From Concurrency Models to Numbers

Performance and Dependability

Holger HERMANNS a,b, and Lijun ZHANG c

a Saarland University – Computer Science, Saarbrücken, Germany
b INRIA Grenoble – Rhône-Alpes, France
c DTU Informatics, Technical University of Denmark, Denmark

Abstract. Discrete-state Markov processes are very common models used for performance and dependability evaluation of, for example, distributed information and communication systems. Over the last fifteen years, compositional model construction and model checking algorithms have been studied for these processes, and variations thereof, especially processes incorporating nondeterministic choices. In this paper we give a survey of model checking and compositional model construction for such processes in discrete and continuous time.

Keywords. Markov chain, Markov decision process, continuous-time, logic, model checking, bisimulation, congruence, compositionality

1. Introduction

Compositional model construction and model checking are two aspects of a modern strategy to guarantee correctness of system designs. In the era of power-aware, wireless and distributed systems, correctness guarantees for such systems must become quantifiable. Instead of just guaranteeing that a system is performing its expected tasks, we are ultimately interested in guaranteeing that the system performs its tasks within given time limits. Or better: that the probability of doing so is above a given threshold.

In this paper we paint the landscape of behavioral models for probability and time, with a focus on concurrent Markov models and process calculi. We discuss foundational and practical aspects of compositional modeling and model checking of such systems. In addition to introducing the basic state-based probabilistic behavioral models, this paper looks at them from two principal viewpoints. On the one hand side, we discuss the compositional construction of concurrent probabilistic models. On the other hand, we discuss algorithmic aspects of model checking for probabilistic extensions of the temporal logic CTL.

We first (Section 2) introduce our core state-based probabilistic behavioral model, discrete-time Markov chains (DTMC). In this model, time advances in discrete steps. We introduce probabilistic computation tree logic, PCTL, and elaborate on algorithms for model checking DTMCs with respect to PCTL properties. We then turn (Section 3) our attention to continuous time Markov chains (CTMC), and discuss continuous stochastic logic, CSL, and its model checking.
In Section 4 we review foundational aspects of modeling concurrent systems based on labeled transition systems (LTS), touching upon synchronous execution, interleaving and handshaking. Based on the insights gained, we then ask how concurrent execution can be incorporated into DTMCs and CTMCs. We find that synchronous execution fits well with the DTMC view, while independent interleaving is natural in the context of CTMCs. We see that the latter can be considered as the limit of the former, in a deep sense. We also find that in full generality, neither DTMCs nor CTMCs are closed under concurrent execution with handshaking. This is unfortunate, because handshaking is the key operation for embedding a full formal semantics of concurrent system behavior into the probabilistic setting. The feature that is missing is nondeterminism.

Section 5 therefore discusses DTMCs with nondeterminism, also known as Markov decision processes (MDP), and reiterates the model construction and model checking problems for MDPs with respect to PCTL. Section 6 is devoted to the CTMC setting with nondeterminism. This gives rise to the model of interactive Markov chains (IMC), and again we discuss model construction for IMCs and touch upon model checking IMC with respect to CSL properties.

Section 7 is devoted to a detailed discussion of the scientific bibliography on which the present paper is based. A broader discussion concludes the paper in Section 8.

2. Discrete-Time Markov Chains

A stochastic process is a family of random variables, taking values in a finite discrete set $S$, usually referred to as the state space of a process. A discrete-time Markov chain (DTMC) is a stochastic process $\{X_n \mid n \in \mathbb{N}\}$ that satisfies the Markov property, which requires that future behavior only depends on the state occupied at the current time. In mathematical terms, the Markov property requires that for all $n \in \mathbb{N}$,

$$P\{X_{n+1} = s' \mid X_n = s, \ldots, X_0 = s_0\} = P\{X_{n+1} = s' \mid X_n = s\}.$$  

Thus, for DTMCs, the fact that the chain was in state $s_{n-1}$ at time $n-1$, in state $s_{n-2}$ at time $n-2$, and so on, up to the fact that it was in state $s_0$ at time 0 is entirely irrelevant. The state $X_n$ is the only relevant history information that is needed to determine the distribution over $S$ at time $n+1$. DTMCs can be described concisely by a triple:

**Definition 2.1** A DTMC is a triple $\mathcal{D} = (S, [P_i]_{i \geq 0}, \text{init})$, where

- $S$ is a non-empty finite set of states,
- for $i \in \mathbb{N}$, $P_i : S \times S \to [0,1]$ is a probabilistic transition matrix defined by $P_i(s,s') = P(X_{i+1} = s' \mid X_i = s)$, and
- $\text{init} : S \to [0,1]$ with $\text{init}(s) = P(X_0 = s)$ is the initial distribution.

Obviously, for each state $s$ and all $i$, it holds that $P_i(s,S) = 1$ where $P_i(s,C) := \sum_{s' \in C} P_i(s,s')$ for $C \subseteq S$. We refer to the function $\omega : \mathbb{N} \to S$ as a step function, which is also called a (sample) path. Let $\Omega$ denote the set of all sample paths of the DTMC. A cylinder set is defined as $\text{Cyl}(s_0, \ldots, s_n) = \{\omega \in \Omega \mid \forall 0 \leq i \leq n, \omega(i) = s_i\}$. Let $\mathcal{F}$ denote the $\sigma$-algebra of the sets in the space $\Omega$ generated by all cylinder sets. Now the DTMC can be defined as the probability triple $(\Omega, \mathcal{F}, P)$, where $P$ is the uniquely determined probability measure satisfying: $P(\text{Cyl}(s))$ equals $\text{init}(s)$, and for $k \geq 0$, 

\[P(Cyl(s_0, \ldots, s_{k+1})) = P(Cyl(s_0, \ldots, s_k)) \cdot P_k(s_k, s_{k+1}).\]

### 2.1. Homogeneous DTMCs

A DTMC is **homogeneous** if its transition probabilities are independent of the time instant of observation. In this case, we gain the freedom to arbitrarily choose the origin of the time axis. In technical terms, homogeneity requires that for \(n' \geq n\),

\[P\{X_{n'} = s' \mid X_n = s\} = P\{X_{n'-n} = s' \mid X_0 = s\}.\]

Homogeneous DTMCs can be described in a very concise way:

**Definition 2.2** A homogeneous DTMC is a triple \(D = (S, P, \text{init})\), where

- \(P : S \times S \to [0, 1]\) is a probabilistic transition matrix which is defined by
  \[P(s, s') = P(X_1 = s' \mid X_0 = s).\]

In the remainder of this section we shall assume that DTMCs are always homogeneous. We say that \(s\) is absorbing if \(P(s, s) = 1\). When making \(s\) absorbing we modify the transition from \(s\) such that \(P(s, s) = 1\) and \(P(s, s') = 0\) for \(s \neq s'\).

For a DTMC, the **transient distribution** \(\pi(n)\) at step \(n\) is defined as a vector (indexed by states) representing the probability to be in state \(s\) at time step \(n\):

\[\pi(n, s) := P(X_n = s).\]

The vector \(\pi(n)\) can also expressed as \(\pi(n) = \text{init} \cdot P^n\). We are also interested in the **long run fraction** of state occupancies when \(n\) tends to infinity. For this, we define the Cesàro limit vector \(lfr\) as

\[lfr(s) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \pi(i, s).\]

The Cesàro limit always exists, and \(lfr(s)\) corresponds to the long run fraction of occupying state \(s\). If \(\lim_{n \to \infty} \pi(n)\) exists, it agrees with the Cesàro limit \(lfr\).

In the context of model checking, we often analyze a DTMC with a single initial state. In this case, we specify it as a subscript for clarification: for example, we write \(P_s\) for the probability measure, \(\pi_s(n)\) for the transient distribution at step \(n\), and \(lfr_s\) for the long run average limit, provided that the initial state is \(s\). Similarly, we sometimes use a superscript to indicate the model being analyzed, for instance \(P^D_s\), \(\pi^D(n)\) and \(lfr^D\) refer to the measures induced by the DTMC \(D\).

![Figure 1. Zeroconf Markov example](image-url)
Example 2.1 In Figure 1, a simple DTMC model for the IPv4 Zeroconf protocol, designed for local home networks, is shown. The initial state $s_0$ is indicated by an incoming arrow.

When a new host joins the network, an address among the $K = 65024$ possible addresses is randomly selected. With $m$ hosts in the network, the collision probability is $q = m/K$. The host asks other hosts whether they are using this address. If a collision occurs, the host tries to detect this by waiting for an answer. The probability that the host gets no answer in the case of a collision is $p$; for example, due to lossy channels. In this case it repeats the question. If after $n$ tries the host does not get any answer, it will erroneously consider the chosen address to be valid.

2.2. Probabilistic CTL

Throughout the whole paper, we will assume that the state space $S$ is always equipped with labels that identify distinguishing state properties. For this, we let $AP$ denote a set of atomic propositions, and assume, whenever needed, a labeling function $L : S \rightarrow 2^{AP}$, such that $L(s)$ specifies the set of properties that hold in state $s$.

Example 2.2 For the Zeroconf example in Figure 1, $AP = \{\text{state, error}\}$, and the labeling of states is depicted near the corresponding states.

The logic probabilistic CTL (PCTL), a variation of the computational tree logic (CTL), is introduced to specify properties over DTMCs:

Syntax. The syntax of PCTL state formulas $\Phi$ and PCTL path formulas $\varphi$ is given by:

$$\Phi := a \mid \Phi \land \Phi \mid \neg \Phi \mid P_J(\varphi) \mid L_J(\Phi)$$

$$\varphi := X \Phi \mid \Phi U \Phi \mid \Phi U^I \Phi$$

where $a \in AP$, $J \subseteq [0, 1]$ is a non-empty interval with rational bounds, and $I$ is a closed interval on $\mathbb{N}$. We introduce the following derived operators: $\text{true} = a \lor \neg a$, $\Phi_1 \lor \Phi_2 = \neg(\neg\Phi_1 \land \neg\Phi_2)$, $\Diamond J \Phi = \text{true} U^I \Phi$ and $\Box I \Phi = \neg(\Diamond I \neg \Phi)$. Often, abbreviations are used for the interval $I$: for example, $\leq p$ denotes $[0, p]$. The operator $L_J(\Phi)$ is referred to as the long run, or steady state operator.

Example 2.3 In the Zeroconf example, the formula $P_{\leq 0.01}(\Diamond^{[0,100]} \text{error})$ expresses that the probability of reaching the error state within 100 steps is bounded by 0.01.

Below we exemplify how various availability properties can be expressed using PCTL, where the atomic proposition $up$ characterizes all states in which the DTMC is operational.

- $L_J(up)$: steady state availability,
- $P_J(\Diamond^{[n,n]} up)$: instantaneous availability at step $n$,
- $P_J(\Phi U^{[n,n]} up)$: conditional instantaneous availability at step $n$,
- $P_J(\Box^{[n,n]} up)$: interval availability,
- $L_J(P_J(\Box^{[n,n]} up))$: steady state interval availability,
- $P_J(\Phi U^{[n,n]} L_J(up))$: conditional time bounded steady state availability.
\textit{Semantics.} The semantics of PCTL state formulas is given by: \( s \models a \) iff \( a \in L(s) \), \( s \models \neg \Phi \) iff \( s \not\models \Phi \), \( s \models \Phi_1 \land \Phi_2 \) iff \( s \models \Phi_1 \) and \( s \models \Phi_2 \). For probabilistic formulas, we have:

\[
s \models P_\omega(\varphi) \iff P_s(\{\omega \in \Omega \mid \omega \models \varphi\}) \in J
\]

where \( P_s(\{\omega \in \Omega \mid \omega \models \varphi\}) \), or \( P_s(\varphi) \) for short, denotes the probability measure of the set of all paths which satisfy \( \varphi \). For the long run operator, the semantics is given by:

\[
s \models L_\omega(\Phi) \iff \sum_{s' \in s} \delta(s')r_{ss'}(s') \in J.
\]

The satisfaction relation for PCTL path formulas is defined as follows. Let \( \omega \in \Omega \) be a path, then,

- \( \omega \models X\Phi \) iff \( (\omega(1)) \models \Phi \),
- \( \omega \models \Phi_1 U\Phi_2 \) iff there exists a \( n \in \mathbb{N} \) such that \( \omega(n) \models \Phi_2 \), and for each integer \( 0 \leq i < n \) we have \( \omega(i) \models \Phi_1 \),
- \( \omega \models \Phi_1 U^i \Phi_2 \) iff there exists a \( n \in \mathbb{N} \) such that \( \omega(n) \models \Phi_2 \), and for each integer \( 0 \leq i < n \) we have \( \omega(i) \models \Phi_1 \).

Obviously, the next state operator \( X\Phi \) can be expressed by a bounded until formula \( true U^{[1, i]} \Phi \). Moreover, the unbounded until formula can also be expressed as \( \Phi_1 U \Phi_2 = \Phi_1 U^i \Phi_2 \). It is not difficult to show that for any PCTL path formulas \( \varphi \), the set \( \{\omega \in \Omega \mid \omega \models \varphi\} \) is measurable.

\textit{Notation.} Let \( D = (S, \mathbf{P}, init) \) be a DTMC. We use \(|S|\) to denote the number of states, and \(|D|\) the number of non-zero probabilistic transitions. For a state \( s \in S \), we use \( \delta_s \) to denote the point distribution (or Dirac distribution) satisfying \( \delta_s(s) = 1 \) and \( \delta_s(s') = 0 \) for \( s' \neq s \). For a PCTL state formula \( \Phi \), we use \( \Im \) to denote the indicator vector for \( \Phi \) defined by: \( \Im_{\Phi}(s) \) equals 1 if \( s \models \Phi \) and 0 otherwise. We use \( \mathbf{m} \) to denote the indicator matrix defined by: \( \mathbf{m}_{\Phi}(s,s') \) equals 1 if \( s \models \Phi \) and \( \mathbf{m}_{\Phi}(s,s') = 0 \) otherwise. We use \( D[\Phi] \) to denote the DTMC obtained from \( D \) by making states that satisfy \( \Phi \) absorbing. Moreover, let \( P_\Phi \) denote the probabilistic transition matrix of \( D[\Phi] \).

\subsection{Model Checking DTMCs}

Let \( D = (S, \mathbf{P}, init) \) be a DTMC, \( s \in S \), and \( \Phi \) be a PCTL state formula. The model checking problem is to determine whether \( s \models \Phi \). The model checking algorithm for PCTL recursively computes the sets \( Sat(\Psi) \) for all state subformulas \( \Psi \) of \( \Phi \). The cases where \( \Psi \) is an atomic proposition, a negation or a conjunction are given by: \( Sat(a) = \{s \in S \mid a \in L(s)\} \), \( Sat(\neg \Psi) = S \setminus Sat(\Psi) \) and \( Sat(\Psi_1 \land \Psi_2) = Sat(\Psi_1) \cap Sat(\Psi_2) \).

The case where \( \Psi \) has the probabilistic operator as its outermost operator is the challenging part, i.e. \( \Psi = P_\omega(\psi) \), and as explained above it is enough to consider the case \( \psi = \Psi_1 U^i \Psi_2 \). By the semantics, checking \( \Psi \) is then equivalent to checking whether \( P_s(\Psi_1 U^i \Psi_2) \in J \). Assume that the sets \( Sat(\Psi_i) \) have already been calculated. For \( i = 1, 2 \), we replace \( \Psi_i \) by fresh atomic propositions \( f_i \) and extend the label of state \( s \) by \( f_i \) if \( s \in Sat(\Psi_i) \). The resulting path formula is then \( f_1 U^i f_2 \), and obviously we have \( P_s(\Psi_1 U^i \Psi_2) = P_s(f_1 U^i f_2) \). Below we consider the computation of \( P_s(f_1 U^i f_2) \).
\
Bounded Until with $I = [0, n]$. We define $g_1(s, n) := P_s^D(f_1 \cup [0, n] f_2)$, and denote by $g_1(n)$ the vector with: $g_1(n)(s) = g_1(s, n)$. By the semantics of PCTL path formulas, making $f_2$-states absorbing will not affect $g_1(s, n)$, and the same holds for states not satisfying $f_1$. Thus, it holds that $g_1(s, n) = \sum_{s' = f_2} \pi_s^{D[-f_1 \lor f_2]}(n, s')$. In matrix form, we then have: $g_1(s, n) = \pi_s^{D[-f_1 \lor f_2]}(n) \cdot i_{f_2} = \delta_s \cdot (P_{\neg f_1} [-f_1 \lor f_2^n] \cdot i_{f_2})$. Therefore:
\
$$g_1(n) = (P_{\neg f_1} [-f_2])^n \cdot i_{f_2}.$$  
(2)
\
In particular, $g_1(s, n)$ equals 1 if $s \in \text{Sat}(f_2)$, and equals 0 if $s \in \text{Sat}(\neg f_1) \setminus \text{Sat}(f_2)$.

Bounded Until with $I = [n, n + c]$, $n > 0$ and $c \geq 0$. Now we define $g_2(s, n) := P_s^D(f_1 \cup [n, n + c] f_2)$, and denote by $g_2(n)$ the vector with: $g_2(n)(s) = g_2(s, n)$. If $\omega = f_1 \cup [n, n + c] f_2$, then $\omega(i) \models f_1$ for all $i < n$. For $s \not\models f_1$, it holds $g_2(s, n) = 0$.

Otherwise, by the law of total probability, we have:
$$g_2(s, n) = \sum_{s' \models f_1} \pi_s^{D[-f_1]}(n - 1, s') \cdot P_s^D(f_1 \cup [1, c + 1] f_2)$$
$$= \sum_{s' \models f_1} \pi_s^{D[-f_1]}(n - 1, s') \cdot \sum_{s' \in S} P(s', s) \cdot P_{s'}^D(f_1 \cup [0, c] f_2).$$

Assume that $g_2(c)$ is already computed according to (2). We can rewrite the above equation in matrix form, and get:
$$g_2(n) = (P_{\neg f_1}[-f_1])^{n-1} \cdot m_{f_1} \cdot P \cdot g_1(c)$$  
(3)
\
where $(P_{\neg f_1})^0$ denotes the identity matrix for the case $n = 1$.

Unbounded Until $I = \mathbb{N}$. As preparation, we define the set of zero states $S_{=0} = \{s \in S \mid P_s(f_1 \cup [0, n] f_2) = 0\}$, and the set of almost-sure states $S_{=1} = \{s \in S \mid P_s(f_1 \cup [n, n + c] f_2) = 1\}$. Intuitively, $S_{=0}$ is the set of states which cannot reach $\text{Sat}(f_2)$. Moreover, $S_{=1}$ is the set of states which can reach $S_{=0}$. Thus, both sets can be computed in linear time $O(|D|)$ by analyzing the underlying graph, for example via simple depth-first search. Now let $S_1 := S \setminus (S_{=1} \cup S_{=0})$. Obviously, for all $s \in S_1$, it holds that $f_1 \not\models L(s)$, $f_2 \not\models L(s)$ and $P_s(f_1 \cup f_2) > 0$. The probabilities $(P_s(f_1 \cup f_2))_{s \in S_1}$ can be characterized by solving the following equation system: for each $s \in S_1$,
$$x_s = \sum_{s' \in S_1} P(s, s') x_{s'} + \sum_{s \in S_{=1}} P(s, u).$$

Let $T$ denote the transition matrix restricted to $S_1$, i.e., $T(s, s') = P(s, s')$ iff $s, s' \in S_1$. Moreover, let the vector $b = (b_s)_{s \in S}$ be defined by $b_s = P_s(s, S_{=1})$. Then, the above equation system can be expressed in matrix form: $x = T \cdot x + b$ with $x = (x_s)_{s \in S_1}$. The vector $(P_s(f_1 \cup [n, n + c] f_2))_{s \in S}$ is the unique solution of the equation system $x = T \cdot x + b$.

The unbounded operator with interval $[c, \infty)$ can then be handled directly: we only need to replace the initial value $g_1(c)$ in (3) by the vector $(P_s(f_1 \cup [n, n + c] f_2))_{s \in S}$. 

**Long run Average Operator** \(L_2(\Phi)\). Assume \(\Phi\) has already been computed. We determine first the set \(B\) of **bottom strongly connected components** (BSCC) of the underlying graph. Applying the unbounded reachability analysis, we can then compute the probability of reaching each BSCC \(B\), i.e. \(P_s(\Phi B)\) where we assume that \(B \in L(s)\) iff \(s \in B\). For each \(B \in B\), the long run average distribution within \(B\), denoted by \(lf^B\), can be computed by solving \(\pi \cdot P|_B = \pi\), where \(P|_B\) is the probability matrix restricted to the set of states \(B\). Finally, \(\sum_{s' \in \text{Sat}(\Phi)} \text{lf}_s(s')\) is obtained as follows:

\[
\sum_{s' \in \text{Sat}(\Phi)} \text{lf}_s(s') = \sum_{B \in B} \left( P_s(\Phi B) \sum_{s' \in B \cap \text{Sat}(\Phi)} \text{lf}_B(s') \right).
\]

**Theorem 2.1** The model checking problem for DTMCs with respect to PCTL state formulas is decidable in polynomial time.

Bounded reachability can be computed by performing matrix vector multiplications. For instance, for the case \(I = [0, n]\) (see (2)), \(g_1(0) = f_2\), and for \(n > 0\), \(g_1(n) = (P[-f_1 \lor f_2]) \cdot g_1(n-1)\). The case where \(I = [n, n+c]\) (see (3)) can be handled in a similar way: \(g_2(0) = g_1(c), g_2(1) = m_{f_1} \cdot P \cdot g_1(c), \) and for \(n > 1\), \(g_2(n) = (P[-f_1]) \cdot g_2(n-1)\). The complexity is linear in \(|D|\) and the step bounds \(n\) and \(n+c\) respectively. For unbounded reachability and the long run average operator, equation systems of size \(|S|\) can be solved in \(O(|S|^3)\), for example with Gaussian elimination.

The overall complexity is then \(O(|\Phi| \cdot (|D|n_{max} + |S|^3))\) where \(|\Phi|\) denotes the length of the formula to be checked and \(n_{max}\) is the maximum step bound of the time bounded until subformulas occurring in \(\Phi\).

**2.4. Bisimulation for DTMCs**

Bisimulation relations have been extensively studied in the literature to compare the behavior of transition systems, and later extended to a quantitative variant of bisimulation for DTMCs.

**Definition 2.3** Let \(D = (S, P, \text{init})\) be a DTMC. A **bisimulation** on \(D\) is an equivalence relation \(R\) on \(S\) such that for all \((s_1, s_2) \in R\), it holds that:

- \(L(s_1) = L(s_2)\), and
- \(P(s_1, C) = P(s_2, C)\) for all \(C \in S/R\).

Two states \(s_1, s_2\) are bisimilar, denoted by \(s_1 \sim s_2\), if there exists a bisimulation relation containing \((s_1, s_2)\).

Intuitively, the first condition states that bisimilar states are equally labeled. In the second condition we require that bisimilar states have the same transition probability of moving to each equivalence class. The bisimulation relation \(\sim\) is an equivalence relation. Moreover, it agrees with PCTL equivalence:

**Theorem 2.2** Let \(D = (S, P, \text{init})\) be a DTMC. Then, \(s_1 \sim s_2\) iff for all PCTL formulas \(\Phi\) it holds that:

\[
s_1 \models \Phi \iff s_2 \models \Phi.
\]
3. Continuous-Time Markov Chains

In DTMCs all transitions take equidistant time units, as suggested by the name discrete-time. In this section we consider continuous-time Markov chains (CTMC), which have transition rates that specify a distribution of transition times.

A stochastic process \( \{X_t \mid t \in \mathbb{R}_{\geq 0}\} \) is a CTMC if it satisfies the Markov property, namely for each sequence of time instances \( t_{n+1} > t_n > t_{n-1} > t_{n-2} > \ldots > t_0 \), we have that for each state \( s' \):

\[
P\{X_{t_{n+1}} = s' \mid X_{t_n} = s_n, \ldots, X_{t_0} = s_0\} = P\{X_{t_{n+1}} = s' \mid X_{t_n} = s_n\} .
\]

As for DTMCs, we shall assume that CTMCs are homogeneous, which requires that we have (for \( t' \geq t \)),

\[
P\{X_{t'} = s' \mid X_t = s\} = P\{X_{t' - t} = s' \mid X_0 = s\} .
\]

We denote by \( P_t \) the matrix defined by: \( P_t(s, s') = P\{X_t = s' \mid X_0 = s\} \).

**Transition Rate Matrix.** In CTMCs, the probability of going from state \( s \) to \( s' \) depends on the length of the interval of observation. As the interval \([0, h]\) becomes very small, the probability of observing a transition is also very small, thus the probability of observing multiple transitions is negligible. The transition rate between \( s \) and \( s' \) is defined by:

\[
Q(s, s') = \lim_{h \to 0} \left\{ \frac{P_h(s, s')}{h} \right\} \text{ if } s' \neq s .
\]

The rate corresponds to the expected number of transitions per unit time, and is as such independent of the length of the interval. It follows that: \( P_h(s, s') = Q(s, s') h + o(h) \) if \( s' \neq s \) where \( o(h) \) subsumes the probabilities of passing through intermediate states between \( s \) and \( s' \) during the interval \([0, h]\). The rate \( Q(s, s) \) is defined by \( Q(s, s) := \lim_{h \to 0} (P_h(s, s) - 1)/h \). By the law of total probability, \( \sum_{s'} P_h(s, s') = 1 \), thus \( Q(s, s) \) can be derived as follows: \( Q(s, s) = - \sum_{s' \neq s} Q(s, s') \). In matrix form, we have

\[
Q = \lim_{h \to 0}\{(P_h - I)/h\}
\]

where \( I \) denotes the identity matrix. We define \( E(s) := -Q(s, s) = \sum_{s' \neq s} Q(s, s') \), and refer to \( E(s) \) as the exit rate of \( s \). We say \( s \) is absorbing if \( E(s) = 0 \), in which case \( P_t(s, s) = 1 \) for all \( t \in \mathbb{R}_{\geq 0} \). When making \( s \) absorbing, we modify the rates from \( s \) such that \( Q(s, s') = 0 \) for all \( s' \).

**Probability Measure.** As for DTMCs, we first describe finite homogeneous CTMCs in an alternative way:

**Definition 3.1** A CTMC is a triple \( C = (S, Q, \text{init}) \), where \( S \) and \( \text{init} \) are defined as for DTMCs, and

- \( Q : S \times S \to \mathbb{R} \) is the transition rate matrix.
We can construct an induced probability transition matrix $P$ for $\mathcal{C}$ defined as follows: $P(s, s) = 1$ if $s$ is absorbing, and otherwise, $P(s, s') = Q(s, s')/E(s)$ for $s \neq s'$. We refer to the DTMC $\text{emb}(\mathcal{C}) = (S, P, \text{init})$ as the embedded DTMC of $\mathcal{C}$. It describes the behavior of the CTMC if we only observe it at the jump times, i.e. the discrete instances of time where a jump occurs.

A right continuous step function $\omega : \mathbb{R}_{\geq 0} \to S$ is called a (sample) path. For $i \in \mathbb{N}$, we denote by $\omega_S[i] = s_i$ the state occupied at the $(i+1)$-th segment of the step function, and by $\omega_T[i]$ the time spent in $\omega_S[i]$, i.e. the length of the step segment starting with $\omega_S[i]$. A finite fragment of a sample path $\omega$ is depicted in Figure 2. Let $\Omega$ denote the set of all sample paths of the CTMC. Let $I_0, I_1, \ldots, I_{k-1}$ be non-empty intervals in $\mathbb{R}_{\geq 0}$. The cylinder set $\text{Cyl}(s_0, I_0, s_1, I_1, \ldots, s_k)$ is defined by:

$$\text{Cyl}(s_0, I_0, \ldots, s_k):= \{ \omega \in \Omega \mid \forall 0 \leq i \leq k. \omega_S[i] = s_i \land \forall 0 \leq i < k. \omega_T[i] \in I_i \}.$$  

Let $\mathcal{F}$ denote the smallest $\sigma$-algebra on $\Omega$ containing all cylinder sets. Now the CTMC can be defined as the probability triple $(\Omega, \mathcal{F}, P)$, where $P$ is the uniquely determined probability measure satisfying: $P(\text{Cyl}(s)) = \text{init}(s)$, and for $k \geq 0$,

$$P(\text{Cyl}(s_0, I_0, \ldots, s_{k+1})) = P(\text{Cyl}(s_0, I_0, \ldots, s_{k})) \cdot P(s_k, s_{k+1}) \cdot \eta(s_k, I_k)$$

where $\eta(s_k, I_k) := e^{-E(s_k)} \inf I_k - e^{-E(s_k)} \sup I_k$ is the probability to take a transition during time interval $I_k$ from $s_k$.

3.1 Transient and Steady State Distributions

**Chapman-Kolmogorov Equations.** For $s, s' \in S$, the Chapman-Kolmogorov equations for a CTMC are given by:

$$P_t(s, s') = \sum_{u \in S} P_t'(s, u)P_{t-t'}(u, s') \quad \forall t', 0 \leq t' \leq t$$

which can be derived directly by applying the Markov property. The above equations can be used to derive $P_t(s, s') = \lim_{h \to 0}(P_{t+h}(s, s') - P_t(s, s'))/h$ as follows:
\[ \dot{P}_t(s, s') = \lim_{h \to 0} \frac{\sum_{u \in S} P_t(s, u) P_h(u, s') - P_t(s, s')}{h} \]
\[ = \sum_{u \neq s'} P_t(s, u) \cdot \lim_{h \to 0} \frac{P_h(u, s')}{h} + P_t(s, s') \cdot \lim_{h \to 0} \frac{P_h(s', s') - 1}{h} \]

which implies that
\[ \dot{P}_t(s, s') = \sum_{u \in S} P_t(s, u) Q(u, s') - \text{known as the Chapman-Kolmogorov forward equations.} \]

In matrix form, they are:
\[ \dot{P}_t = P_t \cdot Q. \quad (5) \]

The solution of the equations is given by the matrix exponential
\[ P_t = \text{init} \cdot e^{Qt}. \]

The Chapman-Kolmogorov backward equations can be derived in a similar way:
\[ \dot{P}_t = Q \cdot P_t. \quad (6) \]

**Transient Distribution.** The transient distribution \( \pi(t) \) at time \( t \) is defined by: \( \pi(s, t) = P(X_t = s) \). Thus, it holds that:
\[ \pi(t) = \text{init} \cdot \dot{P}_t = \text{init} \cdot e^{Qt}. \quad (7) \]

Moreover, by the Chapman-Kolmogorov forward equations (see (5)), it holds that:
\[ \dot{\pi}(t) = \text{init} \cdot \dot{P}_t = \text{init} \cdot P_t \cdot Q = \pi(t) \cdot Q. \]

**Steady State Distribution.** For finite CTMCs, which we consider in this paper, the limit
\[ \lim_{t \to \infty} \pi(t) \]
always exists, which is referred to as the steady state distribution. As with DTMCs, we denote this limit by the vector \( \text{brf} := \lim_{t \to \infty} \pi(t) \). In the steady state,
\[ \dot{\pi}(t) = 0, \]
thus \( \pi \) can be computed by solving the equation system \( \text{brf} \cdot Q = 0 \).

Again, for model checking, we often analyze the CTMC with a single initial state. In this case, we use a subscript for clarification in the same manner as for a DTMC: we use \( P_s \) for the probability measure, \( \pi_s(t) \) for the transient distribution at step \( t \), and \( \text{brf}_s \) for the long run average limit, starting from the initial state \( s \), and use a superscript to indicate the model being analyzed, for instance \( P^C \), \( \pi^C(t) \) and \( \text{brf}^C \) for the CTMC \( C \).

![Figure 3. Triple Modular Redundant System](image)

**Example 3.1** A simple model of a triple modular redundancy system (TMR) is shown in Figure 3, in the form of a fault-tolerant computer voting system consisting of three
processors and a single (majority) voter. States \( s_{i,j} \) model that \( i \in \{0, 1, 2, 3\} \) processors and \( j \in \{0, 1\} \) voter are operational. Failures are modeled by the transition rates, with \( \lambda \) for the processor and \( \nu \) for the voter. A failure of the voter will cause the failure of the entire system. Repair rates are \( \mu \) and \( \delta \) for the processor and the voter, respectively, and the initial state \( s_{3,1} \) is indicated by an incoming arrow.

3.2. Continuous-Time Stochastic Logic

Syntax. The syntax of CSL state formulas \( \Phi \) and CSL path formulas \( \varphi \) is given by:

\[
\Phi := a | \Phi \land \Phi | \neg \Phi | P_J(\varphi) | L_J(\Phi)
\]

\[
\varphi := X^I \Phi | \Phi \cup \Phi | \Phi \cup^I \Phi
\]

where \( a \in AP \), \( J \subseteq [0, 1] \) is an interval with rational bounds, and \( I \) is a non-empty closed interval on \( \mathbb{R}_{\geq 0} \). We assume that the endpoint of the intervals is rational, if it is not infinite. Thus, the syntax of CSL state formulas is identical with PCTL state formulas, and moreover, CSL path formulas extend PCTL path formulas with closed intervals of \( \mathbb{R}_{\geq 0} \) specifying time constraints. Similar derived operators as for PCTL are used, for example \( \Diamond^I \Phi = \text{true} \cup^I \Phi \) and \( \Box^I \Phi = \neg (\Diamond^I \neg \Phi) \). The operator \( L_J(\Phi) \) is referred to as the steady state operator for CTMCs. CSL can be used to express various availability properties for CTMCs, having the same form as for DTMCs.

Example 3.2 Recall we assume that states are equipped with labels to identify state properties. For the TMR example in Figure 3, the labeling of states is depicted next to the corresponding states.

Example 3.3 Below we list a few exemplary properties of interest for the TMR example.

- \( L_{s_{2,1}} \): In the steady state, the probability that the system is in state \( s_{2,1} \) is at most \( p \).
- \( P_{s_{3,1}}(\Diamond^{[0.1]} \text{down}) \): The probability of being down at time \( t \) meets the bound \( > p \).
- \( P_{s_{3,1}}(\text{up}_3 \lor \text{up}_2) \cup^{[0,10]} \text{down} \): The probability of reaching a down-state within 10 time units after having continuously operated with at least two processors is at most 0.01.
- \( L_{s_{3,1}}(P_{s_{2,1}}(\Box^{[0,10]} \neg \text{down})) \): In steady state, with probability at least 0.9, the probability that the system will not go down within 10 time units is at least 0.8.

Semantics. The semantics of CSL state formulas agrees with that for PCTL state formulas (see Section 2.2). The semantics of CSL path formulas is defined as follows. Let \( \omega \in \Omega \) be a path. Then,

- \( \omega \models X^I \Phi \text{ iff } \omega_T[0] \in I \text{ and } \omega_S[1] \models \Phi \),
- \( \omega \models \Phi_1 \cup \Phi_2 \text{ iff there exists a } t \in \mathbb{R}_{\geq 0} \text{ such that } \omega(t) \models \Phi_2, \text{ and for each } 0 \leq t' < t \text{ we have } \omega(t') \models \Phi_1 \),
- \( \omega \models \Phi_1 \cup^I \Phi_2 \text{ iff there exists a } t \in I \text{ such that } \omega(t) \models \Phi_2, \text{ and for each } 0 \leq t' < t \text{ we have } \omega(t') \models \Phi_1 \).
Thus, the unbounded until formula can also be expressed as \( \Phi_1 \cup \Phi_2 = \Phi_1 \cup^{(0, \infty)} \Phi_2 \).

For any CSL path formula \( \varphi \), \( \{ \omega \in \Omega \mid \omega \models \varphi \} \) is measurable.

The notions of indicator vector \( I_\Phi \) and indicator matrix \( m_\Phi \) from DTMCs carry over to CTMCs directly. For a CSL state formula \( \Phi \), let \( C[\Phi] \) denote the CTMC obtained from \( C \) by making states in \( \text{Sat}(\Phi) \) absorbing. Moreover, let \( Q[\Phi] \) denote the rate matrix of \( C[\Phi] \). For \( s \in S \) and a set \( C \subseteq S \), we define \( Q(s, C) := \sum_{s' \in C} Q(s, s') \).

### 3.3. Model Checking CTMCs

Let \( C = (S, Q, \text{init}) \) be a CTMC, \( s \in S \), and \( \Phi \) be a CSL state formula. The model checking problem is to check whether \( s \models \Phi \). Boolean formulas can be handled exactly the same way as in PCTL. The probabilistic next formula \( \mathbb{P}_f(X, \psi) \) is measurable.

Boolean formulas can be handled exactly the same way as in PCTL. The probabilistic next formula \( \mathbb{P}_f(X, \psi) \) can be verified directly: \( P_a(X, \psi) = \eta(s, I) \cdot \sum_{s' \in \text{Sat}(\psi)} P(s, s') \). Obviously, when \( I = \mathbb{R}_{\geq 0} \), this agrees with the procedure for checking \( \mathbb{P}_f(X, \psi) \) in the embedded DTMC.

Below we discuss the case of the probabilistic until formula \( \mathbb{P}_f(f_1 \cup f_2) \) where \( f_1 \) and \( f_2 \) are atomic propositions. We consider the following different cases.

#### Bounded Until with \( \mathbb{I} = [0, t] \)

Slightly abusing our notations, we define \( g_1(s, t) := \mathbb{P}^c_x(f_1 \cup^{[0, t]} f_2) \), and denote by \( g_1(t) \) the vector with \( g_1(t)(s) = g_1(s, t) \). By the semantics of CSL path formulas, making states in \( \text{Sat}(\neg f_1 \lor f_2) \) absorbing would not change \( g_1(s, t) \), thus: \( g_1(s, t) = \sum_{s' \models f_2} \pi_{s'}^{c[\neg f_1 \lor f_2]}(s', t) \).

In matrix form, we have
\[
g_1(s, t) = \pi_{s}^{c[\neg f_1 \lor f_2]}(t) \cdot i_{f_2} = \delta_x \cdot e^{Q[\neg f_1 \lor f_2] \cdot t} \cdot i_{f_2}
\]
where the last equality follows by applying (7). Then we have:
\[
g_1(t) = e^{Q[\neg f_1 \lor f_2] \cdot t} \cdot i_{f_2}.
\]  

Differentiating both sides of the above equation, together with (6), we have:
\[
\dot{g}_1(t) = Q[\neg f_1 \lor f_2] \cdot e^{Q[\neg f_1 \lor f_2] \cdot t} \cdot i_{f_2} = Q[\neg f_1 \lor f_2] \cdot g_1(t)
\]
with the initial condition that \( g_1(0) = i_{f_2} \). These equations are referred to as the Chapman-Kolmogorov equations for bounded reachability \( g_1 \). Equivalently, the element-wise characterization is given by:
\[
\dot{g}_1(s, t) = \sum_{s' \in S} Q[\neg f_1 \lor f_2](s, s')g_1(s', t)
\]
with initial condition \( g_1(s, 0) = 1 \) if \( s \in \text{Sat}(f_2) \) and 0 otherwise. Since states in \( \text{Sat}(\neg f_1 \lor f_2) \) are absorbing, it holds that \( g_1(s, t) = 0 \) for all \( s \models \neg f_1 \lor f_2 \). Thus, for all \( t \), \( g_1(s, t) \) equals 1 if \( s \in \text{Sat}(f_2) \) and equals 0 if \( s \in \text{Sat}(\neg f_1) \setminus \text{Sat}(f_2) \).

#### Bounded Until with \( \mathbb{I} = [t, t + c] \), \( t > 0 \) and \( c \geq 0 \)

Slightly abusing notations, we define \( g_2(s, t) := \mathbb{P}^c_x(f_1 \cup^{[t, t+c]} f_2) \), and denote by \( g_2(t) \) the vector with \( g_2(t)(s) = g_2(s, t) \). By the law of total probability, \( g_2(s, n) \) can be expressed by:
\[
g_2(s, t) = \sum_{s' \models f_1} \pi_{s'}^{c[\neg f_1]}(s', t) \cdot P^c_x(f_1 \cup^{[0,c]} f_2).
\]
Assume that \( g_1(c) \) is already computed according to (8). In matrix form, we have: 
\[
g_2(s, t) = \pi_s e^{-\int f_1(t)} \cdot m_{f_1} \cdot g_1(c).
\]
Applying (7) we get the analytical solution:
\[
g_2(t) = e^{Q[-f_1 t]} \cdot m_{f_1} \cdot g_1(c) = e^{Q[-f_1 t]} \cdot m_{f_1} \cdot e^{Q[-f_1 \lor f_2] c} \cdot 1_{f_2}.
\]
Thus the probability \( P_s(f_1 \cup (t, t+c) f_2) \) for bounded until can be expressed by a product of matrix exponentials (equation (9)). We can now also obtain the Chapman-Kolmogorov equations for \( g_2 \) by differentiating both sides of (9): 
\[
\dot{g}_2(t) = Q[-f_1] \cdot g_2(t).
\]
Thus the probability \( g_2(s, 0) = m_{f_1}(s, s) \cdot g_1(s, c) \). Below we recall the decidability result for CTMC model checking:

**Theorem 3.1** The model checking problem for CTMCs with respect to CSL state formulas is decidable.

The unbounded until operator with \( I = [0, \infty) \) can be computed in the embedded DTMC, namely: 
\[
P_s(f_1, U f_2) = P^{emb(C)}_s(f_1 \cup f_2).
\]
The unbounded until with \( I = [c, \infty) \) can be handled similarly to that DTMCs: we only need to replace the initial value \( g_1(c) \) in (9) by the vector \((P_s(f_1 \cup [0, \infty) f_2))_{s \in S}\). The steady state probabilities can be computed by solving \( b f \cdot Q = 0 \). Now we briefly discuss bounded reachability with the interval \([0, t]\). We want to check whether \( g_1(s, t) \) meets the given bound, for example \( s < p \). We have seen that \( g_1 \) can be expressed by matrix exponential: 
\[
g_1(s, t) = \delta_s \cdot e^{Q[-f_1 \lor f_2] t} \cdot 1_{f_2}.
\]
Aziz et al. [3] have shown that this can be simplified into an algebraic number, and an effective procedure exists for checking whether \( g_1(s, t) < p \).

### 3.4. Uniformization

The decision procedure for CSL exploits results in algebraic and transcendental number theory, but is not easily implementable for time bounded reachability. For this purpose, an approximate algorithm is proposed, in which numerical algorithms based on uniformization are used to approximate the matrix exponential \( e^{Q t} \). The algorithmic core executes most calculations on the uniformized DTMC, as we will explain below. For a CTMC, we say that \( \lambda \) is a uniformization rate if \( \lambda \geq \max_{s \in S} E(s) \).

**Definition 3.2** Let \( C = (S, Q, init) \) be a CTMC. The uniformized DTMC of \( C \) with respect to the uniformization rate \( \lambda \) is \( uni(C) = (S, P_\lambda, init) \) where 
\[
P_\lambda(s, s') = Q(s, s')/\lambda \text{ if } s \neq s' \text{ and } P_\lambda(s, s) = 1 - P_\lambda(s, S \setminus \{s\}).
\]

By definition it holds that \( P_\lambda = I + Q/\lambda \) where \( I \) denotes the identity matrix. First, we consider the approximation of \( g_1(t) \) in (8). Assume that states satisfying \( (\neg f_1 \lor f_2) \) are absorbing. Thus, \( g_1(t) = e^{Q t} \cdot 1_{f_2} \). For \( t > 0 \), then:
\[
g_1(t) = e^{Q t} \cdot 1_{f_2} = e^{(P_\lambda - I) \lambda t} \cdot 1_{f_2} = e^{-\lambda t} \sum_{i=0}^{\infty} \frac{(\lambda t)^i}{i!} \cdot (P_\lambda)^i \cdot 1_{f_2}.
\]
Let \( \psi(i, \lambda t) = e^{-\lambda t} \cdot (\lambda t)^i \), which denotes the \( i \)-th Poisson probability with parameter \( \lambda t \). Moreover, we define a vector \( v_i \) by: \( v_0 = 1_{f_1} \), and \( v_{i+1} = P_{f_1} \cdot v_i \). Thus, it holds \( g_1(t) = \sum_{i=0}^{\infty} \psi(i, \lambda t) v_i \). This is a backward computation starting from the indicator vector \( 1_{f_1} \).

A forward computation would require repeating the computation \( O(|S|) \) times, since an initial state must be fixed a priori.

The vector \( v(i) \) gives the \( i \)-step bounded reachabilities in the uniformized DTMC. Intuitively, any number of transitions might happen in \( C \) within time \( t \), but the probability to see precisely \( i \) transitions within that time is governed by a Poisson probability with parameter \( \lambda t \). Hence, \( g_1(t) \) equals the infinite sum of the reachability probabilities of \( \text{uni}(C) \), weighted by the corresponding Poisson probability.

For a given accuracy \( \varepsilon \), the Fox–Glynn algorithm can be used to approximate the Poisson probabilities \( \psi(i, \lambda t) \) in a stable way in time \( O(\sqrt{\lambda t}) \). The infinite sum will be truncated after \( O(\lambda t) \) steps, ensuring that the error is below \( \varepsilon \). The complexity is linear in the size of the CTMC, and also linear in \( \lambda t \).

The other case with the interval \([t, t + c]\) can be handled in a very similar way. By (9), we have \( g_2(t) = e^{Q[t_1][t]} \cdot m_{f_1} \cdot g_1(c) \) where \( g_1(c) \) can be computed as above. Also a quite similar computation can be derived, by essentially only adapting the initial vector so that \( v_0 = m_{f_1} \cdot g_1(c) \).

The overall complexity for the approximate model checking algorithm is then \( O(|\Phi| \cdot (|C| t_{\text{max}} + |S|^3)) \) where \( |\Phi| \) denotes the length of the formula to be checked and \( t_{\text{max}} \) is the maximum time bound of the time bounded until subformulas occurring in \( \Phi \).

### 3.5. Bisimulation for CTMCs

For an equivalence relation \( R \) over \( S \) and \( s \in S \), we use \([s]_R \) to denote the unique equivalence class containing \( s \). We drop the subscript \( R \) if the equivalence relation considered is clear from the context. Bisimulation is defined for CTMCs as follows:

**Definition 3.3** Let \( C = (S, Q, \text{init}) \) be a CTMC. A bisimulation on \( C \) is an equivalence relation \( R \) on \( S \) such that for all \((s_1, s_2) \in R\), it holds that:

1. \( L(s_1) = L(s_2) \), and
2. \( Q(s_1, C) = Q(s_2, C) \) for all \( C \in S/R \) with \( C \neq [s_1] \).
3. \( Q(s_1, [s_1] \setminus \{s_1\}) = Q(s_2, [s_1] \setminus \{s_2\}) \).

Two states \( s_1, s_2 \) are bisimilar, denoted by \( s_1 \sim s_2 \), if there exists a bisimulation relation containing \((s_1, s_2)\).

Instead of comparing the probability of moving to each equivalence class, in CTMCs we compare the rate moving to them. Since \( Q([s_1], [s_1]) = - \sum_{C \neq [s_1]} Q(s_1, C) \), the second clause implies directly \( Q(s_1, [s_1]) = Q(s_2, [s_1]) \). For the case \( C \) agrees with the equivalence class \([s_1]\), it requires that the exit rates to the set \([s_1] \setminus \{s_1\}\) agree for \( s_1 \) and \( s_2 \). Consider the simple CTMC depicted in Figure 4. States \( s_0 \) and \( s_1 \) are bisimilar only in the case that \( x = 2 \). However, dropping the last clause, they are bisimilar for all \( x \geq 0 \). Notably, dropping the last clause gives the notion of traditional lumpability [49], or equivalently weak bisimulation.

The bisimulation relation \( \sim \) is an equivalence relation, and agrees with the CSL equivalence for CTMCs:
Figure 4. A CTMC for illustrating bisimulations

\textbf{Theorem 3.2} Let $\mathcal{C} = (S, Q, \text{init})$ be a CTMC. Then, $s_1 \sim s_2$ iff for all CSL formulas $\Phi$ it holds that:

$$s_1 \models \Phi \iff s_2 \models \Phi.$$ 

4. Model Construction for Markov Chains

We now investigate how – on the basis of the models we have seen thus far, DTMCs and CTMCs – models of concurrent computation can be constructed. To have a blueprint of a sound solution to this problem at hand, we first consider the non-probabilistic systems, and look at the mother of all concurrency models, labeled transition systems.

4.1. Labeled Transition Systems

Labeled transition systems (LTS) are the most basic model for concurrent, communicating systems. Formally, a LTS is a tuple $T = (S, \text{Act}, \rightarrow, s_0)$ where $\text{Act}$ is a set of actions, $\rightarrow \subseteq S \times \text{Act} \times S$ is the transition relation, and $s_0$ is the initial state. The use of a transition relation (instead of a transition function) makes it possible to represent nondeterminism in a general sense. In particular $s \xrightarrow{\alpha} s'$, $s \xrightarrow{\alpha} s''$ and $s' \neq s''$ is possible.

Given a system composed out of two LTSs running concurrently, how does the composed system behave? This depends on how the real systems being modeled can be faithfully represented. Inspired by process algebra this is often achieved by a composition operator whose semantics is defined via a set of structural operational semantic rules. Below we discuss three generic variants of this approach. We fix $T_1 = (S_1, \text{Act}_1, \rightarrow_1, s_{01})$ and $T_2 = (S_2, \text{Act}_2, \rightarrow_2, s_{02})$ as two LTSs to be composed.

\textit{Full Synchronization.} We require $\text{Act}_1 = \text{Act}_2 = \{\alpha\}$ in this case. The fully synchronous product of $T_1$ and $T_2$ is $T_1 \otimes T_2 = (S_1 \times S_2, \text{Act}, \rightarrow, (s_{01}, s_{02}))$ where $\rightarrow$ is given by:

\[
\frac{s_1 \xrightarrow{\alpha_1} s'_1 \land s_2 \xrightarrow{\alpha_2} s'_2}{(s_1, s_2) \xrightarrow{\alpha} (s'_1, s'_2)}
\]

(10)

Intuitively, a transition in the product is achieved by executing the two transitions with the same action in the components in lock-step, thus at the same time. The restriction to a singleton action set can be relinquished [11], but the flavor of this operator remains, namely that transitions are executed in lock-step.
Interleaving. If $T_1$ and $T_2$ execute completely independent of each other, we have to consider all possible interleaving of all transitions of $T_1$ and $T_2$. The composite transition system is:

$$T_1 \parallel T_2 = (S_1 \times S_2, \text{Act}_1 \cup \text{Act}_2, \rightarrow, \langle s_{01}, s_{02} \rangle)$$

where the transition relation $\rightarrow$ is given by:

$$s_1 \xrightarrow{\alpha_1} s'_1 \quad \text{and} \quad s_2 \xrightarrow{\alpha_2} s'_2 \quad \Rightarrow \quad \langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s'_1, s'_2 \rangle$$

(11)

Here, the idea that the two LTSs may coincidently change state at the same time is not represented. Each transition of the composed LTS is rooted in the transition of one, and precisely one component LTS.

Handshaking. Both full synchronization and interleaving are limited because they lack the possibility of independent progress combined with message passing synchronization. The latter is the key for communicating systems. With handshake communication, two LTSs synchronize their behavior over an agreed set, or alphabet $\text{Syn}$ and interleave their execution otherwise. The parallel composition $T_1 \parallel \text{Syn} T_2 \equiv (S_1 \times S_2, \text{Act}_1 \cup \text{Act}_2, \rightarrow, \langle s_{01}, s_{02} \rangle)$ is defined as follows. For every action $\alpha \in \text{Act}_i \setminus \text{Syn}$, we have the interleaving rule as (11), and for $\alpha \in \text{Syn}$, we have a handshaking communication as in (10). Thus, this operator encompasses the pure interleaving operator when $\text{Syn} = \emptyset$, and the synchronous product when $\text{Syn} = \text{Act}_1 = \text{Act}_2 = \{\alpha\}$. Furthermore, the handshaking concept in this operator can be used to encode unbuffered and (arbitrary-size) buffered message passing, as well as shared-variable communication. This means that this one operator $\parallel \text{Syn}$ can be considered as the generator for a formal semantics of a large variety of concurrent communication systems.

4.2. Synchronous Composition for DTMCs

We now look at the question what happens if one attempts to extend the operators discussed above to the model of DTMCs.

Let $D_1 = (S_1, P_1, \text{init}_1)$ and $D_2 = (S_2, P_2, \text{init}_2)$ be two DTMCs. First, we write $s \xrightarrow{p} s'$ if $p = P_i(s, s') > 0$ for $i = 1, 2$. What happens if we run them in parallel?

It makes perfect sense to describe the parallel composition of two independent DTMCs as simply the synchronous product $D_1 \otimes D_2 = (S_1 \times S_2, P, \text{init}_1 \times \text{init}_2)$ with initial distribution $(\text{init}_1 \times \text{init}_2)(s_1, s_2) = \text{init}_1(s_1) \cdot \text{init}_2(s_2)$, and $P$ determined by:

$$s_1 \xrightarrow{p_1} s'_1 \quad \text{and} \quad s_2 \xrightarrow{q_2} s'_2 \quad \Rightarrow \quad \langle s_1, s_2 \rangle \xrightarrow{p \cdot q} \langle s'_1, s'_2 \rangle$$

(12)

We see that both chains proceed in lock-step, and this leads to the products of the individual transition probabilities occurring. This synchronous composition of DTMCs can also be understood as the product stochastic process of two independent stochastic processes, running on the same discrete time line: Let $\{X_n \mid n \in \mathbb{N}\}$ and $\{Y_n \mid n \in \mathbb{N}\}$ be the independent stochastic processes for $D_1$ and $D_2$ respectively. Moreover, let $\{Z_n \mid n \in \mathbb{N}\}$ with $Z_n = (X_n, Y_n)$ denotes the product stochastic process. Then, it holds that:

$$P(Z_1 = \langle s'_1, s'_2 \rangle \mid Z_0 = \langle s_1, s_2 \rangle) = P(X_1 = s'_1, Y_1 = s'_2 \mid X_0 = s_1, Y_0 = s_2)$$

$$= P(X_1 = s'_1 \mid X_0 = s_1) \cdot P(Y_1 = s'_2 \mid Y_0 = s_2)$$

(13)
The above line can be read as \( P(\langle s'_1, s'_2 \rangle, \langle s_1, s_2 \rangle) = P_1(s_1, s_2)P_2(s'_1, s'_2), \) reflecting the just defined synchronous DTMC product in (12).

Notably, the model of DTMCs is not closed under composition if one uses the interleaving or the handshake operator as a yardstick. This is because both make use of nondeterminism to represent the arbitrary, uncontrollable interleaving of independent actions. Nondeterminism can be represented in LTS, but not in DTMCs.

4.3. Interleaving Composition for CTMCs

We now turn our attention to the question what happens if one attempts to extend the operators discussed above to the model of CTMCs.

Let \( C_1 = (S_1, Q_1, \text{init}_1) \) and \( C_2 = (S_2, Q_2, \text{init}_2) \) be two CTMCs. We write \( s \xrightarrow{\lambda} s' \) if \( \lambda = Q_i(s,s') > 0 \) for \( i = 1, 2. \) In this setting, it now makes perfect sense to define the parallel composition for CTMCs as simply the interleaving \( C_1 || C_2 = (S_1 \times S_2, Q, \text{init}_1 \times \text{init}_2) \) with \( Q \) determined by:

\[
\begin{align*}
\langle s_1, s_2 \rangle & \xrightarrow{\lambda} \langle s'_1, s_2 \rangle \\
\langle s_1, s_2 \rangle & \xrightarrow{\lambda} \langle s_1, s'_2 \rangle
\end{align*}
\] (14)

To understand why this is a perfectly sensible definition, we derive the interleaving composition for CTMCs from the product stochastic process of two independent stochastic processes with the Markov property running in continuous time. Let \( \{X_t \mid t \in \mathbb{R}_{\geq 0}\} \) and \( \{Y_t \mid t \in \mathbb{R}_{\geq 0}\} \) be the independent stochastic processes for \( C_1 \) and \( C_2 \) respectively, and let \( \{Z_t \mid t \in \mathbb{R}_{\geq 0}\} \) with \( Z_t = \langle X_t, Y_t \rangle \) denotes the product stochastic process. Then, a variant of Eqn. (13) holds also for CTMCs:

\[ P(Z_t = \langle s'_1, s'_2 \rangle \mid Z_0 = \langle s_1, s_2 \rangle) = P(X_t = s'_1 \mid X_0 = s_1) \cdot P(Y_t = s'_2 \mid Y_0 = s_2) \] (15)

From this we can derive the rate matrix of the product stochastic processes, in a direct manner. For example, for the case \( \langle s_1, s_2 \rangle \neq \langle s'_1, s'_2 \rangle \), we obtain

\[
Q(\langle s_1, s_2 \rangle, \langle s'_1, s'_2 \rangle) = \lim_{h \to 0} \left\{ \frac{P(Z_h = \langle s'_1, s'_2 \rangle \mid Z_0 = \langle s_1, s_2 \rangle)}{h} \right\}
\]

By Eqn. (15), the right hand side is 0 for the case \( s_1 \neq s'_1 \land s_2 \neq s'_2 \). For the case \( s_1 = s'_1 \land s_2 \neq s'_2 \), it is equal to \( Q_2(s_2, s'_2) \), and for the case \( s_1 \neq s'_1 \land s_2 = s'_2 \), it is equal to \( Q_1(s_1, s'_1) \). This then establishes precisely the interleaving composition for CTMCs introduced in the above operational rule in (14). Notably, the rate for the case \( \langle s_1, s_2 \rangle = \langle s'_1, s'_2 \rangle \) can be determined using Eqn. (4).

Similar to the DTMC case, the other composition operators, full synchronization and handshaking, do not extend naturally to the CTMC case. A link to full synchronization will nevertheless be established next.

4.4. A Connection between DTMCs and CTMCs

This section sheds some more light on the concurrent behavior of Markov chains. Note that the sojourn time in a state of a DTMC follows a geometric distribution, while the
sojourn time follows an exponential distribution if the model is a CTMC. Furthermore, it is well known that the exponential distribution can be considered as the limit of the geometric distribution. We will show that this extends to parallel composition: the interleaving of CTMCs can be considered as limit of the synchronous composition for their discretized DTMCs.

Let $C = (S, Q, \text{init})$ be a CTMC, and $\{X_t\}$ the corresponding stochastic process. We derive the $\Delta$-discretized DTMC $C_\Delta = (S, P_\Delta, \text{init})$. Recall it holds that (see (4)): $Q = \lim_{\Delta \to 0} (P_\Delta - I)/\Delta$ where $I$ denotes the identity matrix. Stated differently, the behavior of a CTMC can be approximated arbitrarily closely by a $\Delta$-discretized DTMC, just by choosing $\Delta$ small enough. Below we discuss that this approximation is compatible with parallel composition: For two CTMCs $C$ and $C'$, let $C_\Delta$ and $C'_\Delta$ denote the corresponding $\Delta$-discretized DTMCs respectively. We now consider the synchronous product $C_\Delta \otimes C'_\Delta$, where the two Markov chains evolve in lock-step in discrete time with the step size – interpreted on a continuous time line – being $\Delta$. Now, how does this product relate to $C ||| C'$, the parallel composition of the CTMCs $C$ and $C'$ under interleaving semantics? The lemma below derives the interleaving semantics for CTMCs through the limiting behavior of their discretized DTMCs evolving synchronously.

**Lemma 4.1** Let $C = (S, Q, \text{init})$ and $C' = (S', Q', \text{init}')$ be two CTMCs, and let $C_\Delta, C'_\Delta$ denote the discretized DTMCs, respectively. Moreover, let $C ||| C'$ be the parallel composition of the two CTMCs $C$ and $C'$ under interleaving semantics. Then,

$$Q^{||} = \lim_{\Delta \to 0} (P^{||}_\Delta - I)/\Delta .$$

The proof is omitted, which can be derived directly applying (13) and (15).

**Example 4.1** For illustration, consider the two CTMCs depicted on the left part of Figure 5. Their interleaved parallel composition is depicted in the middle (state $(s, t)$ is written as $st$), and the synchronous parallel composition of the two discretized DTMCs with respect to $\Delta$ on the right (where self-loops are omitted). Consider states $st$ and $s't'$. By definition: $Q^{||}(st, s't') = \lim_{\Delta \to 0} (P^{||}_\Delta(st, s't')/\Delta) = \lim_{\Delta \to 0} o(\Delta)/\Delta = 0$. Similarly, the rates between other states can be obtained in a similar way.

## 5. Markov Decision Processes

In Section 4.1 we have discussed that the handshaking composition operator is a proper generalization of the other two operators considered (full synchronization and interleaving), and that this one operator can be seen as the nucleus for a formal semantics of a large variety of concurrent communication systems. It is therefore somewhat miserable that we have not been able to extend it to DTMCs and CTMCs. In this section we aim to overcome this problem, by introducing a model that extends DTMCs, but is closed under handshaking composition. The missing feature we need to support is nondeterminism. This leads us to the model class of Markov decision processes (MDP).

**Definition 5.1** A MDP is a tuple $M = (S, \text{Act}, P^d, \text{init})$ where $S$ and $\text{init}$ are defined as for DTMCs, and
Figure 5. Two simple CTMCs and their parallel composition. The loops in the synchronized DTMC are omitted, for example we have a self-loop probability $1 - \lambda \Delta - \mu \Delta + o(\Delta)$ for $st$.

- $\text{Act} = \bigwedge_{s \in S} \text{Act}_s$ is the set of decision vectors, with $\text{Act}_s$ being a finite set of enabled actions that can be taken in state $s \in S$, and
- $P^d$ is a probabilistic matrix for each decision vector $d$ of length $|S|$ with $d(s) \in \text{Act}_s$.

Nondeterministic choices in MDPs are resolved by a scheduler (aka. policy, adversary). A scheduler $\delta$ is a mapping from $\mathbb{N}$ into $\text{Act}$, the set of decision vectors. Let $\mathcal{S}$ denote the set of all schedulers. A scheduler $\delta \in \mathcal{S}$ is stationary (aka. memoryless) if for all $i,j \geq 0$, it holds $\delta(i) = \delta(j)$. An arbitrary scheduler $\delta \in \mathcal{S}$ induces a time-inhomogeneous DTMC $D = (S, \{P^{\delta(n)}\}_{n \geq 0}, \text{init})$ where $P^{\delta(n)}(s,s')$ denotes the probability of moving to $s'$ from $s$ under the decision $\delta(n)(s)$. If $\delta$ is a stationary scheduler, it induces a time-homogeneous DTMC.

The notations for step functions $\omega$, $\Omega$, cylinder sets, $\sigma$-algebra $\mathcal{F}$ for time inhomogeneous DTMCs carry over to (scheduled) MDPs directly. For a scheduler $\delta$, the MDP with initial distribution $\text{init}$ induces a probability space $(\Omega, \mathcal{F}, P^\delta_{\text{init}})$. If we have an initial state $s$ (i.e. $\text{init}(s) = 1$), we write $P^\delta_s$ instead of $P^\delta_{\text{init}}$.

5.1. Model Checking PCTL

We use the logic PCTL from Section 2.2 – without long run operator – to specify properties for MDPs. The semantics is the same as for DTMCs, except for the semantics of probabilistic formulas in Eqn. (1), which is obtained by ranging over all schedulers and the thus induced probability measures:

$$s \vdash P_\delta(\varphi) \iff \forall \delta. P^\delta_s(\{\omega \in \Omega \mid \omega \models \varphi\}) \in J.$$  (16)

We write $P^\delta_s(\varphi)$ to denote $P^\delta_s(\{\omega \in \Omega \mid \omega \models \varphi\})$. Below we discuss how to check the probabilistic operator for MDPs. The next formula is simple, and skipped here. We consider the until formula $\Phi = P_3(f_1 U f_2)$ with atomic propositions $f_1, f_2$. We define the optimal probabilities:

$$P^{opt}_s(f_1 U f_2) = \text{opt}_{\delta \in \mathcal{S}} P^\delta_s(f_1 U f_2)$$  (17)
with \( \text{opt} \in \{\sup, \inf\} \). Obviously, \( s \models \mathbb{P}_{<p}(\varphi) \) iff \( P^\sup_s(\varphi) \leq p \), and \( s \models \mathbb{P}_{\geq p}(\varphi) \) iff \( P^\inf_s(\varphi) \leq p \). Thus, model checking probabilistic formulas reduces to the computation of optimal probabilities. Below we discuss the computation of \( P^\text{opt}_s(f_1 \cup f_2) \) by considering the following different cases.

**Bounded Until with \( I = [0, n] \).** In this case the probability \( P^\text{opt}_s(f_1 \cup [0, n] f_2) \) can be computed as follows. For \( n = 0 \), \( P^\text{opt}_s(f_1 \cup [0, 0] f_2) \) equals 1 if \( s \in \text{Sat}(f_2) \), and equals 0 otherwise. For \( n > 0 \), \( P^\text{opt}_s(f_1 \cup [0, n] f_2) \) equals 1 if \( s \in \text{Sat}(f_2) \), equals 0 if \( s \in \text{Sat}(\neg f_1 \setminus \text{Sat}(f_2)) \), and otherwise:

\[
P^\text{opt}_s(f_1 \cup [0, n] f_2) = \text{opt}_{a \in \text{Act}_s} \sum_{s' \in S} P(s, a, s') P^\text{opt}_{s'}(f_1 \cup [0, n-1] f_2).
\]

The above equation can be used to compute \( v_n(s) := P^\text{opt}_s(f_1 \cup [0, n] f_2) \) efficiently in an iterative manner. Without loss of generality, as for DTMCs, we may assume that \( (\neg f_1 \lor f_2) \)-states are absorbing. Then, initially \( v_0(s) = P^\text{opt}_s(f_1 \cup [0, 0] f_2) \), and \( v_{i+1} = \text{opt}_{a \in \text{Act}_s}(P^d \cdot v_i) \) (optimal in all elements) for \( i \geq 0 \). This is also referred to as the well-known value iteration approach.

**Bounded Until with \( I = [n, n+c] \), \( n > 0 \) and \( c \geq 0 \).** In this case \( P^\text{opt}_s(f_1 \cup [n, n+c] f_2) \) can also be computed by applying a slightly modified version of the above value iteration step. We proceed then as follows: \( P^\text{opt}_s(f_1 \cup [n, n+c] f_2) \) equals 0 if \( s \in \text{Sat}(\neg f_1) \), and otherwise,

\[
P^\text{opt}_s(f_1 \cup [n, n+c] f_2) = \text{opt}_{a \in \text{Act}_s} \sum_{s' \in S} P(s, a, s') P^\text{opt}_{s'}(f_1 \cup [n-1, n-1+c] f_2).
\]

The above equation can be used to compute \( w_n(s) := P^\text{opt}_s(f_1 \cup [n, n+c] f_2) \) efficiently in an iterative manner. As for DTMCs, denote by \( P^d[-f_1] \) the matrix obtained from \( P^d \) by making \( (\neg f_1) \)-states absorbing. Then, \( w_0(s) = P^\text{opt}_s(f_1 \cup [0, c] f_2) \), \( w_1 = \text{opt}_{a \in \text{Act}_s}(m_1 \cdot P^d \cdot w_0) \) and \( w_{i+1} = \text{opt}_{a \in \text{Act}_s}(P^d[-f_1] \cdot w_i) \) (optimal in all elements) for \( i \geq 0 \), which is again referred to as the value iteration approach. Notice it agrees with the matrix vector multiplication for DTMCs (see (3)), which arise as special cases with \( |\text{Act}_s| = 1 \) for all \( s \in S \).

**Unbounded Until \( I = \mathbb{N} \).** We want to compute the probability \( P^\text{opt}_s(f_1 \cup \mathbb{N} f_2) \), which is defined to be optimal reachability probability over all schedulers (see (17)). Stationary schedulers are sufficient for this optimal value. However, a naive approach has exponential complexity by considering all stationary schedulers, as the number of them is exponential. It can be also solved by solving a linear programming problem, which has polynomial complexity. We illustrate how this works for the case \( P^\sup_s \).

As a preparation we define a zero set, as for DTMCs, by: \( S_{=0} = \{ s \in S \mid P^\sup_s(f_1 \cup S f_2) = 0 \} \). As for DTMCs, \( S_{=0} \) corresponds to the set of states which cannot reach \( \text{Sat}(f_2) \) in the underlying graph. Then, the vector \( y \) with \( y(s) = P^\sup_s(f_1 \cup S f_2) \) yields the unique solution of the following linear program:

- \( 0 \leq y(s) \leq 1 \) for all \( s \in S \),
- \( y(s) = 1 \) if \( s \in \text{Sat}(f_2) \), \( y(s) = 0 \) if \( s \in S_{=0} \),
• for all $s$ with $s \not\in \text{Sat}(f_2) \cup S_{=0}$, and for all actions $\alpha \in \text{Act}_s$:

$$y(s) \geq \sum_{s' \in S} P(s, \alpha, s') \cdot y(s')$$

(19)

where $\sum_{s \in S} y(s)$ is minimal. The probability $P_{\text{inf}}^s((f_1 \cup f_2)_1)$ can be computed in a similar way by using $\leq$ in (19) and maximizing $\sum_{s \in S} y(s)$.

Intervals of the form $[n, \infty)$ can now be handled directly: the probability $w_n(s) := P_{\text{opt}}^s((f_1 \cup [n, n+\infty))_1)$ can be obtained by essentially replacing the initial value $w_0$ (cf. (18)) by $w_0(s) := P_{\text{opt}}^s((f_1 \cup [0, \infty))_1)$.

**Theorem 5.1** The model checking problem for MDPs with respect to PCTL state formulas is decidable in polynomial time.

### 5.2. Model Construction

In the context of modeling concurrency, Markov decision processes appear in the form of probabilistic automaton (PA):

**Definition 5.2** A PA is a tuple $\mathcal{M} = (S, \text{Act}, \text{Steps}, \text{init})$ where $S$, init and Act are defined as for MDPs, and

- $\text{Steps} = \bigcup_{s \in S} \text{Steps}_s$ with $\text{Steps}_s \subseteq \text{Act}_s \times \text{Distr}(S)$ where Distr$(S)$ denote the set of distributions over $S$.

We write $s \xrightarrow{\alpha} \mu$ if $(\alpha, \mu) \in \text{Steps}_s$. MDPs can be considered as a special case in which actions are deterministic, i.e. $|\{\mu \mid s \xrightarrow{\alpha} \mu\}| = 1$. PA also conservatively extends LTS, obtained by restricting to point distributions.

**Handshaking.** The handshake communication based parallel synchronization can be extended to PAs directly. Let Syn be the set of an agreed synchronizing alphabet Syn. Let $\mathcal{M}_1 = (S_1, \text{Act}_1, \text{Steps}_1, \text{init}_1)$ and $\mathcal{M}_2 = (S_2, \text{Act}_2, \text{Steps}_2, \text{init}_2)$ be two PAs. We write $s \xrightarrow{\alpha} \delta_1 \mu$ for transitions in $\mathcal{M}_1$. The parallel composition $\mathcal{M}_1 \parallel_{\text{Syn}} \mathcal{M}_2 = (S_1 \times S_2, \text{Act}, \text{Steps}, \text{init}_1 \times \text{init}_2)$ is defined as follows. The decision vector Act is obtained by setting $\text{Act}_s = \langle \text{Act}_i \rangle_s$ if $s \in S_i$. For $\alpha \in \text{Syn}$, we have the handshaking (rendezvous) communication (see (10)):

$$\begin{align*}
\frac{s_1 \xrightarrow{\alpha_1} \mu \land s_2 \xrightarrow{\alpha_2} \mu'}{(s_1, s_2) \xrightarrow{\alpha} \mu \times \mu'}
\end{align*}$$

(20)

where $\mu \times \mu'$ defined by $(\mu \times \mu')(s, s') = \mu(s)\mu'(s')$. For every action $\alpha \in \text{Act}_i \setminus \text{Syn}$, we have the interleaving rule (see (11)):

$$\begin{align*}
\frac{s_1 \xrightarrow{\alpha} \mu}{(s_1, s_2) \xrightarrow{\alpha} \mu \times \delta_2} \quad \frac{s_2 \xrightarrow{\alpha} \mu}{(s_1, s_2) \xrightarrow{\alpha} \delta_1 \times \mu}
\end{align*}$$

(21)

This operator conservatively extends the handshaking operator for LTSs and the synchronous product for DTMCs. It is conservative in the sense that it encompasses the parallel composition for both LTSs and DTMCs.
5.3. Bisimulation for PAs

Definition 5.3 Let $M = (S, \text{Act}, \text{Steps}, \text{init})$ be a PA. A bisimulation on $M$ is an equivalence relation $R$ on $S$ such that for all $(s_1, s_2) \in R$, it holds that:

- $L(s_1) = L(s_2)$, and
- for all $s \xrightarrow{\alpha} \mu$, there exists $s_2 \xrightarrow{\alpha} \mu'$ such that $\mu(s_1, C) = \mu'(s_2, C)$ for all $C \in S/R$.

Two states $s_1, s_2$ are bisimilar, denoted by $s_1 \sim s_2$, if there exists a bisimulation relation containing $(s_1, s_2)$.

The bisimulation relation $\sim$ is an equivalence relation, and it conservatively extends bisimulation for both DTMCs and LTSs. The equivalence between bisimulation and PCTL equivalence for DTMCs does not hold in the PA setting: the reason is the lack of actions in the logic PCTL.

Bisimulation can be lifted to a relation between MDPs with single initial states. This is established by considering their direct sum, defined as follows. Let $M_1 = (S_1, \text{Act}_1, \text{Steps}_1, s_1)$ and $M_2 = (S_2, \text{Act}_2, \text{Steps}_2, s_2)$ be two PAs with initial states $s_1, s_2$ respectively, and $S_1 \cap S_2 = \emptyset$. Then, the direct sum of $M_1$ and $M_2$ is a PA

$$M_1 \oplus M_2 = (S_1 \cup S_2, \text{Act}, \text{Steps}_1 \cup \text{Steps}_2, s_1)$$

where Act as defined for the parallel composition. Then, we write $M_1 \sim M_2$ if and only if $s_1 \sim s_2$ in the direct sum. Now we can state the very important property of bisimulation, namely that bisimulation $\sim$ is a congruence with respect to parallel composition.

Theorem 5.2 Consider two PAs $M_1, M_2$ with single initial states. Then, $M_1 \sim M_2$ implies that $M_1 \parallel_{\text{Syn}} M \sim M_2 \parallel_{\text{Syn}} M$ for all PAs $M$ with single state.

6. Interactive Markov Chains

We now take up the same line of thoughts as we just have developed for MDPs, but now in continuous time, as opposed to discrete time. Recall that we have not been able to extend the handshaking composition to CTMCs. In this section we aim to overcome this problem, by introducing a model that extends CTMCs, but is closed under handshaking composition. The missing feature we need is again nondeterminism. This leads us to the model class of interactive Markov chains (IMC).

6.1. Model Construction for IMCs

CTMCs enrich DTMCs by specifying exponentially distributed sojourn time. The same can be done to extend MDP towards continuous-time Markov decision processes (CTMDP). CTMDP can be obtained by replacing the probability matrix $P^d$ in Definition 5.2 by a rate matrix $Q^d$.

While CTMDPs have lately been studied in the literature in the context of model checking, they are not compositional. To demonstrate why, we revisit the handshaking operator (see (20)) for PAs from a new angle. The operator for PAs can be considered...
as *fusing* the handshaking operator for LTSs and DTMCs, by removing the intermediate state. Intuitively, a transition \( s_1 \xrightarrow{\alpha} s'_1 \) in an \( \mathcal{M}_1 \) can be considered as *combining* a transition \( s_1 \xrightarrow{\alpha} s'_1 \) in a LTS and a transition \( s'_1 \xrightarrow{\mu} \) in a DTMC with \( \mu = \mathbf{P}(s'_1, \cdot) \). The synchronization for PAs in (20) is then obtained by *fusing* the synchronizing LTS transitions (see (10)) \( s_1 \xrightarrow{\alpha} s'_1 \) and \( s_2 \xrightarrow{\alpha} s'_2 \), and DTMCs transitions (see (12)) \( s'_1 \xrightarrow{\mu} \) and \( s'_2 \xrightarrow{\mu} \) respectively, and removing the intermediate state \( \langle s'_1, s'_2 \rangle \).

A similar construction, however, does not work for CTMDPs, since the only meaningful operation we have is to *interleave* the transitions and thus their distributions (We have discussed in Section 4.4 why this is meaningful!). Instead we now would need to fuse continuous distributions. This essentially means that parallel execution of \( s'_1 \) and \( s'_2 \) in two CTMCs needs to lead to an interleaving of CTMC transitions (encoding a so called *phase-type* distribution representing the maximum of two exponential distribution). This observation in turn leads the introduction of *interactive Markov chains* (IMC), which orthogonally combines LTSs and CTMCs:

**Definition 6.1** An IMC is a tuple \( \mathcal{M} = (S, \text{Act}, \text{IT}, \mathbf{Q}, \text{init}) \) where \( S \), \( \text{Act} \) and \( \text{init} \) are defined as for MDPs, \( \text{IT} = \bigcup_s \text{IT}_s \) with \( \text{IT}_s \subseteq \text{Act}_s \times S \) is a set of interactive transitions and \( \mathbf{Q} : S \times S \rightarrow \mathbb{R} \) is the rate matrix as defined for CTMCs.

We write \( s \xrightarrow{\alpha} s' \) if \( (\alpha, s') \in \text{IT}_s \), and \( s \xrightarrow{\lambda} s' \) if \( \mathbf{Q}(s, s') = \lambda > 0 \). As for CTMCs, we use \( E(s) := -\mathbf{Q}(s, s) = \sum_{s' \neq s} \mathbf{Q}(s, s') \) to denote the exit rate of \( s \). IMCs are orthogonally combined from LTSs and CTMCs, thus the handshaking rules for LTSs (see (11) and (10)) together with the interleaving rule for CTMCs (see (14)) form the building blocks for composing IMCs.

### 6.2. Bisimulation for IMCs

**Definition 6.2** Let \( \mathcal{M} = (S, \text{Act}, \text{IT}, \mathbf{Q}, \text{init}) \) be an IMC. A *bisimulation* on \( \mathcal{M} \) is an equivalence relation \( R \) on \( S \) such that for all \( (s_1, s_2) \in R \), it holds that:

- \( L(s_1) = L(s_2) \),
- for all \( s_1 \xrightarrow{\alpha} s'_1 \) there exists \( s_2 \xrightarrow{\alpha} s'_2 \) such that \( (s'_1, s'_2) \in R \),
- \( \mathbf{Q}(s_1, C) = \mathbf{Q}(s_2, C) \) for all \( C \in S_R \) with \( C \neq \{s_1\} \), and
- \( \mathbf{Q}(s_1, [s_1] \setminus \{s_1\}) = \mathbf{Q}(s_2, [s_2] \setminus \{s_2\}) \).

Two states \( s_1, s_2 \) are bisimilar, denoted by \( s_1 \sim s_2 \), if there exists a bisimulation relation containing \( (s_1, s_2) \).

The bisimulation relation \( \sim \) is an equivalence relation, and it conservatively extends bisimulation for CTMCs as well as LTSs. As for PAs, bisimulation can be lifted to a relation between IMCs with single initial states. Now we state the congruence property of bisimulation for IMCs:

**Theorem 6.1** Consider IMCs \( \mathcal{M}_1, \mathcal{M}_2 \) with single initial states. Then, \( \mathcal{M}_1 \parallel \text{Syn} \mathcal{M} \sim \mathcal{M}_2 \parallel \text{Syn} \mathcal{M} \) for all IMCs \( \mathcal{M} \) with a single initial state.

The congruence property holds for CTMCs as special sub-models of IMCs.
6.3. Model Checking IMCs

The set $S_I := \{s \in S \mid \exists s \xrightarrow{\alpha} s'\}$ is referred to as the set of interactive states, and $S_M := S \setminus S_I$ the set of Markovian states. The actions in the IMCs are used for compositional purposes, as seen in the previous subsections. For model checking, IMCs can be made closed, in the sense of turning the actions $\text{Act}_s$ of interactive states into internal actions which cannot be delayed. More precisely, for each interactive state $s$ with $s \xrightarrow{\alpha} s'$, we set $Q(s,u) = 0$ for all $u \in S$.

As for MDPs, the semantics relies on the notion of schedulers, which are however rather involved: a scheduler $\delta$ for a closed IMC is a function resolving the nondeterministic choice at interactive states $S_I$. More precisely, the scheduler $\delta$ chooses an action based on the complete history, including the time spent in each Markovian state and action chosen at each interactive state, until the present state. Especially, considering schedulers that exploit time information leads to more powerful schedulers than un-timed schedulers.

The logic CSL in section 3.2, without the steady state operator, is used to specify properties for IMCs. For each state $s$, the scheduler $\delta$ induces an unique probabilistic measure, denoted by $P^\delta_s$. The semantics for the probabilistic formula and the optimal probability $P^\delta_s(\phi)$ are essentially the same as for MDPs, see (16) and (17), by taking the appropriate sample paths and schedulers for closed IMCs respectively. As for MDPs, the model checking algorithm boils down to the computation of the optimal probabilities $P^\delta_s(f_1 \cup f_2)$. In case $I$ has rational endpoints, an approximate model checking algorithm has lately been proposed, based on discretization. The discretized model can be analyzed in a manner similar to MDP analysis. Details are skipped here, which can be found in [55, 69].

7. Bibliographic Remarks

We now put the material presented in this paper into the scientific context harvested by it. For a more detailed discussion about stochastic processes, Markov chains and their transient and steady state distributions, we refer to [49, 64].

The DTMC model for the IPv4 Zeroconf protocol is taken from [15]. The logic probabilistic CTL (PCTL) is a variation of the computational tree logic (CTL) [22]. The model checking algorithm for DTMCs is introduced by Hansson & Jonsson in [37], see also [9, 11] for further details. The PCTL model checking algorithm for DTMCs is implemented in, for example, PRISM [41] and MRMC [48], in which iterative numerical methods are used to approximate unbounded reachability and steady state probability efficiently.

The TMR case study for CTMCs is taken from [8]. The model checking algorithms for CTMCs are rooted in [3, 8]. The measurability of CSL path formulas can be found in [57]. The decidability results are due to Aziz et al. For bounded reachability probability, we have given the characterizations via Chapman-Kolmogorov equations. The bounded reachability probability can also be expressed equivalently via integral equation systems as proposed in [8]. The approximate algorithm for CTMCs is introduced in [8], in which numerical algorithms based on uniformization [35, 42] are used to approximate the time bounded reachability probability. The truncation of the infinite sum is based on Fox–Glynn algorithm [33]. The backward based computation was first suggested in [47].
Bisimulation for transition systems stems originally from Milner [54]. Larsen and Skou [51] have extended it to a quantitative variant of bisimulation for DTMCs. The logical characterization result for DTMCs is due to Aziz et al. [4]. They used the logic PCTL\textsuperscript{*}, which extends PCTL with LTL style of path formulas. In [28] a sublogic of PCTL with only conjunction and probabilistic next operator is shown to be sufficient for characterizing bisimulations. The bisimulation ~ can be determined efficiently in \(O(|D| \log |S|)\) [27, 65]. Moreover, efficient implementations are available with symbolic data structures [68]. Bisimulations for CTMCs are taken from [12]. As discussed, the bisimulation relations imply the notion of lumpability introduced in [49]. As shown in [26], the lumpability agrees with the notion of weak bisimulation considered in [12, 40]. The logical characterization of CSL for (infinite state) CTMCs is proposed in [29, 57]. The algorithm for deciding bisimulation for CTMCs has essentially the same complexity as DTMCs [27].

The model checking algorithm for PCTL – without long run operator – for MDPs is introduced by Bianco & de Alfaro [14], see [11] for further details. For simplicity, we have not considered the long run behavior, as in [14]. It has however been extensively studied for MDPs in the literature, see [59]. The scheduler class we considered is deterministic, but it can also be extended to allow randomization, namely as functions from \(N\) to distributions over the set of decision vectors, but this does not change the results and algorithms for the settings considered [14]. The reachability probability is reduced to linear programing, which has polynomial complexity [44], while in practice, the simplex algorithm [24] performs better, despite its theoretical exponential complexity. Most probabilistic model checkers, like PRISM or MRMC, employ the value iteration method [13] for approximating the optimal reachability probability.

Probabilistic automata and bisimulations for them are introduced in [61]. Different from DTMCs and CTMCs, the equivalence between bisimulation and PCTL equivalence does not hold right away for PAs. Therefore logics equipped with actions have been proposed for characterizing various bisimulation relations, see [39] for an in-depth discussion. Bisimulation for PAs can be decided in polynomial time [7, 21]. In practice, performing bisimulation minimization as a preprocessing step might not seem that attractive, since the value iteration based procedures are known to be very efficient. However, with Theorem 5.2 [61, 62], compositional minimization can be applied to attack the state space explosion problem: instead of minimizing the whole system, one can minimize components prior to parallel composition. This is for example used in the CADP toolset [34]. The congruence property for PAs holds also for DTMCs [66] and LTSs [54] as specific sub-models, for the operators defined on them. Handshaking based synchronization for LTSs is introduced in [16], and the semantics via a set of structural operational semantic rules is due to Plotkin [58].

CTMDPs with a finite or infinite planning horizon have been studied in [52, 53], and recently in [20]. For model checking algorithms for CTMDPs we refer to [10, 19, 56, 60]. IMCs and bisimulations for them are introduced in [38]. IMCs enjoy the congruence property. As CTMs form a submodel of IMCs, the congruence holds also for CTMCs. Algorithms for deciding bisimulation for IMCs have polynomial time complexity [38]. A discrete-time version of IMC, interactive probabilistic chain (IPC) has been proposed in [23], and can be considered as decomposing a transition \(s \xrightarrow{\alpha} \mu\) in a PA into two transitions \(s \xrightarrow{\alpha} s'\) as in LTSs and \(s' \xrightarrow{\mu}\) as in DTMCs. The model checking algorithm for IMCs in [55, 69] exploits IPCs as the discretized model.
8. Conclusions

The paper has reviewed model construction and model checking approaches for discrete and continuous Markov models. Its greater historic and pragmatic context is described in a recent review article [9].

Apart from the principle models discussed here, various other probabilistic models have been considered in the literature. Reactive, generative and stratified probabilistic models are considered in [66]. Reactive models coincide with MDPs, while generative models are DTMCs with actions labels over the probabilistic branchings instead of state labels (with atomic propositions). Stratified models add more information to generative models such that one can have level-wise probabilistic branching. An action- and state-labeled version of CTMCs is considered in [6], where the rate matrix of a CTMC is decorated with an additional set of actions. In [36, 67], probabilistic systems are considered where states are partitioned into probabilistic states and nondeterministic states, thus called alternating models. A probabilistic state enables a successor distribution, while from a nondeterministic state ordinary transitions may occur, as in labeled transition systems. Interactive probabilistic chains [23] extend alternating models with compositional theories. PAs with continuous state space have also been extensively studied, see [25, 57]. Recently, Markov automata [30, 31] have been proposed, which are essentially the orthogonal extension of PAs and CTMCs. Notably, Markov automata can be constructed by combining the handshaking rules for PAs in (20) and (21), together with the interleaving rule for CTMCs in (14).

In [63] and [32], interval-valued DTMCs (IDTMCs) are discussed, which are DTMCs in which transitions may be decorated with intervals instead of unique probabilities. These Markov chains are motivated via statistical experiments, and can thus be used to model statistical variations. In IDTMCs, both a lower and upper bound of the reachability probability can be obtained [32]. Properties expressed in probabilistic computational tree logic can be analyzed, via a 3-valued PCTL semantics. Then, if the property is satisfied or refuted in the IDTMC, this is also the case in the original system. However, if we cannot conclude whether the property holds or not, refinement steps are necessary. This approach was successfully extended to CTMCs [45] and IMCs [46] with rate intervals.

Other related models include IMCs with input/output [17, 18], Markov chains with costs and rewards [2, 5], and extensions of Markov models with continuous dynamics, for instance probabilistic timed automata [43, 50], and probabilistic hybrid systems [1].

Acknowledgements

The authors are grateful to Hernán Baró Graf, Ernst Moritz Hahn, Hassan Hatefardakani, David Spieler (all Saarland University) and Michael J. A. Smith (Technical University of Denmark) for careful proofreading a preliminary version of this paper. Holger Hermanns is supported by the DFG as part of the Transregional Collaborative Research Centre SFB/TR 14 AVACS, by the European Community’s Seventh Framework Programme under grant agreement no 214755 QUASIMODO, and by the NWO-DFG bilateral project ROCKS. The work of Lijun Zhang has received partial support from MT-LAB, a VKR Centre of Excellence. This paper is the result of an invitation of the academic directors of the 2010 Summer school Marktoberdorf, Manfred Broy, David Harel, and Tony Hoare, to present a lecture series at the school.
References


