Statistical Model Checking for Safety Critical Hybrid Systems: An Empirical Evaluation

Youngjoo Kim[1], Moonzoo Kim[1], and Taihyo Kim[2]

[1] CS Dept. KAIST
Daejeon, South Korea
jerry88@cs.kaist.ac.kr
moonzoo@cs.kaist.ac.kr
[2] Formal Works Inc.
Seoul, South Korea
taihyo.kim@formalworks.com
http://www.formalworks.co.kr

Abstract. As more computing systems are utilized in various areas of our society, the reliability of computing systems becomes a significant issue. However, as the complexity of computing systems increases, conventional verification and validation techniques such as testing and model checking have limitations to assess reliability of complex safety critical systems. Such systems often control highly complex continuous dynamics to interact with physical environments. To assure the reliability of safety critical hybrid systems, *statistical model checking* (SMC) techniques have been proposed. SMC techniques approximately compute probabilities for a target system to satisfy given requirements based on randomly sampled execution traces. In this paper, we empirically evaluated four state-of-the-art SMC techniques on a fault-tolerant fuel control system in the automobile domain. Through the experiments, we could demonstrate that SMC is practically useful to assure the reliability of a safety critical hybrid system and we compared pros and cons of the four different SMC techniques.

1 Introduction

With the rapid advance of computing hardware, more computing systems are utilized in various areas of our society including avionics and automobiles. Consequently, the reliability of computing systems becomes a significant issue to our society. However, as computing power increases, the complexity of computing systems increases rapidly, which causes many challenges to assure reliability of computing systems. Conventional verification and validation (V&V) techniques such as testing and model checking have limitations to assess the reliability of complex safety critical computing systems, since such systems often control highly complex continuous dynamics to interact with physical environments.

To assure the reliability of safety critical hybrid systems, *statistical model checking (SMC)* techniques have been proposed [19, 17, 18, 8, 4, 21, 20, 2]. SMC techniques *approximately* compute probabilities for a target system to satisfy given requirements based on randomly sampled execution traces. Thus, SMC techniques can check the reliability of a safety critical hybrid system without analyzing the complex internal logic of the target system.

However, most literature on the SMC techniques focuses on theoretical aspects of suggested techniques, not their practical applicability to real-world safety critical systems.

In this paper, we empirically evaluated the *effectiveness* (in terms of the precision of the verification result) and *efficiency* (in terms of the verification time) of the following four representative state-of-the-art SMC techniques: *single sampling plan (SSP)*, *statistical probability ratio test (SPRT)*, *Bayesian hypothesis testing (BHT)*, and *Bayesian interval estimation testing (BIET).* [3] We applied these four SMC techniques to a fault-tolerant fuel control system (FFCS), which is a safety critical system for automobiles.

Contributions of this paper are as follows:

- We demonstrated that SMC techniques can assess the reliability of a complex safety critical system.
- We made empirical evaluation of the four state-of-the-art SMC techniques systematically with carefully controlled experiment environments.
- We identified and compared characteristics of the four SMC techniques, based on which precise results can be obtained faster by applying multiple SMC techniques together.

The organization of the paper is as follows. Section 2 overviews the four SMC techniques. Section 3 explains the target FFCS system. Section 4 describes the verification results by using the four SMC techniques on a Matlab/Simulink model of FFCS. Section 5 discusses issues from the empirical study. Section 6 concludes this paper with future work.

2 Background

In general, a model checking technique [1] checks whether a given model $\mathcal{M}$ satisfies a given requirement property ϕ ($\mathcal{M} \models \phi$) or not. A statistical model checking (SMC) technique checks whether a probability for $\mathcal{M}$ to satisfy ϕ is greater than or equal to a given threshold parameter θ ($\mathcal{M} \models P_{\geq\theta}(\phi)$) or not. We specify ϕ in bounded linear temporal logic (BLTL) [20] and that a probability for $\mathcal{M}$ to satisfy ϕ is greater than or equal to a given threshold θ in probabilistic bounded linear temporal logic (PBLTL) [21] (see Section 2.1). To compute the probability, SMC techniques utilize random sampling of execution traces/paths of $\mathcal{M}$ based on statistical techniques.

Figure 1 illustrates the overview of SMC. SMC receives a target model $\mathcal{M}$ which is an executable simulation model and PBLTL formula ϕ with θ. In addition, SMC receives *precision parameters* based on which the accuracy of the calculated probability is decided. SMC consists of three components: *simulator*, *BLTL model checker*, and *statistical analyzer*. Simulator executes $\mathcal{M}$ and generates a sample execution trace σ_i.

[3] In this study, we did not evaluate Chernoff-Hoeffding bound SMC technique [4] due to excessive time cost.

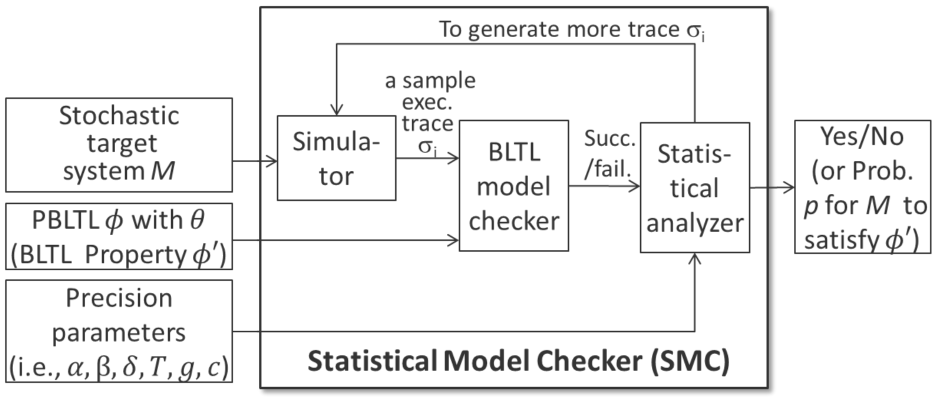
To generate more trace σ_i
a sample exec. trace σ_i
Stochastic target system M
Simula-tor
BLTL model checker
Succ. /fail.
Statis-tical analyzer
Yes/No (or Prob. p for M to satisfy ϕ')
PBLTL ϕ with θ (BLTL Property ϕ')
Precision parameters (i.e., $\alpha, \beta, \delta, T, g, c$)
Statistical Model Checker (SMC)

Fig. 1. SMC overview

BLTL model checker determines if σ_i satisfies ϕ and passes the result (i.e., success if σ_i satisfies ϕ; failure, otherwise) to statistical analyzer. Statistical analyzer calculates a probability p that $\mathcal{M}$ satisfies ϕ by collecting the result regarding if σ_i satisfies ϕ. Statistical analyzer requests simulator to generate σ_{i+1} repeatedly until the number of successful results of σs over the total number of σs is distributed within given precision boundary. Note that SMC does not analyze an internal logic of a target system, and thus SMC can validate complex safety critical systems without the state explosion problem.

More specifically, suppose that $X_1, ..., X_n$ are Bernoulli random variables (i.e., X_i can be either 0 or 1) of the model checking result of ϕ over an execution path σ of $\mathcal{M}$ and p indicates a probability of X_i to become 1 (i.e., $P(X_i = 1) = p$). Since we do not know p exactly, we should estimate p using random sampling techniques with user-given precision parameters. We pick a sample path σ_i from $\mathcal{M}$ by executing $\mathcal{M}$ and test whether σ_i satisfies ϕ or not. If σ_i satisfies ϕ, $x_i = 1$; $x_i = 0$ otherwise. Note that, for estimating p, we should determine a number of sample paths n to check ϕ using statistical techniques. We may obtain n statically by using heuristics or dynamically through iterative sampling.

There are two classes of statistical techniques: *hypothesis testing* (Section 2.2) and *estimation testing* (Section 2.3).

2.1 Probabilistic Bounded Linear Temporal Logic (PBLTL)

To define PBLTL, we first define a syntax and semantics of bounded linear temporal logic(BLTL) [20], and then extend BLTL to PBLTL [21].

For a target model $\mathcal{M}$, SV is a finite set of real-valued state variables. A Boolean predicate over SV is a constraint of the form $y \sim v$, where $y \in SV$, $\sim \in \{\geq, \leq, =\}$, and $v \in \mathbb{R}$. The syntax of the BLTL logic formula ϕ is given by the following grammar:

$$\phi ::= y \sim v \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \wedge \phi_2) \mid \neg\phi_1 \mid (\phi_1 \mathbf{U}^t \phi_2),$$

where $y \in SV$, $\sim \in \{\geq, \leq, =\}$, $v \in \mathbb{R}$, and $t \in \mathbb{R}_{\geq 0}$.

For other temporal operators, we can define $\mathbf{F}^t\phi$ as $True\ \mathbf{U}^t\phi$ and $\mathbf{G}^t\phi$ as $\neg\mathbf{F}^t\neg\phi$. We denote a fact that an execution σ satisfies a property ϕ as $\sigma \models \phi$. We use σ^k to denote a suffix trace of σ starting at step k (σ^0 denotes the original execution σ). We denote the value of a state variable y in σ at step k by $V(\sigma, k, y)$. We define t_k as a time at step k and t as a time bound. The semantics of BLTL on a trace σ^k is defined as follows:

- $\sigma^k \models y \sim v$ iff $V(\sigma, k, y) \sim v$
- $\sigma^k \models \phi_1 \vee \phi_2$ iff $\sigma^k \models \phi_1$ or $\sigma^k \models \phi_2$
- $\sigma^k \models \phi_1 \wedge \phi_2$ iff $\sigma^k \models \phi_1$ and $\sigma^k \models \phi_2$
- $\sigma^k \models \neg\phi_1$ iff $\sigma^k \not\models \phi_1$
- $\sigma^k \models \phi_1\mathbf{U}^t\phi_2$ iff there exists $i \in \mathbb{N}$ such that
  1. $\sum_{0\leq l<i} t_{k+l} \leq t$,
  2. $\sigma^{k+i} \models \phi_2$, and
  3. for each $0 \leq j < i, \sigma^{k+j} \models \phi_1$

A probabilistic bounded linear temporal logic (PBLTL) formula is a formula of the form $P_{\geq\theta}(\phi)$, where ϕ is a BLTL formula and $\theta \in (0, 1)$ is a probability threshold. We denote that a model $\mathcal{M}$ satisfies PBLTL property $P_{\geq\theta}(\phi)$ as $\mathcal{M} \models P_{\geq\theta}(\phi)$, which means that a probability for $\mathcal{M}$ to satisfy ϕ is greater than equal to θ (see [21] for detailed description).

2.2 Hypothesis Testing

For hypothesis testing, we build a hypothesis $H : p \geq \theta$ against an alternative hypothesis $K : p < \theta$ where θ is a threshold over (0,1) and p is a *true probability* that $\mathcal{M}$ satisfies ϕ. Hypothesis testing checks whether H is accepted or not based on the randomly sampled paths. In this paper, we utilize the following three hypothesis testing techniques - *single sampling plan (SSP)*, *sequence probability ratio test (SPRT)*, and *Baysian hypothesis testing (BHT)*.

Single Sampling Plan (SSP) SMC techniques cannot compute a true probability p exactly, but can estimate p within given error bounds. Precision parameters for SSP [17] are *error bounds* α and β, and a half size of *indifference region* δ. For testing a hypothesis H, there are two types of errors such as false negative (also known as a type I error) which rejects a true hypothesis H and false positive (also known as a type II error) which accepts a false hypothesis H. We can bound an error probability of a false negative error within α. Similarly, we can bound an error probability of a false positive error within β. The left side of Figure 2 presents the function of probability L_p of accepting the hypothesis H as a function of p with the probability of a type I error and type II error as exactly α and β. However, we want to give similar probability L_p with $p = \theta$ to $p = \theta - \epsilon$ for arbitrarily small $\epsilon > 0$ for reality. To solve this problem, we introduce *indifference region* (p_1, p_0) around θ where $p_0 = \theta + \delta$, $p_1 = \theta - \delta$, and δ is a half size of indifference region (see right side function in Figure 2). Therefore, instead

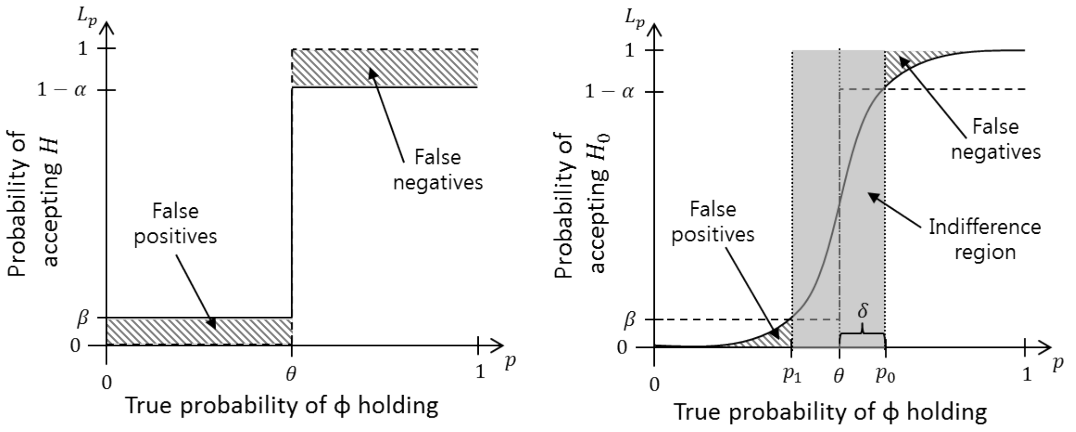

Fig. 2. Function of probability L_p of accepting the hypothesis $H : p \geq \theta$ (left side) and function of probability L_p of accepting the hypothesis $H_0 : p \geq p_0$ with indifference region (right side).

of testing H against K, we use the modified hypothesis $H_0 : p \geq p_0$ against the alternative hypothesis $H_1 : p < p_1$. If the probability p is in (p_1, p_0), then p is sufficiently close to θ so that we do not care which hypothesis is accepted.

For SSP, a user can determine a maximum number of sample paths n and a threshold number of success sample paths c statically. After determining n and c, SSP executes a target program multiple times. If the number of success sample paths that satisfy ϕ are greater than c, then H is accepted; K is accepted otherwise. Then, we can express the probability that the number of success sample paths among n samples are less than c with the cumulative distribution function for binomial distribution $B(n, p)$:

$$F(c; n, p) = \sum_{i=0}^{c} \binom{n}{i} p^i (1-p)^{n-i}.$$

Therefore, we accept H with $1 - F(c; n, p)$ using n and c, and accept K with $F(c; n, p)$ using n and c. We can obtain minimal value for n and c using binary search based algorithm with given p_0, p_1, α, and β. Note that SSP is the only SMC technique that computes the number of required sample paths statically among the SMC techniques utilized in this study.

Sequence Probability Ratio Test (SPRT) SPRT [19, 17, 18, 15] determines a number of required sample paths dynamically at runtime. If another sample path is needed, SPRT generates one more sample path by executing a target system. If the information from generated sample paths are enough to determine hypothesis H_0, SPRT stops executing a target program and outputs the result that H_0 is accepted or not. SPRT uses precision parameter inputs α, β, and δ which are same in SSP.

SPRT operates as follows. After generating mth sample paths of the test, we calculate the quantity

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr[X_i = x_i | p = p_1]}{Pr[X_i = x_i | p = p_0]} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}$$

where $d_m = \sum_{i=1}^{m} x_i$ and x_i is ith observation of $\sigma_i \models \phi$. p_{jm} is the probability of the sequence $x_1, ..., x_m$ with $Pr[X_i = 1] = p_j$ for j=0,1. Therefore, the above quantity makes the ratio of two probabilities, the *probability ratio*. The hypothesis H_0 is accepted if

$$\frac{p_{1m}}{p_{0m}} \leq B,$$

and the hypothesis H_1 is accepted if

$$\frac{p_{1m}}{p_{0m}} \geq A.$$

Otherwise, we should generate $m+1$th sample path of the test. A and B are selected to bound error probability α and β, with $A > B$. In practice, we choose $A = \frac{1-\beta}{\alpha}$ and $B = \frac{\beta}{1-\alpha}$(detailed description is found in [15, 17]).

Bayesian Hypothesis Testing (BHT) BHT [8] dynamically determines the number of sample paths during simulation as same in SPRT. BHT uses two precision parameter inputs such as threshold T of determining H_0 and prior density g for p, the actual probability of satisfying ϕ. In Bayes' theorem, we get prior probability using current information first. After obtaining new information, we can obtain posterior probability refining prior probability. BHT uses Bayes' theorem to determine the number of sample paths of the test.

Let $P(H_0)$ and $P(H_1)$ be the strictly positive *prior probabilities* of accepting H_0 and H_1 and satisfying $P(H_0) + P(H_1) = 1$. Let $d = (x_1, ..., x_n)$ be a sequence of n sample paths of the test. Bayes' theorem states that the *posterior probabilities* of accepting H_0 and H_1 based on observations of d are

$$P(H_0|d) = \frac{P(d|H_0)P(H_0)}{P(d)} \qquad P(H_1|d) = \frac{P(d|H_1)P(H_1)}{P(d)}$$

for every d with $P(d) = P(d|H_0)P(H_0) + P(d|H_1)P(H_1) > 0$.

BHT operates as follows. After generating mth sample paths of the test, we can calculate the quantity

$$\frac{P(H_0|d)}{P(H_1|d)} = \frac{P(d|H_0)}{P(d|H_1)} \cdot \frac{P(H_0)}{P(H_1)}$$

where $d = (x_1, ..., x_m)$. We call the above quantity as the ratio of the posterior probabilities. Here, we define the *Bayes factor* $\mathcal{B}$ of d and hypotheses H_0 and H_1 as follows:

$$\mathcal{B} = \frac{P(d|H_0)}{P(d|H_1)}$$

The *Bayes factor* $\mathcal{B}$ can be interpreted as a measure of the evidence in favor of H_0 and also $\frac{1}{\mathcal{B}}$ can be the evidence in favor of H_1. We introduce a Bayes factor threshold T to test H_0 against H_1 such that $T \geq 1$. The hypothesis H_0 is accepted if $\mathcal{B} > T$, and the hypothesis H_1 is accepted if $\mathcal{B} < \frac{1}{T}$. Otherwise, BHT generates $m+1$th sample path using simulation [4] (detailed description is found in [8]).

2.3 Estimation Testing

Estimation testing can approximately compute p, the probability that the model $\mathcal{M}$ satisfies the given property ϕ expressed by bounded linear temporal logic (BLTL). With p, we can determine whether the probabilistic bounded linear temporal logic (PBLTL) is satisfied or not. For that purpose, we use a following statistical estimation testing technique.

Bayesian Interval Estimation Testing (BIET) BIET [21] dynamically determines the number of sample paths for checking the satisfiability of the model $\mathcal{M}$ with the property ϕ during simulation as SPRT and BHT do. BIET also uses the Bayes' theorem. BIET uses four precision parameter inputs such as a half-size δ' of an estimation interval which will contain p with high probability, the coverage goal c of the estimation interval, and the parameters α', β' of the Beta prior. In fact, BIET estimates interval around the probability p instead of estimating p, but we regard the mean of the estimated interval as $\hat{p}$, the estimated value of *true probability* p, i.e., the estimated interval is $(\hat{p}-\delta', \hat{p}+\delta')$. We call the estimated interval as (t_0, t_1). We have a *coverage goal* such that the probability that the probability satisfying $\mathcal{M} \models \phi$ is in (t_0, t_1) should be over the coverage $c \in (\frac{1}{2}, 1)$. The exact description of the coverage goal is as follows:

$$\int_{t_0}^{t_1} f(u|x_1, ..., x_n)du = c$$

where x_i is ith observation of $\sigma_i \models \phi$ for $i = 1, ..., n$ and n is the number of sample paths. We call the coverage goal as a $100c$ percent *Bayesian interval estimate* of p. Since BIET uses the Bayes' theorem, we need prior information, i.e., prior density of p to obtain prior probability. For simplicity, we focus on the Beta prior with parameters α', β'(See [21] for details).

At mth stage of the test, by Beta prior with α', β', we can calculate the quantity

$$\hat{p} = \frac{x + \alpha'}{m + \alpha' + \beta'}$$

where $x = \sum_{i=1}^{m} x_i$ is the number of success sample paths during m number of sample paths. Next, using $t_0 = \hat{p} - \delta', t_1 = \hat{p} + \delta'$, we can calculate the quantity

$$\gamma = \int_{t_0}^{t_1} f(u|x_1, ..., x_m)du$$

[4] T corresponds to the inverse number of error bounds α and β for SSP and SPRT [21].

where γ is the coverage of m number of sample paths for checking $\mathcal{M} \models \phi$. If $\gamma \geq c$, then BIET stops the simulation and outputs t_0, t_1, and $\hat{p}$. Otherwise, BIET generates $m + 1$th sample path and repeats.

3 Fault-tolerant Fuel Control System

This section overviews a fault-tolerant fuel control system (FFCS) [12] in an automobile domain. We selected FFCS as a target system to apply the SMC techniques for the following reasons:

- FFCS is a safety critical system whose reliability is very important.
- FFCS is a complex real-world application, not a toy example such as ones in probabilistic symbolic model checker (PRISM) [11] benchmarks. Most SMC papers use PRISM benchmarks as their target examples.
- A Simulink/stateflow model of FFCS is publicly available. Thus, it is convenient to build prototypes of the SMC techniques for FFCS based on a Simulink/stateflow simulator.

Figure 3 is an overall diagram of FFCS. FFCS [12] controls a fuel rate to inject fuel based on sensor data for best performance, detects a sensor fault, and shuts down an engine for safety in the presence of multiple sensor failures. FFCS has the following four sensors: throttle angle sensor, speed sensor, exhaust gas oxygen (EGO) sensor, and manifold absolute pressure (MAP) sensor. FFCS receives these four sensor inputs and generates a proper fuel rate and an air-fuel ratio. FFCS consists of the following three components: fuel rate controller, air-fuel ratio calculator, and sensor failure detector. Fuel rate controller receives the four sensor data and calculates a proper fuel rate to make an air-fuel ratio optimal (i.e., 14.6). Air-fuel ratio calculator receives EGO sensor data and a fuel rate and calculates the air-fuel ratio. Sensor failure detector receives all four sensor data and controls the fuel rate controller to increase/decrease the fuel rate in the presence of a single sensor fault or shuts down the engine if multiple sensors fail, since the air-fuel ratio cannot be controlled with failures of multiple sensors.

The size and complexity of the Simulink/stateflow FFCS model in terms of Halstead [3] metrics are described in Table 1. We counted each atomic block (i.e., a module of a mathematical function or control logic) as an operator and each input of an atomic block as an operand of the Simulink/stateflow FFCS model. The automatically generated C code from the model has 8266 LOC in 222 functions.

Table 1. Size and complexity of the FFCS Simulink/stateflow model in Halstead metrics

N_1: # of operators	N_2:# of operands	n_1:# of distinct operators	n_2:# of distinct operands	N:program length (= $N_1 + N_2$)	n: program vocabulary (=$n_1 + n_2$)	V: program volume ($N \times logn$)	D: program difficulty (=$n_1/2 \times N_2/n_2$)	E: program effort (= D $\times$ V)
65	111	35	94	176	129	1234.0	20.7	25500.0

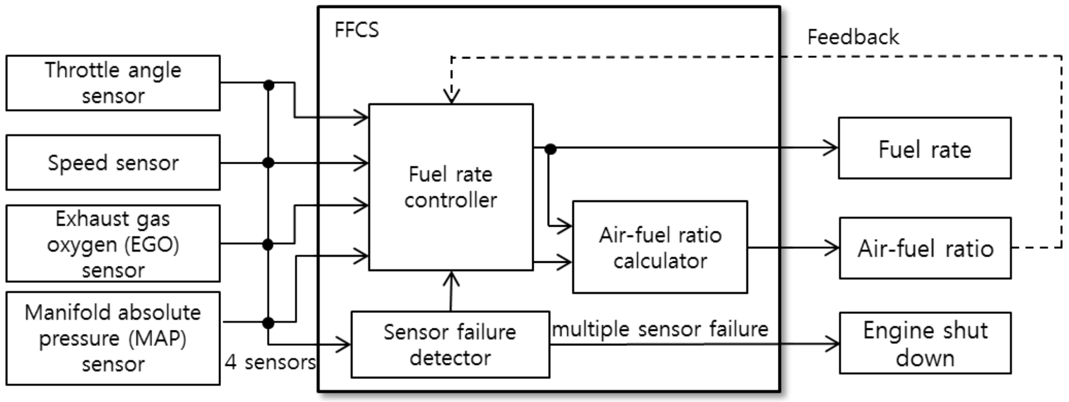

Fig. 3. Block diagram of FFCS

A requirement property for FFCS is that a probability that the fuel rate does not become zero for one second in 100 seconds should be greater than equal to threshold θ. The property is crucial in a real world, because if the fuel rate is zero for one second, then the engine stops and can cause a serious accident. This property can be expressed by PBLTL as follows [21]:

$$P_{\geq\theta}(\neg(F^{100}G^{1}(fuelrate = 0)))$$

4 Experimental Study

We have applied the four SMC techniques to FFCS with precision parameters as independent variables and checked whether FFCS satisfies the given requirement property in PBLTL or not.

4.1 Experiment Setup

We set a stochastic environment for FFCS as follows. The environment of FFCS generates random faults at the EGO, MAP, and speed sensors as [21] does. The random faults are modeled by three independent Poisson processes with different arrival rates [16]. We assume one fault event remains for one second. When a fault event occurs in a sensor, FFCS remains in a failure mode for one second and returns to a normal mode. We utilize the following four inter-arrival fault rates (i.e., mean inter-arrival times of sensor fault) to the three sensors: (3,7,8), (10,8,9), (20,10,20) and (30,30,30).

For the SMC techniques, we use the following precision parameters:

– Hypothesis testing techniques
  - SSP:
    - threshold $\theta \in \{0.5, 0.7, 0.9, 0.99\}$
    - a half-size of indifference region $\delta \in \{0.01, 0.03, 0.05\}$

    * error bounds $\alpha, \beta \in \{0.1, 0.01, 0.001\}$
  - SPRT:
    * threshold $\theta \in \{0.5, 0.7, 0.9, 0.99\}$
    * a half-size of indifference region $\delta \in \{0.01, 0.03, 0.05\}$
    * error bounds $\alpha, \beta \in \{0.1, 0.01, 0.001\}$
  - BHT:
    * threshold $\theta \in \{0.5, 0.7, 0.9, 0.99\}$
    * Bayes factor threshold $T \in \{10, 100, 1000\}$
    * prior density g = uniform density over (0,1)
- Estimation testing technique
  - BIET:
    * interval coverage $c = \{0.9, 0.99, 0.999\}$
    * a half-size of estimation interval $\delta' = \{0.01, 0.03, 0.05\}$
    * parameters of Beta prior $\alpha' = \beta' = 1$ [5]

We performed each experiment five times to obtain average verification result over $[0, 1]$ regarding whether the hypothesis H is accepted or not where H: a probability to satisfy $\phi(= \neg(F^{100}G^{1}(fuelrate = 0)))$ is greater than or equal to θ. In addition, we measured the average verification time for each experiment.

We built a statistical model checker as a Matlab module which runs together with a FFCS model. We use a Matlab simulator as a simulator component to generate an execution trace σ of a Matlab/Simulink FFCS model. Then, the BLTL model checker analyzes if σ satisfies the requirement property ϕ. After the BLTL model checker evaluates σ, the statistical analyzer calculates a required number of sample traces dynamically based on the precision parameters and the number of success/fail sample traces generated so far. If a number of the generated samples reaches the required number, the statistical model checker generates a verification result and terminates the SMC process. Note that all sub-components of SMC are independent from each other and can be re-used for other target systems without modification. Thus, it will not be difficult for practitioners to apply SMC techniques to their safety critical systems. [6]

We used Matlab R2010a for the experiments. All experiments were performed on 64 bit Windows 7 Professional K equipped with a 3 GHz Intel processor and 16 gigabytes of memory.

4.2 Experimental Results

Tables 2-4 describe the experiment results of applying the hypothesis testing techniques to FFCS with fault inter-arrival rate (3,7,8) and $\delta = 0.03$. [7] In these three tables,

- θ is a threshold of the hypothesis H for SSP, SPRT, and BHT

[5] $\alpha' = \beta' = 1$, since we assume the prior density to be a uniform density over $(0, 1)$.

[6] We have released the statistical analyzers using SSP, SPRT, BHT, and BIET techniques publicly at `http://pswlab.kaist.ac.kr/tools/SMC/`.

[7] Full experiment data with the other three fault inter-arrival rates and $\delta \in \{0.01, 0.05\}$ is available at `http://pswlab.kaist.ac.kr/data/hvc2012-expr-results.zip`

Table 2. Experiment result of SSP with fault rate $(3, 7, 8)$ and $\delta = 0.03$

α, β	threshold θ															
	0.5				0.7				0.9				0.99			
	n	m	$acpt$	$time$	n	m	$acpt$	$time$	n	m	$acpt$	$time$	n	m	$acpt$	$time$
0.1	455	255.3	1.0	688.3	386	307.0	1.0	821.5	161	141.5	0.0	381.3	57	5.8	0.0	17.1
0.01	1501	857.8	1.0	2308.1	1261	1001.5	1.0	2686.7	531	468.8	0.0	1256.4	113	5.0	0.0	14.8
0.001	2649	1487.8	1.0	4013.2	2226	1764.3	1.0	4760.8	932	806.8	0.0	2172.5	170	6.0	0.0	20.3

Table 3. Experiment result of SPRT with fault rate $(3, 7, 8)$ and $\delta = 0.03$

α, β	threshold θ											
	0.5			0.7			0.9			0.99		
	n	$acpt$	$time$	n	$acpt$	$time$	n	$acpt$	$time$	n	$acpt$	$time$
0.1	26.6	1.0	17.6	34.0	1.0	22.4	108.4	0.0	71.5	5.6	1.0	3.7
0.01	49.0	1.0	32.3	93.4	1.0	61.6	484.0	0.0	319.4	5.6	1.0	3.7
0.001	72.8	1.0	48.0	127.6	1.0	84.2	786.6	0.0	519.2	11.6	1.0	7.7

Table 4. Experiment result of BHT with fault rate $(3, 7, 8)$

T	threshold θ											
	0.5			0.7			0.9			0.99		
	n	$acpt$	$time$	n	$acpt$	$time$	n	$acpt$	$time$	n	$acpt$	$time$
10	3.6	1.0	2.4	5.0	1.0	3.3	42.2	0.8	27.9	21.0	0.2	13.9
100	7.6	1.0	5.0	26.0	1.0	17.2	3917.2	0.2	2585.4	27.0	0.0	17.8
1000	13.6	1.0	9.0	48.4	1.0	31.9	4013.2	0.2	2648.7	35.2	0.0	23.2

- n is a maximum number of required sample paths and m means an average number of sample paths generated for SSP. For SPRT and BHT, n is an average number of sample paths generated for SPRT and BHT.
- $acpt$ is an average result over $[0, 1]$ regarding the hypothesis H where 0 is 'reject' and 1 is 'accept'
- $time$ is an average verification time for each experiment in seconds

Table 5 describes the experiment result of applying the estimation technique BIET to FFCS with fault inter-arrival rate (3,7,8), where n is an average number of sample paths, $\hat{p}$ is an estimated probability to satisfy ϕ, and $time$ indicates an average verification time in seconds. Tables 2-5 show that n (m for SSP) increases as the precision parameters becomes smaller. For example, for SSP, when α and β decrease from 0.1

Table 5. Experiment result of BIET with fault rate $(3, 7, 8)$

δ'	interval coverage c								
	0.9			0.99			0.999		
	n	$\hat{p}$	$time$	n	$\hat{p}$	$time$	n	$\hat{p}$	$time$
0.05	104.8	0.8835	69.2	273.0	0.8849	180.2	475.5	0.8830	313.8
0.03	276.6	0.8944	182.6	729.4	0.8889	481.4	1191.5	0.8924	786.4
0.01	2733.8	0.8856	1804.3	6696.5	0.8861	4419.7	10924.2	0.8865	7210.0

to 0.001 with threshold $\theta = 0.5$, m increases from 255.3 to 1487.8 (Table 2). Similar tendencies are observed for SPRT, BHT, and BIET.

Regarding Effectiveness (Precision of the Verification Results) All four techniques produce similar results. For hypothesis testing techniques SPRT, SSP, and BHT, the probability for FFCS with the fault inter-arrival rate of sensors (3,7,8) and $\delta = 0.03$ to satisfy the requirement property ϕ is between 0.7 and 0.9. This is because $acpt$s are 1.0 when $\theta \leq 0.7$ while $acpt$s are close to 0 when $\theta \geq 0.9$ in Tables 2-4.[8] Also, note that n of SPRT and BHT increases exponentially as θ increases from 0.5 to 0.9, and decreases sharply from 0.9 to 0.99. For example, for SPRT with α=β=0.1 (Table 3), n becomes 26.6, 34.0, 108.4 and 5.6 as θ becomes 0.5, 0.7, 0.9 and 0.99, respectively. In general, for the hypothesis testing techniques that generates sample paths dynamically (i.e., SPRT and BHT), if a true probability is close to the threshold θ, a large number of sample paths is required to determine whether a given hypothesis H is accepted or not. By the above results, we can conclude that a true probability that FFCS with the fault rate (3,7,8) satisfies the requirement property is close to 0.9. Furthermore, BIET computes the probability between 0.8830 (with $c = 0.999$ and $\delta' = 0.05$) and 0.8944 (with $c = 0.9$ and $\delta' = 0.03$) (Table 5), which is included in the estimated probability interval (0.7,0.9) of the hypothesis testing techniques. Therefore, based on the above analysis of the results, we can conclude that the verification results of the SMC techniques are precise.

Regarding Efficiency (Verification Time) The time taken for each experiment was moderate. The longest experiment took 7210.0 seconds (i.e., around 2 hours) to generate 10924.2 sample paths on average for BIET with $c = 0.999$ and $\delta' = 0.01$ (Table 5). Note that most other experiments took much less time. For example, the longest experiments in SSP, SPRT, and BHT took 4760.8 (α=β=0.001 and θ=0.7) (Table 2), 519.2 (α=β=0.001 and θ=0.9) (Table 3), and 2648.7 (T=1000 and θ=0.9) (Table 4) seconds,

[8] The result of SPRT with $\theta = 0.99$ is not reliable, since the precision of SPRT is low when θ is close to 1. Also, note that n becomes very small (i.e., less than 12) with θ=0.99 in Table 3.

respectively. Therefore, we can conclude that statistical model checking can assure reliability of a complex target system at modest cost. [9]

5 Discussion

5.1 Practicality of Statistical Model Checking

Through the empirical evaluation of the SMC techniques on FFCS, we believe that statistical model checking is practically useful for the following reasons:

- SMC can check a probability for a complex hybrid system to satisfy a given requirement property ϕ. In this project, we could statistically check the probability for FFCS to satisfy ϕ, since we just generated random sample execution paths without analyzing the internal structure of FFCS, which is a great advantage of SMC.
- SMC allows a user to select proper trade-off between verification precision and time cost by selecting appropriate precision parameter values (Section 4.2). In some cases, due to limited project time, it may be more valuable to obtain less precise verification in short time than more precise verification result in much longer time.
- The SMC techniques can obtain precise verification results in a moderate amount of verification time (i.e., less than two hours for the most experiments in Section 4.2). [10]

5.2 Necessity of Proper Precision Parameter Values

We found that, for SSP and SPRT to produce precise verification results, δ should be very small when θ is close to 1. For example, the verification result of SPRT was 'accept' for $\theta = 0.99$ with δ=0.03 (see Table 3), which is considered as an incorrect result, since the other SMC techniques conclude that the estimated probability is between 0.7 and 0.9 (Section 4.2). The reason for these imprecise results of SSP and SPRT is due to the limited size of indifference region. For example, if the threshold θ is 0.99 and $\delta \geq$ 0.01, then p_0 becomes 1, which causes the denominator of the probability ratio $\frac{p_{1m}}{p_{0m}}$ to be 0 when one false sample occurs for SPRT, which can cause imprecise result. For SSP, when n=170 with $\alpha = \beta = 0.001$ and δ= 0.03, a number of success samples should be larger than 169 to accept H. In other words, if one sample path violates ϕ, then the verification finishes immediately with 'reject' result. Therefore, SSP and SPRT should be applied with very small δ when θ is close to 1.

In addition, BHT with threshold $\theta = 0.9$ produced different verification results with different T. With T=10, the verification result was 0.8 (i.e., almost 'accept') on average. However, with T=100 or 1000, the verification results were 0.2 (i.e., almost 'reject') on average. From the results of the other techniques which indicate the true probability $p \in (0.7, 0.9)$ (Section 4.2), we can conclude that the verification result with T=10

[9] SSP takes much more time to generate one sample than the other techniques, since the heuristics of SSP to determine a maximum number of sample paths is very complex.

[10] If the required reliability goal is very high (i.e., from $1 - 10^{-4}$ to $1 - 10^{-5}$ for SIL 4 level [6]), SMC may take multiple weeks.

Table 6. Comparison of the four statistical model checking techniques

	Technique	Precision	Speed	# of sample decision	Applicability
Hypothesis testing	SSP	Low when θ is close to 1	Slow except when θ is close to 1	Static	Low
	SPRT	Low when θ is close to 1	Fast	Dynamic	Middle
	BHT	Middle	Slow when θ is close to *true probability*	Dynamic	High
Estimation testing	BIET	High	Slow	Dynamic	High

was imprecise. This is because T was not sufficiently small enough to obtain a precise verification result. Therefore, proper precision parameter values are important to obtain precise verification results.

5.3 Comparison of the SMC techniques

Table 6 summarizes characteristics of the four SMC techniques. The precision of SSP and SPRT is lower than the other techniques when θ is close to 1 because of the size restriction of the indifference region (Section 5.2). The precision of BIET is higher than the other techniques by the *law of large numbers* [14], because BIET utilizes more samples than the other techniques. BHT achieves a middle level of precision compared to SSP/SPRT and BIET. Regarding verification speed, SSP is slow except when θ is close to 1; when θ is close to 1, SSP is fast (but imprecise) since a number of samples is small. BHT is slow by generating a large number of samples when θ is close to a true probability. BIET is relatively slow due to a large number of samples utilized. SPRT is relatively fast, since it does not have weaknesses of the other techniques in terms of the verification speed. By considering these aspects, the applicability of BHT and BIET is relatively higher than that of SPRT and SSP.

As shown in Table 6, there is no single best SMC technique for all aspects. Thus, a combination of different SMC techniques can achieve precise result faster. For example, many safety critical systems should satisfy requirement property ϕ with very high probability for reliable operations (i.e., θ should be larger than 0.9999). We know that SPRT is faster than BIET, but its precision is low when θ is close to 1. In such cases, we can first apply SPRT to a target system with low θ for fast verification speed. If the verification results for low θ values (i.e., $\theta \in [0.5, 0.7]$) are 'reject', then we do not need to verify a target system further. Otherwise, we use BIET for higher θ (i.e., $\theta \in [0.9, 0.99]$), which is more precise but slower than SPRT, since SPRT is imprecise for θ close to 1. Consequently, this combined method can achieve precise result faster than BIET only. [11]

[11] From this observation, we have developed a hybrid SMC technique which combines SPRT, the fastest SMC technique and BIET, the most accurate SMC technique. We have showed that

6 Conclusion and Future Work

From our empirical study, we demonstrated that SMC techniques can assess the reliability of a complex safety critical system such as FFCS. Based on the statistical techniques, SMC techniques can estimate the reliability of a complex safety critical hybrid system, to which conventional V&V techniques often fail to apply due to high complexity of a target system.

Therefore, we believe that industries on safety critical system domain can benefit from the SMC techniques much. As market competition becomes severe, many companies try hard to improve the quality of their products and to obtain safety certificate such as ISO 26262 [7] for automobiles and DO178B/C [13] for avionics by validating the reliability of the products. However, it has been very difficult to validate the reliability of complex hybrid systems due to aforementioned reasons. SMC can be used to validate the reliability goal assigned to a target system/component effectively and efficiently. In [9], we have demonstrated that SMC can be a solution for validating software reliability at an early development stage to reduce the defect correction cost of conventional software reliability assessment procedures such as IEEE Std.1633 [5].

As future work, to improve the practicality of the SMC techniques further, we plan to collaborate with automobile companies like Hyundai or Kia on the application of the SMC techniques on automobile controllers. In addition, we will develop a safety engineering process to validate software reliability based on the SMC techniques, which is essential to obtain safety certificate.

Acknowledgment

This work was supported by the Excellent Research Center (ERC) of Excellence Program of Korea MEST/NRF of Korea (Grant 2012-0000473), the IT R&D program of MKE/KEIT [10041752, Research and Development of Dual Operating System Architecture with High-Reliable RTOS and High-Performance OS], and Dual Use Technology Program in Korea.

References

1. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods System Design (FMSD)*, 19(1):7–34, 2001.
2. E. Clarke, A. Donze, and A. Legay. Statistical model checking of mixed-analog circuits with an application to a third order delta-sigma modulator. In *Haifa Verification Conference (HVC)*, 2008.
3. M.H. Halstead. *Elements of Software Science*. Elsevier Science Ltd, 1977.
4. T. Herault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2004.
5. IEEE Computer Society. IEEE Std 1633: IEEE Recommend Practice on Software Reliability, 2008.

this hybrid SMC technique improves effectiveness and efficiency compared to a single SMC technique [10].

6. International Electrotechnical Commission (IEC). IEC 61508: Functional safety of electrical/electronic/programmable electronic (E/E/PE) safety related systems, 2005.
7. International Organization for Standardization (ISO). ISO 26262: Road vehicles – functional safety, 2011. `http://www.iso.org/iso/catalogue_detail?csnumber=43464`.
8. S.K. Jha, E.M. Clarke, C.J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. In *Conference on Computational Methods in Systems Biology (CMSB)*, 2009.
9. Y. Kim, O. Choi, M. Kim, J. Baik, and T. Kim. Validating software reliability through statistical model checking: Safer, cheaper, and faster. *IEEE Software*. under review.
10. Y. Kim, M. Kim, and T. Kim. Hybrid statistical model checking technique for reliable safety critical systems. In *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2012.
11. M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification (CAV)*, 2011.
12. J. Lauber, T.M. Guerra, and M. Dambrine. Air-fuel ratio control in a gasoline engine. *International Journal of Systems Science (IJSySc)*, 42(2):277–286, 2011.
13. Radio Technical Commission for Aeronautics (RTCA). Do-178c: Software considerations in airborne systems and equipment certification, 2012.
14. P.K. Sen and J.M. Singer. *Large sample methods in statistics: An Introduction with Applications*. New York: Chapman & Hall, 1993.
15. A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.
16. S. Yi, J. Heo, Y. Cho, and J. Hong. Adaptive mobile checkpointing facility for wireless sensor networks. In *International Conference on Computational Science and Its Applications (ICCSA)*, 2006.
17. H.L.S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, CMU, Jan. 2005.
18. H.L.S. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *Software Tools for Technology Transfer (STTT)*, 8(3):216–228, 2006.
19. H.L.S. Younes and D.J. Musliner. Probabilistic plan verification through acceptance sampling. In *AIPS Workshop on Planning via Model Checking*, 2002.
20. H.L.S. Younes and R.G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Journal Information and Computation (JIC)*, 204(9):1368–1409, 2006.
21. P. Zuliani, A. Platzer, and E.M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. In *Hybrid Systems: Computation and Control (HSCC)*, 2010.