Quantitative Analysis of Communication Scenarios*

Author's Version – September 19, 2017

Clemens Dubslaff and Christel Baier

Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany
{clemens.dubslaff,christel.baier}@tu-dresden.de

Abstract. Message sequence charts (MSCs) and their higher-order formalism in terms of message sequence graphs (MSGs) provide an intuitive way to describe communication scenarios. Naturally, quantitative aspects such as the probability of failure, maximal latency or the expected energy consumption play a crucial role for communication systems. In this paper, we introduce quantitative MSGs with costs or rewards and stochastic timing information in terms of rates. To perform a quantitative analysis, we propose a branching-time semantics for MSGs as possibly infinite continuous-time Markov chains (CTMCs) interpreting delayed choice on the partial-order semantics of MSGs. Whereas for locally synchronized MSGs a finite-state bisimulation quotient can be found and thus, standard algorithms for CTMCs can be applied, this is not the case in general. However, using a truncation-based approach we show how approximative solutions can be obtained. Our implementation shows feasibility of this approach exploiting reliability, resilience, and energy consumption.

1 Introduction

Nowadays, communication protocols have to face many non-functional requirements to be considered during their design process. For instance, the next-generation wireless communication standard 5G will rely on time-bounded requirements concerning latency, energy consumption, and outage probability [28]. To avoid costly and timely redesign steps, these requirements should be considered already in early design phases [32]. An intuitive formalism to describe communication systems widely used in early stages of development is provided by *message sequence charts (MSCs)*, standardized by the ITU [22]. Figure 1 shows an example MSC where a sender s sends two data packages that are in turn acknowledged by a receiver r. The partial-order semantics for MSCs arises from the time-line orderings of each process and the condition that each send event must precede the corresponding receive event.

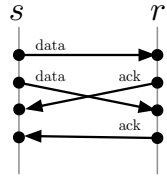

Fig. 1. An MSC

* The authors are supported by Deutsche Telekom Stiftung, the DFG through the collaborative research centre HAEC (SFB 912), the Excellence Initiative by the German Federal and State Governments (cluster of excellence cfAED and Institutional Strategy), the Graduiertenkolleg QuantLA (1763), the 5G Lab Germany, and the DFG/NWO-project ROCKS, and the EU-FP-7 grant MEALS (295261).

In order to specify collections of MSCs, the standard higher-order formalism is given in terms of *high-level MSCs* [22]. In this paper, we deal with *message sequence graphs (MSGs)*, the simplest form of high-level MSCs that are, however, expressively equivalent to high-level MSCs [5]. MSGs can be seen as finite automata over MSCs where the collections of scenarios arise from composing MSCs along accepting paths. The composition of MSCs is performed by glueing the time lines of the processes together and possibly matching unmatched send events and corresponding receive events [15,14]. The verification of MSGs and linear-time requirements has been extensively studied (cf. surveys [30,23]), but most verification problems turn out to be undecidable. However, for MSGs where processes locally synchronize, the execution language can be represented by a finite-state automaton, which renders many linear-time verification problems decidable for this class of MSGs. Also when directly reasoning about the graph structure of the MSCs defined by an MSG, decidability results can be established [25,26,30].

This paper aims to establish a probabilistic model-checking approach for the quantitative analysis of MSGs with annotated costs and stochastic timing information in terms of rates. In order to apply probabilistic model-checking techniques, a branching-time view on (quantitative) MSGs is required, where choices are resolved probabilistically. In the ITU standard [22,21], an interleaving branching-time semantics was specified which relies on *delayed choice*, where any resolution of a choice between the communication scenarios specified by the MSG is avoided until it is inevitable:

> "In the case where alternative MSC sections have common preamble, the choice of which MSC section will be executed is performed after the execution of the common preamble."

Using process algebra, the concept of delayed choice was specified by an intrinsic linear-time operator $\mp$ [27] (cf. the rules **(DC1)** and **(DC2)** below).

$$\frac{x \xrightarrow{\alpha} x' \ \wedge \ y \not\xrightarrow{\alpha}}{(x \mp y) \xrightarrow{\alpha} x'} \textbf{(DC1)} \qquad \frac{x \xrightarrow{\alpha} x' \ \wedge \ y \xrightarrow{\alpha} y'}{(x \mp y) \xrightarrow{\alpha} (x' \mp y')} \textbf{(DC2)}$$

For formal reasoning about MSGs, the process-algebraic semantics proposed in the standard is, however, difficult to use since it includes all facets of the modeling formalism and an algorithmic analysis turns out to be very challenging [34].

Our Contribution. First, we propose a branching-time semantics for MSGs in terms of transition systems by interpreting delayed choice directly on the partial-order semantics of MSGs. With the partial-order semantics of MSGs as an intermediate step, we obtain a uniform approach to support many variants of MSGs that have been proposed in the literature, such as causal MSGs [13] (required, e.g., to specify TCP/IP protocol scenarios) and compositional MSGs [15,9,26] (required, e.g., to specify scenarios of the alternating-bit or stop-and-wait protocol [36]). As an MSG may exhibit infinite communication scenarios, our transition-system semantics cannot be fully constructed in general. However,

we show that the transition system of an MSG can be constructed on-the-fly, crucial for an algorithmic analysis.

Second, according to the (hardware and software) characteristics of the communication system modeled, we annotate communication actions by costs and rates. The arising *quantitative MSG* constitutes a *continuous-time Markov chain (CTMC) semantics*, where the underlying transition system coincides with our branching-time semantics for (non-quantitative) MSGs. The CTMC semantics enables for a quantitative analysis using probabilistic model-checking techniques to reason, e.g., about the communication system's resilience, expected energy costs and the probability of failure.

Third, we investigate applicability of algorithmic analysis on the transition-system and CTMC semantics. Given the undecidability results for linear-time properties and the fact that the transition system of MSGs can be infinite, the undecidability of even basic reachability properties is as expected. However, when the MSG is locally synchronized, its transition system (or CTMC, respectively) has a finite bisimulation index. Then, a bisimilar finite-state transition system (CTMC, respectively) can be constructed and analyzed using standard (probabilistic) model-checking algorithms. For general (not necessarily locally synchronized) MSGs, we show that the probabilities for time-bounded reachability properties are approximable up to an arbitrary precision by constructing the CTMC on-the-fly up to a sufficient depth.

Fourth, we implemented the construction of the CTMC semantics for quantitative MSGs up to a given truncation bound and applied approximative methods to reason about performance measures. This implementation is used to carry out a case study investigating non-functional properties of scenarios from the USB 1.1 protocol and two variants of a classical stop-and-wait protocol. State aggregation with respect to bisimulation equivalence (a.k.a. lumpability for CTMCs) based on the structure of the MSGs allows to establish results within a reasonable time, whereas without such bisimulation techniques we run into timeouts.

Related Work. Several branching-time semantics have been proposed for subclasses of MSGs [25,17,10], however not focusing on delayed choice and without the support for compositional MSGs. Also the automata constructions to describe the execution language of locally synchronized MSGs [29,5,9,13] define inherently a branching-time semantics. These semantics are directly defined on a fixed communication architecture and have to be redefined when the underlying partial-order semantics changes [4].

Within *time-constrained MSGs*, time intervals are annotated to pairs of actions to specify the time horizon in which the second action of the pair has to be performed after the first one [24,1,2]. Similar as within locally synchronized MSGs in the untimed case, subclasses of time-constrained MSGs exhibit a timed automata which accepts the language of timed words satisfying the constraints imposed on action pairs. Also in this paper we annotate timing information to actions of MSGs. However, our approach is orthogonal, as the timing information on actions is modeled stochastically. A simulation-based stochastic analysis of MSCs with annotated rates has been undertaken by [37] using the

trace-generation engine of the tool MÖBIUS [11]. However, a formal framework for a CTMC semantics of MSGs was not specified in [37]. Finite-state discrete-time Markov chains synthesized from from the local views of each process in an MSG annotated with reliability probabilities has been presented by [35]. Their approach disregards the global control of MSGs and could thus be applied for locally synchronized MSGs which are implementable [3].

Probabilistic model checking of infinite-state CTMCs is challenging already independent from the context of MSGs. Model-checking algorithms for highly structured infinite CTMCs (including Jackson queuing networks and quasi-birth-death processes) and properties in *continuous stochastic logic (CSL)* [7] have been presented by Remke et al. (see, e.g., [33]). In order to approximate properties of infinite CTMCs arising from biological systems, [18] presented on-the-fly analysis techniques. Similarly, [16] established approximation methods for arbitrarily structured infinite CTMCs with possible unbounded rates and time-bounded CSL properties.

Outline. We recall basics on modeling communication systems, including MSCs and MSGs in Section 2. Section 3 is advocated to our branching-time semantics for MSGs, its consistency within the standard, and related model-checking problems. Quantitative MSGs and their properties are introduced in Section 4. We illustrate our approach towards approximative performance measures by a case study within our implementation in Section 5 and conclude with Section 6.

2 Preliminaries

We denote by A^∞ ($A^\star$, A^+, A^ω) the set of all (finite, non-empty finite, infinite) *words* over an alphabet A. By w_i we denote the $(i{+}1)$-th symbol of a word w and by $|w|$ the length of w ($|w| = \infty$ if $w \in A^\omega$). The *concatenation* of languages, i.e., subsets of A^∞, is defined as the union of element-wise concatenation of their words. *Atomic actions* are tasks of some process of $P = \{p_1, p_1, ..., p_k\}$ indivisible on the abstraction level of the model and collected in $\Sigma = \bigcup_{p\in P} \Sigma_p$. Here, $\Sigma_p = \{p\}\times(\{!,?\}\times P\setminus\{p\}\times\Gamma\cup\Lambda)$, where Γ and Λ are sets of message and local action labels, respectively. The action $p!q(m) \in \Sigma$ stands for sending a message m from process p to process q, $p?q(m) \in \Sigma$ for p receiving m from q, and $p(a) \in \Sigma$ for a local action a. *Events* are occurrences of atomic actions collected in a set E and assigned to actions through a labeling function $\lambda\colon E \to \Sigma$. We partition E into the set of send events $E_!$, receive events $E_?$, and local events E_l. By E_p we denote the set of events of process p, i.e., $E_p = \{e \in E\colon \lambda(e) \in \Sigma_p\}$. A *(labeled) partial order* is a tuple $\mathcal{P} = (E, \leq, \lambda)$, where $\leq$ is an asymmetric, reflexive and transitive binary relation over E. Isomorphic labeled partial orders are identified. $\mathcal{P}$ is called *total* if for all $e, e' \in E$ with $e \neq e'$ we have either $e < e'$ or $e > e'$. A linearization of $\mathcal{P}$ is a word $w \in \Sigma^\infty$ where there is a bijection $f\colon N \to E$ with $N \subseteq \mathbb{N}$ such that for all $i, j \in N$: $\lambda(f(i)) = w_i$ and if $i < j$ then $f(i) \not\geq f(j)$. We define the *language of* $\mathcal{P}$ as the set of linearizations $\mathrm{L}(\mathcal{P}) \subseteq \Sigma^\infty$. The *downward closure* $\downarrow F$ on sets of events F is defined as $\{e \in E\colon \exists f \in F. e \leq f\}$. We call $\mathcal{P}$ *prefinite* if $\downarrow e = \downarrow\{e\}$ is finite for all $e \in E$. Besides $\mathcal{P}$, let $\mathcal{P}' = (E', \leq', \lambda')$ be

another partial order. If E' is a finite downward-closed subset of E, $\leq' = \leq \cap (E \times E')$ and $\lambda' = \lambda|_{E'}$, then $\mathcal{P}'$ is called a *prefix* of $\mathcal{P}$. The set of all prefixes of $\mathcal{P}$ is denoted by $\mathrm{Pref}(\mathcal{P})$. Concurrent systems are modeled by sets $\mathcal{F}$ of partial orders, called *partial-order families*. Notations for partial orders defined above extend to partial-order families as expected, e.g., the set of all prefixes of $\mathcal{F}$ is $\mathrm{Pref}(\mathcal{F}) = \{\mathcal{P}' \colon \exists \mathcal{P} \in \mathcal{F}.\mathcal{P}' \in \mathrm{Pref}(\mathcal{P})\}$.

2.1 Branching-time Models and Quantitative Annotations

A *transition system (TS)* is a tuple $(S, s_0, \longrightarrow)$, where S is a countable set of states, $s_0 \in S$ is an initial state, and $\longrightarrow \subseteq S \times \Sigma \times S$ is a transition relation. We denote by $En(s)$ the set of enabled actions in s, i.e., the set of all $\alpha \in \Sigma$ where there is an s' with $s \xrightarrow{\alpha} s'$. $\mathcal{T}$ is *deterministic* if for all $s \xrightarrow{\alpha} s'$, $s \xrightarrow{\alpha} s''$ follows $s' = s''$. Likewise, an *execution* is a sequence $\eta = s_0\alpha_0 s_1 \alpha_1 \ldots \in S \times (\Sigma \times S)^\infty$, where $s_i \xrightarrow{\alpha} s_{i+1}$ for all $i < |\eta| - 1$. A *path* is an execution where transition actions are omitted. If $\mathcal{T}$ is deterministic, we identify executions with their projections onto actions and define the language $\mathrm{L}(\mathcal{T}) \subseteq \Sigma^\infty$ as the set of maximal executions. When $C \subseteq S$, we denote by $\Diamond C$ the set of all finite paths π such that $s_{|\pi|-1} \in C$ and $s_i \notin C$ for all $i < |\pi| - 1$.

Quantitative Annotations. A *continuous-time Markov chain (CTMC)* [7] is a tuple $\mathcal{C} = (\mathcal{T}, \mathbf{R}, \mathbf{C})$, where $\mathcal{T} = (S, s_0, \longrightarrow)$ is a transition system, and $\mathbf{R} \colon S \times \Sigma \times S \to \mathbb{Q}_{\geq 0}$ and $\mathbf{C} \colon S \cup S \times \Sigma \times S \to \mathbb{Q}_{\geq 0}$ assign *rates* and *costs*, respectively. For a state $s \in S$ and a set of states $C \subseteq S$, we denote by $\mathbf{Q}(s, s') = \sum_{\alpha \in \Sigma} \mathbf{Q}(s, \alpha, s')$ and $\mathbf{Q}(s, C) = \sum_{s' \in C} \mathbf{Q}(s, s')$, where $\mathbf{Q}$ is either $\mathbf{R}$ or $\mathbf{C}$. We assume that the *exit rate* $\mathbf{E}(s) = \mathbf{R}(s, S)$ converges for all $s \in S$. According to the rates, a race between the outgoing transitions from state s exists, where the probability of state s' winning the race is $\mathbf{P}(s, s') = \mathbf{R}(s, s')/\mathbf{E}(s)$. Paths in $\mathcal{T}$ annotated with exit time points are *timed paths*, i.e., sequences of the form $\vartheta = s_0 t_0 s_1 t_1 \ldots \in S \times (\mathbb{R}_{>0} \times S)^\infty$. The *accumulated costs* along ϑ is defined as

$$\mathbf{C}(\vartheta) \quad = \quad \sum_{i < |\pi| - 1} \big(\mathbf{C}(s_i) \cdot t_i + \mathbf{C}(s_i, s_{i+1}) \big).$$

For $s, s' \in S$, and $u, v \in \mathbb{R}_{\geq 0}$ with $u \leqslant v$, the probability of moving from s to s' after at least time t and at most time T is given by transition probabilities

$$\mathbf{P}(s, [t, T], s') \quad = \quad \mathbf{P}(s, s') \cdot \left(e^{-\mathbf{E}(s)t} - e^{-\mathbf{E}(s)T} \right)$$

We write $\Theta = s_0 I_0 s_1 I_1 \ldots I_{n-1} s_n$ for the set of all timed paths $s_0 t_0 s_1 \ldots t_{n-1} s_n \ldots$ with $t_i \in I_i$ and refer to Θ as a *trajectory*. As usual, we define a probability measure $\Pr_s(\Theta)$ for a state s as product of the transition probabilities for the intervals I_i, $i = 0, ..., n-1$ [7]. *Expected accumulated costs* $\mathrm{Ex}(\Diamond^{\leq t} C)$ are defined as the expectation of the random variable which assigns to some timed path the minimal accumulated costs of either reaching $C \subseteq S$ or the time bound $t \in \mathbb{Q}_{>0}$.

2.2 Scenarios Specifying Communication Systems

To model communication systems, we focus on *message sequence charts (MSCs)* with lost and found messages [22] (also known as *compositional MSCs*) and their

higher-order formalism in terms of *message sequence graphs (MSGs)*, defining possibly infinite collections of MSCs [15,14].

Definition 1 (MSC). *An* MSC *is a prefinite partial order* $\mathcal{M} = (E, \leq, \lambda, \mu)$*, equipped with an injective function* $\mu\colon M \to E_?$ *matching corresponding send events* $M \subseteq E_!$ *and receive events in* $E_?$ *such that* $\leq_\mu = \{(e, \mu(e))\colon e \in M\}$ *is contained in* $\leq$ *and for all* $e \in M$ *with* $\lambda(e) = p!q(m)$ *we have* $\lambda(\mu(e)) = q?p(m)$*. Furthermore, we require* $\leq|_{E_p}$ *to be total for all* $p \in P$*.*

An event not contained in M, $\mu(M)$, or E_l is called *unmatched*. $\mathcal{M}$ is called *basic* if λ is injective, *left-closed* if there are no unmatched receive events, and *closed* if $\mathcal{M}$ is left-closed and there are no unmatched send events. We denote by $\mathbb{MSC}$ the set of MSCs and by $\mathrm{b}\mathbb{MSC}$ the set of basic MSCs. If any confusion is excluded, we abbreviate the empty MSC by ε and the MSC containing only one event labeled by $\alpha \in \Sigma$ by α.

MSC Families, Systems and Graphs. MSCs are composed by "glueing" their process lines together, and possibly matching send and receive events [14,21]. Formally, let $\mathcal{M}_i = (E_i, \leq_i, \lambda_i, \mu_i)$ for $i = 1, 2$ be two disjoint MSCs and let M_i as in Definition 1. The *composition* $\mathcal{M}_1 \cdot \mathcal{M}_2$ is the family of all MSCs $\mathcal{M} = (E, \leq, \lambda, \mu)$ with $E = E_1 \cup E_2$, where (1) $\mathcal{M}|_{E_i} = \mathcal{M}_i$ for $i = 1, 2$ and (2) for all $e \in E_2, e' \in E$ holds $e \leq e' \Rightarrow e' \in E_2$. The latter condition (2) implies that send events in $\mathcal{M}_2$ are not matched with receive events in $\mathcal{M}_1$ and that the events of each process in $\mathcal{M}_1$ are all lower than the events of the same process in $\mathcal{M}_2$. Collections of MSCs are composed through the union of element-wise composition of the MSCs in the families, i.e.,

$$\mathcal{F}_1 \cdot \mathcal{F}_2 \;=\; \{\mathcal{M} \in \mathcal{M}_1 \cdot \mathcal{M}_2 \colon \mathcal{M}_1 \in \mathcal{F}_1, \mathcal{M}_2 \in \mathcal{F}_2\}.$$

We define the composition along a word over basic MSCs $\pi \in \mathrm{b}\mathbb{MSC}^\infty$ by $\mathcal{F}_0 = \{\pi_0\}$ and $\mathcal{F}_i = \mathcal{F}_{i-1} \cdot \pi_i$ for each $0 < i < |\pi|$. This naturally extends to languages $\mathcal{L} \subseteq \mathrm{b}\mathbb{MSC}^\infty$ as the union of all MSC families arising from compositions along $\pi \in \mathcal{L}$. To ease notations, we identify with $\pi \in \mathrm{b}\mathbb{MSC}^\infty$ ($\mathcal{L} \subseteq \mathrm{b}\mathbb{MSC}^\infty$, respectively) the MSC family arising by composition along π ($\mathcal{L}$, respectively). If $\mathcal{L}$ is left-closed, we call $\mathcal{L}$ an *MSC system*. Regular MSC systems are usually provided by *message sequence graphs (MSGs)* [22], which are in essential finite automata over basic MSCs.

Definition 2 (MSG). *An* MSG *is a tuple* $\mathcal{G} = (N, n_0, \to)$*, where* N *is a set of nodes,* $n_0 \in N$ *an initial node, and* $\to \subseteq N \times \mathrm{b}\mathbb{MSC} \times N$ *a transition relation.*

We may employ the same terminology as for transition systems. An MSC $\mathcal{M}$ is accepted by $\mathcal{G}$ if there is a composition along a maximal execution containing $\mathcal{M}$. The MSC system containing all left-closed MSCs accepted by $\mathcal{G}$ is denoted by $\mathcal{S}(\mathcal{G})$. When not stated differently, we assume an MSG to be *safe*, i.e., every accepting execution contains a left-closed MSC (which can be decided [23]).

Example 1. The MSG $\mathcal{G}_{\text{usb}}$ depicted in Figure 2, which is taken from [14], formalizes scenarios of transactions according to the USB 1.1 protocol. The MSG

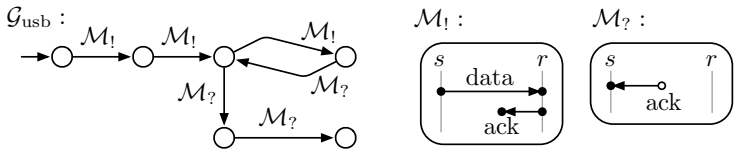

Fig. 2. Scenarios of USB 1.1 transactions by the MSG $\mathcal{G}_{\text{usb}}$

is shown on the left, the assigned MSC labels on the right. A sender s continuously sends data to a receiver r, which returns an acknowledgement directly after it received the data package. However, the sender has some advance by 2-3 messages. The smallest scenario described by $\mathcal{G}_{\text{usb}}$ is the only left-closed MSC contained in the execution $\mathcal{M}_!\mathcal{M}_!\mathcal{M}_?\mathcal{M}_?$ shown in Figure 1.

Locally Synchronized MSGs. For MSGs over closed basic MSCs, [5] and [29] independently stated a syntactic criterion such that a finite automaton accepting the linearizations of the partial-order semantics can be constructed. We deal here with a similar criterion based on [15,14]. For an MSC $\mathcal{M} = (E, \leq, \lambda, \mu)$, the *communication graph* $H_{\mathcal{M}}$ is defined as the directed graph $(P, \to)$, where $p \to q$ iff there exists an event e in $\mathcal{M}$ with $\lambda(e) = p!q(m)$ for some $m \in \Gamma$ or $p = q$ and $\lambda(e) = p(a)$ for some $a \in \Lambda$. The *deficit* $D_{\mathcal{M}} \colon P \times P \times \Lambda \to \mathbb{Z}$ is defined as the number of messages in the communication channels after executing $\mathcal{M}$:

$$D_{\mathcal{M}}(p, q, m) \;=\; |\{e \in E \colon \lambda(e) = p!q(m)\}| - |\{e \in E \colon \lambda(e) = q?p(m)\}|$$

If for every $\mathcal{M}$ contained in a composition along cycles in an MSG $\mathcal{G}$ the communication graph $H_{\mathcal{M}}$ consists of a single non-trivial strongly connected component and $D_{\mathcal{M}}(p, q, m) = 0$ for all $p, q \in P$, $m \in \Lambda$, then $\mathcal{G}$ is called *locally synchronized*. Note that the MSG in Example 1 is locally synchronized.

3 Branching-Time Semantics

The semantics of an MSG with quantitative annotations in form of rates and costs for actions will be given as an infinite-state continuous-time Markov chain. To define this CTMC, we first provide a (non-probabilistic) transition-system semantics for MSGs that is compatible with the ITU standard and the delayed choice interpretations of branchings [21,22]. The transition system will serve as basis for the CTMC semantics, but might be also useful for other (non-stochastic) purposes as well. First, we transfer the concept of delayed choice to the setting of an MSC system $\mathcal{S}$. The set of communication scenarios in $\mathcal{S}$ according to which a common preamble[1] $w \in \Sigma^\star$ could have been executed is

$$\mathcal{S}(w) \;=\; \{\mathcal{P} \in \mathcal{S} \colon w \in \mathrm{Pref}(\mathrm{L}(\mathcal{P}))\}.$$

[1] Check for the definition of delayed choice provided in the introduction.

Hence, interpreting delayed choice for $\mathcal{S}$ reveals that during the execution of w, no choice between the scenarios in $\mathcal{S}(w)$ is made. Furthermore, any reordering $\hat{w}$ of w where $\mathcal{S}(w) = \mathcal{S}(\hat{w})$ yields the same global state of the system. Such reorderings are exactly executions of partial-order prefixes of $\mathcal{S}(w)$ containing only events executed by w and thus can be represented by *w-configurations*

$$\mathrm{Conf}(\mathcal{S}, w) = \{\mathcal{P} \in \mathrm{Pref}(\mathcal{S}) \colon w \in \mathrm{L}(\mathcal{P})\}.$$

The set of all configurations in $\mathcal{S}$ is then defined as

$$\mathrm{Conf}(\mathcal{S}) = \textstyle\bigcup_{w \in \mathrm{Pref}(\mathrm{L}(\mathcal{S}))} \mathrm{Conf}(\mathcal{S}, w).$$

This yields an operational transition-system semantics for MSC systems $\mathcal{S}$ as

$$\mathcal{T}_{\mathcal{S}} \;=\; (\mathrm{Conf}(\mathcal{S}), \{\varepsilon\}, \longrightarrow),$$

where $\longrightarrow$ is defined through $\mathrm{Conf}(\mathcal{S}, w) \xrightarrow{\alpha} \mathrm{Conf}(\mathcal{S}, w\alpha)$ for any $w \in \Sigma^\star$ with $\mathrm{Conf}(\mathcal{S}, w\alpha) \neq \emptyset$. Recall that ε stands for the empty MSC. Due to delayed choice, $\mathcal{T}_{\mathcal{S}}$ is deterministic. As the *level* $\ell v \ell(\mathcal{K}) = \{|\mathcal{P}| \colon \mathcal{P} \in \mathcal{K}\}$ of configurations $\mathcal{K}$ along paths in $\mathcal{T}_{\mathcal{S}}$ increase within each transition, $\mathcal{T}_{\mathcal{S}}$ is also acyclic. Furthermore, $\mathcal{T}_{\mathcal{S}}$ is infinite if $\mathcal{S}$ contains an infinite MSC, such that it cannot be fully constructed in general, e.g., $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ is infinite for an MSG $\mathcal{G}$ containing loops. This transition-system semantics is language consistent with the partial-order semantics in the sense that prefixes of executions agree with the one of $\mathcal{S}(\mathcal{G})$:

Proposition 1. *When $\mathcal{S}$ is an MSC system,* $\mathrm{Pref}(\mathrm{L}(\mathcal{S})) = \mathrm{Pref}(\mathrm{L}(\mathcal{T}_{\mathcal{S}}))$.

Example 2. Let us consider a simple stop-and-wait protocol [36,37] described through the MSG $\mathcal{G}_{\mathrm{stoc}}$ depicted on the left in Figure 3. A sender s sends a data package to a receiver r over an unreliable channel. The receiver provides an acknowledgement "ack" if the data has been transmitted successfully and a negative acknowledgement "nak" if not. A message is considered to be lost after a timeout "to" has been risen on the sender's side. Note that this MSG is neither locally synchronized nor realizable in the sense of [3]. In the center, a part of the infinite-state transition system $\mathcal{T}_{\mathcal{S}(\mathcal{G}_{\mathrm{stoc}})}$ is shown (event labels are abbreviated, e.g., "s!r(data)" by "!d"). Due to delayed choice, the triangular states do not agree: The state reached by !d·to·?d·!d surely matches the first send and receive event. Differently, the filled triangle state reachable through !d·to·!d·?d could match ?d with either one of the both !d events and thus, its configuration has two elements. For further illustrating the impact of delayed choice, we depicted the configurations after an action !n on the right – dashed arrows indicate possible matches of events (i.e., which yield several elements in the configuration) and solid ones indicate sure matchings.

3.1 Model Checking Branching-Time Requirements

Our transition-system semantics for MSGs enables to reason about branching-time properties. However, even basic reachability problems are undecidable:

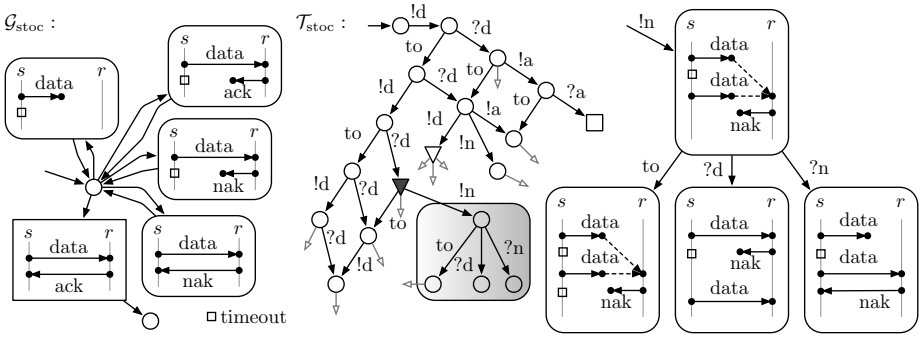

Fig. 3. Stop-and-wait protocol scenarios by the MSG $\mathcal{G}_{\text{stoc}}$ and its TS semantics

Theorem 1. *The* action-enabled reachability problem *for an MSG $\mathcal{G}$ and $A \subseteq \Sigma$, asking whether there is a state $\mathcal{K}$ reachable in $\mathcal{T}_{\mathcal{G}}$ with $En(\mathcal{K}) = A$, is undecidable.*

The proof of this theorem relies on a reduction from the Post's Correspondence Problem as presented in [30] for the linear-time setting and matchings of executions due to delayed choice. Clearly, Theorem 1 renders model checking MSGs against branching-time logics such as ACTL [12] undecidable. However, when $\mathcal{G}$ is locally synchronized, a finite transition system bisimilar to $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ can be constructed, which enables to apply standard algorithms to reason about branching-time requirements on $\mathcal{G}$. To see this, we use a similar construction as presented in [29], where a finite-state automaton $\mathcal{A}_{\mathcal{G}}$ has been constructed which accepts the executions $\mathrm{L}(\mathcal{S}(\mathcal{G}))$ of $\mathcal{G}$. Performing a determinization on $\mathcal{A}_{\mathcal{G}}$ yields a finite transition system, which is bisimilar to $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ since trace equivalence coincides with bisimulation equivalence for deterministic systems [8].

Theorem 2. *For a locally synchronized MSG $\mathcal{G}$, $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ has a finite bisimulation quotient which can be effectively constructed.*

Note that $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ having a finite bisimulation quotient does not necessarily imply that $\mathcal{G}$ is locally synchronized.

4 Quantitative Message Sequence Graphs

We now introduce MSGs with quantitative annotations and a suitable CTMC semantics based on the transition-system semantics of the last section. As we consider events in an MSC as instances of actions, i.e., tasks indivisible on the abstraction level, it is rather natural to extend this interpretation to quantitative annotations on MSC systems. That is, quantitative annotations are mappings from actions to rational values. We focus here on annotation structures we call *profiles*, formalizing the specification of different (hardware or software) setups.

Definition 3. *A* profile *is a tuple $\mathcal{I} = (\mathbf{r}, \mathbf{c}, \sigma)$, where $\mathbf{r} \colon \Sigma \to \mathbb{Q}_{>0}$ assigns rates, $\mathbf{c} \colon \Sigma \to \mathbb{Q}_{\geq 0}$ costs and $\sigma \in \mathbb{Q}_{>0}$ specifies stationary costs.*

The rates $\mathbf{r}$ contain stochastic timing information about the frequency actions are executed. Costs are given by $\mathbf{c}$, formalizing the costs of executing an action, and by σ, formalizing the stationary costs required keep the system operational for one time unit. We call an MSG $\mathcal{G}$ amended with a profile $\mathcal{I}$ *quantitative MSG*.

Definition 4. *Let $\mathcal{G}$ be an MSG with $\mathcal{T}_{\mathcal{S}(\mathcal{G})} = (S, s_0, \longrightarrow)$ and $\mathcal{I} = (\mathbf{r}, \mathbf{c}, \sigma)$ a profile. The CTMC semantics $\mathcal{C}_{\mathcal{G}} = (\mathcal{T}_{\mathcal{S}(\mathcal{G})}, \mathbf{R}, \mathbf{C})$ of the quantitative MSG $(\mathcal{G}, \mathcal{I})$ is defined as follows: $\mathbf{R}(s, \alpha, s') = \mathbf{r}(\alpha)$, $\mathbf{C}(s, \alpha, s') = \mathbf{c}(\alpha)$, and $\mathbf{C}(s) = \sigma$ for all $s, s' \in S$, $\alpha \in \Sigma$ if $s \xrightarrow{\alpha} s'$ and $\mathbf{R}(s, \alpha, s') = \mathbf{C}(s, \alpha, s') = 0$ otherwise.*

Note that these assignments to the transition system of an MSG still obey the rules of delayed choice. The same rate $\mathbf{r}(\alpha)$ is attached to all transitions labeled by α, no matter whether the transitions arise by "merging" two or more α-events according to the rules for delayed choice. This is in contrast to the choice operator $+$ of PEPA [19] and other stochastic process algebras, where $\alpha.x + \alpha.y$ implies a race between two α-labeled transitions. In particular, the exit rate of $\alpha.x + \alpha.y$ is $2{\cdot}\mathbf{r}(\alpha)$, whereas for the delayed choice operator $\mp$, the process $\alpha.x \mp \alpha.y$ is (bisimulation) equivalent to $\alpha.(x \mp y)$ and has the exit rate $\mathbf{r}(\alpha)$.

Example 3. Let us assume that both, the USB 1.1 scenario given in Example 1 and the stop-and-wait protocol of Example 2 use a low-speed USB 1.1 data connection for the communication between sender and receiver. To exemplify how statistical data can be included into early design steps for communication protocols, we define a profile $\mathcal{I} = (\mathbf{r}, \mathbf{c}, \sigma)$ with timings and costs in terms of

	s!r(data)	r?s(data)	s(timeout)	r!s(ack)	s?r(ack)	r!s(nak)	s?r(nak)
rates $\mathbf{r}(\cdot)$	1	1	0.1	10	10	0.5	10
energy $\mathbf{c}(\cdot)$	0.0323	0.0323	0	0.0032	0.0032	0.0032	0.0032

energy consumption within several assumptions inspired by measurements for USB 1.1 devices [31]. Rates are scaled upon sending/receiving one data package which comprises 8 bytes of data, 1 byte package identifier and 2 bytes CRC. Acknowledgements (positive and negative ones) have a package size of 1 byte. With probability 0.05 data is corrupted and with probability 0.01 totally lost. The energy-cost rate is fixed to be constantly $\sigma = 0.0597$, representing the power consumption of an idling USB device. Action energy costs stand for the additional power required to perform an action.

4.1 Model Checking Quantitative Requirements

In the following, let us fix the CTMC semantics $\mathcal{C}_{\mathcal{G}}$ for a quantitative MSG $(\mathcal{G}, \mathcal{I})$. Due to the undecidability result by Theorem 1, we cannot expect decidability for model checking $\mathcal{C}_{\mathcal{G}}$ against even basic qualitative reachability requirements.

Corollary 1. *Given a quantitative MSG* $(\mathcal{G},\mathcal{I})$*,* $A \subseteq \Sigma$*, and a time bound* $t \in \mathbb{Q}_{>0}$*, the decision problems for* $\mathcal{C}_\mathcal{G}$ *whether* $\Pr(\Diamond^{\leq t} D) > 0$ *and whether* $\Pr(\Diamond D) > 0$ *with* $D = \{\mathcal{K} \in \mathrm{Conf}(\mathcal{S}(\mathcal{G})) \colon En(\mathcal{K}) = A\}$ *are undecidable.*

This result directly yields undecidability of model checking quantitative MSGs against action-based CSRL requirements [6]. However, if $\mathcal{G}$ is locally synchronized, Theorem 2 yields that the underlying transition system of $\mathcal{C}_\mathcal{G}$ has a finite bisimulation quotient. Thus, also $\mathcal{C}_\mathcal{G}$ itself has a finite bisimulation quotient as rates and costs only depend on the actions assigned to transitions and $\mathcal{T}_{\mathcal{S}(\mathcal{G})}$ is deterministic. In this case, standard CTMC techniques are applicable.

Time-Bounded Reachability. Towards approximative solutions for arbitrary, possibly infinite $\mathcal{C}_\mathcal{G}$, we exploit the fact that $\mathcal{C}_\mathcal{G}$ is finitely branching and rates and costs are bounded. Thus, the probability and expectation of time-bounded reachability properties can be approximated by investigating only that part of $\mathcal{C}_\mathcal{G}$ which is reachable within a sufficient *truncation depth* [16].

Theorem 3. *Given a quantitative MSG* $(\mathcal{G},\mathcal{I})$*, a decidable set of target states* D *in* $\mathcal{C}_\mathcal{G}$*, a time bound* $t \in \mathbb{Q}_{>0}$ *and a precision* $\delta \in \mathbb{Q}_{>0}$*, one can compute* $x \in \mathbb{Q}_{\geq 0}$ *such that* $|\Pr(\Diamond^{\leq t} D) - x| \leq \delta$*.*

Let us go more into detail. For a set of states $X \subseteq S$, we denote by X_i the set of states $\mathcal{K} \in X$ where $\ell v\ell(\mathcal{K}) = i$ for $i \in \mathbb{N}$. We define $X_{\leq i} = \bigcup_{k=0}^{i} X_i$ and $\mathcal{C}^D_{\leq i}$ as the projection of $\mathcal{C}_\mathcal{G}$ onto $S_{\leq i}$ where all states in $S_i \cup D_{\leq i}$ are made absorbing without any rates or costs assigned. It is clear that by iteratively composing MSCs along paths in $\mathcal{G}$ in a breadth-first fashion and computing their prefixes, executions and configurations, $\mathcal{C}^D_{\leq i}$ can be effectively computed. Starting from i=0 we step-wise increment i and compute the probability $\delta_i = \Pr(\Diamond^{\leq t} S_i)$ in $\mathcal{C}^D_{\leq i}$, using standard methods for finite CTMCs. If $\delta_i \leq \delta$, we are done by computing $x = \Pr(\Diamond^{\leq t} D_{\leq i})$ in $\mathcal{C}^D_{\leq i}$, again by using standard methods.

Similarly, the *expected cumulative costs* $\mathrm{Ex}(\Diamond^{\leq t} D)$ for reaching D or the time bound $t \in \mathbb{Q}_{>0}$ can be approximated up to precision $\delta \in \mathbb{Q}_{>0}$. To see this, let $\mathcal{C}^+_{\leq i}$ arise from $\mathcal{C}^D_{\leq i}$ by making all states $s \in S_i \backslash D$ absorbing with $\mathbf{R}(s,s) = \sum_{\alpha \in \Sigma} \mathbf{r}(\alpha)$, $\mathbf{C}(s,s) = \max_{\alpha \in \Sigma} \mathbf{c}(\alpha)$, and $\mathbf{C}(s) = \sigma$. When Ex^-, Ex, and Ex^+ are the expectations in $\mathcal{C}^D_{\leq i}$, $\mathcal{C}_\mathcal{G}$, and $\mathcal{C}^+_{\leq i}$, respectively, we obtain

$$\eta^- = \mathrm{Ex}^-(\Diamond^{\leq t} D_{\leq i}) \;\leq\; \mathrm{Ex}(\Diamond^{\leq t} D) \;\leq\; \mathrm{Ex}^+(\Diamond^{\leq t} D_{\leq i}) = \eta^+$$

Again, we compute these expectations in the finite CTMCs $\mathcal{C}^D_{\leq i}$ for increasing i until $\eta_i^+ - \eta_i^- \leq \delta$ and choose η_i^- as approximation.

Limitations of the Approach. For time-bounded reachability of undecidable target sets D, our truncation-based approach clearly does not guarantee to succeed. This is the case, e.g., if $D = \{s \in S \colon \Pr_s(\Diamond^{\leq t} U) \geq \tau\}$ for some probability threshold $\tau \in [0,1]$ and a decidable set of states U (see Corollary 1). Such a pattern is useful, e.g., to approximate requirements stated within nested (action-based) CSRL formulas. To evaluate such nested formulas, we can apply a top-down approach that approximates nested probability constraints on-the-fly

by possibly further increasing truncation depths [16]. For the above pattern, this requires, e.g., that during the step-wise construction of $\mathcal{C}^D_{\leq i}$ there is a precision $\gamma \in \mathbb{Q}_{>0}$ such that for all $s \in S_{\leq i}$ we have $\Pr_s(\Diamond^{\leq t} U) \geq \tau$ if $\Pr_s(\Diamond^{\leq t} U) \geq \tau - \gamma$. Also for unbounded reachability probabilities of the form $\Pr(\Diamond D)$, our approach is not applicable in general, as there might be a "drift away from D" such that in $\mathcal{C}^D_{\leq i}$ the probability $\Pr(\Diamond S_i)$ does not converge to 0 when i tends to infinity (e.g., for $\mathcal{G}_{\text{stoc}}$ of Example 2 with a profile where actions of s have much higher rates than actions of r). However, in the case of convergence with limit 0, the approach presented for time-bounded reachability properties yields an approximation technique also for the time-unbounded case.

5 Implementation and Case Study

We implemented the breadth-first level-wise construction of our transition-system semantics for MSGs, the annotation of profiles and the finite-state projection approach [16] to approximate action-based CSL properties [6] using the probabilistic model checker Prism [20]. As Prism does not natively support action-based requirements to reason about, we included the option to encode the last action fired as a state variable in the target state [6]. Clearly, this yields a linear blow-up of the state space in the number of actions not apparent when using specialized action-based model-checking algorithms. The key in our implementation is an efficient construction of the transition-system semantics in a breadth-first fashion to obtain truncations. Given an MSG $\mathcal{G}$, we first implemented the construction directly on the partial-order prefixes of $\mathcal{S}(\mathcal{G})$ generated by compositions along paths in $\mathcal{G}$ up to a sufficient depth. This construction turned out to be very time-consuming, as the sufficient length of paths in $\mathcal{G}$ can be very long due to concurrent behaviors between basic MSCs in $\mathcal{G}$. We thus replaced this approach by generating candidates for enabled actions α in a configuration $\mathcal{K}$ from the local views of each process (cf., e.g., [3]) and then omitted those MSCs in $\mathcal{K}{\cdot}\alpha$ which are not prefixes of $\mathcal{S}(\mathcal{G})$, i.e., let the global control imposed by the $\mathcal{G}$ rule out only locally enabled actions.

The general computation scheme of our tool is depicted in Figure 4. At each constructed truncation ①, we feed Prism with the truncation (but without encoding actions to avoid the linear blow-up) and compute the probability of reaching the states added in the last truncation step within the given time bound ②. When this probability is not greater than the desired precision δ anymore ③, the Prism evaluates the property under consideration ④. Here, we choose the model

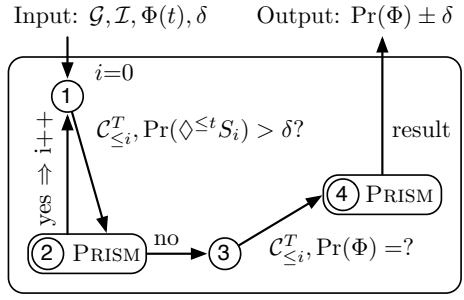

Fig. 4. The general tool scheme

where the last action performed is encoded into the states to allow reasoning about state-based CSRL formulas containing action-based information.

5.1 Models and Requirements in our Case Study

We evaluated our implementation based on the quantitative MSGs provided in Example 3. Besides the CTMCs for $\mathcal{G}_{\text{usb}}$ and $\mathcal{G}_{\text{stoc}}$, we consider a variant $\mathcal{G}_{\text{rstoc}}$ of $\mathcal{G}_{\text{stoc}}$ where the data channel is reliable, i.e., the upper left transition in Figure 3 is omitted. We also exploit the regular structure of the MSGs and introduce backward transitions to bisimilar states on-the-fly during the construction of the truncations, based on the following observations for configurations $\mathcal{K}$:

(usb) Fully executed loops do not have any impact on future behaviors, i.e., if the uniquely defined prefix in $\mathcal{K}$ has the form $\mathcal{M}_!\mathcal{M}_!\mathcal{M}_?\mathcal{N}$, then $\mathcal{K}$ is bisimilar to the configuration $\{\mathcal{M}_!\mathcal{M}_!\mathcal{N}\}$.[2]

(stoc, rstoc) Matches not affected by delayed choice synchronize. That is, if there is a message matched in each prefix $K \in \mathcal{K}$ at the same position, removing preceding events in all prefixes of $\mathcal{K}$ yields a bisimilar configuration.

As requirements for the performance analysis, we investigated the following properties expressed in an action-based CSRL fashion [6]:

(success) $\Pr(\Diamond^{\leq 6}\text{s?r(ack)})$: What is the probability of a successful communication within 6 time units?

(energy) $\text{Ex}_{\text{energy}}(\Diamond^{\leq 6}\text{s?r(ack)})$: What is the expected energy consumed towards a successful communication or reaching the deadline of 6 time units?[3]

(resilience) $\Pr(\Box((\text{s?r(nak)} \vee \text{s(timeout)}) \Rightarrow \mathsf{P}_{\geq 0.9}(\Diamond^{\leq 6}\text{s?r(ack)})))$: What is the probability that always after an unsuccessful communication, with probability at least 0.9 the communication is successful within 6 time units?

The latter **resilience** property can only be expressed by a nested formula, where evaluating the inner probability operator is undecidable in general (see Corollary 1). Furthermore, the property contains a time-unbounded reachability modality, such that our approach is not a priori successful. Fortunately, for the quantitative MSGs we considered in our case study, our approach is still applicable (cf. last paragraph of Section 4).

5.2 Evaluation

For carrying out the case study, we used PRISM version 4.2 on a computer with Intel i7-3720QM processor running at 2.6 Ghz with 8 GBytes of memory. The required truncation depth i is computed for a precision of $\delta = 10^{-6}$ and we chose a precision of $\epsilon = 10^{-12}$ for the PRISM computations. In Table 1, we summarize our

[2] Cutting off fully executed loops of configurations is in the spirit of [10], however, does not always yield a bisimular configuration in general due to delayed choice.

[3] This property corresponds to the PRISM query `R=?[C<=6]` on the transformed model $\mathcal{C}'_{\mathcal{G}}$ as described in Section 4, where all states are terminal after an s?r(ack) action.

Table 1. Results of the truncation-based quantitative analysis

MSG	property	result	depth i		#states		overall time		analysis time	
			$\mathcal{C}_\mathcal{G}$	$\mathcal{C}_\mathcal{G}/\equiv$	$\mathcal{C}_\mathcal{G}$	$\mathcal{C}_\mathcal{G}/\equiv$	$\mathcal{C}_\mathcal{G}$	$\mathcal{C}_\mathcal{G}/\equiv$	$\mathcal{C}_\mathcal{G}$	$\mathcal{C}_\mathcal{G}/\equiv$
$\mathcal{G}_{\text{usb}}$	**success**	0.966	38	**16**	145 (231)	43 (65)	1.3s	0.6s	0.8s	0.3s
	energy	0.351	37	**16**	141 (226)	43 (65)	1.4s	0.6s	0.8s	0.3s
$\mathcal{G}_{\text{rstoc}}$	**success**	0.946	22	11	3252 (5158)	26 (53)	6.5s	0.3s	1.8s	0.2s
	energy	0.225	21	11	2524 (4067)	26 (53)	4.8s	0.3s	1.4s	0.2s
	resilience	0.995	>40	11+1	>400000	28 (58)	>12h	0.4s	–	0.2s
$\mathcal{G}_{\text{stoc}}$	**success**	0.955	21	21	12513 (17824)	4124 (6177)	84.7s	39.2s	5.6s	2.1s
	energy	0.221	21	21	12513 (17824)	4124 (6177)	86.4s	41.0s	10.0s	3.9s
	resilience	0.999	>40	29+0	>400000	69084 (99505)	>12h	1.3h	–	38.2s

results, where $\mathcal{C}_\mathcal{G}/\equiv$ stands for the bisimulation quotient constructed with respect to the bisimulation relations identified above. The number of states are those of the final CTMC and we indicated the number of states in the action-encoded model (required for the final PRISM computation) in brackets. As shown also in the table, generating the truncations of the CTMC semantics of quantitative MSGs is very time-consuming and bisimulation techniques are very promising. This is especially the case for the most simple quantitative MSG $\mathcal{G}_{\text{usb}}$, which is locally synchronized and where after a truncation depth $i = 16$, the bisimulation quotient does not change anymore (see Theorem 2). The **resilience** property is not considered for $\mathcal{G}_{\text{usb}}$, as the channels of the communication scenarios generated are all reliable. We indicated the further truncation steps j required to guarantee correct approximations of the nested action-based CSRL formula for the **resilience** property separately, indicated by $i+j$. Without employing bisimulation techniques, we run into timeouts in all **resilience** experiments.

Remark 1. Besides the *finite-state projection* we applied in our approach, [16] presented other heuristics to compute sufficient truncation depths and saving analysis time. We evaluated their unichain and layered-chain method, but got high truncation depths where the truncation could not be constructed anymore. This is not surprising, as the analysis step is not the bottleneck in our approach, rather than the construction of the truncation (see timings in Table 1).

6 Conclusion

We presented a transition-system semantics for MSGs, which obeys delayed choice and thus follows the ITU standard for MSGs [22]. Different to other branching-time semantics for MSGs, our semantics can be interpreted on all kinds of MSGs where a partial-order semantics is defined, including compositional MSGs [15,9,26] and causal MSGs [13]. Our semantics opens the door towards probabilistic operational semantics for MSGs which may arise by an annotation of stochastic information. Such annotations enable to reason about performance measures, which can trigger redesign steps in early communication protocol development. In this paper, we focused on annotations in terms

of stochastic timing information and costs and presented a (possibly infinite) CTMC semantics for MSGs. We showed that already a simple qualitative time-bounded reachability problem is undecidable, whereas approximative solutions can be obtained choosing a truncation-based approach. By an implementation and a case study we demonstrated that practical relevant properties such as the probability of failure, the expected energy consumption and a probabilistic resilience measure can be approximated using our approach.

Our transition-system semantics as well as the CTMC semantics can be used for many other purposes. For instance, states could be annotated with atomic propositions defined through MSO formulas over MSCs (which are decidable [26]). This enables to reason about state-based branching-time properties aimed already for in [25] but now with obeying delayed choice. More sophisticated annotations for stationary costs can also be imagined, depending for instance on the local state of each process in the spirit of [3] or histories of actions performed to reach the state. We noticed in our case study that bisimulation techniques play an important role for analyzing MSGs. However, finding a suitable bisimulation is not as easy (e.g., cutting of fully executed loops as presented by [10] does not maintain language consistency within a delayed choice semantics). Establishing generic approaches towards bisimulations are left for further work.

References

1. S. Akshay, P. Gastin, M. Mukund, and K. N. Kumar. Model checking time-constrained scenario-based specifications. In *FSTTCS 2010*, pages 204–215, 2010.
2. S. Akshay, B. Genest, L. Hélouët, and S. Yang. Regular set of representatives for time-constrained MSC graphs. *Inf. Process. Lett.*, 112(14-15):592–598, 2012.
3. R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *Software Engineering, IEEE Transactions on*, 29(7):623–633, July 2003.
4. R. Alur, G. J. Holzmann, and D. Peled. An analyzer for message sequence charts. In *SOFTWARE CONCEPTS AND TOOLS*, pages 304–313, 1996.
5. R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proceedings of CONCUR'99*, volume 1664 of *LNCS*, pages 114–129, 1999.
6. C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking Markov chains with actions and state labels. *Software Engineering, IEEE Transactions on*, 33(4):209–224, April 2007.
7. C. Baier, B. Haverkort, H. Hermanns, and J. Katoen. Model-checking algorithms for continuous-time Markov chains. *Software Engineering, IEEE Transactions on*, 29(6):524–541, June 2003.
8. C. Baier and J.-P. Katoen. *Principles of model checking.* The MIT Press, 2008.
9. B. Bollig, M. Leucker, and P. Lucas. Extending compositional message sequence graphs. In *LPAR*, pages 68–85, 2002.
10. J. Chakraborty, D. D'Souza, and K. Narayan Kumar. Analysing message sequence graph specifications. In B. Steffen, editor, *Leveraging Applications of Formal Methods, Verification, and Validation*, volume 6415 of *LNCs*, pages 549–563. Springer Berlin Heidelberg, 2010.
11. D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders. Möbius: An extensible tool for performance and dependability modeling. In B. R.

Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *Computer Performance Evaluation.Modelling Techniques and Tools*, volume 1786 of *LNCS*, pages 332–336. Springer Berlin Heidelberg, 2000.
12. A. Fantechi, S. Gnesi, and G. Ristori. Model checking for action-based logics. *Formal Methods in System Design*, 4(2):187–203, 1994.
13. T. Gazagnaire, B. Genest, L. Hélouët, P. S. Thiagarajan, and S. Yang. Causal message sequence charts. *Theor. Comput. Sci.*, 410(41):4094–4110, 2009.
14. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model-checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6).
15. E. L. Gunter, A. Muscholl, and D. A. Peled. Compositional message sequence charts. In *TACAS'01*, volume 2031 of *LNCS*, pages 496–511. Springer, 2001.
16. E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. Time-bounded model checking of infinite-state continuous-time markov chains. *Fundamenta Informaticae*, 95(1):129–155, 2009.
17. L. Hélouët, C. Jard, and B. Caillaud. An event structure based semantics for high-level message sequence charts. *Math. Struct. in CS*, 12(4):377–402, 2002.
18. T. A. Henzinger, M. Mateescu, and V. Wolf. Sliding window abstraction for infinite Markov chains. In *Proceedings of CAV'09*, 2009.
19. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
20. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.
21. ITU-T. Annex b: Formal semantics of message sequence charts. *Z.120*, v2.2, 1998.
22. ITU-T. Message Sequence Chart (MSC). *Z.120*, Edition 5.0, 2011.
23. K. N. Kumar. The theory of message sequence charts. In *Modern Applications of Automata Theory*, pages 289–324. 2012.
24. P. Lucas. Timed semantics of message sequence charts based on timed automata. *Electronic Notes in Theoretical Computer Science*, 65(6):160 – 179, 2002.
25. P. Madhusudan. Reasoning about sequential and branching behaviours of message sequence graphs. ICALP '01, pages 809–820, London, UK, 2001. Springer-Verlag.
26. P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In R. Hariharan, V. Vinay, and M. Mukund, editors, *FSTTCS'01*, volume 2245, pages 256–267. Springer Berlin Heidelberg, 2001.
27. S. Mauw and M. A. Reniers. High-level message sequence charts. In *SDL Forum*, pages 291–306, 1997.
28. A. M. Mousa. Prospective of fifth generation mobile communications. *International Journal of Next-Generation Networks*, 2012.
29. A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *MFCS'99*, volume 1672 of *LNCS*, pages 81–91. Springer, 1999.
30. A. Muscholl and D. Peled. Deciding properties of message sequence charts. In *Scenarios: Models, Transformations and Tools*, volume 3466 of *LNCS*, pages 575–575. Springer Berlin/Heidelberg, 2005.
31. K. O'Brien, D. Salyers, A. Striegel, and C. Poellabauer. Power and performance characteristics of usb flash drives. In *WoWMoM'08*, pages 1–4, June 2008.
32. R. Pooley and P. King. The unified modeling language and performance engineering. In *IEEE Proceedings - Software*, volume 149, pages 2–10, 1999.
33. A. Remke and B. R. Haverkort. *CSL Model Checking Algorithms for Infinite-State Structured Markov Chains*, volume 4763 of *LNCS*, pages 336–351. Springer, 2007.

34. M. A. Reniers. *Message Sequence Chart: Syntax and Semantics.* PhD thesis, Eindhoven University of Technology, June 1999.
35. G. Rodrigues, D. Rosenblum, and S. Uchitel. Using scenarios to predict the reliability of concurrent component-based software systems. In *Fundamental Approaches to Software Engineering*, volume 3442, pages 111–126. Springer, 2005.
36. A. S. Tanenbaum and D. J. Wetherall. *Computer Networks.* Prentice Hall, 5th edition, 2011.
37. Z. Zhou, F. T. Sheldon, and T. E. Potok. Modeling with stochastic message sequence charts. In *IIIS Proceedings of the International Conference on Computer, Communication, and Control Technology (CCCT'03)*, 2003.