A Tutorial on Interactive Markov Chains

Florian Arnold[1], Daniel Gebler[2], Dennis Guck[1], and Hassan Hatefi[3]

[1] Formal Methods & Tools Group, Department of Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands

[2] Department of Computer Science, VU University Amsterdam,
De Boelelaan 1081a, NL-1081 HV Amsterdam, the Netherlands

[3] Department of Computer Science
Saarland University, 66123 Saarbrücken, Germany

Abstract. Interactive Markov chains (IMCs) constitute a powerful stochastic model that extends both continuous-time Markov chains and labelled transition systems. IMCs enable a wide range of modelling and analysis techniques and serve as a semantic model for many industrial and scientific formalisms, such as AADL, GSPNs and many more. Applications cover various engineering contexts ranging from industrial system-on-chip manufacturing to satellite designs. We present a survey of the state-of-the-art in modelling and analysis of IMCs.
We cover a set of techniques that can be utilised for compositional modelling, state space generation and reduction, and model checking. The significance of the presented material and corresponding tools is highlighted through multiple case studies.

1 Introduction

The increasing complexity of systems and software requires appropriate formal modelling and verification tools to gain insights into the system's possible behaviour and dependability. Imagine the construction of a satellite equipped with hardware components and software systems. Once sent into its orbit, the satellite has to work mostly autonomously. In case of any hardware or software component failure, the required maintenance work is time-consuming and cannot be executed immediately, leading to excessive costs and even complete system failures. To avoid such shortcomings, the system's components need to be highly dependable and any design mistakes must be identified as soon as possible. Rigorous modelling and analysis techniques can help significantly by accompanying the development process from the blue-print to the testing phase. They can answer quantitative questions like "what is the probability that the system fails within 3 years" by synthesising an abstract system model.

In the last years a plethora of formalisms [45, 25, 55, 47, 35, 23] and tools (PRISM [43], ETMCC [39], MRMC [42], YMER [58], VESTA [56] and MAMA

[27]) have been developed for this purpose. The advent of large-scale, distributed, dependable systems requires formal specification and verification methods that capture both qualitative and quantitative aspects of systems. Labelled transition systems (LTS) allow to capture qualitative aspects of software and hardware systems by defining the interaction between system components, but they lack quantitative predicates. On the other hand, continuous time Markov chains (CTMC) allow to model and reason over quantitative aspects of systems. However, CTMCs do not allow to model component dependencies and interaction with the environment.

A prominent formalism to remedy these drawbacks are interactive Markov Chains (IMCs) [35]. IMCs conservatively extend LTSs and CTMCs and thereby allow to accurately model system dependencies as well as quantitative aspects. IMCs strictly separate between nondeterministic choices, called interactive transitions, and exponentially distributed delays, called Markovian transitions. Hence, they can be considered as an extension of CTMCs with nondeterminism or, the other way around, as enriched labelled transition systems. Interactive transitions, denoted as $s \xrightarrow{\alpha} s'$, allow to model actions that are executed in zero time and account for nondeterministic choices by the system's environment. They allow very efficient bisimulation minimisation since quotienting can be done in a compositional fashion. A system's progress over time can be modelled by Markovian transitions, denoted by $s \overset{\lambda}{\dashrightarrow} s'$. They indicate that the system is moving from state s to s' after a delay exponentially distributed with parameter λ, and thereby account for time dependencies between system states.

IMCs are closely related to continuous-time Markov decision processes (CTMDPs), but they are strictly more expressive. CTMDPs closely entangle nondeterminism and stochastic behaviour in their transition relation. The separation of nondeterministic and stochastic choices allows for a well-behaved and natural notion of composition and hierarchy. Recently, IMCs were extended to Markov automata (MA) [23] by adding the possibility of random switching to interactive transitions.

Recent works on model checking opened the door for far-reaching industrial applications. IMCs provide a strict formal semantics of modelling and engineering formalisms such as generalised stochastic Petri nets (GSPNs) [50], Dynamic Fault Trees (DFTs) [12], the Architectural Analysis and Design Language (AADL) [13], and STATEMATE [10]. The powerful compositional design and verification approach of IMCs is applied for instance to Globally Asynchronous Locally Synchronous (GALS) designs [19], supervisory control [48, 49], state-of-the-art satellite design [24], water-treatment facilities [34] and train control systems [10].

This paper aims to give an extensive introduction to IMCs and survey recent developments in the field. Therefore, we present a detailed description of the fundamentals of the IMC formalism. Besides, we introduce related concepts such as CTMDPs and describe their relationship to IMCs. An important aspect of IMCs is that they can be analysed with respect to certain properties. Therefore, we introduce a logic that is capable of specifying important properties like "is the

system running at least 99% of the time?". Furthermore, we provide a rich set of model checking algorithms to efficiently compute and thus check these properties. Especially for time-bounded reachability, expected time and long-run average properties, we give an in-depth description of the algorithms with accompanying examples. Another challenge in a model like IMCs is the state space explosion problem. The prevention of this is a major research topic and covered by this paper in terms of bisimulation minimisation. Therefore, we present the notion of strong and weak bisimulation, and provide an algorithm for computing the bisimulation quotient.

Organisation of the paper. Section 2 introduces the model, semantics and compositional construction methods of IMCs. A survey on model checking techniques is provided in Section 3 and behavioural equivalences and abstraction are discussed in Section 4. Section 5 shows extensions of IMCs, Section 6 provides a number of case studies and applications, and Section 7 concludes the paper.

2 Preliminaries

This section summarises the basic notions and definitions to provide a formal underpinning of the concept of interactive Markov chains [35, 14] and related concepts. The interested reader can find more details in the referred material throughout this section.

Before we describe interactive Markov chains, we give a brief introduction to two widely used models which are related to them. We start with a discrete time and nondeterministic model, namely *Markov Decision Processes* (MDPs). They extend Markov chains by adding nondeterministic decisions.

Definition 1 (Markov Decision Process). *A Markov decision process (MDP) is a tuple* $\mathcal{M} = (S, Act, \mathbf{P}, s_0)$ *where* S *is a finite set of states,* Act *a finite set of actions,* s_0 *the initial state, and* $\mathbf{P}\colon S \times Act \times S \to [0,1]$ *the transition probability function such that* $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\}$ *for all* $s \in S$ *and* $\alpha \in Act$.

MDPs are a well studied model with a wide range of efficient algorithms [53] for various types of analysis. Later on in this survey, we exploit some of those algorithms to solve problems on interactive Markov chains.

Unsurprisingly, CTMDPs are the extension of MDPs to continuous time and are closely related to IMCs.

Definition 2 (Continuous Time Markov Decision Process). *A CTMDP is a tuple* $\mathcal{C} = (S, Act, \mathbf{R}, s_0)$ *where* S *is a finite set of states,* Act *a finite set of actions,* s_0 *the initial state, and* $\mathbf{R}\colon S \times Act \times S \to \mathbb{R}_{>0}$ *a three dimensional rate matrix.*

A CTMDP is a stochastic nondeterministic model that describes the behaviour of a system in continuous time. The delay of each transition (s_1, α, s_2) is exponentially distributed with rate $\mathbf{R}(s_1, \alpha, s_2)$ for $s_1, s_2 \in S$ and $\alpha \in Act$. IMCs extend CTMDPs by breaking the tight connection between nondeterministic and stochastic behaviour.

2.1 Interactive Markov Chains

The Syntax of an IMC. IMCs are finite transition systems with action-labelled interactive transitions, as well as Markovian transitions that are labelled with a positive real number identifying the rate of an exponential distribution. Hence, they strictly separate between interactive and Markovian behaviour. This enables for a wide range of modelling features. On the one hand, based on the action-labelled interactive transitions, IMCs can be used for compositional modelling with intermittent weak bisimulation [35]. On the other hand, the Markovian transitions allow to encode arbitrary distributions in terms of acyclic phase-type distributions [52]. An in depth discussion of the advantages of the IMC formalism is given in [14].

Definition 3 (Interactive Markov Chain). *An* interactive Markov chain *is a tuple* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *where* S *is a nonempty, finite set of states with* initial state $s_0 \in S$*,* Act *is a finite set of actions,* $\longrightarrow \subseteq S \times Act \times S$ *is a finite set of* interactive *transitions and* $\dashrightarrow \subseteq S \times \mathbb{R}_{>0} \times S$ *is a finite set of* Markovian *transitions.*

We abbreviate $(s, \alpha, s') \in \longrightarrow$ by $s \xrightarrow{\alpha} s'$ and $(s, \lambda, s') \in \dashrightarrow$ by $s \overset{\lambda}{\dashrightarrow} s'$. Let:
$- \; IT(s) = \{s \xrightarrow{\alpha} s'\}$ be the set of interactive transitions that leave s, and
$- \; MT(s) = \{s \overset{\lambda}{\dashrightarrow} s'\}$ be the set of Markovian transitions that leave s.
We denote with $MS \subseteq S$ the set of Markovian states, with $IS \subseteq S$ the set of interactive states and with $HS \subseteq S$ the set of hybrid states of an IMC $\mathcal{I}$, where:
$- \; s \in MS$ iff $MT(s) \neq \emptyset$ and $IT(s) = \emptyset$,
$- \; s \in IS$ iff $MT(s) = \emptyset$ and $IT(s) \neq \emptyset$, and
$- \; s \in HS$ iff $MT(s) \neq \emptyset$ and $IT(s) \neq \emptyset$.
Further, we distinguish *external* actions from *internal* τ-actions. Note that a labelled transition system (LTS) is an IMC with $MS = \emptyset$ and $HS = \emptyset$. Further, a continuous-time Markov chain (CTMC) is an IMC with $IS = \emptyset$ and $HS = \emptyset$. Therefore, IMCs are a natural extension of LTSs as well as CTMCs.

The Semantics of an IMC. A *distribution* μ over a countable set S is a function $\mu \colon S \rightarrowtail [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$. If $\mu(s) = 1$ for some $s \in S$, μ is a *Dirac* distribution, and is denoted μ_s. Let $Distr(S)$ be the set of all distributions over a set S. The interpretation of a Markovian transition $s \overset{\lambda}{\dashrightarrow} s'$ is that the IMC moves from state s to s' within d time units with probability $\int_0^d \lambda e^{-\lambda t} dt = (1 - e^{-\lambda \cdot d})$. For a state $s \in MS$, let $\mathbf{R}(s, s') = \sum\{\lambda \mid s \overset{\lambda}{\dashrightarrow} s'\}$ be the total rate to move from state s to s', and let $E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$ be the total outgoing rate of s. If s has multiple outgoing Markovian transitions to different successors, then we speak of a race between these transitions, known as the *race condition*. In this case, the probability to move from s to s' within d time units is $\frac{\mathbf{R}(s,s')}{E(s)} \cdot (1 - e^{-E(s)d})$, utilising that the IMC moves to a successor state s' after a delay of at most d time units with discrete branching probability $\mathbf{P}(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$. As defined on CTMDPs [6], uniformity can also be adapted to IMCs [40]. An IMC is called *uniform* iff there exists an $e \in \mathbb{R}_{\geq 0}$ such that

$\forall s \in MS$ it holds that $E(s) = e$. Thus, the distribution of the sojourn time is the same for all Markovian states if the IMC is uniform.

IMCs are compositional, i. e. if a system comprises several IMC components, then it can be assembled via *parallel composition* of the components. The components can communicate through external actions visible to all of them, while internal τ-actions are invisible and cannot communicate with any other action. Instead of communication, we say in the following that two IMCs synchronize on an action. Consider a state $s \in HS$ with a Markovian transition with rate λ and a τ-labelled transition. We assume that the τ-transition takes no time and is fired immediately since it is not subject to any interaction and cannot be delayed. On the other hand, the probability that the Markovian transition fires immediately is zero. Thus, internal interactive transitions always take precedence over Markovian transitions.

Assumption 1 (Maximal Progress) *In any IMC, internal interactive transitions take precedence over Markovian transitions.*

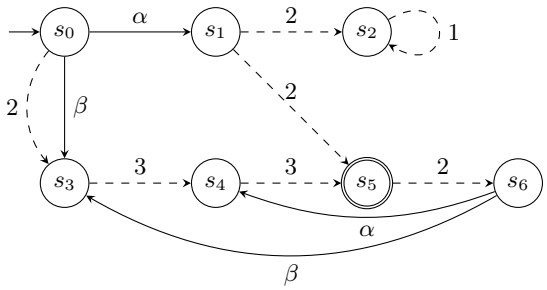


Figure 1: An interactive Markov chain.

Example 1. Let $\mathcal{I}$ be the IMC depicted in Figure 1. Then s_0 is a hybrid state with Markovian transition $s_0 \xrightarrow{2} s_3$ and interactive transitions $s_0 \xrightarrow{\alpha} s_1$ and $s_0 \xrightarrow{\beta} s_3$. We assume that all actions are no longer subject to any further synchronisation. W.l.o.g. we consider α and β as τ-transitions. Hence, we can apply the maximal progress assumption and obtain $s_0 \in IS$ with $s_0 \xrightarrow{\alpha} s_1$ and $s_0 \xrightarrow{\beta} s_3$. Therefore, in s_0 we have to choose between α and β. Since both transitions are fired without delay and take no precedence over each other, this choice has to be made nondeterministicly by a scheduler, see Section 2.3. The same holds for state s_6. If we choose β in s_0, then the successor state is s_3, which is a Markovian state with transition $s_3 \xrightarrow{3} s_4$ with rate $\lambda = 3$. The delay of this transition is exponentially distributed with parameter λ; thus, the transition fires in the next $z \in \mathbb{R}_{\geq 0}$ time units with probability $\int_0^z \lambda e^{-\lambda t} dt = (1 - e^{-3z})$. In

case we choose α in s_0 we reach state s_1, which has two Markovian transitions. We encounter a race condition, and the IMC moves along the transition whose delay expires first. Consequently, the sojourn time in s_1 is determined by the delay of the first transition that executes. The minimum of exponential distributions with parameters $\lambda_1, \lambda_2, \ldots$ is again exponentially distributed with the parameter $\lambda_1 + \lambda_2 + \cdots$. Thus, the sojourn time is determined by the exit rate, in our case we have $E(s_1) = 4$. The probability to move from a state $s \in MS$ to a successor state $s' \in S$ equals the probability that one of the outgoing Markovian transitions from s to s' wins the race. Therefore, the discrete branching probabilities for s_1 are given by $\mathbf{P}(s_1, s_2) = \mathbf{P}(s_1, s_5) = \frac{2}{4} = \frac{1}{2}$. ■

2.2 Behavioural and Measurability Concepts

In this section we define fundamental concepts relating to the behaviour and the measurability of IMCs. We start with the definition of paths and then define the σ-algebra over the set of paths.

Paths. Like in other transition systems, an execution in an IMC is described by a path. We define finite and infinite paths and provide several useful notations and operators relating to paths. Before proceeding with the definition, for the uniformity of notation, we use a distinguished action $\bot \notin Act$ to indicate Markovian transitions and extend the set of actions to $Act_\bot = Act \cup \{\bot\}$. Formally, a finite path is an initial state followed by a finite sequence of interactive and Markovian transitions annotated with times, i. e.

$$\pi \;=\; s_0 \xrightarrow{t_0,\alpha_0} s_1 \xrightarrow{t_1,\alpha_1} \cdots s_{n-1} \xrightarrow{t_{n-1},\alpha_{n-1}} s_n$$

with $\alpha_i \in Act_\bot$, $t_i \in \mathbb{R}_{\geq 0}$, $i = 0 \ldots n-1$ and $s_0 \ldots s_n \in S$. Each step of a path π describes how the IMC evolves from one state to the next; with what action and after spending what sojourn time. For example, when the IMC is in an interactive state $s \in IS$ where only internal actions are enabled, it must immediately (in zero time) choose an enabled action α and go to state s'. This gives rise to the finite path $s \xrightarrow{0,\alpha} s'$. On the other hand, if $s \in MS$, the IMC stays in s for $t > 0$ time units and then moves to the next state s' based on the distribution $\mathbf{P}(s, \cdot)$ by $s \xrightarrow{t,\bot} s'$.

For a finite path π we use $|\pi| = n$ as the length of π and $\pi{\downarrow} = s_n$ as the last state of π. Assume $k \leq n$ is an index, then $\pi[k] = s_k$ is the $k+1$-th state of π. Moreover, the time spent on π up to state $\pi[k]$ is denoted by $\Delta(\pi, k)$ which is zero if $k = 0$, and otherwise $\sum_{i=0}^{k-1} t_i$. We use $\Delta(\pi)$ as an abbreviation for $\Delta(\pi, |\pi|)$. For $t \leq \Delta(\pi)$, let $\pi@t$ denote the set of states that π occupies at time t. Note that $\pi@t$ is in general not a single state, but rather a set of states, as an IMC may exhibit immediate transitions and thus may occupy various states at the same time instant. Operator *Pref* extracts the prefix of length k from path π by $Pref(\pi, k) = s_0 \xrightarrow{t_0,\alpha_0} s_1 \cdots s_{k-1} \xrightarrow{t_{k-1},\alpha_{k-1}} s_k$. By removing the sojourn time from transitions of path π, we obtain a time-abstract path denoted by $\mathsf{abs}(\pi) \;=\; s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots s_{n-1} \xrightarrow{\alpha_{n-1}} s_n$. Furthermore, $Paths^n$ refers to the

Table 1: An example derivation of $\pi@t$ for IMCs.

$t \leq \Delta(\pi, i)$	0	1	2	3	4	5	6	7	$\min j$	$\max j$	$\pi@t$
0	✓	✓	✓	✓	✓	✓	✓	✓	0	3	$\langle s_0 s_1 s_2 s_3 \rangle$
$t_3 - \epsilon$	×	×	×	×	✓	✓	✓	✓	4	-	$\langle s_3 \rangle$
t_3	×	×	×	×	✓	✓	✓	✓	4	5	$\langle s_4 s_5 \rangle$
$t_3 + \epsilon$	×	×	×	×	×	×	✓	✓	6	-	$\langle s_5 \rangle$
$t_3 + t_5$	×	×	×	×	×	×	✓	✓	6	7	$\langle s_6 s_7 \rangle$

set of all paths with length n and $Paths^\star$ to the set of all finite paths. In this context, we add subscript abs for the set of time-abstract paths i. e. $Paths^n_{\mathsf{abs}}$ and $Paths^\star_{\mathsf{abs}}$. A (possibly time-abstract) path could be infinite which means it is constructed by an infinite sequence of (time-abstract) transitions. Accordingly, we use $Paths^\omega$ ($Paths^\omega_{\mathsf{abs}}$) to refer to the set of all (time-abstract) infinite paths.

Example 2. Consider the path

$$\pi = s_0 \xrightarrow{0,\alpha_0} s_1 \xrightarrow{0,\alpha_1} s_2 \xrightarrow{0,\alpha_2} s_3 \xrightarrow{t_3,\perp} s_4 \xrightarrow{0,\alpha_4} s_5 \xrightarrow{t_5,\perp} s_6 \xrightarrow{0,\alpha_6} s_7.$$

Let $0 < \epsilon < \min\{t_3, t_5\}$. The derivations for the sequence $\pi@0, \pi@(t_3-\epsilon), \pi@(t_3)$, $\pi@(t_3 + \epsilon)$ and $\pi@(t_3 + t_5)$ are depicted in Table 1, where ✓ indicates that $t \leq \Delta(\pi, i)$, and × denotes the states where $t > \Delta(\pi, i)$. Further, $\min j$ describes the minimum path length and $\max j$ the maximum path length such that $t \leq \Delta(\pi, j)$. Hence, with $\min j$, $\pi[j]$ describes the first state on path π for the sequence $\pi@t$, respectively for $\max j$ the last state. ■

σ-algebra. Here we recall the definition of σ-algebra for IMCs as described in [40, 50]. First we recapitulate the concept of *compound transitions*. A compound transition is a triple of (t, α, s), which describes the behaviour of the IMC when it waits in its current state for t time units then takes action α and finally evolves to the next state s. The set of all compound transitions over action space Act and state space S is denoted by $\mathrm{CT} = \mathbb{R}_{\geq 0} \times Act_\perp \times S$. As a path in IMCs is composed of a sequence of compound transitions originating from an initial state, first we define a σ-algebra over compound transitions and then extend it over finite and infinite paths. Let $\mathfrak{F}_S = 2^S$ and $\mathfrak{F}_{Act_\perp} = 2^{Act_\perp}$ be σ-algebras over S and $Act_\perp$, respectively. We define the σ-algebra over compound transitions using the concept of Cartesian product of a collection of σ-algebras [4], as $\mathfrak{F}_{\mathrm{CT}} = \sigma(\mathfrak{B}(\mathbb{R}_{\geq 0}) \times \mathfrak{F}_{Act_\perp} \times \mathfrak{F}_S)$, where $\mathfrak{B}(\mathbb{R}_{\geq 0})$ is the Borel σ-algebra over non-negative reals. Furthermore, it can be extended to the σ-algebra over finite paths using the same technique as follows. Let $\mathfrak{F}_{Paths^n} = \sigma\left(\mathfrak{F}_S \times \prod_{i=1}^{n} \mathfrak{F}_{\mathrm{CT}}\right)$ be the σ-algebra over finite paths of length n, then the σ-algebra over finite paths is defined as $\mathfrak{F}_{Paths^\star} = \cup_{i=0}^{\infty} \mathfrak{F}_{Paths^n}$. The σ-algebra over infinite paths is defined using the standard cylinder set construction [4]. We define the cylinder set of a given base $B \in \mathfrak{F}_{Paths^n}$ as $Cyl(B) = \{\pi \in Paths^\omega : Pref(\pi, n) \in B\}$. $Cyl(B)$ is measurable if its base B is measurable. The σ-algebra over infinite paths, $\mathfrak{F}_{Paths^\omega}$, is therefore

the smallest σ-algebra over measurable cylinders. Finally the σ-algebra over the set of paths is the disjoint union of the σ-algebras over the finite paths and the infinite paths.

2.3 Schedulers

In states with more than one outgoing interactive transition the choice of the transition to be taken is nondeterministic, just as in the LTS setting. This nondeterminism is resolved by schedulers. Different classes of schedulers exist in order to resolve nondeterminism for different kinds of objectives. The most general scheduler class maps a finite path to a distribution over the set of interactive transitions that are enabled in the last state of the path:

Definition 4 (Generic Measurable Scheduler [40]). *A* generic scheduler *over IMC* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *is a function,* $D\colon Paths^\star \rightarrowtail Distr(\longrightarrow)$*, where the support of* $D(\pi)$ *is a subset of* $(\{\pi\downarrow\} \times Act \times S) \cap \longrightarrow$ *and* $\pi\downarrow \in IS$*. A generic scheduler is* measurable *iff for all* $T \subseteq \longrightarrow$*,* $D(\cdot)(T) : Paths^\star \rightarrowtail [0,1]$ *is measurable.*

For a finite path π ending in an interactive state, a scheduler specifies how to resolve nondeterminism by defining a distribution over the set of enabled transitions of $\pi\downarrow$. Measurability of scheduler D means that it never resolves nondeterminism in a way that induces a set of paths that is not measurable, i. e. $\{\pi \mid D(\pi)(T) \in B\} \in \mathfrak{F}_{Paths^\star}$ for all $T \subseteq \longrightarrow$ and $B \in \mathfrak{B}([0,1])$, where $\mathfrak{B}([0,1])$ is the Borel σ-algebra over interval $[0,1]$. We use the term $\mathcal{GM}$ to refer to the set of all generic schedulers. Since schedulers in IMCs are closely related to schedulers in CTMDPs, most of the concepts are directly applied from the latter to the former. A slight difference is that schedulers in IMCs resolve nondeterminism only for finite paths that end up in interactive states.

A variety of scheduler classes in CTMDPs [40, 50], which can also be employed in IMCs, has been proposed in order to resolve nondeterminism for different kinds of objectives. These schedulers are classified according to the level of time and history details they use to resolve nondeterminism. Another criterion is whether they are *deterministic*, i. e. the distribution over the set of target transitions is Dirac, or *randomised*. In *history-dependent* schedulers the resolution of nondeterminism on an interactive state may depend on the path is visited upto the state. A scheduler is *hop counting* if all finite paths with the same length lead to the same resolution of nondeterminism. It is *positional* if its decision for a given path is only made based on the last state of the path. On the other hand, schedulers can be *time-dependent*, *total time-dependent* or *time-abstract*. Time-dependent schedulers utilise the whole timing information of a path including the sojourn time of all intermediate states for resolution of nondeterminism, while total time-dependent schedulers only employ the total time that has elapsed to reach the current state for that purpose. No timing information is used by time-abstract schedulers and a path is thus considered time-abstract by them.

The most general class, $\mathcal{GM}$ schedulers, uses the complete trajectory up to the current interactive state to randomly determine the next state. Therefore, they are also called *time- and history-dependent randomised* (THR) schedulers. The class has an excessive power which is not necessary for most types of analysis. For example, for time-abstract criteria like expected reachability, long-run average and unbounded reachability, it suffices to consider *time-abstract positional deterministic* (TAPD) schedulers [27], which are also called *stationary deterministic*. Furthermore, the optimal scheduler for computing time-bounded reachability probabilities is *total time-dependent positional deterministic* (TTPD) [50]. More classes of randomised schedulers are depicted in Table 2. The deterministic version of each class can be obtained under the assumption that $\mathit{Distr}(\longrightarrow)$ is Dirac.

Table 2: Randomised scheduler classes for IMCs. The classification criteria are denoted by TA (Time-Abstract), TT (Total Time-dependent), T (Time-dependent), P (Positional), HOP (HOP counting) and H (History-dependent).

		Abbreviation	Scheduler Signature	Parameters of Scheduler for a given path π
TA	P	TAPR	$D : \mathit{IS} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\pi\downarrow \in \mathit{IS}$
	HOP	TAHOPR	$D : \mathit{IS} \times \mathbb{N} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\pi\downarrow \in \mathit{IS}, \lvert\pi\rvert$
	H	TAHR	$D : \mathit{Paths}^{\star}_{\mathsf{abs}} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\mathsf{abs}(\pi)$ with $\pi\downarrow \in \mathit{IS}$
TT	P	TTPR	$D : \mathit{IS} \times \mathbb{R}_{\geq 0} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\pi\downarrow \in \mathit{IS}, \Delta(\pi)$
	HOP	TTHOPR	$D : \mathit{IS} \times \mathbb{N} \times \mathbb{R}_{\geq 0} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\pi\downarrow \in \mathit{IS}, \lvert\pi\rvert, \Delta(\pi)$
	H	TTHR	$D : \mathit{Paths}^{\star}_{\mathsf{abs}} \times \mathbb{R}_{\geq 0} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\mathsf{abs}(\pi)$ with $\pi\downarrow \in \mathit{IS}, \Delta(\pi)$
T	P	TPR	$D : \mathit{IS} \times \mathbb{R}_{\geq 0} \rightarrowtail \mathit{Distr}(\longrightarrow)$	$\pi\downarrow \in \mathit{IS}$, $\Delta(\pi, \lvert\pi\rvert) - \Delta(\pi, \lvert\pi\rvert - 1)$
	H	THR ($\mathcal{GM}$)	$D : \mathit{Paths}^{\star} \rightarrowtail \mathit{Distr}(\longrightarrow)$	π with $\pi\downarrow \in \mathit{IS}$

Example 3. We define a scheduler over the IMC in Figure 1, which always chooses action α in state s_0 with probability 1. In addition, it selects α and β in state s_6 with probability p and $1-p$, respectively, provided that a path in the set $A(T_1, T_5) = \{s_0 \xrightarrow{0,\alpha} s_1 \xrightarrow{t_1,\perp} s_5 \xrightarrow{t_5,\perp} s_6 : t_1 < T_1 \wedge t_5 < T_5\}$ has been observed. Otherwise, action β (in state s_6) is almost surely picked. Assume that $p = 0.5$, $T_1 = 1$ and $T_5 = 3$, then the scheduler is in the THR ($\mathcal{GM}$) class. It becomes deterministic (THD) by setting $p = 1$ or $p = 0$. By taking $p = 1$, $T_1 = \infty$ and $T_5 = \infty$, $A(T_1, T_5)$ becomes time-abstract and the scheduler is then time-abstract history-dependent deterministic (TAHD). On the other hand when $A(T_1, T_5)$ is replaced by the set $B = \{\pi \in \mathit{Paths}^{\star} : \pi\downarrow = s_6 \wedge \Delta(\pi, \lvert\pi\rvert) \leq 4\}$,

the scheduler is total time-dependent and positional deterministic (TTPD) or randomised (TTPR), depending on the value of p. ■

2.4 Probability Measures

The model induced by an IMC after the nondeterministic choices are resolved by a scheduler is pure stochastic and then can be analysed. To that end the unique probability measure [40, 50] for probability space $(Paths^{\omega}, \mathfrak{F}_{Paths^{\omega}})$ is proposed. Given a state s, a general measurable scheduler D and a set Π of infinite paths, then $\Pr_{s,D}(\Pi)$ denotes the probability of visiting all paths in Π under scheduler D starting from state s. We omit the details due to lack of space.

Zenoness. Due to the presence of immediate state changes, an IMC might exhibit *Zeno behaviour*, where infinitely many interactive transitions are taken in finite time. This is an unrealistic phenomenon characterised by paths π, where $\Delta(\pi, n)$ for $n \to \infty$ does not diverge to ∞. In other words, the time spent in the system may stop increasing if the system follows path π. Accordingly, an IMC $\mathcal{I}$ with initial state s_0 is non-Zeno, if for all schedulers D, $\Pr_{s_0,D}(\{\pi \in Paths^{\omega} \mid \lim_{n\to\infty} \Delta(\pi, n) < \infty\}) = 0$. As the probability of a Zeno path in a finite CTMC is zero [5], IMC $\mathcal{I}$ is non-Zeno, if and only if no strongly connected component with states $T \subseteq IS$ is reachable from s_0. In the remainder of this paper we restrict to models without zenoness.

2.5 Composition

Compositionality is one of the key properties of IMCs. Complex models consisting of various interacting IMCs can be aggregated in a stepwise manner. This allows e. g. to model each subsystem separately and obtain a model of the whole system by applying the following parallel composition.

Definition 5 (Parallel Composition). *Let* $\mathcal{I}_1 = (S_1, Act_1, \longrightarrow_1, \dashrightarrow_1, s_{0,1})$ *and* $\mathcal{I}_2 = (S_2, Act_2, \longrightarrow_2, \dashrightarrow_2, s_{0,2})$ *be IMCs. The* parallel composition *of* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *wrt. synchronisation set* $Syn \subseteq (Act_1 \cap Act_2) \setminus \{\tau\}$ *of actions is defined by:*

$$\mathcal{I}_1 \| \mathcal{I}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, \dashrightarrow, (s_{0,1}, s_{0,2}))$$

where $\longrightarrow$ *and* $\dashrightarrow$ *are defined as the smallest relations satisfying*

1. $s_1 \xrightarrow{\alpha}_1 s_1'$ *and* $s_2 \xrightarrow{\alpha}_2 s_2'$ *and* $\alpha \in Syn$, $\alpha \neq \tau$ *implies* $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2')$
2. $s_1 \xrightarrow{\alpha}_1 s_1'$ *and* $\alpha \notin Syn$ *implies* $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2)$ *for any* $s_2 \in S_2$
3. $s_2 \xrightarrow{\alpha}_2 s_2'$ *and* $\alpha \notin Syn$ *implies* $(s_1, s_2) \xrightarrow{\alpha} (s_1, s_2')$ *for any* $s_1 \in S_1$
4. $s_1 \overset{\lambda}{\dashrightarrow}_1 s_1'$ *implies* $(s_1, s_2) \overset{\lambda}{\dashrightarrow} (s_1', s_2)$ *for any* $s_2 \in S_2$
5. $s_2 \overset{\lambda}{\dashrightarrow}_2 s_2'$ *implies* $(s_1, s_2) \overset{\lambda}{\dashrightarrow} (s_1, s_2')$ *for any* $s_1 \in S_1$.

The two IMCs have to synchronise on actions in *Syn*, i. e. any action $\alpha \in Syn$ needs to be performed by both IMCs at the same time, except if α is an internal action (first condition). The second and third conditions state that any other

action can be performed autonomously by any of the two IMCs. According to the last two conditions, Markovian transitions are interleaved independently. This is justified by the memoryless property of the annotated exponential distributions.

Given a set of IMCs B which need to be synchronised, the computational effort of the composition process is crucially dependent on the order in which these IMCs are aggregated. Crouzen and Hermanns [20] suggested an algorithm based on heuristics to determine a composition order which induces low computing costs. In a first step, the algorithm determines candidate subsets of B up to a certain size. For each subset a metric is calculated which estimates how good the composition of the IMCs in this subset is in keeping the cost of the overall composition low. The IMCs in the subset with the maximal metric are then composed and minimised, as described in Section 4. This process iterates until only one IMC remains in B.

The composition of two or more IMCs involves two steps: After synchronisation on a set of actions, those actions which require no further synchronisation are hidden.

Definition 6 (Hiding). *The* hiding *IMC* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *wrt. the set* A *of actions is the IMC* $\mathcal{I}\backslash A = (S, Act\backslash A, \longrightarrow', \dashrightarrow, s_0)$ *where* $\longrightarrow'$ *is the smallest relation defined by*

1. $s \xrightarrow{\alpha} s'$ *and* $\alpha \notin A$ *implies* $s \xrightarrow{\alpha}{}' s'$
2. $s \xrightarrow{\alpha} s'$ *and* $\alpha \in A$ *implies* $s \xrightarrow{\tau}{}' s'$

Through hiding, interactive transitions annotated with actions in A are transformed into τ-transitions. Further, we distinguish between two classes of IMCs:

- *closed* IMCs, where all interactive transitions are hidden, such that the IMC is not subject to any further synchronisation, and
- *open* IMCs, which still have visible interactive transitions, and can interact with other IMCs.

As we will see next, closed IMCs are closely related to CTMDPs.

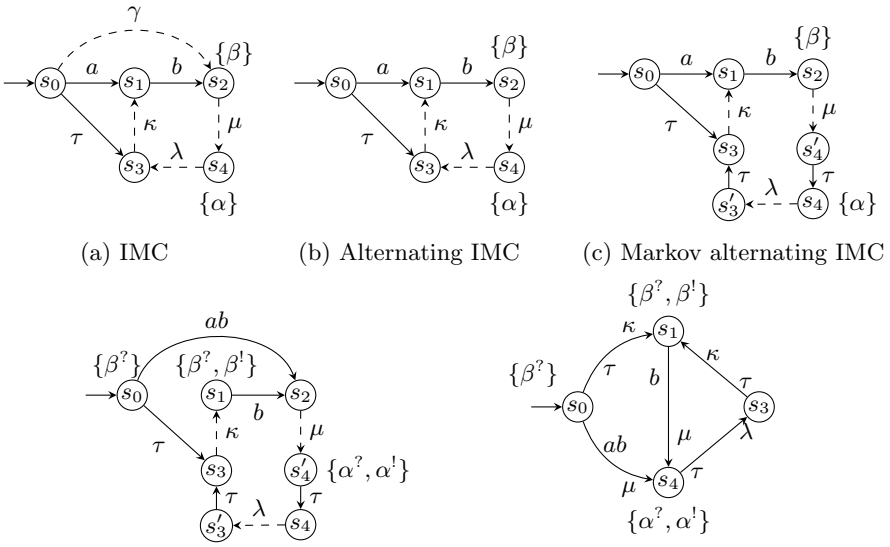
2.6 IMCs versus CTMDPs

The modelling of a system usually involves the composition of various communicating subsystems. Therefore, open IMCs are used to describe those subsystems. Once all open IMCs are composed to a single closed IMC, it is subject to analysis. Note that a CTMDP combines the two transition relations of an IMC in one transition rate matrix. We recapitulate a transformation from an IMC to a CTMDP [40, 38, 37] which preserves important properties of the original IMC, and thus can be used to apply CTMDP analysis techniques [16, 6] on the transformed model.

IMC vs CTMDP. In general, closed IMCs are a generalisation of CTMDPs in which interactive and Markovian transitions are loosely coupled. Therefore, every CTMDP can be converted into an equivalent IMC in a straightforward way. The equivalent IMC is contained in a restricted subclass called *strictly alternating IMCs* that behaves exactly like CTMDPs. Note that in a strictly alternating IMC, Markovian and interactive transitions are exhibited in a strict alternation. The idea of the transformation from an IMC to a CTMDP [40] is to convert a given IMC to a strictly alternating IMC which is essentially a CTMDP.

Given an IMC $\mathcal{I}$, the following steps [38] are applied: (1) obtain an *alternating IMC* by transformation of hybrid states into interactive states, (2) turn all successors of any Markovian state into interactive states to obtain a *Markov Alternating IMC*, (3) transform any immediate successor of all interactive states into Markovian states to obtain an *Interactive Alternating IMC*. By employing these transformation steps, an arbitrary IMC turns into a strictly alternating IMC. The strictly alternating IMC can then be transformed into a corresponding CTMDP in a straightforward way. Here we explain each step by an example.

Alternating IMC. In the first step, IMC $\mathcal{I}$ is transformed into an alternating IMC which does not contain any hybrid state. Owing to closeness of the IMC and imposing Assumption 1 interactive transitions take precedence over Markovian transitions. Hence all emanating Markovian transitions of a hybrid state can be safely eliminated.



(a) IMC (b) Alternating IMC (c) Markov alternating IMC

(d) Strictly alternating IMC (e) Final CTMDP

Figure 2: Step by step transformation of an IMC into a CTMDP

Markov Alternating IMC. The aim of the second step is to make sure that predecessors and successors of any Markov state are interactive. In this step, a fresh interactive state with internal action τ is inserted in between two consecutive Markovian states. Due to immmediate firing of the τ transition, the timing behaviour of the IMC is preserved.

Example 4. The state-labelled IMC (see Section 3) in Figure 2a is closed and subject to analysis. The result of the first two steps of the transformation, namely the alternating IMC and the Markov alternating IMC, are illustrated in Figures 2b and 2c, respectively. ■

Strictly Alternating IMC. After this step we make sure that there is no sequence of interactive transitions, therefore each interactive state is preceded and succeeded by Markovian states. As discussed earlier, a sequence of consecutive interactive transitions occur in zero time and thus can be seen as a single transition labelled by a word of taken actions. Note that the sequence always ends in a Markovian state. There are interactive states in between that have only outgoing and incoming interactive transitions, which are eliminated from the state space. We call those states *vanishing*, and all others *persistent*.

The above transformation is not enough to reconstruct all information from the original model. In order to preserve the semantic structure of the model after eliminating vanishing states, their state labels (atomic propositions) must be adopted by persistent states. In this regard, state labels are decorated with an extra *may* and/or *must* tag. In other words, if starting from an interactive persistent state s, all sequences of interactive transitions ending in Markovian states visit label α, then s will be labelled by $\alpha^!$ (s *must* satisfy α). On the other hand, if there exists such a sequence, s will be labelled by $\alpha^?$ (s *may* satisfy α). Note that *must* labelling implies *may* labelling, as a label that must occur, may also occur. At the end since all labelling information is inherited by interactive persistent states, labels of other states will be removed.

An alternating IMC is transformed into a strictly alternating one after the specified Markov and interactive alternating steps are applied. Since in a strictly alternating IMC, Markovian and interactive transitions exhibit in a strict alternation, the strictly alternating IMC can be interpreted as a CTMDP. It has been proven [38, 40] that the above transformation steps preserve the uniformity and timed reachability of the original model. The transformation is a crucial part of the evaluation of STATEMATE models as will be discussed in Section 6.2.

Example 5. The result of the transformation into the strictly alternating IMC is shown in Figure 2d and the transformed CTMDP is illustrated in Figure 2e. ■

3 Model Checking

Consider we are confronted with a IMC originated from some high level-formalism and a performability requirement. How can one describe this performability

property and then compute the set of satisfying states in the IMC? First of all we need a logic representing the desired property. Then the basic computational procedure of the satisfaction set is a simple recursive descent of the logical formulae.

In this section we provide an overview of the current model checking capabilities of IMCs to provide an answer to the preceded question. We first introduce a logic which is used to specify a wide range of properties and thereafter describe algorithms to check those properties for IMCs.

3.1 Continuous Stochastic Logic

This section describes Continuous Stochastic Logic [5] (CSL), which is suitable to express a broad range of performance and dependability measures. CSL is an extension of Probabilistic Computation Tree Logic (PCTL) [30, 9] to continuous-time Markov models. This section reviews CSL and its related model checking algorithms as introduced in [59, 50] and enriches it with expected reachability and long-run average operators as described in [27]. CSL works on *state-labelled* IMCs.

Definition 7 (State-Labelled IMC). *A state-labelled IMC is a tuple* $\mathcal{I} = (S, Act, \rightarrow, \dashrightarrow, s_0, L)$ *where* $L : S \rightarrowtail 2^{AP}$ *is a* state labelling function *with* AP *as a set of* atomic propositions*. All other elements are as in Definition 3.*

Hence, given an IMC $\mathcal{I}$ and a finite set of *atomic propositions* AP, a *state labelling function* $L : S \rightarrowtail 2^{AP}$ decorates each state with a set of atomic propositions which do hold in that state.

Syntax of CSL. Let $\mathfrak{I}$ be the set of all nonempty nonnegative real intervals with real bounds, then Continuous Stochastic Logic (CSL) for IMCs is defined as follows.

Definition 8 (CSL Syntax). *Let* $a \in AP$, $\mathrm{p} \in [0,1]$, $\mathrm{t} \in \mathbb{R}_{\geq 0}$, $I \in \mathfrak{I}$ *an interval and* $\trianglelefteq \in \{<, \leq, \geq, >\}$*, CSL state and path formulae are described by*

$$\Phi ::= a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\trianglelefteq \mathrm{p}}(\phi) \mid \mathcal{E}_{\trianglelefteq \mathrm{t}}(\Phi) \mid \mathcal{L}_{\trianglelefteq \mathrm{p}}(\Phi)$$
$$\phi ::= \mathcal{X}^I \Phi \mid \Phi \,\mathcal{U}\, \Phi \mid \Phi \,\mathcal{U}^I\, \Phi$$

Except for the last two operators of the state formulae this logic corresponds to the CSL logic defined in [59]. Note that $\mathcal{P}_{\trianglelefteq \mathrm{p}}(\phi)$ denotes the probability of the set of paths that satisfy ϕ. The formula $\mathcal{E}_{\trianglelefteq \mathrm{t}}(\Phi)$ describes the expected time to reach some state satisfying Φ and $\mathcal{L}_{\trianglelefteq \mathrm{p}}(\Phi)$ denotes the average time spent in states satisfying Φ in the long-run.

Given an infinite path $\pi \in Paths^{\omega}$, π satisfies $\mathcal{X}^I \Phi$ if the first transition of π occurs within time interval I and leads to a state that satisfies Φ. Similarly, the bounded until formula $\Phi \mathcal{U}^I \Psi$ is satisfied by π if π visits states that satisfy formula Φ until it reaches a state that satisfies formula Ψ within the time interval I. In contrast to the bounded until, an unbounded until formula does not constrain the time at which π may visit a state which satisfies Ψ. This corresponds to the time interval $[0, \infty)$.

Semantics of CSL. To define the semantics of CSL we first introduce some important notations. We denote with $\gamma(\pi, n)$ the time interval during which a given path π stays in its n-th state. More formally, it equals $[\Delta(\pi, n), \Delta(\pi, n+1)]$ if $\Delta(\pi, n) < \Delta(\pi, n+1)$, and $\{\Delta(\pi, n)\}$ otherwise. Let $V_\Phi : Paths \to \mathbb{R}^{\infty}_{\geq 0}$ be the random variable which defines the elapsed time before visiting some state $s \vDash \Phi$ for the first time. In other words, for an infinite path $\pi = s_0 \xrightarrow{\sigma_0, t_0} s_1 \xrightarrow{\sigma_1, t_1} \cdots$ we have $V_\Phi(\pi) = \min\{t \in \mathbb{R}_{\geq 0} \mid s \in \pi@t \wedge s \vDash \Phi\}$. Furthermore, let $\mathbf{I}_\Phi$ be the characteristic function of Φ, such that $\mathbf{I}_\Phi(s) = 1$ if $s \vDash \Phi$ and otherwise 0. The fraction of time spent in states satisfying Φ on an infinite path π is given by the random variable $A_\Phi(\pi) = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbf{I}_\Phi(\pi@u)du$ [2, 46]. The formal semantics of CSL formulae is then defined as follows.

Definition 9 (CSL Semantics). *Let $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, AP, L, \nu)$ be a state-labelled IMC, $s \in S$, $a \in AP$, $\mathrm{p} \in [0, 1]$, $\mathrm{t} \in \mathbb{R}_{\geq 0}$, $I \in \mathfrak{I}$, $\trianglelefteq \in \{<, \leq, \geq, >\}$, and $\pi \in Paths^\omega$. We define the satisfaction relation $\vDash$ for state formulae: $s \vDash a$ iff $a \in L(s)$, $s \vDash \neg\Phi$ iff $s \nvDash \Phi$, $s \vDash \Phi \wedge \Psi$ iff $s \vDash \Phi \wedge s \vDash \Psi$, and*

$$
\begin{aligned}
s \vDash \mathcal{P}_{\trianglelefteq \mathrm{p}}(\phi) \quad & \text{iff } \forall D \in \mathcal{GM}.\ \Pr_{s,D}(\{\pi \in Paths^\omega \mid \pi \vDash \phi\}) \trianglelefteq \mathrm{p} \\
s \vDash \mathcal{E}_{\trianglelefteq \mathrm{t}}(\Phi) \quad & \text{iff } \forall D \in \mathcal{GM}.\ \int_{Paths^\omega} V_\Phi(\pi)\ \Pr_{s,D}(\,\mathrm{d}\pi) \trianglelefteq \mathrm{t} \\
s \vDash \mathcal{L}_{\trianglelefteq \mathrm{p}}(\Phi) \quad & \text{iff } \forall D \in \mathcal{GM}.\ \int_{Paths^\omega} A_\Phi(\pi) \Pr_{s,D}(\,\mathrm{d}\pi) \trianglelefteq \mathrm{p}
\end{aligned}
$$

For path formulae:

$$
\begin{aligned}
\pi \vDash & \mathcal{X}^I \Phi \quad \text{iff} \quad \pi[1] \vDash \Phi \wedge \Delta(\pi, 1) \in I \\
\pi \vDash & \Phi\, \mathcal{U}^I\, \Psi \quad \text{iff} \quad \exists n \in \mathbb{N}_0.\gamma(\pi, n) \cap I \neq \emptyset \wedge \pi[n] \vDash \Psi \wedge \forall k = 0 \ldots n-1.\pi[k] \vDash \Phi \\
\pi \vDash & \Phi\, \mathcal{U}\, \Psi \quad \text{iff} \quad \exists n \in \mathbb{N}_0.\pi[n] \vDash \Psi \wedge \forall k = 0 \ldots n-1.\pi[k] \vDash \Phi
\end{aligned}
$$

Example 6. Consider a system with the two atomic propositions *up* and *down*. We are interested in the availability of the system and want to know if we are in an *up* state at least 90 percent of the time. This CSL property is described with the long-run average operator $\mathcal{L}_{\geq 0.9}(up)$. It is satisfied, if we are in the set of *up* states with more than 90% in the long-run. We denote the states that satisfy this property with the atomic proposition *available*.

Besides the availability of the system, we are also interested in its safety. Therefore, we want to validate that the probability to reach a *down* state via *up* state is at most 0.01 during the first 5 time units . This condition is expressed by the CSL formula $\mathcal{P}_{\leq 0.01}(up\ \mathcal{U}^{[0,5]}\ down)$. We denote all states that satisfy this property with the atomic proposition *safe*.

With these propositions, one can e.g. investigate if the average time to reach some *available* and *safe* state is at most 10 time units. This is be determined by the CSL formula $\mathcal{E}_{\leq 10}(available \wedge safe)$. ■

3.2 Probability Bounds

Model checking a CSL formula Φ over an IMC $\mathcal{I}$ entails the computation of all sub-formulas Ψ of Φ by determining the satisfaction sets $Sat(\Psi) = \{s \in S \mid s \vDash \Psi\}$.

Just like for other branching-time logics, we recursively compute those sets by starting with the inner most formula, represented by an atomic proposition. In general, we have $Sat(a) = \{s \in S \mid a \in L(s)\}$ for an atomic proposition $a \in AP$, $Sat(\neg\Psi) = S \setminus Sat(\Psi)$ for negation formulae, and $Sat(\Psi_1 \wedge \Psi_2) = Sat(\Psi_1) \cap Sat(\Psi_2)$ for the conjunction of formulae.

Probability Bounds. The proper calculation of $Sat(\mathcal{P}_{\trianglelefteq \mathrm{p}}(\phi))$, however, requires deeper considerations. $Sat(\mathcal{P}_{\trianglelefteq \mathrm{p}}(\phi))$ is defined as:

$$\{s \in S \mid \forall D \in \mathcal{GM}.\ \Pr_{s,D}(\{\pi \in Paths^{\omega} \mid \pi \vDash \phi\}) \trianglelefteq \mathrm{p}\}.$$

In a nutshell, determining this set requires the calculation of the maximum or minimum (depending on $\trianglelefteq$) probability measures induced by all ϕ-satisfying paths starting from state s, where the maximum or minimum are to be taken over all measurable schedulers. Let $p^{\mathcal{I}}_{\max}(s,\phi)$ and $p^{\mathcal{I}}_{\min}(s,\phi)$ be those values respectively. In the following, we show how to compute them for different types of path formulae ϕ. We only consider the maximum, since the minimum can be handled analogously.

Next Formula Assume that $\phi = \mathcal{X}^I\Phi$ and $Sat(\Phi)$ have been already computed. Let $a = \inf I$ and $b = \sup I$. If $s \in MS$ is a Markovian state, then nondeterminism does not occur, and the computation can be done as for CTMCs [5], i.e. $p^{\mathcal{I}}_{\max}(s, \mathcal{X}^I\Phi) = \sum_{s' \in Sat(\Phi)} \mathbf{P}(s,s')(e^{-E(s)a} - e^{-E(s)b})$. For $s \in IS$, we determine the possibility to move directly from s to a Φ-satisfying state. Hence $p^{\mathcal{I}}_{\max}(s, \mathcal{X}^I\Phi) = 1$ if $\exists s' \in S, \alpha \in Act. s \xrightarrow{\alpha} s' \wedge s' \vDash \Phi \wedge 0 \in I$, and it is zero otherwise.

Unbounded Until Formula The evaluation of a given unbounded until formula in an IMC can be reduced to the computation of *unbounded reachability*, which in turn can be reduced to the computation of reachability in a time-abstract model. It utilises the same technique that is used for the model checking of an unbounded until formula in CTMCs [5]. Let $\mathcal{I}$ be an IMC and $\phi = \Phi\ \mathcal{U}\ \Psi$ be an unbounded until formula. We assume that $Sat(\Phi)$ and $Sat(\Psi)$ have already been computed. At first, we reduce the problem to the computation of unbounded reachability in the IMC $\mathcal{I}_{\neg\Phi}$, which is built by turning all states $Sat(\neg\Phi)$ in $\mathcal{I}$ into absorbing states. This is achieved by replacing all outgoing transitions of these states with a single Markovian self loop with an arbitrary rate, so that once a path has entered an absorbing state it cannot leave it anymore. The reasoning behind this transformation is that as soon as a path reaches some state in $Sat(\neg\Phi) \setminus Sat(\Psi)$, regardless of which states will be visited in future, it does not satisfy ϕ. Consequently, making these states absorbing does not affect the evaluation of an unbounded until formula. More formally, let $\Diamond G$ be the set of paths that eventually reach some goal states $G \subseteq S$, then $\forall s \in S.\ p^{\mathcal{I}}_{\max}(s, \Phi\ \mathcal{U}\ \Psi) = p^{\mathcal{I}_{\neg\Phi}}_{\max}(s, \Diamond Sat(\Psi))$.

In a second step, the unbounded reachability problem in $\mathcal{I}_{\neg\Phi}$ can be transformed into the computation of unbounded reachability in a time-abstract model.

We can use a time-abstract model, since the sojourn time in Markovian states is not of importance in the evaluation of unbounded reachability. In other words, it does not matter at which point in time a transition from a Markovian state s to its successor s' occurs. It is sufficient to know the probability $\mathbf{P}(s, s')$ of eventually reaching s' from s. Therefore, it suffices to compute the unbounded reachability in a discrete model in which all interactive transitions of $\mathcal{I}_{\neg\Phi}$ are mimicked and all Markovian transitions are replaced with the corresponding discrete branching probabilities. The discrete model is called the embedded Markov Decision Process induced from $\mathcal{I}_{\neg\Phi}$ and denoted as $emb(\mathcal{I}_{\neg\Phi})$. Formally speaking, the unbounded reachability property in $\mathcal{I}_{\neg\Phi}$ is preserved by the transformation in its embedded MDP, or $\forall s \in S.\ p^{\mathcal{I}_{\neg\Phi}}_{\max}(s, \Diamond Sat(\Psi)) = p^{emb(\mathcal{I}_{\neg\Phi})}_{\max}(s, \Diamond Sat(\Psi))$. In the final step, we can compute the unbounded reachability property in $emb(\mathcal{I}_{\neg\Phi})$ by using, for example, the algorithms described in [7, Chapter 10].

Time-Bounded Until Formula The computation of a time-bounded until formula is more complicated and requires some innovation. As above, the problem can be transformed into the computation of reachability in a first step. Let $\mathcal{I}$ be an IMC, $\phi = \Phi\,\mathcal{U}^I\,\Psi$ with $I \in \mathfrak{I}$ be a CSL formula, and $\Diamond^I G$ denote the set of paths that reach goal states $G \subseteq S$ within interval I. We assume that $Sat(\Phi)$ and $Sat(\Psi)$ has been already computed. Similarly to the unbounded until, all states in $Sat(\Psi)$ are considered to be goal states and all states in $Sat(\neg\Phi)$ are made absorbing. The analysis of time-bounded until analysis is then replaced by the analysis of time-bounded reachability, utilising the following theorem.

Theorem 1 (Bounded Until [50]). *Let $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ be an IMC as before, and $\phi = \Phi\,\mathcal{U}^I\,\Psi$ with $I \in \mathfrak{I}$ be a CSL path formula and $G = Sat(\Psi)$. We construct $\mathcal{I}_{\neg\Phi}$ from $\mathcal{I}$ by making all states in $Sat(\neg\Phi)$ absorbing. Then $\forall s \in S.\ p^{\mathcal{I}}_{\max}(s, \Phi\,\mathcal{U}^I\,\Psi) = p^{\mathcal{I}_{\neg\Phi}}_{\max}(s, \Diamond^I G)$.*

The computation of time-bounded reachability is explained in the following section.

3.3 Time-Bounded Reachability

This section presents the algorithm introduced in [59, 50] which approximates the probabilities of a time-bounded reachability analysis in IMCs. The algorithm is based on a discretisation technique with a predefined approximation error. Given IMC $\mathcal{I}$, interval $I \in \mathfrak{I}$, a set of goal states $G \subseteq S$ and $s \in S$, the technique provides a fixpoint characterisation for the computation of $p^{\mathcal{I}}_{\max}(s, \Diamond^I G)$ (and similarly for $p^{\mathcal{I}}_{\min}(s, \Diamond^I G)$). The characterisation implies that TTPD schedulers are sufficient for this purpose, i. e. $p^{\mathcal{I}}_{\max}(s, \Diamond^I G) = \sup_{D \in \mathrm{TTPD}} \Pr_{s,D}(\Diamond^I G)$. In other words, it suffices to find the optimal scheduler among all TTPD schedulers, which maximises time-bounded reachability. Note that similar results exist for the minimum.

Example 7. Consider the IMC in Figure 3 and assume we want to compute the maximum reachability probability from the initial state s_0 to the goal state s_5

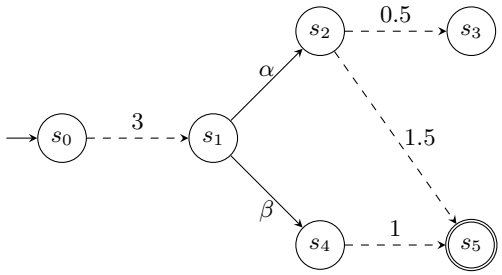


Figure 3: An exemplary IMC.

within 3 time units. Thanks to the simple structure of the IMC, the fixpoint characterisation gives us the closed form of the maximum reachability as well as the optimal TTPD schedule. The optimal decision in state s_1 depends on the time when it is visited. Hence, the scheduler takes action α if the time is less than $3 - \ln(3)$ time units, and action β otherwise. ■

The fixpoint characterisation yields an integral equation system which is in general not tractable [5]. To circumvent this problem, the fixpoint characterisation is approximated by a discretisation technique. The time horizon is divided into equally-sized subintervals with length δ, where δ is assumed to be small enough such that at most one Markovian transition fires with a high probability. Under this assumption we can transform the IMC into its induced interactive probabilistic chain [19], the discrete version of IMCs.

Definition 10 (Interactive Probabilistic Chain). *An* interactive probabilistic chain *(IPC) is a tuple* $\mathcal{D} = (S, Act, \longrightarrow, \dashrightarrow_d, s_0)$*, where* S*,* Act*,* $\longrightarrow$ *and* s_0 *are as in Definition 3 and* $\dashrightarrow_d \subseteq S \times Distr(S)$ *is the set of probabilistic transitions.*

A probabilistic transition specifies the probability with which a state evolves to its successors after one time step. The notion of probabilistic transitions resembles the one-step transition matrix in DTMCs. The concepts of closed and open models can be transferred to IPCs. Additionally, since we do not consider continuous time, paths in an IPC can be seen as time-abstract paths in an IMC, implicitly still counting discretisation steps, and thus discrete time. The most general scheduler classes for IPCs are time-abstract history-dependent randomised (TAHR) schedulers.

Discretisation from IMC to IPC. Below we describe the discretisation technique that transforms an IMC into an IPC. Afterwards, we explain how reachability computation in an IMC can be approximated by an analysis on the corresponding IPC with a proven error bound.

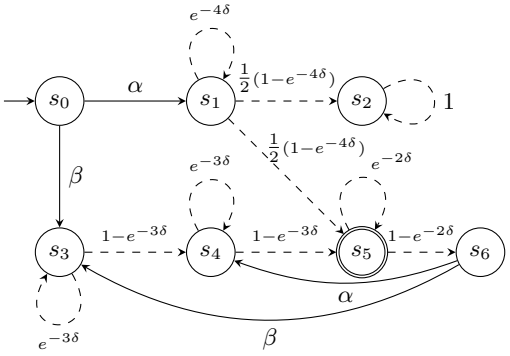
Definition 11 (Discretisation [50]). *Given an IMC* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *and a discretisation constant* δ*,* $\mathcal{I}_\delta = (S, Act, \longrightarrow, \dashrightarrow_\delta, s_0)$ *is the induced IPC*

from $\mathcal{I}$ *with respect to discretisation constant* δ*, where* $\dashrightarrow_\delta = \{(s, \mu^s) \mid s \in MS\}$ *and*

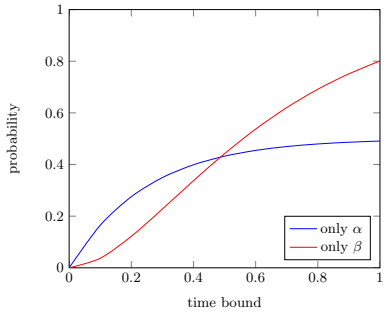
$$\mu^s(s') = \begin{cases} (1 - e^{-E(s)\delta})\mathbf{P}(s, s') & s' \neq s \\ (1 - e^{-E(s)\delta})\mathbf{P}(s, s') + e^{-E(s)\delta} & s' = s \end{cases}$$

This discretisation approximates the original model by assuming that at most one Markovian transition fires in each time-interval of length δ. Accordingly, μ^s specifies the probability that either one or no Markovian transition occurs from state s within each discretisation step. Using the fixpoint characterisation above, it is now possible to relate the probabilities of a reachability analysis in an IMC $\mathcal{I}$ to reachability probabilities in its IPC $\mathcal{I}_\delta$.

Example 8. Consider the IMC in Figure 1 and assume that all actions are internal. Given discretisation constant $\delta > 0$, Figure 4a shows the induced IPC of the original model w.r.t. δ. ■



(a) The induced IPC of the original model. δ is an arbitrary positive discretisation constant.



(b) Approximate maximum time-bounded reachability computed by discretisation.

Figure 4: Time-bounded reachability for the IMC depicted in Figure 1.

Theorem 2 (Discretisation error [50]). *Let* $\mathcal{I} = (S, Act, \rightarrow, \dashrightarrow, s_0)$ *be an IMC,* $G \subseteq S$ *and an interval* I *with rational bounds such that* $a = \inf I, b = \sup I$ *with* $0 \leq a < b$ *and* $\lambda = \max_{s \in MS} E(s)$*. Let* $\delta > 0$ *be such that* $a = k_a\delta, b = k_b\delta$ *for some* $k_a, k_b \in \mathbb{N}$*. Then, for all* $s \in S$ *it holds that*

$$p^{\mathcal{I}_\delta}_{\max}(s, \Diamond^{(k_a,k_b]}G) - k_a\frac{(\lambda\delta)^2}{2} \leq p^{\mathcal{I}}_{\max}(s, \Diamond^I G) \leq p^{\mathcal{I}_\delta}_{\max}(s, \Diamond^{(k_a,k_b]}G) + k_b\frac{(\lambda\delta)^2}{2} + \lambda\delta.$$

Theorem 2 states that the time-bounded reachability property in an IMC $\mathcal{I}$ can be arbitrarily closely approximated by evaluating the same property in the

induced IPC $\mathcal{I}_\delta$. The error bound decreases linearly with smaller discretisation steps δ. It has been recently improved in [33].

The remaining problem is to compute the maximum (or minimum) probability to reach G in an IPC within step bound $k \in \mathbb{N}$. Let $\Diamond^{[0,k]}\, G$ be the set of infinite paths in an IPC that reach a state in G within k steps, and let $p^{\mathcal{D}}_{\max}(s, \Diamond^{[0,k]}\, G)$ denote the maximum probability of those paths that start from state s and are subject to scheduler $\mathcal{D}$. Then, we have $p^{\mathcal{D}}_{\max}(s, \Diamond^{[0,k]}\, G) = \sup_{D \in TA} \Pr_{s,D}(\Diamond^{[0,k]}\, G)$. This expression can be solved by using an adaptation of the well-known value iteration scheme for MDPs to IPCs [59].

The algorithm unfolds the IPC backwards in an iterative manner, starting from the goal states. Each iteration intertwines the analysis of Markovian states and the analysis of interactive states. The main idea is that a path from interactive states to G is split into two parts:(1) reaching Markovian states from interactive states in zero time and (2) reaching goal states from Markovian states in interval $[0, j]$, where j is the step count of the iteration. The computation of the former can be reduced to an unbounded reachability analysis in the MDP induced by interactive states and rewards on Markovian states. For the latter, the algorithm operates on the previously computed reachability probabilities from all Markovian states up to step count j. We can generalise this recipe to step interval-bounded reachability [59].

Example 9. We want to compute the maximum reachability probability from the initial state s_0 to state s_5 of the IMC shown in Figure 1. Consider the induced IPC shown in Figure 4a which discretises the IMC. The maximum step-bounded reachability of the IPC is illustrated in Figure 4b. The optimal decision in state s_0 depends on the time bound. When the time bound is small the optimal action in state s_0 is α, whereas for larger time bounds taking action β yields the maximum reachability. The discretisation constant $\delta = 1.27e-7$ is chosen on the basis of Theorem 2 to guarantee that the error bound is at most 1e-6. Hence, the computation is completed after 8e+6 iterations. ■

3.4 Time-Bounded Reachability in Open IMCs

IMCs feature compositional behaviour which allows them to communicate with their environment. As discussed in Section 2, the class of IMCs which can interact with other IMCs, in particular via parallel composition, is called *open*. Lately, model checking of open IMCs has been studied, where the IMC is considered to be placed in an unknown environment that may delay or influence its behaviour via synchronisation [15]. The approach is restricted to a subclass of IMCs that are non-Zeno and do not contain states that have both internal and external actions enabled at the same time. Let IMC $\mathcal{I}$ satisfy these restrictions and be subject to an environment E, which can be seen as the composition of several other IMCs and has the same external actions as $\mathcal{I}$. IMC $\mathcal{I}$ is then turned into a two-player *controller-environment game*, in which the controller controls $\mathcal{I}$ and the environment controls E. In each state of $\mathcal{I}$ the controller selects one of the enabled internal transitions, if there are some. Otherwise, the environment

either chooses an external action and synchronises $\mathcal{I}$ and E, or it chooses an internal action. Given a set of goal states G and time bound b, the controller tries to maximise the probability to reach the target set G within b time units. The environment tries to prevent the controller from reaching its goal by either delaying synchronisation steps or forcing the controller to take non-optimal paths. In this setup, the time-bounded reachability can be computed by the approximation scheme laid out in [59], which we have discussed above.

3.5 Expected Time

This section presents an algorithm to obtain the minimum and maximum expected time to reach a given set of goal states in an IMC, introduced in [27]: We describe the expected time objective with a fixpoint characterisation, and its transformation into a *stochastic shortest path* (SSP) problem. This SSP problem can then be used to solve the expected time CSL formula. Note that we only consider well-defined IMCs without Zeno paths.

Expected time objective. Let's assume that we already computed $Sat(\Phi)$, and denote this set as our set of goal states G. We want to compute the minimum expected time to reach a state in G from a given state $s \in S$. Thus, we have to consider all possible paths π induced by a given scheduler D. We define the random variable $V_G : Paths \to \mathbb{R}_{\geq 0}$ as the elapsed time before visiting a state in G . For an infinite path $\pi = s_0 \xrightarrow{\sigma_0,t_0} s_1 \xrightarrow{\sigma_1,t_1} \dots$ let $V_G(\pi) = \min\{t \in \mathbb{R}_{\geq 0} | G \cap \pi@t \neq \varnothing\}$ with $\min(\varnothing) = \infty$ [27]. Then the minimal expected time to reach G from $s \in S$ is given by:

$$\mathsf{eT}^{\min}(s, \Diamond G) \;=\; \inf_{D} \mathbb{E}_{s,D}(V_G) \;=\; \inf_{D} \int_{Paths} V_G(\pi)\, \Pr_{s,D}(\,\mathrm{d}\pi). \tag{1}$$

Formula (1) expresses that we have to find a scheduler D which minimises the time until reaching a state in G . We therefore need to consider all paths induced by scheduler D. Note that, by definition of V_G, it is sufficient to consider the time before entering a goal state. Hence, we can transform all goal states into absorbing Markovian states without affecting the expected time reachability. This may result in a much smaller state space, since we can neglect those states that become unreachable from the initial state.

Theorem 3 ([27]). *The function* $\mathsf{eT}^{\min}$ *is a fixpoint of the Bellman operator*

$$v(s) = \begin{cases} \dfrac{1}{E(s)} + \displaystyle\sum_{s' \in S} \mathbf{P}(s, s') \cdot v(s') & \text{if } s \in MS \setminus G \\ \displaystyle\min_{s \xrightarrow{\alpha} s'} v(s') & \text{if } s \in IS \setminus G \\ 0 & \text{if } s \in G. \end{cases}$$

Theorem 3 encodes expression (1) in a Bellman equation, in which we aim to find optimal values $v(s)$ for all states $s \in S$. If we are already in a goal state, we

have by definition that $V_G(\pi) = 0$ with $\pi = s_0 \xrightarrow{\sigma_0, t_0} \ldots$ and $s_0 \in G$. If $s \in IS$ and s has only one outgoing interactive transition, then the expected time is the same as the one of its successor state. In case there is a nondeterministic choice between interactive transitions in s, the next transition is determined by the scheduler. Since we look for the infimum over all schedulers D, we choose the action which induces the lowest expected time in the successor state. If $s \in MS$, we add the sojourn time in state s to the time to reach a state in G over all paths starting in s induced by scheduler D. In other words, we add the sojourn time in state s to the expected sojourn time of each successor state s' weighted with the probability to reach s'.

As a result of Theorem 3, the nondeterminism in $\mathsf{eT}^{\min}(s, \Diamond G)$ can be resolved by using a stationary deterministic scheduler [27]. This implies that the scheduler chooses an action that results in the minimum expected time for each interactive state with a nondeterministic choice. To yield an effective algorithm as well as to show the correctness of Theorem 3, we transform the expected time computation into a non-negative stochastic shortest path (SSP) problem for MDPs. A SSP problem derives the minimum expected cost to reach a set of goal states in a MDP.

Definition 12 (SSP Problem). *A non-negative* stochastic shortest path problem *(SSP problem) is a tuple* $\mathsf{ssp} = (S, Act, \mathbf{P}, s_0, G, c, g)$*, where* $(S, Act, \mathbf{P}, s_0)$ *is an MDP,* $G \subseteq S$ *is a set of* goal *states,* $c : S \setminus G \times Act \to \mathbb{R}_{\geq 0}$ *is a* cost function *and* $g : G \to \mathbb{R}_{\geq 0}$ *is a* terminal cost function*.*

Given a smallest index $k \in \mathbb{N}$ of a path π with $\pi[k] = s_k \in G$, the accumulated costs to reach G on π is given by $\sum_{i=0}^{k-1} c(s_i) + g(s_k)$. The transformation of an IMC into an SSP problem is realized with the following definition.

Definition 13 (SSP for Minimum Expected Time Reachability). *The SSP of IMC* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *for the expected time reachability of* $G \subseteq S$ *is* $\mathsf{ssp}_{\mathsf{eT}^{\min}}(\mathcal{I}) = (S, Act \cup \{\bot\}, \mathbf{P}, s_0, G, c, g)$ *where* $g(s) = 0$ *for all* $s \in G$ *and for all* $s, s' \in S$ *and* $\sigma \in Act \cup \{\bot\}$*:*

$$\mathbf{P}(s, \sigma, s') = \begin{cases} \frac{\mathbf{R}(s,s')}{E(s)} & \text{if } s \in MS \wedge \sigma = \bot \\ 1 & \text{if } s \in IS \wedge s \xrightarrow{\sigma} s' \\ 0 & \text{otherwise, and} \end{cases}$$

$$c(s, \sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \setminus G \wedge \sigma = \bot \\ 0 & \text{otherwise.} \end{cases}$$

The Markovian states are equipped with costs, since they are the states in which time advances. The cost to traverse a Markovian state along path π is determined by the sojourn time. Observe that the Bellman equation in Theorem 3 coincides with the definition of the SSP problem. The uniqueness of the minimum expected cost of an SSP problem [3, 8] implies that $\mathsf{eT}^{\min}(\mathsf{s}, \Diamond \mathsf{G})$ is the unique fixpoint of $v(s)$ [27].

Example 10. Consider IMC $\mathcal{I}$ depicted in Figure 1 with $G = \{s_5\}$ being the set of goal states and s_0 being the initial state. We want to obtain $\mathsf{eT}^{\min}(\mathsf{s_0}, \Diamond \mathsf{G})$.

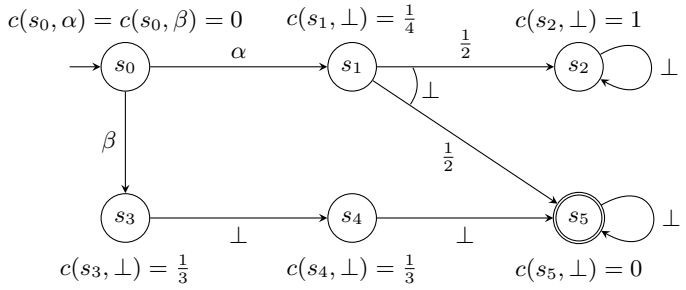


Figure 5: Resulting $\mathsf{ssp}_{\mathsf{eT}^{\min}}$ of the IMC depicted in Figure 1.

In a first step, we make goal state s_5 absorbing. Afterwards, we transform the resulting IMC into the SSP problem depicted in Figure 5. From this SSP problem we can derive the following LP problem, where x_i represents values for s_i:

Maximise $x_0 + x_1 + x_3 + x_4$ subject to:

$$x_0 \leq x_1 \qquad x_1 \leq \frac{1}{4} + \frac{1}{2}x_2 + \frac{1}{2}x_5 \qquad x_3 \leq \frac{1}{3} + x_4 \qquad x_5 = 0$$

$$x_0 \leq x_2 \qquad x_2 \leq 1 + x_2 \qquad x_4 \leq \frac{1}{3} + x_5$$

By solving these equations we obtain $x_0 = \frac{2}{3}, x_1 = \infty, x_2 = \infty, x_3 = \frac{2}{3}, x_4 = \frac{1}{3}$, which yields $\mathsf{eT}^{\min}(s_0, \Diamond G) = \frac{2}{3}$. ■

An analogous approach can be applied to obtain the maximum expected time. In this case, we search for the supremum over all schedulers, and thus, we resolve nondeterministic choices in such a way that the scheduler chooses the actions that maximises the expected time.

3.6 Long-Run Average

In this section we present an algorithm to compute the long-run average (Lra) time spent in a set of goal states, as introduced in [27]. We describe the long-run average objective and a three step procedure to obtain the long-run average and, thus, compute the Lra CSL formula. Again, we only consider well-defined IMCs without Zeno paths.

Long-run average objective. We assume that $Sat(\Phi)$ has already been computed with the technique explained before, and we denote this set as our set of goal states G. Random variable $A_{G,t}(\pi) = \frac{1}{t}\int_0^t \mathbf{1}_G(\pi@u)du$ defines the fraction of time that is spent in G on an infinite path π in $\mathcal{I}$ up to time bound $t \in \mathbb{R}_{\geq 0}$ [2]. Note that $\mathbf{1}_G(s) = 1$ if and only if $s \in G$ and otherwise 0. For the computation of the long-run average we consider the limit $t \to \infty$ for random variable $A_{G,t}$, denoted by A_G. The expectation of A_G under scheduler D and initial state s

then yields the long-run average time spent in G, where the minimum long-run average time spent in G starting from s is defined by:

$$\textsc{Lra}^{\min}(s, G) = \inf_{D} \textsc{Lra}^{D}(s, G) = \inf_{D} \mathbb{E}_{s,D}(A_G). \tag{2}$$

In contrast to the computation of the expected time and time-bounded reachability, we may assume w.l.o.g. that $G \subseteq MS$, since the long-run average time spent in any interactive state is always 0 (see Section 2). In the remainder of this section we give the basic intuition of how to compute the minimum long-run average. The general idea is given by the following three-step procedure:

1. Determine the maximal end components $\{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ of IMC $\mathcal{I}$.
2. Determine $\textsc{Lra}^{\min}(G)$ for each maximal end component $\mathcal{I}_j$.
3. Reduce the computation of $\textsc{Lra}^{\min}(s_0, G)$ in IMC $\mathcal{I}$ to an SSP problem.

The first step can be performed by a graph-based algorithm [1, 17], whereas the latter two can be expressed as LP problems.

Definition 14 (End Component). *An* end component *of IMC $\mathcal{I}$ is a sub-IMC defined by the tuple (S', A) where $S' \subseteq S$ and $A \subseteq Act$ such that:*

- *for all Markovian states $s \in S'$ with $s \xrightarrow{\lambda} s'$ it follows that $s' \in S'$, and*
- *for all interactive states $s \in S'$ and for all $\alpha \in A$ with $s \xrightarrow{\alpha} s'$ it follows that $s' \in S'$, where at least one action is enabled in $s \in S'$.*

Further, the underlying graph of (S', A) must be a strongly connected component.

Note that a *maximal end component* (MEC) is an end component which is not contained in any larger end component.

Long-run average in MECs. For the second step we show that for *unichain* IMCs the computation of $\textsc{Lra}^{\min}(s, G)$ can be reduced to the determination of long-run ratio objectives in MDPs. An IMC is unichain if and only if under any stationary deterministic scheduler it yields a strongly connected graph structure. Note that an MEC is a unichain IMC. At first, we define long-run ratio objectives for MDPs, and then show how to transform them to Lra objectives in unichain IMCs.

Let $\mathcal{M} = (S, Act, \mathbf{P}, s_0)$ be an MDP and $c_1, c_2 : S \times Act_{\perp} \to \mathbb{R}_{\geq 0}$ be cost functions. The operational interpretation is that cost $c_1(s, \alpha)$ is incurred when α is selected in state s, similarly for c_2. The long-run ratio between the accumulated costs c_1 and c_2 along an infinite path π in MDP $\mathcal{M}$ is defined as:

$$\mathcal{R}(\pi) \;=\; \lim_{n \to \infty} \frac{\sum_{i=0}^{n-1} c_1(s_i, \alpha_i)}{\sum_{j=0}^{n-1} c_2(s_j, \alpha_j)}.$$

Example 11. Consider the infinite path $\pi = (s_0 \xrightarrow{2} s_1 \xrightarrow{3} s_2 \xrightarrow{1} s_3 \xrightarrow{4} s_0)^{\omega}$ where $c_2(s_i, \cdot)$ denotes the transition labels and $c_1(s_0, \cdot) = 2$ and $c_1(s_i, \cdot) = 0$ for $1 \leq i \leq 3$. Table 3 depicts the computation of the long-run ratio until $n = 6$. By setting the limit $n \to \infty$ we obtain a fixpoint with $\mathcal{R}(\pi) = \frac{1}{5}$. ■

Table 3: Example computation for the long-run ratio.

n	1	2	3	4	5	6
$\mathcal{R}(\pi)$	$\frac{2}{2} = 1$	$\frac{2}{2+3} = \frac{2}{5}$	$\frac{2}{2+3+1} = \frac{1}{3}$	$\frac{2}{2+3+1+4} = \frac{1}{5}$	$\frac{2+2}{2+3+1+4+2} = \frac{1}{3}$	$\frac{2+2}{2+3+1+4+2+3} = \frac{4}{15}$

The minimum long-run ratio objective for state s of MDP $\mathcal{M}$ is then defined by:

$$R^{\min}(s) \;=\; \inf_D \mathbb{E}_{s,D}(\mathcal{R}) \;=\; \inf_D \sum_{\pi \in Paths_{\mathsf{abs}}} \mathcal{R}(\pi) \cdot \Pr^{\mathsf{abs}}_{s,D}(\pi).$$

$Paths_{\mathsf{abs}}$ denotes the time-abstract paths of the MDP $\mathcal{M}$ and $\Pr^{\mathsf{abs}}_{s,D}$ represents the probability measure on the sets of paths starting from s induced by scheduler D in $\mathcal{M}$. $R^{\min}(s)$ can be obtained by solving a linear programming problem [1]. With real variable k representing $R^{\min}$ and x_s representing each $s \in S$ we have:

Maximise k subject to:

$$x_s \;\leq\; c_1(s,\alpha) - k \cdot c_2(s,\alpha) + \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot x_{s'} \quad \text{for each } s \in S, \alpha \in Act.$$

This system of inequations can be solved by linear programming algorithms, e.g. with the simplex method [54].

Example 12. We take path π from Example 11 and assume that it is the only path in an MDP $\mathcal{M}$. Deriving the system of linear inequations with variables k, x_{s_i} for $0 \leq i \leq 3$ then yields:

Maximise k subject to:

$$\begin{aligned} x_{s_0} &\leq 2 - 2 \cdot k + x_{s_1} & \qquad x_{s_2} &\leq -1 \cdot k + x_{s_3} \\ x_{s_1} &\leq -3 \cdot k + x_{s_2} & \qquad x_{s_3} &\leq -4 \cdot k + x_{s_0} \end{aligned}$$

By solving the inequation system we obtain $k = \frac{1}{5}$, which is the minimum long-run ratio on $\mathcal{M}$. Note that this value equals the product of the long-run ratio as obtained in Example 11 and the probability that path π is chosen, which is 1 in our case. ■

This result can now be transferred to a unichain IMC by transforming it into an MDP with two cost functions.

Definition 15. *Let $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ be an IMC and $G \subseteq S$ a set of goal states. We define the MDP $\mathsf{mdp}(\mathcal{I}) = (S, Act_\perp, \mathbf{P}, s_0)$ with cost functions c_1 and c_2, where $\mathbf{P}$ is defined as in Definition 13 and*

$$c_1(s,\sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \cap G \wedge \sigma = \perp \\ 0 & \text{otherwise,} \end{cases} \qquad c_2(s,\sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \wedge \sigma = \perp \\ 0 & \text{otherwise.} \end{cases}$$

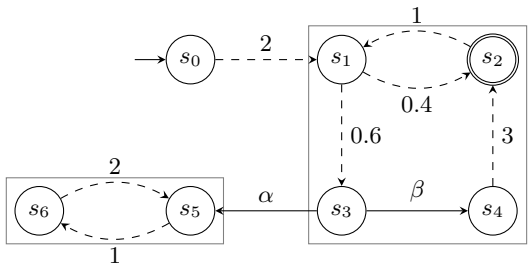


Figure 6: IMC with two maximal end components.

Observe that cost function c_2 keeps track of the average sojourn time in all states $s \in S$ whereas c_1 only does so for states $s \in G$.

For a unichain IMC $\mathcal{I}$, $LRA^{\min}(s, G)$ equals the long-run ratio $R^{\min}(s)$ in the transformed MDP $\mathsf{mdp}(\mathcal{I})$ [27]. Further, in a unichain IMC we have that $\textsc{Lra}^{\min}(s, G)$ and $\textsc{Lra}^{\min}(s', G)$ are the same for any two states s and s'. Therefore, we will omit the state and write $\textsc{Lra}^{\min}(G)$ when considering unichain IMCs.

Reducing Lra *objectives to an SSP problem.* Let $\mathcal{I}$ be an IMC with initial state s_0 and maximal end components $\{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ for $k > 0$, where $\mathcal{I}_j$ has state space S_j. Using this decomposition of $\mathcal{I}$ into maximal end components, we obtain the following result:

Theorem 4 ([27]). [4] *For IMC* $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ *with MECs* $\{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ *with state spaces* $S_1, \ldots, S_k \subseteq S$*, and set of goal states* $G \subseteq S$*:*

$$\textsc{Lra}^{\min}(s_0, G) = \inf_D \sum_{j=1}^{k} \textsc{Lra}_j^{\min}(G) \cdot \Pr^D(s_0 \models \Diamond\Box S_j),$$

where $\Pr^D(s_0 \models \Diamond\Box S_j)$ *is the probability to eventually reach and continuously stay in* S_j *from* s_0 *under policy* D *and* $\textsc{Lra}_j^{\min}(G)$ *is the* Lra *of* $G \cap S_j$ *in unichain MA* $\mathcal{I}_j$*.*

Intuitively we have to find a scheduler that minimises the product of the probability to eventually reach and stay in a MEC and the minimum Lra, over all possible combinations of MECs. We illustrate this procedure more clearly in the following example.

Example 13. Consider the IMC in Figure 6 with $G = \{s_2\}$. It consists of the two maximal end components MEC_1 with $S_1 = \{s_1, s_2, s_3, s_4\}$ and $Act(s_3) = \{\beta\}$, and MEC_2 with $S_2 = \{s_5, s_6\}$. Note that only MEC_1 contains a goal state. Hence, the long-run average for MEC_2 is automatically 0, wheras for MEC_1 it is greater than 0. Since we are looking for the minimum long-run average of s_2

[4] This theorem corrects a small flaw in the theorem for IMCs in [27].

and are starting in s_0, we choose action α in s_3 so that we end up in MEC_2. According to Theorem 4 we have to look for the scheduler that minimises the LRA in such a way that we eventually always stay in the desired MEC. With the choice of α we can neglect the LRA for MEC_1, since we will almost surely leave MEC_1, and thus obtain $\mathrm{LRA}^{\min}(s_0, G) = 0$. ■

The computation of the minimum LRA for IMCs is now reducible to a non-negative SSP problem. In IMC $\mathcal{I}$ we replace each maximal end component $\mathcal{I}_j$ with two fresh states q_j and u_j. Intuitively, q_j represents the MEC $\mathcal{I}_j$ and u_j represents a decision state that has a transition to q_j and contains all outgoing interactive transitions of S_j. Let U denote the set of u_j states and Q the set of q_j states. For simplification, we assume w.l.o.g. that all actions are unique and replace actions of a state $s_i \in S$ by $\tau_{i,j}$ where $j \in \{1 \dots n_i\}$ with $n_i \in \mathbb{N}$ defined as the number of nondeterministic choices in state s_i.

Definition 16 (SSP for Long-Run Average). *Let $\mathcal{I}$, S, $G \subseteq S$, $\mathcal{I}_j$ and S_j be as before. The SSP induced by $\mathcal{I}$ for the long-run average fraction of time spent in G is the tuple* $\mathsf{ssp}_{\mathsf{LRA}^{\min}}(\mathcal{I}) = \left(S \setminus \bigcup_{i=1}^{k} S_i \cup U \cup Q, Act \cup \{\bot\}, \mathbf{P}', s_0, U, c, g\right)$, *where $g(q_j) = \mathrm{LRA}_j^{\min}(G)$ for $q_j \in Q$ and $c(s, \sigma) = 0$ for all s and $\sigma \in Act_\bot$. $\mathbf{P}'$ is defined as follows: Let $S' = S \setminus \bigcup_{i=1}^{k} S_i$. $\mathbf{P}'$ equals $\mathbf{P}$ for all $s, s' \in S'$ and for the new states in U:*

$$\mathbf{P}'(u_i, \tau_{k,l}, s') = \mathbf{P}(s_k, \tau_{k,l}, s') \quad \text{if } s' \in S' \wedge s_k \in S_i \wedge l \in \{1 \dots n_k\} \quad \text{and}$$
$$\mathbf{P}'(u_i, \tau_{k,l}, u_j) = \mathbf{P}(s_k, \tau_{k,l}, s') \quad \text{if } s_k \in S_i \wedge s' \in S_j \wedge l \in \{1 \dots n_k\}$$

Finally, we have: $\mathbf{P}'(q_i, \bot, q_j) = 1 = \mathbf{P}'(u_i, \bot, q_i)$ and $\mathbf{P}'(s, \sigma, u_i) = P(s, \sigma, S_i)$.

Here, $P(s, \sigma, S_i)$ is a shorthand for $\sum_{s' \in S'} \mathbf{P}(s, \sigma, s')$. An example of the SSP transformation of IMC $\mathcal{I}$ from Figure 1 is given in Figure 7.

Example 14. Consider the IMC in Figure 1 with two maximal end components MEC_1 with $S_1 = \{s_2\}$ and MEC_2 with $S_2 = \{s_3, s_4, s_5, s_6\}$. For each MEC we introduce new states u_i and q_i, which substitute the states of MEC_i. Further, we substitute α with $\tau_{1,1}$ and β with $\tau_{1,2}$. Note that both MECs are bottom strongly connected components, which means that, under all schedulers of IMC $\mathcal{I}$, we cannot leave the MEC after entering it. Therefore, decision state u_i has only one outgoing transition to the corresponding q_i state. After the transformation to the IMC in Figure 6, the decision state of MEC_1 has a nondeterministic choice between β, to stay in MEC_1, and α, to leave it. ■

Note that an analogous approach can be applied to obtain the maximum LRA. The main difference is that, in this case, we look for the supremum over all schedulers. In the second and the third step we now resolve the nondeterminsitic choices according to maximise the LRA.

4 Abstraction

In the previous chapter we introduced a number of IMC properties and presented algorithms for their computation. For each presented algorithm the runtime is

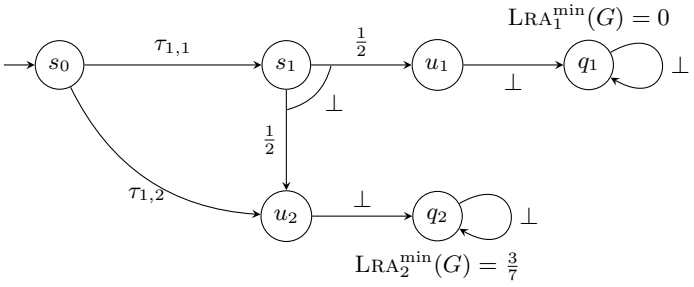


Figure 7: Resulting SSP for $\text{LRA}^{\min}$ of the IMC depicted in Figure 1.

crucially depends on the size of the considered IMC. On average, the complexity of most algorithms grows polynomially in the size of the state space, but in the worst case it grows exponentially resulting in extremely long computation times for complex models. Abstraction provides the means to reduce the state space of investigated IMCs and thereby to reduce the complexity of the verification of certain properties. In this section, we will first define the most important behavioural equivalences, namely strong and weak bisimulation, and outline efficient algorithms to compute bisimulation quotients. We remark that bisimulation can be decided in polynomial time but most coarser behavioural equivalences like trace and testing equivalences are PSPACE-complete [41].

4.1 Behavioural Equivalences

Behavioural equivalences relate states which are indistinguishable for an external observer of the system. In the following we will present the concepts of strong and weak bisimulation. As for non-probabilistic systems, these behavioural equivalences relate states that can mimic each other's behaviour. Weak bisimulation relaxes strong bisimulation by allowing that interactive transitions with visible actions may be interleaved with transitions annotated with the internal action τ. In the context of model checking, the most important application of behavioural equivalences is to provide the means for 'quotienting' a system with respect to the behavioural equivalence to reduce its state space.

Definition 17 (Strong Bisimulation). *Let $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ be an IMC. An equivalence relation $R \subseteq S \times S$ is a* strong bisimulation *on $\mathcal{I}$, iff for all $(s,t) \in R$, $a \in Act$ and $C \in S/R$ we have that:*

- *$s \xrightarrow{a} s'$ for some $s' \in C$ iff $t \xrightarrow{a} t'$ for some $t' \in C$*
- *$\mathbf{R}(s, C) = \mathbf{R}(t, C)$ whenever $s \not\xrightarrow{\tau}$.*

Here, $\mathbf{R}(s, C)$ is a shorthand for $\sum_{s' \in C} \mathbf{R}(s, s')$, as defined in Section 2. The first condition expresses the classical bisimulation condition, requiring that for related states $s \, R \, t$ every interactive transition $s \xrightarrow{a} s'$ can be mimicked by an

interactive transition $t \xrightarrow{a} t'$ such that the target states are again related, i.e. $s' \, R \, t'$. The second condition expresses that related states $s \, R \, t$ need to agree on the cumulative rates of moving from s, respectively t, to any equivalence class C; it thereby corresponds to conditions for lumpability of Markov chains and probabilistic bisimulation of DTMCs [44]. This condition is only required for states which can perform Markovian transitions. Due to the maximal progress assumption, these can only be states that have no internal action τ enabled, denoted by $\not\xrightarrow{\tau}$. Bisimulation relations on IMCs are closed under union which allows to define the largest bisimulation $\sim$ by the union on all bisimulations of the considered IMC. As shown in [35, Theorem. 4.3.1] both parallel composition and hiding are defined with respect to $\sim$. Moreover, time-bounded and unbounded reachability properties are preserved by bisimilar states [51, Theorem 4]. This allows us to reason over IMCs in a compositional manner.

Strong bisimulation is rigid in the sense that it requires the mimicking of interactive transitions for visible and internal actions τ. To achieve a higher degree of abstraction, we relate states that cannot be distinguished by an external observer by considering the visible actions only. For interactive transitions we apply the same machinery as for LTS. We denote by $\xrightarrow{\tau^*}$ the transitive reflexive closure of interactive transitions labelled with the internal action τ. Weak interactive transitions are then given by $\stackrel{a}{\Longrightarrow} = \xrightarrow{\tau^*} \circ \xrightarrow{a} \circ \xrightarrow{\tau^*}$. On the other hand, this does not work for Markovian transitions, since sequencing of Markovian transitions leads to the formation of the more general phase-type distributions. Thus, Markovian transitions need to be mimicked in the same way as in strong bisimulation.

Definition 18 (Weak Bisimulation). *Let $\mathcal{I} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ be an IMC. An equivalence relation $R \subseteq S \times S$ is a* weak bisimulation *on $\mathcal{I}$, iff for all $(s,t) \in R$, $a \in Act$ and $C \in S/R$ we have that:*

- *$s \stackrel{a}{\Longrightarrow} s'$ for some $s' \in C$ if and only if $t \stackrel{a}{\Longrightarrow} t'$ for some $t' \in C$*
- *$\mathbf{R}(s', C) = \mathbf{R}(t', C)$ for some $t \xrightarrow{\tau^*} t'$ and $t' \not\xrightarrow{\tau}$ whenever $s \xrightarrow{\tau^*} s'$ and $s' \not\xrightarrow{\tau}$.*

Weak bisimulation is closed under union and we denote the largest weak bisimulation by $\approx$. Similarly to strong bisimulation, parallel composition and hiding are compatible with $\approx$ [35, Theorem. 4.4.1]. Moreover, weak bisimilarity preserves maximal time-bounded reachability properties [36, Theorem 10]. Strong and weak bisimulation are suitable to compare systems and to reduce their state space by deriving strong bisimilar (resp. weak bisimilar) IMCs with smaller state spaces constructed from the equivalence classes of the strong bisimilarity (resp. weak bisimilarity) and with the interactive and Markovian transitions defined in the natural way [36, Definition 10]. We remark that, in order to reason over the refinement of IMCs, there are appropriate notions of strong and weak simulations available, for which parallel composition and hiding are precongruences [36].

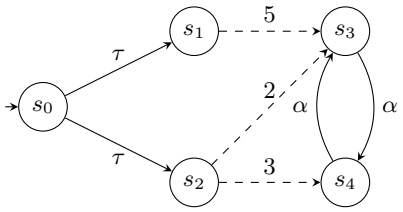


Figure 8: An interactive Markov chain with 5 states.

Example 15. Consider the IMC in Figure 8. We have $s_3 \approx s_4$ and $s_3 \sim s_4$ since these states can mimic each other's behaviour. It follows that $s_1 \approx s_2$ and $s_1 \sim s_2$; the accumulated transition rate into the bisimulation class $C_0 = \{s_3, s_4\}$ is the same for both states. Because s_0 reaches s_1 and s_2 via internal τ transitions we have $s_0 \approx s_1 \approx s_2$, but s_0 is not strongly bisimilar to s_1 and s_2. ■

4.2 Algorithmic Computation of the Strong Bisimulation Quotient

Given an IMC $\mathcal{I}$, we want to determine its counterpart $\mathcal{I}$' in which strongly bisimilar states are collapsed into one state so that $\mathcal{I} \sim \mathcal{I}$'. The result $\mathcal{I}$' of this collapsing process is called the bisimulation quotient. In the following, we present an algorithm based on partition refinement techniques [35]. The core idea of the algorithm is to partition the states in S, and refine the resulting partition step by step. The refinement procedure of one particular partition consists of two stages that validate the two conditions of Definition 14 with respect to a so-called *splitter*. A splitter is a tuple formed by a set of states C and an action a. Given a partition P of S, in each set of P we group all those states together that can reach C via an a-transition, and those that cannot. This process is illustrated in Figure 9. More formally, given a splitter (C, a) we refine the partition according to the first condition in Definition 14 by applying

$$\mathit{Refine}(P, a, C) := \left(\bigcup_{X \in P} \left(\bigcup_{\nu \in \{true, false\}} \Big\{ \{ s \in X \mid \gamma(s, a, C) = \nu \} \Big\} \right) \right) \backslash \{\emptyset\}.$$

The function $\gamma : S \times \mathrm{Act} \times S^* \to \{true, false\}$ applied to (s, a, C) returns true if there is an a-transition from s to at least one state in C. Thus, the function *Refine* splits each set of the partition by grouping those states together that can reach states in C by at least one interactive a-transition and those that cannot. Similarly, the second condition for strongly bisimilar states is validated by refining the partition with

$$\mathit{M\_Refine}(P, C) := \left(\bigcup_{X \in P} \left(\bigcup_{\nu \in \mathbb{R}^+} \Big\{ \{ s \in X | R(s, C) = \nu \} \Big\} \right) \right) \backslash \{\emptyset\}.$$

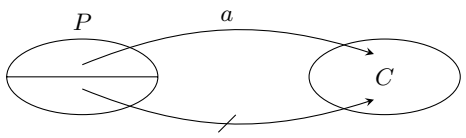


Figure 9: One refinement step.

In other words, *M_Refine* splits each set of the partition P into classes so that all states in one class reach the set C with the same accumulated rate. Note that the result might be a set of singletons in case that all values of $R(s, C)$ are different. These two functions highlight the naming of the tuple (a, C) as splitter: It splits sets of states according to the two conditions of strongly bisimilar states and thereby refines the initial partition step by step.

Equipped with these two functions we construct the algorithm. The algorithm starts with a splitter (a, C) build from an arbitrary action $a \in Act$ an the set of states C either comprising all states which can perform a τ step or all states which cannot perform τ. The original partition consists of one set which contains all states. We then apply *Refine* and *M_Refine* iteratively by choosing a different splitter in each step. After each iteration we possibly obtain a finer partition and, thus, we need to add newly formed classes to the set of splitters. A final observation speeds this procedure up: States with outgoing internal τ-transition will never take Markovian transitions due to the maximal progress assumption. We therefore do not need to apply *M_Refine* on these states. Furthermore, they cannot be strongly bisimilar to states without outgoing τ-transition. For these reasons we evaluate these two classes of states separately from the very start.

Algorithm 3 (Computation of the Strong Bisimulation Quotient)

STRONG-BISIM-QUOTIEN(S, R)

```
 1  S_Part← {{ P' ∈ S | P' -τ↛ }} \ {∅};
 2  U_Part← {{ P' ∈ S | P' -τ→ }} \ {∅};
 3  Spl← Act× (S_Part ∪ U_Part);
 4  while Spl not empty
 5        do
 6           Choose (a, C) in Spl;
 7           Old:=S_Part ∪ U_Part;
 8           S_Part←Refine(S_Part,a,C);
 9           U_Part← Refine(U_Part,a,C);
10           S_Part← M_Refine(S_Part,C);
11           New←(S_Part ∪ U_Part)-Old;
12           Spl← (Spl-{(a,C)}) ∪ (Act × New);
13  return S_Part ∪ U_Part
```

The algorithm can be implemented with a time complexity of $\mathcal{O}((m_I+m_M)\log n)$, where m_I is the number of interactive transitions, m_M the number of Markovian transitions and n the number of states [35].

4.3 Algorithmic Computation of the Weak Bisimulation Quotient

The computation of the weak bisimulation quotient of an IMC is more involved and we will only briefly outline the ideas formalised in [35], which uses an adaptation of the partition refinement technique described above. In contrast to strong bisimulation, weak bisimulation does not mimic internal τ actions to achieve a higher degree of abstraction. We therefore have to identify those states that have no outgoing interactive transition annotated with a τ action, also called *stable states*, and those that have at least one, called *unstable states*. We then partition the whole state space by grouping those states together that can reach stable states by taking only internal τ transitions, and those that cannot. We call the first class C_1 and the latter C_2 . This step requires the a priori computation of the transitive reflexive closure $\xrightarrow{\tau^*}$ of internal τ actions. The algorithm is then similar to the computation of the strong bisimulation quotient: We initialise the set of splitters and refine the classes C_1 and C_2 separately in a stepwise fashion. The class C_2 is refined with function *Refine* and C_1 with *Refine* and *M_RefineS*, where *Refine* is as before and *M_RefineS* an adaptation of *M_Refine*. The algorithm then computes the weak bisimulation quotient in $\mathcal{O}((m'_I+m_M)n)$ time where m'_I is the number of interactive transitions after transitive closure of internal transitions [35].

4.4 Bisimulation Quotient of Acyclic IMCs

In case that the considered IMC is acyclic, the minimum strong bisimulation can be determined at a much lower time-complexity of $\mathcal{O}(m)$ as suggested in [21], where m is the total number of transitions.

Definition 19 (Acyclic IMC). *An IMC P is acyclic, if it does not contain any plausible path π with $k \in \mathbb{N}_{\geq 0}$ and $\pi[k] = s$ such that $\exists n.k < n \leq |\pi|$ with $\pi[n] = s$ for all $s \in S$. A path is plausible, if it does not contain any Markovian transition such that the maximum progress assumption is violated.*

Since S is finite and P is acyclic, there is at least one state which cannot be left by a plausible path. The idea of the following algorithm is to order the states according to their longest distance to such an absorbing state. To do so we define the notion of ranks for IMCs.

Definition 20 (Rank Function). *The rank function $R\colon S \to \mathbb{N}$ is defined by $R(s) = \max\{|\pi| \mid \pi \in Paths^P(s)\}$.*

Here, $Paths^P(s)$ denotes the set of all plausible paths – which are finite – such that $R(s) < \infty$ for all $s \in S$. The observation which sets the groundwork for the algorithm below is that any two strongly bisimilar states have the same rank. Vice-versa, two states with the same rank are strongly bisimular if and only if they fulfil the two conditions of strongly bisimilar states. Note that transitions go only from states with a higher rank to states with a lower rank. This observation is exploited in the following algorithm. First, check states with rank 1 (states with rank 0 are by default strongly bisimilar) for strong bisimilarity. This computation

requires only states with rank 0. We apply the same procedure iteratively to all states with rank 2, then to all states with rank 3 and so forth. In each iteration states with the same rank are analysed by looking at their transitions to states at the next lower rank. The time-complexity $\mathcal{O}(m)$ is defined by the depth-first search which is required for the rank computation [21]. A similar algorithm can be adopted for the computation of the weak bisimulation quotient.

5 Extensions

The remarkable progress in the theoretic developments of IMCs in the last decade has also triggered research on related concepts. In this section we review inhomogeneous IMCs and Markov automata as specific extensions of IMCs.

5.1 Inhomogeneous IMCs

Inhomogeneous IMCs extend IMCs by allowing the annotations of Markovian transitions with functions rather than real values. So far we assumed that the rates of the Markovian transitions were static and independent of time, i.e. the dynamics of IMCs were assumed to be time-homogeneous. However, in many real-world applications the progress of time crucially influences the system's dynamics. Hardware components tend to degrade over time due to oxidation and deterioration, so that their failure rate is a monotonically increasing function over time rather than a static value. Another example is the power extraction rate of a battery, which depends on the remaining amount of stored energy. Those natural phenomena can be accurately captured with the help of time-inhomogeneous IMCs (I^2MCs) as introduced in [29]. Technically, the Markovian transitions $\xrightarrow{\lambda}$ of IMCs (labelled with a positive real number λ) are generalized to transitions of the form $s \xrightarrow{r_{s,s'}(t)} s'$ of I^2MCs (labelled with a continuous functions $r_{s,s'} \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$). The rate to execute the transition $s \xrightarrow{r_{s,s'}(t)} s'$ from s to s' at time t, is determined by $r_{s,s'}(t)$. The state transition probabilities respecting the respective race condition if multiple transitions leave a single state can be formulated analogously [29].

A process algebra and congruence results for weak and strong bisimulation for I^2MCs can be found in [29]. On the other hand, the model checking of inhomogeneous systems is quite intricate since one needs to account for the time-dependent dynamics and it still needs to be investigated further.

5.2 Markov Automata

Markov automata (MA) constitute a compositional behavioural model for continuous time stochastic and nondeterministic systems [23]. Markov automata are on the one hand rooted in interactive Markov chains by extending the expressiveness of IMCs with instantaneous random switching. They are on the other hand an orthogonal composition of probabilistic automata and continuous time Markov chains. Formally, $\mathcal{M} = (S, Act, \longrightarrow, \dashrightarrow, s_0)$ is a Markov automaton,

where all components of $\mathcal{M}$, but $\longrightarrow$, are as Definition 3 and the set of interactive transitions $\longrightarrow$ is a subset of $S \times Act \times Distr(S)$. The definition of interactive transitions characterises the ability of random switching in Markov automata. Consequently, IMCs are special cases of Markov automata, where all distributions which prevail in the set of interactive transitions $\longrightarrow$ are Dirac. Markov automata are expressive enough to capture the complete semantics of generalised stochastic Petri nets and of stochastic activity networks [22]. Due to these attractive semantic and compositionality features, there is a growing interest in tool and technique support for modelling and analysis with MA [57, 26, 32].

6 Case Studies

IMCs have shown their practical relevance in diverse areas where memoryless continuous delay prevails. They serve as the semantics of several modelling and engineering formalisms such as generalised stochastic Petri nets [50], and Architectural Analysis and Design Language (AADL) [13]. Furthermore, they have proven their practical importance in various applications like Globally Asynchronous Locally Synchronous (GALS) designs [19], supervisory control [48, 49], satellite design [24], water-treatment facilities [34], and train control systems [10].

In the following, we will demonstrate how IMCs provide a precise formal semantics and enable compositional design and verification by examples about the industrial specification formalisms of dynamic fault trees [12] and Statemate [10].

6.1 Dynamic Fault Trees with Input/Output IMCs

Fault trees (FT) constitute a prominent formalism in reliability analysis to model how the failure propagation of a system's components induces a failure of the whole system [13, 12]. Its intuitive graphical syntax is often used for reliability analysis in industry. The leaves of a FT represent component failures called *basic events*, and all non-leaves indicate how component failures propagate through the system, modelled by so called *gates*. The root node represents the system failure, called the *top-level event*.

Static fault trees allow the use of the logic AND, OR and VOTING gates. Dynamic Fault Trees (DFT) extend them with a number of dynamic gates, to model common patterns in reliability engineering: functional dependencies can be specified via the FDEP gate; spare management via the SPARE gate ; and sequencing via the PAND gate. Further, each basic event is equipped with a probability distribution showing how the failure behaviour evolves over time.

Example 16 (Cardiac assist system). Figure 10 depicts a DFT representing a cardiac assist system (CAS) [11] consisting of three types of subsystems: the CPU, the motor unit, and the pump unit. If either one of these subsystem fails then the entire CAS fails, as modelled by the top-level OR gate.

The CPU unit consists of a primary (P) and backup (B) CPU, indicated by the SPARE gate. Both are subject to a common cause failure represented by the

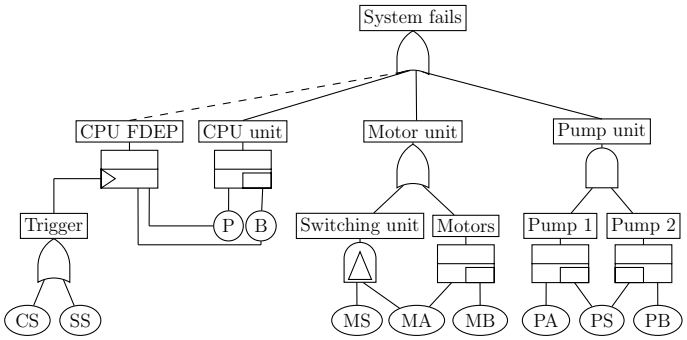


Figure 10: The cardiac assist system DFT.

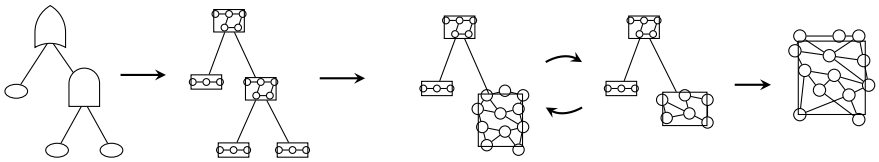
CPU FDEP gate: if either a crossbar switch (CS) or the system supervisor (SS) fails, both become unavailable.

The motor unit consists of a primary (MA) and backup motor (MB). If the primary motor fails and the switching component (MS) is still available, the backup motor is turned on. If the switching component fails afterwards, this can be ignored, as modelled by the PAND gate.

The pump unit consists of two primary pumps (PA and PB) which share a backup pump (PS). Thus, one of the primary pumps can be replaced after failing, and the pump unit fails, if all pumps are unavailable, represented by the AND gate. ■

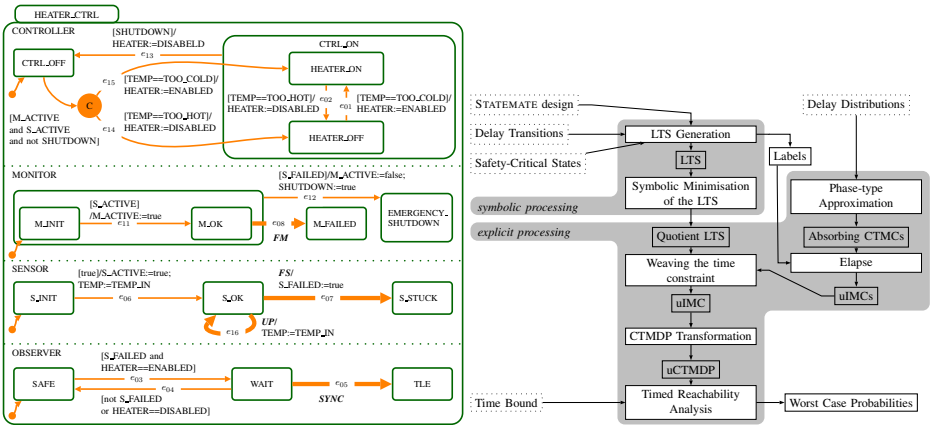
Given a DFT, one is typically interested in calculating the reliability of the whole system over time. An efficient way to do so is the transformation of a DFT into an Input/Output Interactive Markov Chains (I/O-IMC). I/O-IMCs [12] extend IMCs by integrating features from input/output automata. Interactive transitions are partitioned into *input actions* and *output actions*. Input actions can only be taken, if another I/O-IMC executes a matching output action. This refinement enables one to define which component triggers a synchronization and which merely reacts. This can be readily exploited to model the components' dependencies with respect to failure propagation in DFTs.

The process chain from a DFT to its semantical IMC is depicted in Figure 11. The behaviour of each leaf and gate is encoded as an I/O-IMC. The interaction between components is modelled by means of input and matching output actions. Composition and abstraction techniques explained in Section 4 can then be used to aggregate a DFT into one IMC representation for the whole system. Finally, the IMC can be analysed either directly or after the transformation to a CTMDP.



(a) DFT (b) Transformation (c) Composition (d) Minimization (e) IMC

Figure 11: Graphical overview of the compositional aggregation of DFT models.



(a) A heater system [10] (b) The tool chain [10]

Figure 12: An example of STATEMATE design and the tool chain for quantitative evaluation of STATEMATE

6.2 Compositional Performability Evaluation for STATEMATE

STATEMATE [31] is a statechart-based tool set used by engineers in several avionic and automotive companies like AIRBUS and BMW [10]. In this section we explain an approach proposed in [10], which enables performability evaluation of STATEMATE models. It applies various construction, optimisation and analysis techniques including compositional modelling using IMCs. In fact, the use of IMCs plays a crucial role in the model construction part of the methodology. We first recapitulate an example taken from [10] to show the applicability of STATEMATE and then explain the function of IMCs in the methodology.

Example 17. Figure 12a shows a STATEMATE *design* that represents the functional behaviour of a heating system. It consists of a `CONTROLLER`, a `MONITOR`, a `SENSOR`, and an `OBSERVER`. The `SENSOR` repeatedly measures and stores the temperature. However, it is an unreliable component, which means that it might become inactive due to a defect and, therefore, does not update the temperature value. As soon as the `MONITOR` detects a sensor failure, it shuts the system down in order to avoid severe damage. The `CONTROLLER` constantly checks the current temperature and turns the heater on/off when the temperature is too cold/too hot. The `OBSERVER`'s role is to observe whether a *safety-critical* state has been reached. In this example, the high level state `TLE` of the `OBSERVER`, which specifies a situation where the sensor has failed while the heater is still on, is safety-critical. There are also *delay transitions* denoted by bold arrows. They specify the event that is triggered as a delay passes. Here they signal a component failure; for example `FM` and `FC` indicate failures of the monitor and the sensor respectively. ■

The question that is naturally raised in such models is whether the risk of reaching some safety-critical state within a certain time bound is below a certain

threshold. Such a question can be answered by using the tool chain (Figure 12b) devised in [10]. The input of the tool chain consists of several parts including the STATEMATE model, specification of safety-critical states and distributions of delay transitions. Based on these inputs, the tool chain computes the *worst-case probability* to reach some safety-critical state within the provided time bound. The given delay distribution is specified as a uniform IMC (uIMC), which is then composed with the model generated from input STATEMATE design. The result is later transformed into a uniform CTMDP (uCTMDP) by applying the steps in Section 2.6. Finally, a worst case time-bounded reachability analysis [6] is performed on the uCTMDP. As a case study, the proposed technique has been successfully employed in the area of train control systems [10].

6.3 Tool Support

IMCA [27] is a tool for the quantitative analysis of IMCs and was recently extended to Markov automata. In particular, it supports the verification of IMCs against unbounded reachability, time- and interval-bounded reachability, expected-time objectives, and long-run average objectives. Hence, it supports the model-checking algorithms presented in this paper, whereas it is not capable of parsing a CSL formula. IMCA computes the maximum and minimum values for a set of goal states.

CADP [18] supports construction, minimisation and analysis of extended Markovian models including IMCs. It compiles and generates the state space from a specification. The compositional verification engine of CADP then composes a network of communicating IMCs. The tool set also enables minimisation modulo strong and branching bisimulation. Furthermore, it supports the steady-state and transient analysis of the final model via numerical verification techniques or simulation. The analysis is, however, restricted to models that exhibit only spurious nondeterminism.

SCOOP [57] is a tool that symbolically optimises process-algebraic specifications of probabilistic processes including IMCs. Optimisations such as dead-variable reduction and confluence reduction are applied automatically by SCOOP. The optimised state spaces are ready to be analysed by, for instance, CADP or IMCA. Moreover, SCOOP and IMCA constitute a tool chain called MAMA [28], which supports construction, minimisation and analysis of IMCs, among other models (e. g. MA).

MRMC [42] is a model checker for discrete-time and continuous-time Markov reward models. It supports the verification of PCTL and CSL as well as their reward extensions CSRL and PCTRL. There is also a CTMDP extension available[5] which provides recent analysis techniques based on [16].

[5] `http://depend.cs.uni-sb.de/tools/ctmdp/`

7 Conclusion

This paper presents an overview about IMCs, a fruitful combination of CTMCs and LTSs which facilitates the modeling of and reasoning about probabilistic systems. A great strength of IMCs is their compositional semantics which established them as a prominent formalism for a wide range of applications. We presented the theoretical framework of IMCs and introduced related concepts such as composition and schedulers. The main reason for the application of IMCs in system models is the plethora of available analysis techniques. Given a certain model, one typically wants to examine qualitative aspects such as reachability of certain system states but also quantitative aspects such as time-dependent probabilities and long-run behaviour. We provided an overview of state-of-the-art algorithms to answer these questions. A key to efficient computations is the reduction of the state space of the underlying model by exploitation of behavioural equivalences. In this context we introduced the notion of strong and weak bisimulation and presented algorithms to derive bisimulation quotients. Equipped with these techniques, IMCs have been successfully applied to a number of real-world problems and integrated into various models, especially as semantical model. We presented IMCs as the semantics of two industrial specification formalisms: Dynamic fault trees and Statemate.

Despite extensive progress on theoretical results and numerous tools and application developments over the last decade, IMCs still form a highly active field of research. First progress towards the measurability and analysis of IMCs has been made in [40] and future works might exploit IMCs' close entanglement with CTMDPs. Two extensions of IMCs, time-inhomogeneous IMCs and Markov automata, have been introduced. The former exhibits inhomogeneity in its embedded CTMC, while the latter adds the capability of random switching to interactive transitions. Tool support for IMCs is already available, for instance by CADP and IMCA. The latter has recently been integrated with SCOOP as a fully-fledged tool chain, called MaMa, which supports modelling, reduction and analysis of Markov automata, and IMCs as a special case of Markov automata. However, there is still room for improvements in time-bounded computations and possible extensions to Markov reward analysis. Given the advantages of IMCs, especially their expressiveness and compositional semantics, one can investigate opportunities to widen their application range to new concepts and formalisms.

References

[1] de Alfaro, L.: Formal Verification of Probabilistic Systems. Ph.D. thesis, Stanford University (1997)

[2] de Alfaro, L.: How to specify and verify the long-run average behavior of probabilistic systems. In: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 454–465. IEEE (1998)

[3] de Alfaro, L.: Computing minimum and maximum reachability times in probabilistic systems. In: Proceedings of the 10th International Conference on Concurrency Theory (CONCUR). LNCS, vol. 1664, pp. 66–81. Springer (1999)

[4] Ash, R., Doléans-Dade, C.: Probability & Measure Theory. Academic Press (2000)
[5] Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on Software Engineering 29(6), 524–541 (2003)
[6] Baier, C., Hermanns, H., Katoen, J.P., Haverkort, B.R.: Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. Theoretical Computer Science 345(1), 2–26 (2005)
[7] Baier, C., Katoen, J.P.: Principles of model checking, vol. 950. MIT press (2008)
[8] Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Mathematics of Operations Research 16(3), 580–595 (1991)
[9] Bianco, A., de Alfaro, L.: Model checking of probabalistic and nondeterministic systems. In: Foundations of Software Technology and Theoretical Computer Science (FSTTCS). pp. 499–513 (1995)
[10] Böde, E., Herbstritt, M., Hermanns, H., Johr, S., Peikenkamp, T., Pulungan, R., Rakow, J., Wimmer, R., Becker, B.: Compositional dependability evaluation for STATEMATE. IEEE Transactions on Software Engineering 35(2), 274–292 (2009)
[11] Boudali, H., Dugan, J.B.: A Bayesian network reliability modeling and analysis framework. IEEE Transactions on Reliability 55, 86–97 (2005)
[12] Boudali, H., Crouzen, P., Stoelinga, M.: Dynamic fault tree analysis using input/output interactive Markov chains. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 708–717 (2007)
[13] Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V.Y., Noll, T., Roveri, M.: Safety, dependability and performance analysis of extended AADL models. The Computer Journal 54(5), 754–775 (2011)
[14] Bravetti, M., Hermanns, H., Katoen, J.P.: YMCA: Why Markov chain algebra? Electronic Notes in Theoretical Computer Science (ENTCS) 162, 107–112 (2006)
[15] Brázdil, T., Hermanns, H., Krcál, J., Kretínský, J., Rehák, V.: Verification of open interactive Markov chains. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). pp. 474–485 (2012)
[16] Buchholz, P., Hahn, E.M., Hermanns, H., Zhang, L.: Model checking algorithms for CTMDPs. In: Proceedings of the 23rd International Conference on Computer Aided Verification (CAV). pp. 225–242 (2011)
[17] Chatterjee, K., Henzinger, M.: Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In: Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1318–1336. SIAM (2011)
[18] Coste, N., Garavel, H., Hermanns, H., Lang, F., Mateescu, R., Serwe, W.: Ten Years of Performance Evaluation for Concurrent Systems Using CADP. In: Proceedings of the 4th International Conference on Leveraging Applications of Formal Methods, Verification, and Validation. ISoLA, vol. 6416, pp. 128–142 (2010)
[19] Coste, N., Hermanns, H., Lantreibecq, E., Serwe, W.: Towards performance prediction of compositional models in industrial GALS designs. In: Proceedings of the 21st International Conference on Computer Aided Verification (CAV). pp. 204–218 (2009)
[20] Crouzen, P., Hermanns, H.: Aggregation ordering for massively compositional models. In: Proceedings of the10th International Conference onApplication of Concurrency to System Design (ACSD). pp. 171 –180. IEEE (june 2010)

[21] Crouzen, P., Hermanns, H., Zhang, L.: On the minimisation of acyclic models. In: Breugel, F., Chechik, M. (eds.) Proceedings of the 19th International Conference on Concurrency Theory (CONCUR), LNCS, vol. 5201, pp. 295–309. Springer (2008)
[22] Eisentraut, C., Hermanns, H., Zhang, L.: Concurrency and composition in a stochastic world. In: Proceedings of the 21st International Conference on Concurrency Theory (CONCUR). pp. 21–39. LNCS, Springer (2010)
[23] Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 342–351. IEEE (2010)
[24] Esteve, M.A., Katoen, J.P., Nguyen, V.Y., Postma, B., Yushtein, Y.: Formal correctness, safety, dependability, and performance analysis of a satellite. In: Proceedings of the 34th International Conference on Software Engineering (ICSE). pp. 1022–1031. IEEE (2012)
[25] Giacalone, A., Jou, C., Smolka, S.A.: Algebraic reasoning for probabilistic concurrent systems. In: Proceedings of the IFIP TC2 Working Conference on Programming Concepts and Methods. pp. 443–458 (1990)
[26] Guck, D.: Quantitative Analysis of Markov Automata. Master's thesis, RWTH Aachen University (2012)
[27] Guck, D., Han, T., Katoen, J.P., Neuhäußer, M.R.: Quantitative timed analysis of interactive Markov chains. In: Goodloe, A., Person, S. (eds.) Proceedings of the 4th International Conference on NASA Formal Methods (NFM), LNCS, vol. 7226, pp. 8–23. Springer (2012)
[28] Guck, D., Hatefi, H., Hermanns, H., Katoen, J.P., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST). LNCS, vol. 8054, pp. 55–71. Springer (2013)
[29] Han, T., Katoen, J., Mereacre, A.: Compositional modeling and minimization of time-inhomogeneous Markov chains. In: Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control (HSCC). LNCS, vol. 4981, pp. 244–258. Springer (April 2008)
[30] Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
[31] Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., Trakhtenbrot, M.B.: STATEMATE: A working environment for the development of complex reactive systems. IEEE Transactions on Software Engineering 16(4), 403–414 (1990)
[32] Hatefi, H., Hermanns, H.: Model checking algorithms for Markov automata. Electronic Communications of the ECEASST 53 (2012)
[33] Hatefi, H., Hermanns, H.: Improving time bounded reachability computations in interactive markov chains. In: FSEN. pp. 250–266 (2013)
[34] Haverkort, B.R., Kuntz, M., Remke, A., Roolvink, S., Stoelinga, M.: Evaluating repair strategies for a water-treatment facility using Arcade. In: Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 419–424 (2010)
[35] Hermanns, H.: Interactive Markov Chains. Springer, Berlin (2002)
[36] Hermanns, H., Katoen, J.: The how and why of interactive Markov chains. LNCS, vol. 6286, pp. 311–337. Springer (2010)
[37] Hermanns, H., Johr, S.: Uniformity by construction in the analysis of nondeterministic stochastic systems. In: Proceedings of the 37th Annual IEEE/IFIP Inter-

national Conference on Dependable Systems and Networks (DSN). pp. 718–728 (2007)
[38] Hermanns, H., Johr, S.: May we reach it? Or must we? In what time? With what probability? In: MMB. pp. 125–140. VDE Verlag (2008)
[39] Hermanns, H., Katoen, J.P., Meyer-Kayser, J., Siegle, M.: ETMCC: Model checking performability properties of Markov chains. In: Proceedings of the 33rd International Conference on Dependable Systems and Networks (DSN). IEEE Computer Society (2003)
[40] Johr, S.: Model checking compositional Markov systems. Ph.D. thesis, Saarland University (2008)
[41] Kanellakis, P.C., Smolka, S.A.: CCS expressions, finite state processes, and three problems of equivalence. Information and Computation 86(1), 43–68 (1990)
[42] Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The Ins and Outs of the probabilistic model checker MRMC. In: Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems (QEST). pp. 167–176. IEEE (2009)
[43] Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS), LNCS, vol. 2324, pp. 200–204. Springer (2002)
[44] Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing (preliminary report). In: Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 344–352. ACM (1989)
[45] Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94, 1–28 (1991)
[46] López, G., Hermanns, H., Katoen, J.P.: Beyond memoryless distributions: Model checking semi-Markov chains. In: Process Algebra and Probabilistic Methods. Performance Modelling and Verification. (PAPM-PROBMIV). pp. 57–70. No. 2165 in LNCS, Springer (2001)
[47] López, N., Núnez, M.: An overview of probabilistic process algebras and their equivalences. In: Validation of Stochastic Systems, LNCS, vol. 2925, pp. 89–123. Springer (2004)
[48] Markovski, J.: Towards supervisory control of interactive Markov chains: Controllability. In: 11th International Conference on Application of Concurrency to System Design (ACSD). pp. 108–117 (2011)
[49] Markovski, J.: Towards supervisory control of interactive Markov chains: Plant minimization. In: 9th IEEE International Conference on Control and Automation (ICCA). pp. 1195–1200 (2011)
[50] Neuhäußer, M.R.: Model checking nondeterministic and randomly timed systems. Ph.D. thesis, RWTH Aachen University (2010)
[51] Neuhäußer, M.R., Katoen, J.P.: Bisimulation and logical preservation for continuous-time Markov decision processes. In: Proceedings of the 18th International Conference on Concurrency Theory (CONCUR). pp. 412–427. Springer (2007)
[52] Pulungan, R.: Reduction of Acyclic Phase-Type Representations. Ph.D. thesis, Universität des Saarlandes, Saarbruecken, Germany (2009)
[53] Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming, vol. 414. John Wiley & Sons (2009)
[54] Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons (1998)

[55] Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2, 250–273 (1995)
[56] Sen, K., Viswanathan, M., Agha, G.: VESTA: A statistical model-checker and analyzer for probabilistic systems. In: Proceedings of the 2nd International Conference on the Quantitative Evaluation of Systems (QEST). pp. 251–252. IEEE (2005)
[57] Timmer, M., Katoen, J.P., van de Pol, J., Stoelinga, M.: Efficient modelling and generation of Markov automata. In: Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR). LNCS, vol. 7454, pp. 364–379. Springer (2012)
[58] Younes, H.L.: Ymer: A statistical model checker. In: Proceedings of the 17th International Conference on Computer Aided Verification (CAV). LNCS, vol. 3576, pp. 429–433. Springer (2005)
[59] Zhang, L., Neuhäußer, M.: Model checking interactive Markov chains. In: Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 6015, pp. 53–68. Springer (2010)