Quantitative Modeling and Analysis of Service-Oriented Real-Time Systems using Interval Probabilistic Timed Automata

Christian Krause, Holger Giese

Technische Berichte Nr. 56

des Hasso-Plattner-Instituts für
Softwaresystemtechnik
an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 56

Christian Krause | Holger Giese

Quantitative Modeling and Analysis of Service-Oriented Real-Time Systems using Interval Probabilistic Timed Automata

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de/ abrufbar.

Universitätsverlag Potsdam 2012
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Druck: docupoint GmbH Magdeburg

ISBN 978-3-86956-171-4

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
URL http://pub.ub.uni-potsdam.de/volltexte/2012/5784/
URN urn:nbn:de:kobv:517-opus-57845
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-57845

Abstract

One of the key challenges in service-oriented systems engineering is the prediction and assurance of non-functional properties, such as the reliability and the availability of composite interorganizational services. Such systems are often characterized by a variety of inherent uncertainties, which must be addressed in the modeling and the analysis approach. The different relevant types of uncertainties can be categorized into (1) epistemic uncertainties due to incomplete knowledge and (2) randomization as explicitly used in protocols or as a result of physical processes.

In this report, we study a probabilistic timed model which allows us to quantitatively reason about non-functional properties for a restricted class of service-oriented real-time systems using formal methods. To properly motivate the choice for the used approach, we devise a requirements catalogue for the modeling and the analysis of probabilistic real-time systems with uncertainties and provide evidence that the uncertainties of type (1) and (2) in the targeted systems have a major impact on the used models and require distinguished analysis approaches.

The formal model we use in this report are Interval Probabilistic Timed Automata (IPTA). Based on the outlined requirements, we give evidence that this model provides both enough expressiveness for a realistic and modular specification of the targeted class of systems, and suitable formal methods for analyzing properties, such as safety and reliability properties in a quantitative manner. As technical means for the quantitative analysis, we build on probabilistic model checking, specifically on probabilistic time-bounded reachability analysis and computation of expected reachability rewards and costs.

To carry out the quantitative analysis using probabilistic model checking, we developed an extension of the PRISM tool for modeling and analyzing IPTA. Our extension of PRISM introduces a means for modeling probabilistic uncertainty in the form of probability intervals, as required for IPTA. For analyzing IPTA, our PRISM extension moreover adds support for probabilistic reachability checking and computation of expected rewards and costs. We discuss the performance of our extended version of PRISM and compare the interval-based IPTA approach to models with fixed probabilities.

Keywords: service-oriented systems, real-time systems, quantitative analysis, formal verification methods

Zusammenfassung

Eine der wichtigsten Herausforderungen in der Entwicklung von Service-orientierten Systemen ist die Vorhersage und die Zusicherung von nicht-funktionalen Eigenschaften, wie Ausfallsicherheit und Verfügbarkeit von zusammengesetzten, interorganisationellen Diensten. Diese Systeme sind oft charakterisiert durch eine Vielzahl von inhärenten Unsicherheiten, welche sowohl in der Modellierung als auch in der Analyse eine Rolle spielen. Die verschiedenen relevanten Arten von Unsicherheiten können eingeteilt werden in (1) epistemische Unsicherheiten aufgrund von unvollständigem Wissen und (2) Zufall als Mittel in Protokollen oder als Resultat von physikalischen Prozessen.

In diesem Bericht wird ein probabilistisches, Zeit-behaftetes Modell untersucht, welches es ermöglicht quantitative Aussagen über nicht-funktionale Eigenschaften von einer eingeschränkten Klasse von Service-orientierten Echtzeitsystemen mittels formaler Methoden zu treffen. Zur Motivation und Einordnung wird ein Anforderungskatalog für probabilistische Echtzeitsysteme mit Unsicherheiten erstellt und gezeigt, dass die Unsicherheiten vom Typ (1) und (2) in den untersuchten Systemen einen Einfluss auf die Wahl der Modellierungs- und der Analysemethode haben.

Als formales Modell werden *Interval Probabilistic Timed Automata* (IPTA) benutzt. Basierend auf den erarbeiteten Anforderungen wird gezeigt, dass dieses Modell sowohl ausreichende Ausdrucksstärke für eine realistische und modulare Spezifikation als auch geeignete formale Methoden zur Bestimmung von quantitativen Sicherheits- und Zuverlässlichkeitseigenschaften bietet. Als technisches Mittel für die quantitative Analyse wird probabilistisches Model Checking, speziell probabilistische Zeit-beschränkte Erreichbarkeitsanalyse und Bestimmung von Erwartungswerten für Kosten und Vergütungen eingesetzt.

Um die quantitative Analyse mittels probabilistischem Model Checking durchzuführen, wird eine Erweiterung des PRISM-Werkzeugs zur Modellierung und Analyse von IPTA eingeführt. Die präsentierte Erweiterung von PRISM ermöglicht die Modellierung von probabilistischen Unsicherheiten mittels Wahrscheinlichkeitsintervallen, wie sie für IPTA benötigt werden. Zur Verifikation wird probabilistische Erreichbarkeitsanalyse und die Berechnung von Erwartungswerten durch das Werkzeug unterstützt. Es wird die Performanz der PRISM-Erweiterung untersucht und der Intervall-basierte IPTA-Ansatz mit Modellen mit festen Wahrscheinlichkeitswerten verglichen.

Schlagwörter: Service-orientierte Systme, Echtzeitsysteme, Quantitative Analysen, Formale Verifikation

Contents

Chapter 1

Introduction

The prediction and assurance of quantitative properties of composite service-oriented real-time systems imposes a number of requirements on the used modeling and verification approaches. Particularly challenging is the fact that these systems are characterized by a variety of uncertainties which are moreover of different natures. In general, two different relevant categories of uncertainties can be distinguished: (1) empirical uncertainties due to incomplete knowledge and (2) probabilistic and stochastic uncertainties due to protocol designs or random physical processes. Incorporating these two different kinds of uncertainties into a modeling and verification approach is highly non-trivial, since the derived quantitative predictions are sensitive to formally incorrect assumptions for uncertain properties. For example, the assumption of a probabilistic decision making in settings where choices are actually controlled by a possibly unknown software component inevitable leads to incorrect predictions.

In this report, we study the formal model Interval Probabilistic Timed Automata (IPTA) which we argue is suitable for describing service-oriented real-time systems with uncertainties. In order to motivate and justify our modeling choice, we devise a requirements catalogue for the modeling and verification of the targeted class of systems. We particularly characterize a number of uncertainty dimensions and show how they are formally represented in IPTA. The mathematical concepts for describing these phenomena are (a) nondeterminism, (b) probabilistic modeling, and (c) interval-based specification of probabilities, which can be seen as a combination of nondeterministic and probabilistic behavior.

Regarding the quantitative analysis, we mainly consider safety and reliability properties in this report. An example of a quantitiative safety properties is 'the probability that the system reaches a critical error state is less than 1%'. A typical reliability property is the assurance of a bounded response time of a service with a high probability, such as 'the response time of a service is less than 200ms for at least 95% of the requests'. A major challenge for the verification of such properties is that they often make assertions about the functional behavior, real-time constraints as well as probabilities. In the course of this report, we show that such properties can be formally specified and analyzed using IPTA.

Applicability of the verification approach was one of the steering requirements for the work in this report. Therefore, we developed a tool for automatically carrying out the quantitative analysis of IPTA. Specifically, we have developed an extension of the PRISM tool for model checking IPTA in the context of this report, which we present in detail here and evaluate using benchmarks and a comparison of IPTA with models that allow only the use of fixed probabilities.

1.1 Contributions

This report builds on results presented in [21]. In the following, we give an overview of the contributions of this report.

We present a catalogue of 17 requirements for the modeling and the analysis of probabilistic real-time systems with uncertainties. The inherent uncertainties due to incomplete knowledge and probabilistic phenomena play a key role and are therefore discussed in more detail.

We give a comprehensive formal presentation of Interval Probabilistic Timed Automata (IPTA) and provide evidence that this model is expressive enough for a realistic specification of the targeted class of systems. In particular, we show how service-oriented real-time systems can be formally specified in a modular way using IPTA.

Regarding quantitative analysis, we target safety and reliability properties in settings with time-bounded contracts between participants in a service-oriented system. We motivate such settings by the existence of service-level agreements (SLAs) which limit the time window in which properties are guaranteed. As technical means for the quantitative analysis, we build on probabilistic time-bounded reachability checking and computation of expected reachability reward and costs.

To carry out the quantitative analysis using probabilistic model checking, we developed an extension of the PRISM tool for modeling and analyzing IPTA. Our extension of PRISM introduces a syntax for modeling probabilistic uncertainty in the form of probability intervals, as required for IPTA. For analyzing IPTA, our PRISM extension moreover adds support for probabilistic reachability checking and computation of expected rewards and costs. We discuss the performance of our tool and compare the interval-based IPTA approach to models with fixed probabilities.

1.2 Organization

The rest of this report is organized as follows. In Chapter 2, we device a requirements catalogue for modeling and analyzing service-oriented real-time systems with uncertainties. In Chapter 3, we present our formal modeling approach for probabilistic real-time systems using Interval Probabilistic Timed Automata. In Chapter 4, we review the requirements presented in Chapter 2 and give evidence that IPTA provide a suitable modeling approach for the targeted settings. In Chapter 5, we present the formal means for a quantitative analysis of IPTA. In Chapter 6, we present our tool support based on an extension of PRISM and evaluate our approach. Chapter 7 contains related work. In Chapter 8, we present our conclusions.

Chapter 2

Requirements

Service-oriented systems engineering requires means for describing and reasoning about both functional and non-functional behavioral aspects. The systems to be modeled are often characterized by a variety of inherent uncertainties, which must be addressed in the modeling and the verification approach. The different types of uncertainties can be categorized in:

(1) epistemic uncertainties due to incomplete knowledge,

(2) randomization as used in certain protocols or as a result of physical processes,

(3) linguistic ambiguities due to imprecise or informal descriptions.

In this chapter, we review requirements for a quantitative modeling and verification approach for service-oriented real-time systems with uncertainties. In particular, we give evidence that nondeterminism and probabilistic / stochastic models provide a formal and technical means for a proper handling of uncertainties according to (1) and (2), respectively. Since our focus lies on *formal* modeling, the impact of linguistic uncertainties in (3) is not relevant and therefore not covered in this report.

2.1 Modeling with Uncertainties

2.1.1 Basic Behavior Modeling with Uncertainties

Service-oriented computing (SOC) and model-driven engineering (MDE) have evolved to two of the most influencing paradigms in the IT-world in the last decade. On the one hand, service-oriented computing facilitates the development of interoperable, interorganizational, distributed software components. On the other, model-driven engineering provides a methodology for defining systems using high-level models and to automatically derive implementations through model transformation and code generation.

An application of the model-driven paradigm to SOC can be found in the *Service oriented architecture Modeling Language* (SoaML) [32]. SoaML is an open source specification project initiated by the OMG and consist in its core of a UML profile and a meta-model for specifying service-oriented systems.

For the definition of a system as a composition of a set of (primitive or composite) services, SoaML provides among other concepts the notions of a *service interface* and a *service contract*. Service inter-

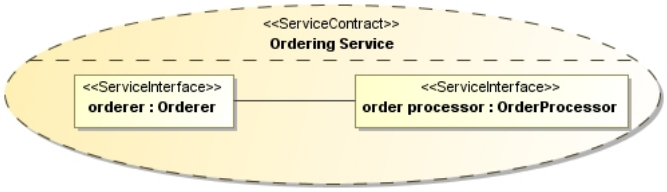

Figure 2.1: A service contract specified in SoaML notation [32].

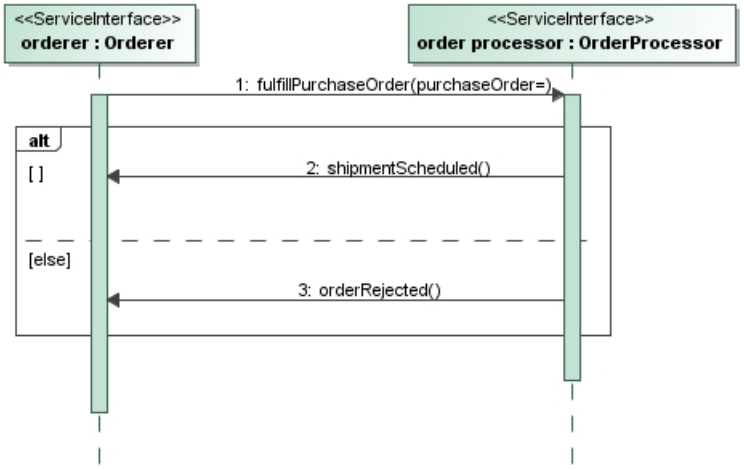

Figure 2.2: A protocol for a service contract defined using a UML sequence diagram [32].

faces describe the publicly available communication ports and the supported message types of a service. Service contracts are used to specify the terms and conditions that all participants of the contract agree on. In particular, service contracts hold information about the used service interfaces and specify the choreography between the participants, e.g. in terms of protocols.

Figure 2.1 depicts a SoaML diagram notation of an example service contract. This contract consists of two services of type ⟪Orderer⟫ and ⟪OrderProcessor⟫. The details of these services, in particular the communication ports and message types, are omitted in this notation. However, to be useful for a verification of functional properties, a behavioral model describing the interaction between these two services is required.

Since SoaML is based on the UML, various diagram types can be used for modeling the choreography in service contracts. For instance, Figure 2.2 describes the allowed interactions using a UML sequence diagram. In this simple example, the orderer may send an order using the method fulfillPurchaseOrder() on which the order processor either replies affirmatively using the message shipmentScheduled() or which is signaled as failed using the message orderRejected(). The decision making within the order processor can be modeled explicitly using a guard or another diagram or can be assumed as non-deterministic. Nondeterminism therefore allows to express uncertainty, which, however, is of a purely qualitative nature.

Note again that various behavioral models can be used to specify service contracts, e.g. UML activity diagrams or state machines. However, to be useful for verification, formal models that describe the complete behavior and not only a partial view of the system are required (e.g. finite-state automata).

Based on the sketched concepts of the SoaML, we now summarize the requirements of our modeling approach with respect to the functional and architectural properties in service-oriented systems.

Basic Requirements

(R1) Specification of stateful, behavioral service interfaces

(R2) Synchronous communication via a set of publicly accessible ports

(R3) Qualitative uncertainty expressed using nondeterminstic choice

(R4) Modular design and parallel composition

(R5) No enforced architectural style

(R6) Formal model with support for verification

Note that the behavior of service-oriented systems is additionally characterized by *structure dynamics*, i.e. by dynamically changing architectures due to rebinding of service endpoints or reconfiguration at run-time. These characteristics are not considered in this report and require behavioral models based on local rewriting of structures rather than finite-state automata models. Furthermore, due to the scale of service-oriented systems and the late binding and replacement of service links at runtime, a monolithic approach to modeling and analysis cannot work. Instead, a compositional approach employing abstraction in form of agreed upon service contracts is required. We leave this requirement also as future work.

2.1.2 Real-Time Modeling with Uncertainties

Real-time behavior plays a key role in embedded systems, e.g. missing a deadline can lead to a total system failure. Therefore, it is highly important that modeling approches for such systems can faithfully represent real-time behavior, e.g. using strict timing constraints, such as deadlines.

Besides this classical notion of *hard* real-time behavior, in service-oriented systems the missing of deadlines often only degrades the system's Quality of Service (QoS), thus leading to *soft* real-time requirements. Nevertheless, the non-functional properties and particularly the QoS properties can be as important as functional properties for determining the value of a service in the market. The assurance of QoS properties such as reliability and performance in service contracts is particularly challenging because of the inherent interdependencies between the participants of a service contract. Therefore, to be able to provide assurances of a service-oriented system, it is crucial to incorporate also QoS properties into service contracts in the form of assume–guarantee statements.

A large number of QoS properties that are relevant in real-world applications are related to time. For instance, the throughput of a communication channel is a measure for its performance and is defined as the number of delivered messages per time unit. In service-oriented settings, reliability also plays an important role. For example, the response time of a server is an important measure since clients may switch to another service provider if they do not receive a response within a reasonable amount of time. In such scenarios, timeout values of clients can be seen as deadlines which should be met by a service implementation in order to be useful. Response time guarantees therefore constitute a standard example for real-time behavior in service-oriented systems.

To provide an example for the relevance of response time guarantees in service-oriented systems, we consider an industry standard for the specification of QoS properties in service contracts, also referred to as Service Level Agreements (SLAs). A specific notation for SLAs is the XML-based Web Service Level Agreement (WSLA) [17, 7] language, which provides a syntax to specify QoS metrics and guarantees for web services. Listing 2.1 contains an adaptation of a WSLA specification presented in [17] in which a response time metric is defined. This metric measures the percentage of server responses which are

Listing 2.1: A real-time metric in WSLA.

```
<Metric name="NormalResponsePercentage" type="float" unit="Percentage">
  <Source>ServiceProvider</Source>
  <Function resultType="float" xsi:type="wsla:PercentageLessThanThreshold">
    <Metric>ResponseTime</Metric>
    <Value>
      <LongScalar>20</LongScalar>   <!-- Normal responses should take less than 20ms -->
    </Value>
  </Function>
</Metric>
```

delivered within a deadline of 20ms. Thus, the ability of specifying and verifying real-time behavior plays an important role for service-oriented systems engineering.

Uncertainty in the form of nondeterminism plays also an important role in real-time modeling. Specifically, uncertainty can be expressed by specifying time windows in which the triggering of transitions is based again on nondeterministic choice.

Real-Time Modeling Requirements

(R7) Observable timed behavior: execution steps occur at specific time points in a dense time domain

(R8) Measuring of durations

(R9) Specification of real-time behavior using time-dependent guards on transitions, e.g. minimal waiting times or hard deadlines

(R10) Nondeterminism for choosing a specific time point within an allowed time interval

2.1.3 Probabilistic Modeling with Uncertainties

For a realistic modeling of service-oriented real-time systems, incorporating uncertainty using probabilistic behavior is necessary. For example, service providers usually guarantee an upper limit for a response time up to a small percentage of requests for which it is allowed that the deadline is missed. As a specific example, consider the WSLA-excerpt in Listing 2.2 which uses the response time metric in Listing 2.1 to define a response time guarantee of at least 95% of responses that are delivered within the deadline of 20ms. Thus, probabilities can be used for the specification of *soft* deadlines, i.e. deadlines which are allowed to be missed for a given percentage of executions.

The compliance of a service implementation with an SLA such as the response time guarantee in Listing 2.2 is usually checked either at runtime by means of monitoring, or at design time using simulations or statistical model checking. The quantitative nature of SLAs and particularly the probabilistic features provide more flexibility for the specification of non-functional properties while still allowing to formally analyze them. Similarly to response time guarantees, probabilistic models also facilitate the specification of unreliable media, such as lossy communication channels.

Probabilistic real-time behavior can be moreover found in systems that make explicit use of randomization to achieve a functional goal. For instance, in communication and multimedia protocols such as the IEEE 1394 FireWire Root Contention Protocol [13], randomization in the form of *electronic coin tossing* is used to elect a root node among a number of interconnected nodes with initially equal

Listing 2.2: A probabilistic response time guarantee in WSLA.

```
<Obligations>
  <ServiceLevelObjective name="ResponseTimeGuarantee">
    <Obliged>ServiceProvider</Obliged>
    <Expression>
      <Predicate xsi:type="GreaterEqual">
        <SLAParameter>NormalResponsePercentage</SLAParameter>
        <Value>0.95</Value>          <!-- At least 95% normal responses -->
      </Predicate>
    </Expression>
  </ServiceLevelObjective>
<Obligations>
```

status (also referred to as a *randomized leader election* protocol). Hence, probabilistic behavior can be relevant for both non-functional and functional properties.

We emphasize here that probabilistic behavior is not a replacement for nondeterminism. The assumption that a nondeterministic choice is governed by a probability mechanism is valid only in special cases and therefore cannot be justified in general. Informally, probabilistic choices produce predictable long-run or average behavior, whereas nondeterministic choices result in completely unpredictable behavior. Thus, it is highly important to keep these two concepts completely separate (see, e.g., [36]).

Regarding uncertainty, it is often desirable to allow for a specification of probability intervals, as opposed to fixed probabilities. For instance, probability intervals are commonly required for the specification of SLAs, such as the response time guarantee in Listing 2.2. Here a probability of at least 95% is required which is formally represented by the probability interval $[0.95, 1]$.

Probabilistic Requirements

(R11) Support for probabilistic choices in addition to nondeterministic choices

(R12) Probabilistic uncertainty using probability intervals

(R13) Combination of uncertain probabilistic and uncertain real-time behavior: *uncertain probabilistic real-time behavior*

2.1.4 Stochastic Modeling

Probabilistic real-time behavior as described in the previous section is limited to discrete probabilistic branching. Consequently, stochastic behavior, i.e. behavior where the choice for a point in time is steered by a continuous-time probability distribution is not supported. Such stochastic behavior can be modeled, e.g., using continuous-time Markov chains and is often used to describe random physical processes, such as hardware failures, and inter-arrival times of customers in a shop. Stochastic behavior is particularly useful to model events based on the expected time of their occurrence. However, note that hard real-time behavior as demanded in requirements (R9)–(R10) cannot be expressed using probability distributions with an unbounded time domain, such as exponential distributions.

For the targeted class of systems and settings in this report, stochastic behavior is less important than probabilistic real-time behavior. On the one hand, the typical interaction time of clients with services (e.g. in a session) is comparatively short which means that probabilities for hardware failures in such an interaction do not change in practice. Since we focus on on-request probabilistic behavior, modeling uncertain failure behavior can be realized using discrete probabilistic choices, as demanded in (R11).

On the other hand, the modeling of inter-arrival times of requests at a server, which is a standard application of stochastic processes, is primarily important for reasoning about long-run characteristics such as average availability and performance. As we argue in Section 2.2, our focus for the quantitative analysis of service-oriented real-time systems lies on on-request behavioral properties, such as safety and reliability properties. For these settings and properties, probabilistic real-time behavior as demanded in requirement (R13) provides sufficient expressiveness and therefore we do not have to rely on stochastic behavior modeling.

2.2 Quantitative Analysis

In this section, we summarize the requirements for a quantitative analysis for models featuring the modeling requirements (R1)–(R13) in the previous section.

2.2.1 Probabilistic Time-Bounded Reachability

The classical version of the reachability problem in model checking is to determine whether a set of target states, e.g. given by a logical state formula, can be reached from the initial state or not. Since we target a modeling language with quantitative aspects, specifically time and probabilities, it is reasonable to reformulate the reachability problem based on these quantitative properties.

Time-Bounded Reachability extends the classical reachability problem by requiring to reach the set of target state within a given amount of time. An example of a time-bounded reachability problem is the requirement that in a given interaction between a client and a server, the server sends a response within a given number of time units after the client has sent a request.

Probabilistic Reachability generalizes the classical reachability problem by demanding to compute a probability for reaching the target states, as opposed to simply *'yes'* or *'no'* answer. If the model also supports nondeterminism, a minimal and maximal probability should be computed. An example for a probabilistic reachability problem is to determine the (minimal and maximal) probability that a client receives a response from a server after the client has sent a request.

Probabilistic Time-Bounded Reachability is an orthogonal combination of the time-bounded and probabilistic reachability problems. An example of probabilistic time-bounded reachability is to determine the (minimal and maximal) probability that a client receives a response from a server within a given number of time units after the client has sent a request.

For service-oriented systems, probabilistic time-bounded reachability is a means that enables service providers to estimate the 'goodness' of their services by quantifying their non-functional properties. A quantitative analysis for probabilistic time-bounded reachability problems can moreover be a starting point for the specification of service level agreements.

In the context of service-oriented real-time systems, relevant quantitative properties that can be formalized using probabilistic reachability properties include safety and reliability of systems. Safety properties usually assert that the probability for reaching a failure state is sufficiently small. Note that such a safety property can be also defined in the context of an SLA by specifying an upper bound for the probability of a failure event. Additionally, if the failure can be caused by a (stochastic) malfunctioning of a hardware component, it is often desirable to fix a bounded mission time in which the failure state is reached. Thus, the combination of time-bounded and probabilistic reachability is important here. As described above, another common example for probabilistic real-time properties are bounded response time guarantees, such as the service level objective in the WSLA excerpt in Listing 2.2. Note that

estimating probabilities for slow responses and thereby effectively checking soft deadlines provides in practice more useful information for the reliability of (real-time) services than average response times.

Probabilistic Reachability Requirements

(R14) Logic for the formal specification of probabilistic time-bounded reachability properties

(R15) Availability of model checking algorithms and verification tools for (R14)

2.2.2 Expected Reachability Costs and Rewards

In addition to probabilistic time-bounded reachability problems, it is often useful to calculate the accumulated costs or rewards until a set of target states is reached. A typical expected value of interest is the amount of time that has elapsed until a target state is reached. Depending on the concrete system, the elapsed time may be considered as a cost or as a reward. For example, it is common to try to minimize the response time of a server, whereas the expected time until a failure occurs should be maximized. Note also that in real-time systems the goal is usually to compute absolute time values, whereas in other applications time is measured as a discrete number of steps, e.g. the number of communicated messages or the number of rounds until a goal state is reached.

Besides expected values related to time, a wide range of other quantitative measures can be derived using expected reachability analysis. In particular, expected costs include the utilization or energy consumption of (hardware) components, and the charges for using a third-party service. Expected rewards on the other hand include the collected fees for a provided service or collected advertising revenue.

Note that in models supporting nondeterminism, is is in general not possible compute exact values for expected costs and rewards, but only minimal and maximal values.

Expected Reachability Costs and Rewards Requirements

(R16) Logic for the formal specification of properties referring to expected costs and rewards, in particular for expected durations

(R17) Availability of model checking algorithms and verification tools for (R16)

2.3 Conclusions

In this chapter, we have identified 16 requirements for the modeling and analysis of probabilistic real-time systems with uncertainties. For the three main modeling dimensions, i.e., (1) the functional behavior, (2) the real-time behavior, and (3) the probabilistic behavior, we have shown that inherent nondeterminism which arises from incomplete knowledge leads to uncertainties. We have argued that for a realistic modeling it is crucial to incorporate these uncertainties. Consequently, the quantitative analysis methods must be also capable of dealing with the uncertainties in the model.

Chapter 3

Foundations: Interval Probabilistic Timed Automata

Automata are a standard approach for formally modeling and analyzing system behavior. As a relevant automata model in the context of this report, Probabilistic Timed Automata (PTA) [26] constitute an expressive model for real-time behavior with both probabilistic and nondeterministic parts. Interval Probabilistic Timed Automata (IPTA) [38] have been recently introduced as an extension of PTA and additionally allow to explicitly model an uncertainty for probabilities. In IPTA, probabilities for events are not given as fixed values, but as intervals. An important aspect of this model is that the actual probability for an event may change over time. In this chapter we give a detailed formal presentation of the syntax and semantics of the IPTA model.

3.1 Preliminaries

We now recall some preliminary notions including the concept of a discrete probability distribution and standard definitions from the theory of Timed Automata [2].

3.1.1 Discrete Probability Distributions

For a finite set S, $Dist(S)$ is the set of *probability distributions* over S, i.e., the set of functions $\mu : S \to [0,1]$, such that $\sum_{s \in S} \mu(s) = 1$. The *point distribution* μ_s^1 is the unique distribution on S with $\mu(s) = 1$.

3.1.2 Clocks, Valuations and Constraints

Let $\mathbb{R}_+$ denote the set of non-negative reals. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of variables in $\mathbb{R}_+$, called *clocks*. An $\mathcal{X}$-*valuation* is a map $v : \mathcal{X} \to \mathbb{R}_+$. For a subset $X \subseteq \mathcal{X}$, $v[X := 0]$ denotes the valuation v' with $v'(x) = 0$ if $x \in X$ and $v'(x) = v(x)$ if $x \notin X$. For $d \in \mathbb{R}_+$, $v + d$ is the valuation v'' with $v''(x) = v(x) + d$ for all $x \in \mathcal{X}$. A *clock constraint* ζ on $\mathcal{X}$ is an expression of the form $x \bowtie c$ or $x - y \bowtie c$ such that $x, y \in \mathcal{X}$, $c \in \mathbb{R}_+$ and $\bowtie \in \{\leq, <, >, \geq\}$, or a conjunction of clock constraints. A clock valuation v satisfies ζ, written as $v \triangleright \zeta$ if and only ζ evaluates to true when all clocks $x \in \mathcal{X}$ are substituted with their clock value $v(x)$. Let $CC(\mathcal{X})$ denote the set of all clock constraints over $\mathcal{X}$.

3.2 Syntax

In this section, we present the syntax of interval probabilistic timed automata and introduce a means to compose them. Our formalization is based on a syntactical notion of probability intervals.

Definition 3.1 (Interval distribution)
Let S be a finite set. A probability interval distribution *λ on S is a pair of functions $\lambda = \langle \lambda^\ell, \lambda^u \rangle$ with $\lambda^\ell, \lambda^u : S \to [0,1]$, such that $\lambda^\ell(s) \leq \lambda^u(s)$ for all $s \in S$ and furthermore:*

$$\sum_{s \in S} \lambda^\ell(s) \leq 1 \leq \sum_{s \in S} \lambda^u(s) \tag{3.1}$$

The set of probability interval distributions over S is denoted by $IntDist(S)$. The support *of λ is defined as the set $Supp(\lambda) = \{s \in S \mid \lambda^u(s) > 0\}$. Let λ_s^1 be the unique interval distribution that assigns $\langle 1,1 \rangle$ to s, and $\langle 0,0 \rangle$ to all $t \in S, t \neq s$.*

A probability interval distribution λ is a symbolic representation of the non-empty and in most cases infinite set of probability distributions that respect all lower and upper interval bounds, formally given by the set: $\{\ \mu \in Dist(S) \mid \forall s \in S : \ell(s) \leq \mu(s) \leq u(s)\ \}$. If clear from the context, we may abuse notation and identify λ with this set. Note that interval distributions are equivalent to the notion of *closed interval specifications* in [14]. However, the explicit definition using lower and upper interval bounds in our model enables a syntactical treatment of interval distributions such as the minimality notion defined in the following.

Definition 3.2 (Minimal interval distribution)
An interval distribution λ on S is called minimal *if for all $s \in S$ the following conditions hold:*

(1) $\lambda^u(s) + \sum_{t \in S, t \neq s} \lambda^\ell(t) \leq 1$

(2) $\lambda^\ell(s) + \sum_{t \in S, t \neq s} \lambda^u(t) \geq 1$

Minimal interval distributions have the property that the bounds of all intervals can be reached (but not necessarily at the same time). Even though minimality is formally not required in most of the properties that we consider, it is often a desirable requirement since it can serve as a sanity check for a specification. For instance, consider a specification for a communication protocol in which a message is send to either component A or B. Assume that the choice of the receiver is of probabilistic nature and that for both A and B the probability is given by the interval $[0.4, 0.5]$. Syntactically, this forms a correct interval distribution according to Definition 3.1. However, it is obvious that the lower bound of 0.4 can never be reached and that in fact, the probability for choosing A or B is exactly 0.5, respectively. The minimality condition in Definition 3.2 can be used to circumvent such design mistakes. Note also that it is always possible to derive a minimal interval distribution from a non-minimal one by *pruning* the interval bounds, e.g., by setting $\lambda^u(s) := 1 - \sum_{t \in S, t \neq s} \lambda^\ell(t)$ if condition (2) is violated for the state s. In the following we define our central notion of interval probabilistic timed automata (IPTA) [38].

Definition 3.3 (Interval probabilistic timed automaton)
An interval probabilistic timed automaton *$\mathcal{I} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob, \mathcal{L})$ consists of:*

- *a finite set of locations L with $L^0 \subseteq L$ the set of initial locations,*
- *a finite set of action $\mathcal{A}$,*
- *a finite set of clocks $\mathcal{X}$,*
- *a clock invariant assignment function $inv : L \to CC(\mathcal{X})$,*

- *an interval probabilistic edge relation* $prob \subseteq L \times CC(\mathcal{X}) \times IntDist(2^{\mathcal{X}} \times L)$*, and*
- *a labeling function* $\mathcal{L} : L \to 2^{AP}$ *assigning atomic propositions to locations.*

Example 3.4 (Interval probabilistic timed automaton)
Figure 3.1 depicts a graphical visualization of an IPTA which models a simple server. The set of locations is given by $L = \{l_1, l_2, l_3\}$ *with* $L_0 = \{l_1\}$ *the only initial location. We assume that the atomic propositions are the location names. The set of actions is given by* $\mathcal{A} = \{request, response\}$ *and there is a single clock* $\mathcal{X} = \{x\}$*. Nondeterministic choices are indicated by multiple outgoing edges from a location (this example does not contain any nondeterminism). Probabilistic choices are visualized using a forking of edges in filled circles. For example, the edge via the action* $request$ *targets* l_2 *with a probability in the interval* $[0.95, 1]$*, and* l_3 *with a probability in the interval* $[0, 0.05]$*. Note also that the* $request$*-edge resets the clock* x*. The location* l_2 *has a clock invariant* $x \leq 20$ *and the* $response$*-edge from* l_2 *to* l_1 *has a guard* $x \leq 20$*.* T *is an integer constant greater than 20 and denotes a timeout. Intuitively, the IPTA models a simple server which answers to requests with a probability of* 0.95 *or higher within 20 time units.*

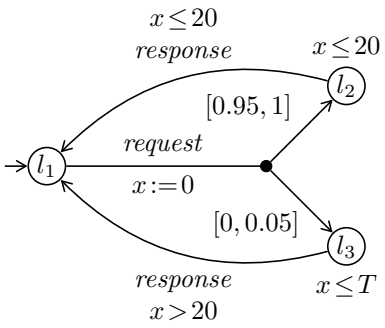

Figure 3.1: IPTA model of a simple server.

3.2.1 Costs and Rewards

In order to reason about expected costs (or rewards), IPTA can be equipped with a cost function which defines costs or rewards that accumulate as time passes. Formally, cost functions for IPTA are defined as follows (cf. [24]).

Definition 3.5 (Cost function)
Given an IPTA $\mathcal{I} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob, \mathcal{L})$*. A* cost function *for* $\mathcal{I}$ *is a pair* $(c, \mathbf{r})$ *consisting of:*

- $c : L \times \mathcal{A} \to \mathbb{R}_+$ *a function assigning the cost in each location in* L *of executing each action in* $\mathcal{A}$*,*
- $\mathbf{r} \in \mathbb{R}_+$ *the rate at which the costs are accumulated as time passes (independent of the location).*

Passed time is a special case of a cost or reward and can be formally defined by a cost function $(c, 1)$ where c assigns 0 to all location-action pairs. Also note that IPTA together with a cost function form a generalization of uniformly priced timed automata [3].

3.3 Semantics

In this section we give semantics of interval probabilistic timed automata in terms of two semantical models. On the one hand, semantics of IPTA can be given in terms of *Timed Interval Probabilistic Systems* (TIPS) [38], which are an extension of Interval Markov Decision Processes [31]. On the other, IPTA can be reduced to *Probabilistic Timed Automata* (PTA) [26], which we interpret here as IPTA with point intervals. Both semantics yield naive, i.e., infinite models. However, symbolic representations enable model checking and refinement analysis of IPTA as we will show later.

3.3.1 Timed Interval Probabilistic Systems

Definition 3.6 (Timed interval probabilistic system)
A timed interval probabilistic system $\mathcal{T} = (S, S^0, \mathcal{A}, Steps, \mathcal{L})$ *consists of:*

- *a set of states* S *with* $S^0 \subseteq S$ *the set of initial states,*
- *a set of actions* $\mathcal{A}$*, such that* $\mathcal{A} \cap \mathbb{R}_+ = \emptyset$*,*
- *a probability transition function* $Steps : S \to 2^{(\mathcal{A} \cup \mathbb{R}_+) \times IntDist(S)}$*, such that, if* $(a, \lambda) \in Steps(s)$ *and* $a \in \mathbb{R}_+$*, then* λ *is a point interval distribution, and*
- *a labeling function* $\mathcal{L} : S \to 2^{AP}$ *assigning atomic propositions to states.*

The operational semantics of a timed interval probabilistic system can be understood as follows. A *probabilistic transition*, written as $s \xrightarrow{a,\lambda,\mu} s'$, is made from a state $s \in S$ by:

(1) nondeterministically selecting an action/duration and interval distribution pair $(a, \lambda) \in Steps(s)$,

(2) nondeterministically choosing a probability distribution $\mu \in \lambda$,

(3) making a probabilistic choice of target state s' according to μ.

A *path* of a timed interval probabilistic system is a non-empty finite or infinite sequence of probabilistic transitions:

$$\omega = s_0 \xrightarrow{a_0,\lambda_0,\mu_0} s_1 \xrightarrow{a_1,\lambda_1,\mu_1} s_2 \xrightarrow{a_2,\lambda_2,\mu_2} \dots$$

where for all $i \in \mathbb{N}$ it holds that $s_i \in S$, $(a_i, \lambda_i) \in Steps(s_i)$, $\mu_i \in \lambda_i$ and $\mu_i(s_i) > 0$. We denote with $\omega(i)$ the $(i+1)$th state of ω, and with $last(\omega)$ the last state of ω, if it is finite. Moreover, we define $step(\omega, i) = a_i$ (the action or duration associated with the $(i+1)$th transition.

An *adversary* is a particular resolution of the nondeterminism in a timed interval probabilistic system $\mathcal{T}$. Formally, an adversary A for $\mathcal{T}$ is a function mapping every finite path ω of $\mathcal{T}$ to a triple (a, λ, μ), such that $(a, \lambda) \in Steps(last(\omega))$ and $\mu \in \lambda$. We restrict ourselves to *time-divergent* adversaries, i.e., we require that time has to advance beyond any given time bound. This is a common restriction in real-time models to rule out unrealizable behavior. The set of all time-divergent adversaries of $\mathcal{T}$ is denoted by $Adv_{\mathcal{T}}$.

For any $s \in S$ and adversary $A \in Adv_{\mathcal{T}}$, we define $Paths^A_{finite}(s)$ and $Paths^A_{full}(s)$ as the sets of all finite and infinite paths starting in s that correspond to A, respectively. Under a given adversary, the behavior of a timed interval probabilistic system is purely probabilistic. Formally, an adversary for a timed interval probabilistic system induces an infinite Discrete-Time Markov Chain (DTMC) and thus a probability measure $Prob^A_s$ on the set of paths $Paths^A_{full}(s)$ (cf. [18] for details).

In the following, we define the semantics of an interval probabilistic timed automaton in terms of a timed interval probabilistic system.

Definition 3.7 (TIPS semantics)
Let $\mathcal{I} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob, \mathcal{L})$ be an interval probabilistic timed automaton. The TIPS semantics *of $\mathcal{I}$ is the timed interval probabilistic system $\mathcal{T}_\mathcal{I} = (S, S^0, \mathcal{A}, Steps, \mathcal{L}')$ where:*

- $S \subseteq L \times \mathbb{R}_+^\mathcal{X}$, *such that $\langle l, v\rangle \in S$ if and only if $v \triangleright inv(l)$,*
- $S_0 = \{\ \langle l, v[\mathcal{X} := 0]\rangle \mid l \in L^0\ \}$,
- $\langle a, \lambda\rangle \in Steps(\langle l, v\rangle)$ *if and only if one of the following conditions holds:*
  - *Time transitions:* $a \in \mathbb{R}_+$, $\lambda = \lambda^1_{\langle l, v+t\rangle}$ *and* $v + t' \triangleright inv(l)$ *for all* $0 \leq t' \leq t$
  - *Discrete transitions:* $a \in \mathcal{A}$ *and there exists* $\langle l, \zeta, \hat{\lambda}\rangle \in prob$ *such that* $v \triangleright \zeta$ *and for any* $\langle l', v'\rangle \in S$ *it holds that:*
    - $\lambda^\ell(l', v') = \sum_{X \subseteq \mathcal{X} \wedge v' = v[X:=0]} \hat{\lambda}^\ell(X, l')$
    - $\lambda^u(l', v') = \sum_{X \subseteq \mathcal{X} \wedge v' = v[X:=0]} \hat{\lambda}^u(X, l')$,
- $\mathcal{L}'(\langle l, v\rangle) = \mathcal{L}(l)$ *for all* $\langle l, v\rangle \in S$.

3.3.2 Probabilistic Timed Automata

Alternatively to the TIPS semantics presented above, it is also possible to *flatten* an interval probabilistic timed automaton to a probabilistic timed automaton (PTA) [26]. In this approach, the choice for a particular probability distribution in an interval distribution is naturally encoded using nondeterminism, which is also available in PTA.

In our approach, we view probabilistic timed automata as special interval probabilistic timed automata in which all intervals are points.

Definition 3.8 (Probabilistic timed automaton)
An interval probabilistic timed automaton $\mathcal{P} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob, \mathcal{L})$ is also called probabilistic timed automaton *if and only if for all $\langle l, \zeta, a, \lambda\rangle \in prob$ it holds that λ contains only point intervals, i.e., $\lambda^\ell(X, l) = \lambda^u(X, l)$ for all $X \subseteq \mathcal{X}$ and $l \in L$.*

Definition 3.9 (PTA semantics)
Let $\mathcal{I} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob, \mathcal{L})$ be an interval probabilistic timed automaton. The PTA semantics *of $\mathcal{I}$ is the probabilistic timed automaton $\mathcal{P}_\mathcal{I} = (L, L^0, \mathcal{A}, \mathcal{X}, inv, prob', \mathcal{L})$ such that:*

$$\langle l, \zeta, a, \lambda\rangle \in prob \wedge \mu \in \lambda \quad \Leftrightarrow \quad \langle l, \zeta, a, \mu\rangle \in prob'$$

Thus, the PTA semantics of an IPTA simply replaces interval probabilistic choices by nondeterministic choices over the infinite set of probability distributions that respect the interval bounds.

PTA*-Encoding

The semantics of an interval distribution, i.e., the set of all probability distributions that respect the bounds of all its intervals, is in general infinite. However, it is possible to encode any finite IPTA into an equivalent finite PTA. This encoding, which we also refer to as PTA*, works as follows:

- The actions, clocks and locations of the PTA are the same as in the IPTA.
- For every transition $s \xrightarrow{a} \lambda$ in the IPTA we add all corners of λ interpreted as a polytope. Formally, for every ordering of the set $Supp(\lambda)$ we add the transition $s \xrightarrow{a} \mu_\lambda^{\max}$ to the PTA where $\mu_\lambda^{\max}$ is the probability distribution as defined in Section 5.2.2.

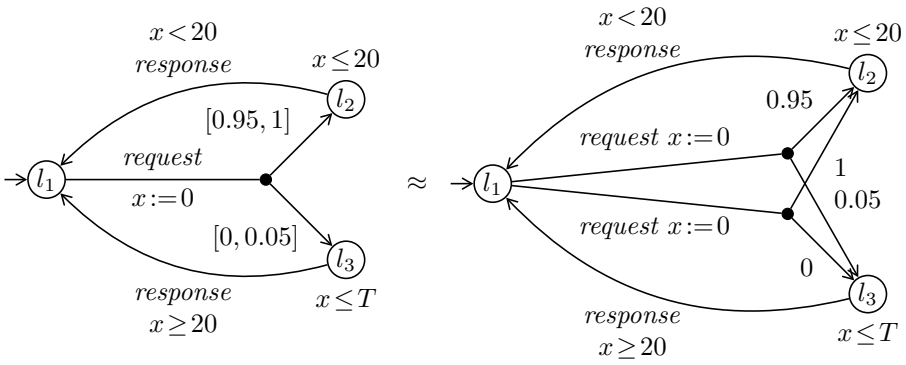

Figure 3.2: PTA*-encoding of a simple server IPTA.

Example 3.10 (PTA*-encoding)
Figure 3.2 depicts an example PTA encoding. The interval probabilistic edge via the intervals* $[0.95, 1]$ *and* $[0, 0.05]$ *is mapped to two probabilistic edges via 0.95 and 0.05, and 1 and 0, respectively.*

Note that the PTA*-encoding is similar to the MDP reduction of IMDPs in [31]. With respect to IPTA, we show the correctness of the encoding for probabilistic reachability properties in Lemma 5.1. However, note that the number of generated transitions in the PTA is exponential in the size of the support of the transition. Thus, there is a significant blow-up in the size of the model. We give an evaluation of the run-times of the different model checking approaches in Section 6.2.2.

Parallel Composition

Modular specification of systems is an important modeling feature since it offers the possibility to define complex systems by composing a set of more basic components. For probabilistic timed automata, a parallel composition operator (cf. [24]) facilitates modular specifications. In the following, we recall the definition of the parallel operator for PTA.

Definition 3.11 (Parallel composition [24])
Let $\mathcal{P}_i = (L_i, L_i^0, \mathcal{A}_i, \mathcal{X}_i, inv_i, prob_i, \mathcal{L}_i)$ *for* $i \in \{1, 2\}$ *be two probabilistic timed automata. Their* parallel composition *is defined as:*

$$\mathcal{P}_1 \parallel \mathcal{P}_2 = (L_1 \times L_2, L_1^0 \times L_2^0, \mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{X}_1 \cup \mathcal{X}_2, inv, prob, \mathcal{L})$$

such that

- $\mathcal{L}(\langle l_1, l_2\rangle) = \mathcal{L}_1(l_1) \cup \mathcal{L}_2(l_2)$ *for all* $l_1 \in L_1, l_2 \in L_2$,
- $inv(\langle l_1, l_2\rangle) = inv_1(l_1) \wedge inv_2(l_2)$ *for all* $l_1 \in L_1, l_2 \in L_2$,
- $\langle\langle l_1, l_2\rangle, \zeta, a, \lambda\rangle \in prob$ *if and only if one of the following conditions hold:*
  - *(1)* $a \in \mathcal{A}_1 \setminus \mathcal{A}_2$ *and there exists* $\langle l_1, \zeta, a, \lambda_1\rangle \in prob_1$ *such that* $\lambda = \lambda_1 \otimes \lambda^1_{\langle\emptyset, l_2\rangle}$
  - *(2)* $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$ *and there exists* $\langle l_2, \zeta, a, \lambda_2\rangle \in prob_2$ *such that* $\lambda = \lambda^1_{\langle\emptyset, l_1\rangle} \otimes \lambda_2$
  - *(3)* $a \in \mathcal{A}_1 \cap \mathcal{A}_2$ *and there exists* $\langle l_i, \zeta_i, a, \lambda_i\rangle \in prob_i$ *such that* $\lambda = \lambda_1 \otimes \lambda_2$ *and* $\zeta = \zeta_1 \wedge \zeta_2$

where for any $l_i \in L_i$, $X_i \subseteq \mathcal{X}_i$*:*

$$\lambda_1 \otimes \lambda_2(X_1 \cup X_2, \langle l_1, l_2\rangle) \stackrel{\text{def}}{=} \lambda_1(X_1, l_1) \cdot \lambda_2(X_2, l_2)$$

The parallel composition of probabilistic timed automata synchronizes transitions via shared actions, and interleaves all other actions. In the rest of this paper, we use parallel composition also for IPTA and define it by first applying the PTA*-encoding to both IPTA, and then applying the parallel operator for PTA as in Def. 3.11.

3.4 Conclusions

Interval probabilistic time automata are an expressive model for probabilistic real-time behavior with uncertainties. In this chapter, we have recalled the formal definitions for IPTA without discussing their features with respect to the requirements given in Chapter 2. A detailed discussion on this topic is given in Chapter 4.

Chapter 4

Modeling with Uncertainties

There exists a large variety of informal, semi-formal and formal models that can be used to describe service-oriented systems. In this chapter, we focus on formal, automata-based models which extend the basic modeling techniques by probabilistic real-time aspects. We present here their modeling concepts and describe how they can be exploited for service-oriented real-time systems modeling and discuss to what extent they cover the requirements in Chapter 2.

4.1 Basic Behavior Modeling with Uncertainties

Basic formal behavior modeling of service-oriented systems can be realized using labeled transitions systems. Labeled transition systems are particularly well-suited for modeling the control-flow, e.g., of a service and the communication with its environment. Multiple labeled transition systems can be composed using a binary parallel operator, often written as '$\|$'. This parallel operator combines the states of the two labeled transition systems pair-wise and allows to synchronize transitions with the same labels and to interleave transitions with unshared labels. We present a formal definition of a parallel operator for a more advanced automata model in Section 3.3.2.

Example 4.1 (Basic behavior modeling with uncertainties)
Figure 4.1 depicts a basic behavior model of a composite service-oriented system consisting of a client, a server and a database back-end. The client model describes a scenario in which the client sends an unbounded number of requests to the server and waits for responses. The client model includes uncertainty expressed using nondeterminism, since the number of requests (or if it terminates at all) is not specified.

The server reads a request and either immediately sends a response, or first queries a database and then sends the response. The database accepts queries, processes them and returns the result. Note that for simplicity, we model the communication between the components using mere synchronizations and in particular without modeling the direction and contents of the communication. Note also that this general approach also allows multi-party synchronizations. Furthermore, communication channels can be explicitly modeled using additional labeled transition systems, e.g. to specify buffered or blocking communication.[1]

[1] Lossy communication channels can be also modeled using labeled transition systems. However, the loss of a message can depend only on a deterministic or nondeterministic choice. For a more realistic modeling of lossy communication channels using probabilities, see Section 4.3.

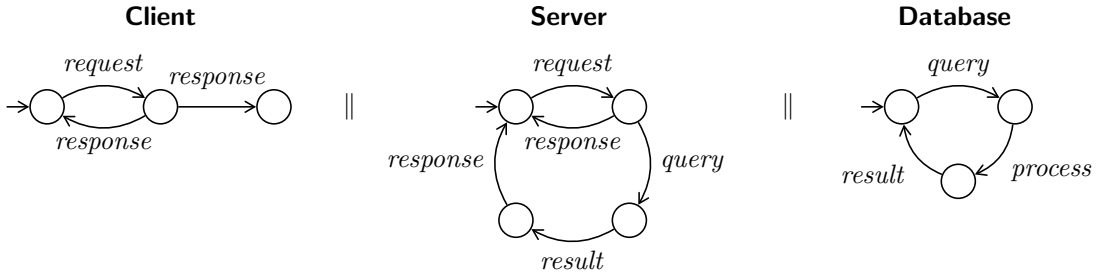

Figure 4.1: Basic behavior modeling using labeled transition systems.

We claim that labeled transition systems, as informally described here, provide a suitable, formal modeling language for basic behavior modeling of service-oriented systems with uncertainties. Uncertainties can be expressed only in qualitative way though, i.e., using nondeterminism. Labeled transition systems fulfill the requirements (R1)–(R6) in Section 2.1.1.

4.2 Real-Time Modeling with Uncertainties

The behavior modeling in labeled transition systems is essentially based on discrete, untimed execution steps and therefore does not account for modeling real-time behavior. To express real-time constraints, the formal model of Timed Automata [2] can be used which extends labeled transition systems by a concept for real-valued clocks which can be used to measure durations. The states in a timed automaton are referred to as *locations* and can be augmented with clock invariants which essentially model the allowed duration the automaton can stay in a location. The transitions in a timed automaton are called *edges* and can be annotated with 1) a clock guard that constrains the enabledness of an edge based on the current clock values, and 2) a number of clock resets which allow to model cyclic timed behavior by setting the value a clock back to 0.

Example 4.2 (Real-time modeling with Uncertainties)
Figure 4.2 depicts a client / server application modeled as a parallel composition of two timed automata. The client model contains a clock y and uses a positive constant T which denotes a time-out value. In the first step, the client spawns a request and resets its clock. The client allows to receive a response in less than T time units and fires a time-out transition if this is not possible within the deadline. The server uses its own clock x to measure time. On a request action the server nondeterministically either moves into state q_1 in which a response is guaranteed to be sent within 20 time units, or into state q_2 in which the answer takes longer than 20 time units or does not happen at all. Note that analogously to this example, timed automata can be used real-time communication channels.

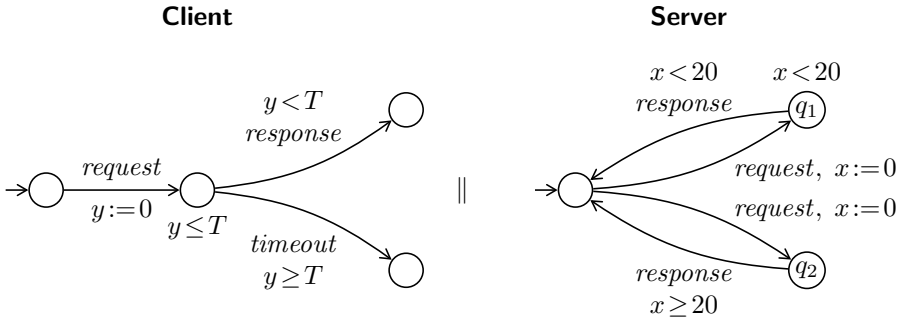

Figure 4.2: Real-time modeling of a client / server application using timed automata.

Timed automata are able to express real-time behavior using clock invariant and guards which can be used to model forced waiting times and deadlines. Since timed automata also inherit the modeling features of labeled transition systems, they support the requirements (R1)–(R10) in Chapter 2. In particular, uncertain real-time is supported since the triggering of an edge in an enabled time window is nondeterministic.

Example 4.2, however, also shows a major limitation of timed automata, i.e., their inability to model probabilistic behavior. Specifically, the decision in the server component whether the response can be sent within 20 time units or not is nondeterministic. Therefore, in this model it is not possible to distinguish between ideal servers which always answer on time and completely unreliable servers that never answer within the deadline. Similarly, the modeling of lossy communication channels using timed automata cannot be done in a realistic way.

4.3 Probabilistic Real-Time Modeling with Uncertainties

A realistic modeling of reliability properties of real-time systems requires not only the ability to express real-time constraints, but also concepts for expressing the likelihood of a particular timed behavior. This can be achieved by not only supporting nondeterministic, but also probabilistic choices. A formal model that supports probabilistic real-time modeling are Interval Probabilistic Timed Automata [38] (IPTA), which extend timed automata by probabilistic interval edges[2].

Example 4.3 (Probabilistic real-time modeling)
Figure 4.3 depicts an IPTA for a client / server application in which the server answers a request within 20 time units with a probability of at least 95%. For this purposes, the nondeterministic choice between the two request-edges in the server in Example 4.2 has been replaced by a single probabilistic interval edge with two target locations. Intuitively, on every incoming request, the server moves into state q_1 with a probability in the interval $[0.95, 1]$, and to state q_2 with a probability in the interval $[0, 0.05]$. Thereby, this IPTA server model accurately describes the response time guarantee in the service level agreement introduced in Section 4.3. Note also that in the same way lossy communication channels can be modeled using probabilities.

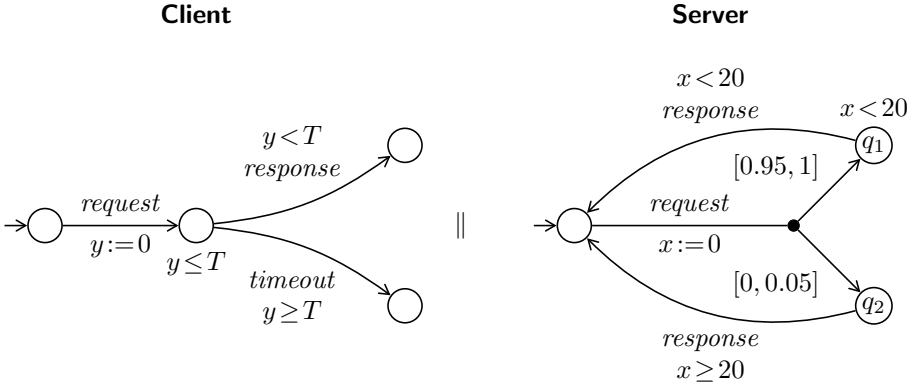

Figure 4.3: Probabilistic real-time modeling using interval probabilistic timed automata.

IPTA provide sufficient expressiveness to model uncertain probabilistic real-time behavior for service-oriented systems and particularly fulfill requirements (R1)–(R13) in Chapter 2. Uncertain probabilistic behavior is supported using probabilities intervals, as opposed to fixed probabilities. In the rest of this report, we use IPTA as our primary modeling language.

[2] The concept of probabilistic interval edges is based on Interval-Valued Discrete-Time Markov Chains [31] (IDTMCs).

Chapter 5

Quantitative Analysis

In this chapter, we formally define the property specification language for model checking IPTAs and sketch the symbolic algorithms for verifying probabilistic time-bounded reachability properties and expected reward properties.

5.1 Property Specification

Probabilistic real-time properties can be expressed using the formal specification language Probabilistic Timed Computation Tree Logic (PTCTL). Timing constraints in PTCTL are expressed using a set of *system clocks* $\mathcal{X}$, which are the clocks from the IPTA to be checked, and a set of *formula clocks* $\mathcal{Z}$, which is disjoint from $\mathcal{X}$. The syntax of PTCTL is defined as follows (cf. [26]):

$$\phi ::= a \mid \zeta \mid \neg\phi \mid \phi \vee \phi \mid z.\phi \mid \mathcal{P}_{\sim\kappa}[\phi \,\mathcal{U}\, \phi]$$

where:

- $a \in AP$ is an atomic proposition,
- $\zeta \in CC(\mathcal{X} \bigcup \mathcal{Z})$ is a clock constraint over all system and formula clocks,
- $z.\phi$ with $z \in \mathcal{Z}$ is a reset quantifier, and
- $\mathcal{P}_{\sim\kappa}[\_]$ is a probabilistic quantifier with $\sim \in \{\leq, <, >, \geq\}$ and $\kappa \in [0, 1]$ a probability threshold.

As an example for the specification of a combined probabilistic and timed property, the requirement for a bounded response time, e.g. 'with a probability of at least 95% a response is sent within 20ms' can be formalized in PTCTL as the formula:

$$z.\mathcal{P}_{\geq 0.95}[\mathrm{true}\,\mathcal{U}\,(\mathit{responseSent} \wedge z < 20)]$$

Furthermore, it is possible to specify properties over system clocks, e.g. the formula:

$$\mathcal{P}_{\leq 0.05}[(x \geq 4)\mathcal{U}(z = 8)]$$

represents the property 'with a probability of at most 5%, the system clock x exceeds 4 before 8 time units elapse'. In the remainder of this chapter, we focus on probabilistic reachability and expected values properties. For the complete formal semantics of PTCTL, we refer to [26].

5.2 Symbolic Model Checking

In this section, we recall the symbolic approach for PTCTL model checking, as introduced for probabilistic timed automata in [27], and adapted for interval probabilistic timed automata in [38].

5.2.1 Operations on Symbolic States

Since the timed interval probabilistic systems that we generate as semantics for an interval probabilistic timed automaton are in general infinite (cf. Definition 3.7), it is crucial to find a finite representation which can be used for model checking. For this purpose, a notion of symbolic state is considered in [27, 38], which is formally given by a pair (l, ζ) of a location l and a clock constraint ζ, also referred to as *zone* in this context. A symbolic state (l, ζ) is a finite representation of the set of state and formula clock valuations $\{ \langle\langle l, v\rangle, \mathcal{E}\rangle \mid v, \mathcal{E} \triangleright \zeta \}$. Based on this finite representation using the notion of zones, PTCTL model checking can be carried out by recursively evaluating the parse tree of a given formula, computing the set of reachable symbolic states.

5.2.2 Probabilistic Time-Bounded Reachability

The probabilistic quantifier $\mathcal{P}_{\sim\kappa}[\_]$ in PTCTL can be reduced to computing the minimum and maximum probabilities for reaching the given set of symbolic target states [27]. Computing these probabilities is known as the problem of *probabilistic reachability* and can be formally stated as follows.

Let A be an adversary for a timed interval probabilistic system $\mathcal{T} = (S, s_0, \mathcal{A}, Steps, \mathcal{L})$, and $F \subseteq S$ be a set of target states. The probability of reaching F from a state $s \in S$ is defined as:

$$p_s^A(F) = Prob_s^A\{\omega \in Paths_{full}^A(s) \mid \exists i \in \mathbb{N} : \omega(i) \in F\} \tag{5.1}$$

Then, the *minimal and maximal reachability probabilities* of F are defined as:

$$p^{\min}(F) = \inf_{A \in Adv_{\mathcal{T}}} p_{s_0}^A(F) \qquad p^{\max}(F) = \sup_{A \in Adv_{\mathcal{T}}} p_{s_0}^A(F) \tag{5.2}$$

A central property of interval probabilistic timed automata is that their infinite PTA semantics and symbolic TIPS semantics are equivalent with respect to probabilistic reachability. Therefore, it is valid to consider IPTA as a symbolic representation of an infinite, but equivalent, PTA.

Lemma 5.1 (Probabilistic reachability for TIPS and PTA semantics)
Given an interval probabilistic timed automaton $\mathcal{I}$. Let $\mathcal{T}_{\mathcal{I}} = (S, S^0, \mathcal{A}, Steps, \mathcal{L})$ be its TIPS semantics and $\mathcal{T}_{\mathcal{P}_{\mathcal{I}}} = (S, S^0, \mathcal{A}, Steps', \mathcal{L})$ the TIPS model of its PTA semantics $\mathcal{P}_{\mathcal{I}}$. Then for any $F \subseteq S$ the following holds:

$$p_{\mathcal{T}_{\mathcal{I}}}^{\min}(F) = p_{\mathcal{T}_{\mathcal{P}_{\mathcal{I}}}}^{\min}(F) \qquad \text{and} \qquad p_{\mathcal{T}_{\mathcal{I}}}^{\max}(F) = p_{\mathcal{T}_{\mathcal{P}_{\mathcal{I}}}}^{\max}(F) \tag{5.3}$$

Proof. *For any λ that occurs $\mathcal{T}_{\mathcal{I}}$ we know that $\mu_\lambda^{\max} \in \lambda$, and thus that $\mu_\lambda^{\max}$ occurs also in $\mathcal{T}_{\mathcal{P}_{\mathcal{I}}}$. Conversely, for any $\mu \in \lambda$ that occurs in $\mathcal{T}_{\mathcal{P}_{\mathcal{I}}}$ and any probability vector p over S, it holds that:*

$$\sum_{t \in Supp(\lambda)} \mu(t) \cdot p(t) \;\leq\; \sum_{t \in Supp(\lambda)} \mu_\lambda^{\max}(t) \cdot p(t)$$

Therefore $\mu_\lambda^{\max}$ suffices to compute the correct maximum probability of reaching F (analogously for the minimum probability). □

Iterative Algorithm

The minimum and maximum probabilities for a set of target states in a timed interval probabilistic system can be computed using an iterative algorithm [31, 38] known as *value iteration*, which is used to solve the *stochastic shortest path problem* [4] for (interval) Markov decision processes.

Let $\mathcal{T} = (S, S^0, \mathcal{A}, \mathit{Steps}, \mathcal{L})$ be a timed interval probabilistic system and $F \subseteq S$ be a set of target states. Moreover, let $\overline{F} \subseteq S$ be the set of states from which F cannot be reached. We define $(p_n)_{n\in\mathbb{N}}$ as the sequence of probability vectors over S, such that for any $s \in S$:

- $p_n(s) = 1$ if $s \in F$ for all $n \in \mathbb{N}$,
- $p_n(s) = 0$ if $s \in \overline{F}$ for all $n \in \mathbb{N}$,
- $p_n(s)$ is computed iteratively if $s \in S \setminus (F \cup \overline{F})$ by:

$$p_n(s) = 0$$
$$p_{n+1}(s) = \max_{(a,\lambda)\in \mathit{Steps}(s)} \sum_{t\in \mathit{Supp}(\lambda)} \mu_\lambda^{\max}(t) \cdot p_n(t)$$

where we consider an ordering $t_1, t_2, \ldots t_N$ of the set of states $\mathit{Supp}(\lambda)$, such that the vector $p_n(t_1), p_n(t_2), \ldots, p_n(t_N)$ is in descending order, and $\mu_\lambda^{\max}$ is given by:

$$\mu_\lambda^{\max}(t_m) = \min\left(\lambda^u(t_m), \left(1 - \sum_{i=1}^{m-1} \mu_\lambda^{\max}(t_i) - \sum_{i=m+1}^{N} \lambda^\ell(t_i)\right)\right)$$

with $m \in \{1, \ldots, N\}$.[1]

Then $p_n(s_0)$ converges to $p^{\max}(F)$ for $n \to \infty$. For a correctness proof of this algorithm see [38].

Complexity

Except for the additional sorting of the support states in the value iteration algorithm, the complexity for computing the maximum and minimum probabilities for IPTA is the same as for PTA. In general, PTCTL model checking of (Interval) Probabilistic Timed Automata is EXPTIME-complete. However, for certain subclasses of PTCTL the model checking problem can be shown to be PTIME-complete [15].

5.2.3 Expected Reachability Costs and Rewards

For IPTA equipped with cost functions as in Definition 3.5, it is desirable to compute the expected accumulated cost or reward until a set of target states is reached.

Given a timed interval probabilistic system $\mathcal{T} = (S, s_0, \mathcal{A}, \mathit{Steps}, \mathcal{L})$ generated from an IPTA, a set of target states $F \subseteq S$ and an adversary A for $\mathcal{T}$. For a cost function $(c, \mathbf{r})$ we define $\mathit{statecost}_{(c,\mathbf{r})}((l, v), a)$ for any location-clock valuation pair $(l, v) \in S$ and $a \in \mathcal{A} \cup \mathbb{R}_+$ as:

$$\mathit{statecost}_{(c,\mathbf{r})}((l, v), a) = \begin{cases} c(l, a) & \text{if } a \in \mathcal{A} \\ a \cdot \mathbf{r} & \text{otherwise} \end{cases}$$

[1] We define $\sum_{i=k}^{m} x \stackrel{\text{def}}{=} 0$ if $k > m$.

For any path $\omega \in Paths^A_{full}(s_0)$ let $pathcost(c, \mathbf{r})(\omega)$ be the total cost accumulated along ω until a state in F is reached:

$$pathcost(c, \mathbf{r})(\omega) = \sum_{i=1}^{\min(j \mid \omega(j) \in F)} statecost(\omega(i-1), step(w, i-1))$$

Now, the expected cost with respect to the probability measure $Prob^A_{s_0}$ is given by [24]:

$$e^A_{s_0}(c, \mathbf{r}) = \int_{\omega \in Paths^A_{full}(s_0)} pathcost(c, \mathbf{r})(\omega) dProb^A_{s_0}$$

The minimum and the maximum expected cost is respectively given by the infimum and the supremum of the expected costs with respect to all adversaries.

Computing Expected Reachability Costs and Rewards

Expected reachability costs and reward as defined above can be computed using an integral time model, specifically using the *digital clocks* semantics for PTA [24]. Note that this semantics has the restriction that strict comparisons in clock constraints are not allowed.

5.3 Conclusions

Symbolic representations of IPTA using digital clocks [24] and zones [27] facilitate the quantitative analysis of IPTA using probabilistic model checking. As specific analysis methods we considered the verification of probabilistic time-bounded reachability properties and the computation of expected costs and rewards for reaching a set of target states. Therefore, the described analysis methods provide a means for tackling requirements (R14)–(R17) as defined in Section 2.2. Note, however, due to the inherent nondeterminism in IPTA, only minimum and maximum probabilities and expected values can be computed.

Chapter 6

Tool Support and Evaluation

In this chapter, we describe a tool for the quantitative analysis of IPTA and give an evaluation of its modeling features and performance.

6.1 Tool Support

At the time of writing, PRISM 4.0 [23] was the latest version of the probabilistic model checker developed at the University of Oxford. For various probabilistic models including PTA, PRISM provides verification methods based on explicit and symbolic model checking, and discrete-event simulation. However, PRISM 4.0 does not support IPTA or any other interval-valued Markov chain models.

We have extended PRISM 4.0 with support for IPTA. Our tool is available at `www.mdelab.org/?p=50` and published under an open source license. In the rest of this section, we describe the usage of our tool for specifying and analyzing IPTA.

6.1.1 System Specification

Listing 6.1 contains the PRISM code for an extended version of the client / server application in Example 4.3. The specification consists of two modules which respectively model the IPTA of a server and a client. Note that since PRISM supports a variety of probabilistic models, it is necessary two specify the module type using the keyword `ipta` for interval probabilistic timed automata in line 1. The syntax for IPTA extends the syntax for PTA which is indicated using the module type `pta` as part of PRISM 4.0. In lines 3–6 we specified a number of constants to be used in the module definitions. These constants are usually set as command-line parameters when executing the model checker.

The server module in lines 8–19 is contains three locations specified using the variable `s` which ranges over $0 \dots 3$, and uses the clock `x` to measure durations. Additionally, the variable `w` is used to keep track of the number of slow requests that occurred during the execution.

For the definition of probability intervals as needed for IPTA, we added the new binary operator $\sim$ to the PRISM language which can be used in the specification of commands. A command in the IPTA version of PRISM corresponds to a probabilistic interval edge and takes the form

$$\texttt{[action] guard} \rightarrow \texttt{interval}_1\texttt{:update}_1 \texttt{ + ... + } \texttt{interval}_n\texttt{:update}_n\texttt{;}$$

Listing 6.1: Client / server example in the IPTA version of PRISM

```
1  ipta
2
3  const double L; // Lower probability bound for normal response times
4  const double U; // Upper probability bound for normal response times
5  const int REQUESTS; // Number of requests
6  const int TIMEOUT = 1000; // Timeout value
7
8  module Server
9    s : [0..2] init 0;
10   w : [0..REQUESTS] init 0; // Number of slow responses
11   x : clock;
12   invariant
13     (s=0 ⇒ x≤100) & (s=1 ⇒ x≤20) & (s=2 ⇒ x≤TIMEOUT)
14   endinvariant
15
16   [request] (s=0 & w<REQUESTS) → (L∼U):(s'=1)&(x'=0)
17                                + ((1-U)∼(1-L)):(s'=2)&(w'=w+1)&(x'=0);
18   [response] (s=1 & x≤20) | (s=2 & x>20) → (s'=0)&(x'=0);
19 endmodule
20
21 module Client
22   t : [0..REQUESTS] init 0;
23   y : clock;
24   invariant
25         (y<=TIMEOUT)
26   endinvariant
27
28   [request] t<REQUESTS → (t'=t+1)&(y'=0);
29   []        t=REQUESTS → (y'=0);
30 endmodule
31
32 label "lessThan50PercentSlow" = (t=REQUESTS & w<REQUESTS/2);
33
34 rewards
35    true : 1;
36 endrewards
```

where `action` denotes the label of the edge, `guard` is a condition that restricts the enabledness of the edge, $\texttt{update}_i$ define the target locations and possible clock resets, and $\texttt{interval}_i$ are either probability intervals written as $\ell \sim u$, or fixed probabilities which are formally interpreted as point intervals (where the lower and upper bounds coincide). Therefore, any `pta` model is also a valid `ipta` model in our tool.

As described above, guards and updates in PRISM can be used respectively to restrict the enabledness of an edge or to specify clock resets for edges. Another important modeling feature of IPTA is the possibility to specify clock invariants for locations. In PRISM 4.0, clock invariants are specified using the keywords `invariant...endinvariant`, as shown in lines 12–14 and 24–26 of Listing 6.1.

The probabilistic real-time behavior of the server module in the example is specified using the two commands in line 16 and 18. In the initial location (`s=0`), the server can fire a `request`-action, resetting the clock (`x'=0`) and moving to the location (`s=1`) with a probability in the interval $[\texttt{L}, \texttt{U}]$, and to the location (`s=2`) with a probability in the interval $[1-\texttt{U}, 1-\texttt{L}]$. When moving to (`s=2`), the counter variable `w` for slow responses is increased using the update (`w'=w+1`). The initial state is reached

again using two `response`-edges. One of them leaves the location (`s=1`) in 20 time units or less, the other leaves (`s=2`) after more then 20 time units. Thus, the specification models a simple server with a probabilistic response time guarantee, i.e. the server responds to a request within 20 time units with a probability of at least `L`.

The client is specified in lines 21–30 and performs a pre-defined number of requests, given by the constant `REQUESTS`, and then terminates. This allows us to control and count the number of subsequent requests and (slow) responses and to reason about probabilities for specific scenarios, such as the probability that less than 50% of all requests will result in a slow response. This particular property is encoded using the label `lessThan50PercentSlow` in line 32. Note also that this definition of the client provides a convenient way to scale the size of the state space by increasing the number of requests, i.e. the constant `REQUESTS`. This is particularly useful for conducting benchmarks, e.g. for measuring the run-times of the model checker for different model sizes (cf. Section 6.2.2).

In order to be able to reason about expected times in the specified model, we additionally need to define a state reward, which is specified in lines 34–36. In the real-time models supported by PRISM, state rewards accumulate at a rate related to time elapsing. Therefore, to compute expected times later we simply associate a reward of 1 to all locations.

For the two modules defined in Listing 6.1, automatically PRISM forms the system to be analyzed as the parallel composition of the server and the client, (cf. Definition 3.11).

The PRISM specification language for PTA already provides an expressive formalism to model probabilistic real-time service-oriented systems. Adding support for interval-based definition of uncertain probabilistic behavior using the '$\sim$' operator, the IPTA extension of PRISM covers the relevant modeling concepts and fulfills the requirements (R1)–(R13) as defined in Chapter 2.

6.1.2 Property Specification and Verification

In addition to a detailed system modeling, the specification of properties and their analysis is vital. In particular, we have stated in requirements (R14)–(R17) that the specification and verification of probabilistic time-bounded reachability properties and expected rewards and costs are important for reasoning about safety and reliability in service-oriented real-time systems.

Probabilistic Time-Bounded Reachability

In PRISM, probabilistic reachability properties can be specified as

```
Pmin=? [ F target ]   and   Pmax=? [ F target ]
```

where `target` is a label for a set of target states, such as `"lessThan50PercentSlow"` in line 32 of Listing 6.1. The keyword `Pmin` refers to the minimum probability whereas `Pmax` to the maximum probability for reaching the target states, as formally defined in Section 5.2.2.

As an example, we computed the minimum and maximum probabilities for less than 50% of slow responses for 10 requests and lower and upper probability bound of `L=0.7` and `U=0.8`. The minimum probability for this property was computed as 0.8497316674 and the maximum probability as 0.9672065024 by the IPTA version of PRISM.

Formulas for querying minimum and maximum probabilities as above can be also extended using time bounds, thus allowing to specify time-bounded probabilistic rechability properties, as discussed in Section 2.2.1. The PRISM syntax for such properties is

`Pmin=? [ F≤T target ]` and `Pmax=? [ F≤T target ]`

where `T` is a time bound. For our example, we computed the probabilities for the server having processed 5 or more requests within 200 time units, i.e. the property `Pmin/Pmax=? [ F≤200 (t≥5)&(s=0) ]`. The minimum probability was computed as 0.20664 and the maximum probability as 0.66384 for this time-bounded property.

Regarding the implementation, PRISM includes an engine which implements an algorithm for probabilistic reachability analysis of PTA based on stochastic two-player games [22] per default. For IPTA, we extended this engine by adding support for interval edges and adapting the value iteration algorithm as described in Section 5.2.2. Thus, our extension of PRISM can be used to specify and verify probabilistic time-bounded reachability properties as required in (R14)–(R15).

Expected Reachability Rewards and Costs

The syntax for querying expected rewards in PRISM is analogously to the notation for probabilistic reachability properties. To compute the minimum and the maximum expected reward, the following formulas can be used:

`Rmin=? [ F target ]` and `Rmax=? [ F target ]`

where `target` specifies a set of target states. For example, the expected time for the server having processed 5 requests can be calculated using the formula `Rmin/Rmax=? [ F (t=5)&(s=0) ]`. Using the IPTA-version of PRISM we computed the minimum expected time as 1.0 and the maximum expected time as 97.5 for this property.

PRISM currently supports expected rewards and costs only using an engine based on the digital clocks semantics [24] of PTA. To enable the computation of expected values for IPTA, we extended this engine also with support for interval edges. Thus, our extension of PRISM can be used for specifying and verifying expected costs and rewards for IPTA as required in (R16)–(R17).

6.2 Evaluation

To evaluate the correctness and the performance of the IPTA extension of PRISM, we conducted a number of experiments whose results are summarized in the following.

6.2.1 Difference to Sampling

An important aspect of the interval-based specification of uncertain probabilities is the fact that the semantics of IPTA allows to choose different specific values in the intervals during the execution. We argue here that therefore, the probabilistic behavior in IPTA cannot be mimicked using sampling in the probability intervals.

To support the above claim, we conducted an experiment where we computed the minimum and the maximum reachability probabilities for a property of the client/server example in Listing 6.1 where we set `L=0.7`, `U=0.8` and `REQUESTS=2`. Thus, the model is essentially the same as the parallel composition of IPTA in Example 4.3 with the difference that the client makes two requests and the server has slightly different failure probabilities. Using the IPTA version of PRISM, we then calculated the minimum and

maximum probabilities for the property that one out of two responses was slow: `(t=2 & w=1)`. The computed minimum and maximum probabilities are:

$$p^{\min}_{\texttt{ipta}} = 0.30, \qquad p^{\max}_{\texttt{ipta}} = 0.45$$

To illustrate the difference to approaches with fixed probabilities, we also encoded this example as a `pta` model, where we tested the following probabilities for normal response times: y=0.7, 0.75 and 0.8. For this model and the above property, we obtain the following probabilities:

$$p^{(y=0.7)}_{\texttt{pta}} = 0.42 \qquad p^{(y=0.75)}_{\texttt{pta}} = 0.375 \qquad p^{(y=0.8)}_{\texttt{pta}} = 0.32$$

It is obvious that these three samples are not sufficient to obtain the actual minimum and maximum probabilities as predicted using the IPTA model. In fact, no fixed value for y in the interval $[0.7, 0.8]$ produces the correct results, because the probability for the chosen property is minimal / maximal when y is chosen differently for each request. To illustrate this situation we computed the solutions analytically, depicted in the graph in Figure 6.1.

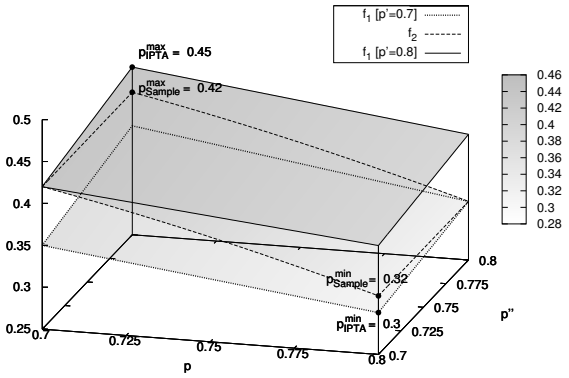

Figure 6.1: Analytic solutions for the property 'one out of two response is slow'

The plane in the middle represents the solution for the sampling-based `pta` approach, which reaches a minimum probability of 0.32 for y=0.7 and a maximum probability of 0.42 for y=0.8. The upper and lower plane depict the IPTA version which reaches a minimum and maximum probabilities of 0.3 and 0.45, respectively. Therefore, the sampling approach using PTA is not sufficient for determining the correct minimum and maximum probabilities in the original IPTA model.

6.2.2 Performance

In addition to the correctness tests of our implementation, we also investigated the model checking run-times of different encodings of the client / server example. To scale the application, we increased the number of requests performed by the client in our running example and compared the different run-times of PRISM.

Table 6.1 summarizes the run-times of our IPTA version of PRISM for three different encodings of the above example:

(1) PTA: sampling where a single probability distribution in the interval distribution is tested;

(2) IPTA: the original model as in Listing 6.1;

(3) PTA*: the encoding of the original IPTA as described in Section 3.3.2;

The measured run-times indicate that the checking of the PTA version was the fastest. However, we have shown in Section 6.2.1 that such a naive analysis using sampling does not produce the correct results.

While the PTA* version yields the correct results, the numbers show that the direct checking of the IPTA is more efficient. This is due to the fact the number of transitions to be checked in PTA* encoding is higher than in the original IPTA. The actual numbers of the transitions in the example are listed in Table 6.2. Note that in the simple client / server example, the support sets of the transitions are very small (of size 1 or 2). We expect that with a greater branching of transitions, the performance loss using the PTA* encoding gets significantly worse.

#Requests	#States	PTA	IPTA	PTA*
10	235	0.752	0.804	0.816
20	865	2.274	2.625	2.888
30	1,895	7.274	7.818	9.225
40	3,325	19.170	21.662	25.990
50	5,155	43.573	47.908	57.847

Table 6.1: Runtime in seconds for computing minimum probabilities for 'less than 50% slow responses'

#Requests	PTA	IPTA	PTA*
10	339	339	521
20	1,269	1,269	2,031
30	2,799	2,799	4,541
40	4,929	4,929	8,051
50	7,659	7,659	12,561

Table 6.2: Number of transitions for different encodings of the client/server example

6.3 Conclusions

In this chapter, we have introduced an extension of the PRISM tool for modeling and analysis of IPTA. For this purpose, we have (1) extended the modeling language of PRISM with an operator for specifying probability intervals and (2) adapted the model checking algorithms for probabilistic reachability and expected costs and rewards. Moreover, we have shown using an example that naive sampling of probabilities in intervals does not yield the correct results. Using benchmarks, we have shown that the PTA*-encoding as presented in Section 3.3.2 produces a significant blow-up of the semantic model to be analyzed which also results in longer run-times of the model checker.

Chapter 7

Related Work

7.1 Probabilistic Timed Automata

Probabilistic reachability and expected reachability analysis for PTA based on an integral model of time (digital clocks) is studied in [24]. The zone-based algorithm for symbolic PTCTL model checking of PTA is introduced in [27]. We rely on both approaches since probabilistic reachability analysis can be more efficiently carried out using the zone-based approach, while computing expected values related to time requires the digital clocks semantics of PTA in the current version of the PRISM tool.

Probabilistic timed simulation and bisimulation for PTA and an EXPTIME-algorithm to decide whether two PTA are (bi)similar are discussed in [35]. A notion of probabilistic time-abstracting bisimulation for PTA is introduced in [5]. An abstraction technique for MDPs based on stochastic two-player games can be found in [22].

Continuous Probabilistic Timed Automata [25] provide a means for modeling stochastic phenomena in probabilistic real-time models. However, since only decidability results and no practical tool support is readily available for this model, it is not applicable in our work.

The PRISM tool [23] natively supports model checking of probabilistic timed automata using two engines, respectively based on a digital clocks [24] and a stochastic two-player game semantics [22]. IPTA can be either encoded into PTA and then analyzed or directly modeled and mode efficiently analyzed using the IPTA extension of PRISM presented in this report. For an overview of other tools that support verification of PTA we refer to the related tools section in [23].

7.2 Interval Probabilistic Systems and Logics

Interval-based probabilistic models and their use for specification and refinement / abstraction have been studied already in '91 in [14]. PCTL model checking of interval Markov chains is introduced in [31]. Symbolic model checking for IPTA is presented in [38] based on the approaches in [27, 31]. However, no tool support or evaluation is given. Moreover, we show here that IPTA can also be encoded into PTA and provide empirical evidence that a direct modeling and analysis of IPTA is more efficient than encodings in PTA.

Probabilistic temporal interval networks [30] are constraint satisfaction problems in which the nodes

model temporal intervals and the edges probabilistic interval relations. In [30], a path consistency algorithm is presented, which – similarly to our approach – uses a lexicographical order on interval relations. However, probabilistic reachability is not considered.

Probabilistic interval temporal logic [10] is a probabilistic extensions of interval temporal logic (ITL) and the duration calculus (DC) with infinite intervals. The work approach consists of a Hilbert-style proof systems for such logics.

An overview of real-time and probabilistic temporal logics is given in [20] and includes a number of interval temporal logics such as ITL, and probabilistic logics, such as PCTL* and the probabilistic μ-calculus.

7.3 Quantitative Analysis of Service-Oriented Systems

Quality prediction of service compositions based on probabilistic model checking with PRISM is suggested in [9]. A comparison of different QoS models for service-oriented systems and an extension of the UML for quantitative models is given in [16]. A formal syntax for service level agreements of web services can be given using WSLA [17, 7]. In the area of channel-based coordination of services, a compositional QoS model and analysis using PRISM is presented in [28], and a performance analysis approach based on discrete event simulation is introduced in [37].

Chapter 8

Conclusions and Future Work

One of the aims of this report was to show that formal models, specifically with the introduction of Interval Probabilistic Timed Automata (IPTA), have reached a level of expressiveness in the last five years that allows to model service-oriented real-time systems including their inherent uncertainties. These uncertainties arise from a number of unavoidable phenomena, such as incomplete knowledge and probabilistic / stochastic behavior. Since the formal modeling of such systems uncertain behavioral properties alone does not yield usable predictions, we have investigated the state of the art for analysis techniques for IPTA and their applicability to real-world scenarios.

We have characterized a class of service-oriented real-time systems using a catalogue of requirements for both their modeling and their quantitative analysis. As one of the applications of our approach, we have shown how IPTA can be analyzed to make predictions for safety and reliability properties, such as probabilistic response-time guarantees. Furthermore, IPTA are sufficiently expressive to describe uncertain behavior due to hardware failure for on-request-based scenarios, i.e. where the probability of a hardware failure does not depend on the passage of time.

We have shown in this report that the quantitative analysis of important properties, such as safety and reliability properties for the targeted class of systems and settings can be formalized and performed automatically using tools. As a concrete contribution of this report, we have developed an extension of the PRISM model checker which allows to automate the formal verification of IPTA. We have shown the applicability of our approach and tool using benchmarks and compared our modeling approach using IPTA with models that only allow fixed probabilities. Additionally, we have shown that probabilistic uncertainty modeled using probability intervals is a key ingredient for specifying quantitative guarantees about the non-functional properties of services as commony described in service level agreements. Moreover, since IPTA inherit the expressiveness of timed automata, the approach can be applied to a wide range of systems – from services with uncertain QoS properties to hard real-time systems.

As future work, we plan to incorporate structure dynamics into the modeling approach in order to deal with dynamically changing architectures, e.g. due to rebinding of service endpoints or reconfiguration at run-time. Such characteristics require behavioral models based on local rewriting of structures rather than finite-state automata models. Therefore, we plan to develop a modeling approach based on rewrite systems, such as the rewrite logic in MAUDE [6] or typed attributed graph transformation systems [8]. Probabilistic and timed extensions of these approaches (see, e.g., [1, 29, 12, 11]) will provide rigorous methods for a quantitative modeling and analysis of the targeted class of service-oriented systems with structure dynamics.

In order to achieve better scalability and to analyze open systems, e.g., due to late binding of service implementations, we additionally aim at developing compositional reasoning schemes. For this purpose,

we plan to exploit variants of probabilistic timed simulation relations [35] for checking compliance of concrete services with their abstract specifications, and to investigate their compositionality and the preservation of relevant classes of quantitative properties. In the context of service-oriented systems, abstractions will furthermore provide a suitable concept for a formal account of service contracts. Compositional analysis approaches will moreover allow us to lift properties shown for a composition of abstract service models to the level of a concrete system.

Bibliography

[1] G. Agha, J. Meseguer, and K. Sen. PMaude: Rewrite-based specification language for probabilistic object systems. *Electron. Notes Theor. Comput. Sci.*, 153:213–239, 2006. DOI: 10.1016/j.entcs.2005.10.040.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. DOI: 10.1016/0304-3975(94)90010-8.

[3] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, and J. Romijn. Efficient guiding towards cost-optimality in uppaal. In *Proc. TACAS'01*, LNCS 2031, pages 174–188. Springer, 2001. DOI: 10.1007/3-540-45319-9_13.

[4] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. DOI: 10.1287/moor.16.3.580.

[5] T. Chen, T. Han, and J. P. Katoen. Time-abstracting bisimulation for probabilistic timed automata. In *TASE'08*, pages 177–184. IEEE Comp. Soc., 2008. DOI: 10.1109/TASE.2008.29.

[6] M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002. DOI: 10.1016/S0304-3975(01)00359-0.

[7] A. Dan, R. Franck, A. Keller, R. King, and H. Ludwig. Web Service Level Agreement (WSLA) language specification. URL: http://www.research.ibm.com/wsla/documents.html, 2002.

[8] H. Ehrig, U. Prange, and G. Taentzer. Fundamental theory for typed attributed graph transformation. In *Graph Transformations*, LNCS 3256, pages 161–177. Springer, 2004. DOI: 10.1007/978-3-540-30203-2_13.

[9] S. Gallotti, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Quality prediction of service compositions through probabilistic model checking. In *QoSA'08*, LNCS 5281, pages 119–134. Springer, 2008. DOI: 10.1007/978-3-540-87879-7_8.

[10] D. P. Guelev. Probabilistic interval temporal logic and duration calculus with infinite intervals: Complete proof systems. *Logical Methods in Computer Science*, 3(3), 2007. DOI: 10.2168/LMCS-3(3:3)2007.

[11] S. Gyapay, D. Varró, and R. Heckel. Graph transformation with time. *Fundamenta Informaticae*, 58:1–22, 2003.

[12] R. Heckel, G. Lajios, and S. Menge. Stochastic graph transformation systems. *Fundamenta Informaticae*, 74:63–84, 2006. DOI: 1231199.1231203.

[13] IEEE 1394–1995. URL: http://standards.ieee.org, 1995. Institute of Electrical and Electronics Engineers. IEEE Standard for a High Performance Serial Bus.

[14] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *LICS'91*, pages 266–277. IEEE Comp. Soc., 1991.

[15] M. Jurdzinski, J. Sproston, and F. Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Log. Meth. in Comp. Sci.*, 4(3), 2008. DOI: 10.2168/LMCS-4(3:12)2008.

[16] I. Jureta, C. Herssens, and S. Faulkner. A comprehensive quality model for service-oriented systems. *Software Quality Journal*, 17:65–98, 2009. DOI: 10.1007/s11219-008-9059-2.

[17] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11:57–81, 2003. DOI: 10.1023/A:1022445108617.

[18] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Springer, 2nd edition, 1976.

[19] S. Kemper. Compositional construction of real-time dataflow networks. In *Proc. COORDINATION'10*, LNCS 6116, pages 92–106. Springer, 2010. DOI: 10.1007/978-3-642-13414-2_7.

[20] S. Konur. Real-time and probabilistic temporal logics: An overview. *CoRR*, abs/1005.3200, 2010.

[21] C. Krause and H. Giese. Model checking probabilistic real-time properties for service-oriented systems with service level agreements. In *Proc. INFINITY'11*, EPTCS 73, pages 64–78. Open Publishing Association, 2011. DOI: 10.4204/EPTCS.73.8.

[22] M. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for Markov decision processes. In *Proc. QEST'06*, pages 157–166. IEEE Computer Society, 2006. DOI: 10.1109/QEST.2006.19.

[23] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. CAV'11*, LNCS 6806, pages 585–591. Springer, 2011. DOI: 10.1007/978-3-642-22110-1_47.

[24] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29:33–78, 2006. DOI: 10.1007/s10703-006-0005-2.

[25] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *CONCUR'00*, LNCS 1877, pages 123–137. Springer, 2000. DOI: 10.1007/3-540-44618-4_11.

[26] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282:101–150, 2002. DOI: 10.1016/S0304-3975(01)00046-9.

[27] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. Comput.*, 205:1027–1077, 2007. DOI: 10.1016/j.ic.2007.01.004.

[28] Y.-J. Moon, A. Silva, C. Krause, and F. Arbab. A compositional model to reason about end-to-end QoS in stochastic Reo connectors. *Science of Computer Programming*, 2011. DOI: 10.1016/j.scico.2011.11.007.

[29] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation*, 20:161–196, 2007. DOI: 10.1007/s10990-007-9001-5.

[30] V. Ryabov and A. Trudel. Probabilistic temporal interval networks. *Proc. TIME'04*, pages 64–67, 2004. DOI: 10.1109/TIME.2004.1314421.

[31] K. Sen, M. Viswanathan, and G. Agha. Model-checking Markov chains in the presence of uncertainties. In *Proc. TACAS'06*, LNCS 3920, pages 394–410. Springer, 2006. DOI: 10.1007/11691372_-26.

[32] SoaML specification. URL: `http://www.omg.org/spec/SoaML`.

[33] J. Sproston. *Model Checking for Probabilistic Timed and Hybrid Systems*. PhD thesis, School of Computer Science, University of Birmingham, 2001.

[34] J. Sproston. Model checking for probabilistic timed systems. In *Validation of Stochastic Systems*, LNCS 2925, pages 299–316. Springer, 2004. DOI: `10.1007/978-3-540-24611-4_6`.

[35] J. Sproston and A. Troina. Simulation and bisimulation for probabilistic timed automata. In *Proc. FORMATS'10*, LNCS 6246, pages 213–227. Springer, 2010. DOI: `10.1007/978-3-642-15297-9_17`.

[36] M. Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78:176–198, 2002.

[37] C. Verhoef, C. Krause, O. Kanters, and R. v. d. Mei. Simulation-based performance analysis of channel-based coordination models. In *Proc. COORDINATION'11*, LNCS 6721, pages 187–201. Springer, 2011. DOI: `10.1007/978-3-642-21464-6_13`.

[38] J. Zhang, J. Zhao, Z. Huang, and Z. Cao. Model checking interval probabilistic timed automata. In *Proc. ICISE'09*, pages 4936–4940. IEEE Comp. Soc., 2009. DOI: `10.1109/ICISE.2009.749`.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
55	978-3-86956-169-1	**Proceedings of the 4th Many-core Applications Research Community (MARC) Symposium**	Peter Tröger, Andreas Polze (Eds.)
54	978-3-86956-158-5	**An Abstraction for Version Control Systems**	Matthias Kleine, Robert Hirschfeld, Gilad Bracha
53	978-3-86956-160-8	**Web-based Development in the Lively Kernel**	Jens Lincke, Robert Hirschfeld (Eds.)
52	978-3-86956-156-1	**Einführung von IPv6 in Unternehmensnetzen: Ein Leitfaden**	Wilhelm Boeddinghaus, Christoph Meinel, Harald Sack
51	978-3-86956-148-6	**Advancing the Discovery of Unique Column Combinations**	Ziawasch Abedjan, Felix Naumann
50	978-3-86956-144-8	**Data in Business Processes**	Andreas Meyer, Sergey Smirnov, Mathias Weske
49	978-3-86956-143-1	**Adaptive Windows for Duplicate Detection**	Uwe Draisbach, Felix Naumann, Sascha Szott, Oliver Wonneberg
48	978-3-86956-134-9	**CSOM/PL: A Virtual Machine Product Line**	Michael Haupt, Stefan Marr, Robert Hirschfeld
47	978-3-86956-130-1	**State Propagation in Abstracted Business Processes**	Sergey Smirnov, Armin Zamani Farahani, Mathias Weske
46	978-3-86956-129-5	**Proceedings of the 5th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
45	978-3-86956-128-8	**Survey on Healthcare IT systems: Standards, Regulations and Security**	Christian Neuhaus, Andreas Polze, Mohammad M. R. Chowdhuryy
44	978-3-86956-113-4	**Virtualisierung und Cloud Computing: Konzepte, Technologiestudie, Marktübersicht**	Christoph Meinel, Christian Willems, Sebastian Roschke, Maxim Schnjakin
43	978-3-86956-110-3	**SOA-Security 2010 : Symposium für Sicherheit in Service-orientierten Architekturen ; 28. / 29. Oktober 2010 am Hasso-Plattner-Institut**	Christoph Meinel, Ivonne Thomas, Robert Warschofsky et al.
42	978-3-86956-114-1	**Proceedings of the Fall 2010 Future SOC Lab Day**	Hrsg. von Christoph Meinel, Andreas Polze, Alexander Zeier et al.
41	978-3-86956-108-0	**The effect of tangible media on individuals in business process modeling: A controlled experiment**	Alexander Lübbe
40	978-3-86956-106-6	**Selected Papers of the International Workshop on Smalltalk Technologies (IWST'10)**	Hrsg. von Michael Haupt, Robert Hirschfeld
39	978-3-86956-092-2	**Dritter Deutscher IPv6 Gipfel 2010**	Hrsg. von Christoph Meinel und Harald Sack
38	978-3-86956-081-6	**Extracting Structured Information from Wikipedia Articles to Populate Infoboxes**	Dustin Lange, Christoph Böhm, Felix Naumann

ISBN 978-3-86956-171-4
ISSN 1613-5652