} \in DR_T(G)$, let $PM_{H'}(s, \tilde{s})$ be the probability to change from s to $\tilde{s}$ for the matrix $\mathbf{H}'$.

From the proof of Theorem 5.1, we have $\mathbf{P}_r^* = Diag(SL_r)(\mathbf{P}_r - \mathbf{I}) + \mathbf{I}$, where SL_r is the reordered (by moving vanishing states to the first positions) self-loops abstraction vector of G in $DTMC(G)$. Let SL_C and SL_F be the self-loops abstraction subvectors of G for the submatrices $\mathbf{C}$ and $\mathbf{F}$, respectively, i.e. the "head" of length $n-m$ and the "tail" of length m, taken from the vector SL_r, with the following elements: $\forall s \in DR_V(G)\ SL_C(s) = SL_r(s)$ and $\forall s \in DR_T(G)\ SL_F(s) = SL_r(s)$. Then we have

$$\mathbf{P}_r^* = \begin{pmatrix} Diag(SL_C) & \mathbf{0} \\ \mathbf{0} & Diag(SL_F) \end{pmatrix} \begin{pmatrix} \mathbf{C}-\mathbf{I} & \mathbf{D} \\ \mathbf{E} & \mathbf{F}-\mathbf{I} \end{pmatrix} + \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} Diag(SL_C)(\mathbf{C}-\mathbf{I})+\mathbf{I} & Diag(SL_C)\mathbf{D} \\ Diag(SL_F)\mathbf{E} & Diag(SL_F)(\mathbf{F}-\mathbf{I})+\mathbf{I} \end{pmatrix}.$$

Hence, $\mathbf{C}^* = Diag(SL_C)(\mathbf{C}-\mathbf{I})+\mathbf{I}$, $\mathbf{D}^* = Diag(SL_C)\mathbf{D}$, $\mathbf{E}^* = Diag(SL_F)\mathbf{E}$, $\mathbf{F}^* = Diag(SL_F)(\mathbf{F}-\mathbf{I})+\mathbf{I}$.

Then $(\mathbf{P}^*)^\diamond = \mathbf{F}^* + \mathbf{E}^*\mathbf{G}'\mathbf{D}^* = Diag(SL_F)(\mathbf{F}-\mathbf{I}) + \mathbf{I} + Diag(SL_F)\mathbf{E}\mathbf{G}'Diag(SL_C)\mathbf{D} = Diag(SL_F)((\mathbf{F} + \mathbf{E}\mathbf{G}'Diag(SL_C)\mathbf{D})-\mathbf{I})+\mathbf{I}$. Let us explore the matrix $\mathbf{G}'Diag(SL_C)$. The matrix $\mathbf{G}'$ can have two different forms, depending on whether the loops among vanishing states exist in $EDTMC(G)$, hence, we consider two cases.

1. There exist *no loops among vanishing states* in $EDTMC(G)$. We have $\exists l \in I\!N\ \forall k > l\ (\mathbf{C}^*)^k = \mathbf{0}$ and $\mathbf{G}' = \sum_{k=0}^{l} (\mathbf{C}^*)^k$.

   Then there are no loops among *different* vanishing states in $DTMC(G)$ (but self-loops may exist in vanishing states), since no loop among *different* states is removed and all self-loops (in the non-absorbing states) are removed in $EDTMC(G)$, with respect to $DTMC(G)$.

   Let there are no self-loops in vanishing states in $DTMC(G)$. In such a case, $\forall s \in DT_V(G)\ SL_C(s) = SL(s) = 1$ and $Diag(SL_C) = \mathbf{I}$. We have $\mathbf{C}^* = Diag(SL_C)(\mathbf{C}-\mathbf{I}) + \mathbf{I} = \mathbf{I}(\mathbf{C}-\mathbf{I}) + \mathbf{I} = \mathbf{C}$ and $\mathbf{G}' = \sum_{k=0}^{l} (\mathbf{C}^*)^k = \sum_{k=0}^{l} \mathbf{C}^k = \mathbf{G}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{G}\mathbf{I} = \mathbf{G}$.

   Let there are self-loops in vanishing states in $DTMC(G)$. In such a case, $\mathbf{G} = (\mathbf{I}-\mathbf{C})^{-1}$. Note that $\mathbf{C} \neq \mathbf{I} \neq \mathbf{C}^*$, since there exist no absorbing vanishing states in $DTMC(G)$. It is easy to prove by induction on $l \in I\!N$ that $\mathbf{G}'(\mathbf{I}-\mathbf{C}^*) = \left(\sum_{k=0}^{l} (\mathbf{C}^*)^k\right)(\mathbf{I}-\mathbf{C}^*) = \mathbf{I} - (\mathbf{C}^*)^{l+1}$. Since $(\mathbf{C}^*)^{l+1} = \mathbf{0}$, we get $\mathbf{G}'(\mathbf{I}-\mathbf{C}^*) = \mathbf{I}-\mathbf{0} = \mathbf{I}$. In a similar way, we show that $(\mathbf{I}-\mathbf{C}^*)\mathbf{G}' = \mathbf{I}$. We have $\lim_{k\to\infty}(\mathbf{C}^*)^k = \mathbf{0}$. Hence, $\mathbf{G}' = (\mathbf{I}-\mathbf{C}^*)^{-1} = (\mathbf{I} - Diag(SL_C)(\mathbf{C}-\mathbf{I}) - \mathbf{I})^{-1} = (Diag(SL_C)(\mathbf{I}-\mathbf{C}))^{-1} = (\mathbf{I}-\mathbf{C})^{-1}Diag(SL_C)^{-1} = \mathbf{G}Diag(SL_C)^{-1}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{G}Diag(SL_C)^{-1}Diag(SL_C) = \mathbf{G}$.

2. There exist *loops among vanishing states* in $EDTMC(G)$. We have $\lim_{k\to\infty}(\mathbf{C}^*)^k = \mathbf{0}$ and $\mathbf{G}' = (\mathbf{I}-\mathbf{C}^*)^{-1}$.

   Then there are loops among vanishing states in $DTMC(G)$, since no loop among states is removed and self-loops are possibly added in $DTMC(G)$, with respect to $EDTMC(G)$. Hence, $\lim_{k\to\infty}(\mathbf{C})^k = \mathbf{0}$ and $\mathbf{G} = (\mathbf{I}-\mathbf{C})^{-1}$.

   We have $\mathbf{G}' = (\mathbf{I}-\mathbf{C}^*)^{-1} = (\mathbf{I} - Diag(SL_C)(\mathbf{C}-\mathbf{I}) - \mathbf{I})^{-1} = (Diag(SL_C)(\mathbf{I}-\mathbf{C}))^{-1} = (\mathbf{I}-\mathbf{C})^{-1}Diag(SL_C)^{-1} = \mathbf{G}Diag(SL_C)^{-1}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{G}Diag(SL_C)^{-1}Diag(SL_C) = \mathbf{G}$.

In the both cases above, we get $\mathbf{G}'Diag(SL_C) = \mathbf{G}$. Hence, $(\mathbf{P}^*)^\diamond = Diag(SL_F)((\mathbf{F}+\mathbf{E}\mathbf{G}'Diag(SL_C)\mathbf{D}) - \mathbf{I}) + \mathbf{I} = Diag(SL_F)((\mathbf{F}+\mathbf{E}\mathbf{G}\mathbf{D}) - \mathbf{I}) + \mathbf{I} = Diag(SL_F)(\mathbf{P}^\diamond - \mathbf{I}) + \mathbf{I}$.

Let $s, \tilde{s} \in DR_T(G)$. The EDTMC for $RDTMC(G)$ is denoted by $ERDTMC(G)$ and has the probabilities $(PM^\diamond)^*(s,\tilde{s})$ to change from s to $\tilde{s}$. The RDTMC for $EDTMC(G)$ is denoted by $REDTMC(G)$ and has the probabilities $(PM^*)^\diamond(s,\tilde{s})$ to change from s to $\tilde{s}$. The EDTMC for $REDTMC(G)$ is denoted by $EREDTMC(G)$ and has the probabilities $((PM^*)^\diamond)^*(s,\tilde{s})$ to change from s to $\tilde{s}$.

Further, let SL_H and $SL_{H'}$ be the self-loops abstraction vectors of G for the matrices $\mathbf{H}$ and $\mathbf{H}'$, respectively. We have $(\mathbf{P}^*)^\diamond = \mathbf{F}^* + \mathbf{H}' = \mathbf{F}^* + Diag(SL_F)\mathbf{E}\mathbf{G}\mathbf{D} = \mathbf{F}^* + Diag(SL_F)\mathbf{H}$. Hence, $\mathbf{H}' = Diag(SL_F)\mathbf{H}$ and $\forall s, \tilde{s} \in DR_T(G)\ PM_{H'}(s,\tilde{s}) = SL_F(s)PM_H(s,\tilde{s})$. Since there are no self-loops in $\mathbf{F}^*$, we conclude that $(SL^*)^\diamond = SL_{H'}$ is the self-loops abstraction vector of G in $REDTMC(G)$.

- Let $PM_F(s,s) + PM_H(s,s) = PM^\diamond(s,s) < 1$ and $PM_F(s,s), PM_H(s,s) > 0$, i.e. s is non-absorbing in $RDTMC(G)$ and there exist self-loops associated with s in $DTMC(G)$ and *extra* self-loops (in addition to those inherited from $DTMC(G)$) in $RDTMC(G)$.

  In $ERDTMC(G)$, we have $(PM^\diamond)^*(s,\tilde{s}) = SL^\diamond(s)PM^\diamond(s,\tilde{s}) = \frac{PM^\diamond(s,\tilde{s})}{1-PM^\diamond(s,s)} = \frac{PM^\diamond(s,\tilde{s})}{1-PM_F(s,s)-PM_H(s,s)} = \frac{\frac{PM^\diamond(s,\tilde{s})}{1-PM_F(s,s)}}{1-\frac{PM_H(s,s)}{1-PM_F(s,s)}} = \frac{SL_F(s)PM^\diamond(s,\tilde{s})}{1-SL_F(s)PM_H(s,s)}$. Then the self-loops abstraction factor in s in $RDTMC(G)$ is $SL^\diamond(s) = \frac{SL_F(s)}{1-SL_F(s)PM_H(s,s)} = SL_F(s)SL_{H'}(s)$, where $SL_{H'}(s) = \frac{1}{1-SL_F(s)PM_H(s,s)}$ is the self-loops abstraction factor in s in $REDTMC(G)$. Thus, $(PM^\diamond)^*(s,\tilde{s}) = SL_F(s)SL_{H'}(s)PM^\diamond(s,\tilde{s})$.

  In $EREDTMC(G)$, we have $((PM^*)^\diamond)^*(s,\tilde{s}) = (SL^*)^\diamond(s)(PM^*)^\diamond(s,\tilde{s}) = SL_{H'}(s)(PM^*)^\diamond(s,\tilde{s}) = SL_{H'}(s)SL_F(s)PM^\diamond(s,\tilde{s}) = (PM^\diamond)^*(s,\tilde{s})$.

  The other three cases (no self-loops associated with s in $DTMC(G)$, no extra self-loops associated with s in $RDTMC(G)$, or no *any* self-loops associated with s in $RDTMC(G)$) are treated analogously, by replacing $PM_F(s,s)$ or/and $PM_H(s,s)$ with zeros.

- Let $PM_F(s,s) + PM_H(s,s) = PM^\diamond(s,s) = 1$ and $PM_F(s,s), PM_H(s,s) > 0$, i.e. s is absorbing in $RDTMC(G)$ and there exist self-loops associated with s in $DTMC(G)$ and *extra* self-loops (in addition to those inherited from $DTMC(G)$) in $RDTMC(G)$.

  In $ERDTMC(G)$, we have $(PM^\diamond)^*(s,s) = 1$ by definition of the EDTMC, since $PM^\diamond(s,s) = 1$.

  In $REDTMC(G)$, the probability of a self-loop associated with s is $(PM^*)^\diamond(s,s) = PM_{H'}(s,s) = SL_F(s)PM_H(s,s) = \frac{PM_H(s,s)}{1-PM_F(s,s)} = \frac{1-PM_F(s,s)}{1-PM_F(s,s)} = 1$.

  In $EREDTMC(G)$, we have $((PM^*)^\diamond)^*(s,s) = 1 = (PM^\diamond)^*(s,s)$ by definition of the EDTMC, since $(PM^*)^\diamond(s,s) = 1$.

  The other three cases (no self-loops associated with s in $DTMC(G)$, no extra self-loops associated with s in $RDTMC(G)$, or no *any* self-loops associated with s in $RDTMC(G)$) are treated analogously, by replacing $PM_F(s,s)$ or/and $PM_H(s,s)$ with zeros.

Thus, $((\mathbf{P}^*)^\diamond)^* = (\mathbf{P}^\diamond)^*$ and $EREDTMC(G) = ERDTMC(G)$. □

A.2 Proof of Proposition 6.2

Like it has been done for strong equivalence in Proposition 8.2.1 from [164], we shall prove the following fact about step stochastic bisimulation. Let us have $\forall j \in \mathcal{J}\ \mathcal{R}_j : G \underline{\leftrightarrow}_{ss} G'$ for some index set $\mathcal{J}$. Then the transitive closure of the union of all relations $\mathcal{R} = (\cup_{j\in\mathcal{J}}\mathcal{R}_j)^+$ is also an equivalence and $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$.

Since $\forall j \in \mathcal{J}\ \mathcal{R}_j$ is an equivalence, by definition of $\mathcal{R}$, we get that $\mathcal{R}$ is also an equivalence.

Let $j \in \mathcal{J}$, then, by definition of $\mathcal{R}$, $(s_1,s_2) \in \mathcal{R}_j$ implies $(s_1,s_2) \in \mathcal{R}$. Hence, $\forall \mathcal{H}_{jk} \in (DR(G) \cup DR(G'))/_{\mathcal{R}_j}$ $\exists \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \mathcal{H}_{jk} \subseteq \mathcal{H}$. Moreover, $\exists \mathcal{J}'\ \mathcal{H} = \cup_{k\in\mathcal{J}'}\mathcal{H}_{jk}$.

We denote $\mathcal{R}(n) = (\cup_{j\in\mathcal{J}}\mathcal{R}_j)^n$. Let $(s_1,s_2) \in \mathcal{R}$, then, by definition of $\mathcal{R}$, $\exists n > 0\ (s_1,s_2) \in \mathcal{R}(n)$. We shall prove that $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$ by induction on n.

It is clear that $\forall j \in \mathcal{J}\ \mathcal{R}_j : G \underline{\leftrightarrow}_{ss} G'$ implies $\forall j \in \mathcal{J}\ ([G]_\approx, [G']_\approx) \in \mathcal{R}_j$ and we have $([G]_\approx, [G']_\approx) \in \mathcal{R}$ by definition of $\mathcal{R}$.

It remains to prove that $(s_1, s_2) \in \mathcal{R}$ implies $SJ(s_1) = 0 \ \Leftrightarrow\ SJ(s_2) = 0$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ $\forall A \in \mathbb{N}^{\mathcal{L}}_{fin}\ PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$.

- $n = 1$

  In this case, $(s_1, s_2) \in \mathcal{R}$ implies $\exists j \in \mathcal{J}\ (s_1, s_2) \in \mathcal{R}_j$. Since $\mathcal{R}_j : G \underline{\leftrightarrow}_{ss} G'$, we get $SJ(s_1) = 0 \ \Leftrightarrow$ $SJ(s_2) = 0$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \forall A \in \mathbb{N}^{\mathcal{L}}_{fin}$

$$PM_A(s_1, \mathcal{H}) = \sum_{k \in \mathcal{J}'} PM_A(s_1, \mathcal{H}_{jk}) = \sum_{k \in \mathcal{J}'} PM_A(s_2, \mathcal{H}_{jk}) = PM_A(s_2, \mathcal{H}).$$

- $n \to n + 1$

  Suppose that $\forall m \leq n\ (s_1, s_2) \in \mathcal{R}(m)$ implies $SJ(s_1) = 0 \ \Leftrightarrow\ SJ(s_2) = 0$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ $\forall A \in \mathbb{N}^{\mathcal{L}}_{fin}\ PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$.

  Then $(s_1, s_2) \in \mathcal{R}(n+1)$ implies $\exists j \in \mathcal{J}\ (s_1, s_2) \in \mathcal{R}_j \circ \mathcal{R}(n)$, i.e. $\exists s_3 \in (DR(G) \cup DR(G'))$ such that $(s_1, s_3) \in \mathcal{R}_j$ and $(s_3, s_2) \in \mathcal{R}(n)$.

  Then, like for the case $n = 1$, we get $SJ(s_1) = 0 \ \Leftrightarrow\ SJ(s_3) = 0$ and $PM_A(s_1, \mathcal{H}) = PM_A(s_3, \mathcal{H})$. By the induction hypothesis, we get $SJ(s_3) = 0 \ \Leftrightarrow\ SJ(s_2) = 0$ and $PM_A(s_3, \mathcal{H}) = PM_A(s_2, \mathcal{H})$. Thus, $SJ(s_1) = 0 \ \Leftrightarrow\ SJ(s_3) = 0 \Leftrightarrow\ SJ(s_2) = 0$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \forall A \in \mathbb{N}^{\mathcal{L}}_{fin}$

$$PM_A(s_1, \mathcal{H}) = PM_A(s_3, \mathcal{H}) = PM_A(s_2, \mathcal{H}).$$

By definition, $\mathcal{R}_{ss}(G, G')$ is at least as large as the largest step stochastic bisimulation between G and G'. It follows from the proved above that $\mathcal{R}_{ss}(G, G')$ is an equivalence and $\mathcal{R}_{ss}(G, G') : G \underline{\leftrightarrow}_{ss} G'$, hence, it is the largest step stochastic bisimulation between G and G'. □

A.3 Proof of Theorem 6.1

At some points, the present proof for dtsdPBC goes along the lines from the respective proofs for PBC [41, 40], tPBC [183] and sPBC [201].

Let $N = Box_{dtsd}(E)$. We define a relation $\mathcal{R} = (\{([G]_\approx, Q_G), (Q_G, [G]_\approx) \mid [G]_\approx \in DR(\overline{E}),\ (N, Q_G) = Box_{dtsd}(G)\})^+$, where $^+$ is the transitive closure operation. It is easy to see that $\mathcal{R}$ is equivalence, since by construction it is symmetric, transitive and reflexive (just apply transitivity to each pair $([G]_\approx, Q_G), (Q_G, [G]_\approx)$). We shall demonstrate that $\mathcal{R} : TS(\overline{E}) \underline{\leftrightarrow}_{ss} RG(\underline{Box_{dtsd}(\overline{E})})$.

Clearly, $[\overline{E}]_\approx \in DR(\overline{E})$ and $Box_{dtsd}(\overline{E}) = \overline{Box_{dtsd}(E)} = \overline{N} = (N, Q_{\overline{N}}) = (N, Q_{\overline{E}})$. Hence, $([\overline{E}]_\approx, Q_{\overline{E}}) \in \mathcal{R}$.

It remains to check the step stochastic bisimulation transfer property. Let $([G]_\approx, Q_G) \in \mathcal{R}$. By Proposition 3.1, we can suppose that $G \in SaOpRegDynExpr$, i.e. all enabled waiting multiactions from G (even those not overlined or underlined) have the consistent timer value superscripts, which is very important when composing the subexpressions.

Then for a process state $[G]_\approx \in DR(\overline{E})$, the related net state $Q_G = (M_G, V_G) \in RS(\overline{N})$ is consistent and has the following properties. First, M_G is the marking of the marked and clocked dtsd-box $(N, (M_G, V_G)) = Box_{dtsd}(G)$ (which is an LDTSDPN, since $G \in SaOpRegDynExpr$). Second, by construction of the timer valuation functions V_G and I_G, for each waiting transition $t \in Tw_N$ with $\Lambda_N(t) = \varrho_{(\alpha, \natural_l^\theta)}$, if $(\alpha, \natural_l^\theta) \in \mathcal{WL}(G)$ then we have $V_G(t) = I_G((\alpha, \natural_l^\theta))$. Otherwise, if $(\alpha, \natural_l^\theta) \notin \mathcal{WL}(G)$ then either t is obtained from a relabeling f of some transition $v \in T_N$, and we have $V_G(t) = V_H(v)$ for a subexpression $H[f]$ of G; or t is resulted from synchronization on an action a of some transitions $v, w \in T_N$ and we have $V_G(t) = \max\{V_H(v), V_H(w)\}$ for a subexpression H sy a of G. In the both cases, $V_G(t)$ is completely defined by the timer valuation function V_H, applied to some transitions of the marked and clocked dtsd-box $Box_{dtsd}(H)$. Then by induction of the expression structure, we can finally prove that $V_G(t)$ is completely defined by I_G, applied to some waiting multiactions from $\mathcal{WL}(G)$. Note that any waiting multiaction affected by restriction in G has no corresponding transition in $Box_{dtsd}(G)$. Therefore, I_G (hence, $[G]_\approx$) may contain even more information (namely, the timer values of the restricted waiting multiactions) than needed to define V_G. Thus, several process states (which differ just in the timer value superscripts of the restricted waiting multiactions) may be related to one net state, as the example above this theorem demonstrates.

Let us prove by induction on the structure of dynamic expressions and corresponding dtsd-boxes that $Exec([G]_\approx)$ and $Fire(Q_G)$ are isomorphic. This means that for every $\Upsilon \in Exec([G]_\approx)$ there exists $U \in Fire(Q_G)$ such that U consists of the transitions *corresponding* to the activities from Υ and vice versa: $(\alpha, \kappa)_\iota \in$

$\Upsilon \Leftrightarrow t_\iota \in U$, where $\Lambda_N(t_\iota) = \varrho_{(\alpha,\kappa)}$. Thus, the *corresponding* activities and transitions have the same probabilities (in case of stochastic multiactions and transitions), or delays and weights (in case of deterministic multiactions and transitions), as well as the same multiaction labels and numberings. We can write $U = U(\Upsilon)$ and $\Upsilon = \Upsilon(U)$, to indicate such a correspondence.

Actually, each Υ and the *corresponding* U are completely defined by the sets of their numberings $Num(\Upsilon) = \{\iota \mid (\alpha,\kappa)_\iota \in \Upsilon\} = \{\iota \mid t_\iota \in U\} = Num(U)$, since each activity and transition have a unique numbering. Moreover, $Exec([G]_\approx)$ and $Fire(Q_G)$ are completely defined by the sets of their numberings $Num(Exec([G]_\approx)) = \{Num(\Upsilon) \mid \Upsilon \in Exec([G]_\approx)\} = \{Num(U) \mid U \in Fire(Q_G)\} = Num(Fire(Q_G))$.

- If $final(G)$ then $G \approx \underline{E}$, $stang([G]_\approx)$ and $Exec([G]_\approx) = Exec([\underline{E}]_\approx) = \{\emptyset\}$. On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(\underline{E}) = \underline{N} = (N, Q_{\underline{N}}) = (N, Q_{\underline{E}})$ and $Fire(Q_G) = Fire(Q_{\underline{E}}) = \{\emptyset\} = Exec([G]_\approx)$.

- If $G = \overline{(\alpha, \natural_l^\theta)_\iota^\delta}$ and $\theta \in \mathbb{N}_{\geq 2}$, $l \in \mathbb{R}_{>0}$, $\delta \in \{2, \ldots, \theta\}$, then $stang([G]_\approx)$ and $Exec([G]_\approx) = \{\emptyset\}$. On the other hand, $Box_{dtsd}(G) = (N_{(\alpha,\natural_l^\theta)_\iota^\delta}, ({}^\bullet t_\iota, (t_\iota, \delta)))$, where $\Lambda_N(t_\iota) = \varrho_{(\alpha,\natural_l^\theta)}$, and $Fire(Q_G) = Fire(({}^\bullet t_\iota, (t_\iota, \delta))) = \{\emptyset\} = Exec([G]_\approx)$.

- If $G = \overline{(\alpha, \rho)_\iota}$ and $\rho \in (0; 1)$ then $stang([G]_\approx)$ and $Exec([G]_\approx) = \{\emptyset, \{(\alpha, \rho)_\iota\}\}$. On the other hand, $Box_{dtsd}(G) = (N_{(\alpha,\rho)_\iota}, ({}^\bullet t_\iota, (t_\iota, \emptyset)))$, where $\Lambda_N(t_\iota) = \varrho_{(\alpha,\rho)}$, and $Fire(Q_G) = Fire(({}^\bullet t_\iota, (t_\iota, \emptyset))) = \{\emptyset, \{t_\iota\}\}$, which is isomorphic to $Exec([G]_\approx)$.

- If $G = \overline{(\alpha, \natural_l^0)_\iota}$ and $l \in \mathbb{R}_{>0}$ then $vanish([G]_\approx)$ and $Exec([G]_\approx) = \{\{(\alpha, \natural_l^0)_\iota\}\}$. On the other hand, $Box_{dtsd}(G) = (N_{(\alpha,\natural_l^0)_\iota}, ({}^\bullet t_\iota, (t_\iota, \emptyset)))$, where $\Lambda_N(t_\iota) = \varrho_{(\alpha,\natural_l^0)}$, and $Fire(Q_G) = Fire(({}^\bullet t_\iota, (t_\iota, \emptyset))) = \{\{t_\iota\}\}$, which is isomorphic to $Exec([G]_\approx)$.

- If $G = \overline{(\alpha, \natural_l^\theta)_\iota^1}$ and $\theta \in \mathbb{N}_{\geq 1}$, $l \in \mathbb{R}_{>0}$, then $wtang([G]_\approx)$ and $Exec([G]_\approx) = \{\{(\alpha, \natural_l^\theta)_\iota\}\}$. On the other hand, $Box_{dtsd}(G) = (N_{(\alpha,\natural_l^\theta)_\iota^1}, ({}^\bullet t_\iota, (t_\iota, 1)))$, where $\Lambda_N(t_\iota) = \varrho_{(\alpha,\natural_l^\theta)}$, and $Fire(Q_G) = Fire(({}^\bullet t_\iota, (t_\iota, 1))) = \{\{t_\iota\}\}$, which is isomorphic to $Exec([G]_\approx)$.

- If $G = H; E$, where $H \in SaOpRegDynExpr$, $E \in RegStatExpr$, then

$$Exec([H; E]_\approx) = \begin{cases} Exec([H]_\approx), & \neg final(H); \\ Exec([\overline{E}]_\approx) & final(H). \end{cases}$$

  On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(H; E) = (Box_{dtsd}(\lfloor H \rfloor; E), Q_{H;E})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, $Box_{dtsd}(\overline{E}) = \overline{N_E} = (N_E, Q_{\overline{N_E}})$, we have

$$Fire(Q_{H;E}) = \begin{cases} Fire(Q_H), & M_H \neq N_H^\circ; \\ Fire(Q_{\overline{N_E}}), & M_H = N_H^\circ; \end{cases}$$

  which is isomorphic to $Exec([H; E]_\approx)$.

- If $G = E; H$, where $E \in RegStatExpr$, $H \in SaOpRegDynExpr$, then

$$Exec([E; H]_\approx) = Exec([H]_\approx).$$

  On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(E; H) = (Box_{dtsd}(E; \lfloor H \rfloor), Q_{E;H})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, we have

$$Fire(Q_{E;H}) = Fire(Q_H);$$

  which is isomorphic to $Exec([E; H]_\approx)$.

- If $G = H[]E$, where $H \in SaOpRegDynExpr$, $E \in RegStatExpr$, then

$$Exec([H[]E]_\approx) = \begin{cases} Exec([H]_\approx), & \neg init(H) \vee \\ & (init(H) \wedge wtang([H]_\approx) \wedge stang([\overline{E}]_\approx)) \vee \\ & (init(H) \wedge vanish([H]_\approx) \wedge tang([\overline{E}]_\approx)); \\ Exec([\overline{E}]_\approx), & (init(H) \wedge stang([H]_\approx) \wedge wtang([\overline{E}]_\approx)) \vee \\ & (init(H) \wedge tang([H]_\approx) \wedge vanish([\overline{E}]_\approx)); \\ Exec([H]_\approx) \cup Exec([\overline{E}]_\approx), & (init(H) \wedge stang([H]_\approx) \wedge stang([\overline{E}]_\approx)) \vee \\ & (init(H) \wedge wtang([H]_\approx) \wedge wtang([\overline{E}]_\approx)) \vee \\ & (init(H) \wedge vanish([H]_\approx) \wedge vanish([\overline{E}]_\approx)). \end{cases}$$

On the other hand, $Box_{dtsd}(G) = \underline{Box_{dtsd}}(H[]E) = (Box_{dtsd}(\lfloor H \rfloor[]E), Q_{H[]E})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, $Box_{dtsd}(\overline{E}) = \overline{N_E} = (N_E, Q_{\overline{N_E}})$, we have

$$Fire(Q_{H[]E}) = \begin{cases} Fire(Q_H), & M_H \neq {}^\circ N_H \vee \\ & (M_H = {}^\circ N_H \wedge wtang(Q_H) \wedge stang(Q_{\overline{N_E}})) \vee \\ & (M_H = {}^\circ N_H \wedge vanish(Q_H) \wedge tang(Q_{\overline{N_E}})); \\ Fire(Q_{\overline{N_E}}), & (M_H = {}^\circ N_H \wedge stang(Q_H) \wedge wtang(Q_{\overline{N_E}})) \vee \\ & (M_H = {}^\circ N_H \wedge tang(Q_H) \wedge vanish(Q_{\overline{N_E}})); \\ Fire(Q_H) \cup Fire(Q_{\overline{N_E}}), & (M_H = {}^\circ N_H \wedge stang(Q_H) \wedge stang(Q_{\overline{N_E}})) \vee \\ & (M_H = {}^\circ N_H \wedge wtang(Q_H) \wedge wtang(Q_{\overline{N_E}})) \vee \\ & (M_H = {}^\circ N_H \wedge vanish(Q_H) \wedge vanish(Q_{\overline{N_E}})); \end{cases}$$

which is isomorphic to $Exec([H[]E]_\approx)$.

If $G = E[]H$, where $E \in RegStatExpr,\ H \in SaOpRegDynExpr$, then the constructions are similar.

- If $G = H \| Z$, where $H, Z \in SaOpRegDynExpr$, then

$$Exec([H\|Z]_\approx) = \begin{cases} Exec([H]_\approx), & (wtang([H]_\approx) \wedge stang([Z]_\approx)) \vee \\ & (vanish([H]_\approx) \wedge tang([Z]_\approx)); \\ Exec([Z]_\approx), & (stang([H]_\approx) \wedge wtang([Z]_\approx)) \vee \\ & (tang([H]_\approx) \wedge vanish([Z]_\approx)); \\ Exec([H]_\approx) \odot Exec([Z]_\approx), & wtang([H]_\approx) \wedge wtang([Z]_\approx); \\ Exec([H]_\approx) \cup Exec([Z]_\approx) \cup & \\ (Exec([H]_\approx) \odot Exec([Z]_\approx)), & (stang([H]_\approx) \wedge stang([Z]_\approx)) \vee \\ & (vanish([H]_\approx) \wedge vanish([Z]_\approx)), \end{cases}$$

where $Exec([H]_\approx) \odot Exec([Z]_\approx) = \{\Upsilon + \Phi \mid \Upsilon \in Exec([H]_\approx),\ \Phi \in Exec([Z]_\approx)\}$.

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(H\|Z) = (Box_{dtsd}(\lfloor H \rfloor \| Z), Q_{H\|Z})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, $Box_{dtsd}(Z) = (Box_{dtsd}(\lfloor Z \rfloor), Q_Z)$, we have

$$Fire(Q_{H\|Z}) = \begin{cases} Fire(Q_H), & (wtang(Q_H) \wedge stang(Q_Z)) \vee (vanish(Q_H) \wedge tang(Q_Z)); \\ Fire(Q_Z), & (stang(Q_H) \wedge wtang(Q_Z)) \vee (tang(Q_H) \wedge vanish(Q_Z)); \\ Fire(Q_H) \odot Fire(Q_Z), & wtang(Q_H) \wedge wtang(Q_Z); \\ Fire(Q_H) \cup Fire(Q_Z) \cup & \\ (Fire(Q_H) \odot Fire(Q_Z)), & (stang(Q_H) \wedge stang(Q_Z)) \vee (vanish(Q_H) \wedge vanish(Q_Z)), \end{cases}$$

where $Fire(Q_H) \odot Fire(Q_Z) = \{U \cup T \mid U \in Fire(Q_H),\ T \in Fire(Q_Z)\}$; which is isomorphic to $Exec([H\|Z]_\approx)$.

- If $G = H[f]$, where $H \in SaOpRegDynExpr$, then

$$Exec([H[f]]_\approx) = \{f(\Upsilon) \mid \Upsilon \in Exec([H]_\approx)\}.$$

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(H[f]) = (Box_{dtsd}(\lfloor H \rfloor[f]), Q_{H[f]})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, we have

$$Fire(Q_{H[f]}) = \{f(U) \mid U \in Fire(Q_H)\},$$

where $f(U) = \{t_\iota \in U \mid \Lambda_H(t_\iota) = \varrho_{(\alpha,\kappa)},\ \Lambda_{H[f]}(t_\iota) = \varrho_{(f(\alpha),\kappa)}\}$; which is isomorphic to $Exec([H[f]]_\approx)$.

- If $G = H\ \mathsf{rs}\ a$, where $H \in SaOpRegDynExpr$, then

$$Exec([H\ \mathsf{rs}\ a]_\approx) = \{\Upsilon - \Upsilon_a \mid \Upsilon \in Exec([H]_\approx)\},$$

where $\Upsilon_a = \{(\alpha,\kappa)_\iota \in \Upsilon \mid (a \in \alpha) \vee (\hat{a} \in \alpha)\},\ a \in Act$.

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(H\ \mathsf{rs}\ a) = (Box_{dtsd}(\lfloor H \rfloor\ \mathsf{rs}\ a), Q_{H\ \mathsf{rs}\ a})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, we have

$$Fire(Q_{H\ \mathsf{rs}\ a}) = \{U \setminus U_a \mid U \in Fire(Q_H)\},$$

where $U_a = \{t_\iota \in U \mid \Lambda_H(t_\iota) = \varrho_{(\alpha,\kappa)},\ (a \in \alpha) \vee (\hat{a} \in \alpha)\},\ a \in Act$; which is isomorphic to $Exec([H\ \mathsf{rs}\ a]_\approx)$.

- If $G = H\ \mathsf{sy}\ a$, where $H \in SaOpRegDynExpr$, then

$$Exec([H\ \mathsf{sy}\ a]_\approx) = \begin{cases} Exec([H]_\approx) \cup \{\Upsilon + \{(\alpha \oplus_a \beta, \rho \cdot \chi)_{(\iota_1)(\iota_2)}\} \mid \\ \Upsilon + \{(\alpha,\rho)_{\iota_1}\} + \{(\beta,\chi)_{\iota_2}\} \in Exec([H]_\approx),\ a \in \alpha,\ \hat{a} \in \beta\}, & stang([H]_\approx); \\ Exec([H]_\approx) \cup \{\Upsilon + \{(\alpha \oplus_a \beta, \natural^\theta_{l+m})_{(\iota_1)(\iota_2)}\} \mid \\ \Upsilon + \{(\alpha, \natural^\theta_l)_{\iota_1}\} + \{(\beta, \natural^\theta_m)_{\iota_2}\} \in Exec([H]_\approx),\ a \in \alpha,\ \hat{a} \in \beta\}, & wtang([H]_\approx); \\ Exec([H]_\approx) \cup \{\Upsilon + \{(\alpha \oplus_a \beta, \natural^0_{l+m})_{(\iota_1)(\iota_2)}\} \mid \\ \Upsilon + \{(\alpha, \natural^0_l)_{\iota_1}\} + \{(\beta, \natural^0_m)_{\iota_2}\} \in Exec([H]_\approx),\ a \in \alpha,\ \hat{a} \in \beta\}, & vanish([H]_\approx). \end{cases}$$

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}(H\ \mathsf{sy}\ a) = (Box_{dtsd}(\lfloor H \rfloor\ \mathsf{sy}\ a), Q_{H\ \mathsf{sy}\ a})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, we have

$$Fire(Q_{H\ \mathsf{sy}\ a}) = \begin{cases} Fire(Q_H) \cup \{U \cup \{t_{(\iota_1)(\iota_2)}\} \mid \Lambda_{H\ \mathsf{sy}\ a}(t_{(\iota_1)(\iota_2)}) = \varrho_{(\alpha \oplus_a \beta, \rho \cdot \chi)}, \\ U \cup \{v_{\iota_1}, w_{\iota_2}\} \in Fire(Q_H),\ \Lambda_H(v_{\iota_1}) = \varrho_{(\alpha,\rho)},\ \Lambda_H(w_{\iota_2}) = \varrho_{(\beta,\chi)}, \\ a \in \alpha,\ \hat{a} \in \beta\}, & stang(Q_H); \\ Fire(Q_H) \cup \{U \cup \{t_{(\iota_1)(\iota_2)}\} \mid \Lambda_{H\ \mathsf{sy}\ a}(t_{(\iota_1)(\iota_2)}) = \varrho_{(\alpha \oplus_a \beta, \natural^\theta_{l+m})}, \\ U \cup \{v_{\iota_1}, w_{\iota_2}\} \in Fire(Q_H),\ \Lambda_H(v_{\iota_1}) = \varrho_{(\alpha, \natural^\theta_l)},\ \Lambda_H(w_{\iota_2}) = \varrho_{(\beta, \natural^\theta_m)}, \\ a \in \alpha,\ \hat{a} \in \beta\}, & wtang(Q_H); \\ Fire(Q_H) \cup \{U \cup \{t_{(\iota_1)(\iota_2)}\} \mid \Lambda_{H\ \mathsf{sy}\ a}(t_{(\iota_1)(\iota_2)}) = \varrho_{(\alpha \oplus_a \beta, \natural^0_{l+m})}, \\ U \cup \{v_{\iota_1}, w_{\iota_2}\} \in Fire(Q_H),\ \Lambda_H(v_{\iota_1}) = \varrho_{(\alpha, \natural^0_l)},\ \Lambda_H(w_{\iota_2}) = \varrho_{(\beta, \natural^0_m)}, \\ a \in \alpha,\ \hat{a} \in \beta\}, & vanish(Q_H); \end{cases}$$

which is isomorphic to $Exec([H\ \mathsf{sy}\ a]_\approx)$.

- If $G = [H * E * F]$, where $H \in SaOpRegDynExpr,\ E, F \in RegStatExpr$, then

$$Exec([[H * E * F]]_\approx) = \begin{cases} Exec([H]_\approx), & \neg final(H); \\ Exec([\overline{E}]_\approx), & (final(H) \wedge wtang([\overline{E}]_\approx) \wedge stang([\overline{F}]_\approx)) \vee \\ & (final(H) \wedge vanish([\overline{E}]_\approx) \wedge tang([\overline{F}]_\approx)); \\ Exec([\overline{F}]_\approx), & (final(H) \wedge stang([\overline{E}]_\approx) \wedge wtang([\overline{F}]_\approx)) \vee \\ & (final(H) \wedge tang([\overline{E}]_\approx) \wedge vanish([\overline{F}]_\approx)); \\ Exec([\overline{E}]_\approx) \cup Exec([\overline{F}]_\approx), & (final(H) \wedge stang([\overline{E}]_\approx) \wedge stang([\overline{F}]_\approx)) \vee \\ & (final(H) \wedge wtang([\overline{E}]_\approx) \wedge wtang([\overline{F}]_\approx)) \vee \\ & (final(H) \wedge vanish([\overline{E}]_\approx) \wedge vanish([\overline{F}]_\approx)). \end{cases}$$

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}([H * E * F]) = (Box_{dtsd}(\lfloor H \rfloor * E * F), Q_{[H*E*F]})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, $Box_{dtsd}(\overline{E}) = \overline{N_E} = (N_E, Q_{\overline{N_E}})$, $Box_{dtsd}(\overline{F}) = \overline{N_F} = (N_F, Q_{\overline{N_F}})$, we have

$$Fire(Q_{[H*E*F]}) = \begin{cases} Fire(Q_H), & M_H \neq N^\circ_H; \\ Fire(Q_{\overline{N_E}}), & (M_H = N^\circ_H \wedge wtang(Q_{\overline{N_E}}) \wedge stang(Q_{\overline{N_F}})) \vee \\ & (M_H = N^\circ_H \wedge vanish(Q_{\overline{N_E}}) \wedge tang(Q_{\overline{N_F}})); \\ Fire(Q_{\overline{N_F}}), & (M_H = N^\circ_H \wedge stang(Q_{\overline{N_E}}) \wedge wtang(Q_{\overline{N_F}})) \vee \\ & (M_H = N^\circ_H \wedge tang(Q_{\overline{N_E}}) \wedge vanish(Q_{\overline{N_F}})); \\ Fire(Q_{\overline{N_E}}) \cup Fire(Q_{\overline{N_F}}), & (M_H = N^\circ_H \wedge stang(Q_{\overline{N_E}}) \wedge stang(Q_{\overline{N_F}})) \vee \\ & (M_H = N^\circ_H \wedge wtang(Q_{\overline{N_E}}) \wedge wtang(Q_{\overline{N_F}})) \vee \\ & (M_H = N^\circ_H \wedge vanish(Q_{\overline{N_E}}) \wedge vanish(Q_{\overline{N_F}})); \end{cases}$$

which is isomorphic to $Exec([[H * E * F]]_\approx)$.

- If $G = [E * H * F]$, where $E, F \in RegStatExpr,\ H \in SaOpRegDynExpr$, then

$$Exec([[E{*}H{*}F]]_{\approx}) = \begin{cases} Exec([H]_{\approx}), & (\neg init(H) \wedge \neg final(H)) \vee \\ & ((init(H) \vee final(H)) \wedge wtang([H]_{\approx}) \wedge stang([\overline{F}]_{\approx})) \vee \\ & ((init(H) \vee final(H)) \wedge vanish([H]_{\approx}) \wedge tang([\overline{F}]_{\approx})); \\ Exec([\overline{F}]_{\approx}), & ((init(H) \vee final(H)) \wedge stang([H]_{\approx}) \wedge wtang([\overline{F}]_{\approx})) \vee \\ & ((init(H) \vee final(H)) \wedge tang([H]_{\approx}) \wedge vanish([\overline{F}]_{\approx})); \\ Exec([H]_{\approx}) \cup Exec([\overline{F}]_{\approx}), & ((init(H) \vee final(H)) \wedge stang([H]_{\approx}) \wedge stang([\overline{F}]_{\approx})) \vee \\ & ((init(H) \vee final(H)) \wedge wtang([H]_{\approx}) \wedge wtang([\overline{F}]_{\approx})) \vee \\ & ((init(H) \vee final(H)) \wedge vanish([H]_{\approx}) \wedge vanish([\overline{F}]_{\approx})). \end{cases}$$

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}([E * H * F]) = (Box_{dtsd}(E * \lfloor H \rfloor * F), Q_{[E*H*F]})$, and for $Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H),\ Box_{dtsd}(\overline{F}) = \overline{N_F} = (N_F, Q_{\overline{N_F}})$, we have

$$Fire(Q_{[E*H*F]}) = \begin{cases} Fire(Q_H), & (M_H \neq^{\circ} N_H \wedge M_H \neq N_H^{\circ}) \vee \\ & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge wtang(Q_H) \wedge stang(Q_{\overline{N_F}})) \vee \\ & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge vanish(Q_H) \wedge tang(Q_{\overline{N_F}})); \\ Fire(Q_{\overline{N_F}}), & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge stang(Q_H) \wedge wtang(Q_{\overline{N_F}})) \vee \\ & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge tang(Q_H) \wedge vanish(Q_{\overline{N_F}})); \\ Fire(Q_H) \cup Fire(Q_{\overline{N_F}}), & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge stang(Q_H) \wedge stang(Q_{\overline{N_F}})) \vee \\ & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge wtang(Q_H) \wedge wtang(Q_{\overline{N_F}})) \vee \\ & ((M_H =^{\circ} N_H \vee M_H = N_H^{\circ}) \wedge vanish(Q_H) \wedge vanish(Q_{\overline{N_F}})); \end{cases}$$

which is isomorphic to $Exec([[E * H * F]]_{\approx})$.

- If $G = [E * F * H]$, where $E, F \in RegStatExpr,\ H \in SaOpRegDynExpr$, then

$$Exec([[E * F * H]]_{\approx}) = \begin{cases} Exec([\overline{F}]_{\approx}), & (wtang([\overline{F}]_{\approx}) \wedge init(H) \wedge stang([H]_{\approx})) \vee \\ & (vanish([\overline{F}]_{\approx}) \wedge init(H) \wedge tang([H]_{\approx})); \\ Exec([H]_{\approx}), & \neg init(H) \vee \\ & (stang([\overline{F}]_{\approx}) \wedge init(H) \wedge wtang([H]_{\approx})) \vee \\ & (tang([\overline{F}]_{\approx}) \wedge init(H) \wedge vanish([H]_{\approx})); \\ Exec([\overline{F}]_{\approx}) \cup Exec([H]_{\approx}), & (stang([\overline{F}]_{\approx}) \wedge init(H) \wedge stang([H]_{\approx})) \vee \\ & (wtang([\overline{F}]_{\approx}) \wedge init(H) \wedge wtang([H]_{\approx})) \vee \\ & (vanish([\overline{F}]_{\approx}) \wedge init(H) \wedge vanish([H]_{\approx})). \end{cases}$$

On the other hand, $Box_{dtsd}(G) = Box_{dtsd}([E * F * H]) = (Box_{dtsd}(E * F * \lfloor H \rfloor), Q_{[E*F*H]})$, and for $Box_{dtsd}(\overline{F}) = \overline{N_F} = (N_F, Q_{\overline{N_F}}),\ Box_{dtsd}(H) = (Box_{dtsd}(\lfloor H \rfloor), Q_H)$, we have

$$Fire(Q_{[E*F*H]}) = \begin{cases} Fire(Q_{\overline{N_F}}), & (wtang(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge stang(Q_H)) \vee \\ & (vanish(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge tang(Q_H)); \\ Fire(Q_H), & M_H \neq^{\circ} N_H \vee \\ & (stang(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge wtang(Q_H)) \vee \\ & (tang(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge vanish(Q_H)); \\ Fire(Q_{\overline{N_F}}) \cup Fire(Q_H), & (stang(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge stang(Q_H)) \vee \\ & (wtang(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge wtang(Q_H)) \vee \\ & (vanish(Q_{\overline{N_F}}) \wedge M_H =^{\circ} N_H \wedge vanish(Q_H)); \end{cases}$$

which is isomorphic to $Exec([[E * F * H]]_{\approx})$.

Thus, we have proved that $Exec([G]_{\approx})$ and $Fire(Q_G)$ are isomorphic. Note that the probability functions $PF(\Upsilon, [G]_{\approx})$ and $PT(\Upsilon, [G]_{\approx})$ depend only on the structure of $Exec([G]_{\approx})$, as well as on the probabilities of stochastic multiactions and weights of deterministic multiactions from its elements. Analogously, $PF(U, Q_G)$ and $PT(U, Q_G)$ depend only on the structure of $Fire(Q_G)$, as well as the probabilities of stochastic transitions and weights of deterministic transitions from its elements. Further, $PF(\Upsilon, [G]_{\approx})$ and $PT(\Upsilon, [G]_{\approx})$ are

respectively defined in the same way (using the same formulas and cases) as $PF(U, Q_G)$ and $PT(U, Q_G)$, for each pair of the *corresponding* (multi)set of activities Υ and transition set U. Obviously, the isomorphism of $Exec([G]_\approx)$ and $Fire(Q_G)$ guarantees coincidence of their structure as well as the mentioned probabilities and weights. Hence, if U *corresponds* to Υ then $PF(\Upsilon, [G]_\approx) = PF(U, Q_G)$ and $PT(\Upsilon, [G]_\approx) = PT(U, Q_G)$.

We also have $\mathcal{L}(\Upsilon) = \mathcal{L}(U)$, where $\mathcal{L}(U) = \sum_{\{t \in U | \Lambda_G(t) = \varrho_{(\alpha,\kappa)}\}} \alpha$ is the *multiaction part* of a set of transitions $U \subseteq T_N$. Thus, each transition $[G]_\approx \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$ in $TS(\overline{E})$ has a corresponding one $Q_G \xrightarrow{U}_{\mathcal{P}} \widetilde{Q}$ in $RG(\overline{N})$ with $\mathcal{L}(\Upsilon) = \mathcal{L}(U)$ and vice versa. Observe that the structure of the plain and operator dtsd-boxes in dtsdPBC is similar to that of the plain and operator boxes in PBC. Hence, like in PBC [41, 40], we can prove that $\tilde{s} = [\widetilde{G}]_\approx$ and $\widetilde{Q} = Q_{\widetilde{G}} = (M_{\widetilde{G}}, V_{\widetilde{G}})$ with $(N, Q_{\widetilde{G}}) = Box_{dtsd}(\widetilde{G})$ for the dynamic expression $\widetilde{G}$ such that $G \xrightarrow{\Upsilon} \widetilde{G}$. The only fine point here is to check that $I_{\widetilde{G}}$ and $V_{\widetilde{G}}$ are respectively obtained from I_G and V_G just by exploring $Ena([\widetilde{G}]_\approx)$ and $Ena(M_{\widetilde{G}})$ (which are similar up to restricted activities, with a care of relabeling and synchronization, as based on the corresponding overlinings and markings), as well as by checking whether $vanish([G]_\approx)$ and $vanish(Q_G)$ (which are correlated, as defined via the isomorphic $Exec([G]_\approx)$ and $Fire(Q_G)$). Therefore, by construction of $\mathcal{R}$, we get $([\widetilde{G}]_\approx, Q_{\widetilde{G}}) \in \mathcal{R}$.

The step stochastic bisimulation transfer property states that if $([G]_\approx, Q_G) \in \mathcal{R}$ then $(SJ([G]_\approx) = 0 \;\Leftrightarrow\; SJ(Q_G) = 0)$ and $\forall \mathcal{H} \in (DR(G) \cup RS(N_G))/_{\mathcal{R}}\ \forall A \in \mathbb{N}^{\mathcal{L}}_{fin}$ it holds $[G]_\approx \xrightarrow{A}_{\mathcal{P}} \mathcal{H} \;\Leftrightarrow\; Q_G \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$.

The fact $SJ([G]_\approx) = 0 \;\Leftrightarrow\; SJ(Q_G) = 0$ follows from isomorphism of $Exec([G]_\approx)$ and $Fire(Q_G)$, since $SJ([G]_\approx) = 0 \;\Leftrightarrow\; vanish([G]_\approx)$ and $SJ(Q_G) = 0 \;\Leftrightarrow\; vanish(Q_G)$.

Let $\mathcal{H} \in (DR(G) \cup RS(N_G))/_{\mathcal{R}}$. We have $PM_A([G]_\approx, \mathcal{H}) = \sum_{\{\Upsilon | \exists [\widetilde{G}]_\approx \in \mathcal{H}\ [G]_\approx \xrightarrow{\Upsilon}_{\mathcal{P}} [\widetilde{G}]_\approx,\ \mathcal{L}(\Upsilon) = A\}} PT(\Upsilon, [G]_\approx) = \sum_{i=1}^{n} PT(\Upsilon_i, [G]_\approx)$. Then we take the *corresponding* sets of transitions $U_1, \ldots, U_n \subseteq T_N$, such that $A = \mathcal{L}(\Upsilon_i) = \mathcal{L}(U_i)$ and $PT(\Upsilon_i, [G]_\approx) = PT(U_i, Q_G)$ $(1 \leq i \leq n)$, hence, $PM_A([G]_\approx, \mathcal{H}) = \sum_{i=1}^{n} PT(\Upsilon_i, [G]_\approx) = \sum_{i=1}^{n} PT(U_i, Q_G) \leq \sum_{\{U | \exists Q_{\widetilde{G}} \in \mathcal{H}\ Q_G \xrightarrow{U}_{\mathcal{P}} Q_{\widetilde{G}},\ \mathcal{L}(U) = A\}} PT(U, Q_G) = PM_A(Q_G, \mathcal{H})$. By symmetry of the correspondence between the (multi)sets of activities and sets of transitions, we get $PM_A([G]_\approx, \mathcal{H}) \geq PM_A(Q_G, \mathcal{H})$, hence, $PM_A([G]_\approx, \mathcal{H}) = PM_A(Q_G, \mathcal{H})$. Thus, we conclude that $[G]_\approx \xrightarrow{A}_{\mathcal{P}} \mathcal{H} \;\Leftrightarrow\; Q_G \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$. □

A.4 Proof of Proposition 7.4

Let $\mathcal{K}, \widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}_{ss}(G)}$ and $s \in \mathcal{K}$. The EDTMC for the quotient of $EDTMC(G)$ is denoted by $EDTMC'(G)$ and has the probabilities $PM'(\mathcal{K}, \widetilde{\mathcal{K}})$ to change from $\mathcal{K}$ to $\widetilde{\mathcal{K}}$.

- Let $PM(s,s) + PM(s, \mathcal{K}\backslash\{s\}) = PM(s, \mathcal{K}) < 1$ and $PM(s,s), PM(s, \mathcal{K}\backslash\{s\}) > 0$, i.e. $s, \mathcal{K}$ are non-absorbing and there exist self-loops associated with s in $DTMC(G)$ and with $\mathcal{K}$ in the quotient of $EDTMC(G)$.

  In $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, we have $PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) PM(\mathcal{K}, \widetilde{\mathcal{K}}) = \frac{PM(\mathcal{K}, \widetilde{\mathcal{K}})}{1 - PM(\mathcal{K}, \mathcal{K})} = \frac{PM(s, \widetilde{\mathcal{K}})}{1 - PM(s, \mathcal{K})} = \frac{PM(s, \widetilde{\mathcal{K}})}{1 - PM(s,s) - PM(s, \mathcal{K}\backslash\{s\})} = \frac{\frac{PM(s, \widetilde{\mathcal{K}})}{1 - PM(s,s)}}{1 - \frac{PM(s, \mathcal{K}\backslash\{s\})}{1 - PM(s,s)}} = \frac{SL(s) PM(s, \widetilde{\mathcal{K}})}{1 - SL(s) PM(s, \mathcal{K}\backslash\{s\})}$. Then $SL_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = \frac{SL(s)}{1 - SL(s) PM(s, \mathcal{K}\backslash\{s\})} = SL(s) SL'(s, \mathcal{K})$, where $SL'(s, \mathcal{K}) = \frac{1}{1 - SL(s) PM(s, \mathcal{K}\backslash\{s\})}$ is the self-loops abstraction factor in the equivalence class $\mathcal{K}$ with respect to the state $s \in \mathcal{K}$ for the quotient of $EDTMC(G)$.

  In $EDTMC'(G)$, we have $PM'(\mathcal{K}, \widetilde{\mathcal{K}}) = \frac{\sum_{\tilde{s} \in \widetilde{\mathcal{K}}} PM^*(s, \tilde{s})}{1 - \sum_{s' \in \mathcal{K}\backslash\{s\}} PM^*(s, s')} = \frac{\sum_{\tilde{s} \in \widetilde{\mathcal{K}}} SL(s) PM(s, \tilde{s})}{1 - \sum_{s' \in \mathcal{K}\backslash\{s\}} SL(s) PM(s, s')} = \frac{SL(s) \sum_{\tilde{s} \in \widetilde{\mathcal{K}}} PM(s, \tilde{s})}{1 - SL(s) \sum_{s' \in \mathcal{K}\backslash\{s\}} PM(s, s')} = \frac{SL(s) PM(s, \widetilde{\mathcal{K}})}{1 - SL(s) PM(s, \mathcal{K}\backslash\{s\})} = PM^*(\mathcal{K}, \widetilde{\mathcal{K}})$.

  The other three cases (no self-loops associated with s in $DTMC(G)$, with $\mathcal{K}$ in the quotient of $EDTMC(G)$, or with both) are treated analogously, by replacing $PM(s,s)$ or/and $PM(s, \mathcal{K} \setminus \{s\})$ with zeros.

- Let $PM(s,s) + PM(s, \mathcal{K} \setminus \{s\}) = PM(s, \mathcal{K}) = 1$ and $PM(s,s), PM(s, \mathcal{K} \setminus \{s\}) > 0$, i.e. $\mathcal{K}$ is absorbing in $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and there exist self-loops associated with s in $DTMC(G)$ and with $\mathcal{K}$ in the quotient of $EDTMC(G)$.

  In $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, we have $PM^*(\mathcal{K}, \mathcal{K}) = 1$ by definition of the EDTMC, since $PM(\mathcal{K}, \mathcal{K}) = PM(s, \mathcal{K}) = 1$.

  In the quotient of $EDTMC(G)$, the probability of a self-loop associated with $\mathcal{K}$ is $\sum_{s' \in \mathcal{K}\backslash\{s\}} PM^*(s, s') = \sum_{s' \in \mathcal{K}\backslash\{s\}} SL(s) PM(s, s') = SL(s) \sum_{s' \in \mathcal{K}\backslash\{s\}} PM(s, s') = SL(s) PM(s, \mathcal{K}\backslash\{s\}) = SL(s)(1 - PM(s,s)) = \frac{1 - PM(s,s)}{1 - PM(s,s)} = 1$.

  In $EDTMC'(G)$, we have $PM'(\mathcal{K}, \widetilde{\mathcal{K}}) = 1 = PM^*(\mathcal{K}, \mathcal{K})$ by definition of the EDTMC, since in the quotient of $EDTMC(G)$, the probability of a self-loop associated with $\mathcal{K}$ is 1.

  The other two cases (no self-loops associated with s in $DTMC(G)$ or with $\mathcal{K}$ in the quotient of $EDTMC(G)$) are treated analogously, by replacing $PM(s,s)$ with zero or taking $\mathcal{K} = \{s\}$ when $PM(s, \mathcal{K} \setminus \{s\}) = 0$.

Thus, $(\mathbf{P}^*)^*_{\underline{\leftrightarrow}_{ss}} = \mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$ and $EDTMC'(G) = EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$. □

A.5 Proof of Proposition 7.5

Let $\mathbf{P}_r$ be the reordered (by moving vanishing states to the first positions) TPM for $DTMC(G)$. Like in Section 5, we reorder the states from $DR(G)$ so that the first rows and columns of $\mathbf{P}_r$ will correspond to the states from $DR_V(G)$ and the last ones will correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. Then the reordered TPM for $DTMC(G)$ can be decomposed as

$$\mathbf{P}_r = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}.$$

The elements of the $(n-m) \times (n-m)$ submatrix $\mathbf{C}$ are the probabilities to move from vanishing to vanishing states, and those of the $(n-m) \times m$ submatrix $\mathbf{D}$ are the probabilities to move from vanishing to tangible states. The elements of the $m \times (n-m)$ submatrix $\mathbf{E}$ are the probabilities to move from tangible to vanishing states, and those of the $m \times m$ submatrix $\mathbf{F}$ are the probabilities to move from tangible to tangible states.

The TPM $\mathbf{P}^\diamond$ for $RDTMC(G)$ is the $m \times m$ matrix, calculated as

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD},$$

where the elements of the matrix $\mathbf{G} = \sum_{k=0}^{\infty} \mathbf{C}^k$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states.

By the note after Proposition 6.1, $\mathcal{R}_{ss}(G) \subseteq (DR_T(G))^2 \uplus (DR_V(G))^2$. Hence, $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$, all states from $\mathcal{K}$ are tangible, when $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$, or all of them are vanishing, when $\mathcal{K} \in DR_V(G)/_{\mathcal{R}}$.

Let $\mathbf{V}_r$ be the reordered (by moving vanishing states and their equivalence classes to the first positions) collector matrix for $\mathcal{R}_{ss}(\overline{F})$ and $\mathbf{W}_r$ be the (accordingly) reordered distributor matrix for $\mathbf{V}_r$. We reorder the states from $DR(G)$ and the equivalence classes from $DR(G)/_{\mathcal{R}_{ss}(G)}$ as follows. The first rows of $\mathbf{V}_r$ will correspond to the states from $DR_V(G)$ and the first columns of $\mathbf{V}_r$ will correspond to the equivalence classes from $DR_V(G)/_{\mathcal{R}_{ss}(G)}$, whereas the last rows of $\mathbf{V}_r$ will correspond to the states from $DR_T(G)$ and the last columns of $\mathbf{V}_r$ will correspond to the equivalence classes from $DR_T(G)/_{\mathcal{R}_{ss}(G)}$. The first rows of $\mathbf{W}_r$ will correspond to the equivalence classes from $DR_V(G)/_{\mathcal{R}_{ss}(G)}$ and the first columns of $\mathbf{W}_r$ will correspond to the states from $DR_V(G)$, whereas the last rows of $\mathbf{W}_r$ will correspond to the equivalence classes from $DR_T(G)/_{\mathcal{R}_{ss}(G)}$ and the last columns of $\mathbf{W}_r$ will correspond to the states from $DR_T(G)$.

Let $|DR(G)/_{\mathcal{R}_{ss}(G)}| = l$ and $|DR_T(G)/_{\mathcal{R}_{ss}(G)}| = k$. Note that tangible (vanishing) states can only belong to the equivalence classes of tangible (vanishing) states. Then the reordered collector and distributor matrices can be decomposed as

$$\mathbf{V}_r = \begin{pmatrix} \mathbf{V}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_F \end{pmatrix}, \ \mathbf{W}_r = \begin{pmatrix} \mathbf{W}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_F \end{pmatrix},$$

where $\mathbf{0}$ are the matrices consisting only of zeros, all those matrices of the appropriate sizes. The elements of the $(n-m) \times (l-k)$ submatrix $\mathbf{V}_C$ are the probabilities to move from vanishing states to the equivalence classes of vanishing states, and those of the $m \times k$ submatrix $\mathbf{V}_F$ are the probabilities to move from tangible states to the equivalence classes of tangible states. The elements of the $(l-k) \times (n-m)$ submatrix $\mathbf{W}_C$ are the probabilities to move from the equivalence classes of vanishing states to vanishing states, and those of the $k \times m$ submatrix $\mathbf{W}_F$ are the probabilities to move from the equivalence classes of tangible states to tangible states.

We have

$$\mathbf{W}_r\mathbf{V}_r = \begin{pmatrix} \mathbf{W}_C\mathbf{V}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_F\mathbf{V}_F \end{pmatrix} = \mathbf{I},$$

hence, $\mathbf{W}_C\mathbf{V}_C = \mathbf{I}$ and $\mathbf{W}_F\mathbf{V}_F = \mathbf{I}$.

Since tangible and vanishing states always belong to the equivalence classes of the same kind, the quotienting (by $\underline{\leftrightarrow}_{ss}$) and reordering (by moving vanishing states and their equivalence classes to the first positions) are permutable. The quotiented reordered TPM may only differ from the reordered quotiented TPM up to the order of the equivalence classes of tangible states and the order of the equivalence classes of vanishing states. To avoid such a difference, we rearrange the equivalence classes of the same kind in increasing order of the smallest indices of the states from them while keeping the equivalence classes of vanishing states at the first positions.

Then $\mathbf{P}_r\mathbf{V}_r = \mathbf{V}_r\mathbf{P}_{r\underline{\leftrightarrow}_{ss}}$ and $\mathbf{P}_{r\underline{\leftrightarrow}_{ss}} = \mathbf{W}_r\mathbf{P}_r\mathbf{V}_r$. We have

$$\mathbf{P}_r\mathbf{V}_r = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}\begin{pmatrix} \mathbf{V}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_F \end{pmatrix} = \begin{pmatrix} \mathbf{C}\mathbf{V}_C & \mathbf{D}\mathbf{V}_F \\ \mathbf{E}\mathbf{V}_C & \mathbf{F}\mathbf{V}_F \end{pmatrix},$$
$$\mathbf{V}_r\mathbf{P}_{r\underline{\leftrightarrow}_{ss}} = \begin{pmatrix} \mathbf{V}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_F \end{pmatrix}\begin{pmatrix} \mathbf{C}_{\underline{\leftrightarrow}_{ss}} & \mathbf{D}_{\underline{\leftrightarrow}_{ss}} \\ \mathbf{E}_{\underline{\leftrightarrow}_{ss}} & \mathbf{F}_{\underline{\leftrightarrow}_{ss}} \end{pmatrix} = \begin{pmatrix} \mathbf{V}_C\mathbf{C}_{\underline{\leftrightarrow}_{ss}} & \mathbf{V}_C\mathbf{D}_{\underline{\leftrightarrow}_{ss}} \\ \mathbf{V}_F\mathbf{E}_{\underline{\leftrightarrow}_{ss}} & \mathbf{V}_F\mathbf{F}_{\underline{\leftrightarrow}_{ss}} \end{pmatrix}.$$

Hence, $\mathbf{C}\mathbf{V}_C = \mathbf{V}_C\mathbf{C}_{\underline{\leftrightarrow}_{ss}}$, $\mathbf{D}\mathbf{V}_F = \mathbf{V}_C\mathbf{D}_{\underline{\leftrightarrow}_{ss}}$, $\mathbf{E}\mathbf{V}_C = \mathbf{V}_F\mathbf{E}_{\underline{\leftrightarrow}_{ss}}$, $\mathbf{F}\mathbf{V}_F = \mathbf{V}_F\mathbf{F}_{\underline{\leftrightarrow}_{ss}}$.

Let us demonstrate that $\mathbf{G}\mathbf{V}_C = \mathbf{V}_C\mathbf{G}_{\underline{\leftrightarrow}_{ss}}$. Since $\mathbf{G} = \sum_{k=0}^{\infty}\mathbf{C}^k$, it is sufficient to prove $\left(\sum_{k=0}^{l}\mathbf{C}^k\right)\mathbf{V}_C = \mathbf{V}_C\sum_{k=0}^{l}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k$ by induction on $l \in \mathbb{N}$ and then take a limit $l \to \infty$.

- $l = 0$

  We have $\left(\sum_{k=0}^{0}\mathbf{C}^k\right)\mathbf{V}_C = \mathbf{C}^0\mathbf{V}_C = \mathbf{I}\mathbf{V}_C = \mathbf{V}_C = \mathbf{V}_C\mathbf{I} = \mathbf{V}_C\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^0 = \mathbf{V}_C\sum_{k=0}^{0}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k$.

- $l \to l+1$

  Suppose that $\left(\sum_{k=0}^{l}\mathbf{C}^k\right)\mathbf{V}_C = \mathbf{V}_C\sum_{k=0}^{l}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k$. Then $\left(\sum_{k=0}^{l+1}\mathbf{C}^k\right)\mathbf{V}_C = \left(\mathbf{I} + \mathbf{C}\sum_{k=0}^{l}\mathbf{C}^k\right)\mathbf{V}_C = \mathbf{V}_C + \mathbf{C}\mathbf{V}_C\sum_{k=0}^{l}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k = \mathbf{V}_C + \mathbf{V}_C\mathbf{C}_{\underline{\leftrightarrow}_{ss}}\sum_{k=0}^{l}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k = \mathbf{V}_C\left(\mathbf{I} + \mathbf{C}_{\underline{\leftrightarrow}_{ss}}\sum_{k=0}^{l}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k\right) = \mathbf{V}_C\sum_{k=0}^{l+1}\mathbf{C}_{\underline{\leftrightarrow}_{ss}}^k$.

Next, $\mathbf{P}^{\diamond}\mathbf{V}_F = (\mathbf{F}+\mathbf{E}\mathbf{G}\mathbf{D})\mathbf{V}_F = \mathbf{F}\mathbf{V}_F + \mathbf{E}\mathbf{G}\mathbf{D}\mathbf{V}_F = \mathbf{V}_F\mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}\mathbf{G}\mathbf{V}_C\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \mathbf{V}_F\mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}\mathbf{V}_C\mathbf{G}_{\underline{\leftrightarrow}_{ss}}\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \mathbf{V}_F\mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{V}_F\mathbf{E}_{\underline{\leftrightarrow}_{ss}}\mathbf{G}_{\underline{\leftrightarrow}_{ss}}\mathbf{D}_{\underline{\leftrightarrow}_{ss}} = \mathbf{V}_F(\mathbf{F}_{\underline{\leftrightarrow}_{ss}} + \mathbf{E}_{\underline{\leftrightarrow}_{ss}}\mathbf{G}_{\underline{\leftrightarrow}_{ss}}\mathbf{D}_{\underline{\leftrightarrow}_{ss}}) = \mathbf{V}_F\mathbf{P}_{\underline{\leftrightarrow}_{ss}}^{\diamond}$. After left-multiplying by $\mathbf{W}_F$ the resulting equality $\mathbf{P}^{\diamond}\mathbf{V}_F = \mathbf{V}_F\mathbf{P}_{\underline{\leftrightarrow}_{ss}}^{\diamond}$, we finally get

$$(\mathbf{P}^{\diamond})_{\underline{\leftrightarrow}_{ss}} = \mathbf{W}_F\mathbf{P}^{\diamond}\mathbf{V}_F = \mathbf{P}_{\underline{\leftrightarrow}_{ss}}^{\diamond}.$$

□

A.6 Proof of Proposition 8.1

By Proposition 6.1, $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G) \cup DR_V(G'))/_{\mathcal{R}})$. Hence, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.

By definition of the steady-state PMFs for SMCs, $\forall s \in DR_V(G)\ \varphi(s) = 0$ and $\forall s' \in DR_V(G')\ \varphi'(s') = 0$. Thus, $\forall \mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}\ \sum_{s\in\mathcal{H}\cap DR(G)}\varphi(s) = \sum_{s\in\mathcal{H}\cap DR_V(G)}\varphi(s) = 0 = \sum_{s'\in\mathcal{H}\cap DR_V(G')}\varphi'(s') = \sum_{s'\in\mathcal{H}\cap DR(G')}\varphi'(s')$.

By Proposition 5.1, $\forall s \in DR_T(G)\ \varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}$ and $\forall s' \in DR_T(G')\ \varphi'(s') = \frac{\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')}$, where ψ and ψ' are the steady-state PMFs for $DTMC(G)$ and $DTMC(G')$, respectively. Thus, $\forall \mathcal{H}, \widetilde{\mathcal{H}} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}\ \sum_{s\in\mathcal{H}\cap DR(G)}\varphi(s) = \sum_{s\in\mathcal{H}\cap DR_T(G)}\varphi(s) = \sum_{s\in\mathcal{H}\cap DR_T(G)}\left(\frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}\right) = \frac{\sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})} = \frac{\sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s)}{\sum_{\widetilde{\mathcal{H}}}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR_T(G)}\psi(\tilde{s})}$ and $\sum_{s'\in\mathcal{H}\cap DR(G')}\varphi'(s') = \sum_{s'\in\mathcal{H}\cap DR_T(G')}\varphi'(s') = \sum_{s'\in\mathcal{H}\cap DR_T(G')}\left(\frac{\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')}\right) = \frac{\sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')} = \frac{\sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')}{\sum_{\widetilde{\mathcal{H}}}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR_T(G')}\psi'(\tilde{s}')}$.

It remains to prove that $\forall \mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}\ \sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s) = \sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')$. Since $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G) \cup DR_V(G'))/_{\mathcal{R}})$, the previous equality is a consequence of the following one: $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \sum_{s\in\mathcal{H}\cap DR(G)}\psi(s) = \sum_{s'\in\mathcal{H}\cap DR(G')}\psi'(s')$.

Standard proof continuation.

It is sufficient to prove the previous statement for transient PMFs only, since $\psi = \lim_{k\to\infty}\psi[k]$ and $\psi' = \lim_{k\to\infty}\psi'[k]$. We proceed by induction on k.

- $k = 0$

  Note that the only non-zero values of the initial PMFs of $DTMC(G)$ and $DTMC(G')$ are $\psi[0]([G]_{\approx})$ and $\psi[0]([G']_{\approx})$. Let $\mathcal{H}_0$ be the equivalence class containing $[G]_{\approx}$ and $[G']_{\approx}$. Then $\sum_{s\in\mathcal{H}_0\cap DR(G)}\psi[0](s) = \psi[0]([G]_{\approx}) = 1 = \psi'[0]([G']_{\approx}) = \sum_{s'\in\mathcal{H}_0\cap DR(G')}\psi'[0](s')$.

  As for other equivalence classes, $\forall \mathcal{H} \in ((DR(G) \cup DR(G'))/_{\mathcal{R}}) \setminus \mathcal{H}_0$ we have $\sum_{s\in\mathcal{H}\cap DR(G)}\psi[0](s) = 0 = \sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[0](s')$.

- $k \to k+1$

  Let $\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{H}$. We have $\forall \widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}}$ $s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}}$. Therefore, $PM(s_1, \widetilde{\mathcal{H}}) = \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}}\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1\}} PT(\Upsilon, s_1) =$

$\sum_{A\in I\!N^{\mathcal{L}}_{fin}} \sum_{\{\Upsilon|\exists \tilde{s}_1\in\widetilde{\mathcal{H}}\ s_1\xrightarrow{\Upsilon}\tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon,s_1) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}} PM_A(s_1,\widetilde{\mathcal{H}}) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}} PM_A(s_2,\widetilde{\mathcal{H}}) =$
$\sum_{A\in I\!N^{\mathcal{L}}_{fin}} \sum_{\{\Upsilon|\exists \tilde{s}_2\in\widetilde{\mathcal{H}}\ s_2\xrightarrow{\Upsilon}\tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon,s_2) = \sum_{\{\Upsilon|\exists \tilde{s}_2\in\widetilde{\mathcal{H}}\ s_2\xrightarrow{\Upsilon}\tilde{s}_2\}} PT(\Upsilon,s_2) = PM(s_2,\widetilde{\mathcal{H}})$. Since we have the previous equality for all $s_1,s_2\in\mathcal{H}$, we can denote $PM(\mathcal{H},\widetilde{\mathcal{H}}) = PM(s_1,\widetilde{\mathcal{H}}) = PM(s_2,\widetilde{\mathcal{H}})$. Note that transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s\in DR(G)\ PM(s,\widetilde{\mathcal{H}}) = PM(s,\widetilde{\mathcal{H}}\cap DR(G))$. The same is true for $DR(G')$.

By induction hypothesis, $\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s) = \sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')$. Further,
$\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k+1](\tilde{s}) = \sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{s\in DR(G)}\psi[k](s)PM(s,\tilde{s}) =$
$\sum_{s\in DR(G)}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k](s)PM(s,\tilde{s}) = \sum_{s\in DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s}) =$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s}) =$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{\{\Upsilon|s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s) =$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\{\Upsilon|\exists\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)\ s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s) =$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(s,\widetilde{\mathcal{H}}) = \sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(\mathcal{H},\widetilde{\mathcal{H}}) =$
$\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s) = \sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s') =$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')PM(\mathcal{H},\widetilde{\mathcal{H}}) = \sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}'\cap DR(G')}\psi'[k](s')PM(s',\widetilde{\mathcal{H}}) =$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\{\Upsilon|\exists\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s') =$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{\{\Upsilon|\exists\tilde{s}'\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s') =$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}') =$
$\sum_{s'\in DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}') = \sum_{s'\in DR(G')}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k](s')PM(s',\tilde{s}') =$
$\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{s'\in DR(G')}\psi'[k](s')PM(s',\tilde{s}') = \sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k+1](\tilde{s}')$. □

Alternative proof continuation.
Thus, we should now prove that $\forall\mathcal{H}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}\ \sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}\psi_i = \sum_{\{j|s'_j\in\mathcal{H}\cap DR(G')\}}\psi'_j$.

The steady-state PMF $\psi=(\psi_1,\ldots,\psi_n)$ for $DTMC(G)$ is a solution of the linear equation system

$$\begin{cases}\psi\mathbf{P}=\psi\\ \psi\mathbf{1}^T=1\end{cases}.$$

Then, for all i $(1\le i\le n)$, we have

$$\begin{cases}\sum_{j=1}^n\mathcal{P}_{ji}\psi_j=\psi_i\\ \sum_{j=1}^n\psi_j=1\end{cases}.$$

By definition of $\mathcal{P}_{ij}$ $(1\le i,j\le n)$ we have

$$\begin{cases}\sum_{j=1}^n PM(s_j,s_i)\psi_j=\psi_i\\ \sum_{j=1}^n\psi_j=1\end{cases}.$$

Let $\mathcal{H}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}$ and $s_1,s_2\in\mathcal{H}$. We have $\forall\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}\ \forall A\in I\!N^{\mathcal{L}}_{fin}$
$s_1\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}\ \Leftrightarrow\ s_2\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. Therefore, $PM(s_1,\widetilde{\mathcal{H}}) = \sum_{\{\Upsilon|\exists\tilde{s}_1\in\widetilde{\mathcal{H}}\ s_1\xrightarrow{\Upsilon}\tilde{s}_1\}}PT(\Upsilon,s_1) =$
$\sum_{A\in I\!N^{\mathcal{L}}_{fin}}\sum_{\{\Upsilon|\exists\tilde{s}_1\in\widetilde{\mathcal{H}}\ s_1\xrightarrow{\Upsilon}\tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}}PT(\Upsilon,s_1) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}}PM_A(s_1,\widetilde{\mathcal{H}}) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}}PM_A(s_2,\widetilde{\mathcal{H}}) =$
$\sum_{A\in I\!N^{\mathcal{L}}_{fin}}\sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}}\ s_2\xrightarrow{\Upsilon}\tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}}PT(\Upsilon,s_2) = \sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}}\ s_2\xrightarrow{\Upsilon}\tilde{s}_2\}}PT(\Upsilon,s_2) = PM(s_2,\widetilde{\mathcal{H}})$. Since we have the previous equality for all $s_1,s_2\in\mathcal{H}$, we can denote $PM(\mathcal{H},\widetilde{\mathcal{H}}) = PM(s_1,\widetilde{\mathcal{H}}) = PM(s_2,\widetilde{\mathcal{H}})$. Note that transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s\in DR(G)\ PM(s,\widetilde{\mathcal{H}}) = PM(s,\widetilde{\mathcal{H}}\cap DR(G))$. The same is true for $DR(G')$.

Let $\mathcal{H}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}$. We sum the left and right parts of the first equation from the system above for all i such that $s_i\in\mathcal{H}\cap DR(G)$. The resulting equation is

$$\sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}\sum_{j=1}^n PM(s_j,s_i)\psi_j = \sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}\psi_i.$$

Let us denote the aggregate steady-state PMF for $DTMC(G)$ by $\psi_{\mathcal{H}\cap DR(G)} = \sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}\psi_i$. Then, for the left part of the equation above, we get
$\sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}\sum_{j=1}^n PM(s_j,s_i)\psi_j = \sum_{j=1}^n\psi_j\sum_{\{i|s_i\in\mathcal{H}\cap DR(G)\}}PM(s_j,s_i) = \sum_{j=1}^n PM(s_j,\mathcal{H})\psi_j =$
$\sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}}\sum_{\{j|s_j\in\widetilde{\mathcal{H}}\cap DR(G)\}}PM(s_j,\mathcal{H})\psi_j =$

$\sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} \sum_{\{j|s_j\in\widetilde{\mathcal{H}}\cap DR(G)\}} PM(\widetilde{\mathcal{H}},\mathcal{H})\psi_j =$
$\sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} PM(\widetilde{\mathcal{H}},\mathcal{H}) \sum_{\{j|s_j\in\widetilde{\mathcal{H}}\cap DR(G)\}} \psi_j = \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} PM(\widetilde{\mathcal{H}},\mathcal{H})\psi_{\widetilde{\mathcal{H}}\cap DR(G)}$.

For the left part of the second equation from the system above, we have
$\sum_{j=1}^{n} \psi_j = \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} \sum_{\{j|s_j\in\widetilde{\mathcal{H}}\cap DR(G)\}} \psi_j = \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} \psi_{\widetilde{\mathcal{H}}\cap DR(G)}$.

Thus, the aggregate linear equation system for $DTMC(G)$ is

$$\begin{cases} \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} PM(\widetilde{\mathcal{H}},\mathcal{H})\psi_{\widetilde{\mathcal{H}}\cap DR(G)} = \psi_{\mathcal{H}\cap DR(G)} \\ \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} \psi_{\widetilde{\mathcal{H}}\cap DR(G)} = 1 \end{cases}.$$

Let us denote the aggregate steady-state PMFs for $DTMC(G')$ by $\psi'_{\mathcal{H}\cap DR(G')} = \sum_{\{j|s'_j\in\mathcal{H}\cap DR(G')\}} \psi'_j$. Then, in a similar way, the aggregate linear equation system for $DTMC(G')$ is

$$\begin{cases} \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} PM(\widetilde{\mathcal{H}},\mathcal{H})\psi'_{\widetilde{\mathcal{H}}\cap DR(G')} = \psi'_{\mathcal{H}\cap DR(G')} \\ \sum_{\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}} \psi'_{\widetilde{\mathcal{H}}\cap DR(G')} = 1 \end{cases}.$$

Let $(DR(G)\cup DR(G'))/_{\mathcal{R}} = \{\mathcal{H}_1,\ldots,\mathcal{H}_l\}$. Then the aggregate steady-state PMFs $\psi_{\mathcal{H}_k\cap DR(G)}$ and $\psi'_{\mathcal{H}_k\cap DR(G')}$ $(1\leq k\leq l)$ satisfy the same aggregate system of $l+1$ linear equations with l independent equations and l unknowns. The aggregate linear equation system has a unique solution, when a single aggregate steady-state PMF exists. This is the case here, since in Section 5 we have demonstrated that $DTMC(G)$ has a single steady state iff $SMC(G)$ has, and aggregation preserves this property [73]. Hence, $\psi_{\mathcal{H}_k\cap DR(G)} = \psi'_{\mathcal{H}_k\cap DR(G')}$ $(1\leq k\leq l)$. □

A.7 Proof of Theorem 8.1

Let $\mathcal{H}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}$ and $s,\bar{s}\in\mathcal{H}$. We have $\forall\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}\ \forall A\in\mathbb{N}^{\mathcal{L}}_{fin}\ s\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}} \Leftrightarrow \bar{s}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. The previous equality is valid for all $s,\bar{s}\in\mathcal{H}$, hence, we can rewrite it as $\mathcal{H}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$ and denote $PM_A(\mathcal{H},\widetilde{\mathcal{H}}) = PM_A(s,\widetilde{\mathcal{H}}) = PM_A(\bar{s},\widetilde{\mathcal{H}})$. Note that transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s\in DR(G)\ PM_A(s,\widetilde{\mathcal{H}}) = PM_A(s,\widetilde{\mathcal{H}}\cap DR(G))$. The same is true for $DR(G')$.

Let $\Sigma = A_1\cdots A_n$ be a derived step trace of G and G'. Then $\exists\mathcal{H}_0,\ldots,\mathcal{H}_n\in(DR(G)\cup DR(G'))/_{\mathcal{R}}\ \mathcal{H}_0\xrightarrow{A_1}_{\mathcal{P}_1}\mathcal{H}_1\xrightarrow{A_2}_{\mathcal{P}_2}\cdots\xrightarrow{A_n}_{\mathcal{P}_n}\mathcal{H}_n$. We now intend to prove that the sum of probabilities of all the paths starting in every $s_0\in\mathcal{H}_0$ and going through the states from $\mathcal{H}_1,\ldots,\mathcal{H}_n$ is equal to the product of $\mathcal{P}_1,\ldots,\mathcal{P}_n$:

$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1}) = \prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i).$$

We prove this equality by induction on the derived step trace length n.

- $n=1$
  $\sum_{\{\Upsilon_1|s_0\xrightarrow{\Upsilon_1}s_1,\ \mathcal{L}(\Upsilon_1)=A_1,\ s_1\in\mathcal{H}_1\}} PT(\Upsilon_1,s_0) = PM_{A_1}(s_0,\mathcal{H}_1) = PM_{A_1}(\mathcal{H}_0,\mathcal{H}_1)$.
- $n\to n+1$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n,\Upsilon_{n+1}|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n+1)\}} \prod_{i=1}^{n+1} PT(\Upsilon_i,s_{i-1}) =$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \sum_{\{\Upsilon_{n+1}|s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}}$
  $\prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1})PT(\Upsilon_{n+1},s_n) =$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}}$
  $\left[\prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1}) \sum_{\{\Upsilon_{n+1}|s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}} PT(\Upsilon_{n+1},s_n)\right] =$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1})PM_{A_{n+1}}(s_n,\mathcal{H}_{n+1}) =$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1})PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1}) =$
  $PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1}) \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1}) =$
  $PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1}) \prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i) = \prod_{i=1}^{n+1} PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i)$.

Let $s_0,\bar{s}_0\in\mathcal{H}_0$. We have
$PT(A_1\cdots A_n,s_0) = \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1}) =$
$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}} \prod_{i=1}^{n} PT(\Upsilon_i,s_{i-1}) =$

$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i) =$
$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \sum_{\{\overline{\Upsilon}_1,\ldots,\overline{\Upsilon}_n \mid \bar{s}_0 \xrightarrow{\overline{\Upsilon}_1} \cdots \xrightarrow{\overline{\Upsilon}_n} \bar{s}_n,\ \mathcal{L}(\overline{\Upsilon}_i)=A_i,\ \bar{s}_i \in \mathcal{H}_i\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\overline{\Upsilon}_i, \bar{s}_{i-1}) =$
$\sum_{\{\overline{\Upsilon}_1,\ldots,\overline{\Upsilon}_n \mid \bar{s}_0 \xrightarrow{\overline{\Upsilon}_1} \cdots \xrightarrow{\overline{\Upsilon}_n} \bar{s}_n,\ \mathcal{L}(\overline{\Upsilon}_i)=A_i,\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\overline{\Upsilon}_i, \bar{s}_{i-1}) = PT(A_1 \cdots A_n, \bar{s}_0)$.
Since we have the previous equality for all $s_0, \bar{s}_0 \in \mathcal{H}_0$, we can denote $PT(A_1 \cdots A_n, \mathcal{H}_0) = PT(A_1 \cdots A_n, s_0) = PT(A_1 \cdots A_n, \bar{s}_0)$.

By Proposition 8.1, $\sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) = \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s')$. We now can complete the proof:
$\sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) PT(\Sigma, s) = \sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) PT(\Sigma,\mathcal{H}) = PT(\Sigma,\mathcal{H}) \sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) =$
$PT(\Sigma,\mathcal{H}) \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s') = \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s') PT(\Sigma,\mathcal{H}) = \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s') PT(\Sigma, s')$. □

A.8 Proof of Proposition 8.2

Let us present two facts, which will be used in the proof.

1. By Proposition 6.1, $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G) \cup DR_V(G'))/_{\mathcal{R}})$. Hence, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.

2. Let $\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{H}$. We have $\forall \widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}\ \forall A \in I\!N^{\mathcal{L}}_{fin}$ $s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}} \ \Leftrightarrow\ s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}}$. Therefore, $PM(s_1, \widetilde{\mathcal{H}}) = \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}}\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1\}} PT(\Upsilon, s_1) =$ $\sum_{A\in I\!N^{\mathcal{L}}_{fin}} \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}}\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_1) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}} PM_A(s_1, \widetilde{\mathcal{H}}) = \sum_{A\in I\!N^{\mathcal{L}}_{fin}} PM_A(s_2, \widetilde{\mathcal{H}}) =$ $\sum_{A\in I\!N^{\mathcal{L}}_{fin}} \sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}}\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_2) = \sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}}\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2\}} PT(\Upsilon, s_2) = PM(s_2, \widetilde{\mathcal{H}})$. Since we have the previous equality for all $s_1, s_2 \in \mathcal{H}$, we can denote $PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(s_1, \widetilde{\mathcal{H}}) = PM(s_2, \widetilde{\mathcal{H}})$. Note that transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s \in DR(G)\ PM(s, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}} \cap DR(G))$. The same is true for $DR(G')$. Hence, for all $s \in \mathcal{H} \cap DR(G)$, we obtain $PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}} \cap DR(G)) = PM(\mathcal{H} \cap DR(G), \widetilde{\mathcal{H}} \cap DR(G))$. The same is true for $DR(G')$. Finally, $PM(\mathcal{H} \cap DR(G), \widetilde{\mathcal{H}} \cap DR(G)) = PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(\mathcal{H} \cap DR(G'), \widetilde{\mathcal{H}} \cap DR(G'))$.

Let us now prove the proposition statement for the sojourn time averages.

- Let $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.

  Then we have $\mathcal{H} \cap DR(G) = \mathcal{H} \cap DR_V(G) \in DR_V(G)/_{\mathcal{R}}$ and $\mathcal{H} \cap DR(G') = \mathcal{H} \cap DR_V(G') \in DR_V(G')/_{\mathcal{R}}$. By definition of the average sojourn time in an equivalence class of states, we get $SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR_V(G)) = 0 = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR_V(G')) = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G'))$.

- Let $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$.

  Then we have $\mathcal{H} \cap DR(G) = \mathcal{H} \cap DR_T(G) \in DR_T(G)/_{\mathcal{R}}$ and $\mathcal{H} \cap DR(G') = \mathcal{H} \cap DR_T(G') \in DR_T(G')/_{\mathcal{R}}$. By definition of the average sojourn time in an equivalence class of states, we get $SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR_T(G)) = \frac{1}{1-PM(\mathcal{H}\cap DR_T(G), \mathcal{H}\cap DR_T(G))} =$ $\frac{1}{1-PM(\mathcal{H}\cap DR(G), \mathcal{H}\cap DR(G))} = \frac{1}{1-PM(\mathcal{H},\mathcal{H})} = \frac{1}{1-PM(\mathcal{H}\cap DR(G'), \mathcal{H}\cap DR(G'))} = \frac{1}{1-PM(\mathcal{H}\cap DR_T(G'), \mathcal{H}\cap DR_T(G'))} =$ $SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR_T(G')) = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G'))$.

Thus, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ we have $SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G'))$. The proposition statement for the sojourn time variances is proved similarly to that for the averages. □