Statistical Runtime Checking of Probabilistic Properties

Usa Sammapun Insup Lee Oleg Sokolsky
University of Pennsylvania

Abstract

Probabilistic correctness is another important aspect of reliable systems. A soft real-time system, for instance, exhibits probabilistic behaviors from tolerating some degrees of deadline misses. Since probabilistic systems may behave differently from their probabilistic models depending on their current environments, checking the systems at runtime can provide another level of assurance for their probabilistic correctness.

Runtime verification is a technique for checking correctness of a system at runtime by observing a system execution and checking it against its property specification. To check probabilistic properties, runtime verification can adopt a statistical technique used in model checking to check probabilistic properties. The statistical technique simulates, samples many execution paths, and estimates probabilities by counting successful samples against all samples. One particular difficulty in using this technique in runtime verification, however, is that runtime verification has only one execution path and cannot simply collect many different execution paths as in statistical probabilistic model checking. Therefore, this one execution path, usually in a form of a trace of states or events, needs to be broken down into different individual samples, which can be done only if a probabilistic system being observed has repeated or periodic behaviors such as soft real-time schedulers or network protocols.

To break apart one execution, runtime verification must be able to distinguish one repeated behavior from another by just looking at the system trace. The trace may also contain other states or events unrelated to probabilistic properties. Nonetheless, breaking apart one execution can be done via conditional probabilities. Written in terms of probabilistic properties, one can specify as given a condition A, does the probability that an outcome B occurs fall within a given range? For example, given that a real-time task has started, is the probability that it finishes within its deadline greater than 0.8? If we can relate the condition A to the outcome B within the trace, a set of A and B can be collected as one individual sample, and a sequence of the set can be collected as many different individual samples. The probability observed from the system can be estimated by counting the condition A with the outcome B against all A. After the probabilities are estimated, runtime verification uses statistical analysis such as hypothesis testing to provide a systematic procedure with an adequate level of confidence to determine statistically whether a system satisfies a probabilistic property.

The above statistical technique for checking probabilistic properties is applied to a runtime verification framework called MaC or Monitoring and Checking. MaC provides expressive specification languages based on Linear Temporal Logics to specify system properties. Once the properties are specified, MaC observes the system by retrieving system information from

probes instrumented into the system prior to the execution. MaC then checks the execution against the system properties and reports any violations. The main aspect of MaC is its formal property specification. MaC specification is built upon two elements: events and conditions. Events occur instantaneously during execution, whereas conditions represent system states that hold for a duration of time and can be $true$ or $false$. For example, an event denoting a call to a method $init$ occurs at the instant the control is passed to the method, and a condition $v < 5$ holds as long as the value v is less than *5*. Events and conditions can be composed using boolean operators such as negation, conjunction, disjunction, and other temporal operators.

Probabilistic properties can be specified by quantifying these events with probabilities. Using conditional probabilities to break one execution path into several samples, the probabilistic events are defined as follows. Given that an event a occurs, does the probability that an event b occurs fall within a given range? The probabilities are estimated by counting the number of an event b that occurs in response to an event a against the total number of an event a. The hypothesis testing is then used to decide whether the estimated probability falls within the given range of a probability quantified in a probabilistic event. A common statistics technique of z-score is used to tell statistically how far apart the estimated probability from the quantified probability. A threshold is set up with some levels of confidence to test whether the estimated probability falls with the quantified probability. If not, a violation is reported. We have applied this technique to check probabilistic properties in wireless sensor network applications.

The implementation is done using sliding windows by collecting a fixed number of samples and shifting the window appropriately over time. The sliding windows can detect different probabilistic behaviors over time. For example, a soft real-time system may behave well during normal loads but poorly during overloaded periods. When using all samples without the sliding windows, the difference in these situations may not be detected.

MaC does not provide probabilistic conditions because of possible inconsistency in the results of hypothesis testing. To understand the reason, we give possible definitions of probabilistic conditions. One can specify them as given that a MaC condition a is true, what is the probability that a MaC condition b is true? The sample can be collected by counting the number of states (or the time duration) where a and b are true against the number of states (or the time duration) where a is true. These definitions are sensitive to the way the system is instrumented or the time metric used to measure the duration of time. Fine-grained instrumentation (or time metric) gives more states (or larger duration) than coarse-grained ones. In either case, the resulted probability estimation using different levels of instrumentation or time metric may still be the same, only the numbers of samples that are different. However, z-score is sensitive to the number of samples and may produce different results even though the estimated probabilities are the same leading to possible inconsistency in reporting violations.

Other existing runtime verification frameworks that provide probabilistic properties do not use any statistical analysis to support the estimated probabilities. Most of their semantics also uses propositions, equivalent to MaC conditions, as their basic elements and are subject to the possible inconsistency of hypothesis testing results. MaC however provides events in addition to conditions and can use the statistical technique to check probabilistic events.

Our contributions are: 1) we provide a general statistical technique for checking probabilistic properties at runtime, 2) the technique is applied to and implemented in an existing runtime verification framework called MaC, and 3) a discussion is given about possible inconsistency when using the statistical technique with conditions or other runtime verification frameworks.