석사 학위논문

Master's Thesis

신뢰성이 높은 안전필수시스템을 위한 하이브리드 통계적 모델 체킹 방법

Hybrid Statistical Model Checking Technique for Reliable Safety Critical Systems

김 영 주 (金 映 周 Kim, Youngjoo)

전산학과

Department of Computer Science

KAIST

2013

신뢰성이 높은 안전필수시스템을 위한 하이브리드 통계적 모델 체킹 방법

Hybrid Statistical Model Checking Technique for
Reliable Safety Critical Systems

Hybrid Statistical Model Checking Technique for Reliable Safety Critical Systems

Advisor : Professor Kim, Moonzoo

by

Kim, Youngjoo
Department of Computer Science
KAIST

A thesis submitted to the faculty of KAIST in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Computer Science . The study was conducted in accordance with Code of Research Ethics[1].

2012. 12. 17.

Approved by

Professor Kim, Moonzoo

[Advisor]

[1]Declaration of Ethical Conduct in Research: I, as a graduate student of KAIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

신뢰성이 높은 안전필수시스템을 위한 하이브리드 통계적 모델 체킹 방법

김 영 주

위 논문은 한국과학기술원 석사학위논문으로
학위논문심사위원회에서 심사 통과하였음.

2012년 12월 17일

MCS
20113125

김 영 주. Kim, Youngjoo. Hybrid Statistical Model Checking Technique for Reliable Safety Critical Systems. 신뢰성이 높은 안전필수시스템을 위한 하이브리드 통계적 모델 체킹 방법. Department of Computer Science . 2013. 44p. Advisor Prof. Kim, Moonzoo. Text in English.

ABSTRACT

Reliability of safety critical systems such as nuclear power plants and automobiles has become a significant issue to our society. As more computing systems are utilized in these safety critical systems, there are high demands for verification and validation (V&V) techniques to assure the reliability of such complex computing systems. However, as the complexity of computing systems increases, conventional V&V techniques such as testing and model checking have limitations, since such systems often control highly complex continuous dynamics. To improve the reliability of such systems, *statistical model checking* (SMC) techniques have been proposed. SMC techniques can check if a target system satisfies given requirements through statistical methods. In this thesis, first, we have emperically evaluated four state-of-the-art SMC techniques in the automobile domain to see the applicability of SMC for assuring the reliability of safety critical systems and compare pros and cons of the four different SMC techniques. Second, we propose a new hybrid SMC technique that integrates sequential probability ratio test (SPRT) technique and Bayesian interval estimation testing (BIET) technique to achieve precise verification results quickly. In our experiment, the new hybrid SMC was around 4 times faster than BIET. In addition, we demonstrate the effectiveness and efficiency of this hybrid SMC technique by applying the hybrid SMC technique to three safety critical systems in the automobile domain. Finally, as a solution for validating software reliability at an early stage, we propose a methodology utilizing statistical model checking (SMC) techniques. Reliability validation is performed by comparing the allocated reliability goal with the calculated reliability using the probabilities and the relative weight values for the safety functional requirements. By conducting reliability validation early, we can prevent the propagation of the reliability allocation errors and design errors into the later phases. Thereby, we can achieve safer, cheaper, and faster development of safety critical systems.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Various areas of our life utilize computing systems such as smart phones, medical devices, and automobile controllers. Consequently, the reliability of computing systems becomes a significant issue to our society and various international standards have been proposed and applied to assure reliability of such systems. For example, avionics domain has DO-178C [20] as a standard for reliable software, automobile domain has a functional safety standard ISO 26262 [11], and medical electrical equipment domain has IEC 60601 [9] as a technical standard for the safety and effectiveness.

However, as computing power increases, the complexity of computing systems also increases rapidly, which causes many challenges to assure the reliability of computing systems. In particular, the size and complexity of software in a computing system has increased quickly. Although software reliability has been studied actively [18], conventional verification and validation (V&V) techniques for software such as testing and model checking [4] have limitations to assure the reliability of complex safety critical computing systems. One reason for this difficulty is that such systems often control highly complex continuous dynamics to interact with physical environments. In addition, since safety critical systems consist of both hardware and software and interact with a physical environment that often behaves non-deterministically (e.g., condition of road surface for automobiles or wind speed for airplanes), we should analyze target hardware and software with its environment together as a stochastic process [22]. However, conventional V&V techniques for software have difficulty analyzing target systems in such contexts.

To improve the reliability of safety critical systems, *statistical model checking (SMC)* techniques [28, 26, 27, 7, 30, 5, 12] have been proposed. SMC techniques approximately compute probabilities for a target system to satisfy given requirements based on randomly sampled execution traces. Thus, SMC techniques can assure the reliability of a complex target system statistically without analyzing the internal logic of a 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 thesis, first, we have 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 [13]: *single sampling plan (SSP) [26], statistical probability ratio test (SPRT) [28], Bayesian hypothesis testing (BHT) [12], and Bayesian interval estimation testing (BIET) [30].* We applied these four SMC techniques to a fault-tolerant fuel control system (FFCS), which is a safety critical system for automobiles. Through the empirical study, we observed that these SMC techniques have different strong points and weak points which may complement one another.

Second, from the above observation, we developed a new hybrid SMC technique which combines SPRT, the fastest SMC technique, and BIET, the most precise SMC technique. This hybrid SMC technique achieves precise verification result fast. Although precise verification result is a top priority for safety critical systems, the time cost of verification cannot be ignored in practice. Thus, we can improve the reliability of safety critical systems more practically by applying our new hybrid SMC technique. To demonstrate the effectiveness and efficiency of this hybrid SMC technique, we have applied this hybrid SMC technique to three safety critical systems in the automobile domain - an automatic transmission control system (ATCS), an anti-lock braking system (ABS), and a fault-tolerant fuel control system (FFCS). Through the experiments, we confirmed that our hybrid SMC technique improves effectiveness and efficiency compared to a single SMC technique.

Finally, we propose an effective methodology to validate the reliability goal of a safety critical system at the early stage of a lifecycle by utilizing statistical model checking (SMC) techniques as a step to help obtain safety certification such as IEC 61508 and ISO 26262. SMC observes the execution behaviors and produces the probability of the system to satisfy given safety functional requirement using statistical methods. Thus, SMC can be used to validate the software reliabilities of complex safety-critical systems. Furthermore, many safety critical system domains such as automobile or avionics have adopted model driven development (MDD). Thus, industries producing safety critical systems can incorporate the proposed reliability validation framework seamlessly and validate the software reliability of a target system safer, cheaper, and faster.

Chapter 2 overviews related four state-of-the-art SMC techniques. Chapter 3 explains the empirical evaluation of four state-of-the-art SMC techniques on FFCS in automobile domain. Chapter 4 discusses issues from the empirical study. Chapter 5 describes a new hybrid SMC algorithm. Chapter 6 explains the three target systems: ATCS, ABS, and FFCS and describes the SMC results by using single SMC techniques and the hybrid technique on ATCS, ABS, and FFCS. Chapter 7 discusses issues from the empirical study of the hybrid SMC technique. Chapter 8 proposes the software reliability validation framework using SMC technique. Chapter 9 explains the case study of the proposed framework on FFCS target system. Chapter 10 concludes this thesis with future work.

Chapter 2. Overview of SMC Techniques

This chapter overviews the general concept of SMC (see Section 2.1), explains a bounded linear temporal logic (BLTL) and a probabilistic BLTL (PBLTL) which are used in SMC techniques (see Section 2.2), and briefly describes several state-of-the-art SMC techniques (see Section 2.3-2.4).

2.1 SMC Framework

SMC computes probabilities for a target model to satisfy given requirement properties based on randomly sampled simulation traces. Figure 2.1 illustrates the overview of SMC. SMC receives a target model $\mathcal{M}$ which is an executable simulation model and *a bounded linear temporal logic (BLTL)* formula ϕ which formally represents a safety functional requirement of a target system. 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. The simulator executes $\mathcal{M}$ and generates a sample execution trace σ_i. The BLTL model checker determines if σ_i satisfies ϕ and passes the result (i.e., success if σ_i satisfies ϕ; failure, otherwise) to the statistical analyzer. The statistical analyzer calculates a probability p that $\mathcal{M}$ satisfies ϕ by collecting the result regarding if σ_i satisfies ϕ. Statistical analyzer generates σ_is repeatedly until the number of successful results of σ_is over the total number of σ_is 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 state explosion problems.

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 2.3) and *estimation testing* (Section 2.4).

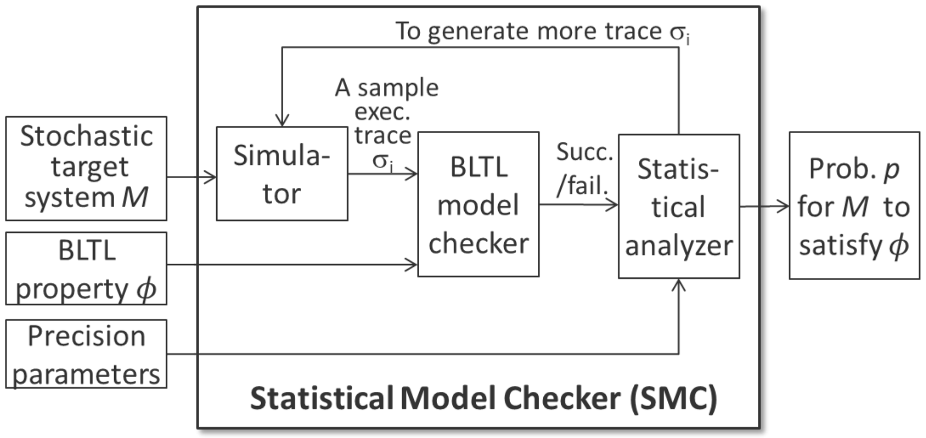


Figure 2.1: Framework of SMC techniques

2.2 Probabilistic Bounded Linear Temporal Logic

We define a syntax and semantics of bounded linear temporal logic (BLTL) [29] and PBLTL [30]. 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 or equal to θ (see [30] for detailed description).

2.3 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 thesis, we utilize the following three hypothesis testing techniques - *single sampling plan (SSP)*, *sequence probability ratio test (SPRT)*, and *Baysian hypothesis testing (BHT)*.

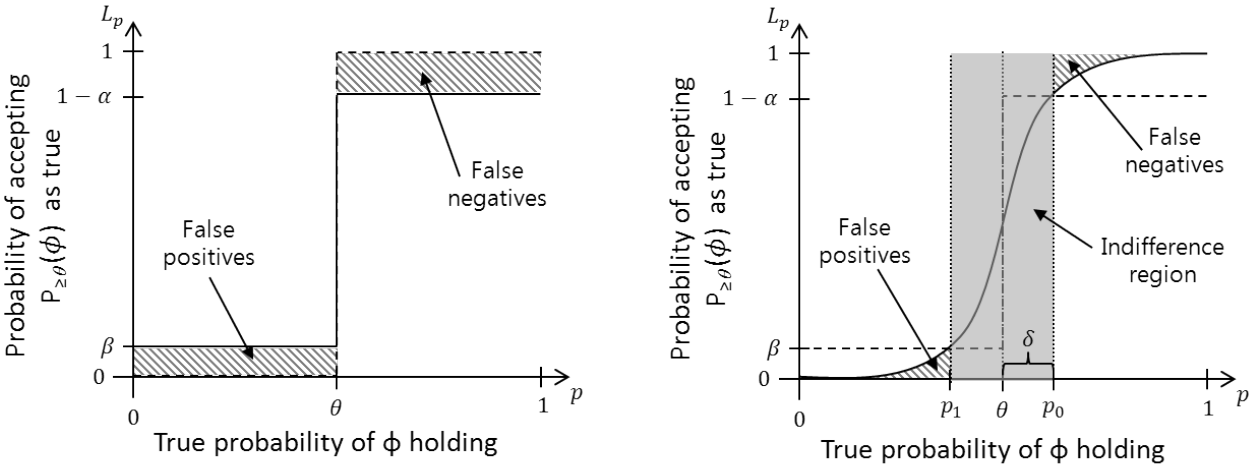


Figure 2.2: Function of probability L_p of accepting the hypothesis $H : p \geq \theta$ (left side) and function of probability L_p of accepting the hypothesis $H_0 : p \geq p_0$ with indifference region (right side).

2.3.1 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 [26] 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 2.2 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 2.2). Therefore, instead 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.

2.3.2 Sequential Probability Ratio Test

Sequential probability ratio test (SPRT) is a hypothesis testing technique introduced by Younes et al. [28]. SPRT [28, 26, 27, 23] determines a number of required sample paths dynamically at runtime. The main goal of SPRT is to decide if $\mathcal{M} \models P_{\geq\theta}[\phi]$ with a small number of sample paths. If another sample path is needed, SPRT generates one more sample path by executing a target system. If the information from generated sample paths is enough, SPRT stops executing the target program and produces an answer regarding $\mathcal{M} \models P_{\geq\theta}[\phi]$. SPRT uses precision parameter inputs *error bounds* α and β, and a half size of *indifference region* δ. The detailed description of SPRT is as follows.

Before building a hypothesis for hypothesis testing of SPRT, we introduce the indifference region. Basically, 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 if H is accepted or not based on the randomly sampled paths. 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. We call α and β as error bounds. The left side of Figure 2.2 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 of $p = \theta$

and $p = \theta - \epsilon$ for arbitrarily small $\epsilon > 0$ for reality. To solve this problem, we introduce an 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 2.2). Therefore, instead 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.

Now, we describe the algorithm of SPRT. First, we obtain a sample path σ_i of a target system by simulating the target system and model-check if the sample path σ_i satisfies the given property ϕ (see Section 2.1). 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 [23, 26]).

Note that SPRT can be imprecise with same indifference region value δ when the threshold θ is close to 1. The reason for the imprecise result of 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 path occurs, which can cause imprecise result. Therefore, δ should be very small when θ is close to 1, which requires large number of samples.

2.3.3 Bayesian Hypothesis Testing (BHT)

BHT [12] 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 [1] (detailed description is found in [12]).

2.4 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.

2.4.1 Bayesian Interval Estimation Testing

Bayesian interval estimation testing (BIET) is an estimation testing based SMC technique. Estimation testing can approximately compute p, the probability that the model $\mathcal{M}$ satisfies the given property ϕ expressed by

[1] T corresponds to the inverse number of error bounds α and β for SSP and SPRT [30].

bounded linear temporal logic (BLTL). With p, we can determine if the probabilistic bounded linear temporal logic (PBLTL) is satisfied. For that purpose, we use a following statistical estimation testing technique.

BIET [30] dynamically determines the number of sample paths for checking the satisfiability of the model $\mathcal{M}$ with the property ϕ during simulation as SPRT does. In Bayes' theorem, we get prior probability using current information first. After obtaining new information, we can obtain posterior probability refining prior probability. BIET uses the Bayes' theorem to determine the number of sample paths of the test.

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'.

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

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

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.

Note that BIET is fast when the estimated probability $\hat{p}$ is close to 0 or 1 [30], whereas BIET is extremely slow (i.e., extremely larger number of samples is required) when $\hat{p}$ is close to $\frac{1}{2}$. With this advantage of BIET, BIET can easily apply the problem for safety critical system since the probability standard of satisfiability for safety critical system should be usually close to 1 or 0.

Chapter 3. Emperical Evaluation of The SMC Techniques on FFCS

In this chapter, we describe FFCS as our target system (Section 3.1) and explain our experiments of applying the four SMC techniques (i.e., SSP, SPRT, BHT, and BIET) to FFCS with precision parameters as independent variables and checking whether FFCS satisfies the given requirement property in PBLTL or not (Section 3.2-3.3).

3.1 Fault-tolerant Fuel Control System

Figure 3.1 is an overall diagram of a fault-tolerant fuel control system (FFCS). FFCS [16] 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: a sensor failure detector & estimator (SFDE), an airflow calculator, and a fuel calculator. The SFDE receives four sensor data as input and generates four sensor data as output and the engine-shut-down command used only when multiple sensor failures occur. The airflow calculator receives four sensors data from the sensor failure detector & estimator (SFDE) component and estimates an airflow value with feedback correction value. The fuel calculator receives the estimated airflow data and the feedback correction data from the airflow calculator component and calculates the fuel rate which keeps an air-fuel ratio optimal.

A requirement property for FFCS is that the fuel rate does not become zero for one second in 100 seconds should be greater than equal to probability 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 [30]:

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

3.2 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 [30] does. The random faults are modeled by three independent Poisson processes with different arrival rates [24]. 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

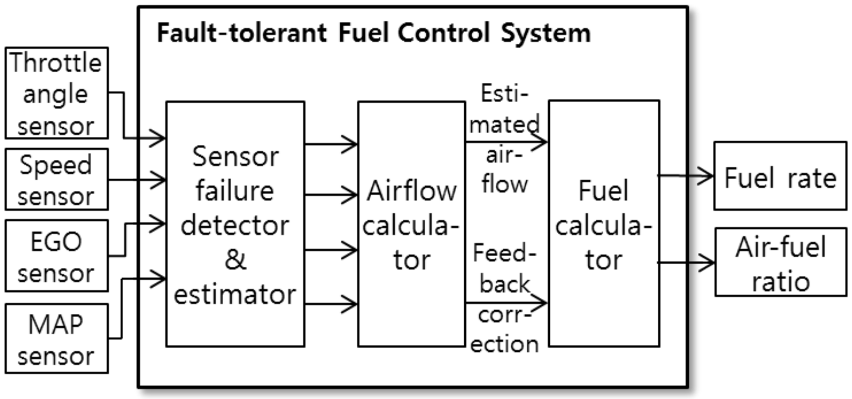


Figure 3.1: Block diagram of FFCS

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 $\alpha' = \beta' = 1$ [1]

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

Table 3.1: 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 3.2: 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

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. [2]

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.

3.3 Experimental Results

Tables 3.1-3.3 describe the experiment results of applying the hypothesis testing techniques to FFCS with fault inter-arrival rate (3,7,8) and $\delta = 0.03$. [3] In these three tables,

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

[3] 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 3.3: 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

Table 3.4: 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

- θ is a threshold of the hypothesis H for SSP, SPRT, and BHT
- 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 3.4 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 3.1-3.4 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 to 0.001 with threshold $\theta = 0.5$, m increases from 255.3 to 1487.8 (Table 3.1). Similar tendencies are observed for SPRT, BHT, and BIET.

3.3.1 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 3.1-3.3.[4] Also, note that n of SPRT and BHT increases exponentially as θ increases from 0.5

[4]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 3.2.

to 0.9, and decreases sharply from 0.9 to 0.99. For example, for SPRT with α=β=0.1 (Table 3.2), 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 3.4), 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.

3.3.2 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 3.4). 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 3.1), 519.2 (α=β=0.001 and θ=0.9) (Table 3.2), and 2648.7 (T=1000 and θ=0.9) (Table 3.3) seconds, respectively. Therefore, we can conclude that statistical model checking can assure reliability of a complex target system at modest cost. [5]

[5] 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.

Chapter 4. Discussion of The Empirical Evaluation

Through the empirical evaluation of the SMC techniques on FFCS, we found three discussion issues which are practicality of SMC in industry (Section 4.1), impacts of precision parameter values (Section 4.2), and comparison of the four SMC techniques (Section 4.3).

4.1 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 3.3). 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 3.3). [1]

4.2 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 3.2), 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 3.3). 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

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

Table 4.1: 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

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 3.3), we can conclude that the verification result with T=10 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.

4.3 Comparison of the SMC techniques

Table 4.1 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. The precision of BIET is higher than the other techniques by the *law of large numbers* [21], 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 4.1, 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.

Chapter 5. Hybrid SMC Algorithm

We develop a hybrid SMC technique to improve efficiency and effectiveness by combining SPRT whose verification speed is fast (i.e., small number of samples is required) and BIET whose verification precision is high (i.e., the number of false positive and false negative results is small) [13]. Figure 5.1 describes how the hybrid SMC technique checks if a target system model $\mathcal{M}$ satisfies a property ϕ in BLTL for a probability threshold θ [1] with precision parameters par_S for SPRT and par_B for BIET. The algorithm first applies SPRT multiple times with dynamically increasing probability threshold θ_{SPRT} until a verification result is 'reject' (lines 15–18) or θ_{SPRT} becomes larger than or equal to a threshold th_{S2B} where $0.5 < th_{S2B} \leq \theta$ (lines 5–20). If θ_{SPRT} becomes larger than or equal to th_{S2B}, the algorithm applies BIET to obtain a precise verification result (lines 21–34).

The detail of the algorithm is as follows. First, the algorithm calls $SPRT()$ m_S times (lines 6–10), which applies SPRT to $\mathcal{M}$ with regard to ϕ and θ_{SPRT} with par_S (line 8). A result of $SPRT()$ is 'accept' (i.e., 1) or 'reject' (i.e., 0). After m_S trials of $SPRT()$, the algorithm calculates an average accept decision value $accept_{avg}$ over the m_S trials (line 11). If $accept_{avg}$ is less than a user-given accept decision threshold th_{acpt}, the algorithm decides that the verification result of $\mathcal{M} \models P_{\geq\theta}(\phi)$ is 'reject' (line 16) and terminates (line 18). Otherwise (i.e., $accept_{avg} \geq th_{acpt}$), the algorithm increases θ_{SPRT} from the initial value 0.5 (line 3) to 0.75, 0.875, 0.9375 and so on (line 14) until θ_{SPRT} becomes larger than or equal to th_{S2B} through the while loop in lines 5-20.

If θ_{SPRT} becomes larger than or equal to a user-given probability threshold th_{S2B} for applying BIET, the algorithm calls $BIET()$ for m_B times (lines 23–27), which applies BIET to $\mathcal{M}$ for ϕ with precision parameters par_B (line 25). Based on the estimated probability p obtained from $BIET()$, the algorithm calculates an average estimated probability p_{avg} over the m_B trials (line 28). If p_{avg} is greater than or equal to θ, then the algorithm decides that the verification result is 'accept' (lines 29–30); 'reject', otherwise (lines 31–32).

Note that the hybrid SMC algorithm can save a large amount of time cost compared to BIET, if a probability for $\mathcal{M}$ to satisfy ϕ is far from a given probability threshold θ. For example, if the probability is less than 0.5, the algorithm terminates after executing $SPRT()$ only once without executing $BIET()$ whose time cost is very high (see Table 3.4). The algorithm executes $BIET()$ if the probability is close to θ (which is usually close to 1 for requirement properties of safety critical systems), which is necessary since SPRT becomes imprecise when θ is close to 1 (Section 2.3.2).

[1]We assume that θ is close to 1, since we develop a hybrid SMC algorithm for safety critical systems whose reliability criteria are very high and, thus, requirement properties are given with high threshold values.

```
Input:
$\mathcal{M}$: a model
$\phi$: BLTL property
$\theta$: probability threshold of $\mathcal{M} \models \phi$
$par_S$: precision parameters of SPRT, $par_B$: precision parameters of BIET
$th_{acpt}$: accept decision threshold over [0,1]
$th_{S2B}$: probability threshold to change from SPRT to BIET
$m_S$: a number of trials for SPRT, $m_B$: a number of trials for BIET
Output:
$answer$: result of $\mathcal{M} \models P_{\geq\theta}(\phi)$
$p_{avg}$: average estimated probability of $\mathcal{M} \models \phi$ by BIET if BIET is applied; N/A otherwise
1  $SMC_{hyb}(\mathcal{M}, \phi, \theta, par_S, par_B, th_{acpt}, th_{S2B}, m_S, m_B)\{$
2  $accept_{sum} = 0$; // sum of accept decisions by SPRT
3  $\theta_{SPRT} = 0.5$; // initial probability threshold for SPRT
4  // SPRT for fast verification
5  while $\theta_{SPRT} < th_{S2B}$ do
6      for $i = 1 \rightarrow m_S$ do
7          $accept = SPRT(\mathcal{M}, \phi, \theta_{SPRT}, par_S)$; // Checks $\mathcal{M} \models P_{\geq\theta_{SPRT}}(\phi)$ using SPRT
8          Add $accept$ to $accept_{sum}$;
9      end
10     $accept_{avg} = accept_{sum}/m_S$;
11     if $accept_{avg} \geq th_{acpt}$ then
12         $\theta_{SPRT} = \theta_{SPRT} + (1 - \theta_{SPRT})/2$; // next probability threshold for SPRT
13     else
14         $answer = `reject'$;
15         $p_{avg}$ = N/A;
16         return $answer$ and $p_{avg}$;
17     end
18 end
19 // BIET for precise verification
20 $p_{sum} = 0$; // sum of estimated probabilities by BIET
21 for $i = 1 \rightarrow m_B$ do
22     $p = BIET(\mathcal{M}, \phi, par_B)$; // Checks $\mathcal{M} \models \phi$ using BIET
23     Add $p$ to $p_{sum}$;
24 end
25 $p_{avg} = p_{sum}/m_B$;
26 if $p_{avg} \geq \theta$ then
27     $answer = `accept'$;
28 else
29     $answer = `reject'$;
30 end
31 return $answer$ and $p_{avg}$;
32 $\}$
```

Figure 5.1: Hybrid SMC algorithm

Chapter 6. Experimental Study of Hybrid SMC Technique

We have applied SPRT, BIET, and the hybrid SMC technique to ATCS, ABS, and FFCS with precision parameters as independent variables to check if these target systems satisfy the given requirement properties in PBLTL. In addition, we have compared the results of the hybrid SMC technique with the results of SPRT and BIET. We used Simulink/stateflow models of the three systems included in the Matlab R2010a example directory.

6.1 Target Safety Critical Systems

This section presents an overview of the following three safety critical systems in automobile domain:

- Automatic transmission control system (ATCS) [17]
- Anti-lock braking system (ABS) [2]
- Fault-tolerant fuel control system (FFCS) [16]

We selected these systems as target systems to apply SPRT, BIET, and the hybrid statistical model checking (SMC) technique (Chapter 5) for the following reasons:

- These three automobile systems [16, 2, 17] are safety critical systems whose reliability is very important. Many researchers are working to address the reliability issues on safety critical systems [3, 18, 25].
- The three automobile systems are complex real-world applications, not a toy example such as ones in probabilistic symbolic model checker (PRISM) [15] benchmarks.
- Simulink/stateflow models of the three automobile systems are publicly available in Matlab R2010a. Thus, it is convenient to build a prototype tool for the SMC techniques by using a Simulink/stateflow simulator.

6.1.1 Automatic Transmission Control System

An automatic transmission control system (ATCS) changes an engine gear automatically to drive smoothly. A main task of ATCS is to select a proper engine gear. As described in Figure 6.1, ATCS receives inputs regarding car speed, throttle, brake pressure (and engine RPM as a feedback) and calculates an engine RPM and a gear state. ATCS consists of a torque converter and a transmission control unit. The torque converter calculates an impeller torque value to deliver power to control the engine RPM based on the engine RPM and the gear state (i.e., if the impeller torque increases/decreases, the engine RPM increases/decreases). With the sensor inputs on car speed,

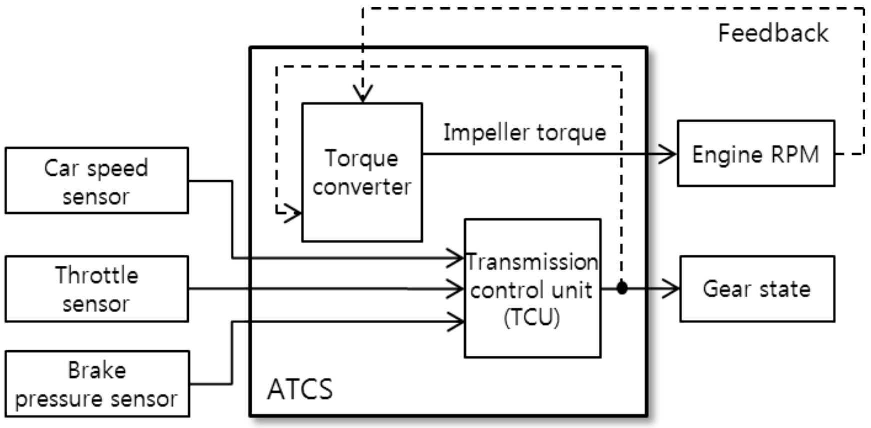


Figure 6.1: Block diagram of ATCS

throttle, and brake pressure, transmission control unit (TCU) selects a proper gear. Based on throttle and brake pressure values, TCU calculates a up-threshold and a down-threshold of a car speed. If a current car speed is greater than the up-threshold or less than the down-threshold, TCU changes the engine gear to keep the engine RPM in safe range.

The size and complexity of the Simulink/stateflow ATCS model in terms of the Halstead metrics [6] are described in Table 6.1. 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 ATCS model. The automatically generated C code from the model has 2353 LOC in 71 functions.

A requirement property for ATCS is that the engine RPM is less than 6000 for 30 seconds [1] should be greater than or equal to probability θ. The property is important in real world, because if the engine RPM is constantly over 6000, the engine becomes overheated and can be damaged. The property can be expressed in PBLTL as follows:

$$P_{\geq\theta}[G^{30}(engineRPM < 6000)]$$

6.1.2 Anti-lock Braking System

An anti-lock braking system (ABS) is a safety system that repeatedly increases and decreases the brake pressure to allow the wheels of a car to interact with the road surface continuously as directed by a driver while braking. Thus, ABS can prevent the wheels from locking up and avoid skidding, which can enhance the safety of driving by improving vehicle control and decreasing stopping distances. As described in Figure 6.2, ABS has the following three sensors: a car speed sensor, a wheel speed sensor, and a brake pedal sensor. ABS receives data from these sensors and generates the brake pressure and slip as outputs, where slip indicates how properly a wheel of a car is controlled. ABS consists of a bang-bang controller and a hydraulic control unit. The bang-bang

[1]We set the time duration to monitor as 30 seconds, since a default simulation time of the Simulink model of ATCS included in Matlab R2010a is 30 seconds.

Table 6.1: Size and complexity of the Simulink models of ATCS, ABS, and FFCS in Halstead metrics

Target system	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 × V)
ATCS	31	46	27	39	77	66	465.4	15.9	7410.9
ABS	27	36	19	36	63	55	364.2	9.5	3460.1
FFCS	65	111	35	94	176	129	1234.0	20.7	25500.0

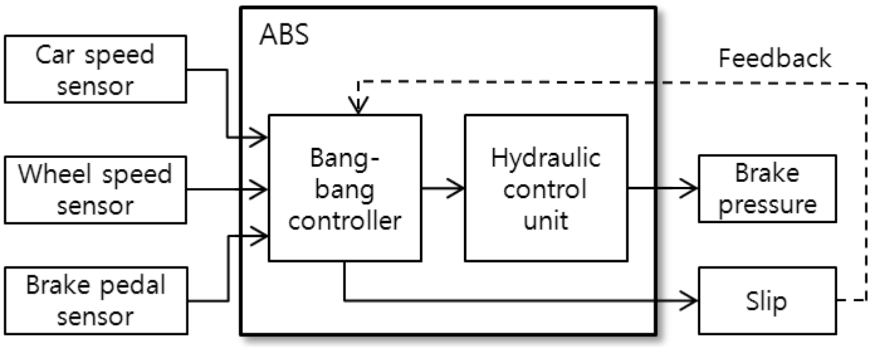


Figure 6.2: Block diagram of ABS

controller receives data from the three input sensors and commands the hydraulic control unit to increase/decrease the brake pressure. In addition, when the brake pedal is pressed, the bang-bang controller calculates slip as follows:

$$slip = 1 - \frac{wheelspeed}{carspeed}$$

When the wheel speed is equal to the car speed, slip becomes zero. When the wheel speed is zero (i.e., the wheel is locked), slip becomes one, which means that the driver loses control of the car. There is an ideal slip value (which is 0.2) that maximizes the adhesion between the wheel and the road and minimizes the stopping distance with available friction. The bang-bang controller tries to adjust slip close to the ideal slip value by controlling the hydraulic control unit.

The size and complexity of the Simulink/stateflow ABS model in terms of the Halstead metrics are described in Table 6.1. The automatically generated C code from the model has 3443 LOC in 27 functions.

A requirement property for ABS is that for 17 seconds [2], when the brake pedal is pressed and the car speed is greater than 5 m/s, slip is less than or equal to 0.9, should be larger than or equal to probability θ. The property is important in real world, because if slip becomes close to 1 when a car is driving, the wheel can be locked and a driver loses control of the car. The property can be expressed in PBLTL as follows:

$$P_{\geq\theta}[G^{17}((brakepressed \wedge carspeed > 5) \rightarrow slip \leq 0.9)]$$

[2]We set the time duration to monitor as 17 seconds, since a default simulation time of the Simulink model of ABS included in Matlab R2010a is 17 seconds.

6.1.3 Fault-tolerant Fuel Control System

Refer to Section 3.1.

6.2 Experiment Setup

6.2.1 Environment Setup

We used the input value generation modules provided in the Simulink/stateflow models of FFCS, ATCS, and ABS without modification. In addition, we built the stochastic environments for the three automobile systems as follows:

- **ATCS**: we built a stochastic environment to ATCS by modeling a random delay to transfer the engine RPM value from the engine to the torque converter. [3] This random delay is modeled by exponential distribution [14]. We selected a 'passing maneuver' scenario from the options of the ATCS model, which simulates a situation that a driver opens the throttle 100% after 15 seconds. We utilize the following four delay rates (i.e., mean delay times of transmission in seconds) $\lambda \in \{0.01, 0.02, 0.03, 0.04\}$.

- **ABS**: we built a stochastic environment of ABS that generates random delay to the command from the bang-bang controller to the hydraulic control unit. [4] The random delay of the command is modeled by exponential distribution [14]. We use a model of ABS representing a single wheel, which can be duplicated multiple times to create a model for a multi-wheel vehicle. We utilize the following four delay rates (in seconds) $\lambda \in \{0.001, 0.003, 0.005, 0.007\}$.

- **FFCS**: we built a stochastic environment model for FFCS that generates random faults at the EGO, MAP, and speed sensors as Zuliani et al. [30] did. The random faults are modeled by three independent Poisson processes with different arrival rates [24]. We assume one fault event remains for one second. When a fault event occurs in a sensor, FFCS remains in a failure mode in 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).

6.2.2 Precision Parameter Setup

We use the following precision parameters for SPRT and BIET:

- SPRT:

[3]This random delay is a real factor, not an artificial one. ATCS has an electronic circuit to deliver data from one sub-component to another and the data transfer can be delayed non-deterministically due to non-deterministic scheduling and bus contention among multiple sub-component.

[4]This random delay is a real factor for the similar reason of the one in ATCS.

Table 6.2: Experiment result of SPRT for ATCS with $\lambda = 0.03$ and $\delta = 0.03$ for the five trials

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	110	1.0	69.9	215	1.0	143.9	343	0.0	221.8	18	1.0	12.6
0.01	270	1.0	171.0	375	1.0	301.1	410	0.0	347.1	41	1.0	27.1
0.001	395	1.0	249.0	563	1.0	361.1	985	0.0	636.7	45	1.0	30.2

Table 6.3: Experiment result of BIET for ATCS with $\lambda = 0.03$ for the five trials

δ'	interval coverage c								
	0.9			0.99			0.999		
	n	$\hat{p}$	$time$	n	$\hat{p}$	$time$	n	$\hat{p}$	$time$
0.05	630	0.8594	416.6	1550	0.8654	1011.9	2665	0.8636	1753.2
0.03	1845	0.8544	1208.6	3340	0.9000	2181.1	6475	0.8805	4356.5
0.01	14150	0.8810	9551.8	36540	0.8740	26281.2	58870	0.8762	42945.1

  - a half-size of indifference region $\delta \in \{0.01, 0.03, 0.05\}$
  - error bounds $\alpha, \beta \in \{0.1, 0.01, 0.001\}$

- BIET [5]:
  - interval coverage $c \in \{0.9, 0.99, 0.999\}$
  - a half-size of estimation interval $\delta' \in \{0.01, 0.03, 0.05\}$
  - parameters of Beta prior $\alpha' = \beta' = 1$ (since we assume the prior density to be a uniform density over $(0, 1)$)

We performed each experiment five times to obtain average verification result over $[0, 1]$ regarding if the hypothesis H_0 is accepted where H_0: a probability for $\mathcal{M}$ to satisfy ϕ is greater than or equal to $\theta + \delta$. For the experiments, we used $\theta \in \{0.5, 0.7, 0.9, 0.99\}$. In addition, we measured the total verification time and total number of samples for each experiment.

For the hybrid SMC technique, we set θ=0.99. This is because the hybrid SMC technique targets safety critical systems which require high reliability, which can be specified with PBLTL with high θ values. We use the following precision parameters which are similar to those of the SPRT and BIET experiments:

- precision parameters for SPRT par_S: $\delta \in \{0.01, 0.03, 0.05\}$, $\alpha, \beta \in \{0.1, 0.01, 0.001\}$.
- precision parameters for BIET par_B: $c \in \{0.9, 0.99, 0.999\}$, $\delta' \in \{0.01, 0.03, 0.05\}$, $\alpha' = \beta' = 1$.

[5] Our parameters are similar to those of Zuliani et al. [30], where they use interval coverage c as 0.99 and 0.999 and half-size of estimation interval δ as 0.01 and 0.05. To identify tendency of the experimental results more, we used more parameters.

- threshold for accept decision over $[0, 1]$ th_{acc}=0.5
- the probability threshold to apply BIET instead of SPRT th_{S2B}=0.95
- the number of trials for SPRT $m_S = 5$
- the number of trials for BIET $m_B = 5$

6.2.3 Experiment Platform

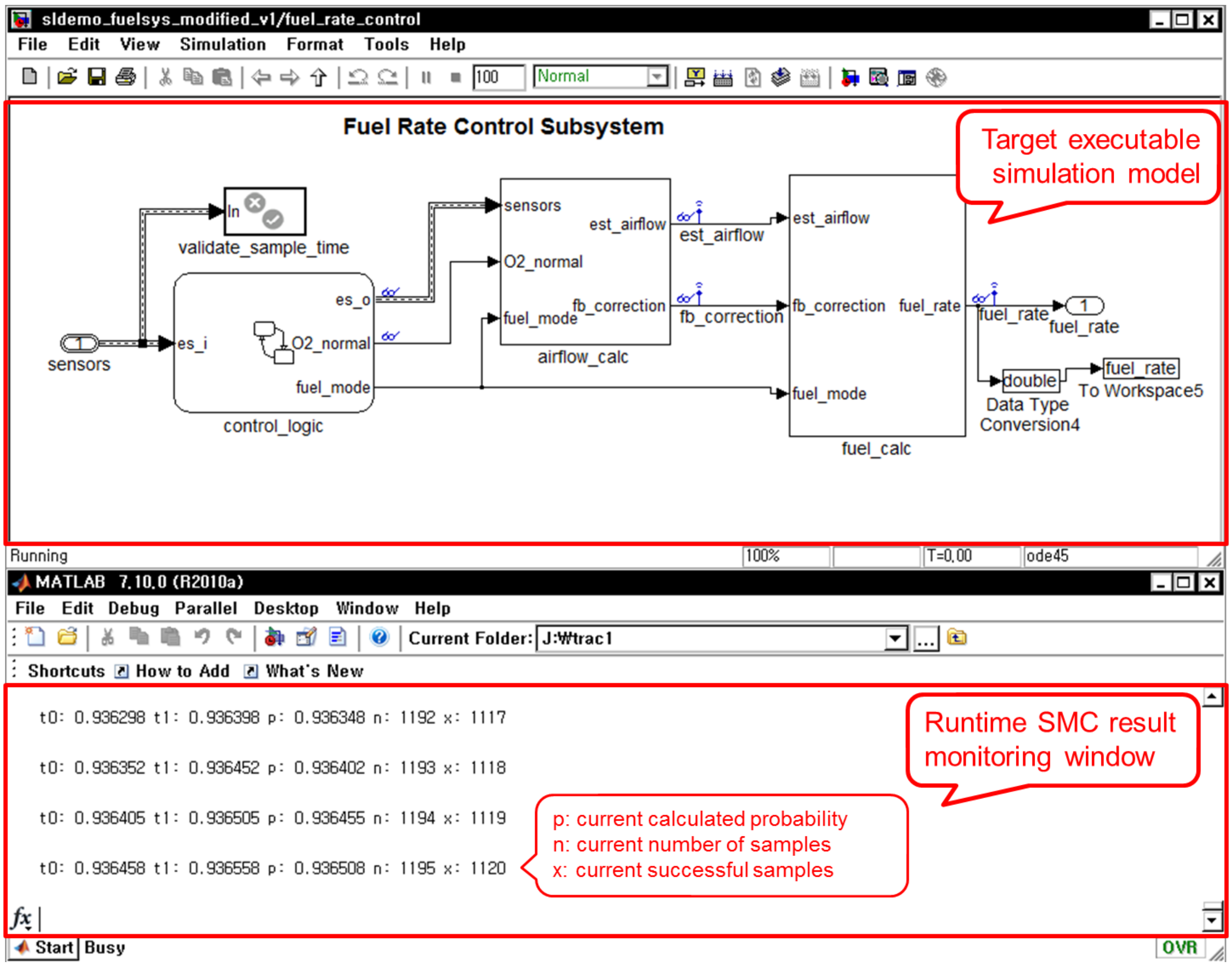
Figure 6.3 shows the overall snapshot of running FFCS simulation (see the upper window) together with SMC (see the lower command window). At the upper window of Figure 6.3, the three component blocks correspond to the components of FFCS in Figure 2.1 (for example, the `control_logic` block corresponds to Sensor failure detector component). All four sensor inputs are represented by a "sensors" block and the fuel rate output is represented by the `fuel_rate` block. At In the lower command-line window, the SMC tool displays variable values related necessary to calculate the probability for FFCS to satisfy ϕ. Specifically, p is a calculated probability and n is a the total number of sample simulation traces so far. In addition, x is a number of successful sample traces so far. For example, the last line of the low window indicates that 1195 sample traces have been generated until nowto this point and 1120 traces among them satisfy ϕ. Also the same line indicates that the probability for FFCS to satisfy ϕ is calculated as 0.936508 currently (note that the snapshot of Figure 6.3 shows on-going SMC process, not the final result).

We built a statistical model checker as a Matlab module, which executes the Simulink/stateflow models for FFCS, ATCS, and ABS and monitors inputs and outputs of the models to check if ϕ is satisfied on a current sample path. After each execution of the models, the SMC module calculates a required number of samples dynamically based on the precision parameters and the number of success/fail samples generated so far. If a number of the generated samples reaches the required number, the SMC module generates a verification result. The SMC module for SPRT is around 80 lines long. The SMC module for BIET is around 70 lines long. The hybrid SMC module is around 200 lines long. We used Matlab R2010a for the experiments. All experiments were performed on 64 bit Windows 7 Professional equipped with a 3 GHz Intel processor and 16 gigabytes of memory.

6.3 Results of SPRT and BIET

Tables 6.2 and 6.3 describe the experiment results of applying SPRT with $\delta = 0.03$ and BIET to ATCS respectively when the delay rate λ=0.03. [6] In Tables 6.2 and 6.3, n is a total number of required sample execution paths for the five trials and $time$ is total verification time taken for the five trials in seconds. $acpt$ in Table 6.2 is

[6]Full experiment data of applying SPRT and BIET to ATCS, ABS, and FFCS is available at `http://pswlab.kaist.ac.kr/data/issre2012-expr-results.zip`


sldemo_fuelsys_modified_v1/fuel_rate_control
File Edit View Simulation Format Tools Help
100
Normal
Fuel Rate Control Subsystem
Target executable simulation model
validate_sample_time
sensors
es_i
es_o
O2_normal
fuel_mode
control_logic
est_airflow
fb_correction
airflow_calc
fuel_rate
fuel_calc
double
Data Type Conversion4
To Workspace5
Running
100%
T=0.00
ode45
MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: J:₩trac1
Shortcuts How to Add What's New
Runtime SMC result monitoring window
p: current calculated probability
n: current number of samples
x: current successful samples
Start Busy
OVR

```
t0: 0.936298 t1: 0.936398 p: 0.936348 n: 1192 x: 1117

t0: 0.936352 t1: 0.936452 p: 0.936402 n: 1193 x: 1118

t0: 0.936405 t1: 0.936505 p: 0.936455 n: 1194 x: 1119

t0: 0.936458 t1: 0.936558 p: 0.936508 n: 1195 x: 1120
```

Figure 6.3: Screenshot of the SMC experiment on FFCS

an average result over $[0, 1]$ regarding the hypothesis H_0 where 0 is 'reject' and 1 is 'accept'. $\hat{p}$ in Table 6.3 is an estimated probability for $\mathcal{M} \models \phi$.

Table 6.2 shows that the probability for ATCS with λ=0.03 and $\delta = 0.03$ to satisfy the requirement property ϕ (=$G^{30}(engineRPM < 6000)$) is between 0.7 and 0.9. This is because $acpt$s are 1.0 when $\theta \leq 0.7$ while $acpt$s are 0.0 when $\theta = 0.9$ in Table 6.2 (the verification result of SPRT with a high θ value like 0.99 should not be trusted due to the characteristics of SPRT [28]).

In addition, we can conclude that the probability is close to 0.9, since n of SPRT increases as θ increases from 0.5 to 0.9 and decreases sharply from 0.9 to 0.99. For example, Table 6.2 shows that n becomes 110, 215, 343, and 18 as θ becomes 0.5, 0.7, 0.9, and 0.99 with α=β=0.1. This tendency of n indicates that the true probability for ATCS with λ=0.03 to satisfy ϕ is close to 0.9, since SPRT requires a large number of sample paths to check a given hypothesis H_0 if a true probability is close to θ [28]. Furthermore, the verification result of BIET coincides with that of SPRT, since Table 6.3 shows that the estimated probability $\hat{p}$ is between 0.8544 (with $c = 0.9$ and $\delta' = 0.03$) and 0.9000 (with $c = 0.99$ and $\delta' = 0.03$).

For the verification speed, Tables 6.2 and 6.3 show that SPRT is much faster than BIET. For example, the

Table 6.4: Experiment result of the hybrid SMC for ATCS with $\theta = 0.99$, $\delta = 0.03$, $\delta' = 0.01$, $c = 0.99$

α, β	delay rate λ from engine to torque convertor															
	0.01				0.02				0.03				0.04			
	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$
0.1	1710	0.9956	1	1256.1	1710	0.9956	1	1173.7	1066	N/A	0	698.9	1334	N/A	0	858.9
0.01	2315	0.9956	1	1740.8	2315	0.9956	1	1642.6	4795	N/A	0	3081.9	2946	N/A	0	1884.6
0.001	2905	0.9956	1	2320.2	2905	0.9956	1	2102.7	7804	N/A	0	6020.4	3833	N/A	0	2952.4

Table 6.5: Experiment result of hybrid SMC for ABS with $\theta = 0.99$, $\delta = 0.03$, $\delta' = 0.01$, $c = 0.99$

α, β	delay rate λ from bang-bang controller to hydraulic control unit															
	0.001				0.003				0.005				0.07			
	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$
0.1	1814	0.9953	1	986.5	6511	0.9826	0	2905.9	8247	0.9773	0	3854.4	952	N/A	0	382.4
0.01	2417	0.9953	1	1344.8	8006	0.9806	0	3619.4	9151	0.9770	0	4290.1	2238	N/A	0	890.2
0.001	3179	0.9950	1	1815.3	8541	0.9810	0	3906.1	9326	0.9791	0	4334.0	3684	N/A	0	1465.5

Table 6.6: Experiment result of hybrid SMC for FFCS with $\theta = 0.99$, $\delta = 0.03$, $\delta' = 0.01$, $c = 0.99$

α, β	sensor fault rates															
	$(3, 7, 8)$				$(10, 8, 9)$				$(20, 10, 20)$				$(30, 30, 30)$			
	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$	n	$\hat{p}$	$acpt$	$time$
0.1	1299	N/A	0	3359.6	14442	0.9575	0	36399.3	3180	0.9920	1	7990.0	2121	0.9944	1	5362.0
0.01	5369	N/A	0	13893.4	14130	0.9620	0	35894.1	4651	0.9906	1	11786.0	3747	0.9926	1	9556.4
0.001	7320	N/A	0	19059.9	16010	0.9592	0	41014.6	5809	0.9895	0	14792.1	3512	0.9939	1	9017.2

maximum time spent by SPRT in Table 6.2 is 636.7 seconds with $\theta = 0.9$ and α=β=0.001, which is less than time costs of BIET in Table 6.3 except when BIET is applied with low precision parameters $\delta' = 0.05$ and $c = 0.9$ (416.6 seconds).

Thus, if a given PBLTL formula has a high θ value like 0.99, it is a good idea to apply SPRT first with low θ values (SPRT result with high θ value should not be trusted) in hope of eliminating the need to apply BIET. For example, suppose that we should check $P_{\geq\theta}[G^{30}(engineRPM < 6000)]$ for ATCS with λ=0.03 and $\theta = 0.99$. With α=β=0.1, SPRT takes 435.6 seconds in total (=69.9+143.9+221.8) to conclude that ATCS does not satisfy the given PBLTL formula with $\theta = 0.99$ by checking cases with θ as 0.5, 0.7, and 0.9 in order (Table 6.2); the verification result with $\theta = 0.9$ is 'reject', which consequently makes the result with $\theta = 0.99$ as 'reject'. However, if we apply BIET, we will obtain the same verification result with higher time cost except a case with $\delta' = 0.05$ and $c = 0.9$ (416.6 seconds (Table 6.3)). The hybrid SMC technique (Figure 5.1) is developed to utilize this observation for precise and fast verification.

6.4 Results of the Hybrid SMC Technique

Tables 6.4-6.6 present the experiment results of the hybrid SMC technique on ATCS, ABS, and FFCS for $\theta = 0.99$ with $\delta = 0.03$, $\delta' = 0.01$, and $c = 0.99$, respectively. n is a total number of sample paths required by SPRT and BIET in the hybrid algorithm for each experiment. $\hat{p}$ is an estimated probability obtained by BIET for each experiment. If BIET is not applied because SPRT rejects a hypothesis H_0 before reaching th_{S2B}, then $\hat{p}$ is N/A. $acpt$ is a result over [0,1] regarding the hypothesis H_0 where 0 is 'reject' and 1 is 'accept'. $time$ is total verification time taken for each experiment in seconds.

6.4.1 Verification Results

For ATCS, Table 6.4 shows that the corresponding hypothesis H_0 with $\theta = 0.99$ is accepted for two delay rates $\lambda \in \{0.01, 0.02\}$ (i.e., $\mathcal{M} \models P_{\geq\theta}[G^{30}(engineRPM < 6000)]$ and rejected for delay rates $\lambda \in \{0.03, 0.04\}$. For the experiments with $\lambda \in \{0.03, 0.04\}$, SPRT rejected H_0 and $BIET$ was not applied; thus, corresponding $\hat{p}$s are marked as 'N/A'. This result coincides with the results of SPRT and BIET, since SPRT concludes that ATCS with λ=0.03 does not satisfy the PBLTL formula with $\theta = 0.9$ (i.e., $acpt$s are all 0.0 in Table 6.2) and BIET concludes that the probability for ATCS with λ=0.03 to satisfy $G^{30}(engineRPM < 6000)$ is between 0.8544 and 0.9000 (Section 6.3).

An interpretation of this result is that ATCS may not operate correctly if an engine RPM value is transferred from the engine to the torque converter with long delay (i.e., delay rate λ in exponential distribution is larger than or equal to 0.03 seconds), since long delay of the data transfer can prevent ATCS from operating promptly. In addition, we can obtain a practical implication that, to achieve required high reliability specified by the PBLTL formula with $\theta = 0.99$, ATCS should use a data-transfer component that transfers data from the engine to the torque converter with delay rate $\lambda \leq 0.02$ or revise the ATCS design to satisfy the PBLTL formula with $\theta = 0.99$ even with long delay of the data transfer.

Similarly, for ABS, Table 6.5 shows that the corresponding hypothesis H_0 with $\theta = 0.99$ is accepted for delay rate λ=0.001 (i.e., $\mathcal{M} \models P_{\geq\theta}[G^{17}((brakepressed \wedge carspeed > 5) \rightarrow slip \leq 0.9)]$), and is rejected for larger delay rates. For FFCS, Table 6.6 shows that the corresponding hypothesis H_0 with $\theta = 0.99$ is accepted for fault ratios (20,10,20) (except α=β=0.001) and (30,30,30) (i.e., $\mathcal{M} \models P_{\geq\theta}[\neg(F^{100}G^{1}(fuelrate = 0))]$), and is rejected for more frequent fault ratios (3,7,8) and (10,8,9).

6.4.2 Verification Speeds

The hybrid SMC technique shows an order of magnitude faster verification speed compared to BIET for the experiments where the probability for $\mathcal{M} \models \phi$ is less than th_{S2B}. [7] For example, for ATCS with λ=0.03, the hybrid technique spent 698.9 seconds (with α=β=0.1, δ=0.03, $\delta' = 0.01$, and c=0.99) to 6020.4 seconds (with α=β=0.001, $\delta = 0.03$, $\delta' = 0.01$, and c=0.99) (Table 6.4), while BIET spent 26281.2 seconds for the same precision parameters (i.e., $\delta' = 0.01, c = 0.99$) (Table 6.3). The hybrid technique is much faster than BIET for ATCS with λ=0.03, since SPRT of the hybrid technique concludes that ATCS with λ=0.03 does not satisfy the PBLTL formula with $\theta_{SPRT} = 0.9375$. Since $\theta_{SPRT} = 0.9375 < th_{S2B} = 0.95$, the hybrid technique does not

[7] Comparison between the verification speed of the hybrid technique and that of SPRT is not meaningful, since SPRT result is imprecise for a large θ value like 0.99.

apply BIET and conclude that ATCS with $\lambda = 0.03$ does not satisfy the given PBLTL formula with $\theta = 0.99$. As BIET takes an order of magnitude larger time cost than SPRT (Tables 6.2–6.3), the hybrid technique can reduce a large amount of time cost by removing the time cost of BIET.

However, for the experiments where the probability for $\mathcal{M} \models \phi$ is larger than th_{S2B}, the hybrid technique shows slower verification speed compared to BIET. For example, for ATCS with λ=0.02, the hybrid technique spent 1173.7 seconds (with α=β=0.1, δ=0.03, δ'=0.01, and c=0.99) to 2102.7 seconds (with α=β=0.001, δ=0.03, δ'=0.01, and c=0.99) (Table 6.4), while BIET spent 820.1 seconds for the same precision parameters (i.e., δ'=0.01 and c=0.99) (see `http://pswlab.kaist.ac.kr/data/issre2012-expr-results.zip`). This larger time cost of the hybrid technique is due to the additional applications of SPRT for $\theta_{SPRT} \in \{0.5, 0.75, 0.875, 0.9375\}$.

For ABS and FFCS, we make similar observations to the experiments for ATCS. For the cases where the probability for $\mathcal{M} \models \phi$ is less than th_{S2B}, the hybrid technique is much faster than BIET. For the other cases, the hybrid technique is slower than BIET.

Chapter 7. Discussion of Hybrid SMC Technique

Through the case studies of the proposed hybrid SMC technique on ABS, ATCS, and FFCS, we found three discussion issues such as effectiveness and efficiency of hybrid SMC technique (Section 7.1), independence between complexity of target system and SMC cost (Section 7.2), and usefulness of SMC techniques to obtain a safety certificate (Section 7.3).

7.1 Effective and Efficient Hybrid SMC Technique

Through the empirical evaluation of the hybrid statistical model checking technique on ATCS, ABS, and FFCS, we found that the hybrid technique is faster and more accurate than a single SMC technique (Section 6.4). This improvement is achieved by utilizing the different advantages of SPRT and BIET selectively, namely fast verification speed of SPRT and precise verification result of BIET (Section 6.3).

The hybrid SMC technique applies SPRT and BIET selectively, because significance of verification speed and that of verification precision vary depending on a probability p for $\mathcal{M}$ to satisfy a requirement property ϕ. Suppose that if p is distant from θ (e.g., $|\theta - p| \geq 0.1$), precision may not be very important, because small error (e.g. +0.01 or -0.01) in an estimated probability does not affect an accept/reject decision on H_0. In this case, the hybrid technique applies SPRT for fast verification without much concern for precision. If p is close to θ, however, precision becomes important, because a small error (e.g. +0.01 or -0.01) may affect an accept/reject decision on H_0 easily. In this case, the hybrid technique applies BIET for precise verification result.

Since we are targeting safety critical systems where PBLTL requirements often have θ values close to 1 (e.g., 0.99 or 0.999) for high reliability, the hybrid SMC technique can apply SPRT for relatively low θ_{SPRT} values first (e.g., 0.5, 0.75, etc.) in hope to conclude a 'reject' decision fast with little concern for precision (a case where p is distant from θ). If SPRT concludes 'accept' decisions for the relatively low θ_{SPRT}s (i.e., a case where p is close to θ), the hybrid SMC algorithm applies BIET for precise verification result. Therefore, the hybrid SMC technique can produce a final verification result (i.e., accept/reject of H_0) fast and precisely.

Although precise verification result is of the highest priority for SMC, we cannot ignore the time cost. Since the available project time in industry is always limited, the efficiency of verification techniques is of important concern, too. For example, ISO-26262 [11] requires that the reliability of the safety critical system components should be higher than 99.999% level. To obtain such high reliability through SMC, the time cost of SMC will be significantly large (it can take several days to several weeks). Therefore, verification speed is also a critical issue as well as precision and our hybrid SMC technique can be useful for practical application of SMC techniques to improve the reliability of safety critical systems.

7.2 Independence between Complexity of Target System and SMC Cost

We found that the complexity of a target system does not affect the cost of the hybrid SMC technique. For example, although FFCS is more complex than the other systems (e.g., program effort E of FFCS is 25500.0, while those of ATCS and ABS are 7410.9 and 3460.1 respectively (Table 6.1)), for similar estimated probability $\hat{p}$ with the same precision parameters, a number of sample execution paths n for FFCS is similar to those for ATCS and ABS. [1] For the five experiments with α=β=0.1 in Tables 6.4-6.6 whose $\hat{p} > 0.99$, the numbers of execution paths ns for these experiments are similar.

- ATCS with λ=0.01 or 0.02: $\hat{p} = 0.9956$ and $n = 1710$
- ABS with λ=0.001: $\hat{p} = 0.9953$ and $n = 1814$
- FFCS with the sensor fault rates (30,30,30): $\hat{p} = 0.9944$ and $n = 2121$
- FFCS with the sensor fault rates (20,10,20): $\hat{p} = 0.9920$ and $n = 3180$

As shown above, although the complexities of ATCS, ABS, and FFCS are different, the cost of the hybrid SMC technique for these target systems does not change much for similar $\hat{p}$ (i.e., 0.9920–0.9956). A slightly increasing number of n from 1710 to 3180 for decreasing $\hat{p}$ from 0.9956 to 0.9920 is due to the characteristics of BIET; BIET requires more sample paths as $\hat{p}$ decreases from 1 (Section 2.4.1). Therefore, we can expect that SMC techniques can be applied to large complex safety critical systems to assure their reliability.

7.3 SMC Techniques to Obtain a Certificate of Safety Standards

There are various international standards (e.g., DO-178C [20] for avionics domain, ISO-26262 [11] for automobile domain, IEC-60601 [9] for medical electrical equipment domain, etc.) to assure reliability of safety critical systems. Since products with a certificate can have a strong competitive power in market, manufacturers spend a large amount of man power and project time to acquire a high-level certificate for safety standards. For example, automobile manufacturers such as BMW and GM start to apply ISO 26262 standard for safety critical components.

To obtain a high-level certificate, vendors should provide strong cases or 'proof' that their products achieve high reliability. For example, ISO-26262 requires that a vendor of automobile components should apply formal verification techniques to the components to obtain a certificate of automotive software integrity level (ASIL) D. However, conventional formal verification techniques such as state model checking and theorem proving are difficult to apply for the purpose due to the state space explosion problem and lack of field engineers who are proficient in deductive proof.

[1] For different target systems, we should use n as a measure of the SMC cost, not $time$, since $time$ varies depending on the execution time of a target system.

From our experience of applying various SMC techniques for safety critical systems on automobile domain such as ATCS, ABS, and FFCS, we expect that the hybrid SMC technique can be applied successfully to obtain a high-level certificate of ISO 26262. A main reason is that the hybrid SMC technique is reasonably fast and precise (Section 7.1). For example, it takes less than 12 hours to verify FFCS with most precise parameters with regard to the PBLTL formula. Since most of the time cost is due to the simulation cost, SMC itself will take much less time to check other PBLTL formulas if any by utilizing saved sample traces. Second reason is that the cost of the SMC techniques is independent of the complexity of a target system (Section 7.2), since SMC techniques do not analyze the complex internal logic of a target system. Finally, SMC techniques are easy to Second reason is the strong applicability of SMC; SMC requires only executable target system/model.

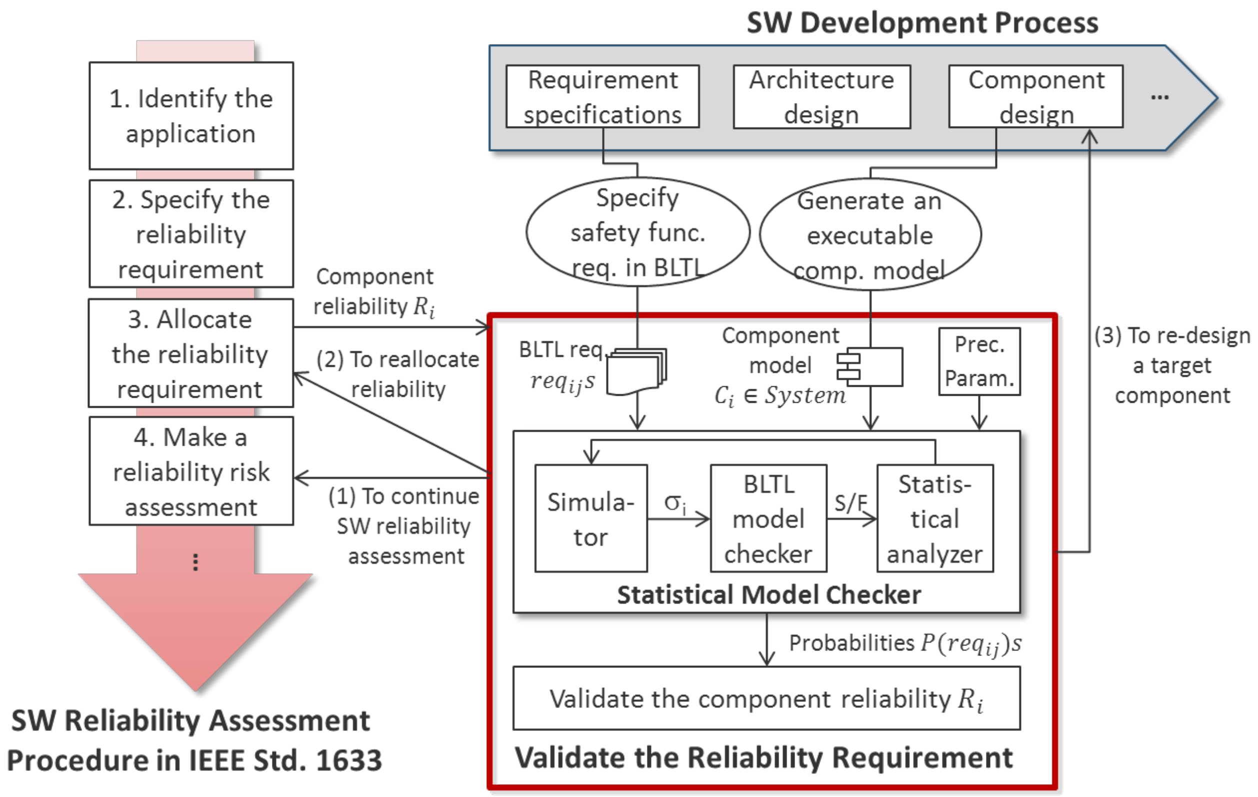
Chapter 8. Validating Software Reliability through Statistical Model Checking

During the last couple of decades, the proportion of software in safety critical systems has significantly increased. Thus, to assure the high level safety, it is essential to improve software reliability. Consequently, it has become very important to implement and acquire highly reliable software and to satisfy the safety requirements imposed by the functional safety standards such as IEC 61508 and ISO 26262 [8]. Safety Integrity Level (SIL) in IEC 61508 or Automobile Safety Integrity Level (ASIL) is defined as a measure of the quality or dependability of a system [10, 11]. Safety integrity level for a target system is determined from the assessment of three important factors: Improved Reliability, Failure to Safety, and Verification & Validation. Improved Reliability is the probability of a safety-related system satisfactorily performing the required safety functions under all stated conditions within a stated period of time. To develop a highly reliable software intensive system, a reliability goal is allocated for a target system according to a target SIL/ASIC level after hazard analysis and risk assessment. Then, software reliabilities are allocated to each software component at the early stage of a lifecycle. The allocated reliability of each component is validated through failure detection during the testing phases (e.g., system testing and acceptance testing), since there is no way to know whether the allocated component reliabilities can satisfy the overall system reliability goal at the earlier phases. This late validation can miss subtle defects due to limited project time and lead to a high software development cost and delayed project delivery. Recently, several software reliability prediction models have been introduced to quantitatively manage software reliability at early development phases (i.e., architecture or design phases) based on the structure and usage profile of the components in a software system [18]. However, these approaches have some limitations. First, the software reliability models are unrealistic due to lack of empirical data, especially in the early development phase [3]. Second, these models assume that the reliability of each target component is known, which is not true for software components in real-world, unlike hardware components.

In this chapter, we propose our new SMC-based software reliability validation framework to validate the reliability of the safety critical system in early development stage (see Section 8.1).

8.1 SMC-based Software reliability validation framework

We propose a new framework to validate the allocated component reliabilities in the early development stage through a statistical model checking (SMC) technique in the early development stage by extending the software



SW Development Process
Requirement specifications
Architecture design
Component design
...
Specify safety func. req. in BLTL
Generate an executable comp. model
1. Identify the application
2. Specify the reliability requirement
3. Allocate the reliability requirement
4. Make a reliability risk assessment
Component reliability R_i
(2) To reallocate reliability
(1) To continue SW reliability assessment
(3) To re-design a target component
BLTL req. $req_{ij}s$
Component model $C_i \in System$
Prec. Param.
Simula-tor
σ_i
BLTL model checker
S/F
Statis-tical analyzer
Statistical Model Checker
Probabilities $P(req_{ij})s$
Validate the component reliability R_i
Validate the Reliability Requirement
SW Reliability Assessment Procedure in IEEE Std. 1633

Figure 8.1: Software reliability validation framework

reliability assessment procedure in IEEE Std. 1633 [1]. Specifically, we extend IEEE Std. 1633 by adding a new step, “Validate the Reliability Requirement” after the “Allocate the reliability requirement” step in the software reliability assessment procedure. Since a good reliability validation depends on analyzing a target system as if it was operated in the real field, an operational profile, which is a quantitative characterization of how the system will be used, can be used in the framework [19]. Figure 8.1 shows the an overview of our software reliability validation framework which validates the reliability goal using a SMC technique. The detailed process of the framework is as follows.

1. Based on the reliability goal of a target system obtained at in the “2.2. Specify the reliability requirement” step in the software reliability assessment procedure of IEEE Std. 1633, a reliability goal R_i is allocated to each component C_i at in the “3.3. Allocate the reliability requirement” step.

2. At In the “Validate the Reliability Requirement” step (see the central box in Figure 8.1), SMC generates random sample execution traces σ_is repeatedly until σ_is generated are enough to calculate the probability that C_i satisfies req_{ij} (i.e., $P(req_{ij})$). If not, SMC simulates C_i again to generate more sample traces.

3. After calculating $P(req_{ij})$s for all req_{ij}s, the framework validates the allocated reliability goal R_i of C_i by comparing R_i with calculated reliability R'_i obtained based on $P(req_{ij})$s and corresponding weight values for req_{ij} (see the bottom box of “Validate the Reliability Requirement” of Figure 8.1). If R'_i satisfies the

assigned reliability goal R_i (i.e., $R'_i >= R_i$), the process of the software reliability validation continues for the next component C_{i+1} with regard to R_{i+1}. If the calculated reliabilities of all components satisfy the allocated reliability goals, the framework continues the remaining software reliability assessment procedure (see the leftward arrow "(1) To continue SW reliability assessment" in Figure 8.1).

4. If R'_i does not satisfy the assigned reliability goal R_i, the reliability goals of all components should be reallocated (see the leftward arrow "(2) To reallocate reliability" in Figure 8.1). If the reallocation keeps failing, it may indicate that the target component was designed incorrectly. Thus, after several trials of the reliability reallocation, the component C_i should be redesigned to improve the reliability of the target component (see the upward arrow "(3) To re-design a target component" in Figure 8.1).

Chapter 9. Case Study of SMC-based Software Reliability Validation Framework in Automobile

This chapter presents an overview of a fault-tolerant fuel control system (FFCS) in an automobile domain, which is our main target system to apply the SMC-based software reliability validation framework (see Section 9.1). Furthermore, this chapter describes the experiment to apply the SMC-based software reliability validation framework to FFCS, through which we have demonstrated the advantages of the proposed framework (see Section 9.2).

9.1 Target system of case study: fault-tolerant fuel control system

In Figure 3.1, we present the overall diagram of a fault-tolerant fuel control system (FFCS) [16], which is a safety critical component of the engine controller in a consumer vehicle. FFCS controls the fuel rate to inject 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 input and generates a proper fuel rate and an air-fuel ratio. FFCS consists of the following three components: Sensor Failure Detector & Estimator (SFDE), airflow calculator, and fuel calculator (see Section 3.1 for more explanation).

9.1.1 Sensor failure detector and estimator

Figure 9.1 is a block diagram for the SFDE of a FFCS. The SFDE receives four sensor data as input and generates four sensor data as output and the engine-shut-down command used only when multiple sensor failures occur. The SFDE consists of a sensor failure detector and a sensor data estimator. The sensor failure detector receives all four sensor data and decides if each sensor is failed. Sensor failure detector delivers all sensor data, and, if a sensor fails, notifies the sensor data estimator of the sensor failure. If multiple sensors fail, the sensor failure detector shuts down the engine since the air-fuel ratio cannot be controlled.

9.1.2 Airflow Calculator

Figure 9.2 is a block diagram of the airflow calculator of FFCS. The airflow calculator receives four sensors data from the sensor failure detector & estimator (SFDE) component and estimates an airflow value with feedback correction value. The airflow calculator consists of an airflow estimator and an airflow corrector. The airflow

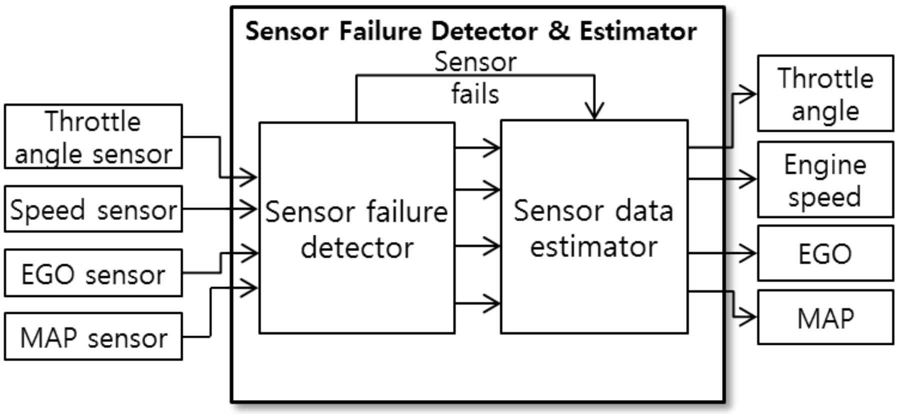


Figure 9.1: Block diagram of SFDE

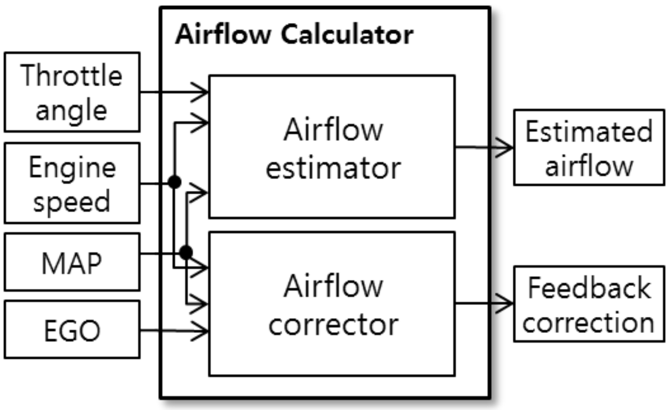


Figure 9.2: Block diagram of Airflow Calculator

estimator estimates an airflow value based on throttle sensor, speed sensor, and MAP sensor data. The airflow corrector calculates a feedback correction value based on EGO sensor, speed sensor, and MAP sensor data to obtain more accurate fuel rate in the next component, the fuel calculator.

9.1.3 Fuel Calculator

Figure 9.3 is a block diagram of the fuel calculator of FFCS. The fuel calculator receives the estimated airflow data and the feedback correction data from the airflow calculator component and calculates the fuel rate which keeps an air-fuel ratio optimal. The fuel calculator consists of a fuel calculator and a compensator. The fuel calculator calculates a feed-forwarded fuel rate based on the estimated airflow value, which makes the air-fuel ratio optimal. The compensator calibrates the feed-forwarded fuel rate with the feedback correction value and generates the fuel rate which is finally supplied to the engine and also the air fuel ratio.

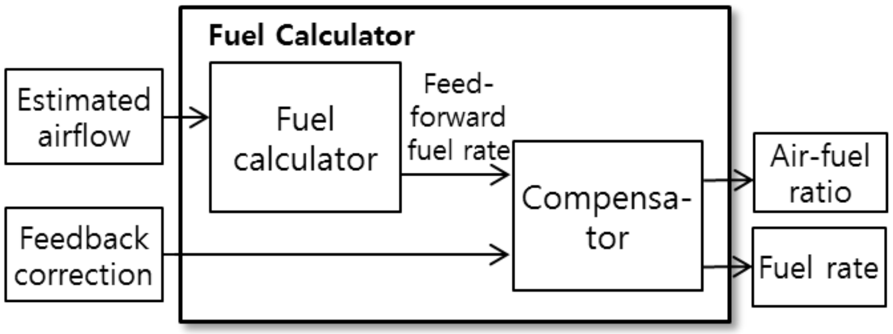


Figure 9.3: Block diagram of Fuel Calculator

9.2 SMC-based Software Reliability Validation Framework Experiments on FFCS

In this section, the validation method for allocating the reliability goal of FFCS and generating several safety functional requirements is described in Section 9.2.1, the experimental setting for applying SMC is described in Section 9.2.2, and the experimental result of SMC for each component of FFCS is described in Section 9.2.3.

9.2.1 Validation Method of the Software Reliability of FFCS

We specify the reliability goal for FFCS as 0.9999 (ASIL D in ISO 26262 [11] requires $1 - 10^{-3}$ to $1 - 10^{-9}$ reliability goal). Since all components of FFCS (i.e., the SFDE, the airflow calculator, and the fuel calculator) are combined sequentially, the reliability of a target system R_T can be calculated by multiplying the reliabilities of the components of the target R_i's as follows, where n is a total number of components in the system:

$$R_T = \prod_{i=1}^{n} R_i'.$$

To satisfy the total reliability of FFCS (i.e., 0.9999), we allocated the reliability goals for the components of FFCS following the advice from a field expert on the automobile engine controller:

- Sensor failure detector & estimator (SFDE): 0.99997
- Airflow calculator: 0.99997
- Fuel calculator: 0.99997

A basic principle to specify safety functional requirements for reliability validation is to describe a requirement for each output of a component (for example, we specify four requirements for the SFDE, each of which corresponds to output values of throttle angle, speed, EGO, and MAP). This is because a main task of a component is to compute output values and, thus, the reliability of a component is closely related with the output values.

Through the discussion with the field expert, we identified a total of eight requirements for the SFDE, airflow calculator, and fuel calculator. For example, the SFDE has the following four safety functional requirements for the four corresponding outputs:

- $req_{throttle}$: The throttle output should not be out of the throttle opening range from 3% to 90%.
- req_{speed}: The engine speed output should not exceed 628 rad/sec (= 6000 rpm).
- req_{EGO}: During the initial warm-up period, EGO output should not be out of the range [0,1]. After the warm-up, EGO output should be between 0.03 and 0.97.
- req_{MAP}: The MAP output should not exceed 1 atmosphere.

We can calculate the reliability of a component R'_i by using a weight to each requirement as follows, where $w_{req_{ij}}$ is a weight value for requirement req_{ij}, and $P(req_{ij})$ is the probability result for req_{ij}.

$$R'_i = \sum_{req_{ij} \in REQ)} (w_{req_{ij}} \times P(req_{ij}))$$

Through the discussion with the field expert, we determined the weight values: $w_{throttle}$=0.11, w_{speed}=0.45, w_{EGO}=0.09, and w_{MAP}=0.35. This indicates that speed and MAP sensors are more important for the reliability of the SFDE than throttle and EGO sensors.

9.2.2 Experiment Setting of SMC

We used a Simulink/Stateflow model of a FFCS in the Matlab R2010a. We simulated the FFCS model using Matlab simulator to generate sample execution traces. To validate if the FFCS model satisfies the reliability goal (i.e., 0.9999), we applied a Bayesian Interval Estimation Testing (BIET) SMC technique. To obtain precise probability result (i.e., $1 - 10^{-4}$ goal), we set the SMC precision parameters δ= 0.00005 and c= 0.9999 for BIET (see Section 6.2.3 for detailed tool explanation).

We built a stochastic environment model for FFCS that generates random faults at the sensors. We made a random fault generator module and connected this module to the sensors. The random faults are modeled by four independent Poisson processes with different arrival rates. Each mean inter-arrival fault rate of each sensor is given as follows: (throttle, speed, EGO, MAP) = (8, 10, 9, 7). For simplicity, we assume that all the operations of FFCS have the same occurrence rate. For a larger and complex system, the operational profile must be considered so that the most frequently used operation will have the most testing. The BLTL model checker evaluates safety functional requirements (i.e., $req_{throttle}$, req_{speed}, req_{EGO}, and req_{MAP}) over Matlab/Simulink simulation traces. The current model checker (implemented as a proof-of-concept prototype in 500 lines of Matlab script) was implemented in a specific way to evaluate the eight safety functional properties. We plan to implement

Table 9.1: The SMC result for validating reliability of SFDE

Component	Requirement	Probability	No. of samples	No. of failed samples	Verification time (hr)	Calculated reliability
Sensor Failure Detector & Estimator (SFDE)	$req_{throttle}$	0.999889	776747	85	318.91	0.999973
	req_{speed}	0.999989	92098	0	38.35	
	req_{EGO}	0.999933	533735	35	222.22	
	req_{MAP}	0.999989	92098	0	38.31	
Airflow Calculator	$req_{FeedbackCorrection}$	0.999959	293055	11	121.97	0.999950
	$req_{EstimatedAirflow}$	0.999947	452185	23	193.29	
Fuel Calculator	$req_{FuelRate}$	0.999972	177813	4	75.99	0.999954
	$req_{AirFuelRatio}$	0.999914	638753	54	273.45	

a general model checker which can evaluate arbitrary BLTL formulas over Matlab/Simulink simulation traces and release the model checker publicly. The BIET statistical analyzer was implemented in 50 lines of Matlab script. The BIET analyzer is independent from the model checker and safety functional requirements (we have released the BIET analyzer publicly at `http://pswlab.kaist.ac.kr/tools/SMC`). Note that these two components of SMC (once we complete the implementation of a general BLTL model checker) can be re-used for other target systems without modification. Thus, it will not be difficult for practitioners to apply the proposed software reliability validation framework based on SMC to their safety critical systems. All experiments were performed on 64 bit Windows 7 Professional equipped with a Intel i5 3.40 GHz and 8 GB of memory. We used Matlab R2010a for the experiments.

9.2.3 SMC Result

Table 9.1 describes the experiment result of applying SMC to the SFDE. For each safety functional requirement, the table describes the probability that the SFDE satisfies the corresponding safety functional requirement, the number of sample traces generated, the number of failed sample traces, and the verification time in hours. Finally, based on the probabilities in Table 1, we can calculate the estimated reliability R'_i of the SFDE with the weight values as follows.

$$R'_i = 0.11 \times 0.999889 + 0.45 \times 0.999989 + 0.09 \times 0.999933 + 0.35 \times 0.999989 \neq 0.999973$$

Since the reliability goal of the SFDE is 0.99997, the calculated reliability of the SFDE by SMC is larger than the reliability goal and we can conclude that the SFDE satisfies the allocated reliability goal. In each requirement's probability result in SFDE component, the probability results of the requirement req_{speed} and requirement req_{MAP} are higher than the reliability goal of this component, (i.e., all 0.999989), but for the requirement $req_{throttle}$ and req_{EGO}, the probability results are lower than the reliability goal of this component (i.e., 0.999889 for throttle and 0.999933 for EGO). Since the experiments for req_{speed} and req_{MAP} have no failed samples, both probability results are same.

For $req_{throttle}$ and req_{EGO}, due to the number of failed samples (85 for $req_{throttle}$ and 35 for req_{EGO}), their probability results are low. In spite of low probability results of $req_{throttle}$ and req_{EGO}, the reliability of SFDE, component is higher than the reliability goal of this component since the weight values of $req_{throttle}$ and req_{EGO} are lower than req_{speed} and req_{MAP}. This indicates that speed and MAP sensors have more criticality than throttle and EGO sensors. In cases of remained components, the calculated reliability result of the airflow calculator component is 0.999950, and that of the fuel calculator component is 0.999954. The estimated reliabilities of the airflow calculator component and the fuel calculator component are failed to achieve their reliability goals. For the airflow calculator component, the probability results of all two requirements, $req_{FeedbackCorrection}$ and $req_{EstimatedAirflow}$ are low (i.e., 0.999959 and 0.999947). This result indicates that the reliability goals of all components should be reallocated or both factors (i.e., feedback correction and estimated airflow) might have problems in the airflow calculator component. For the fuel calculator component, the probability result of requirement $req_{FuelRate}$ is higher than the reliability goal of this component (i.e., 0.999972), but the probability result of $req_{AirFuelRatio}$ are much lower than that of this component (i.e., 0.999914). Thus, the reliability goals of all components should be reallocated or air fuel ratio factor might have defects in the fuel calculator component. Therefore, the possible choices are that the reliability goals of all components should be reallocated or these two components should be redesigned to achieve the reliability goal of each component. The prior suggestion by the proposed framework is reallocating the reliability goals of all components of FFCS.

Chapter 10. Conclusion and Future Work

At the outset of this thesis, we believe that SMC technique can be helpful for achieving safety certificates of safety critical systems such as ISO-26262 for automobile domain and DO-178B/C for avionics soon.

From the empirical evaluation of four state-of-the-art SMC techniques on FFCS, we have 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.

We have developed a new hybrid SMC technique which integrates SPRT and BIET. By applying this new hybrid technique to three safety critical systems in the automobile domain (i.e., ATCS, ABS, and FFCS), we have demonstrated that the hybrid SMC technique achieves precise verification results fast compared to a single SMC technique - SPRT or BIET. In our experiment, our hybrid SMC technique was around 4 times faster than BIET.

The SMC technique can be utilized for the early validation of software reliability in modeling phase, which is a salient contribution for ensuring high reliability of safety critical systems. Until the SMC techniques have been proposed, it was almost impossible to validate the reliabilities of software components for safety critical systems at the early stage of the development life cycle, due to complex hybrid characteristics caused by continuous dynamics to interact with physical environment and discrete computations to handle modal operations.

As many safety critical systems such as automobiles and avionics are developed using a model-driven development (MDD) approach, the proposed validation framework can be seamlessly integrated in an existing development process of industries. Thus, the proposed methodology based on SMC techniques can be realistically adopted by industries and can contribute to increasing the reliability of software as well as decreasing the overall development cost through early detection of design faults or incorrect reliability allocation, etc.

As future work, we will collaborate with Hyundai motor company to apply the hybrid SMC technique to real control components of automobiles. We believe that the hybrid technique can provide more scientific assurance about the reliability of components than conventional testing techniques. In addition, we plan to use this hybrid technique in a process to obtain an ISO-26262 certificate.

References

[1] IEEE Std. 1633. Ieee recommend practice on software reliability. *IEEE Computer Society*, 2008.

[2] D. Antic, V. Nikolic, and D. Mitic. Sliding mode control of anti-lock braking system: An overview. *Automatic Control and Robotics*, 9(1):41–58, 2010.

[3] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik. Early prediction of software component reliability. In *ICSE*, 2008.

[4] 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.

[5] E.M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In *ATVA*, 2011.

[6] M.H. Halstead. *Elements of Software Science*. Elsevier Science Ltd, 1977.

[7] T. Herault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, 2004.

[8] D.S. Herrmann. Software safety and reliability. *IEEE Computer Society*, 1999.

[9] International Electrotechnical Commission (IEC). IEC 60601: Medical electrical equipment - part 1: General requirements for basic safety and essential performance, 2005.

[10] International Electrotechnical Commission (IEC). IEC 61508: Functional safety of electrical/electronic /programmable electronic (E/E/PE) safety related systems, 2005.

[11] International Organization for Standardization (ISO). ISO 26262: Road vehicles – functional safety, 2011. `http://www.iso.org/iso/catalogue_detail?csnumber=43464`.

[12] 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 *CMSB*, 2009.

[13] Y. Kim, M. Kim, and T. Kim. Statistical model checking for safety critical hybrid systems: An empirical evaluation. In *HVC*, 2012.

[14] C.W. Kirhwood. System dynamics methods: A quick introduction. Technical report, 1998. Arizona State University.

[15] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, 2011.

[16] 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.

[17] G. Li and J. Hu. Modeling and analysis of shift schedule for automatic transmission vehicle based on fuzzy neural network. In *WCICA*, 2010.

[18] M.R. Lyu. Software reliability engineering: A roadmap. In *Workshop on the Future of Software Engineering (FOSE)*, 2007.

[19] J.D. Musa. Operational profiles in software-reliability engineering. *IEEE Software*, 10(2):14–32, 2008.

[20] Radio Technical Commission for Aeronautics (RTCA). Do-178c: Software considerations in airborne systems and equipment certification, 2012.

[21] P.K. Sen and J.M. Singer. *Large sample methods in statistics: An Introduction with Applications*. New York: Chapman & Hall, 1993.

[22] X. Teng, H. Pham, and D. R. Jeske. Reliability modeling of hardware and software interactions, and its applications. *IEEE Transactions on Software Engineering (TSE)*, 55, 2006.

[23] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[24] S. Yi, J. Heo, Y. Cho, and J. Hong. Adaptive mobile checkpointing facility for wireless sensor networks. In *ICCSA*, 2006.

[25] J. Yoo, E. Jee, and S. Cha. Formal modeling and verification of safety-critical software. *IEEE Software*, 26(3):42–49, 2009.

[26] H.L.S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, CMU, Jan. 2005.

[27] 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.

[28] H.L.S. Younes and D.J. Musliner. Probabilistic plan verification through acceptance sampling. In *AIPS Workshop on Planning via Model Checking*, 2002.

[29] 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.

[30] P. Zuliani, A. Platzer, and E.M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. In *HSCC*, 2010.

Summary

Hybrid Statistical Model Checking Technique for Reliable Safety Critical Systems

원자력 발전소나 자동차와 같은 안전필수시스템의 신뢰성은 우리 사회에서 매우 중요한 이슈가 되고 있다. 이런 안전필수시스템에 점점 더 많은 컴퓨팅 시스템들이 이용됨에 따라, 이런 복잡한 컴퓨팅 시스템들의 신뢰성을 보증하는 확인 및 검증 기법들의 수요가 증가하고 있다. 그러나 컴퓨팅 시스템들의 복잡도가 증가함에 따라, 테스팅이나 모델 체킹 등과 같은 기존의 확인 및 검증 기법들은 한계를 가진다. 왜냐하면 이런 시스템들은 종종 매우 복잡한 연속 역학을 제어하기 때문이다. 이런 안전필수시스템들의 신뢰성을 높이기 위해, 통계적 모델 체킹 방법들이 제안되었다. 통계적 모델 체킹 기법들은 주어진 대상 시스템이 주어진 안전 요구사항을 만족하는지를 통계적 기법들로 검사한다. 이 학위 논문에서, 우리는 크게 세 가지의 연구를 수행하였다. 첫째로, 최신의 통계적 모델 체킹 기법 네 가지를 자동차 안전필수시스템에 적용하는 사례연구를 통해 통계적 모델 체킹 기법의 안전필수시스템 신뢰성 보증에 대한 실제적 적용 가능성을 확인하였고 네개의 다른 기법들에 대한 각각의 장단점을 비교 분석하였다. 두 번째로, 우리는 새로운 하이브리드 통계적 모델 체킹 기법을 제안한다. 하이브리드 통계적 모델 체킹 기법은 첫 번째 비교 분석 연구를 통해 알게된 속도가 빠른 기법인 sequential probability ratio test (SPRT) 기법과 정확한 기법인 Bayesian interval estimation testing (BIET) 기법을 통합해 정확한 검증 결과를 빠르게 얻기 위한 기법이다. 우리는 하이브리드 통계적 모델 체킹 기법의 효과 및 효율성을 보이기 위해 우리의 기법을 세개의 자동차 안전필수시스템에 적용했다. 실험결과로부터 우리는 하이브리드 통계적 모델 체킹 기법이 BIET 기법보다 최대 4배 더 빠르고, BIET의 정확도를 유지하며, SPRT보다 정확함을 확인했다. 마지막으로, 소프트웨어 신뢰성을 소프트웨어 개발 프로세스의 이른 단계에서 검증하기 위한 해결방안으로, 통계적 모델 체킹 기법들을 활용해 검증하는 프레임웍을 제안한다. 그리고 제안한 프레임웍을 자동차 안전필수시스템에 적용하는 사례 연구를 통해 실제 안전성 인증을 획득하는 데에 대한 제안한 프레임웍의 적용가능성을 보였다.

감 사 의 글

이 논문을 완성하기까지 주위의 모든 분들로부터 수많은 도움을 받았습니다. 우선 항상 저를 걱정하시고 아껴주시는 사랑하는 부모님께 감사드립니다. 매일매일 해주신 전화는 저에게 많은 힘이 되었습니다. 그리고 항상 저를 굳게 믿고 따르는 사랑하는 동생 김민지 에게도 고맙다는 말을 전하고 싶습니다.

두 번째로, 저를 연구실로 받아주시고 많은 가르침을 주신 김문주 교수님께 감사드립니다. 부족한 저를 잘 이끌어주셔서 석사과정 2년 동안 연구란 어떤 것인지 많이 배우게 해주셨습니다. 그리고 부족한 저를 후배로 잘 이끌어주신 김윤호 박사과정과 홍신 박사과정께도 감사드립니다. 석사 생활에 대해 많은 조언과 가르침을 주셨습니다. 또한 한 때 같은 연구실 생활을 했던 Duc Bui Hoang에게도 감사드립니다. 저에게 친절하게 잘 대해주었습니다. 그리고 우리 연구실에서 박사후 과정을 하고 있는 Matthew Staats에게 감사드립니다. 서로 언어 교환 활동을 하며 많은 얘기를 나눌 수 있어 참 좋았습니다. 그리고 연구실 후배들, 안재민, 문석현, 연광흠 석사과정에게도 감사드립니다. 앞으로 남은 연구 생활 잘 버티길 바랍니다. 학부연구생으로 있는 박용배 학생에게도 감사드립니다. 석사과정 가면 참 잘할 것 같은 꿈나무 입니다.

세 번째로, 2년여 동안 함께 석사생활을 한 동기들에게 감사드립니다. 우선 김지혜, 손희석, 이명준, 최장호 에게 감사합니다. 석사 생활동안 서로 힘든 일, 슬픈 일, 좋은 일이 있을 때마다 서로 공유하고 함께할 수 있어 좋았습니다. 또한 룸메이트 고은비에게 감사합니다. 항상 열린 마음으로 저를 대해주고 힘을 주어 참 고마웠 습니다. 그리고 여자동기들 홍지혜, 김보경, 김현아에게도 감사드립니다. 가끔이지만 함께 나눈 대화가 많은 힘이 되었습니다. 이민행 오빠에게 감사드립니다. 1년 반동안 많은걸 배우고 느끼게 해주었습니다. 원하는 바 를 꼭 이루었으면 좋겠습니다. 김용기 오빠에게 감사드립니다. 무슨 일이 있을 때마다 하소연을 해도 항상 잘 받아주셨던 큰 오빠셨습니다. 그리고 많은 남자동기들 김상채, 조재상, 이민우, 우상정, 이제희, 한대희, 신민호, 서재백 등 에게도 감사드립니다. 석사기간동안 추억도 많이 쌓고 이것저것 많은 도움을 받았습니다. 11학번 후기 이현지와 정현지에게도 감사드립니다. 가끔 만나 함께 대화를 나눈 것이 많은 힘이 되었습니다. 그외의 나머지 선후배, 동기들에게 감사드립니다. 어딜 가든 잘할 것 입니다.

마지막으로, 중학교 친구들, 고등학교 친구들, 엘리트 학원 친구들, 대학교 친구들, 월스트리트 어학원 언니오빠동생들, 비트컴퓨터 학원 오빠, 인턴 동기들 등등 에게도 감사드립니다. 평소에 바쁘다는 핑계로 연락을 잘 못해도 항상 이해해주고, 힘이 되는 말을 해주어 감사합니다. 동측 식당 아주머니께도 감사드립니다. 항상 웃는 얼굴로 저 예뻐해주셔서 너무 감사했습니다. 이 이외에도 저와 인연을 함께한 많은 사람들께 감사드립니다.

저의 이 작은 결실이 그분들께 조금이나마 보답이 되기를 바랍니다.

이 력 서

이 름 :	김 영 주
생 년 월 일 :	1988년 2월 19일
출 생 지 :	경기도 군포시 산본동 한양아파트 1214-1503
본 적 지 :	경기도 군포시 산본동 한양아파트 1214-1503
주 소 :	대전 유성구 구성동 373-1 한국과학기술원 세종관 1213호
E-mail 주 소 :	jerry88@cs.kaist.ac.kr

학 력

2003. 3. – 2006. 2.	홍진고등학교
2006. 3. – 2011. 2.	이화여자대학교 수학과 부전공: 컴퓨터공학과 (B.S.)
2011. 2. – 2013. 2.	한국과학기술원 전산학과 (M.S.)

경 력

2009. 9. – 2010. 3.	LG전자 인턴

학회활동

1. **Youngjoo Kim**, Y. Kim, and M. Kim, *Case Study on Testing with KLEE Concolic Testing Tool*, Korean Institute of Information Scientists and Engineers, Seoul (Korea), Nov 25-26, 2011. (Best presentation award)
2. Y. Kim, M. Kim, **Youngjoo Kim**, and Y. Jang, *Industrial Application of Concolic Testing Approach: A Case Study on libexif by Using CREST and KLEE*, Intl. Conf. on Software Engineering (ICSE), Software Engineering in Practice (SEIP) track, Jun 2-9, 2012.
3. **Youngjoo Kim**, U. Jung, Y. Kim, and M. Kim, *Comparison of Search Strategies of KLEE Concolic Testing Tool*, Journal of KIISE: Computing Practices and Letters, Vol 18, Num 4, Apr 2012.
4. **Youngjoo Kim** and M. Kim, *Hybrid Statistical Model Checking Technique for Reliable Safety Critical Systems*, IEEE Intl. Symp. on Software Reliability Engineering (ISSRE), Nov 28-30, 2012.

5. **Youngjoo Kim**, M. Kim, and T. Kim, *Statistical Model Checking for Safety Critical Hybrid Systems: An Empirical Evaluation*, Haifa Verification Conference (HVC), Nov 6-8, 2012.

연 구 업 적

1. **Youngjoo Kim**, O. Choi, M. Kim, J. Baik, and T. Kim, *Validating Software Reliability through Statistical Model Checking: Safer, Cheaper, and Faster*, IEEE Software, 2012. (under review.)