POWVER

Technical Report 2016-08

Title:	**Compositional Construction of Importance Functions in Fully Automated Importance Splitting**
Authors:	Carlos E. Budde, Pedro R. D'Argenio, Raúl E. Monti
Report Number:	2016-08
ERC Project:	Power to the People. Verified.
ERC Project ID:	695614
Funded Under:	H2020-EU.1.1. – EXCELLENT SCIENCE
Host Institution:	Universität des Saarlandes, Dependable Systems and Software Saarland Informatics Campus
Published In:	VALUETOOLS 2016

This report contains an author-generated version of a publication in VALUETOOLS 2016.

Please cite this publication as follows:

Carlos E. Budde, Pedro R. D'Argenio, Raúl E. Monti.
Compositional Construction of Importance Functions in Fully Automated Importance Splitting.
10th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2016,
Taormina, Italy, October 26–28, 2016, Proceedings. ICST 2017, ISBN 978-1-63190-141-6. 30–37.

Compositional Construction of Importance Functions in Fully Automated Importance Splitting

Carlos E. Budde, Pedro R. D'Argenio, and Raúl E. Monti
FAMAF, Universidad Nacional de Córdoba – CONICET
{cbudde,dargenio,rmonti}@famaf.unc.edu.ar

ABSTRACT

Importance splitting is a technique to accelerate discrete event simulation when the value to estimate depends on the occurrence of rare events. It requires a guiding *importance function* typically defined in an *ad hoc* fashion by an expert in the field, who could choose an inadequate function. In this article we present a compositional and automatic technique to derive the importance function from the model description, and analyze different composition heuristics. This technique is linear in the number of modules, in contrast to the exponential nature of our previous proposal. This approach was compared to crude simulation and to importance splitting using typical *ad hoc* importance functions. A prototypical tool was developed and tested on several models, showing the feasibility and efficiency of the technique.

1. INTRODUCTION

Nowadays, systems are required to have a high degree of resilience and dependability. Determining properties that fail with extremely small probability in complex models can be computationally very demanding. Though such properties can be efficiently calculated using numerical tools, this is limited to finite Markov models, and, moreover, the representation through an adequate data structure needs to fit in the computer memory. Beyond this class of models, calculations are limited to Monte Carlo simulation methods. However, standard Monte Carlo simulation is impractical when the probability of the event under analysis is extremely low: it will easily require an enormous amount of sampling to obtain an acceptable confidence level of the estimated probability, in order to compensate for the high variance induced by the rare occurrences of such event.

To reduce this considerable need for simulation runs, efficient Monte Carlo simulation techniques have been tailored to deal with rare events. These can be largely divided into two conceptually different techniques: *importance sampling* and *importance splitting* methods. We focus on *importance splitting* techniques, see e.g. [14, 18, 19]. Importance splitting works by decomposing the state space in multiple levels where, ideally, the rare event is at the top level and a level is higher as the probability of reaching the rare event grows. The estimation of the rare probability is obtained as the product of the estimates of the (not so rare) conditional probabilities of moving one level up. As a consequence, the effectiveness of this technique crucially depends on an adequate grouping of states into levels. *Importance functions* are the means to assign a value to each state so that, if perfect, such value is directly related to the likelihood of reaching the rare event. It is desirable that a state in the rare set receives the highest importance and the importance of a state decreases according to the probability of reaching a rare state from it.

Usually, an expert in the area of the system provides the importance function in an *ad hoc* manner. A badly chosen function can deteriorate the effectiveness of the technique. With some notable exceptions [1,8,11,15], automatic derivation of importance functions has received scarce attention.

In [1] we presented preliminary results on an effective technique to derive automatically an importance function. The algorithm works by applying inverse breadth first search (BFS) on the underlying graph of the stochastic process, labelling each state with the shortest distance to a rare state. The importance of each state is then defined as the difference between the maximum distance and its actual distance. Though this technique is not limited to Markov models, it still requires a finite graph which fits in the computer memory. Unfortunately such graph grows exponentially with the number of modules that conform the model of the system.

In this paper, we improve on this technique by obtaining the importance function in a compositional manner. We consider the system modelled as a network of interacting modules, where each module is described in terms of an input/output stochastic automaton (IOSA) and the interaction is defined through standard parallel composition [6]. The technique we propose works by applying the method of [1] per module, previous analysis of how the local states relate to the property under study, and the final importance function is obtained by composing the modular functions.

VALUETOOLS '16 October 26–28, 2016, Taormina, Italy
© 2018 ACM. ISBN 978-1-63190-141-6.
DOI: 10.4108/eai.25-10-2016.2266501

Contrarily to the technique of [1], this way of calculating the importance function grows linearly with the number of modules that conform the system model.

The paper is organized as follows. Sec. 2 explains the foundations of our specification language. Sec. 3 briefly describes the importance splitting technique and the RESTART method. In Sec. 4 we introduce our technique for the compositional derivation of importance functions. Sec. 5 reports the performance of our technique on four different case studies. The paper concludes in Sec. 6.

2. FORMAL SETTING

2.1 System models

We describe our models using input/output stochastic automata (IOSA) [6]. An IOSA is a variant of stochastic automata [4, 5] which allows the description of stochastic decision processes in which the occurrence of events (or actions) are governed by continuous random variables called clocks. IOSA considers two disjoint type of actions: input actions and output actions, which should synchronize when different IOSAs interact in a parallel composition. Output actions are locally controlled by the IOSA modelling a particular module. Thus, the occurrence time of output actions is controlled by a random variable. Instead, inputs are externally controlled and hence their occurrence time can only depend on their interaction with outputs in a parallel composition.

Definition 1. *An* input/output stochastic automaton *(IOSA for short) is a structure $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of labels partitioned into disjoint sets of input labels $\mathcal{A}^I$, and output labels $\mathcal{A}^O$, $\mathcal{C}$ is a finite set of clocks such that each $x \in \mathcal{C}$ has associated a continuous probability measure μ_x on $\mathbb{R}$ s.t. $\mu_x(\mathbb{R}_{>0}) = 1$, $\rightarrow \subseteq \mathcal{S} \times \mathcal{C} \times \mathcal{A} \times \mathcal{C} \times S$ is a transition relation (we generally write $s \xrightarrow{C,a,C'} s'$ instead of $(s, C, a, C', s)' \in \rightarrow$), $\mathcal{C}_0$ is the set of clocks that are initialized in the initial state, and $s_0 \in \mathcal{S}$ is the initial state. In addition an IOSA should satisfy the following constraints:*

1. *If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^I$, then $C = \emptyset$.*
2. *If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^O$, then C is a singleton set.*
3. *If $s \xrightarrow{\{x\},a_1,C_1} s_1$ and $s \xrightarrow{\{x\},a_2,C_2} s_2$ then $a_1 = a_2$, $C_1 = C_2$ and $s_1 = s_2$.*
4. *If $s \xrightarrow{\{x\},a,C} s'$ then, for all $t \xrightarrow{C_1,b,C_2} s$, either $x \in C_2$, or $x \notin C_1$ and there exists $t \xrightarrow{\{x\},c,C_3} t'$.*
5. *If $s_0 \xrightarrow{\{x\},a,C} s$ then $x \in C_0$.*
6. *For every $a \in \mathcal{A}^I$ and state s, there exists $s \xrightarrow{\emptyset,a,C} s'$.*
7. *For every $a \in \mathcal{A}^I$, if $s \xrightarrow{\emptyset,a,C_1} s_1$ and $s \xrightarrow{\emptyset,a,C_2} s_2$, $C'_1 = C'_2$ and $s_1 = s_2$.*

The occurrence of an output action is controlled by the expiration of clocks. Thus, whenever $s \xrightarrow{\{x\},a,C} s'$ and the system is in state s, output action a will occur once the value of clock x reaches 0. At this point, the system moves to state s' setting the values of every clock $y \in C$ to a value sampled according to the distribution μ_y. An input transition $s \xrightarrow{\emptyset,a,C} s'$, can potentially occur at any time which will be defined once the action interacts with an output.

Restrictions 1-7 ensure that: 1. input and output behave in reactive and generative fashion, respectively, 2. the class of IOSAs is closed w.r.t. parallel composition, and 3. every *closed* IOSA (i.e., an IOSA with no input actions), is deterministic (or fully probabilistic) in the sense that it almost never reaches a state where two possible actions can be taken at the same time. See [6] for detail explanations and proofs. We remark that a closed IOSA defines a generalized semi-Markov process (GSMP). This is important since fully probabilistic models, such as GSMPs, are amenable to simulation in the general case.

Table 1: Parallel composition of IOSAs

$$\frac{s_1 \xrightarrow{C,a,C'}_1 s'_1 \quad a \in \mathcal{A}_1 \backslash \mathcal{A}_2}{s_1 || s_2 \xrightarrow{C,a,C'} s'_1 || s_2} \qquad \frac{s_2 \xrightarrow{C,a,C'}_2 s'_2 \quad a \in \mathcal{A}_2 \backslash \mathcal{A}_1}{s_1 || s_2 \xrightarrow{C,a,C'} s_1 || s'_2}$$

$$\frac{s_1 \xrightarrow{C_1,a,C'_1}_1 s'_1 \quad s_2 \xrightarrow{C_2,a,C'_2}_2 s'_2}{s_1 || s_2 \xrightarrow{C_1 \cup C_2,a,C'_1 \cup C'_2} s'_1 || s'_2}$$

Since we intend outputs to be autonomous (or locally controlled), we do not allow synchronization between outputs. Besides, we need to avoid name clashes on the clocks, so that the intended behaviour of each component is preserved and also to ensure that the resulting composed automata is indeed an IOSA. Thus we require to compose only *compatible* IOSAs, and only compatible IOSAs are allowed to compose.

Definition 2. *Two IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$ are said to be* compatible *if they do not share output actions nor clocks.*

Given two compatible IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$, the parallel composition *$\mathcal{I}_1 || \mathcal{I}_2$ is a new IOSA $(\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0^1 || s_0^2)$ where (i) $\mathcal{A}^O = \mathcal{A}_1^O \cup \mathcal{A}_2^O$ (ii) $\mathcal{A}^I = (\mathcal{A}_1^I \cup \mathcal{A}_2^I) \setminus \mathcal{A}^O$ (iii) $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ (iv) $\mathcal{C}_0 = \mathcal{C}_0^1 \cup \mathcal{C}_0^2$ and $\rightarrow$ is the smallest relation defined by rules in Table 1 where we annotate $s||t$ instead of (s, t).*

The parallel composition works as the usual CSP parallel composition [10]: The two IOSAs should synchronize on actions of equal name, but each component may execute independently if the action belongs only to it in an interleaving manner. Thus, the first two rules in Table 1 correspond to the interleaving case in which each component proceeds independently, while the third corresponds to the synchronization case. Due to compatibility, synchronization only happens between an input and an output. Moreover the union $C_1 \cup C_2$ of clocks in the conclusion of the third rule let the input of one component inherit the clock of the output from the other component, and moreover, this action becomes an output in the resulting composed IOSA.

So far we gave an introduction to the fundamentals of IOSA. However, our prototype tool actually implements a more friendly syntax similar to that of PRISM [13], using modules and integer and boolean variables, with guarded expressions. Thus, a local state in a module is given by particular valuations of the local variables, and a global state is given by the union of local states (w.l.o.g. we assume that different modules have different variables names).

2.2 Rare event properties

In this work we study transient and long run logical properties interpreted on the state space of an IOSA. *Transient* properties are used to calculate the probability of reaching a set G of *goal states* before visiting any *reset state* from the (disjoint) set R. (For simulation purposes the probability of reaching a state in $G \uplus R$ has to be 1.) Let $bexp_1$ and $bexp_2$ be two *propositional formulas* which, when interpreted on the states of an IOSA, identify the complement of R and the set G respectively. Then the transient property is characterized by the PCTL formula

$$\mathsf{P}(bexp_1 \,\mathsf{U}\, bexp_2) \qquad (1)$$

where U denotes the unbounded until operator from LTL and $\mathsf{P}(\Phi)$ denotes the probability of observing any state that satisfies formula Φ.

We also study the behaviour of systems in the long run or steady state situation. *Long run* analysis focuses on the quantification of a property once the system has reached an equilibrium. In particular, the steady state probability of a set G of *goal states*, identified by some propositional formula $bexp$ in the state space of an IOSA, is the portion of time in which any state in G is visited in the long run and is characterized by the CSL property

$$\mathsf{S}(bexp) \qquad (2)$$

Both types of properties are recurring in the literature of rare event simulation, see e.g. [3, 7, 9, 19] for the case of properties like (1), and [16, 19] for properties like (2).

3. IMPORTANCE SPLITTING

When a parameter is estimated using Monte Carlo simulation, the speed and overall efficiency of the method is highly dependent on the precision required for the estimate. Confidence intervals are commonly used to convey a notion of how far the produced estimate may be from the actual value. As a general rule, whichever the confidence interval construction method, the simulations "length" grows with the tightness desired for the interval. In particular several rare event scenarios are known to require a number of samples which grows exponentially on the model size [12].

Importance splitting (IS for short) aims to speed up the occurrence of a rare event without modifications on the system dynamics (see [14] and references therein.) The general idea in IS is to favour the "promising runs" that approach the rare event by saving the states they visit at certain predefined checkpoints. Replicas of these runs are created from those checkpoint states, which continue evolving independently from then on. Contrarily, simulation runs deemed to steer away from the rare event are identified and killed, avoiding the use of computational power in fruitless calculi. The likelihood of visiting a goal state from any other state s is called the *importance* of s. Variations in such importance determine when should a simulation run be split or killed, as the importance value crosses some given *thresholds* up or down, respectively. This procedural description of IS has a sound statistical equivalent. For a comparison see [1].

We focus on the RESTART method, a version of IS with multiple thresholds, fixed splitting and deterministic discards of unpromising simulations [16–20]. A RESTART run can be depicted as in Fig. 1 where the horizontal axis represents the simulation progress and the vertical axis the importance value of the current state. The run starts from an initial state and evolves until the first threshold T_1 is crossed *upwards*. This takes the path from zone Z_0 below threshold T_1 into zone Z_1 between T_1 and T_2. As this happens the state is saved and $s_1 - 1$ replicas or *offsprings* of the path are created. See A in Fig. 1, where the *splitting* for T_1 is $s_1 = 3$. This follows the idea of rewarding promising simulations: up-crossing a threshold suggests the path heads towards a goal state. From then on the s_1 simulations will evolve independently. As they continue, one of them may hit the upper threshold T_2, activating the same procedure: $s_2 - 1$ offsprings are generated from it and set to evolve independently. See B on T_2; here, the splitting is $s_2 = 2$.

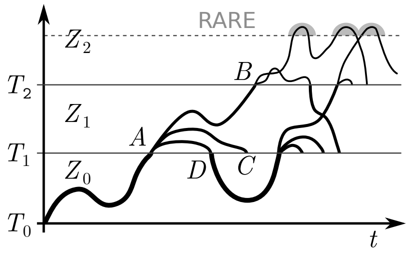

Figure 1: RESTART importance splitting

However, it could also happen that some simulation hits T_1 again, meaning the path is leading *downwards*. This simulation steers away from the goal set and RESTART deals with it discarding the run right away (see C in Fig. 1). In each zone Z_i there exists nonetheless an *original simulation*, which crossed threshold T_i upwards generating the $s_i - 1$ offsprings. This run is allowed to survive a down-crossing of threshold T_i (see D in Fig. 1).

In this setting all simulations reaching a goal state went through the replication procedure, which stacked up on every threshold crossed. Simply counting these hits would introduce a bias, because the *relative weight* of the runs in upper zones decreases by an amount equivalent to the splitting of the thresholds. In consequence, each rare event observed is pondered by the relative weight of the simulation from which it stemmed. If all the goal states exist beyond the uppermost threshold like in Fig. 1, then it suffices to divide the observed quantity of rare events by $\text{SPLIT}_{\mathsf{MAX}} \doteq \prod_{i=1}^{n} s_i$. Otherwise more involved labelling mechanisms are needed.

4. MODULAR COMPUTATION OF THE IMPORTANCE FUNCTION

It is clear from Sec. 3 that importance splitting simulations are entirely guided by the *importance function* which defines the importance of each state. This function conveys the locations where the simulation effort should be intensified, and it is from its definition that many other settings of IS are usually derived. Importance functions are defined in most situations in an *ad hoc* fashion by an expert in the field of the particular system model. With a few exceptions in some specific areas ([8, 11, 15]) automatic derivation of importance functions is still a novel field for general systems.

4.1 Basic importance function derivation

Consider a single-module system $\mathcal{M}$ described in e.g. the IOSA language. The importance of each state s in $\mathcal{M}$ is formally defined as the probability of observing the rare event after visiting s. In the setting of the logical properties $\mathsf{P}(bexp_1 \,\mathsf{U}\, bexp_2)$ and $\mathsf{S}(bexp)$ presented in Sec. 2, the rare event is the set G of *goal states*, defined as the system states satisfying the boolean expressions $bexp_2$ and $bexp$ respectively. Therefore if one could track or at least conjecture a path leading from s to a goal state, some notion of the distance between s and the rare event may be determined and used to choose an appropriate importance for the state.

A simultaneous backwards-reachability analysis starting from the set G serves this purpose. Each layer of states visited is labelled with decreasing importance, computing the shortest path leading from each state to G by means of a Breadth-First Search routine of complexity $\mathcal{O}(k \cdot n)$, where n is the size of the state space and k is the branching degree of the underlying graph of the model. Albeit $k = n$ in the worst case, k is normally several orders of magnitude smaller than n. The pseudocode for this algorithm is shown in Fig. 2, where s_0 stands for the initial state of $\mathcal{M}$.

Using this strategy one can indeed obtain in very short computational time a good importance function to use with the IS technique of choice [1]. The thresholds can then be selected either arbitrarily, using e.g. some fixed approach ("*set one every three importance values*"), or adaptively by means of an algorithm that exercises the model dynamics. Whenever feasible we prefer the later alternative due to its flexibility and verified efficiency. Adaptive computation of the thresholds has been observed to skip, for instance, the lowermost importance values and place the first threshold at importance $T_1 > 0$, resulting in a decrease of the splitting overflow and a consequent improvement in the estimation times/precision. A few techniques have been devised in the field of adaptive algorithms, most notably *Adaptive Multilevel Splitting* and its successor *Sequential Monte Carlo* by F. Cérou and A. Guyader in [2, 3].

4.2 The modular approach

One of the strengths of the IOSA language is the possibility to describe the system as a set of simpler disjoint modules whose parallel composition captures the desired behaviour. However, the algorithm from Fig. 2 works on a single module. The question is how to apply that to a *set of modules*.

A simple *monolithic* solution is a straightforward application of IOSA's compositionality: since the parallel composition of compatible IOSA modules is itself an IOSA, we could as well construct such global module (from now on referred to as *the model*) and derive the importance function as previously described. This approach is, as a matter of fact, not only feasible but also effective, as it will be shown in Sec. 5. It is nonetheless also limited in scope since it does not scale. The BFS algorithm requires an explicit representation of the state space of the composed IOSA (and actually of the whole adjacency matrix), which grows exponentially with the number of modules involved in the composition.

```
Input: module $\mathcal{M}$
Input: goal state set $G \neq \varnothing$
$g(G) \leftarrow 0$
queue.push($G$)
repeat
  $s \leftarrow$ queue.pop()
  for all $s' \in \mathcal{M}$.predecessors($s$) do
    if $s'$ not visited then
      $g(s') \leftarrow g(s) + 1$
      queue.push($s'$)
    end if
  end for
until queue.is_empty() or $s_0$ visited
$g(s) \leftarrow g(s_0)$ for every non visited state $s$
$f(s) \leftarrow g(s_0) - g(s)$ for every state $s$
return importance function $f$
```

Figure 2: Basic importance function derivation

In order to avoid this problem it is essential to find a modular solution, some way to compute and store the importance separately on each module. The goal is then to build *local importance functions*, that is, local to each module in the system, which will be used later to construct the *global importance function* of the model.

The first challenge one finds in this direction is the identification of the rare event in the local state of each module. Recall that the propositional formulas of the properties from Sec. 2 (viz. *bexp*, *bexp*$_1$ and *bexp*$_2$) are interpreted on the state space of the composed model. The names of variables from any module can occur in the expressions of these formulas. Since we need to identify, in every module taken individually, the local states corresponding to the global rare event, the question is then how to interpret such "global formulas" locally on each module.

To illustrate the issue consider the example of a tandem-network whose queues q_1 and q_2 have capacity $C < \infty$. Declare one module per queue, as it might come naturally, and let us study the long-run property (2) for three definitions of the rare event *bexp*: (a) $q_2 = C$, (b) $q_1 = C \;\wedge\; q_2 = C$ and (c) $q_1 > \frac{C}{2} \;\Rightarrow\; q_2 = C$. All three definitions speak of an overflow in the second queue, but the role of the first queue is harder to grasp. In (a) variable q_1 is missing, so we ignore the module of the first queue when deriving the local importance functions, as it does not change the validity of the formula $q_2 = C$. In other words, for (a) the local importance function of the module of the first queue should be null. In (b) q_1 does appear in *bexp*; more precisely one may define the states satisfying $q_1 = C$ as the *local rares*, and apply the backwards BFS from them. Finally, (c) should be analyzed more carefully, since there *bexp* $\equiv \neg\big(q_1 > \frac{C}{2}\big) \;\vee\; q_2 = C$. Therefore the local rares in the module of the first queue are exactly those which *do not* satisfy $q_1 > \frac{C}{2}$.

The difficulty is hence, when standing on a single module, how to interpret the occurrence of its local variables in the atomic propositions from the formulas *bexp*, *bexp*$_1$ and *bexp*$_2$. As we have seen, it is not necessarily clear whether such occurrences should be taken positively or negatively.

To cope with this difficulty we ask for the formulas *bexp*, *bexp*$_1$ and *bexp*$_2$ to be expressed in *disjunctive normal form* (*DNF*), i.e. a disjunction of *clauses*, each of which is a conjunction of *literals* (i.e. of atomic propositions or negated atomic propositions). This approach imposes no restriction on the description of the rare event, since any propositional formula can be equivalently written in DNF. The difficulty in the interpretation of the atomic propositions is however solved since the literal clearly states if it should be interpreted positively or negatively.

The procedure to follow is now direct: first project, for the current module under study, the rare event DNF formula into the set of local variables. If the projection is empty then this module plays no role in the rare event and ought to be discarded for importance computation. Otherwise use the resulting formula, also in DNF and containing solely local variables names, to identify the local rare states in this module. Once this set is known the basic importance function derivation can be applied. The pseudocode of this algorithm is presented in Fig. 3.

This way we compute and store the local state importance in a per-module basis, which *grows linearly* with the number of modules in the model. The result is a notion of local importance functions, which still need to be combined amongst modules to compute the global importance function.

4.3 Composition of local importance functions

In the description of RESTART (Sec. 3) the technique is oblivious of the way in which the importance of the system is computed or stored. RESTART simply needs the importance of the current state of the model after every transition is taken. The same transparency is required by all known IS techniques and even by the adaptive algorithms from [2, 3]. Hence if we are to use the modular approach described in Sec. 4.2, we need a composition procedure or function to decide the importance of each state of the composed model, taking as input the local importance of each module.

One option is to let the user settle the matter via an *ad hoc* choice. He would have to provide an algebraic expression using the local importance functions as variables, which would be used at every step of the simulation to combine the local importance values. Recall for instance the tandem-network example from Sec. 4.2 and let `Queue1` and `Queue2` be the local importance functions of the modules representing the first and second queue respectively. The user could specify composition functions such as `Queue1+Queue2`, `0.5 Queue1 + Queue2`, or `(1+Queue1)*(1+Queue2)`.

Alternatively, we could use an associative binary operator to combine all local importance functions with. Natural candidates are addition, product, `max` and `min`. As an example take the first definition of the rare event in the tandem-network, $q_2 = C$. Using `+` or `max` as compositional operand would yield the same result, since the first queue has a null local importance function. If on the other hand we take the second rare event definition, $q_1 = C \wedge q_2 = C$, then `max` and `+` will behave differently, most likely in favour of the later.

The previous analysis shows the definition of the rare event, or more generally the expression of the property, may have a critical impact on the efficiency of the composition function chosen. Furthermore, since we request the properties to be expressed in DNF, we could exploit the structure of the formula to identify specific arithmetical operands or even algebraic structures to associate to each logical operand. We are currently investigating a way to automatically map the disjunctions and conjunctions to their best-match arithmetical counterparts. Our last studies are leading us towards the use of semi-rings such as `(max,+)` and `(+,*)`, which could be thought of as naturally corresponding to the `(`$\vee$`,`$\wedge$`)` structure of DNF formulas. The results of these studies and also of the comparisons between the different ways to build the importance function (*ad hoc*, automatic monolithic and automatic per-module) are presented in Sec. 5.

```
Input: modules set $\{\mathcal{M}_i\}_{i=1}^n$
Input: global rare event formula $bexp$
$bexp$.assert_DNF()
for all $\mathcal{M}_i$ do
    $bexp_i \leftarrow bexp$.project($\mathcal{M}_i$.variables())
    if $bexp_i \equiv$ true then
        $f_i \leftarrow$ null
    else
        $G_i \leftarrow \mathcal{M}_i$.identify_states ($bexp_i$)
        $f_i \leftarrow$ derive_importance_function ($\mathcal{M}_i, G_i$)
    end if
end for
return local importance functions set $\{f_i\}_{i=1}^n$
```

Figure 3: Local importance functions computation

As a final remark notice that using the product to combine local importance functions could lead to problems whenever a null importance value is encountered. As a workaround in such cases the functions where updated after construction, replacing every importance value i with 2^i (e.g. the values $0, 1, 2, \ldots$ map into $1, 2, 4, \ldots$) This solved the issue and set the computed importance values further apart, with interesting consequences in the IS simulations.

5. EXPERIMENTAL VALIDATION

To validate our approach we selected four case studies from the literature and analyzed them using a self-developed software tool. We remark the tool is still in a prototypical stage, accounting for the high timeouts chosen for the executions. To validate correctness, the results estimated with IS simulation were compared against the published results in all cases where an exact IOSA reproduction of the model could be devised. Otherwise, and restricted to markovian systems, a numerical approximation by the model checking tool PRISM was used as reference value. Our estimates where also compared against the analytic solution of the problem whenever this was available.

All estimates were computed together with a confidence interval, i.e., the confidence level and precision of the estimations were taken into account. Two methods were used to build these intervals depending on the type of the property. The standard interval $\left[\,\overline{X} - z_{1-\frac{\alpha}{2}} \cdot {}^{\hat{\sigma}}\!/\!\sqrt{N}\ ,\ \overline{X} + z_{1-\frac{\alpha}{2}} \cdot {}^{\hat{\sigma}}\!/\!\sqrt{N}\,\right]$ was used for estimates corresponding to the long run properties as in (2), where N is the number of independent simulation runs, $\overline{X}$ stands for the mean value of the sample, $100(1-\alpha)\%$ is the confidence level and ${}^{\hat{\sigma}}\!/\!\sqrt{N} \cdot z_{1-\frac{\alpha}{2}}$ is the semi-precision or error margin. Transient properties like (1) are inherently binomial, because every simulation almost surely succeeds or fails to encounter a rare event, and one is after the (weighed) number of successful simulations. Since the samples are binomially distributed we can choose a confidence interval construction method better fitted than the standard method described above. The *Wilson score interval* [21] is specially tailored for situations when the proportion $p \doteq \frac{\#\mathsf{succ}}{N}$ takes extreme values, viz. $p \approx 0$ or $p \approx 1$, which clearly fits our situation best.

The precision of the intervals was chosen relative to the expected value of the estimate. This means that the smaller the probability of the rare event, the tighter the confidence interval built for the estimate. Even though this has an obviously negative impact on the simulation times, the goal was to generate as useful an interval as possible.

Simulations were requested to reach a specified confidence level and precision within certain time limit. The precise values of the three criteria (i.e. the confidence, the relative precision and the timeout) are subject to each particular experiment. Simulations reaching the wall time limit were truncated; in these cases the data of interest is the precision achieved at truncation. For each experiment we varied some model parameter, testing the performance of the simulation methods for decreasing values of the probability of the rare event (henceforth "p"), which overall ranged from $p \approx 1.02 \cdot 10^{-2}$ to $p \approx 2.45 \cdot 10^{-13}$. From now on IFUN will denote "importance function". To validate the performance of our approach, for each model and parameter value we tested the following simulation strategies where applicable: RESTART using the *compositional* IFUN described in Sec. 4, RESTART using the *monolithic* IFUN built from the composed model (the "monolithic solution" from Sec. 4.2), RESTART using any known or devised *ad hoc* IFUN, and *standard Monte Carlo*. All RESTART simulations were tested with different split values.

Experimentation was carried out in two computer settings: the cluster *Mendieta* featuring 8-cores 2.7 GHz Intel Xeon E5-2680 processors, each with 32 GiB 1333MHz of available DDR3 RAM; and the server *Jupiteracе*, with a 12-cores 2.40GHz Intel Xeon E5-2620v3 processor and 128 GiB 2133MHz of available DDR4 RAM.

5.1 Tandem queue

Consider a tandem network consisting of two connected queues where customers arrive at the first one following a Poisson process with rate λ. After being served by server 1 at rate μ_1 they enter the second queue, where they are attended by server 2 at rate μ_2. The event of interest is an overflow in the second queue for maximum capacity $C < \infty$. This model has received considerable attention in the literature [7–9,19]. We studied this model in [1] and use it here to validate the correction of the tool used for experimentation.

We follow the setting from [7] which has an exact analytic solution. The first queue is initially empty and the second has a single customer. We measured the probability of full occupancy in the second queue before it emptied, i.e. an instance of Property (1). The model was simulated for $(\lambda, \mu_1, \mu_2) = (3, 2, 6)$. The maximum capacities studied were $C = 8, 10, 12, 14$, for which the corresponding probabilities are 5.62e-6, 3.14e-7, 1.86e-8, and 1.14e-9. Simulations had to reach a 90% confidence level with precision equal to 20% of the estimate. Estimates were successfully validated against the exact analytic solution. RESTART simulations were run for split values $2, 3, 6, 9, 12$. Experimentation was performed in Mendieta, comparing standard MC and RESTART simulations using the IS strategies *compositional*, *monolithic*, and the *ad hoc* 'q_2', viz. counting the second queue occupancy.

Standard Monte Carlo simulations were the slowest by a factor of at least 5x and failed to meet the stopping criteria for $C > 10$ within 2 hours of wall time execution. Contrarily, none of the RESTART simulations timed-out and the performance obtained with all IFUNs is comparable. Overall RESTART simulations took 2 to 17 seconds for $C = 8$ (mean time of 8.7 s), 7 to 119 s for $C = 10$ (mean: 44.7 s), 20 to 550 s for $C = 12$ (mean: 171.2 s), and 88 to 3242 s for $C = 14$ (mean: 33 m). Times varied significantly for the different split values, with no best candidate for all queue sizes and importance techniques. The automatic derivation of the importance function never took longer than 30 ms.

There was a strong similarity in performance between the automatically computed *compositional* IFUN and the *ad hoc* strategy 'q_2'. This was expected: since the property involves a variable from the module of the second queue only, the function derived by our algorithm ignores the module of the first queue and yields something very much like counting the elements of the second queue. Notice 'q_2' is the best *ad hoc* strategy reported in [1] for the tandem queue model.

We also studied the long run behaviour for the saturation of the second queue. The maximum capacities studied were $C = 10, 13, 16, 18, 21$, for which the corresponding long run probabilities are 7.25e-6, 2.86e-7, 1.12e-8, 1.28e-9, and 4.94e-11. Simulations had to reach a 90% confidence level with precision of 40%. Estimates were successfully validated against a numerical approximation by PRISM. We tested the same IS strategies mentioned above with the split values $2, 3, 6, 11$. Experiments were run in Mendieta.

Table 2: Simulation times on triple tandem queue

IFUN	Split	**a**	**b**	**c**	**d**	**e**	**f**
		h:mm	*h:mm*	*h:mm*	*h:mm*	*h:mm*	*h:mm*
mono.	3	0:18	0:08	0:33	0:19	*4.9e-10	0:16
	6	0:21	0:04	0:04	0:52	0:29	1:17
	11	0:11	0:02	0:04	0:05	0:02	0:08
comp.	3	0:58	0:11	0:28	1:22	1:12	1:19
	6	0:24	0:03	0:05	0:06	0:30	0:10
	11	0:15	0:03	0:09	2:28	3:41	1:51
ah$_1$	3	1:20	0:18	0:41	2:56	3:05	*7.4e-10
	6	0:11	0:04	0:20	0:11	1:30	0:41
	11	0:27	0:06	0:08	*1.3e-9	*4.3e-10	2:47
ah$_2$	3	1:15	0:10	0:57	0:36	0:58	3:45
	6	0:26	0:18	0:04	1:06	3:12	1:29
	11	0:10	0:22	0:13	0:13	0:34	3:13
MC		*4.1e-9	1:37	*4.7e-8	*8.5e-9	*1.0e-9	*2.6e-9

Again standard Monte Carlo took always the longest, timing out after 6 hours for the two biggest queue sizes. For $C = 21$ some RESTART simulations also timed out, depending non-uniformly on the split value. Overall RESTART simulations took 1 to 4 s for $C = 10$ (mean: 2.33 s), 4 to 48 s for $C = 13$ (mean: 24.1 s), 9 to 6702 s for $C = 16$ (mean: 20 m) 1 to 166 m for $C = 18$ (mean: 33 m), and for $C = 21$ the only successful simulations within the time bound where the *ad hoc* strategy for splitting 6 and 11, the compositional strategy for splitting 3 and 11, and the monolithic strategy for splitting 6. As before the automatic derivation of the importance function never took longer than 30 ms.

5.2 Triple tandem queue

Consider now a non-markovian tandem network operating under the same principles but consisting of three queues with Erlang-distributed service times. External arrivals follow a Poisson process with rate $\lambda = 1$. The shape parameter α of the service times is either 2 or 3 consistently in all queues. The load at the third queue is always $^1/_3$, meaning the scale parameter μ_3 of the Erlang in the third queue takes the values $^1/_6$ and $^1/_9$ when α is 2 and 3 respectively. The rare event of interest is the long run saturation probability at the third queue, i.e. an instance of Property (2).

This model was studied in [16] starting from initially empty queues. The scale parameters μ_1 and μ_2 at the first and second queues, as well as the threshold capacity C at the third queue, were chosen to keep the steady state probability in the same order of magnitude for all case studies. We chose a rare event probability of the order of $5 \cdot 10^{-9}$. The values $(\alpha, \mu_1, \mu_2, C)$ for the case studies **a**–**f** are $[(2, {}^1/_3, {}^1/_4, 10), (3, {}^2/_3, {}^1/_6, 7), (2, {}^1/_6, {}^1/_4, 11), (3, {}^1/_9, {}^1/_6, 9), (2, {}^1/_{10}, {}^1/_8, 14), (3, {}^1/_{15}, {}^1/_{12}, 12)]$. Simulations had to reach a 90% confidence level with precision equal to 20% of the estimated parameter within 4 hours of wall time execution. Experimentation was carried out in Jupiteracе and the resulting simulation times are in Table 2, where "mono.", "comp." and "ah" stand for the IS strategies *monolithic*, *compositional* and *ad hoc* respectively. Simulations which failed to meet the confidence criteria within 4 hours are labelled '*' and the achieved precision for a 90% confidence level is reported if available.

Table 3: Simulation times on queue with breakdowns

IFUN	Split	K40	K60	K80	K100	K120	K140	K160
		m:ss	*m:ss*	*m:ss*	*m:ss*	*m:ss*	*m:ss*	*m:ss*
mono.	3	0:03	0:18	6:50	315:54	2:43	*5.4e-11	6:44
	6	0:01	0:10	0:14	46:55	*4.9e-11	*6.2e-12	*-
	9	0:01	1:01	3:33	1:44	4:15	0:46	*1.5e-12
	12	0:01	0:08	40:18	0:23	0:44	1:50	6:23
comp.	3	0:03	0:19	5:04	333:23	1:04	*5.4e-11	6:19
	6	0:01	0:18	0:19	48:18	*4.9e-11	*6.2e-12	*-
	9	0:01	1:01	3:17	0:39	8:04	0:42	*1.5e-12
	12	0:01	0:09	54:42	0:39	1:22	2:30	5:58
ad hoc	3	0:03	0:18	4:51	311:15	1:04	*5.4e-11	5:51
	6	0:00	0:16	0:17	43:57	*4.6e-11	*6.2e-12	*-
	9	0:01	0:57	3:07	0:36	7:37	0:39	*1.5e-12
	12	0:01	0:08	50:37	0:36	1:14	2:08	5:23
MC		0:17	10:26	317:56	*6.5e-9	*2.8e-9	*-	*-

All but one of the standard Monte Carlo simulations failed within the time bound imposed. Two *ad hoc* importance strategies were tested: counting just the occupancy in q_3 (ah_1) and counting also the occupancy in the first two queues with the weight coefficients proposed in [16] (ah_2). Our results suggest their performance is strongly dependent on the splitting used, with no clear optimal choice, see e.g. case studies **c** and **e**. The general behaviour of the compositional IFUN resembles that of ah_1 but proved better in most scenarios. In any case and letting aside a few outliers (e.g. **d** and **f** for splitting 6 and **e** for splitting 3), the monolithic global IFUN was the fastest in all case studies. We believe this is due to the amount of information available during the iterations of the basic derivation algorithm (Fig. 2), which spreads along the whole model and thus considers the state of all three queues. The main advantage in comparison to similar *ad hoc* approaches like ah_2 is the automatization of the method, here also crowned by an excellent performance.

5.3 Queue with breakdowns

This case study is taken from [12] and models a network where the inputs come from several sources operating in parallel, which are of type $i \in \{1, 2\}$ and have exponential on/off times with parameters α_i and β_i respectively. Whenever a source of type i is active it sends packets at rate λ_i to the only system buffer. Queued packets are handled by a server which breaks down at rate γ and gets fixed at rate δ, processing at rate μ when functional. We estimate the probability of the buffer reaching maximum capacity K before emptying, i.e. an instance of Property (1).

The system starts with a single queued packet and a broken server. There are five sources of each type, all down initially except for one of type 2. The sources parameters are $(\alpha_1, \beta_1, \lambda_1) = (3, 2, 3)$ and $(\alpha_2, \beta_2, \lambda_2) = (1, 4, 6)$, the server parameters are $(\gamma, \delta, \mu) = (3, 4, 100)$ and the queue capacities tested were $K = 40, 60, 80, \ldots, 160$, for which the corresponding probabilities are 4.59e-4, 1.25e-5, 3.72e-7, 9.59e-9, 2.86e-10, 8.44e-12, and 2.45e-13.

We set the confidence level at 90%, the precision at 40% and the wall time limit at 6 hours. Results from computations performed in Mendieta are shown in Table 3. Standard Monte Carlo simulations timed-out for $K > 80$. The importance strategies used in RESTART simulations were the automatic 'compositional' and 'monolithic' variants, and the best *ad hoc* approach reported in [1], namely counting the number of queued packages. All three showed very similar performance in spite of the high variance related to the splitting value chosen. The best simulation times for $K = 40, 80, 160$ were ∼1 s, 14 s, and 323 s respectively. The corresponding IFUN derivation times were below 4.4 s, 9.32 s, and 23.11 s for the monolithic approach, and below 0.17 s, 0.34 s, 0.64 s for the compositional approach.

5.4 Database with redundancy

The final case study is a database system consisting of disks arranged in clusters, disk controllers, and processors. As it was studied in [19], for redundancy 'R' the system is composed of two types of processors (with R copies of each type), two types of controllers (with R copies of each type), and six disk clusters (with $R + 2$ disks each). Units lifetime is exponentially distributed with failure rates μ_d, μ_c and μ_p for disks, controllers and processors resp. A unit can fail, with equal probability, in one of two modes, whose repair rates are 1 per hour and 0.5 per hour. The system is *operational* as long as *less than* R processors of each type, R controllers of each type, and R disks on each cluster, have failed. The property of interest is the steady state probability of a system breakdown, i.e. an instance of Property (2).

From the point of view of IS this has a *flat structure* which differs from all previous models: few discrete events produce a breakdown despite the high number of system components. This is aggravated by a low failure rate of disks w.r.t. processors and controllers: a breakdown is likely to be caused by broken proc./ctrl. of the same type, rather than by failures on the clusters. There is no trivially efficient way to compute the importance.

We propose five compositional importance functions which exercise the analysis from Sec. 4.3: AC_1 is a simple addition of the local IFUN of every module; AC_2 adds the product of (the local IFUNs of) all the disks in the database, the product of all the controllers, and the product of all the processors; AC_3 follows a similar approach but is finer-grained, since it multiplies (the local IFUNs of) all components in every cluster or processor/controller of the same type, adding these together; finally AC_4 and AC_5 consider the DNF formulation of the property. The first implements the `(max,+)` semi-ring, so every failure configuration is a sum (e.g. $\text{disk}_3^6 + \text{disk}_4^6$ for $R = 2$), and the resulting maximum is the global system importance. AC_5 implements `(+,*)`, so every failure configuration is a product and all these are summed together. Local importance was linear for $\{\mathsf{AC}_i\}_{i=1,4}$ and exponential for $\{\mathsf{AC}_i\}_{i=2,3,5}$, e.g. a state at distance d from the rare event had importance 2^d. Interestingly, AC_4 coincides with the *ad hoc* approach followed in [19]. The monolithic approach could not be tested due to the memory requirements.

Like in [19] simulations were run for increasing redundancy values (we used $R = 2, 3, 4, 5$). However, a randomized policy of the repairman cannot be expressed in the current IOSA syntax. For that reason the estimates produced are not comparable to those reported in the cited article. Furthermore the failure rates chosen were $(\mu_p, \mu_c, \mu_d) =$ (2e-2, 2e-2, 1e-2). The resulting long run probabilities estimated with this setting were in the order of 1.10e-2, 8.68e-5, 4.47e-7, and 1.13e-9 for the indicated redundancies. Unlike previous experiments simulations were run for fixed time periods (3 s, 1 m, 1 h and 6 h, for the respective redundancies). The goal was to build the narrowest possible interval for each time budget. Simulations were run on Jupiterace

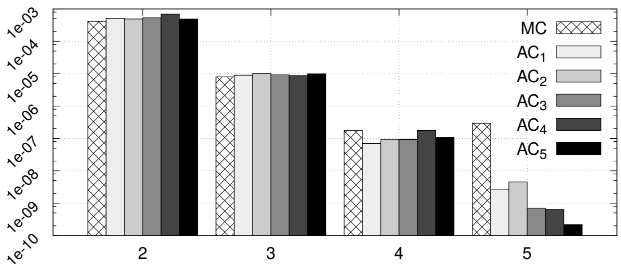

Figure 4: Estimation precisions on database

and the resulting precisions for a 90% confidence level are shown in Fig. 4, where the x axis groups the different simulations per redundancies and the y axis shows the achieved interval widths. The plot shows the results using split = 3; the outcomes for split $\in \{2, 5, 6\}$ were similar.

As expected the benefits of RESTART pay off only for the higher values of R: standard Monte Carlo (MC) competed against IS for redundancies 2 and 3. However for $R = 4$ the compositional IFUNs performed better, and for $R = 5$ the event is too rare for MC. Notice how in this last case the semi-ring composition variants performed best. Derivation times for all importance functions were below 2 s in all cases.

6. CONCLUSIONS

In this work we have presented a novel technique to automatically derive an importance function following a compositional construction. Though we focused on RESTART, such function could be used with any IS technique.

We studied the performance of our algorithms in case studies taken from the literature, matching or improving the performance of the best *ad hoc* alternatives to our knowledge. Overall and between the two automatic approaches proposed, the monolithic variant outperformed the compositional one to a minor extent. This was to be expected since it counts with global behaviour information, lost in the modular projections of the compositional variant.

In contrast both the computation time and memory storage requirements needed to compute the compositional importance function are orders of magnitude lower than the monolithic variant, which in particular could not be computed for the database case study. Besides, the monolithic method performs badly in graphs with high connectivity, since any rare state is close to the initial state and hence the IFUN would have a small image. Contrarily, in the compositional method, the composition of local IFUNs may yield a composed IFUN with a large image. This situation would be precisely the case in the database case study.

Experimentation shows RESTART is extremely sensitive to the split value chosen, even when measures are taken to mitigate the starvation (resp. overhead) of choosing a splitting too small (resp. big). A possible workaround would be to run the simulation tool parallelly for different splittings, stopping at the first successful estimation. When clusters like Mendieta are available this poses no major challenge.

Our research would also benefit from an extension of the IOSA syntax. The repairman model is a typical agent in rare event simulation, for which various repair policies exist. Referring to the state of the modules by means of arrays or similar constructs would help us implement such policies.

Acknowledgments

We thank José Villén-Altamirano for his help on the triple tandem queue. This work was supported by grants ANPCyT PICT-2012-1823, SeCyT-UNC 05/BP12 and 05/B497, and the ERC Advanced Grant 695614 (POWVER) while the second author visited Saarland University. Experiments were run on computers provided by CCAD and FAMAF-UNC.

7. REFERENCES

[1] C. E. Budde, P. R. D'Argenio, and H. Hermanns. Rare event simulation with fully automated importance splitting. In *EPEW 2015*, *LNCS* 9272, pages 275–290. Springer, 2015.

[2] F. Cérou, P. Del Moral, T. Furon, and A. Guyader. Sequential Monte Carlo for rare event estimation. *Statistics and Computing*, 22(3):795–808, 2012.

[3] F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

[4] P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, 1999.

[5] P. R. D'Argenio and J.-P. Katoen. A theory of stochastic systems part I: Stochastic automata. *Inf. Comput.*, 203(1):1–38, 2005.

[6] P. R. D'Argenio, M. D. Lee, and R. E. Monti. Input/Output Stochastic Automata - Compositionality and Determinism. In *FORMATS 2016*, *LNCS* 9884, pages 53–68. Springer, Springer, 2016.

[7] M. J. J. Garvels. *The splitting method in rare event simulation*. PhD thesis, Department of Computer Science, University of Twente, 2000.

[8] M. J. J. Garvels, J.-K. C. W. Van Ommeren, and D. P. Kroese. On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371, 2002.

[9] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600, 1999.

[10] C. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[11] C. Jegourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In *CAV 13*, *LNCS* 8044, pages 576–591. Springer, 2013.

[12] D. P. Kroese and V. F. Nicola. Efficient estimation of overflow probabilities in queues with breakdowns. *Performance Evaluation*, 36:471–484, 1999.

[13] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, *LNCS* 6806, pages 585–591. Springer, 2011.

[14] P. L'Ecuyer, F. Le Gland, P. Lezaud, and B. Tuffin. Splitting techniques. In *Rare Event Simulation using Monte Carlo Methods*, pages 39–61. John Wiley & Sons, Ltd, 2009.

[15] D. Reijsbergen, P.-T. de Boer, W. Scheinhardt, and B. Haverkort. Automated rare event simulation for stochastic Petri nets. In *QEST 2013*, *LNCS* 8054, pages 372–388. Springer, 2013.

[16] J. Villén-Altamirano. RESTART simulation of networks of queues with erlang service times. In *Winter Simulation Conference*, WSC '09, pages 1146–1154. Winter Simulation Conference, 2009.

[17] M. Villén-Altamirano, A. Martínez-Marrón, J. Gamo, and F. Fernández-Cuesta. Enhancement of the accelerated simulation method RESTART by considering multiple thresholds. In *Proc. 14th Int. Teletraffic Congress*, pages 797–810, 1994.

[18] M. Villén-Altamirano and J. Villén-Altamirano. RESTART: a method for accelerating rare event simulations. In *Queueing, Performance and Control in ATM (ITC-13)*, pages 71–76. Elsevier, 1991.

[19] M. Villén-Altamirano and J. Villén-Altamirano. The rare event simulation method RESTART: efficiency analysis and guidelines for its application. In *Network Performance Engineering*, *LNCS* 5233, pages 509–547. Springer, 2011.

[20] J. Villén-Altamirano. Asymptotic optimality of RESTART estimators in highly dependable systems. *Reliability Engineering & System Safety*, 130:115 – 124, 2014.

[21] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.