Formal Verification with Confidence Intervals to Establish Quality of Service Properties of Software Systems

Radu Calinescu, *Senior Member, IEEE,* Carlo Ghezzi, *Fellow, IEEE,* Kenneth Johnson, *Member, IEEE,* Mauro Pezzé, *Senior Member, IEEE,* Yasmin Rafiq, *Student Member, IEEE,* and Giordano Tamburrelli

***Abstract*—Formal verification is used to establish the compliance of software and hardware systems with important classes of requirements. System compliance with functional requirements is frequently analysed using techniques such as model checking, and theorem proving. In addition, a technique called quantitative verification supports the analysis of the reliability, performance, and other quality-of-service (QoS) properties of systems that exhibit stochastic behaviour. In this paper, we extend the applicability of quantitative verification to the common scenario when the probabilities of transition between some or all states of the Markov models analysed by the technique are unknown, but observations of these transitions are available. To this end, we introduce a theoretical framework, and a tool chain that establish confidence intervals for the QoS properties of a software system modelled as a Markov chain with uncertain transition probabilities. We use two case studies from different application domains to assess the effectiveness of the new quantitative verification technique. Our experiments show that disregarding the above source of uncertainty may significantly affect the accuracy of the verification results, leading to wrong decisions, and low-quality software systems.**

***Index Terms*—Quantitative verification, probabilistic model checking, quality-of-service requirements, software systems, Markov chains.**

Acronyms and Abbreviations

FACT	Formal verificAtion with Confidence inTervals
HTTP	HyperText Transfer Protocol
IDTMC	Interval-valued Discrete-Time Markov Chain
LWB	Low-power Wireless Bus
MC	Markov Chain
PCTL	Probabilistic Computation Tree Logic
PMC	Parametric Markov Chain
QoS	Quality of Service
UMC	Uncertain Markov Chain
WSN	Wireless Sensor Network

Notation

$\mathcal{M}$	Markov chain
s, s_1, s_2, …	states of a Markov chain
S	set of states of a Markov chain
n	number of states of a Markov chain
p_{ij}	probability of transition from state s_i to state s_j
$[\underline{p}_{ij}, \overline{p}_{ij}]$	confidence interval for transition probability p_{ij}
$1-\alpha_i$	confidence level of simultaneous confidence intervals $[\underline{p}_{i1}, \overline{p}_{i1}]$, $[\underline{p}_{i2}, \overline{p}_{i2}]$, …, $[\underline{p}_{in}, \overline{p}_{in}]$
π	path over a Markov chain
$Paths^{\mathcal{M}}(s)$	set of infinite paths over $\mathcal{M}$ that start with s
Pr_s	probability measure over $Paths^{\mathcal{M}}(s)$
AP	atomic proposition set
$\mathcal{P}$	probabilistic PCTL operator
$\mathcal{R}$	reward PCTL operator
$\bowtie$	relational operator (any of $<$, $\leq$, $\geq$, $>$)
p	probability
Φ, Φ_1, Φ_2	PCTL state formulae
$\Phi(p_{i_1 j_1}, \ldots)$	algebraic expression of state formula Φ
$\Phi_1 \mathrm{U} \Phi_2$	unbounded until PCTL formula
$\Phi_1 \mathrm{U}^{\leq k} \Phi_2$	bounded until PCTL formula, $k \in \mathbb{N}$
$\mathrm{F}\Phi$	future state PCTL formula
$\mathrm{F}^{\leq k}\Phi$	bounded future state PCTL formula, $k \in \mathbb{N}$
Ψ	PCTL path formula
$\models$	satisfaction relation over the states S and the paths $Paths^{\mathcal{M}}(s)$, $s \in S$, of a Markov chain
$[a, b]$	confidence interval for analysed QoS property
$1-\alpha$	confidence level for analysed QoS property
O	set of state transition observations
N	number of observations
$\mathrm{Prob}(x \in X)$	probability that $x \in X$
R1, **R2**, …	QoS requirements

R. Calinescu and Y. Rafiq are with the Department of Computer Science at the University of York, UK.

C. Ghezzi is with the Department of Electronics and Information at Politecnico di Milano, Italy.

K. Johnson is with the School of Computer and Mathematical Sciences at Auckland University of Technology, New Zealand.

M. Pezzè is with the University of Milano Bicocca, Italy, and the Università della Svizzera italiana, Switzerland.

G. Tamburrelli is with Vrije Universiteit, Amsterdam, Netherlands.

Manuscript received September 15, 2014.

I. Introduction

SOFTWARE finds many applications in business- and safety-critical systems in domains as diverse as e-commerce, healthcare, and defence. In these applications, software enables activities that are too complex, too costly, or too dangerous for humans to carry out alone. The consequences of failures in such *critical software* are enormous, and can include financial loss or loss of human lives. To avoid these consequences, critical software has to comply with strict functional and quality-of-service (QoS) requirements at all times.

Formal methods are among the most effective and widely used means for developing high-quality software systems.

They comprise mathematically based techniques with a track record of supporting multiple stages of the software lifecycle in real-world scenarios [1], [2]. Established techniques such as formal specification [3], design by contract [4], and model checking [5] focus primarily on modelling and analysing functional aspects of software systems. In contrast, a more recently introduced technique called *quantitative verification* [6] supports the analysis of reliability, performance, and other QoS properties of software. This technique models software behaviour using finite state transition systems such as Markov chains and probabilistic automata; and uses temporal logics extended with probabilities, costs, and rewards to express and analyse QoS properties of the modelled software. Examples of properties that can be established using quantitative verification include the probability that a fault occurs within a specified time period, and the expected response time of a software system in a given scenario.

Extensive research over the past decade has produced efficient quantitative verification algorithms for a wide range of probabilistic, nondeterministic, and timed automata models [7], [8], [9]. The implementation of these algorithms within *probabilistic model checkers* such as PRISM [10], [11], MRMC [12], [13], and Ymer [14] has led to a wide adoption of the technique in software engineering, and many other domains. In software engineering, quantitative verification has been proven effective in modelling and analysing reliability, performance, and cost-related QoS properties of software architectures both at design time [15], and more recently at run time [16], [17]. Successful applications range from service selection in service-oriented architectures [18], [19], and configuration of cloud-deployed software [20], [21], to QoS property analysis for software product lines [22], and dynamic power management [23].

Furthermore, recent research tackling the limited scalability of quantitative verification has produced several promising solutions. These solutions include parametric [24], assume-guarantee [25], [26], and incremental [27], [28] quantitative verification techniques; and approaches that reduce the size of quantitative verification problems by using caching, look ahead, and nearly-optimal reconfiguration [29].

Despite these advances, one problem remains largely unsolved. The use of quantitative verification assumes that the analysed models accurately reflect the actual behaviour of the real software. Like in the case of traditional model checking, it is typically possible to satisfy this assumption with respect to the structure of the models. However, the probabilities associated with the state transitions of quantitative verification models are much more difficult to establish correctly. The current practice of using their point estimates provided by domain experts, or inferred through *model fitting* [30] to log data or run-time observations, yields values affected by unquantified estimation errors. The verification compounds and propagates these errors, in ways that are unknown but likely to be significant, given the nonlinear characteristics of these models. This raises concerns about the validity of decisions based on quantitative verification results, and limits the applicability of the technique.

Our paper addresses this important limitation of quantitative verification. To this end, we introduce a new quantitative verification approach that generates confidence intervals for the analysed properties of a Markov chain (MC). Our Formal verificAtion with Confidence inTervals (FACT) approach analyses MCs with unknown state transition probabilities, when observations of these transitions are available from logs or run-time observations of the modelled system. Given a property of the modelled software system that is formally expressed in probabilistic temporal logic, and a confidence level $1-\alpha \in (0,1)$, FACT synthesises a $1-\alpha$ confidence interval for the property. As a result, FACT enables the rigorous analysis of the reliability, performance, cost, and other QoS properties of software systems when the transition probabilities between the states of the MCs used to model these systems are approximated by sets of observations. The main contributions of the paper include the following.

1) We devise the first theoretical framework for quantitative verification with confidence intervals.
2) We present a tool chain that implements the FACT theoretical framework.
3) We describe two case studies that show the effectiveness of the FACT approach on software systems from different application domains.

The rest of the paper is organised as follows. Section II provides the necessary background on Markov chains, probabilistic computation tree logic, and probabilistic model checking; and Section III introduces a running example that we use to illustrate the steps of our verification approach. Sections IV through V describe the FACT theoretical framework, and the tool chain we assembled to reify this theory. The evaluation of the FACT approach using two case studies from different application domains is presented in Section VI. This section is followed by a discussion of several limitations of our approach, and of possible ways to overcome them in Section VII. Next, Section VIII compares our approach with related research, and Section IX concludes the paper with a summary and a discussion of future work.

II. Preliminaries

A. Markov chains

Definition 1. A *Markov chain* (MC) over a set of atomic propositions AP is a tuple

$$\mathcal{M} = (S, s_1, \mathbf{P}, L) \tag{1}$$

comprising a finite set of states $S = \{s_1, s_2, \ldots, s_n\}$, an initial state s_1, a transition probability matrix $\mathbf{P} : S \times S \to [0,1]$, and a labelling function $L : S \to 2^{AP}$. For any states $s_i, s_j \in S$, $\mathbf{P}(s_i, s_j)$ represents the probability of transitioning from state s_i to state s_j, so $\sum_{s_j \in S} \mathbf{P}(s_i, s_j) = 1$. For simplicity, we will often use the notation $p_{ij} = \mathbf{P}(s_i, s_j)$.

A *path* π over $\mathcal{M}$ is a possibly infinite sequence of states from S such that, for any adjacent states s and s' in π, $\mathbf{P}(s, s') > 0$. The m-th state on a path π, $m \geq 1$, is denoted $\pi(m)$. Finally, for any state $s \in S$, $Paths^{\mathcal{M}}(s)$ represents the set of all infinite paths over $\mathcal{M}$ that start with state s.

To compute the probability that a Markov chain (1) behaves in a specified way when in state $s \in S$, we use a *probability measure* $\Pr_s$ defined over $Paths^{\mathcal{M}}(s)$ such that [31], [32]

$$\Pr_s(\{\pi \in Paths^{\mathcal{M}}(s) \mid \pi = s_{i_1}s_{i_2}s_{i_3}\ldots s_{i_m}\ldots\}) = p_{i_1i_2}p_{i_2i_3}\cdots p_{i_{m-1}i_m}, \tag{2}$$

where $\{\pi \in Paths^{\mathcal{M}}(s) \mid \pi = s_{i_1}s_{i_2}s_{i_3}\ldots s_{i_m}\ldots\}$ represents the set of all infinite paths that start with the prefix $s_{i_1}s_{i_2}s_{i_3}\ldots s_{i_m}$ (i.e., the *cylinder set* of this prefix). Further details about this probability measure and its properties are available from [31], [32].

Our FACT approach to formal verification operates with parametric Markov chains (also called *incomplete Markov chains* [33], or *uncertain Markov chains* [34]).

Definition 2. A *parametric Markov chain* (PMC) is a Markov chain (1) for which some of the transition probabilities p_{ij} are parameters with domain $[0, 1]$.

Note that PMC transition probabilities could also be specified as rational functions over a set of variables with domain $\mathbb{R}$ [35], [36], [37]. Because our approach involves the computation of confidence intervals for each unknown transition probability, we adopted the above definition, which treats transition probabilities as variables. We discuss the extension of FACT to PMCs with transition probabilities specified as rational functions in Section VII.

To extend the range of system properties verified using Markov chains, they can be augmented with *rewards*. These are nonnegative values associated with the states and transitions of an MC or PMC.

They correspond to system properties such as throughput or profit. Depending on the analysed property, these values can also be interpreted as costs. Examples of costs that can be expressed using reward-augmented MCs and PMCs include resource use, energy consumption, and price. Accordingly, the verification of reward-based properties aims to establish that costs comply with upper bounds specified by the system requirements, and rewards satisfy the required lower bounds.

Definition 3. A *reward structure* over a Markov chain $\mathcal{M} = (S, s_1, \mathbf{P}, L)$ is a pair of functions $(\underline{\rho}, \iota)$ comprising a *state reward function* $\underline{\rho} : S \to \mathbb{R}_{\geq 0}$ (a vector), and a *transition reward function* $\iota : S \times S \to \mathbb{R}_{\geq 0}$ (a matrix).

B. Probabilistic Computation Tree Logic

To formally verify properties of software systems, these properties are expressed using precise mathematical formalisms. Markov chains support the quantitative verification of QoS requirements expressed in *probabilistic computation tree logic* [38], [39], which is a temporal logic with the syntax defined below.

Definition 4. Let AP be a set of atomic propositions; and $a \in AP$, $p \in [0, 1]$, $k \in \mathbb{N}$, $r \in \mathbb{R}$, and $\bowtie \in \{\geq, >, <, \leq\}$. Then a *state formula* Φ, and a *path formula* Ψ in probabilistic computation tree logic (PCTL) are defined by the grammar

$$\Phi ::= true \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathcal{P}_{\bowtie p}[\Psi], \tag{3}$$

$$\Psi ::= X\Phi \mid \Phi \mathrm{U} \Phi \mid \Phi \mathrm{U}^{\leq k}\Phi; \tag{4}$$

and a *reward state formula* is defined by the grammar

$$\Phi ::= \mathcal{R}_{\bowtie r}[\mathrm{I}^{=k}] \mid \mathcal{R}_{\bowtie r}[\mathrm{C}^{\leq k}] \mid \mathcal{R}_{\bowtie r}[\mathrm{F}\Phi] \mid \mathcal{R}_{\bowtie r}[\mathrm{S}]. \tag{5}$$

State formulae include the logical operators $\wedge$ and $\neg$, which allow the formulation of disjunction ($\vee$), implication ($\Rightarrow$), and *false*. State formulae extend *computation tree logic* [40] by replacing the universal path quantifier $\mathcal{A}$ and the existential path quantifier $\mathcal{E}$ with the probabilistic operator $\mathcal{P}$, which specifies bounds on the probability of the system evolution.

The semantics of PCTL are defined with a satisfaction relation $\models$ over the states S and the paths $Paths^{\mathcal{M}}(s)$, $s \in S$, of an MC (1). Thus, $s \models \Phi$ means Φ is satisfied in state s, or Φ is true in state s. For any state $s \in S$, we have: $s \models true$; $s \models a$ iff $a \in L(s)$; $s \models \neg\Phi$ iff $\neg(s \models \Phi)$; and $s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$. A state formula $\mathcal{P}_{\bowtie p}[\Psi]$ is satisfied in a state s if the probability of the future evolution of the system satisfying Ψ satisfies $\bowtie p$:

$$s \models \mathcal{P}_{\bowtie p}(\Psi) \text{ iff } \Pr_s(\{\pi \in Paths^{\mathcal{M}}(s) \mid \pi \models \Psi\}) \bowtie p.$$

The semantics of the three path formulae from (4) are described below.

- The *next state* formula $X\Phi$ is satisfied by a path π iff Φ is satisfied in the next state of π (i.e., in state $\pi(2)$).
- The *bounded until* formula $\Phi_1 \mathrm{U}^{\leq k}\Phi_2$ is satisfied by a path π iff Φ_1 is satisfied in each of the first x states of π for some $x < k$, and Φ_2 is satisfied in the $(x + 1)$-th state of π.
- The *unbounded until* formula $\Phi_1 \mathrm{U} \Phi_2$ is satisfied by a path π iff Φ_1 is true in each of the first $x > 0$ states of π, and Φ_2 is true in the $(x+1)$-th state of π.

The notation $\mathrm{F}^{\leq k}\Phi \equiv true\mathrm{U}^{\leq k}\Phi$, and $\mathrm{F}\Phi \equiv true\mathrm{U}\Phi$ are used when the first part of a bounded until, and until formula, respectively, are $true$.

In addition, given a reward structure in the form from Definition 3, PCTL was extended with *reward constraints* that support the specification of both expected and cumulative rewards [41]. Thus, the reward operator $\mathcal{R}$ can be used to analyse the expected cost at timestep k ($\mathcal{R}_{\bowtie r}[\mathrm{I}^{=k}]$), the expected cumulative cost up to time step k ($\mathcal{R}_{\bowtie r}[\mathrm{C}^{\leq k}]$), the expected cumulative cost to reach a future state that satisfies a property Φ ($\mathcal{R}_{\bowtie r}[\mathrm{F}\Phi]$), and the expected steady-state reward in the long run ($\mathcal{R}_{\bowtie r}[\mathrm{S}]$). The model checking algorithms from [41] can be used to analyse these reward properties efficiently.

Further details about the semantics of PCTL, and reward-extended PCTL are available from [38], [39], and from [41], respectively.

C. Probabilistic model checking

Efficient *probabilistic model checkers* such as PRISM [11], MRMC [12], and Ymer [14] employ symbolic model checking algorithms to automate the analysis of PCTL-specified properties of MCs. These algorithms establish if a formula $\mathcal{P}_{\bowtie p}[\Psi]$ is satisfied by calculating the actual probability that Ψ is satisfied, and comparing it with the bound p. Therefore, calculating the actual probability does not add any complexity, and the extended PCTL syntax $\mathcal{P}_{=?}[\Psi]$ can be used to obtain this probability. The extended syntax is applicable to the outermost P operator of a $\mathcal{P}_{\bowtie p}[\Psi]$ formula, and is compliant with the input of most probabilistic model checkers.

This extended syntax also applies to reward PCTL formulae, for which $\mathcal{R}_{=?}[\mathrm{I}^{=k}]$, $\mathcal{R}_{=?}[\mathrm{C}^{\leq k}]$, $\mathcal{R}_{=?}[\mathrm{F}\Phi]$, and $\mathcal{R}_{=?}[\mathrm{S}]$ are used in a similar way. Note that PCTL formulae that use this extended syntax can define QoS attributes (or QoS properties) of the modelled system, while PCTL formulae that use a fixed bound can be used to define QoS requirements.

The traditional approach to verifying PCTL-encoded properties of a PMC involves using point estimates of its unknown transition probabilities, to derive an MC that *refines* the original PMC. This MC is then analysed using a probabilistic model checker as described above. The transition probability estimates can come from domain experts, or can be inferred from execution traces of the modelled system. In the latter case, the execution traces may be obtained from system logs [42], [43], or through system run-time monitoring [44].

D. Parametric model checking

Related research proposed an alternative approach to verifying PMCs termed *parametric model checking* [24], [35], [36], [37], [45], [46]. Given a PCTL property Φ, parametric model checking produces a symbolic expression of Φ in which the PMC parameters appear as variables. The symbolic expression is a multivariate rational function, and its synthesis may be computationally expensive. However, this synthesis is performed only once, after which the expression can be used for multiple purposes, e.g., for trend as well as sensitivity or perturbation analysis [47], and to evaluate the QoS property associated with Φ at run time, when the values of the PMC parameters are determined [24].

Research on parametric model checking has led to significant advances over the past decade. The first approach to parametric model checking is due to Daws [45]. This approach uses a language-theoretic technique to convert the PMC into a finite automaton that is then used to obtain a regular expression whose recursive evaluation yields the required symbolic expression. The approach works for reachability properties without nested probabilistic operators, but is limited to PMCs with low numbers of states n because the regular expression is of size $n^{\Theta(\log n)}$. This limitation is alleviated in [35] by using a combination of techniques for state-space reduction, and for the early evaluation of the regular expression. This approach outperforms Daws' original solution in many scenarios of practical relevance, although its worst-case performance is the same as in [45].

The recent research in [37] significantly improves the efficiency of parametric model checking through using a new state elimination strategy based on recursively decomposing the analysed PMC into strongly connected components, and a new method for executing operations such as the greatest common divisor directly on partial factorisations of polynomials. The experimental results presented in [37] show that the new approach speeds up parametric model checking by several orders of magnitude compared to previous solutions.

Finally, parametric model checking is supported by the latest version of the probabilistic model checker PRISM, as well as by dedicated verification tools such as PARAM [46].

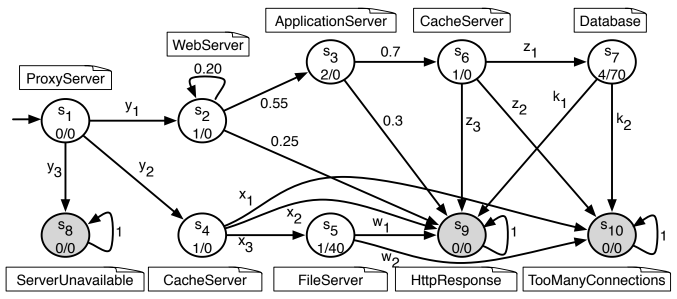

Fig. 1. PMC $\mathcal{M}_{\mathrm{webApp}}$ modelling the handling of an HTTP request.

III. Running Example

We will illustrate the steps of our FACT approach using a business-critical web application comprising an HTTP proxy server, a web server, and an application server. To serve client requests, the web application accesses structured data stored in a database, and static content (e.g., text files, and images) located on a file server. Both types of content are cached by ad-hoc cache servers.

The parametric Markov chain in Fig. 1 models the functionality that handles an HTTP request within the web application. Each PMC state represents a stage of the handling process. The initial state s_1 corresponds to the request being received, and the shaded states are *absorbing states*, i.e., states that once entered cannot be left. These states indicate the outcome of the request handling, i.e., whether the request handling succeeds (state s_9) or fails due to an unavailable server (state s_8), or to the overloading of a component of the application (state s_{10}). The states s_2 to s_7 correspond to the web application performing the operations indicated by their labels.

The transitions between the *transient states* s_1 through s_7, and between these states and the absorbing states, model the control flow for handling a request. For example, the transitions (s_1, s_2), and (s_1, s_4) model the events of dynamic content has been requested, and static content has been requested, respectively; and the self-transition (s_2, s_2) corresponds to an HTTP self-redirect. The probabilities of the outgoing transitions from states s_1 and s_4 through s_7 are unknown, or may change over time. For example, the transitions (s_4, s_9), and (s_6, s_9) model the cache hit for the file server, and the database, respectively, both of which have probabilities that depend on the (unknown) distribution of the user requests.

The non-absorbing states of the PMC are labelled by a pair of numbers in the form n_1/n_2. The values n_1 and n_2 represent the *average cost* of the operation associated with the non-absorbing state in tenths of a cent, and its *average duration* in milliseconds, respectively. These values define two reward structures over the PMC, a cost structure (n_1), and a response time structure (n_2), where the reward values not shown in Fig. 1 (i.e., those associated with the PMC absorbing states and transitions) are all zero.

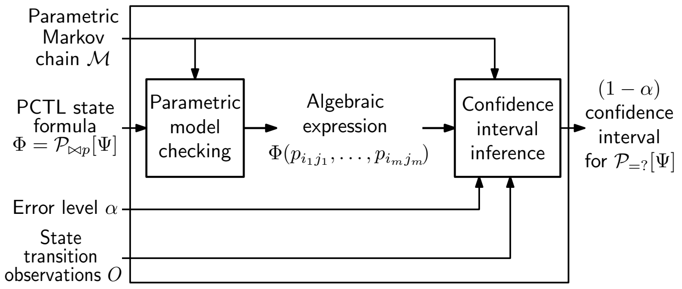
Parametric Markov chain $\mathcal{M}$
PCTL state formula $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$
Error level α
State transition observations O
Parametric model checking
Algebraic expression $\Phi(p_{i_1j_1},\ldots,p_{i_mj_m})$
Confidence interval inference
$(1-\alpha)$ confidence interval for $\mathcal{P}_{=?}[\Psi]$

Fig. 2. FACT confidence interval synthesis.

IV. Approach

A. Overview

Our FACT approach establishes confidence intervals for PCTL-formalised QoS properties of a software system, and checks whether these confidence intervals satisfy the constraints specified by the QoS requirements of the system. The four inputs taken by FACT are shown in Fig. 2, and are described below.

1) The first input is a parametric Markov chain $\mathcal{M} = (S, s_1, \mathbf{P}, L)$ of the analysed system.
2) The second input is a PCTL state formula Φ for a QoS requirement of interest; Φ can be a probabilistic state formula $\mathcal{P}_{\bowtie p}[\Psi]$ or a reward formula (5).
3) The third input is an error level $\alpha \in (0, 1)$.
4) The final input is an observation function $O : S \to \mathbb{N}^n$ that maps each state $s_i \in S$ to a tuple

$$O(s_i) = (N_{i1}, N_{i2}, \ldots, N_{in}) \tag{6}$$

such that N_{ij} represents the number of transitions from state s_i to state s_j, $1 \le j \le n$, during a fixed period of time over which all transitions were observed and counted. We will sometimes refer to O as the *observation set*, in the sense that it represents the set of mappings $O = \{s_1 \mapsto O(s_1), s_2 \mapsto O(s_2), \ldots, s_n \mapsto O(s_n)\}$.

For the analysis of a PCTL state formula $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$, FACT synthesises a $(1-\alpha)$ confidence interval $[a, b] \subseteq [0, 1]$ for the probability that Ψ is satisfied in the initial state s_1 given the observations O. Thus, if our FACT procedure is applied repeatedly under identical conditions, a fraction of $(1-\alpha)$ of the generated confidence intervals will contain the value $\mathrm{Pr}_{s_1}(\{\pi \in Paths^{\mathcal{M}}(s_1) \mid \pi \models \Psi\})$. We write formally

$$\mathrm{Prob}\left(\mathrm{Pr}_{s_1}(\{\pi \in Paths^{\mathcal{M}}(s_1) \mid \pi \models \Psi\}) \notin [a, b]\right) < \alpha.$$

Next, FACT establishes if $[a, b] \bowtie p$ using the rules below, which extend the application of the relational operators $<$, $\le$, $\ge$, and $>$ to the comparison between an interval $[a, b] \subseteq \mathbb{R}$ and a scalar $x \in \mathbb{R}$:

$$\begin{array}{l} [a,b] < x \text{ iff } b < x, \\ [a,b] \le x \text{ iff } b \le x, \\ [a,b] \ge x \text{ iff } a \ge x, \text{ and} \\ [a,b] > x \text{ iff } a > x. \end{array} \tag{7}$$

We can now define the following concepts.

Definition 5. If the $(1-\alpha)$ confidence interval $[a, b]$ calculated above satisfies $[a, b] \bowtie p$, we write

$$s_1 \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi] \tag{8}$$

to indicate that $\mathcal{P}_{\bowtie p}[\Psi]$ is satisfied with confidence $(1-\alpha)$ in state s_1, given the observations O. We will sometimes write

$$\mathcal{M} \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi] \tag{9}$$

to indicate that, given the observations O, $\mathcal{P}_{\bowtie p}[\Psi]$ is satisfied with confidence level $(1-\alpha)$ in the initial state of $\mathcal{M}$.

We handle the analysis of a reward formula $\mathcal{R}_{\bowtie r}[\cdot]$ from (5) similarly. First, we synthesise a $(1-\alpha)$ confidence interval $[a, b] \subseteq \mathbb{R}_+$ for the expected reward $\mathcal{R}_{=?}[\cdot]$. Next, we establish if $[a, b] \bowtie r$, in which case we use the notation $\mathcal{M} \models_{\alpha,O} \mathcal{R}_{\bowtie r}[\cdot]$ to indicate that the expected reward satisfies $\bowtie r$ with probability at least $(1-\alpha)$ under the observations O.

Given a model $\mathcal{M}$, a formula Φ, an error level α, and a set of observations O with the characteristics described above, the FACT synthesis of $(1-\alpha)$ confidence intervals for Φ is performed in two stages (Fig. 2).

1) *Parametric model checking*. In this stage, FACT generates an algebraic expression for Φ by using parametric model checking as explained in Section II-C. This algebraic expression is a multivariate rational function $\Phi(p_{i_1,j_1}, p_{i_2,j_2}, \ldots, p_{i_mj_m})$ that depends on some or all unknown transition probabilities of $\mathcal{M}$ [36], i.e., $1 \le m \le n^2$, and $1 \le i_k, j_k \le n$, for $1 \le k \le m$.
2) *Confidence interval inference*. In its second stage, FACT uses the observations O to obtain confidence intervals for the unknown transition probabilities that appear in the algebraic expression, and uses them to derive the required confidence interval for Φ. The calculations carried out in this stage are based on the FACT theoretical framework presented in the next section.

Example 1. Suppose that we want to establish with 95% confidence if the web application from our running example satisfies the QoS requirement that at least 2% of the requests are handled without accessing the web server or the file server (e.g., to decide if maintaining a cache server for web pages is justified). Also, assume that we monitored the behaviour of the system, and collected observations of all its outgoing transitions from the states of the PMC $\mathcal{M}_{\text{webApp}}$ in Fig. 1 that are associated with unknown transition probabilities.

We can address this problem by performing the confidence interval synthesis from Fig. 2 for the four inputs described below.

1) The parametric Markov chain $\mathcal{M}$ is $\mathcal{M}_{\text{webApp}}$.
2) The PCTL state formula Φ is $\mathcal{P}_{\ge 0.02}[\Psi]$, where

$$\Psi = \neg(\mathsf{WebServer} \vee \mathsf{FileServer})\, U\, \mathsf{HttpResponse}. \tag{10}$$

3) The error level is $\alpha = 0.05$.
4) The elements of the observation function $O : S \to \mathbb{N}^{10}$ are defined by the observed outgoing transitions of $\mathcal{M}_{\text{webApp}}$. As an example, assume that the observed outgoing transitions from state s_1 of $\mathcal{M}_{\text{webApp}}$ comprise 2705 transitions to s_2, 3174 transitions to s_4, and 5

transitions to s_8; and that the observed outgoing transitions from s_4 consist of 2975 transitions to s_5, 187 transitions to s_9, and 12 transitions to s_{10}. In these circumstances, $O(s_1) = (0, 2705, 0, 3174, 0, 0, 0, 5, 0, 0)$, and $O(s_4) = (0, 0, 0, 0, 2975, 0, 0, 0, 187, 12)$.

To establish if $\mathcal{M} \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi]$, the first FACT stage uses parametric model checking to obtain an algebraic expression for Φ. The only infinite path that satisfies our path property Ψ is $\pi = s_1 s_4 s_9 s_9 \ldots$. Therefore, according to (2), the algebraic expression for our PCTL state formula is $\Phi(x_2, y_2) = x_2 y_2$. This expression is used to compute a $(1-\alpha) = 0.99$ confidence interval for $\mathcal{P}_{=?}[\Psi]$ in the second FACT stage. To describe this computation, we need the following theoretical framework.

B. Theoretical framework

We start by defining the concept of simultaneous confidence intervals for a multinomial distribution [48], [49], [50].

Definition 6. Given a multinomial distribution with $n > 1$ possible outcomes of probabilities $x_1, x_2, \ldots, x_n$, where $\sum_{i=1}^n x_i = 1$, and a value $\alpha \in (0, 1)$, the intervals $I_1, I_2, \ldots, I_n \subseteq [0, 1]$ are *simultaneous* $(1-\alpha)$ *confidence intervals* for $x_1, x_2, \ldots, x_n$ iff $\text{Prob}\,(x_1 \in I_1; x_2 \in I_2; \ldots; x_n \in I_n) \geq 1 - \alpha$.

We can now introduce a result that enables the derivation of a confidence interval for an algebraic expression from simultaneous confidence intervals for disjoint subsets of its parameters.

Proposition 1. *Let $\mathcal{M} = (S, s_1, \mathbf{P}, L)$ be a PMC over an atomic proposition set AP, and $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$ be a PCTL state formula over AP. Assume that, for each state $s_i \in S$ with unknown outgoing transition probabilities, $[\underline{p}_{i1}, \overline{p}_{i1}]$, $[\underline{p}_{i2}, \overline{p}_{i2}]$, ..., $[\underline{p}_{in}, \overline{p}_{in}]$ are* simultaneous $(1-\alpha_i)$ confidence intervals *for the transition probabilities from s_i to each of the n states in S. If the result of the parametric model checking of Φ is*

$$\text{Pr}_{s_1}(\{\pi \in \mathit{Paths}^{\mathcal{M}}(s_1) \,|\, \pi \models \Psi\}) = \Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}),$$

where $p_{i_1 j_1}$, $p_{i_2 j_2}$, ..., $p_{i_m j_m}$ are $m > 0$ unknown transition probabilities of $\mathcal{M}$, and

$$\begin{aligned} a &= \min_{\substack{\underline{p}_{i_1 j_1} \leq p_{i_1 j_1} \leq \overline{p}_{i_1 j_1} \\ \ldots \\ \underline{p}_{i_m j_m} \leq p_{i_m j_m} \leq \overline{p}_{i_m j_m} \\ \sum_{j=1}^n p_{ij} = 1 \, \mathit{for}\, i \in I}} \Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}) \\ b &= \max_{\substack{\underline{p}_{i_1 j_1} \leq p_{i_1 j_1} \leq \overline{p}_{i_1 j_1} \\ \ldots \\ \overline{p}_{i_m j_m} \leq p_{i_m j_m} \leq \overline{p}_{i_m j_m} \\ \sum_{j=1}^n p_{ij} = 1 \, \mathit{for}\, i \in I}} \Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}) \\ \alpha &= 1 - \textstyle\prod_{i \in I} (1 - \alpha_i) \end{aligned} \tag{11}$$

with $I = \{i_1, i_2, \ldots, i_m\}$, then $[a, b]$ is a $(1-\alpha)$ confidence interval for Φ, i.e.,

$$\text{Prob}\,(\Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}) \notin [a, b]) < \alpha.$$

Proof. For any $i \in I$, $[\underline{p}_{i1}, \overline{p}_{i1}]$, $[\underline{p}_{i2}, \overline{p}_{i2}]$, ..., $[\underline{p}_{in}, \overline{p}_{in}]$ are *simultaneous* $(1-\alpha_i)$ *confidence intervals* for p_{i1}, p_{i2}, ..., p_{in}, so

$$\forall i \in I \bullet \text{Prob}\left(\bigwedge_{(i,j) \in IJ} \underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \right) \geq 1 - \alpha_i,$$

where $IJ = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$. Given the memoryless property of Markov chains, the transitions from different states s_i are s-independent, so

$$\text{Prob}\left(\bigwedge_{i \in I} \bigwedge_{(i,j) \in IJ} \underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \right) \geq \prod_{i \in I} (1-\alpha_i) = 1 - \alpha.$$

Because additionally $\sum_{j=1}^n p_{ij} = 1$ is always true for any $i \in I$, we have $\text{Prob}(a \leq \Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}) \leq b) \geq 1 - \alpha$, which proves the proposition. $\square$

Example 2. Consider again the PMC $\mathcal{M}$ from Fig. 1, and the property Φ with the algebraic expression $\Phi(x_2, y_2) = x_2 y_2$ from Example 1. Suppose that for each state s_i with unknown outgoing transition probabilities from $\mathcal{M}$ we established $(1-\alpha_i)$ confidence intervals for the transition probabilities from s_i to other states of $\mathcal{M}$. The property we want to analyse depends on the transition probabilities x_2, and y_2, which correspond to outgoing transitions from the PMC states s_4, and s_1, respectively. We therefore focus our attention on these two states. Let the confidence intervals associated with state s_1 be $[\underline{y}_1, \overline{y}_1]$, $[\underline{y}_2, \overline{y}_2]$, and $[\underline{y}_3, \overline{y}_3]$ for y_1, y_2, and y_3, respectively; and the confidence intervals associated with state s_4 be $[\underline{x}_1, \overline{x}_1]$, $[\underline{x}_2, \overline{x}_2]$, and $[\underline{x}_3, \overline{x}_3]$ for x_1, x_2, and x_3. The three elements from (11) are

$$\begin{aligned} a &= \min_{\substack{\underline{x}_2 \leq x_2 \leq \overline{x}_2, \underline{y}_2 \leq y_2 \leq \overline{y}_2 \\ x_1 + x_2 + x_3 = 1, y_1 + y_2 + y_3 = 1}} x_2 y_2 = \underline{x}_2 \underline{y}_2, \\ b &= \max_{\substack{\underline{x}_2 \leq x_2 \leq \overline{x}_2, \underline{y}_2 \leq y_2 \leq \overline{y}_2 \\ x_1 + x_2 + x_3 = 1, y_1 + y_2 + y_3 = 1}} x_2 y_2 = \overline{x}_2 \overline{y}_2, \\ \alpha &= 1 - (1-\alpha_1)(1-\alpha_4). \end{aligned}$$

Thus, according to Proposition 1, $[\underline{x}_2 \underline{y}_2, \overline{x}_2 \overline{y}_2]$ is a $(1-\alpha)$ confidence interval for Φ.

We use the result from Proposition 1 to derive a sufficient condition for a PMC to satisfy a PCTL formula with a confidence level $(1-\alpha)$ under a set of observations O.

Proposition 2. *Let $\mathcal{M} = (S, s_1, \mathbf{P}, L)$ be a PMC over an atomic proposition set AP, O an observation function (6), $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$ a PCTL state formula over AP, and $\Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m})$ the result of the parametric model checking of Φ, where $p_{i_1 j_1}$, $p_{i_2 j_2}$, ..., $p_{i_m j_m}$ are unknown transition probabilities of $\mathcal{M}$ as before. For any state s_i associated with observations $O(s_i) = (N_{i1}, N_{i2}, \ldots, N_{in})$, and for any state $s_j \in S$, further let*

$$\underline{p}_{ij} = \left(f_{ij} + \frac{z^2_{\alpha_i/2}}{2N_i} - z_{\alpha_i/2} \sqrt{\frac{f_{ij}(1-f_{ij}) + z^2_{\alpha_i/2}/4N_i}{N_i}} \right) \cdot \frac{N_i}{N_i + z^2_{\alpha_i/2}},$$

and

$$\overline{p}_{ij} = \left(f_{ij} + \frac{z^2_{\alpha_i/2}}{2N_i} + z_{\alpha_i/2} \sqrt{\frac{f_{ij}(1-f_{ij}) + z^2_{\alpha_i/2}/4N_i}{N_i}} \right) \cdot \frac{N_i}{N_i + z^2_{\alpha_i/2}} \tag{12}$$

where $z_{\alpha_i/2}$ is the $(1-\alpha_i/2)$ quantile of the standard normal distribution, $N_i = \sum_{j=1}^{n} N_{ij}$, and $f_{ij} = N_{ij}/N_i$. Finally, suppose that the bounds $\underline{p}_{ij}$ and $\overline{p}_{ij}$ from (12), and the associated α_i values are used to calculate a, b, and α as in (11), and assume that $[a,b] \bowtie p$. Under these circumstances, $\mathcal{M} \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi]$.

Proof. The unknown outgoing transition probabilities p_{i1}, p_{i2}, $\ldots$, p_{in} of a PMC state s_i represent the success probabilities of a multinomial distribution with n success categories. According to [48], the intervals $[\underline{p}_{i1}, \overline{p}_{i1}]$, $[\underline{p}_{i2}, \overline{p}_{i2}]$, $\ldots$, $[\underline{p}_{in}, \overline{p}_{in}]$ defined by (12) are simultaneous $(1-\alpha_i)$ confidence intervals for the success probabilities of an n-category multinomial distribution for which N_i s-independent trials lead to N_{i1}, N_{i2}, $\ldots$, N_{in} observations of the n success categories. Therefore, $[\underline{p}_{i1}, \overline{p}_{i1}]$, $[\underline{p}_{i2}, \overline{p}_{i2}]$, $\ldots$, $[\underline{p}_{in}, \overline{p}_{in}]$ are simultaneous $(1-\alpha_i)$ confidence intervals for p_{i1}, p_{i2}, $\ldots$, p_{in}. It follows from Proposition 1 that

$$\text{Prob}\left(\Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m}) \notin [a,b]\right) < \alpha.$$

Because additionally $[a,b] \bowtie p$, we deduce that the probability that $\mathcal{P}_{\bowtie p}[\Psi]$ is not satisfied in state s_1 is below α, which according to Definition 5 is denoted

$$\mathcal{M} \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi].$$

□

Note that numerous studies propose and discuss different simultaneous confidence intervals for the success probabilities of multinomial distributions, e.g., [51], [49], [48], [50], [52]. The confidence intervals (12) were proposed by Kwong and Iglewicz [48], and achieve a good trade-off between computational complexity and precision, where by precision we mean the width of the confidence interval. Nevertheless, other results from literature can be substituted in (12) without affecting the validity of the FACT theoretical framework.

Example 3. We will use the result from Proposition 2 to compute a 95% confidence interval for the property $\Phi(x_2, y_2) = x_2 y_2$ from our running example. Given the observations O from Example 1, the number of outgoing transitions from states s_1, and s_4 are respectively

$$N_1 = 5884, \text{ and } N_4 = 3174; \tag{13}$$

and the frequencies for these transitions are

$$\begin{array}{lll} f_{1,2} = \frac{2705}{5884}, & f_{1,4} = \frac{3174}{5884}, & f_{1,8} = \frac{5}{5884}, \\ f_{4,5} = \frac{2975}{3174}, & f_{4,9} = \frac{187}{3174}, & f_{4,10} = \frac{12}{3174} \end{array}. \tag{14}$$

We use these frequencies and the result in (12) to calculate confidence intervals for the transition probabilities $x_2 = p_{4,9}$ and $y_2 = p_{1,4}$, which Φ depends on. We calculate the two confidence intervals with confidence levels

$$(1-\alpha_1) = (1-\alpha_4) = \sqrt{0.95}, \tag{15}$$

because this will support the computation of the confidence interval for Φ with confidence level $(1-\alpha) = (1-\alpha_1)(1-\alpha_4) = 0.95$, as established in Example 2. Using (12), and the values in (13) through (15) to calculate confidence intervals for x_2, and y_2, we obtain

$$\underline{x}_2 \approx 0.0386,\ \overline{x}_2 \approx 0.0805,\ \underline{y}_2 \approx 0.5069,\ \overline{y}_2 \approx 0.5718,$$

so $[\underline{x}_2 \underline{y}_2, \overline{x}_2 \overline{y}_2] \approx [0.0195, 0.0461]$ is a 0.95 confidence interval for Φ. Note that the 0.02 bound from the QoS requirement introduced in Example 1 belongs to this confidence interval. Thus, the confidence interval cannot be used to decide with 95% confidence whether our web application handles at least 2% of the requests without accessing the web server or the file server. A solution for reducing the size of the 0.95 confidence interval so that this decision can be made is presented in the remainder of this section.

Propositions 1 and 2 provide the means for calculating a $(1-\alpha) = \prod_{i \in I}(1-\alpha_i)$ confidence interval for a PCTL formula Φ from $(1-\alpha_i)$ confidence intervals for the unknown transition probabilities that Φ depends on. However, this result does not address the selection of suitable α_i values for a given PCTL formula Φ, and a given set of observations O. In other words, the results so far do not include a method for choosing effective α_i values starting from the $(1-\alpha)$ confidence level for which we want to obtain a confidence interval for Φ.

The naive method of choosing all $\alpha_i = 1-(1-\alpha)^{\frac{1}{\#I}}$ as in Example 3 is suboptimal.[1] To see this result, suppose that the algebraic expression for a PCTL formula associated with the PMC from our running example in Fig. 1 is $\Phi(x_1, y_2) = 0.9x_1 + 0.0001y_2$, and that we want to obtain a 0.95 confidence interval for Φ. Because x_1, and y_2 are transition probabilities associated with states s_4, and s_1, respectively, we need to select α_4 and α_1 such that $(1-\alpha_4)(1-\alpha_1) = 0.95$. A quick look at $\Phi(x_1, y_2)$ tells us that our formula is much more sensitive to the variation of x_1 than it is to the variation of y_2. Accordingly, a good choice of α_4 and α_1 is one that yields a narrow $(1-\alpha_4)$ confidence interval for x_1, at the expense of obtaining a much wider $(1-\alpha_1)$ confidence interval for y_2. Thus, the combination $(1-\alpha_4) = 0.951$ and $(1-\alpha_1) = 0.95/0.951 = 0.998$ will lead to a much narrow 0.95 confidence interval for Φ than the naive combination $(1-\alpha_4) = (1-\alpha_1) = \sqrt{0.95} = 0.974$. A similar analysis shows that this result is also true for the algebraic expression $\Phi'(x_1, y_2) = 0.5x_1 + 0.5y_2$ that is equally sensitive to the variation of x_1 and y_2, if the number of observations of transitions from states s_4 and s_1 satisfies $N_4 \ll N_1$.

To account for the logical dependencies discussed above, FACT uses the *hill-climbing optimisation* heuristic detailed in Algorithm 1. Given the algebraic expression $\Phi(p_{i_1 j_1}, p_{i_2 j_2}, \ldots, p_{i_m j_m})$ of a PCTL formula, a set of observations O, and an error level α, this heuristic synthesises a $(1-\alpha)$ confidence interval for Φ whose width is minimised by using hill climbing to adjust the confidence levels $(1-\alpha_i)$, $i \in I = \{i_1, i_2, \ldots, i_m\}$.

The algorithm works as follows. At any time during its execution, the local variables $bestA$ and $bestB$ store the bounds of the narrowest confidence interval identified by the heuristic. The two variables are initialised with bounds

[1]Note that $\#I$, the number of elements in $I = \{i_1, i_2, \ldots, i_m\}$, is below m if $i_1, i_2, \ldots, i_m$ are not all different.

Algorithm 1 Confidence interval synthesis.

```
function ConfInterval($\Phi(p_{i_1j_1},\ldots,p_{i_mj_m})$, $O$, $\alpha$)
    $bestA \leftarrow 0$, $bestB \leftarrow 1$
    for all $i \in I$ do
        $\alpha_i \leftarrow 1-(1-\alpha)^{\frac{1}{\#I}}$
    end for
    $noImprovementSteps \leftarrow 0$
    while $noImprovementSteps \le MAX\_STEPS$ do
        for all $(i,j) \in I \times \{1,2,\ldots,n\}$ do
            calculate $\underline{p}_{ij}$, $\overline{p}_{ij}$ in (12) for $O$ and $\alpha_i$
        end for
        calculate $a$, $b$ in (11)
        if $b-a < bestB - bestA$ then
            $bestA \leftarrow a$, $bestB \leftarrow b$
            $noImprovementSteps \leftarrow 0$
        else
            $noImprovementSteps++$
        end if
        randomly adjust $\alpha_i$, $i \in I$, s.t. $\prod_{i\in I}(1-\alpha_i) = 1-\alpha$
    end while
    return $[bestA, bestB]$
end function
```

corresponding to the most conservative confidence interval possible (i.e., $[0,1]$) in line 2, and their final values define the confidence interval returned by the algorithm in line 20. The for loop in lines 3 through 5 initialises the confidence levels $(1-\alpha_i)$, $i \in I$, with the same value $1-(1-\alpha)^{1/\#I}$.

As discussed above, these values typically produce suboptimal results. Therefore, the while loop in lines 7 through 19 uses hill climbing to adjust them to reduce the width of the confidence interval $[bestA, bestB]$ in a number of steps that involve

- calculating $(1-\alpha_i)$ confidence intervals $[\underline{p}_{ij}, \overline{p}_{ij}]$ for the unknown transition probabilities used in (11) (lines 8 through 10);
- obtaining the confidence interval $[a,b]$ corresponding to these confidence intervals (line 11);
- retaining the new confidence interval bounds a, b if they represent an improvement over the bounds $bestA$, $bestB$ (lines 12 and 13); and
- performing a random adjustment of the confidence levels $(1-\alpha_i)$, $i \in I$ (line 18), as explained shortly in this section.

The hill-climbing while loop in lines 7 through 19 is executed until no improvement over the current $bestA$, $bestB$ confidence interval bounds is found for MAX_STEPS consecutive steps, where MAX_STEPS is a parameter of the algorithm. To this end, the local variable $noImprovementSteps$ is used to count the number of consecutive steps that do not reduce the width of the confidence interval $[bestA, bestB]$. This variable is initialised in line 6, incremented in line 16, and reset to zero when a better confidence interval $[bestA, bestB]$ is identified (in line 14).

This completes the description of the algorithm, with the exception of the random adjustment of the α_i, $i \in I$, values in line 18. The technique that FACT uses to perform this adjustment involves randomly selecting two elements $i, i' \in I$, $i \neq i'$, and applying the change

$$\begin{array}{l} 1-\alpha_i \leftarrow (1-\alpha_i)x \\ 1-\alpha_{i'} \leftarrow (1-\alpha_{i'})/x \end{array},$$

where x is a random value drawn repeatedly from a uniform distribution over $[0.9, 1.1]$ until the new values of $(1-\alpha_i)$ and $(1-\alpha_{i'})$ are both in the interval $[0,1]$. Note that the value of the product $(1-\alpha_i)(1-\alpha_{i'})$ is not affected by this adjustment, so the property $\prod_{i\in I}(1-\alpha_i) = 1-\alpha$ (which is true after the for loop in lines 3 through 5) will continue to hold after each instance of this adjustment.

To analyse the time complexity of the algorithm, note that each iteration of the while loop in lines 7 through 19 involves the constant-time calculation of up to n^2 confidence intervals in lines 8 through 10, $O(1)$ operations in lines 12 through 17, the random adjustment of the α_i confidence levels in line 18, and the calculation of (11) in line 11. Unless n is extremely large, or the expression $\Phi(p_{i_1j_1}, p_{i_2j_2}, \ldots, p_{i_mj_m})$ in (11) is particularly simple, the time taken by the calculation of (11) dwarfs that of the other operations.

Example 4. Consider one last time the QoS requirement $\mathcal{P}_{\ge 0.02}[\Psi]$ from our running example, with Ψ given by (10). The 0.95 confidence interval $[0.0195, 0.0461]$ that we computed for $\mathcal{P}_{=?}[\Psi]$ in Example 3 was insufficient to establish with 95% confidence whether the requirement was satisfied or not. We therefore used the implementation of Algorithm 1 described in Section V to obtain a tighter confidence interval. The execution of the algorithm for $MAX\_STEPS = 30$ completed after 85 iterations of the while loop in lines 7 through 19, and we obtained the 10.07% narrower 0.95 confidence interval $[0.0208, 0.0449]$, so we can conclude with 95% confidence that the QoS requirement from our running example is satisfied.

Finally, remember that the aim of calculating the confidence interval returned in line 20 of Algorithm 1 is to establish whether $\mathcal{M} \models_{\alpha,O} \mathcal{P}_{\bowtie p}[\Psi]$ by checking if $[bestA, bestB] \bowtie p$ (cf. Definition 5). In case of a negative answer, there are two scenarios.

1) When $p \notin [bestA, bestB]$, p is on the wrong side of the interval $[bestA, bestB]$ with respect to rules (7), e.g., $p > bestB$ for $\bowtie = <$. In this scenario, $\mathcal{P}_{\bowtie p}[\Psi]$ is not satisfied with confidence level $(1-\alpha)$, given the observations O.
2) When $p \in [bestA, bestB]$, the confidence interval is not narrow enough to decide whether the property $\mathcal{P}_{\bowtie p}[\Psi]$ is satisfied or not with the required confidence level, and given the available observations. In this case, additional observations can be used to obtain a narrower confidence interval, which may allow a decision to be made. The width of the confidence interval defined by the bounds (12) is strictly decreasing with the number of observations, and converges asymptotically to zero, so it is theoretically possible to obtain a $[bestA, bestB]$ confidence interval that is as narrow as desired. However, this action may require an impractically high number of

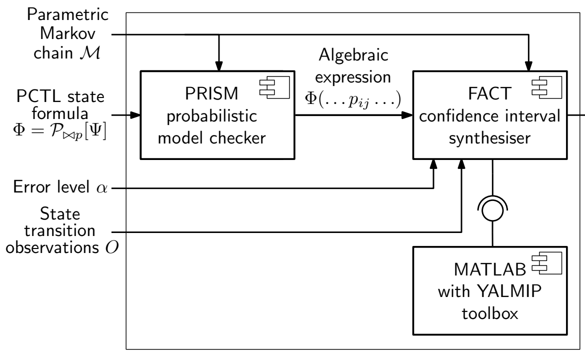
Parametric Markov chain $\mathcal{M}$
PCTL state formula $\Phi = \mathcal{P}_{\bowtie p}[\Psi]$
Error level α
State transition observations O
PRISM probabilistic model checker
Algebraic expression $\Phi(\ldots p_{ij} \ldots)$
FACT confidence interval synthesiser
$(1-\alpha)$ confidence interval for $\mathcal{P}_{=?}[\Psi]$
MATLAB with YALMIP toolbox

Fig. 3. High-level architecture of the FACT tool chain

observations. Thus, it might be practically unfeasible to establish if some QoS properties are satisfied with the required level of confidence.

When the FACT analysis shows that a required QoS property is not satisfied, or cannot provide a definite answer, then the system designer has the option to modify the system architecture or parameters, and to redo the analysis.

V. FACT Tool Chain

To automate the two stages of the FACT process (Fig. 2), we assembled a tool chain that integrates established mathematical analysis tools and libraries, and a confidence interval synthesiser that we developed specifically for FACT.

Fig. 3 shows the high-level architecture of the FACT tool chain.

We use the probabilistic model checker PRISM [11] to obtain an algebraic expression for the analysed property in the *parametric model checking* stage of FACT. Starting with version 4.2, PRISM supports the parametric model checking of PCTL formulae[2]. More precisely, PRISM parametric model checking [35], [36] supports the analysis of PMC models with transition probabilities specified as functions over a set of parameters. Depending on the analysed PCTL formula, the result is given either as an algebraic expressions (i.e., rational function over the parameters), or as a mapping from regions of these parameters to rational functions or truth values.

For the *confidence interval inference* stage of FACT, we developed a prototype confidence interval synthesiser tool in Java. Our tool implements the FACT theoretical framework from Section IV-B, and uses MATLAB [53], and the MATLAB optimisation toolbox YALMIP [54], [55] to calculate the confidence interval $[a, b]$ in line 11 of Algorithm 1. Our FACT confidence interval synthesiser is freely available from http://www-users.cs.york.ac.uk/~raduc/FACT. Likewise, the YALMIP toolbox is free of charge to use, and can be downloaded from the project wiki [56].

VI. Evaluation

To evaluate the effectiveness and generality of FACT, we used our approach and tool chain to establish confidence intervals for the QoS properties of two software systems from different application domains, and across a wide range of scenarios. The two case studies serve different purposes. The former shows the potential of the approach, the role of its parameters, and its performance. The latter case study shows how FACT concretely supports engineers in establishing the QoS properties of a real-world embedded system.

For both case studies, we first implemented simulators of the analysed systems. We then used these simulators to obtain observations of the events corresponding to the state transitions from the system PMCs. To assess whether FACT produces correct results, we instantiated (i.e., fixed) the unknown transition probabilities when collecting the observations. Next, we used FACT to establish confidence intervals for the QoS properties of each system starting from its observations. This approach is the intended use of FACT in a real scenario. Finally, we calculated the actual value of the same QoS properties starting from the transition probability values used in the simulation. This step is not possible in a real scenario, and is not part of FACT. Its only purpose was to allow us to compare the confidence intervals produced by FACT for a range of QoS properties to the actual values of these properties.

A. Web application case study

1) Description: For the first case study, we considered the business-critical web application from our running example described in Section III. Table I shows the QoS requirements for this application, in plain English, and formally specified in PCTL. Requirements **R1** through **R3** are expressed as state PCTL formulae (3), while requirements **R4** through **R5** formalise cost and response time constraints as reward PCTL formulae (5).

As explained in Section II-C, each QoS requirement is expressed by a PCTL formula that uses a fixed bound to constrain the value of a QoS property of the system. For example, the QoS property associated with requirement **R2** is $\mathcal{P}_{=?}[\mathrm{F}\ \mathsf{HttpResponse}]$ (i.e., the probability of successfully handling a request), and **R2** asks its value to be at least 0.995.

2) Experimental setup: To evaluate the effectiveness of FACT in analysing the QoS properties of the web application, we implemented a MATLAB simulator of the application. We then ran the simulator with the PMC parameter values from Table II, and we collected a wide range of state-transition observations (6). Finally, we used these observations, and the FACT tool chain in Fig. 3, to obtain confidence intervals for the QoS properties of the web application. The purpose of these experiments was to provide empirical evidence for answering the following questions.

(a) What is the impact of varying the $1-\alpha$ confidence level in Fig. 3 on the confidence interval produced by FACT?
(b) How does the size of the observation set affect the precision of confidence intervals?
(c) What are some typical numbers of observations required to establish if the system complies with its requirements, at difference confidence levels?
(d) How effective is the hill climbing Algorithm 1 at improving the precision of confidence intervals?

[2]PRISM's implementation of parametric model checking re-implements the techniques previously included in the PARAM model checker [46].

TABLE I
QoS REQUIREMENTS FOR THE WEB APPLICATION

ID	Informal description	PCTL formula
R1	*Cache hit probability*: "At least 65% of the requests are handled without accessing the database or the file server."	$\mathcal{P}_{\geq 0.65}[\neg(\text{Database} \vee \text{FileServer}) U \text{HttpResponse}]$
R2	*Reliability*: "The probability of successfully handling a request must be at least 0.995."	$\mathcal{P}_{\geq 0.995}[\text{F HttpResponse}]$
R3	*Complexity bound*: "88% of the requests must be successfully processed within 4 operations."	$\mathcal{P}_{\geq 0.88}[\text{F}^{\leq 4} \text{ HttpResponse}]$
R4	*Cost*: "The average cost for handling a request must be less 2.5 tenths of a cent."	$\mathcal{R}^{cost}_{\leq 2.5}[\text{F HttpResponse} \vee \text{ServerUnavailable} \vee \text{TooManyConnections}]$
R5	*Response time*: "The average response time must be less than 15 milliseconds."	$\mathcal{R}^{response}_{\leq 15}[\text{F HttpResponse} \vee \text{ServerUnavailable} \vee \text{TooManyConnections}]$

TABLE II
PMC PARAMETERS USED IN THE EXPERIMENTS

State	Outgoing transition probabilities
s_0	$y_1 = 0.4053$, $y_2 = 0.5946$, $y_3 = 0.0001$
s_3	$x_1 = 0.5790$, $x_2 = 0.0005$, $x_3 = 0.4205$
s_4	$w_1 = 0.9998$, $w_2 = 0.0002$
s_5	$z_1 = 0.25065$, $z_2 = 0.00125$, $z_3 = 0.7481$
s_6	$k_1 = 0.9996$, $k_2 = 0.0004$

(e) What is the performance of the FACT toolset on a typical computer?

3) Experimental results: For the *parametric model checking* stage of the FACT analysis, we used version 4.2 of the probabilistic model checker PRISM to obtain algebraic expressions for the QoS properties associated with requirements **R1** through **R5** in Table I. These expressions are shown in Table III.

For the *confidence interval inference* stage of FACT, we used observations from the execution of our MATLAB implementation of the PMC in Fig. 1. Fig. 4 shows the $1-\alpha$ confidence intervals obtained for four typical sets of observations that correspond to N observations of outgoing transitions from each of the PMC states associated with unknown transition probabilities, where $1-\alpha \in \{0.85, 0.86, \ldots, 0.99\}$, and $N \in \{5000, 10000, 25000, 100000\}$. For simplicity, we will use the notation $O^{5\text{K}}$, $O^{10\text{K}}$, $O^{25\text{K}}$, and $O^{100\text{K}}$ to refer to the four observation sets.

Note that these results include the actual values of the QoS properties associated with requirements **R1** through **R5**. These values were obtained using PRISM to analyse the Markov chain obtained by fixing all PMC parameters in Fig. 1 as indicated in Table II, and are provided for reference. Of course, these values are not available when FACT is used in a real-world scenario. The graphs show also the requirement bounds from Table I, with shaded areas for the parts of these graphs that correspond to requirements being violated.

We analysed the results in Fig. 4 to answer questions (a) through (c) from Section VI-A2 as follows.

(a) For any fixed number of observations N, the width of the confidence interval increases supralinearly with the confidence level $1-\alpha$. This result is true for all analysed QoS properties. Furthermore, the 0.99 confidence interval for each of the requirements **R1** through **R5** from our case study are at least twice as wide as the 0.85 confidence interval for the same requirement, across all examined numbers of observations N. As a consequence, given a fixed set of observations, establishing compliance with a requirement is typically possible only up to a certain level of confidence. As an example, for observation set $O^{10\text{K}}$, we can state that **R1** is satisfied with confidence level 0.94; or, using the notation in (9), $\mathcal{M}_{\text{webApp}} \models_{0.06, O^{10\text{K}}}$ **R1**, where $0.06 = 1 - 0.94$ represents the error level. In contrast, it is impossible to establish that the requirement is satisfied with a 0.95 confidence level for $O^{10\text{K}}$: $\mathcal{M}_{\text{webApp}} \not\models_{0.05, O^{10\text{K}}}$ **R1**.

(b) The width of the interval $[bestA, bestB]$ computed by Algorithm 1 is much smaller in experiments with larger observation sets O. For instance, the intervals $[bestA, bestB]$ obtained for requirement **R2** from Table I and the observation sets $O^{5\text{K}}$ and $O^{10\text{K}}$ were not above the **R2** bound of 0.995 (i.e., $bestA \not> 0.995$) for any $1-\alpha \geq 0.85$. In contrast, for the observation set $O^{20\text{K}}$ and $0.85 \leq 1-\alpha \leq 0.95$, the $[bestA, bestB]$ interval produced by Algorithm 1 satisfied $bestA > 0.995$. Thus, according to Proposition 2, $\mathcal{M}_{\text{webApp}} \models_{0.05, O^{20\text{K}}}$ **R2**. Similarly, in the experiment corresponding to $O^{100\text{K}}$, Algorithm 1 produced intervals with $bestA > 0.995$ for all $1-\alpha \leq 0.99$, so $\mathcal{M}_{\text{webApp}} \models_{0.01, O^{100\text{K}}}$ **R2**.

(c) Clearly, the number of observations required to establish if the system complies with its QoS requirements with a fixed level of confidence is influenced by how close the requirement bounds are to the actual (unknown) value of the associated QoS property. This effect is illustrated qualitatively by the FACT analysis results for requirements **R2** and **R3**. For **R2**, a 0.99 confidence interval that does not contain the requirement bound 0.995 is obtained only for the 100,000-observation set, as the requirement bound is less than 0.005 away from the actual value. In contrast, the requirement bound for **R3** is more than 0.04 away from the actual value, and 10,000 observations are sufficient to establish that **R3** is satisfied with 0.99 confidence level.

4) Hill climbing analysis: To evaluate the effectiveness of the hill climbing technique employed by FACT, we recorded the bounds of the confidence interval $[bestA, bestB]$ at the end

TABLE III
ALGEBRAIC EXPRESSIONS FOR QOS REQUIREMENTS **R1-R5**

ID	Algebraic expression
R1	$(160y_2x_1 - 77z_1y_1 - 77z_2y_1 + 160y_1)/160$
R2	$(-160w_1y_2x_1 - 160w_1y_2x_2 + 77k_1z_1y_1 + 160y_2x_1 - 77z_1y_1 - 77z_2y_1 + 160w_1y_2 + 160y_1)/160$
R3	$(508y_1 + 385y_1z_3 + 1000x_1y_2 + 1000y_2x_3w_1)/1000$
R4	$(-160y_2x_1 - 160y_2x_2 + 308z_1y_1 + 497y_1 + 320y_2)/160$
R5	$(-640y_2x_1 - 640y_2x_2 + 539z_1y_1 + 640y_2)/16$

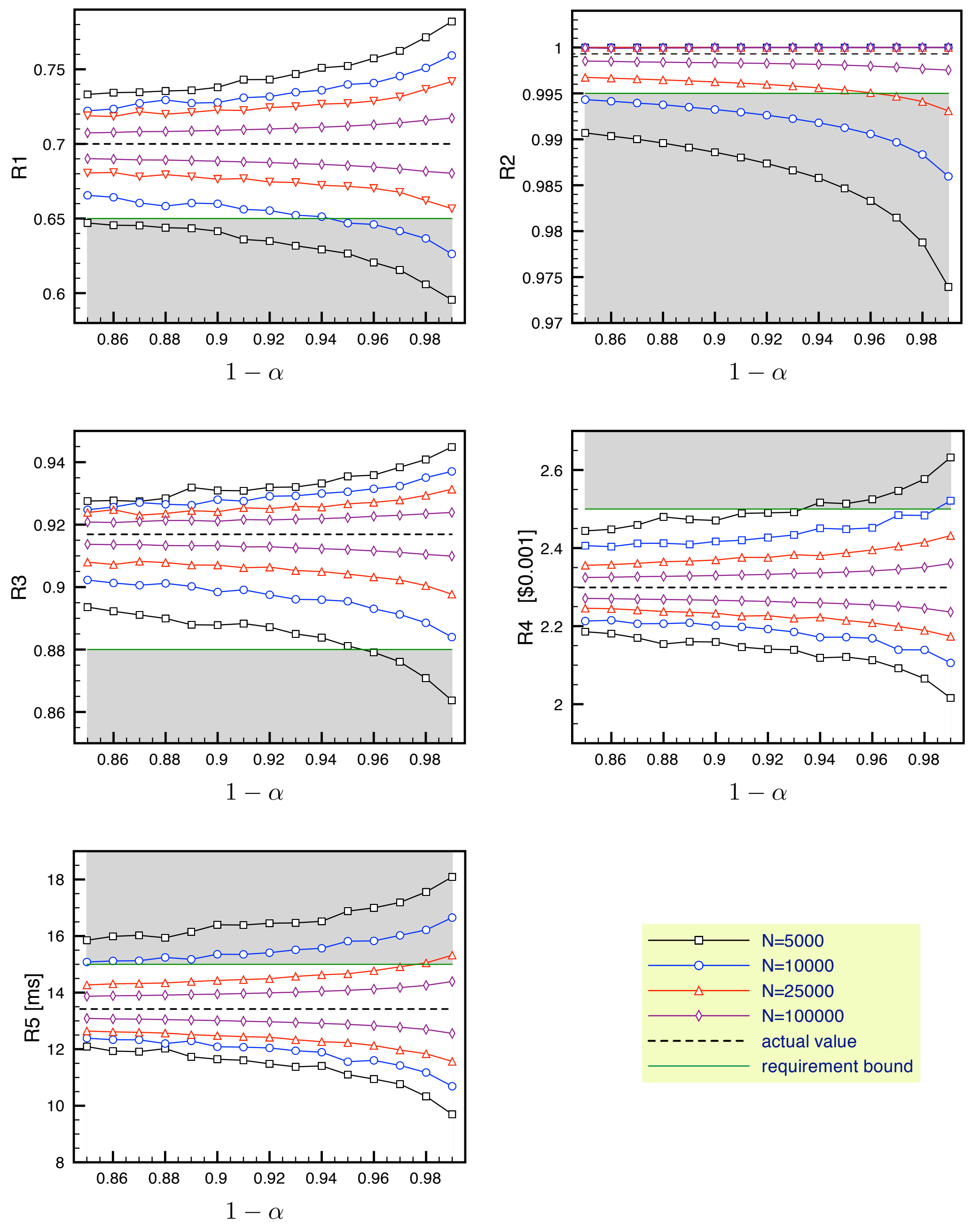

Fig. 4. FACT $1-\alpha$ confidence intervals for requirements **R1** through **R5**, and observation sets of size $N \in \{5000, 10000, 25000, 100000\}$. The shaded areas correspond to violations of the requirement bounds, and the actual values are given for the PMC parameters in Table II.

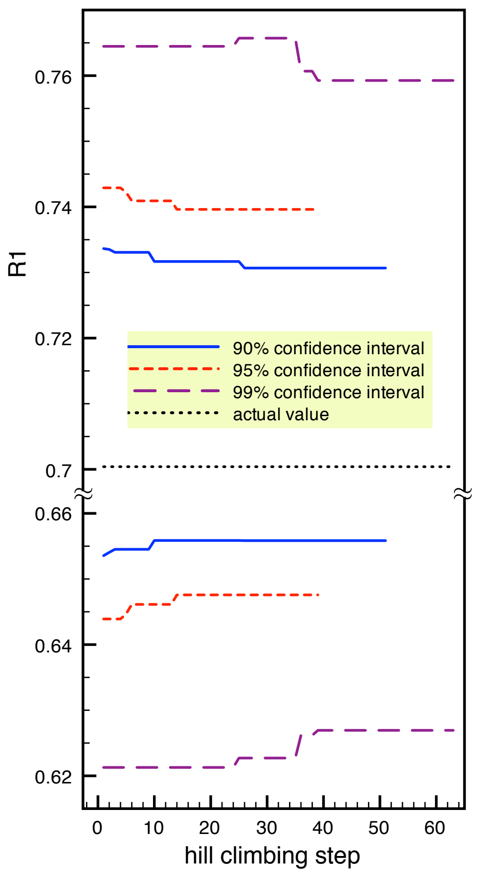

Fig. 5. Hill-climbing refinement of the confidence interval for requirement **R1**.

TABLE IV
REDUCTION IN THE SIZE OF CONFIDENCE INTERVALS THROUGH HILL-CLIMBING OPTIMISATION, FOR THE EXPERIMENT IN FIG. 5

$1-\alpha$	**initial interval**	**final interval**	**steps**	**reduction**
0.90	$[0.6535, 0.7336]$	$[0.6558, 0.7306]$	51	6.6%
0.95	$[0.6439, 0.7429]$	$[0.6475, 0.7396]$	39	6.9%
0.99	$[0.6212, 0.7644]$	$[0.6269, 0.7592]$	63	7.6%

of each iteration of the while loop from lines 7 through 19 of Algorithm 1.

Fig. 5 depicts a typical refinement of the confidence interval for requirement **R1** during the execution of Algorithm 1 with $MAX\_STEPS = 30$, for three confidence levels, and the observation set $O^{10\mathrm{K}}$ from above. This diagram illustrates the general trend of obtaining a larger reduction in the size of the confidence intervals for larger confidence levels, for example a 7.6% reduction for the 0.99 confidence interval compared to a 6.6% reduction for the 0.90 confidence interval (Table IV). Thus, we can answer question (d) from Section VI-A2.

(d) Hill climbing optimisation can improve the precision of confidence intervals significantly, enabling a more accurate analysis of QoS properties.

5) Performance analysis:[3] The FACT verification time is dominated by its *confidence interval inference* stage, with the *parametric model checking* stage taking below 200ms to obtain the algebraic expression in Table III for each of the five requirements. Note also that the first FACT stage needs to be performed only once for each requirement, whereas the second stage must be executed for each confidence level used in the analysis. Table V shows the execution times for this second stage, and the experiments reported in Fig. 4. In these experiments, the hill-climbing optimisation was stopped after $MAX\_STEPS = 50$ steps that did not generate a reduction in the size of the analysed confidence interval. This rule explains why several experiments took precisely 50 optimisation steps; in these experiments, the hill climbing could not find an improvement within its first MAX_STEPS steps. The MAX_STEPS value used for these experiments was larger than the one from the hill climbing analysis so we could stretch the approach when evaluating its performance. Based on these results, we can address our last question from Section VI-A2.

(e) The time taken to establish confidence intervals using the FACT toolset is dominated by the *confidence interval inference* stage. Within this stage, the hill-climbing optimisation of the initial solution takes most of the time (above 98% for the experiments in Table V). The average time required for one execution of optimisation step ranged between 2.33s and 3.04s, and further investigation showed that this range corresponds to the invocation of the MATLAB–YALMIP optimisation engine for the calculation of the confidence interval bounds (11). The overall time to obtain a confidence interval varied between approximately 2 and 4 minutes on an off-the-shelf laptop computer, so we concluded that the FACT tool chain was sufficiently efficient to support the QoS analysis of the non-trivial system in our case study.

To ensure that the results described above were not skewed by the choice of PMC transition probabilities from Table II, we repeated the experiments used to produce them for 100 randomly selected instantiations of the transition probabilities. Given the large number of such experiments, we considered only the commonly used confidence levels $(1-\alpha) \in \{0.95, 0.99\}$, and we ran the experiments for observation sets of size $N = 10,000$. Thus, we used the FACT tool chain to compute the two confidence intervals for each of the five QoS requirements in Table I, and each of the 100 parameter instantiations, running 1000 experiments in total.

For each experiment, we recorded the synthesised confidence interval, the decrease in the size of this interval due to the FACT hill climbing, and the total time taken to compute the confidence interval. These experimental results are summarised in Table VI, and are in line with the findings presented earlier in this section. In particular, the confidence interval sizes for all requirements are similar to those reported

[3]All experiments were carried out using a standard OS X 10.9.4 MacBook Pro computer with a 2GHz Intel Core i7 processor, and 8GB 1600MHz DDR3 RAM for the PRISM parametric model checking; and an OS X 10.9.4 Macbook Pro computer with a 2.3GHz Intel Core i7 processor, and 16 GB 1600 MHz DDR3 RAM for the confidence interval inference.

TABLE V
Confidence interval inference time and hill-climbing optimisation steps

ID	5000-observation set, O^{5K}			10000-observation set, O^{10K}			25000-observation set, O^{25K}			100000-observation set, O^{100K}		
	average hill climbing steps	average step time [s]	average time[†] [s]	average hill climbing steps	average step time [s]	average time[†] [s]	average hill climbing steps	average step time [s]	average time[†] [s]	average hill climbing steps	average step time [s]	average time[†] [s]
R1	63.5	2.45	158.07	56.2	2.31	132.25	53.6	2.39	130.7	50.0	2.40	122.52
R2	50.0	3.04	155.35	50.0	2.68	137.15	50.0	2.65	135.27	50.0	2.59	132.24
R3	50.0	2.59	132.58	50.0	2.46	125.63	50.0	2.45	125.28	50.0	2.42	123.55
R4	87.2	2.35	208.08	50.0	2.46	125.63	72.8	2.33	172.36	50.0	2.49	136.43
R5	100.4	2.40	244.12	98.6	2.42	241.39	100.6	2.43	247.33	74.8	2.44	198.47

[†]The average time includes the time taken by the hill-climbing optimisation steps and by the other operations of Algorithm 1.

TABLE VI
Performance of confidence interval computation over 100 random PMC parameter instantiations

	0.95 confidence interval computation							0.99 confidence interval computation					
	confidence interval size		hill-climbing gain[†] [%]		total time [s]			confidence interval size		hill-climbing gain[†] [%]		total time [s]	
ID	mean	SD	mean	SD	mean	SD	ID	mean	SD	mean	SD	mean	SD
R1	0.164	0.029	2.66	3.03	186.0	44.5	**R1**	0.237	0.042	2.70	3.02	381.7	75.0
R2	0.109	0.025	10.8	4.67	165.7	37.5	**R2**	0.164	0.037	7.53	3.45	324.7	48.7
R3	0.123	0.028	6.27	3.32	187.5	40.7	**R3**	0.180	0.041	4.64	2.69	385.9	74.4
R4	0.362	0.074	11.8	3.74	213.8	59.2	**R4**	0.545	0.112	8.27	2.68	415.4	87.7
R5	4.904	0.840	14.4	4.48	272.3	100.3	**R5**	7.436	1.202	10.2	2.73	570.5	189.7

[†]Decrease in confidence interval size due to hill climbing, relative to the confidence interval size without hill climbing.

in Fig. 4 for observation sets of size $N = 10,000$, and the two confidence levels, with small variations. As expected, the interval sizes are higher for the 0.99 confidence intervals. The use of hill climbing to reduce the confidence interval size produced average gains of between 2.66% and 14.4%, similar to those reported earlier in Fig. 5. Finally, the average time to calculate a confidence interval was under 275s, and under 571s for the 0.95, and 0.99 confidence intervals, respectively. The longer time needed to compute confidence intervals for higher $(1-\alpha)$ confidence levels is due to an increased sensitivity of the interval size to variations in the confidence level when $(1-\alpha)$ approaches 1.0. For instance, the difference in size between 0.985 and 0.99 confidence intervals is much larger than the difference in size between 0.945 and 0.95 confidence intervals. Therefore, the hill climbing will take longer to compute a stable confidence interval when operating with confidence levels close to 1.0.

B. The Low-power Wireless Bus case study

1) Description: In this section, we present the application of the FACT approach to a real-world embedded software system from the Wireless Sensor Networks (WSNs) [57] domain. Designing and implementing an efficient WSN system requires the careful evaluation of multiple engineering choices, to identify effective trade-offs between the quality properties of the system. In particular, energy efficiency is the main QoS aspect that drives the design of these systems.

One of the mainstream approaches to designing energy-efficient WSN systems relies on evaluating alternative configurations through formal modelling and analysis (e.g., [58]). Accordingly, our case study applies FACT formal verification to a WSN system that uses the Low-power Wireless Bus (LWB) communication protocol [59]. LWB is a recently proposed WSN communication protocol that turns a multi-hop low-power wireless network into an infrastructure similar to a shared bus, where all nodes are potential receivers of all data. It achieves this structure by mapping all traffic demands on a type of fast *network floods* (i.e., transmissions of one packet from an originator node to all other nodes in the network) called Glossy floods [60]. To avoid collisions between floods, LWB uses a time-triggered operation: nodes communicate according to a global communication schedule that determines when a node is allowed to initiate a flood.

The protocol operates within communication rounds that repeat with a *round period* T, computed at the host, and based on the current traffic demands. Every round consists of a number of non-overlapping communication slots. In each slot, at most one node puts a message on the bus (initiates a Glossy flood), whereas all other nodes read the message from the bus (receive and relay the flood). A round starts and ends with a slot allocated by the host to distribute the communication schedule. The detailed description of the protocol is beyond the scope of this paper, and can be found in [59].

2) Modelling LWB nodes: LWB has configuration parameters whose choice influences its *radio on-time* (i.e., the time a receiver node has its radio turned on during a network flood), and thus the energy consumption of a node. In particular, the choice of the LWB round period T in the acceptable range $1000\text{ms} \leq T \leq 60,000\text{ms}$ can lead to significantly different energy consumptions. The selection of this parameter

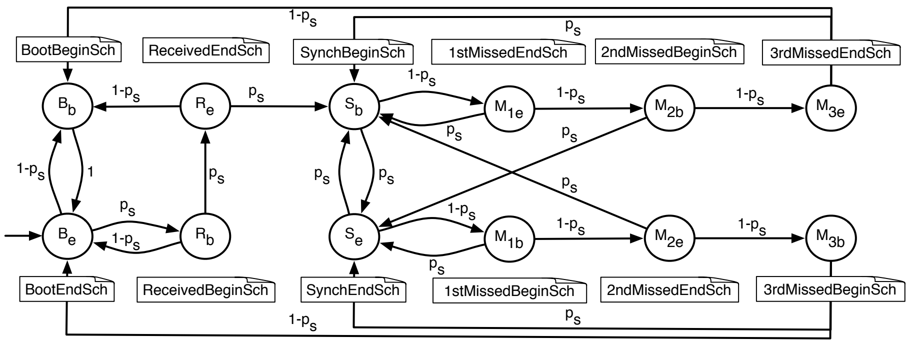

Fig. 6. Parametric Markov chain model of an LWB node, taken from [58].

TABLE VII
REWARD STRUCTURES FOR THE LWB MODEL IN FIG. 6

State	Energy consumption per period ('period')	Start-up energy consumption ('startup')
B_e	(T-1000)/T	T-1000
B_b	1000/T	1000
R_e	35/T	35
R_b	145/T	145
S_e	19/T	19
S_b	129/T	0
M_{1e}	19/T	0
M_{1b}	133/T	0
M_{2e}	28/T	0
M_{2b}	28/T	0
M_{3e}	35/T	0
M_{3b}	35/T	0

TABLE VIII
LWB QOS REQUIREMENTS

ID	Informal description	PCTL formula
R1	*Steady-state power consumption*: "The radio on-time per period should be less than 5%."	$\mathcal{R}^{\text{period}}_{\leq 0.05}$[S]
R2	*Start-up power consumption*: "The start-up radio on-time should be less than 40s."	$\mathcal{R}^{\text{startup}}_{\leq 40}$[F SynchBeginSch]

is based on the analysis of the worst-case radio on-time of a node, which represents a measure of the worst-case energy consumption of a WSN system. This analysis uses the parametric Markov chain (PMC) of a LWB node introduced in [58], and shown in Fig. 6. The unknown transition probability p_s in this PMC represents the probability for the node to receive a schedule that was sent by the host.

For the analysis of the worst-case energy consumption of an LWB node, the PMC in Fig. 6 is augmented with the two reward structures detailed in Table VII. The former structure associates each PMC state with the fraction of a communication round for which the LWB node has the radio switched on. The latter structure maps each PMC state to its radio on-time during the start-up synchronisation between the node and the host. Justifying the assumptions and accuracy of this model is beyond the scope of our paper; these details can be found in [58].

3) Using FACT to select the LWB round period: Table VIII shows a pair of QoS requirements that are commonly used to drive the selection of the round period T. Requirement **R1** measures the expected steady-state energy consumption, while requirement **R2** measures the energy consumption during the initial startup synchronisation with the host. The PCTL formulae for these requirements correspond to the two reward structures from Table VII.

Given these requirements, and the PMC in Fig. 6, engineers can use the FACT tool chain for the selection of a suitable round period T as follows. First, as the probability of successful schedule receipt p_s is typically unknown, experiments using the testbeds and settings described in [58] must be carried out to estimate it empirically. The experimental data are encoded as observation sets (6), and a confidence level $1-\alpha$ for the analysis of the two requirements is decided. The observation sets and confidence level are then used as inputs for the FACT tool chain, together with the PMC model from Fig. 6, and the PCTL formulae in Table VIII. Given these inputs for multiple values of the round period T, our tool chain synthesises $1 - \alpha$ confidence intervals for each of the two QoS properties, and each value of T.

Figs. 7 and 8 show the results of this FACT analysis for a typical 1000-observation set (obtained through model simulations for $p_s = 0.8$), and a confidence level $1-\alpha = 0.99$. These 0.99 confidence intervals illustrate the trade-off between the steady-state energy usage (which decreases with T) and the start-up energy usage (which increases when T increases). In this scenario, the two QoS requirements are satisfied with probability 0.99 when the round period T is selected in the interval $[8000\text{ms}, 14,000\text{ms}]$, because the threshold from requirement **R1** is met for $T \geq 8000\text{ms}$, and the **R2** threshold is met for $T \leq 14,000\text{ms}$.

In contrast, using a point estimator instead of the confidence interval generated by FACT may lead to suboptimal or invalid choices for the LWB round period. As an example, the application of the maximum likelihood estimator (MLE) [61] to the 1000 observations of p_s used in Figs. 7 and 8 yields the point estimate $\hat{p}_s = 0.794$. The dotted MLE plots in Figs. 7 and 8 show the variation of the two QoS properties

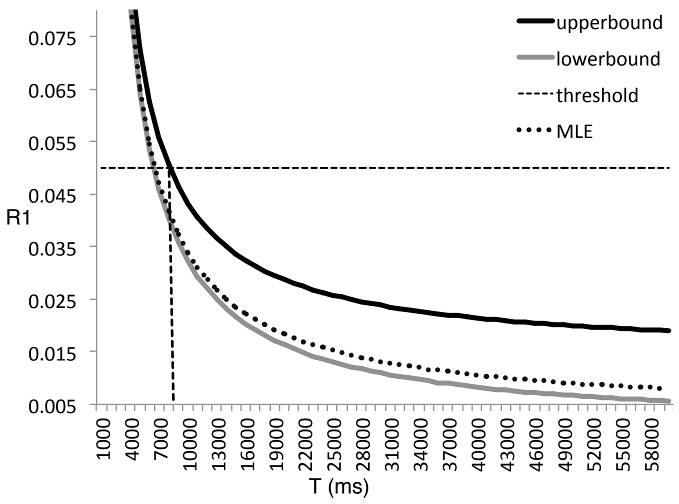

Fig. 7. Steady-state energy consumption with increasing round period values.

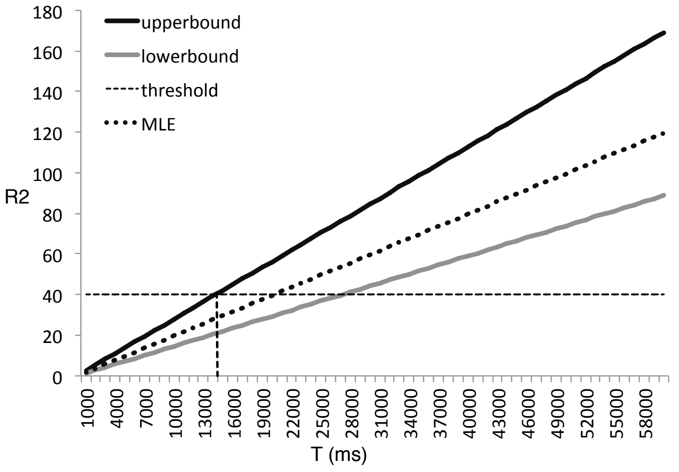

Fig. 8. Start-up energy consumption with increasing round period values.

with T when $p_s = \hat{p}_s = 0.794$ in the PMC model from Fig. 6. According to these results, the two QoS requirements are satisfied for T values in the interval $[6000\text{ms}, 20,000\text{ms}]$. This interval is much wider than the 0.99 confidence interval generated by FACT, and the selection of a T value within it provides no guarantees that the QoS requirements will be satisfied with at least some minimum probability. FACT addresses this important drawback of point estimates by enabling engineers to select LWB configurations guaranteed to satisfy the quality requirements of the system with the required minimum probability.

4) Performance: We carried out all the experiments for the LWB case study using the computers with the specification in Section VI-A5. As for our first case study, the *parametric model checking* FACT stage completed in sub-second time (i.e., 172ms for **R1**, and 184ms for **R2**). The algebraic expressions generated by the PRISM probabilistic model checker for this stage are shown in Table IX.

In the *confidence interval synthesis* stage of the FACT approach, the 0.99 confidence intervals for **R1** and **R2** from

TABLE IX
ALGEBRAIC EXPRESSIONS FOR LWB QOS REQUIREMENTS **R1** AND **R2**

ID	Algebraic expression
R1	$(12p_s^{10} - 427p_s^9 + (T + 2322)p_s^8 - (6T + 5972)p_s^7 + (16T + 9815)p_s^6 - (24T + 12027)p_s^5 + (22T + 11102)p_s^4 - (14T + 6277)p_s^3 + (8T+2455)p_s^2 - (4T+355)p_s + T)/(2Tp_s^8 - 12Tp_s^7 + 34Tp_s^6 - 54Tp_s^5 + 52Tp_s^4 - 32Tp_s^3 + 18Tp_s^2 - 8Tp_s + 2T)$
R2	$(-1000p_s^3 + 1035p_s^2 - 355p_s + T)/(p_s^3)$

Figs. 7 and 8 were obtained for $T = 1000\text{ms}, 2000\text{ms}, \ldots, 60,000\text{ms}$. This calculation involved evaluating the algebraic expressions in Table IX for each of these values of T, and using the resulting expressions as input for the FACT confidence interval synthesiser in Fig. 3. Because the only unknown transition probability in these expressions is p_s, the only way to obtain a 0.99 confidence interval for **R1** and **R2** is to start with a 0.99 confidence interval for p_s. Thus, hill-climbing optimisation to identify better combinations of confidence levels for the unknown transition probabilities was not required for the analysis of the LWB system. As a result, each of the 120 confidence interval calculations (corresponding to the two QoS requirements and the 60 values of T mentioned above) took below 500ms.

VII. DISCUSSION

The experiments described in the previous section show the feasibility and benefits of our FACT theoretical framework and tool chain for formal verification with confidence intervals. In particular, we demonstrated the application of FACT within two case studies from different domains, to support the analysis of several types of QoS properties. The applicability of FACT to other domains and to larger systems depends on the capabilities of its two components from Fig. 2, i.e., parametric model checking and confidence interval inference.

Parametric model checking, in particular, is computationally expensive. The complexity of the early techniques for the computation of symbolic expressions for reachability PCTL properties for an n-state model is as high as $n^{\Theta(\log n)}$ [45]. However, parametric model checking is a new area of research, and newer techniques operate with much lower overheads. Thus, the approach from [35] is much faster than [45] in many practical scenarios, and the most recent result from this area reports speed-ups of up to several orders of magnitude [37]. The probabilistic model checker used for the parametric model checking step of the FACT tool chain, i.e., PRISM [11], implements the techniques from [35], [36], and was able to generate all the symbolic expressions for each of the QoS properties from our smaller but realistic case studies in under 200ms. Using a model checker that implements the efficient parametric model checking technique from [37] will enable FACT to scale to much larger system sizes.

Confidence interval inference is also computationally expensive. Our FACT tool chain is based on the MATLAB optimisation toolbox YALMIP [54], [55] configured to use its own, free global optimization solver based on [62], which needs 2s to 3s to execute each step of the FACT hill climbing algorithm for the QoS properties in Section VI. However, YALMIP can also be configured to use the much faster, state-of-the-art industrial optimiser Gurobi [63], which again provides an opportunity for scaling up FACT to larger system sizes than presented in the previous section.

Finally, the FACT theoretical framework presented in this paper considers PMCs whose unknown transition probabilities are specified as variables. This approach differs from the PMC definition in, e.g., [35], [36], [37], where transition probabilities are expressed as fractions of polynomials over a set of variables (i.e., the system parameters). Extending the FACT theoretical framework to handle transition probabilities expressed as rational functions of the system parameters is certainly possible, and would involve operating with observations and confidence intervals of the system parameters instead of the unknown transition probabilities. The main implication would be an increase in the complexity of the symbolic expressions of the analysed QoS properties. This may happen if complex rational functions are used to specify the unknown transition probabilities. However, the current research [35], [37] is based on case studies that use rational functions similar to those from our case studies [64]. Several examples of these rational functions include p and $1 - p$, and q and $1 - q$ for the two sets of unknown transition probabilities of a Zeroconf protocol [65] PMC available from [66], and identically formulated functions for the two sets of unknown transition probabilities of a Crowds protocol [67] PMC available from [68].

VIII. Related work

Markov chains are widely used for the verification of reliability, performance, and other QoS properties of software and hardware systems. However, in many practical applications, the MC state transition probabilities are estimated experimentally, leading to uncertainty and imprecisions that may affect the accuracy of the results. To the best of our knowledge, FACT is the first approach that supports formal verification of Markov chains that exploits confidence intervals to quantitatively account for this uncertainty. In this section, we survey several approaches that investigated similar or related challenges.

Interval-valued discrete-time Markov chains (IDTMC) were introduced in [69] to incorporate uncertainty in traditional MCs by assuming that transition probability values lie within a range or interval of possible values. IDTMCs were subsequently refined and generalised to models that include generic convex sets of probabilities in [70]. The approaches in [69], [70] addressed the important problem of incorporating uncertainty into Markov chains. However, their scope is limited to the model definition. Unlike FACT, these approaches do not tackle the challenge of propagating the uncertainty captured by the model to properties of the model expressed in probabilistic temporal logic.

Benedikt *et al.* [33] provide lower and upper bounds on the complexity of evaluating ω-regular specification satisfiability in IDTMCs, and an *expectation maximisation* algorithm that starts from an IDTMC and produces a sequence of refinements of increasing probability of satisfying an ω-regular property. The Tulip model checking tool [33] uses this algorithm to calculate an approximation for the maximum probability with which an IDTMC model satisfies one of these properties. This approximation is then used for the iterative refinement and modification of an existing model, to obtain a model variant that satisfies the original property (see also the existing approaches for *model repair* [71]). Accordingly, the technique devised by Benedikt *et al.* [33] is the inverse of our FACT approach, which starts from a given uncertain model, and establishes if the model meets a specification with the required confidence level.

Uncertainty in PCTL verification is taken into account by [34] through modelling systems as IDTMCs interpreted semantically as either uncertain Markov chains (UMCs) or interval Markov decision processes (IMDPs). The former interpretation is often useful in scenarios where state transitions of the system are known to lie within a specific interval of probabilities, while the latter interpretation generalises MDPs to an uncountable set of non-deterministic choices to be taken at each state transition, modelling variations within the system's environment during execution. Chatterjee *et al.* [72] extend PCTL to express ω-regular conditions. Model checking of these specifications are considered under various semantic interpretations of IDTMCs, proving upper and lower complexity bounds. Chen *et al.* [73] improve these results by demonstrating that the reachability probability for IDTMCs coincides under UMC and IMDPs semantics, and is P-complete. They use an ellipsoid method to yield a polynomial-time verification algorithm. The PCTL model checking problem is shown to be P-complete under the IMDP semantics, and reducible to the *square-root-sum* problem under the UMC semantics. [74] extends interval-bounded models to more general forms of likelihood models and ellipsoidal models, to improve transition uncertainty in cases where probabilities are determined experimentally.

Although the above results address the verification of PCTL formulae over models affected by uncertainty, several unique characteristics distinguish FACT from them. First, FACT operates with parametric Markov chains with unknown transition probabilities specified as sets of observations. This operation makes our approach particularly suitable for practical applications, in which the transition probabilities are unknown, and need to be derived from observations. Second, FACT is uniquely capable of establishing confidence intervals for PCTL properties at any requested confidence level. Thirdly, our approach uses hill-climbing optimisation to reduce the width of its confidence intervals over a number of iterations. Last but not least, FACT can be readily used in practice, thanks to a tool chain that automates all steps of the approach.

In another area of related research, [47], [75] use perturbation analysis to compute bounds for the probabilistic model checking of parameterised discrete-time Markov chains. These bounds measure the sensitivity of the model to parameter changes, predict the maximal variations in verification results with respect to the amount of perturbation in the model, and may also specify boundaries for the variation of a verification result. This work complements the approach presented in our paper, as it analyses the relevance of different model parameters for a given PCTL property. In contrast, FACT handles the propagation of uncertainty in the model parameters during the verification process, to establish if the system behaves as expected, with the required level of confidence.

Finally, Bortolussi *et al.* [76] explore statistical model checking to determine the satisfiability probability of metric interval temporal logic formulae for continuous-time Markov chains with parametric uncertainty. They provide approximations of satisfaction functions as an estimate of the probability for formula satisfiability for all parameter values from observations of individual runs of the system. Given samples of model parameters, the approach evaluates the probability of satisfaction of the property of interest. This differs from FACT, which focuses instead on discrete-time parametric Markov chains, and synthesises confidence intervals for the analysed property at any required confidence level.

IX. Conclusions, and Future Work

We introduced FACT, the first tool-supported approach for establishing the quality properties of software systems using formal verification with confidence intervals. Our paper contributes to the research on the formal verification for complex software systems by proposing a novel approach to synthesising confidence intervals for PCTL verification over parametric Markov models, given a set of observations of the system, and a required level of confidence. The proposed solution has been implemented using a tool chain that integrates established model checking and mathematical analysis tools, and a freely available tool that we developed to automate the confidence interval inference stage of the approach.

The evaluation of the FACT approach and tool chain in two case studies from different domains show that FACT produces useful confidence intervals, across wide ranges of confidence levels and observation sets. For the verified QoS properties (depending on between one and seven unknown transition probabilities), the synthesis of the confidence intervals took between half a second (for univariate properties) and just over four minutes (for properties depending on multiple unknown probabilities). The further exploration of the FACT performance and scalability represents an area of future work for our project. However, note that the two stages are impacted differently by the size of the verified Markov chain. Thus, the parametric model checking stage will be heavily dependent on the size of the model. Note, however, that this stage needs to be executed only once for a given model and property. In contrast, the confidence interval inference stage of FACT is executed each time when a new set of observations is available, or a confidence interval is required at a new level of confidence. We hypothesise that the time required to carry out this stage is *s*-independent of the original model size, and depends primarily on the number of unknown transition probabilities from the parametric Markov chain. A large number of experiments will be needed to assess the validity of this hypothesis.

The two case studies presented in the paper show that FACT can be used to analyse multiple quality aspects of software systems, including reliability, performance, cost, and energy consumption. In future work, we will explore the possibility to extend the approach to other modelling formalisms and verification logics, with a view to support the verification of additional quality properties of software systems. Continuous-time Markov chains and continuous stochastic logic represent strong candidates for this extension, as they enable the modelling and analysis of system characteristics not covered by discrete-time Markov models and PCTL.

In recent work, we advocated the run-time use of quantitative verification in business- and safety-critical systems that need to self-adapt to changes in their environment or requirements [16], and we successfully applied run-time quantitative verification to multiple software systems [17], [18], [19], [27], [44]. The integration of FACT with these results has the potential to provide stronger guarantees that self-adaptive software systems meet their QoS requirements than is currently possible otherwise.

Acknowledgement

This work was partly funded by the UK Engineering and Physical Sciences Research Council grant EP/H042644/1 and the FP7-PEOPLE-2011-IEF European Commission Programme project 302648-RunMore.

References

[1] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Sureys*, vol. 28, no. 4, pp. 626–643, Dec. 1996. [Online]. Available: http://doi.acm.org/10.1145/242223.242257

[2] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM Comput. Surv.*, vol. 41, no. 4, pp. 19:1–19:36, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592434.1592436

[3] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan, "Using formal specifications to support testing," *ACM Comput. Surv.*, vol. 41, no. 2, pp. 9:1–9:76, Feb. 2009. [Online]. Available: http://doi.acm.org/10.1145/1459352.1459354

[4] B. Meyer, "Applying 'design by contract'," *Computer*, vol. 25, no. 10, pp. 40–51, Oct 1992.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[6] M. Kwiatkowska, "Quantitative verification: Models techniques and tools," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC-FSE '07. New York, NY, USA: ACM, 2007, pp. 449–458. [Online]. Available: http://doi.acm.org/10.1145/1287624.1287688

[7] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen, "Model-checking algorithms for continuous-time Markov chains," *Software Engineering, IEEE Transactions on*, vol. 29, no. 6, pp. 524–541, June 2003.

[8] M. Kwiatkowska, G. Norman, and D. Parker, "Modelling and verification of probabilistic systems," in *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, ser. CRM Monograph Series, P. Panangaden and F. van Breugel, Eds. American Mathematical Society, 2004, vol. 23, pp. 93–215.

[9] M. Z. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, no. 1, pp. 33–78, 2006.

[10] M. Z. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with PRISM: A hybrid approach," *Int. Journal on Software Tools for Technology Transfer(STTT)*, vol. 6, no. 2, pp. 128–142, Aug. 2004.

[11] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[12] J.-P. Katoen, M. Khattri, and I. S. Zapreev, “A Markov reward model checker,” in *Quantitative Evaluation of Systems*. Los Alamitos: IEEE Computer Society, 2005, pp. 243–244.

[13] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, “The ins and outs of the probabilistic model checker MRMC,” *Performance Evaluation*, vol. 68, no. 2, pp. 90 – 104, 2011, advances in Quantitative Evaluation of Systems {QEST} 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531610000660

[14] H. L. S. Younes, “Ymer: A statistical model checker,” in *Computer Aided Verification*, ser. LNCS, K. Etessami *et al.*, Eds. Springer, 2005, vol. 3576, pp. 429–433.

[15] S. Gallotti, C. Ghezzi, R. Mirandola, and G. Tamburrelli, “Quality prediction of service compositions through probabilistic model checking,” in *Quality of Software Architectures. Models and Architectures*, ser. Lecture Notes in Computer Science, S. Becker, F. Plasil, and R. Reussner, Eds. Springer Berlin Heidelberg, 2008, vol. 5281, pp. 119–134. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87879-7_8

[16] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, “Self-adaptive software needs quantitative verification at runtime,” *Communications of the ACM*, vol. 55, no. 9, pp. 69–77, September 2012.

[17] A. Filieri, C. Ghezzi, and G. Tamburrelli, “A formal approach to adaptive software: continuous assurance of non-functional requirements,” *Formal Asp. Comput.*, vol. 24, no. 2, pp. 163–186, 2012.

[18] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, “Dynamic QoS management and optimization in service-based systems,” *IEEE Trans. Softw. Eng.*, vol. 37, pp. 387–409, 2011.

[19] R. Calinescu, K. Johnson, and Y. Rafiq, “Developing self-verifying service-based systems,” in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, Nov 2013, pp. 734–737.

[20] K. Johnson, S. Reed, and R. Calinescu, “Specification and quantitative analysis of probabilistic cloud deployment patterns,” in *Hardware and Software: Verification and Testing*, ser. Lecture Notes in Computer Science, K. Eder, J. Louren??o, and O. Shehory, Eds. Springer Berlin Heidelberg, 2012, vol. 7261, pp. 145–159. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34188-5_14

[21] D. Perez-Palacin, R. Calinescu, and J. Merseguer, “Log2cloud: Log-based prediction of cost-performance trade-offs for cloud deployments,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC ’13. New York, NY, USA: ACM, 2013, pp. 397–404. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480442

[22] C. Ghezzi and A. M. Sharifloo, “Model-based verification of quantitative non-functional properties for software product lines,” *Information and Software Technology*, vol. 55, no. 3, pp. 508 – 524, 2013, special Issue on Software Reuse and Product Lines Special Issue on Software Reuse and Product Lines. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584912001516

[23] R. Calinescu and M. Z. Kwiatkowska, “Using quantitative analysis to implement autonomic IT systems,” in *Proceedings of the 31st International Conference on Software Engineering, ICSE 2009*. IEEE Computer Society, 2009, pp. 100–110.

[24] A. Filieri, C. Ghezzi, and G. Tamburrelli, “Run-time efficient probabilistic model checking,” in *Proceedings of the 33rd International Conference on Software Engineering*. IEEE Computer Society, 2011, pp. 341–350.

[25] P. Inverardi, P. Pelliccione, and M. Tivoli, “Towards an assume-guarantee theory for adaptable systems,” in *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS ’09. ICSE Workshop on*, may 2009, pp. 106 –115.

[26] M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, “Assume-guarantee verification for probabilistic systems,” in *TACAS’10*. Springer, 2010, pp. 23–37.

[27] K. Johnson, R. Calinescu, and S. Kikuchi, “An incremental verification framework for component-based software systems,” in *Proc. 16th Intl. ACM Sigsoft Symposium on Component-Based Software Engineering*, 2013, pp. 33–42.

[28] V. Forejt, M. Kwiatkowska, D. Parker, H. Qu, and M. Ujma, “Incremental runtime verification of probabilistic systems,” in *Runtime Verification*, ser. LNCS, vol. 7687. Springer, 2012, pp. 314–319.

[29] S. Gerasimou, R. Calinescu, and A. Banks, “Efficient runtime quantitative verification using caching, lookahead and nearly-optimal reconfiguration,” in *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014, to appear.

[30] G. Su and D. Rosenblum, “Asymptotic bounds for quantitative verification of perturbed probabilistic systems,” in *Formal Methods and Software Engineering*, ser. Lecture Notes in Computer Science, L. Groves and J. Sun, Eds. Springer Berlin Heidelberg, 2013, vol. 8144, pp. 297–312. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41202-8_20

[31] J. G. Kemeny, J. L. Snell, and A. W. Knapp, *Denumerable Markov Chains, 2nd edition*, ser. Graduate Texts in Marhematics. Springer, 1976, vol. 40.

[32] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[33] M. Benedikt, R. Lenhardt, and J. Worrell, “LTL model checking of interval Markov chains,” in *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS’13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 32–46. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36742-7_3

[34] K. Sen, M. Viswanathan, and G. Agha, “Model-checking Markov chains in the presence of uncertainties,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, H. Hermanns and J. Palsberg, Eds. Springer Berlin Heidelberg, 2006, vol. 3920, pp. 394–410. [Online]. Available: http://dx.doi.org/10.1007/11691372_26

[35] E. Hahn, H. Hermanns, and L. Zhang, “Probabilistic reachability for parametric Markov models,” *International Journal on Software Tools for Technology Transfer*, vol. 13, no. 1, pp. 3–19, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10009-010-0146-x

[36] E. M. Hahn, T. Han, and L. Zhang, “Synthesis for PCTL in parametric Markov decision processes,” in *NASA Formal Methods*. Springer, 2011, pp. 146–161.

[37] N. Jansen, F. Corzilius, M. Volk, R. Wimmer, E. Abraham, J.-P. Katoen, and B. Becker, “Accelerating parametric probabilistic verification,” in *QEST 2014*, ser. Lecture Notes in Computer Science, G. Norman and W. Sanders, Eds., 2014, vol. 8657, pp. 404–420.

[38] F. Ciesinski and M. Größer, “On probabilistic computation tree logic,” in *Validation of Stochastic Systems - A Guide to Current Research*, ser. LNCS, C. Baier *et al.*, Eds., vol. 2925. Springer, 2004, pp. 147–188.

[39] H. Hansson and B. Jonsson, “A logic for reasoning about time and reliability,” *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[40] M. Ben-Ari, Z. Manna, and A. Pnueli, “The temporal logic of branching time,” in *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL ’81. New York, NY, USA: ACM, 1981, pp. 164–176. [Online]. Available: http://doi.acm.org/10.1145/567532.567551

[41] S. Andova, H. Hermanns, and J.-P. Katoen, “Discrete-time rewards model-checked,” in *FORMATS 2003*, ser. Lecture Notes in Computer Science, K. G. Larsen and P. Niebert, Eds., 2004, vol. 2791, pp. 88–104.

[42] C. Ghezzi, M. Pezze, and G. Tamburrelli, “Adaptive rest applications via model inference and probabilistic model checking,” in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 1376–1382.

[43] C. Ghezzi, M. Pezzè, M. Sama, and G. Tamburrelli, “Mining behavior models from user-intensive web applications.” in *ICSE*, 2014, pp. 277–287.

[44] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, “Model evolution by run-time parameter adaptation,” in *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*. IEEE, 2009, pp. 111–121.

[45] C. Daws, “Symbolic and parametric model checking of discrete-time Markov chains,” *ICTAC*, pp. 280–294, 2005.

[46] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang, “PARAM: A model checker for parametric Markov models,” in *Computer Aided Verification*. Springer, 2010, pp. 660–664.

[47] G. Su and D. S. Rosenblum, “Perturbation analysis of stochastic systems with empirical distribution parameters,” in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 311–321. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568256

[48] K.-S. Kwong and B. Iglewicz, “On singular multivariate normal distribution and its applications,” *Computational statistics & data analysis*, vol. 22, no. 3, pp. 271–285, 1996.

[49] L. A. Goodman, “On simultaneous confidence intervals for multinomial proportions,” *Technometrics*, vol. 7, no. 2, pp. 247–254, 1965.

[50] C. Quesenberry and D. Hurst, “Large sample simultaneous confidence intervals for multinomial proportions,” *Technometrics*, vol. 6, no. 2, pp. 191–195, 1964.

[51] S. Fitzpatrick and A. Scott, “Quick simultaneous confidence intervals for multinomial proportions,” *Journal of the American Statistical Association*, vol. 82, no. 399, pp. 875–878, 1987.

[52] C. P. Sison and J. Glaz, "Simultaneous confidence intervals and sample size determination for multinomial proportions," *Journal of the American Statistical Association*, vol. 90, no. 429, pp. 366–369, 1995.

[53] T. A. Davis, "MATLAB Primer," Eight Edition, CRC Press, 2010.

[54] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[55] ——, "Automatic robust convex programming," *Optimization methods and software*, vol. 27, no. 1, pp. 115–129, 2012. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10556788.2010.517532

[56] YALMIP project wiki, 2015. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/

[57] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, *Wireless sensor networks*. Springer, 2004.

[58] M. Zimmerling, F. Ferrari, L. Mottola, and L. Thiele, "On modeling low-power wireless protocols based on synchronous packet transmissions," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, 2013, pp. 546–555.

[59] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 1–14.

[60] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 73–84.

[61] W. R. Pestman, *Mathematical statistics: an introduction*. Walter de Gruyter, 1998, vol. 1.

[62] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

[63] Gurobi Optimizer 6.0 website, 2015. [Online]. Available: http://www.gurobi.com.

[64] PARAM model checker case study repository, 2015. [Online]. Available: http://depend.cs.uni-sb.de/tools/param/casestudies/.

[65] H. Bohnenkamp, P. van der Stok, H. Hermanns, and F. Vaandrager, "Cost-optimization of the IPv4 Zeroconf protocol," in *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, June 2003, pp. 531–540.

[66] PARAM IPv4 Zeroconf case study, 2015. [Online]. Available: http://depend.cs.uni-sb.de/tools/param/casestudies/#zeroconf.

[67] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[68] PARAM Crowds protocol case study, 2015. [Online]. Available: http://depend.cs.uni-sb.de/tools/param/casestudies/#crowds.

[69] I. O. Kozine and L. V. Utkin, "Interval-valued finite Markov chains," *Reliable Computing*, vol. 8, no. 2, pp. 97–113, Apr. 2002. [Online]. Available: http://link.springer.com/article/10.1023/A%3A1014745904458

[70] D. Škulj, "Discrete time Markov chains with interval probabilities," *International Journal of Approximate Reasoning*, vol. 50, no. 8, pp. 1314 – 1329, 2009, special Section on Interval/Probabilistic Uncertainty. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888613X0900111X

[71] E. Bartocci, R. Grosu, P. Katsaros, C. R. Ramakrishnan, and S. A. Smolka, "Model repair for probabilistic systems," in *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems: Part of the Joint European Conferences on Theory and Practice of Software*, ser. TACAS'11/ETAPS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 326–340. [Online]. Available: http://dl.acm.org/citation.cfm?id=1987389.1987428

[72] K. Chatterjee, K. Sen, and T. A. Henzinger, "Model-checking omega-regular properties of interval Markov chains," in *Foundations of Software Science and Computational Structures*, ser. Lecture Notes in Computer Science, R. Amadio, Ed. Springer Berlin Heidelberg, Jan. 2008, no. 4962, pp. 302–317. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-78499-9_22

[73] T. Chen, T. Han, and M. Kwiatkowska, "On the complexity of model checking interval-valued discrete time Markov chains," *Information Processing Letters*, vol. 113, no. 7, pp. 210–216, Apr. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019013000100

[74] A. Puggelli, W. Li, A. Sangiovanni-Vincentelli, and S. Seshia, "Polynomial-time verification of PCTL properties of mdps with convex uncertainties," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 527–542. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39799-8_35

[75] G. Su and D. Rosenblum, "Asymptotic bounds for quantitative verification of perturbed probabilistic systems," in *Formal Methods and Software Engineering*, ser. Lecture Notes in Computer Science, L. Groves and J. Sun, Eds. Springer Berlin Heidelberg, 2013, vol. 8144, pp. 297–312. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41202-8_20

[76] L. Bortolussi and G. Sanguinetti, "Smoothed model checking for uncertain continuous time Markov chains," *CoRR*, vol. abs/1402.1450, 2014.

Radu Calinescu is a Senior Lecturer in Large-Scale Complex IT Systems at the University of York, UK. Prior to this, he was a Lecturer at Aston University, UK, and a Senior Researcher on the Formal Verification research theme at the University of Oxford. He holds a DPhil in Computation from the University of Oxford, and was awarded a British Computer Society Distinguished Dissertation Award. He has over ten years of academic and industrial research experience in developing complex software systems in areas including adaptive systems, model-driven architectures, and cloud computing. He has chaired or has been on the program committees of multiple international conferences on autonomic, adaptive, and complex systems. He is a Senior Member of the IEEE, and a member of the Editorial Board of Springer's Computing journal.

Carlo Ghezzi is a Professor and Chair of Software Engineering in the Department of Electronics and Information at Politecnico di Milano. He is the Rectors delegate for research, past member of the Academic Senate and of the Board of Governors, and past Department Chair. He is an ACM Fellow, an IEEE Fellow, and a member of the Italian Academy of Sciences (Istituto Lombardo). He received the SIGSOFT Distinguished Service Award. He is a member-at-large of the ACM Council. He is a member of the editorial board of the IEEE Transactions on Software Engineering, Communications of the ACM, Science of Computer Programming, Service Oriented Computing and Applications, and Software Process Improvement and Practice. He was program chair of ESEC, program co-chair of ICSE, and general chair of ICSE and ICSOC. He has been a keynote at ESEC, ICSE, ETAPS, and ICSOC. He is a member of the IFIP WG 2.9 on Requirements Engineering. He has authored over 150 papers in international journals and conferences on various aspects of programming languages and software engineering, and 3 books. His present research interests are in rigorous approaches to the design and evolution of software for pervasive distributed systems.

Kenneth Johnson is a Lecturer in The School of Computer and Mathematical Sciences at Auckland University of Technology, New Zealand. He received his Ph.D. in Computer Science from Swansea University, UK in 2007. He has held post-doctorate research positions at the University of York, Aston University and INRIA, Rennes. His research interests are formal modelling and verification of large-scale systems. Most recently, he has focused on automated model-based analysis of quality-of-service properties of systems at runtime. He is a member of the IEEE, and serves on several program committees for international conferences featuring formal methods and cloud computing technology.

Mauro Pezzé received the Laurea degree in computer science from the University of Pisa, Italy, in 1984, and the doctorate degree in computer science from the Politecnico di Milano, Italy, in 1989. He is a professor of computer science at the University of Milano Bicocca, and the Università della Svizzera italiana, Switzerland. His general research interests are in the areas of software testing and analysis, autonomic computing, self-healing software systems, service-base applications, and service level agreement protection. Prior to joining the University of Milan Bicocca and the University of Lugano as full professor, he was an assistant and an associate professor at the Politecnico di Milano, and a visiting researcher at the University of Edinburgh and the University of California, Irvine. He is an associate editor of the ACM Transactions on Software Engineering and Methodology, and member of the steering committees of the ACM International Conference on Software Testing and Analysis (ISSTA) and the International Conference on Software Engineering (ICSE). He is coauthor of the book Software Testing and Analysis, Process, Principles and Techniques (John Wiley, 2008), and he is the author or coauthor of more than 80 refereed journal and conference papers. He is a senior member of the IEEE, and a member of the IEEE Computer Society.

Yasmin Rafiq is a Ph.D. student at the University of York, UK. She received an MRes in Photonic Networks, and a BSc in Computer Science, both from Aston University, UK. She has active research interests in the areas of run-time modelling and verification of self-adaptive systems. Her current research is focused on online model learning for quality-of-service engineering, with applications to the verification of non-functional requirements for adaptive computer systems, analysis of non-functional requirements of complex software systems, and service-based architectures.

Giordano Tamburrelli is currently an Assistant Professor at Vrije Universiteit in Amsterdam (NL). Previously he has been a Marie Curie Fellow at the USI University in Lugano (CH), and a Ph.D. student at Politecnico di Milano (IT). He received a M.Sc. degree in computer science from the University of Illinois at Chicago (US), and a M.Sc. degree in computer science engineering at the Politecnico di Milano in a joint degree program. He has active research interests in the areas of run-time modeling and verification of systems and software. His main focus is on automated model-based analysis of non-functional requirements of complex software systems and service-based architectures.