Using Probabilistic Analysis for the Certification of Machine Control Systems

Atif Mashkoor
Software Competence Center Hagenberg,
Hagenberg, Austria
Email: firstname.lastname@scch.at

Osman Hasan
School of Electrical Engineering
& Computer Science,
National University of Sciences
& Technology,
Islamabad, Pakistan
Email: firstname.lastname@seecs.edu.pk

Wolfgang Beer
Software Competence Center Hagenberg,
Hagenberg, Austria
Email: firstname.lastname@scch.at

***Abstract*—Traditional testing techniques often reach their limits when employed for the assessment of critical Machine Control Systems as they contain a large amount of random and unpredictable components. The probabilistic analysis approach can assist in their evaluation by providing a subjective evidence of their safety and reliability. The synergy of probabilistic analysis and expressiveness of higher-order logic theorem proving results into convincing modelling and reasoning of several stringent safety cases for the certification of software-controlled systems.**

I. Introduction

Software-controlled systems have become ubiquitous in our daily lives. The proliferation of software has spanned from transportation to industrial informatics and from power generation to health-care systems. While our routine activities increasingly depend on software-intensive systems, the safety-critical, distributed, heterogeneous, dynamic and often unpredictable nature of such systems produces several complex challenges.

The grand challenge of such systems is that their incorrect use or unsafe development may bring human life on stake. We often read such news when planes crash or trains collide. However, transportation-related calamities are only one facet of the overall gloomy picture of accidents caused by software faults. As they make highlights in the mainstream media, they are hot topics in active research. What we are oblivious to are the accidents related to less talked about yet critical domains, such as Machine Control Systems (MCS). Today, a generation of high-assurance MCS can be found in areas as diverse as advanced manufacturing, medical technologies, energy production, natural resources management, etc.

MCS are devices that manage, command, direct or regulate the behaviour of industrial processing units. As shown by Figure 1, a typical MCS is composed of some realtime Programmable Logic Controllers (PLCs) and a non-realtime User Interface (UI) that helps interacting with PLCs. PLCs, in turn, are of two types. First type contains some domain-specific commands pertaining to the processing unit that control motors based on signals from sensors and actuators. The other type is comprised of stringent safety regulations that need to be certified before being functional.

The malfunctioning of MCS can result into a minor injury, irreversible severe injury, amputation or even death. Although, we do not have detailed publicly available investigation reports of accidents caused by disturbances in MCS performing safety functions, yet some information is presented in [1]. The report is based on analysis of 700 industrial incidents occurred between 1996-2002 in Poland alone. According to the report, 41% of severe accidents were caused by improper functioning of the system. Software faults were among the notable reasons.

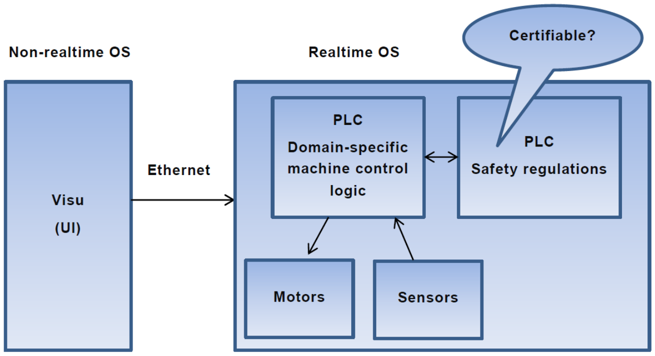

Fig. 1. A typical MCS

Industrial incidents are not new phenomena. Standard techniques to avoid accidents [2] are in place since last several decades. However, with the proliferation of software, industry saw a paradigm shift when mechanical controls were replaced by their software counterparts and along new technology came new challenges. As software started becoming pervasive, it also became unprecedentedly complex.

Due to the critical nature of software components of MCS, there are numerous properties in which they must exceed other software systems, such as resilience, safety, security, and reliability. Such systems must also be certified. Certification, as defined by the charter of Software Certification Consortium (SCC)[1], demonstrates the assurance that the system has met its relevant technical standards and specifications, and it can be trusted to provide its services safely, securely and effectively.

There are several approaches towards software certification, e.g., argument-based, process-based and product-based. However, in the past few years, argument-based safety case

[1] http://cps-vo.org/group/scc

approach [3] is gaining popularity among regularity regimes for the certification of software-intensive systems. It is also the explicit part of U.K. MoD Defence Standard 00-56. A safety case, in fact, is an evidence based on a convincing and valid argument that the designed system is adequately safe for a given application in the given environment.

Formal methods are usually the part of most of the software certification techniques. Their use is often recommended for the development of safety-critical systems involving higher Safety Integrity Level (SIL), IEC 61508, and DO-178C standards. Formal methods help specifying system requirements and properties using mathematical and logical notations which are then amenable to correctness. Correctness amounts to both verification, the practice where it is determined that product is being built correctly and meets its specification, and validation, the practice where it is ensured that the product meets customers requirements.

Standard verification and validation techniques are comprised of activities like standard theorem proving, model checking, animation and simulation. Each of these techniques has some inherent limitations and do not cope very well with the posed challenges in the development of highly complex and safety-critical MCS.

A standard method to guarantee the safety properties in a formal specification is to show that the invariant, a logical representation of safety, is preserved. We prove a theorem to satisfy the property. While adequate in most of the cases, less so to justify the absence of component failures and emergence of potential hazardous situations. In contrast to safety and deadlock-freedom, fairness and liveness proofs are not so straightforward. The techniques like animation, simulation, model checking can provide help in such cases but only up to some extent.

The problem with the animation is that currently available animators have strong limitations on the types of specifications they can animate. We have seen how the class of executable specifications can be extended with the help of few safe transformations [4]. Yet, there are some formal texts on which animators fail [5]. For those texts, we propose to use simulation rather than animation; the specification is translated into a program. But the problem with the translation is that currently available translators are in infancy stages and have their own weaknesses, such as expressiveness of the targeted languages. Approximation using closest counterparts does not fulfil the true purpose.

Model checking involves the construction of a precise state-based mathematical model of the given system. The state-space model is then subjected to exhaustive analysis to automatically verify that it satisfies a set of formally represented properties. The problem of this approach is that it can only be used to analyse systems that can be expressed as finite state machines. Another major limitation is the state-space explosion. The state-space of a real-world MCS can be very large, or sometimes even infinite and thus impossible to be explored completely within limited resources of time and memory.

Higher-order theorem proving, on the other hand, is very flexible in terms of tackling a variety of systems, thanks to its expressiveness. However, this flexibility comes at the cost of enormous user efforts in terms of manual development of system models and interactive verification due to the undecidable nature of the underlying logic. Thus, developing realistic models of MCS, which usually involve significant amount of continuous and non-deterministic elements, and verifying them for all possible scenarios could be very tedious, if not impossible.

Probabilistic analysis overcomes the above mentioned limitations. Rather than ensuring absolute correctness, the main idea here is to ensure that the system behaves in the desired manner with an acceptable probability. This not only simplifies system modelling but also eases the verification task considerably. Probabilistic analysis is often used as an evidence to justify the claim of safety and reliability [3].

The main aim of this paper is to demonstrate how probabilistic analysis approach can provide a subjective evidence that can support the claim of system safety and reliability. Rest of the paper is organized as following: An overview about the existing formal probabilistic analysis approaches is given in Section II. Section III introduces the main concepts used in this paper, i.e., safety cases and higher-order logic theorem proving. Section IV describes how probabilistic algorithms can be formalized. Section V presents the proposed formal probabilistic analysis approach. Section VI applies the proposed approach to a simple case-study of multi-robot systems. The paper is finally concluded in Section VII.

II. Related Work

In recent years, the use of argument-based safety case approach has emerged as a popular technique for the certification of software-controlled critical systems. In [6], authors developed a conceptual model, based on the IEC 61508 standard, to characterize the chain of safety evidence that underlies safety arguments about software. In [7] and [8], authors present approaches that extend the border of safety case analysis towards general software properties. These approaches, though rigorous, are not fully formal.

In [9], an approach is presented to show how a safety case analysis based on SMT sovlers and infinite bounded model checkers can be used as the certification argument for the adaptive systems. The work is then extended in [10] with the proposition of a framework based on probabilistic analysis. The mathematics presented in [10] is then formalized by probabilistic kernels-based mathematical approach in [11]. The approach is illustrated using an example of a conflict detection system for an aircraft. Our proposed approach, as compared to these aforementioned works, is much more powerful in terms of handling a larger set of problems, for instance MCS. Moreover, unlike our approach, the mathematical framework of [11] has yet to be formalized in a theorem prover.

Probabilistic model checking [12], [13] is the most widely used formal method for probabilistic analysis of systems that can be modeled as Markov Chains. Like traditional model checking, it involves the construction of a precise state-based mathematical model of the given probabilistic system, which is then subjected to exhaustive analysis to verify if it satisfies a set of formally represented probabilistic properties. Some notable probabilistic model checkers include PRISM, Ymer and VESTA. Besides the accuracy of the results, the most promising feature of probabilistic model checking is the ability to perform the analysis automatically. On the other hand, it is limited to systems that can only be expressed as Markov chains. Another major limitation of the probabilistic model checking approach is state-space explosion. The state-space of a probabilistic system can be very large, or sometimes even infinite. Thus, at the outset, it is impossible to explore the entire state-space with limited resources of time and memory. Thus, the probabilistic model checking approach, even though is capable of providing exact solutions, is quite limited in terms of handling a variety of probabilistic analysis problems. Similarly, some algorithms implemented in these model checking tools are based on numerical methods. For example, a well-known iterative method, i.e., the Power method, is applied to compute the steady-state probabilities (or limiting probabilities) of the Markov chain in PRISM [14]. For this reason, most of the stationary properties analyzed in model checkers are time bounded. Moreover, probabilistic model checking tools often utilize unverified algorithms and optimization techniques. Finally, probabilistic model checking cannot be used to verify generic mathematical expressions corresponding to probabilistic and statistical properties. Thus, the verified properties involve values that are expressed in a computer based notation, such as fixed or floating point numbers, which also introduces some degree of approximation in the results. The B [15] and Event-B [16] methods have also been extended to support probabilistic analysis. The main idea here is either to use a probabilistic choice operator to reason about termination conditions [17] or to use the semantics of a Markov process to reason about the reliability or performance characteristics of the given system [18]. But these extensions of the B-method cannot be used to reason about generic mathematical expressions for probabilistic or statistical properties. Similarly, such formalisms are not mature enough to model and reason about all different kinds of continuous probability distributions. Due to the continuous nature of the MCS, both the probabilistic model checking and Event-B based techniques cannot be used to capture their true behaviour and thus, to the best of our knowledge, the use of probabilistic and quantitative assessment for the formal verification of MCS is almost non-existent.

The above mentioned limitations can be overcome by using higher-order-logic theorem proving for conducting the formal probabilistic analysis of MCS. We can capture the true continuous and randomized behavior of the given MCS in higher-order logic by leveraging upon its expressiveness. Moreover, generic mathematical expressions corresponding to the probabilistic and statistical properties of interest can be formally verified within the sound core of a theorem prover. Due to the formal nature of the models and properties and the inherent soundness of the theorem proving approach, probabilistic analysis carried out in this way will be free from any approximation and precision issues. The foremost criteria for conducting the formal probabilistic analysis of MCS in a theorem prover is the availability of a formalized probability theory, which will in turn allow us to express probabilistic notions in higher-order logic, and formal reasoning support for the probability distribution and statistical properties of random variables in a theorem prover. Various higher-order-logic formalizations of probability theory can be found in the literature, e.g., [19], [20], [21]. The formalizations by Mhamdi [20] and Hölzl [21] are based on extended real numbers (including $\pm\infty$) and also include the formalization of Lebesgue integral for reasoning about statistical properties. This way, they are more mature than Hurd's [19] formalization of measure and probability theories, which is based on simple real numbers and offers a limited support for reasoning about statistical properties [22]. However, the former formalizations do not support a particular probability space like the one presented in Hurd's work. Due to this distinguishing feature, Hurd's formalization [19] has been utilized to verify sampling algorithms of a number of commonly used discrete and continuous random variables based on their probabilistic and statistical properties [22]. Since formalized random variables variables and their formally verified properties play a vital role in the probabilistic analysis of MCS systems, as will be illustrated later in this paper, we propose to utilize Hurd's formalization of measure and probability theories for the probabilistic analysis of MCS. Since Hurd's theories are available in the HOL4 theorem prover so we have also used HOL4 for the current work.

III. Background

A brief introduction to the safety cases, high-order-logic theorem proving and the HOL4 theorem prover is provided in this section to facilitate the understanding of the rest of the paper.

A. Safety case

A safety case, as described by [3], is an evidence that provides a convincing and valid argument about the safety of a system in a given domain. The main elements of a safety case are the claim about a property, its evidence, and its argument.

A claim is usually about a system meeting certain properties, such as reliability, availability, safety and accuracy. An evidence provides the basis of a safety argument. This can be either facts, assumptions, or even sub-claims. Arguments link an evidence to claims. An argument can either be deterministic, probabilistic, or qualitative. The choice of the argument will depend upon the available evidence and the type of claim. System reliability claims are often justified in terms of probability of failure of its components. Arguments can be based on the design of a system, its development process, or simulated and prior field experiences.

B. HOL4 Theorem Prover

HOL4 is an interactive theorem prover that allows conducting proofs in higher-order logic, which is implemented by using the simple type theory of Church [23] along with Hindley-Milner polymorphism [24]. The logic in HOL4 is represented by Meta Language (ML), which is a strongly-typed functional programming language. The core of HOL4 consists of only 5 axioms and 8 inference rules. Building upon these foundations, HOL4 has been successfully used to formalize many classical mathematical theories and verify a wide variety of software and hardware systems.

HOL4 allows four kinds of terms, i.e., constants, variables, function applications, and lambda-expressions. It also supports polymorphism, i.e., types containing type variables, which is a distinguishing feature of higher-order logic. Semantically, types are denoted by sets and terms are denoted by memberships of these sets. Formulas, sequents, axioms, and theorems are represented by terms of Boolean types.

The formal definitions and theorems can be stored as a HOL4 theory file in computers. These theories can then be re-sued by loading them in a HOL4 session. This kind of a reusability feature is very helpful in reducing the human interaction as we do not have to go through the tedious process of regenerating already available proofs using the basic axioms and primitive inference rules. Various classical mathematical results have been formalized and saved as HOL4 theories. The HOL4 theories that are of particular interest to the current work include the theory of Booleans, lists, positive integers and *real* analysis, measure and probability theory. One of the primary motivations of selecting the HOL4 theorem prover for our work was to benefit from these existing mathematical theories.

The formal proofs in HOL4 can be conducted in both forward and backward manner. In the forward proof method, the previously proven theorems are used along with the inference rules to verify the desired theorem. The forward proof method usually requires the exact details of a proof in advance and is thus not very straightforward. In the backward proof method, ML functions called *tactics* are used to break the main proof goals into simple sub-goals. Some of these intermediate subgoals can be discharged by matching axioms or assumptions or by applying built-in automatic decision procedures. This process is repeated until all the sub goals are discharged, which concludes the proof for the given theorem.

IV. Formalization of probabilistic algorithms

Building upon the recently developed measure-theoretic formalization of probability theory [19], we can conduct formal probabilistic analysis [25] in the sound core of a higher-order logic theorem prover. The system can be modelled as a probabilistic algorithm where its randomized components can be specified using appropriate random variables and its probabilistic and statistical properties can be verified using the proven probability axioms. The ability of higher-order logic to formalize continuous and randomized systems, and to verify all sorts of probabilistic and statistical properties makes it the meritorious approach for the formal probabilistic analysis of MCS.

The unpredictable components of a system can be modelled as higher-order logic functions that make random choices by using the required number of bits from an infinite sequence of random bits $\mathbb{B}^\infty$. These functions return the result along with the remaining portion of the infinite Boolean sequence to be used by other programs. Thus, the data type of a probabilistic algorithm that takes a parameter of type α and returns a value of type β is:

$$\mathcal{F} : \alpha \rightarrow \mathbb{B}^\infty \rightarrow \beta \times \mathbb{B}^\infty$$

For illustration, consider the example of a Bernoulli($\frac{1}{2}$) random variable that returns 1 or 0 with equal probability $\frac{1}{2}$. It can be formally modelled as follows:

```
⊢ bit s = (if shd s then 1 else 0, stl s)
```

where variable *s* represents the infinite Boolean sequence and functions `shd` and `stl` return the *head* and *tail* of their sequence argument, respectively. Probabilistic algorithms can be expressed in a more compact way without explicitly mentioning the Boolean sequence that is passed around by using more general state-transforming monads where the states are the infinite Boolean sequences.

```
⊢ ∀ a s. unit a s = (a,s)
⊢ ∀ f g s. bind f g s = g (fst (f s))
        (snd (f s))
```

The `unit` operator is used to lift values to the monad, and the `bind` is the monadic analogue of function application. The function *bit* can be defined using the monadic notation as:

```
⊢ bit_monad = bind sdest
        (λb.if b then unit 1 else unit 0)
```

where `sdest` provides the head and tail of a sequence as a pair (*shd* s, *stl* s).

Hurd [19] constructed a probability space on infinite Boolean sequence and formalized a measure theoretic formalization of probability theory in the higher-order-logic theorem prover HOL. Building upon these foundations, we can verify all the basic laws of probability as well as probabilistic properties of any algorithm that is specified using the infinite Boolean sequence. For example, we can verify the Probability Mass Function (PMF) of the function `bit` as:

$\vdash \mathbb{P}\ \{\texttt{s | fst (bit s) = 1}\} = \frac{1}{2}$

where the HOL function `fst` selects the first component of a pair and $\mathbb{P}$ is the probability function that maps sets of infinite Boolean sequences to real numbers between 0 and 1 .

Just like the `bit` function, we can formalize and verify any random variable using the above mentioned approach. For example, the Bernoulli and Uniform random variables have been verified using the following PMF relations [19]:

Lemma 1: *PMF of Bernoulli(p) Random Variable*

```
⊢ ∀p. 0 ≤ p ∧ p ≤ 1 ⇒
    ℙ {s | fst (bern_rv p s)=1} = p
```

Lemma 2: *PMF of Uniform(m) Random Variable*

$\vdash \forall\texttt{m x. x} < \texttt{m} \Rightarrow$
$\quad\mathbb{P}\ \{\texttt{s | fst (unif\_rv m s) = x}\} = \frac{1}{\texttt{m}}$

The function `ber_rv` models an experiment with two outcomes; 1 and 0, whereas p represents the probability of obtaining a 1. Similarly, the function `unif_rv` assigns equal probability to each element in the set $\{0, 1, \cdots, (m-1)\}$ and thus ranges over a finite number of positive integers. Such pre-formalized random variables and their proven properties greatly facilitate the analysis of MCS as, in most cases, the unpredictable nature of its components can be expressed in terms of well-known random variables.

Expectation theory also plays a vital role in probabilistic analysis as it is lot easier to judge performance issues based on the average of a characteristic, which is a single number, rather than its distribution function. The expectation of a discrete random variable can be defined as a higher-order logic function as follows [26]:

Definition 1: *Expectation of Discrete Random Variables*

```
⊢ ∀R. expec R =
    suminf (λn.n * ℙ {s | fst (R s) = n})
```

where suminf represents the HOL formalization of the infinite summation of a real sequence f, i.e., $\lim_{k\to\infty} \sum_{n=0}^{k} f(n)$. The function expec can be used to verify the expectation of any discrete random variable that attains values in positive integers. For example, the higher-order logic theorem corresponding to the expectation of the Bernoulli random variable has been formally verified in [26] as follows:

Lemma 3: *Expectation of Bernoulli(p) Random Variable*

```
⊢ ∀p. 0 ≤ p ∧ p ≤ 1 ⇒
    expec (λs.bern_rv p s) = p
```

where $(\lambda x.t)$ represents a lambda abstraction function in HOL that maps its argument x to $t(x)$.

The continuous random variables can also be formalized by building upon the above mentioned measure-theoretic formalization of probability theory. The main idea behind their formalization is to transform a Standard Uniform random variable to other continuous random variables based on the principles of the non-uniform random number generation [22]. The Standard Uniform random variable can be modeled by using the formalization approach for discrete random variables and the formalization of the mathematical concept of limit of a *real* sequence [27]:

$$\lim_{n\to\infty} (\lambda n. \sum_{k=0}^{n-1} (\frac{1}{2})^{k+1} X_k) \tag{1}$$

where X_k denotes the outcome of the k^{th} random bit; $True$ or $False$ represented as 1 or 0, respectively. This formalization [22] is used along with the formalization of the CDF function to formally verify the correctness of the Inverse Transform Method (ITM), i.e., a well known non-uniform random generation technique for generating non-uniform random variables for continuous probability distributions for which the inverse of the CDF can be represented in a closed mathematical form. Formally, it can be verified for a random variable X with CDF F using the Standard Uniform random variable U as follows [22].

$$Pr(F^{-1}(U) \leq x) = F(x) \tag{2}$$

The formalized Standard Uniform random variable can now be used to formally specify any continuous random variable for which the inverse of the CDF can be expressed in a closed mathematical form as $X = F^{-1}(U)$. Whereas, its CDF can be verified based on simple arithmetic reasoning, using the formally verified ITM, given in Equation (2). This approach has been successfully utilized to formalize and verify Exponential, Uniform, Rayleigh and Triangular random variables [22].

The expectation for a continuous random variable X is defined on a probability space (Ω, Σ, P) [22], is as follows:

$$E[X] = \int_{\Omega} X dP \tag{3}$$

where the integral represents the Lebesgue integral. In order to facilitate the formal reasoning about statistical properties of continuous random variables, the following two alternate expressions for the expectation have been verified [22]. The first expression is for the case when the given continuous random variable X is bounded in the positive interval $[a, b]$

$$\begin{aligned} E[X] = \lim_{n\to\infty} \sum_{i=0}^{2^n-1} (a + \frac{i}{2^n}(b-a)) \\ \mathbb{P}\left\{a + \frac{i}{2^n}(b-a) \leq X < a + \frac{i+1}{2^n}(b-a)\right\} \end{aligned} \tag{4}$$

and the second one is for an unbounded positive random variable.

$$E[X] = \lim_{n\to\infty} \sum_{i=0}^{n2^n-1} \frac{i}{2^n} \mathbb{P}\left\{\frac{i}{2^n} \leq X < \frac{i+1}{2^n}\right\} + n\mathbb{P}(X \geq n) \tag{5}$$

Both of the above expressions do not involve any concepts from Lebesgue integration theory and are based on the well-known arithmetic operations like summation, limit of a real sequence, etc. Thus, users can simply utilize them, instead of Equation (3), to reason about the expectation properties of their random variables and gain the benefits of the original Lebesgue based definition. The formal verification details for these expressions are given in [22]. These expressions are further utilized to verify the expected values of Uniform, Triangular and Exponential random variables [22].

The above mentioned formalization plays a pivotal role for the formal probabilistic analysis of MCS. In the next section, we illustrate the usefulness and practical effectiveness of the foundational formalization, presented in this section, by providing a step-wise approach for analyzing a MCS using HOL4.

V. THE TRIPTYCH ANALYSIS APPROACH

Figure 2 depicts the proposed triptych formal probabilistic analysis approach. It primarily builds upon the existing formalization of probability theory (including the infinite Boolean sequence space), and commonly used random variables and their proved properties. The approach is as follows:

1) The first step is to formalize the true behaviour of the given MCS including its unpredictable and continuous components in higher-order logic. The randomized behaviours would be captured using appropriate discrete [19] and continuous random variables [22] and the continuous aspects can be modeled using the formalized real numbers [27].

2) The second step is to utilize the formally specified model, i.e., the higher-order-logic function that represents the behavior of the given MCS, to formally express desired system properties as higher-order logic proof goals. Due to the random nature of the model, the properties are also either probabilistic or statistical. For this purpose, the existing higher-order-logic functions of probability [19], expectation and variance [22] can be utilized.
3) The third and final step is to formally verify the developed higher-order logic proof goals using a theorem prover. The existing theorems corresponding to probability distribution functions, expectation and variance of commonly used random variables [19], [22] greatly facilitate in minimizing the required user interaction to prove these goals.

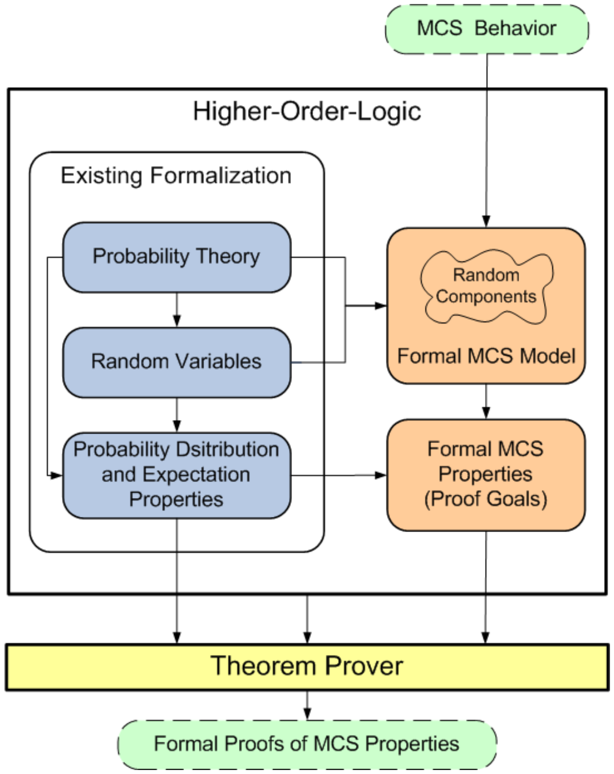

Fig. 2. The triptych probabilistic analysis of MCS

We now illustrate the utilization and practical effectiveness of this approach by providing a simple case study.

VI. Collision detection in multi-robot systems: a case study

The use of autonomous multi-robot systems is on rise in the development of MCS. They offer numerous advantages over single-robot systems, e.g., more spacial coverage, redundancy, reconfigurability and throughput. However, the design of these robots involves many challenges due to their potentially unknown and dynamic environments. Moreover, due to their autonomous nature, avoiding collisions with other robots of the system is also a major challenge [28]. The number of robots and the arena size is chosen at the design time such that potential collisions are minimized. However, estimating these parameters is not a straightforward task due to the unpredictable nature of the potential collisions. Probabilistic techniques can help in this realm.

We are interested in finding the average number of collisions for the given system with k mobile robots moving in an arena of n distinct locations. A collision happens when two or more robots are on the same location at the same time. The randomized behavior of the collision can be modelled as a Bernoulli random variable X with two outcomes, 1 or 0, as follows:

$$X_{ij} = \begin{cases} 1 & \text{if robot } i \text{ and } j \text{ are using the same location;} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

corresponding to each pair (i, j) of the k robots in the group such that $0 \leq i \leq j < k$. We can reasonably assume that the location of any robot at any particular time instant is uniformly and independently distributed among the n available slots. Under these conditions, the total number of collisions can be computed as follows:

$$\sum_{i=0}^{k-1} \sum_{j=i+1}^{k-1} X_{ij} \tag{7}$$

The formal probabilistic analysis of the above mentioned collision detection problem can now be done by following the triptych approach outlined in Figure 2. The first step is to formalize the system in higher-order logic. The behaviour of the given system can be expressed in terms of the Bernoulli and Uniform random variables using the following recursive functions.

Definition 2: *Collision Detection in Multi-Robot System*

```
⊢ (∀ n. col_detect_helper 0 n = unit 0) ∧
  (∀ k n. col_detect_helper(k+1) n =
    bind (col_detect_helper k n)
   (λa. bind (bern_rv
   (ℙ{s | fst(unif_rv n s) =
     fst(unif_rv n (snd (unif_rv n s)))}))
       (λb. unit (b + a))))
⊢ (∀ n. col_detect 0 n = unit 0) ∧
  (∀ k n. col_detect (k + 1) n =
    bind (col_detect k n)
      (λa. bind
    (col_detect_helper k n)
      (λb. unit (b + a))))
```

The functions `col_detect_helper` and `col_detect` model the inner and outer summations of Equation (7), respectively. The variable `bern_rv` models the randomness given in Equation (6), with the probability of success equal to the probability of the event when two independent Uniform(n) random variables generate the same values. The two Uniform random variables in the above definition correspond to the locations of two robots in the system based on the above mentioned assumptions. The independence between the two Uniform(n) random variables is ensured because of the fact that the second uniform random variable on the right-hand-side of the equality utilizes the remaining portion of the infinite Boolean sequence from the first Uniform(n) random variable that is on the left-hand-side of the equality. Thus, the function `col_detect` accepts two parameters k and n, which represents the number of robots in the system and the number of available locations in the areana, respectively, and it returns the total number of pairs of robots having the same location at a given time, which is the number of collisions.

The second step is to formalize the property of interest in higher-order logic. We are interested in the average number of collisions and that can be expressed using the expectation function, given in Definition 1, as follows:

Theorem 1: *Average Number of Collisions*
$\vdash \forall \texttt{k n. } 0 < \texttt{n} \wedge 2 \leq \texttt{k} \Rightarrow$
$\quad \texttt{expec (col\_detect k n)} = \frac{\texttt{k(k-1)}}{\texttt{2n}}$

The assumptions in the above theorem ensure that the number of locations are more than 0 and the population is at least 2 in order to have 1 pair at minimum.

The third step is to verify the proof goal in a theorem prover. We verify it using the HOL theorem prover. The interactive reasoning process was primarily based on performing induction on the variable k, Lemmas 1, 2 and 3, and some formally verified probability theory laws like the linearity of expectation. The manually developed proof script for this verification relied heavily on already existing formalizations of probability theory [19] and random variables [22]. Theorem 1 provides very useful insights into the collision detection in a multi-robot system. It can be clearly observed that the expected number of collisions would be less than 1 if $k(k-1) \leq 2n$. This means that if we have $\sqrt{2n}+1$ or more robots in the system, then on average we can expect at least one collision.

The above example clearly demonstrates the effectiveness of the proposed approach in analyzing MCS. Due to the formal nature of the model and the inherent soundness of higher-order logic theorem proving, we have been able to verify the desired property with full precision. Same results could not be obtained using simulation or animation techniques, neither the problem could be described as a Markov chain and thus could not be analysed using a probabilistic model checker. Another distinguishing feature of Theorem 1 is its generic nature and the universal quantification on all the variables. Thus, the theorem can be instantiated with any possible values to obtain the average number of collisions for any specific system. Finally, the theorem explicitly provides all the assumptions under which it has been verified. This is usually not the case when we are verifying theorems by hand using the traditional paper-and-pencil proof methods as the mathematicians may forget to pen down all the required assumptions that are required for the validity of their analysis. These missing assumptions may lead to erroneous designs as well and thus the proposed approach overcomes this problem.

These additional benefits have been gained at the cost of the time and effort spent, while formalizing the system and formally reasoning about its properties, by the user. But, the fact that we were building on top of already verified results in the theorem prover helped significantly in this regard as the analysis, described in this section, only consumed approximately 500 lines of HOL code and 30 man-hours by an expert HOL user.

VII. Conclusion

We have presented a framework that enables us to unify the formal verification technique with quantitative reasoning for the engineering of safe and reliable MCS. The point is that in addition to analyse the absolute correctness, which at times is not possible, a probabilistic analysis can provide a better insight about the model. In this fashion, it is easier for stakeholders to obtain a probability of occurrence of a hazard in terms of the likelihood of components failures. We can now provide an evidence, as a certification argument, that the probability of violation of a safety requirement is sufficiently and acceptably small. Moreover, given the soundness of higher-order-logic theorem proving, the analysis results are completely reliable. Similarly, the high expressiveness of higher-order logic enables us to analyze a wide range of MCS. Thus, the proposed approach can be of great benefit for the analysis of MCS used in safety-critical applications, such as medicine and transportation.

Acknowledgment

The authors would like to thank...

References

[1] M. Dźwiarek, “An analysis of accidents caused by improper functioning of machine control systems,” *International Journal of Occupational Safety and Ergonomics*, vol. 10, no. 2, pp. 129–136, 2004.

[2] H. Heinrich, *Industrial Accident Prevention: A Scientific Approach.* McGraw Hill Inc., 1941.

[3] P. Bishop and R. Bloomfield, “A methodology for safety case development,” in *SAFETY-CRITICAL SYSTEMS SYMPOSIUM, BIRMINGHAM, UK, FEB 1998.* Springer-Verlag, ISBN 3-540-76189-6, 1998.

[4] A. Mashkoor and J.-P. Jacquot, “Stepwise validation of formal specificaitons,” in *18th Asia-Pacific Software Engineering Conference (APSEC'11)*, Ho Chi Minh City, Vietnam, 2011.

[5] F. Yang and J.-P. Jacquot, “Scaling up with event-b: a case study,” in *Proceedings of the Third international conference on NASA Formal methods*, ser. NFM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 438–452.

[6] R. Panesar-Walawege, M. Sabetzadeh, L. Briand, and T. Coq, “Characterizing the chain of evidence for software safety cases: A conceptual model based on the iec 61508 standard,” in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, april 2010, pp. 335 –344.

[7] E. Fong, M. Kass, T. Rhodes, and F. Boland, “Structured assurance case methodology for assessing software trustworthiness,” in *Secure Software Integration and Reliability Improvement Companion (SSIRI-C), 2010 Fourth International Conference on*, june 2010, pp. 32 –33.

[8] P. J. Graydon, J. C. Knight, and E. A. Strunk, “Assurance based development of critical systems,” in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 347–357.

[9] J. Rushby, “A safety-case approach for certifying adaptive systems,” in *In AIAA Infotech@Aerospace Conference, American Institute of Aeronautics and Astronautics, John Rushby*, 2009.

[10] J. Rushby, “Formalism in safety cases,' in *Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium*, C. Dale and T. Anderson, Eds. Bristol, UK: Springer, Feb. 2010, pp. 3–17.

[11] H. Herencia-Zapana, G. Hagen, and A. Narkawicz, “Formalizing probabilistic safety claims,” in *Proceedings of the Third international conference on NASA Formal methods*, ser. NFM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 162–176.

[12] C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen, “Model Checking Al-gorithms for Continuous time Markov Chains”,. I*EEE Transactions on Software Engineering*, vol. 29, no. 4, pp 524–541, 2003.

[13] J. Rutten, M. Kwaiatkowska, G. Normal, and D. Parker. “Mathematical Tech-niques for Analyzing Concurrent and Probabilisitc Systems”, *CRM Monograph Series*. vol. 23, American Mathematical Society, 2004.

[14] M. Kwiatkowska, G. Norman, and D. Parker, “Prism: Probabilistic symbolic model checker,” *Computer Performance Evaluation Modelling Techniques and Tools*, vol. 2324, no. 4, pp. 200–204, 2002.

[15] J.-R. Abrial, *The B Book*. 1996.

[16] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[17] S. Hallerstede and T. S. Hoang, “Qualitative probabilistic modelling in event-b,” in *Proceedings of the 6th international conference on Integrated formal methods*, ser. IFM’07, Springer-Verlag, 2007, pp. 293–312.

[18] A. Tarasyuk, E. Troubitsyna, and L. Laibinis, “Towards probabilistic modelling in event-b,” in *Proceedings of the 8th international conference on Integrated formal methods*, ser. IFM’10. Springer-Verlag, 2010, pp. 275–289.

[19] J. Hurd, “Formal verification of probabilistic algorithms,” PhD Thesis, University of Cambridge, Cambridge, UK, 2002.

[20] T. Mhamdi, O. Hasan and S. Tahar, “Formalization of Entropy Measures in HOL”. in *Proceedings of the 3rd International Conference on Interactive Theorem Proving*, ser. ITP’11, Springer-Verlag, 2011, pp. 233–248.

[21] J. Holzl and A. Heller, “Three Chapters of Measure Theory in Isabelle/HOL”. in *Proceedings of the 3rd International Conference on Interactive Theorem Proving*, ser. ITP’11, Springer-Verlag, 2011, pp. 135–151.

[22] O. Hasan and S. Tahar, “Formal Probabilistic Analysis: A Higher-order Logic based approach”. in: *Proceedings of the 2nd International Conference on Abstract State Machine (ASM), Alloy, B, and Z*, ser. ABZ’10. Springer-Verlag, 2010, pp. 2–19.

[23] A. Church, “A Formulation of the Simple Theory of Types.”, *Journal of Symbolic Logic*, vol. 5, pp. 56–68, 1940.

[24] R. Milner, “A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, vol. 17, pp. 348–375, 1977.

[25] A. Mashkoor and O. Hasan, “Formal probabilistic analysis of cyber-physical transportation systems,” in *12th International Conference on Computational Science and Its Applications (ICCSA’12)*, Salvador de Bahia, Brazil, 2012.

[26] O. Hasan and S. Tahar, “Using theorem proving to verify expectation and variance for discrete random variables,” *J. Autom. Reasoning*, vol. 41, no. 3-4, pp. 295–323, 2008.

[27] J. Harrison. “Theorem Proving with the Real Numbers“, Springer-Verlag, 1998.

[28] L. E. Parker, “Multiple mobile robot teams, path planning and motion coordination in,” in *Encyclopedia of Complexity and Systems Science* Springer, 2009, pp. 5783–5800.