Electronic Preprint for Journal of Information Processing Vol.28

Regular Paper

Automating Time-series Safety Analysis for Automotive Control Systems Using Weighted Partial Max-SMT

Shuichi Sato[1,a)] Shogo Hattori[2,b)] Hiroyuki Seki[2] Yutaka Inamori[1] Shoji Yuen[2]

Received: May 8, 2019, Accepted: November 7, 2019

Abstract: We propose a method to automate the detection of signal disturbance for a given unsafe property. To incorporate a signal disturbance, we introduce an auxiliary variable, called a *cushion variable*, for each signal variable to store a value altered by the disturbance that causes unintended state transitions. The signal disturbance is defined to negate the equalities between signal variables and their cushion variables. We develop a method to efficiently detect the signal disturbance by using a weighted partial maximum satisfiability modulo theories (Max-SMT) technique as a set of variables altered by faults resulting in an undesirable condition. By assigning the weights properly to the equations, we control the derivation of signal disturbance patterns with the required property. We present an experimental application of our method to a simplified cruise control system as a practical case study in two well-known methods of safety analysis, namely system theoretic process analysis (STPA) and fault tree analysis (FTA), for the automatic detection of time-series signal disturbances.

Keywords: safety analysis, reliability design, FTA, STPA, automotive control systems, state transition systems, trace formula, time-series analysis

1. Introduction

Modern automotive systems comprise numerous electronic control units (ECUs) connected over a controller area network. With the advances in wireless network technology, automotive systems can now be connected to external networks as seen in vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication [8], and to electronic devices such as smartphones through Wi-Fi or Bluetooth. Designing automotive control systems often requires elaborate safety analyses owing to these connections which might increase the chances of unexpected signal disturbances.

Several approaches have been proposed for automobile safety analysis, of which the major ones include fault tree analysis (FTA) [14], failure mode and effect analysis (FMEA) [12], and the hazard and operability study (HAZOP) [28]. System theoretic process analysis (STPA) [29], [30] has recently been proposed as a new safety analysis technique.

We focus on the analysis of an unsafe situation caused by signal disturbances. In automobile control, it is difficult to perfectly prevent signal disturbances from occurring and therefore the system needs to be carefully designed while taking these disturbances into account. In the safety analysis approaches above, analyzing the impact of time-series multi-signal disturbances on safety using these techniques is time-consuming in the case of complex systems. To ease the analysis, we propose automating the analysis of signal disturbances leading to undesirable conditions by checking the properties of traces with a bounded length. Here undesirable conditions mean system conditions that are undesirable in terms of safety. This study focuses on the time-series safety analysis of the undesirable conditions of the system. We characterize the transition system as a trace formula [11], [22], which comprises a set of traces. With no signal disturbance, the trace formula with the property of undesirable conditions is not satisfied because the system is designed to avoid these conditions. If the system reaches an undesirable condition, some signal values must be altered through signal disturbances.

When some signal values are altered by some fault, the undesirable condition may become satisfiable because of the mismatch between the trace formula and the undesirable conditions. To incorporate this mismatch, we introduce auxiliary variables, called *cushion variables*, for the original signal variables. A signal disturbance assigns different values to signal variables and the corresponding cushion variables. As constraints, we add equations between the original and the cushion variables, which may not hold when a signal disturbance occurs. The property of undesirable conditions can be satisfied by negating the equations between the signal variables and the corresponding cushion variables. Identifying these equations, which enable the property of undesirable conditions to hold, leads to detection of a signal disturbance.

In designing automotive control systems, an undesirable condition needs to be considered over a certain period of time. In this respect, the expression of an undesirable condition is required to

[1] Toyota Central R&D Lab. Inc., Nagakute, Aichi 480–1192, Japan
[2] Nagoya University, Nagoya, Aichi 464–8601, Japan
[a)] shuichi-sato@mosk.tytlabs.co.jp
[b)] hatsutori@sqlab.jp

address its time series. Furthermore, to efficiently identify signal disturbances causing undesirable conditions by using the minimum possible number of incorrect values, we limit the number of failures needed to acquire intermittent multi-signal disturbances.

The satisfiability checking of the formula allowing some failure is formalized as follows. By allowing to violate the equations between signal variables and cushion variables, the trace formula and fault property can be *true* simultaneously. We obtain a value assignment by regarding the cushion variable equations as soft clauses in the weighted partial maximum satisfiability modulo theories (Max-SMT) problem. To enumerate all signal disturbances causing the undesirable condition, we add blocking clauses repeatedly after obtaining a value assignment.

The main contribution of this paper is the development of an automated method for locating signal disturbances causing undesirable conditions in various safety analysis processes. Our method can automate the process of obtaining time-series signal disturbances using a weighted partial Max-SMT solver. This paper generalizes and extends the method proposed by Hattori [17] and Sato [33], and performs new experiments to show that the proposed method applies to most prevalent safety analysis processes.

The rest of this paper is organized as follows: Section 2 describes the model of the problem as constraints with the signal patterns causing undesirable conditions by violating equalities between signal variables and cushion ones. In Section 3, we present a method to solve the problem using a weighted partial Max-SMT. In Section 4, we show two case studies for a simplified automotive control system with cruise control to demonstrate the effectiveness of the proposed method. Section 5 discusses related work, and Section 6 presents concluding remarks and describes the future work.

2. Behavioral Model by Constraints

We model the behavior of an automotive control system as a set of finite traces of state transitions with signal variables in discrete event systems, where a signal variable stores a value for controlling the system and a state is characterized by a value assignment. A set of bounded traces is characterized by constraints over signal variables, called *trace formulae*, as a series of equations between signal variables. A behavioral property holds for traces if the conjunction of the property and the trace formula is satisfiable.

2.1 Trace Formulae

In this study an automotive control system is expressed as a finite-state transition model where a state is distinguished by an assignment of values to signal variables and a control mode. In this model a state transition occurs through a value update of signal variables. A *trace formula* [11], [22] is a Boolean formula satisfied by value assignments for traces that are obtained by unrolling cycles in the transition system for a fixed number of times. In a trace formula, the variables indexed by unrolling steps are used and a state transition is expressed as a conjunction of equations among these variables. A trace formula is satisfied only when all indexed variables are assigned to express the transition exactly. We convert a finite-state transition system to the trace formula with the bounded length of K.

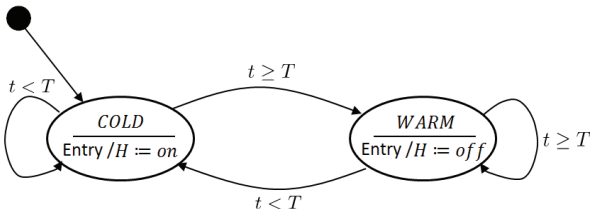


Fig. 1 Simple heater controller.

Let $M = (S, X, s_{init}, W)$ be given, where $S = \{s_1, \cdots, s_m\}$ is the set of control modes, X is the set of signal variables, $s_{init} \in S$ is the initial control mode, and W is the set of transitions between control modes. Here, $w \in W$ is given as a triple (s, g, f, s'), where $s, s' \in S$, g indicates a guard condition a constraint over X in s, and f shows a constraint over X in s and X' in s'. The state of M is (s, ν_X), where $s \in S$ and ν_X is a value assignment for X. For a state (s, ν_X) and transition $w = (s, g, f, s')$, a state transition of M is given as $(s, \nu_X) \rightarrow (s', \nu'_X)$ when $\nu_X, \nu'_X \models f(X, X')$, meaning that assignment ν_X for X and assignment ν'_X for X' satisfy the constraint $f(X, X')$.

A *trace* of M is an alternating sequence of control modes and transitions beginning with the initial control mode $md^{(0)}$. Given a trace $tr = md^{(0)}w^{(1)}md^{(1)}w^{(2)}\cdots w^{(k)}md^{(k)}$, $X^{(i)}$ is the set of signal variables for $md^{(i)}$. A trace formula for tr of M is in the form

$$\bigwedge_{i \leq K} \left(\left(\bigwedge_{x^{(i)} \in X^{(i)}} (x^{(i)} = v^{(i)}) \right) \wedge g^{(i+1)}(X^{(i)}) \wedge f^{(i+1)}(X^{(i)}, X^{(i+1)}) \right)$$

where $w^{(i)} = (md^{(i)}, g^{(i+1)}, f^{(i+1)}, md^{(i+1)})$, $v^{(i)}$ is a possible value for a signal variable $x^{(i)}$ and $\nu_X^{(i)}, \nu_X^{(i+1)} \models f^{(i+1)}(X^{(i)}, X^{(i+1)})$. A trace formula is satisfiable if and only if there exists a sequence of signal variable assignments along the trace.

For example, **Fig. 1** specifies a simple heater controller as the transition system $(\{COLD, WARM\}, \{H, t\}, COLD, W)$, where

$$W = \left\{ \begin{array}{l} (COLD,\ \ t < T \wedge H = on,\ \ COLD), \\ (COLD,\ \ t \geq T \wedge H = off,\ \ WARM), \\ (WARM,\ \ t < T \wedge H = on,\ \ COLD), \\ (WARM,\ \ t \geq T \wedge H = off,\ \ WARM) \end{array} \right\}$$

and T is a fixed constant as the target temperature.

Beginning from the *COLD* mode, where the heater is *on* ($H := on$), the temperature continues to rise until it reaches T. When the temperature becomes T, the transition from *COLD* to *WARM* occurs and the heater is turned *off* ($H := off$). When the temperature decreases below T, the heater is turned *on* by setting $H := on$.

Suppose T is 25 and t is initially 20 and the trace is given as *COLD*; *COLD*; *COLD*; *COLD*; *WARM*; *WARM*; *WARM*; *COLD*. Here we omit the description of the transitions since a single transition exists between control modes. A satisfiable trace formula for the trace is

$$\begin{aligned} & t^{(0)} = 20 \wedge H^{(0)} = on \wedge t^{(1)} = 22 \wedge H^{(1)} = on \wedge t^{(2)} = 24 \\ & \quad \wedge H^{(2)} = on \wedge t^{(3)} = 26 \wedge H^{(3)} = on \wedge t^{(4)} = 26 \\ & \quad \wedge H^{(4)} = off \wedge t^{(5)} = 25 \wedge H^{(5)} = off \wedge t^{(6)} = 24 \\ & \quad \wedge H^{(6)} = off \wedge t^{(7)} = 24 \wedge H^{(7)} = on. \end{aligned}$$

The trace formula for those traces with a length equal to or less than K is denoted as $\mathsf{TF}^{\leq K}$. In what follows, we only assign an integer to each signal variable. (For a Boolean variable, 1 is *true* and 0 is *false*.)

2.2 Property of Undesirable Condition

An automotive control system is designed to never meet undesirable conditions if it stays in an incorrect state just for a moment (due to, e.g., electrical noise). But it may reach undesirable conditions when it stays in incorrect states for more than a certain period. For example, a vehicle in cruise-control mode can reach the undesirable condition that the control system causes an unexpected delay for issuing the acceleration commands. Thus, an undesirable condition (UDC) is reasonably expressed by a formula satisfied by a time series of the assignment of improper values to n-consecutive variables in traces. For this n, we introduce $n\text{-}UDC_F^{\leq K}$ as the UDC property over a trace as follows:

$$n\text{-}UDC_F^{\leq K} \equiv \exists i.(i \leq K - n + 1) \\ \wedge F(md^{(i)}, X^{(i)}, md^{(i+1)}, X^{(i+1)}, \cdots, md^{(i+n-1)}, X^{(i+n-1)})$$

where K is the trace bound length, F is a predicate defined over the control modes and the signal variables, and $md^{(i)}$ and $X^{(i)}$ are, respectively, a control mode and a vector of signal variables at the execution step i. In the example of a vehicle in cruise control mode, as mentioned above, F comprises variables indicating an acceleration command and a distance to the leading vehicle and n is the number of certain consecutive clock cycles. Automotive control systems have some signals that maintain safety. We assume that UDCs can be detected by observing these values.

2.3 Signal Disturbances by Cushion Variables

Provided that the system is meticulously designed and no fault causing the property of the undesirable condition occurs, $\mathsf{TF}^{\leq K} \wedge n\text{-}UDC_F^{\leq K}$ is not satisfiable. If ν_X is an assignment that satisfies $\mathsf{TF}^{\leq K}$, then ν_X never satisfies $n\text{-}UDC_F^{\leq K}$. However, let ν_X be an assignment where some values of signal variables are changed and ν_X may satisfy $n\text{-}UDC_F^{\leq K}$. We specify the unintended change in signal values as a *signal disturbance*. Signal disturbances are regarded as mismatches among signal variables in the execution fragments. To present the mismatches, we explicitly assign values different from the original values to make the faults hold along with the trace. For this purpose, we introduce an extra variable, called a *cushion variable*, to each signal variable. Here each signal variable for $X^{(i)}$ is expressed by $u_j^{(i)}(j = 1, \cdots, Q)$, where Q is the number of signal variables. The cushion variable for $u_j^{(i)}$ is written as $u_j^{c(i)}$. With no disturbance, $u_j^{(i)} = u_j^{c(i)}$ holds for all js and all i-th steps in the trace formula. If a signal value is altered by signal disturbance, a different value is assigned to the cushion variable regardless of the value of the original signal variable. It is possible to trace failure points by checking for the equations between the original signal variables and cushion variables. The equation is:

$$\Omega_X^K \equiv \bigwedge_{i \leq K} \bigwedge_{j \leq Q} u_j^{(i)} = u_j^{c(i)}.$$

A value assignment not satisfying Ω_X^K may contain disturbed signals. Given a signal value assignment, a disturbed signal pattern is defined as a set of equations negated by the assignment. In what follows, we write σ for a signal value assignment including cushion variables.

Definition 1 (Signal disturbance pattern)

Let X be a set of signal variables. A *signal disturbance pattern* for signal value assignment σ, $DSP_X(\sigma)$ is a set of equations

$$DSP_X(\sigma) = \{u_j^{(i)} = u_j^{c(i)} \mid \sigma(u_j^{(i)}) \neq \sigma(u_j^{c(i)}), 1 \leq i \leq K, 1 \leq j \leq Q\}$$

We modify a trace formula $\mathsf{TF}^{\leq K}$ by replacing $u_j^{(i)}$ on the left-hand side of the equations with $u_j^{c(i)}$ for all $1 \leq i \leq K$ and $1 \leq j \leq Q$. The modified trace formula with the cushion variables is written as $\mathsf{TF}_c^{\leq K}$.

$\mathsf{TF}_c^{\leq K} \wedge \Omega_X^K \wedge n\text{-}UDC_F^{\leq K}$ is clearly not satisfiable under the above assumption. Removing the equations between the signal and cushion variables allows the faults causing the property of an undesirable condition. This is presented by removing the signal disturbance pattern from Ω_X^K. σ satisfies $\mathsf{TF}_c^{\leq K}$ under a predicate F, written as $\sigma \models_F \mathsf{TF}_c^{\leq K}$, iff $\mathsf{TF}_c^{\leq K} \wedge (\Omega_X^K - DSP_X(\sigma)) \wedge n\text{-}UDC_F^{\leq K}$ is *true*. In general, for a given K and F, conducting the fault analysis basically reduces to the task of finding an assignment σ such that $\sigma \models_F \mathsf{TF}_c^{\leq K}$.

2.4 Intermittent Signal Disturbance

In designing a system, *intermittent signal disturbances* are usually the most difficult to find. Although an intermittent signal disturbance has fewer signal disturbances than consecutive disturbances, it still leads to undesirable conditions. To adjust the scope of the signal disturbances, we limit the number of value alterations in the case of a signal disturbance pattern within a certain period of execution fragments. The constraints are defined as follows:

$$\Psi \equiv \forall i, j, \quad 1 \leq i \leq K - p + 1, \quad 1 \leq j \leq N. \\ \sum_{r=0}^{p-1} R(u_j^{(i+r)}, u_j^{c(i+r)}) \leq L. \qquad (1)$$

where

$$R(u_j^{(i)}, u_j^{c(i)}) = \begin{cases} 0 & \text{if } u_j^{(i)} = u_j^{c(i)} \\ 1 & \text{if } u_j^{(i)} \neq u_j^{c(i)}. \end{cases}$$

Here, L is given based on the intermittent signal disturbance that engineers assume. Ψ restricts traces such that signal disturbance occurs no more than L times in p execution steps.

3. Detecting Signal Disturbances by Satisfiability Using Weighted Partial Max-SMT Solvers

In the proposed method, signal disturbances are automatically detected using the weighted partial Max-SMT solver. We explain the outline of the weighted partial Max-SMT problem in Section 3.1, and describe how to detect signal disturbances by satisfiability checking with weighted partial Max-SMT solvers in Section 3.2.

3.1 Weighted Partial Max-SMT Problem

The weighted partial Max-SMT problem is an extension of the

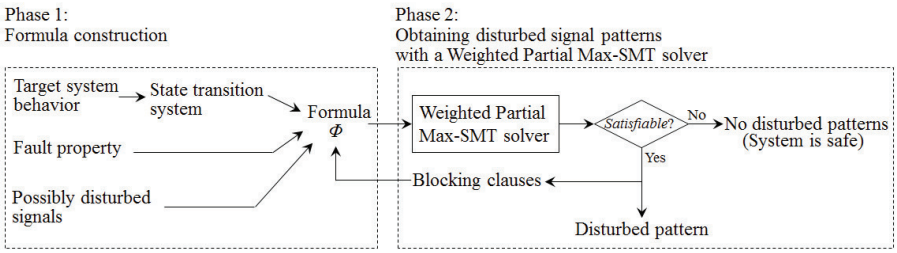


Fig. 2 Outline of proposed method.

partial Max-SMT problem, which is a combination of the partial Max-SAT problem [3] and the SMT problem [31]. In this section, we first explain the partial Max-SAT and SMT problems, and then introduce the weighted partial Max-SMT problem.

For a Boolean formula $\phi = C_1 \wedge C_2 \wedge ... \wedge C_n$ in conjunction normal form, the Max-SAT problem is to find an assignment of Boolean variables that satisfies the maximum number of clauses.

The *partial Max-SAT problem* is an extension of the Max-SAT problem. For a Boolean formula $\phi = C_1^H \wedge ... \wedge C_l^H \wedge C_1^S \wedge ... \wedge C_m^S$, we call C_i^H a hard clause ($1 \leq i \leq l$) and C_j^S a soft clause ($1 \leq j \leq m$). The partial Max-SAT problem finds an assignment such that, for a given ϕ, all hard clauses are satisfied and the maximum possible number of soft clauses is satisfied.

The *SMT* problem is a generalization of the SAT problem. Given a formula as the conjunction of clauses in first-order logic, an answer to the SMT problem determines if there exists an assignment of variables that satisfies all clauses. While the SAT problem only accepts Boolean formulas (propositional formulas), the SMT problem supports background theories, such as linear arithmetic, the theory of lists, the theory of arrays, and the theory of bit vectors. Compared with the SAT problem, the SMT problem is more useful because the formula can be simplified owing to its expressiveness.

The *partial Max-SMT* problem accepts a formula in first-order logic with various background theories, as is the case with the SMT problem, and the answer is an assignment that satisfies all hard clauses and the maximum possible number of soft clauses, as is the case with the partial Max-SAT problem.

The *weighted partial Max-SMT* problem is an extension of the partial Max-SMT problem. For each $i = 1, ..., n$, the soft clause C_i^S in the partial Max-SMT problem is assigned a cost $cost(C_i^S)$. The weighted partial Max-SMT problem finds an assignment that minimizes $\sum_i f(C_i^S)$, where $f(C_i^S) = 0$ if C_i^S is satisfied and $f(C_i^S) = cost(C_i^S)$ if C_i^S is not satisfied. The partial Max-SMT problem can be thought of as a weighted partial Max-SMT problem in which the cost is unity for all soft clauses.

3.2 Detecting Signal Disturbances by Satisfiability

Having assumed that a signal disturbance is a set of unsatisfied equations between the signal and cushion variables, we can automate the detection of signal disturbances by deriving the set of unsatisfied equations that cause $\mathsf{TF}_c^{\leq K} \wedge n\text{-}UDC_F^{\leq K}$ to hold.

Our method consists of the two phases shown in **Fig. 2**. The repetition of these phases enables the enumeration of signal disturbances causing undesirable conditions. Phase 1 constructs a formula based on the behavior of the target system with bound K, lists of signals that may be disturbed, and a property of undesirable condition. In Phase 2, the pattern of the disturbance signals causing faults is automatically extracted by a weighted partial Max-SMT solver. Each phase is described in detail below.

In Phase 1, given a state transition system, a property of undesirable condition, and a set of variables X that may be disturbed, we create a formula Φ as follows:

$$\Phi \equiv \mathsf{TF}_c^{\leq K} \wedge n\text{-}UDC_F^{\leq K} \wedge \Psi \wedge \Omega_X^K. \quad (2)$$

where Ψ specifies the extra constraints such as the intermittent conditions.

In Phase 2, we mark Ω_X^K in Φ as *soft clauses* and the remainder of Φ as *hard clauses*. We then solve the constraint as the weighted partial Max-SMT problem.

We specify weights for all equations in Ω_X^K. There may be many signal disturbance patterns that lead to undesirable conditions. The weights are used to control the order of deriving signal disturbance patterns. Specifying weights requires a heuristic that depends upon the expected signal disturbance patterns. If we assign uniform weights to all soft clauses, this minimizes the number of signal alterations, because a weighted partial Max-SMT problem is solved by minimizing the sum of weights of soft clauses that are not satisfied. Phase 2 is automated by applying a weighted partial Max-SMT solver such as Yices to Φ. The solver attempts to find a variable assignment that satisfies all hard constraints and soft clauses with the minimum sum of weights. If the solver finds such an assignment, the values of each variable and the soft clauses that are not satisfied are returned. Then, the obtained set of soft clauses forms a signal disturbance pattern.

The following are typical examples of the weight policy. If a signal disturbance pattern with the shortest period is required, we assign smaller weights for earlier steps, namely $wt_j^{(i)} < wt_j^{(i')}$ iff $i < i'$. For example, $wt_j^{(i)} = i$ can be assigned to equations in Ω_X^K. Given that a weighted partial Max-SMT solver falsifies soft clauses with small weights, a signal disturbance pattern $DSP_X(\sigma)$ consisting of smaller values of i is more likely to be found.

By contrast, if a uniform weight is assigned to all equations in Ω_X^K, we do not have a preference for signal disturbance patterns. Changing L in the intermittent constraint Φ to a smaller value is intended to find a subtle combination of signal disturbances that may cause undesirable conditions.

Blocking clauses We add the hard clauses to Φ, rejecting the acquired signal disturbance patterns as blocking clauses, and repeat Phase 2. For example, if $DSP_X(\sigma) = \{(u_1^{(1)} = u_1^{c(1)}), (u_4^{(3)} = u_4^{c(3)})\}$ is acquired as the disturbed pattern, we add the following

hard clauses:

$$u_1^{(1)} = u_1^{c(1)} \quad \vee \quad u_4^{(3)} = u_4^{c(3)}.$$

Owing to the existence of these hard clauses, the solver finds a different pattern satisfying $u_1^{(1)} = u_1^{c(1)}$ or $u_4^{(3)} = u_4^{c(3)}$, instead of the same patterns already obtained. We repeat the procedure until Φ is not satisfied to enumerate all signal disturbance patterns for the undesirable condition. Although this procedure may not terminate with a little repetition, observing the part of the repeated results is very useful for engineers in order to find a solution for avoiding the undesirable condition by discovering the key signals in the obtained disturbed patterns.

4. Case Study

We present a case study for a simplified automotive control system with a cruise control function. Section 4.1 describes the target system, and Section 4.2 shows the experimental results, where our method is applied in the process of two well-known safety analyses. In Section 4.3, we discuss the results. Since we are using the same example in Ref. [33], some figures and tables for explanations of the system are taken from our previous paper [33].

4.1 Target System

We use a simplified automotive control system with an adaptive cruise control functionality for the case study. This system has three ECUs: adaptive cruise control, neutral transmission control (TC), and an arbiter (ABT). **Table 1** shows the behavior of these ECUs. When the leading vehicle runs further away from our car, an acceleration command is issued by the cruising functionality unless the brake pedal is pressed under the cruise control. As a result, the car accelerates if the transmission gears are properly engaged. The simplified automotive control system is supposed to have the specification that a vehicle needs to move forward by issuing the acceleration command even when it is in a stop state. This might be considered as a correct behavior in some cars. This property is just for a simple example to demonstrate the existence of inconsistency in the system.

Table 1 ECUs in target system.

Name	Function
ACC	Controls acceleration and deceleration in accordance with leading vehicle
TC	Shifts into neutral gear during brief stops to improve gas mileage
ABT	Arbitrates multiple control requests

Figure 3 shows an overview of the signal flows in the automotive control system and the signals used in this system are listed in **Table 2**. Each ECU is executed periodically to process the signals given in Table 2.

CarModel in Fig. 3 shows the physical behavior of the vehicle, where linear arithmetic, comparison, and conditional branch operations are included. The control logic in ACC ECU includes a comparison between VehicleSpeed and LeadingVehicleSpeed. CarModel has linear arithmetic functions to calculate VehicleSpeed. The ACC and TC ECUs contain transitions in the control mode. For example, a transition in the ACC ECU occurs among each control mode according to the guard condition g_{ij} as shown in **Fig. 4**, where i and j indicate the control mode of the ECU before and after the transition, respectively. When the control mode is equal to 1 in the ACC ECU, ACC_AccelControlData / ACC_BrakeControlData can be generated.

4.2 Experimental Results

We applied our method to the simplified automotive control system by performing two well-known safety analyses: FTA and STPA. Section 4.2.1 and Section 4.2.2 present the results for FTA and STPA, respectively. Signals except IGSWOn in Table 2 are possibly disturbed in this experiment. The experiments are benchmarked on a machine with Intel Core TM i5-3470 RAM 6.0 GB and Microsoft Windows 7 Professional. We used the Yices SMT solver [13] 1.0.29 as a weighted partial Max-SMT solver in this experiment.

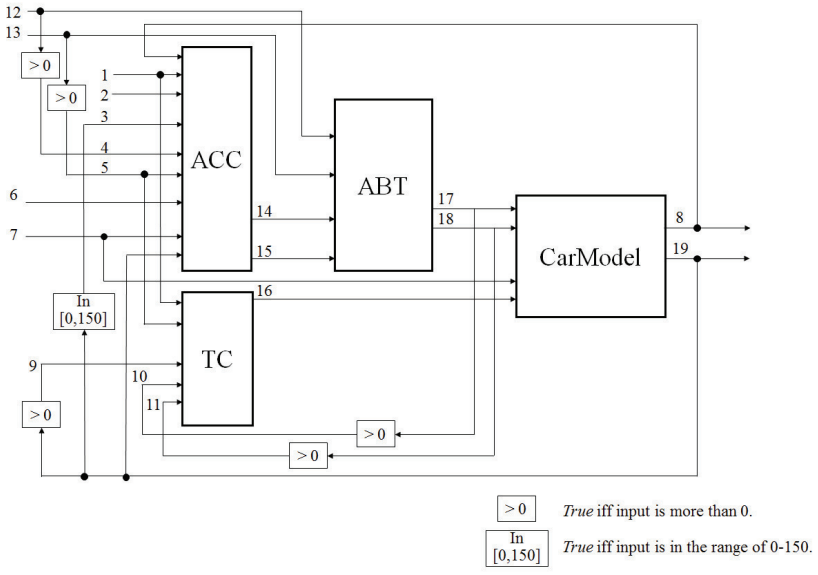


Fig. 3 Overview of simplified automotive control system. Each number refers to an explanatory entry in Table 2.

Table 2 Target system signals.

No.	Name	Type	Meaning
1	IGSWOn	bool	*True* iff ignition switch is on
2	RadarCruiseSWOn	bool	*True* iff ACC main switch is on
3	VehicleSpeedOK	bool	*True* iff vehicle speed is in [0,150]
4	AccelPedalOn	bool	*True* iff gas pedal is stepped on
5	BrakePedalOn	bool	*True* iff brake pedal is stepped on
6	ShiftRange	int	Shift range (−2:P −1:R 0:N 1–5:D)
7	LeadingVehicleSpeed	int	Speed of leading vehicle
8	Distance	int	Distance to leading vehicle
9	VehicleMoving	bool	*True* iff vehicle is moving
10	ABT_AccelControlOn	bool	*True* iff ABT_AccelControlData > 0
11	ABT_BrakeControlOn	bool	*True* iff ABT_BrakeControlData > 0
12	AccelPedal	int	Extent to which gas pedal is depressed
13	BrakePedal	int	Extent to which brake pedal is depressed
14	ACC_AccelControlData	int	Acceleration control value from ACC ECU
15	ACC_BrakeControlData	int	Braking control value from ACC ECU
16	TC_NeutralControlData	bool	Neutral control value from TC ECU
17	ABT_AccelControlData	int	Integrated acceleration control value
18	ABT_BrakeControlData	int	Integrated braking control value
19	VehicleSpeed	int	Speed of vehicle

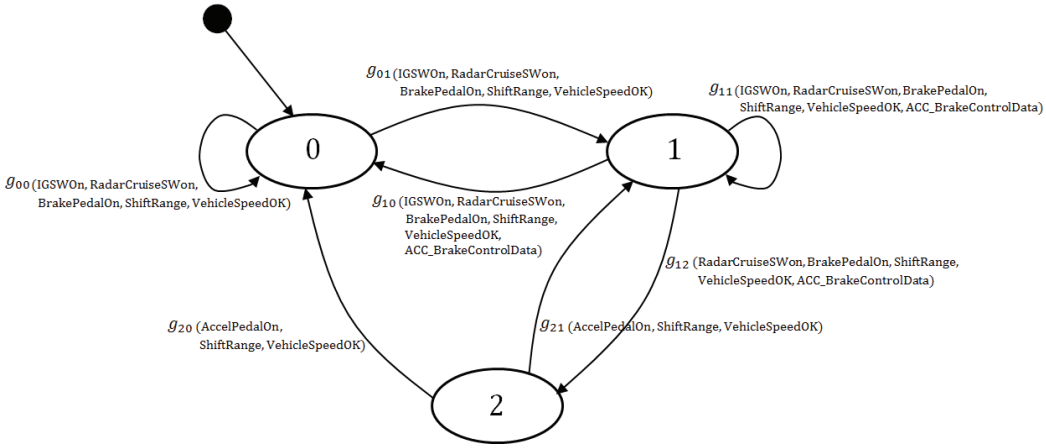


Fig. 4 Transition among each control mode in ACC ECU.

4.2.1 Application to FTA

FTA is the most popular and traditional approach to performing safety analysis for automobiles. In FTA, an engineer lists the hazardous states of the vehicle and hierarchically enumerates the factors related to their occurrences. We investigate a hazardous state in which the vehicle on the road does not accelerate in cruise control mode even though the leading vehicle moves away from it. As described in Section 4.1, the ACC function normally tries to maintain a certain distance from the leading vehicle based on the information provided by the on-board sensor used to measure that distance. Applying FTA to the system encountering this hazard, the fault tree is derived as in **Fig. 5**. We use the hazard, "Keeps stopping in cruise control mode for n consecutive times even though the leading vehicle moves away from it." To generate the disturbed signal patterns leading to the hazardous state in Fig. 5 by way of the signal disturbance, let $n\text{-}UDC_F^{\leq K}$ be defined as follows:

$$
\begin{aligned}
n\text{-}UDC_F^{\leq K} \equiv\ & \exists\ i. 1 \leq i \leq K - n + 1 \\
& \wedge \bigwedge_{r=0}^{n-1} (LeadingVehicleSpeed^{(i+r)} > 0 \\
& \wedge\ Distance^{(i+r)} > C_d \\
& \wedge\ BrakePedal^{(i+r)} = 0 \wedge AccelPedal^{(i+r)} = 0 \\
& \wedge\ RadarCruiseSWOn^{(i+r)} = true \\
& \wedge\ VehicleSpeed^{(i+r)} = 0).
\end{aligned}
$$

K, n, and C_d were set to 10, 5, and 70, respectively. L in Eq. (1) was set to 1. Values were assigned to the variables as follows:

- $IGSWon^{(i)} = true$
- $RadarCruiseSWOn^{(i)} = true$
- $ShiftRange^{(i)} = 4$
- $BrakePedal^{(i)} = 0$
- $LeadingVehicleSpeed^{(i)}$ changes as: 30, 60, 90, 90, 120, 120, ...
- $VehicleSpeed_{init} = 0$

Here $VehicleSpeed_{init}$ indicates the initial value of VehicleSpeed. The weight of each equation in Ω_X^K was set to 10.

We did not obtain any disturbed pattern consisting of a single signal. This means that the automotive control system we use does not reach the hazardous state by the intermittent disturbance of a single signal. **Table 3** shows one of the experimental results, which is a signal disturbance pattern consisting of two signals. In this pattern, BrakeControlData is disturbed to 21 or namely, the cushion variable for BrakeControlData is equal to 21, at the execution step $i = 2$, and BrakePedal is disturbed to 21 or namely, the cushion variable for BrakePedal is

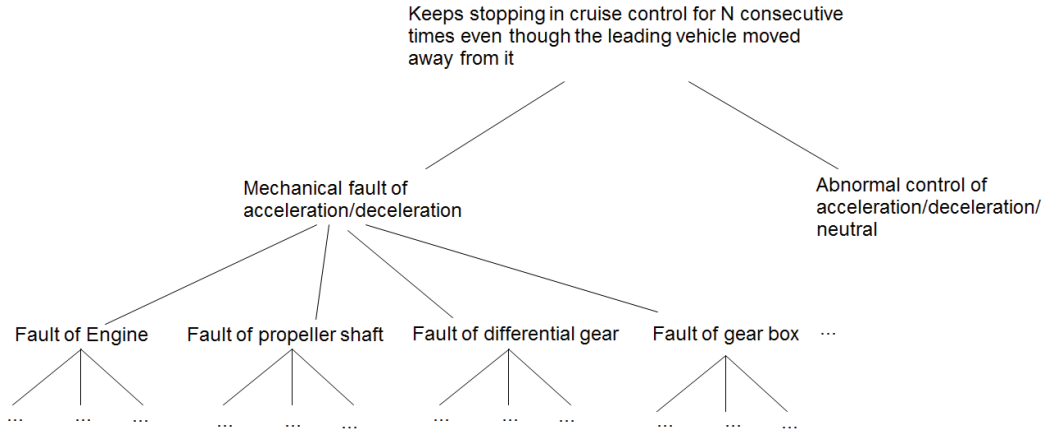


Fig. 5 Fault tree.

Table 3 Example including signal disturbance pattern with two signals.

Step i	BrakePedal		BrakeControlData	
	Normal Value	Disturbance Result	Normal Value	Disturbance Result
1	0	0	0	0
2	0	0	0	21
3	0	0	0	0
4	0	21	21	21
5	0	0	0	0

Table 4 Pattern of signals directly affecting vehicle speed.

Step i	TC_NeutralControlData	BrakeControlData	AccelControlData
1	*false*	0	0
2	*false*	21	160
3	*true*	0	220
4	*false*	21	220
5	*true*	0	280

equal to 21, at $i = 4$. These disturbances indirectly affect the satisfiability of $n\text{-}UDC_F^{\leq K}$, with a delay owing to the characteristics of the state transition system representing the automotive control system. **Table 4** shows the pattern of the signals directly affecting the speed of a vehicle, namely AccelControlData, BrakeControlData, and TC_ NeutralControlData. The signal disturbance presented in Table 3 causes the signal pattern presented in Table 4. Through the use of the signal pattern given in Table 4, the vehicle stops at $1 \leq i \leq 5$. We obtained other patterns consisting of BrakeControlData and BrakePedal by adding blocking clauses that refrained from generating the pattern given in Table 3. The target system has numerous intermittent disturbance patterns when we consider a time series, even if the number of disturbance signals is limited to two. In particular, the existence of continuous signals, such as VehicleSpeed, leads to various disturbances leading to $n\text{-}UDC_F^{\leq K}$. We repeatedly added blocking clauses that refrained from generating the patterns that consist of signal combinations included in the obtained patterns and obtained **Table 5**. This table lists all names of the signals in each pattern consisting of two signals obtained by the proposed method. The obtained signal disturbance patterns could have caused a situation in which the system kept stopping for five consecutive time steps in the cruise mode, despite increasing the distance to the leading vehicle.

Figure 6 shows the computation time required to obtain a disturbance signal pattern with different numbers of possibly disturbed signals. Each value is an average of 10 computation times. The possibly disturbed signals were randomly selected 10 times, except for the case where the number of possibly disturbed signals is equal to 0. In Fig. 6, no signal disturbance patterns were obtained when the numbers of possibly disturbed signals were 0 and 1. Even if the number of possibly disturbed signals was increased, the computation time was not increased.

Table 5 Signals in each disturbance pattern.

Signal Names	
VehicleSpeed	LeadingVehicleSpeed
BrakePedal	Distance
BrakePedal	VehicleSpeed
VehicleSpeed	Distance
ShiftRange	VehicleSpeed
VehicleSpeed	BrakePedalOn
VehicleSpeed	BrakeControlOn
Distance	ACC_BrakeControlData
LeadingVehicleSpeed	ACC_BrakeControlData
Distance	BrakeControlData
VehicleSpeed	BrakeControlData
VehicleSpeed	ACC_BrakeControlData
LeadingVehicleSpeed	BrakeControlData
LeadingVehicleSpeed	Distance
VehicleSpeed	ACC_AccelControlData
BrakePedal	ACC_BrakeControlData
BrakePedal	LeadingVehicleSpeed
VehicleSpeed	AccelControlDataDistance
VehicleSpeed	TC_NeutralControlData
VehicleSpeed	VehicleSpeedOK
RadarCruiseSWOn	VehicleSpeed
ACC_BrakeControlData	BrakeControlData
BrakePedal	BrakeControlData

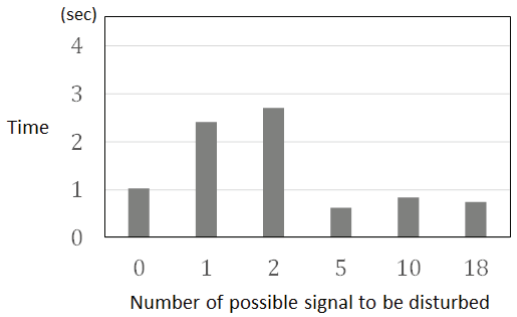


Fig. 6 Computation time with different numbers of possibly disturbed signals in FTA.

Step 1: Identify Unsafe Control Actions
Step 2: Identify Causes of Unsafe Control Actions

Fig. 7 STPA procedure.

4.2.2 Application to STPA

STPA [29], [30] has recently been proposed as a new safety analysis technique for complex systems. STPA has two fundamental procedures as shown in **Fig. 7**, where an unsafe control action (UCA) is a control action, such as an operation command to an actuator, that may lead to a hazardous state. Engineers perform these procedures by using the guide words. Abdulkhaleq [2] has shown that analyzing each control path's possibility of causing UCAs in Step 2 is time-consuming and requires detailed knowledge on the target system. It is particularly difficult to treat intermittent multi-signal disturbances arising from some undesired incidents such as temporal wire disconnections, when we use STPA in the later design phase, because we have to consider a huge number of time-series patterns in multiple signals.

In this experiment, we investigate the same hazardous state as the one in Section 4.2.1. Applying STPA to the system encountering this hazard, Step 1 derived a UCA (unsafe control action): An acceleration command is not provided for five consecutive clock cycles in cruise control mode, even though the leading vehicle moves further away. Let a $n\text{-}UDC_F^{\leq K}$ be defined as follows:

$$
\begin{aligned}
n\text{-}UDC_F^{\leq K} \equiv\ & \exists i.1 \leq i \leq K-n+1 \wedge \\
\textstyle\bigwedge_{r=0}^{n-1}\ & (LeadingVehicleSpeed^{(i+r)} > 0 \wedge Distance^{(i+r)} > C_d \\
\wedge\ & BrakePedal^{(i+r)} = 0 \wedge AccelPedal^{(i+r)} = 0 \\
\wedge\ & RadarCruiseSWOn^{(i+r)} = true \\
\wedge\ & ABT\_AccelControlData^{(i+r)} = 0).
\end{aligned}
$$

K, n, and C_d were set to 10, 5, and 70, respectively. L in Eq. (1) was set to 1. The same values as those in Section 4.2.1 were assigned to variables $IGSWOn^{(i)}$, $RadarCruiseSWOn^{(i)}$, $ShiftRange^{(i)}$, $BrakePedal^{(i)}$, $LeadingVehicleSpeed^{(i)}$, and $VehicleSpeed_{init}$.

We did not obtain any signal pattern consisting of a single signal. This fact is the same as the application of the proposed method to the FTA process. **Table 6** shows one of the experimental results, which is the signal disturbance pattern consisting of two signals. In this pattern, VehicleSpeed is disturbed to 151 at the execution step $i = 2$ and BrakePedalOn is disturbed to *true* at $i = 4$. We added blocking clauses that refrained from generating the patterns that consist of the signal combination included in the obtained patterns and obtained **Table 7**. This table lists the names of all signals in each pattern consisting of two signals obtained by the proposed method. The obtained signal disturbance patterns can cause a situation in which the system did not provide an acceleration command at $1 \leq i \leq 5$ in cruise mode, despite increasing the distance to the leading vehicle.

Figure 8 shows the computation time required to obtain a disturbance signal pattern with different numbers of possibly disturbed signals. In Fig. 8, no signal disturbance patterns are obtained when the numbers of possibly disturbed signals are 0 and 1. As in the case of application to FTA, the computation time was not increased even if the number of possibly disturbed signals was increased.

Table 6 Example including signal disturbance pattern with two signals.

Step i	VehicleSpeed		BrakePedalOn	
	Normal Value	Disturbance Result	Normal Value	Disturbance Result
1	0	0	*false*	*false*
2	0	151	*false*	*false*
3	0	0	*false*	*false*
4	0	0	*false*	*true*
5	0	0	*false*	*false*

Table 7 Signals in each disturbance pattern.

Signal Names	
ShiftRange	VehicleSpeed
RadarCruiseSWOn	VehicleSpeed
VehicleSpeedOK	VehicleSpeed
BrakePedalOn	VehicleSpeed

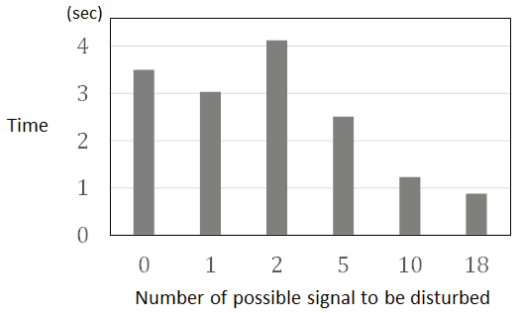


Fig. 8 Computation time with different numbers of possibly disturbed signals in STPA.

We see that all four patterns in Table 7 have VehicleSpeed. This fact indicates that keeping VehicleSpeed not disturbed by some means can eliminate the possibility of causing the satisfaction of $n\text{-}UDC_F^{\leq K}$ by two-signal disturbances [*1]. This observation shows that no two-signal disturbances can cause $n\text{-}UDC_F^{\leq K}$. If we use the constraint that VehicleSpeed is not disturbed, we obtain three-signal disturbances in **Table 8**.

4.3 Discussion

Manually obtaining the disturbed signal patterns in both the FTA and STPA processes is a time-consuming task. For two-signal disturbances of five execution steps, the total possible number of value pairs for two signals was approximately 1.5×10^{14}, even if each signal was binary; for three-signal disturbances of five execution steps, the total number was approximately 2.8×10^{19}. Hence, enumerating the disturbed signal patterns leading to undesirable conditions by hand is difficult, even for short time series. The proposed method can be applied to obtain the disturbed patterns in both the FTA and STPA processes, as seen in Sections 4.2.1 and 4.2.2.

As described in Section 4.2.2, our method provides a better understanding of the signals essential for maintaining vehicle safety by observing the signal patterns. This understanding leads engineers to design the minimum countermeasures for ensuring safety.

From the benchmark in Sections 4.2.1 and 4.2.2, the computation time to find a signal disturbance stays reasonably short in

[*1] It is a design decision whether a certain mechanism is introduced in the system to protect a critical signal (*VehicleSpeed*, in this example) from disturbance, though protecting all signals against disturbance is unrealistic.

Table 8 Example including signal disturbance pattern with three signals in STPA.

Step i	ShiftRange		VehicleSpeedOK		LeadingVehicleSpeed	
	Normal Value	Disturbance Result	Normal Value	Disturbance Result	Normal Value	Disturbance Result
1	4	4	*true*	*true*	30	30
2	4	3	*true*	*true*	60	60
3	4	4	*true*	*true*	90	−21
4	4	4	*true*	*false*	90	90
5	4	4	*true*	*true*	120	120

the FTA and STPA processes even if the number of possibly disturbed signals is increased. We expect the proposed method to be put to practical use in the earlier design phase when some measures against signal disturbances are not considered. That is to say, it is expected that engineers can design optimal safety measures from an overall perspective by obtaining the signals essential for maintaining safety with the proposed method before some safety measures are designed.

5. Related Work

Other safety analysis techniques are found in the automotive functional safety standard, ISO 26262 [21]. Here we mention FMEA [12] and HAZOP [28] since these are often used for automobiles.

FMEA is used to analyze component failures to identify the effect of the failures in a consistent manner, where the component includes systems and subsystems as well as parts. FMEA was originally applied to hardware failures. But today we see some works to apply this methodology to software systems [24]. In FMEA, we first enumerate component failures. After that, the effects of these failures are considered based on the enumeration. On the contrary, the proposed method is used to find out the causes that lead to undesired phenomena. Therefore, the proposed method cannot be used in the FMEA process.

HAZOP was originally proposed for chemical plants, but recently, some studies have applied it to software [32]. Nowadays, HAZOP is used in the automotive field. It is defined in IEC 61882 [19]. In HAZOP, the deviations from expected control actions are assumed based on guide words and the hazards from these deviations and the factors causing them are extracted based on the specification of the system. In other words, HAZOP includes both top-down and bottom-up approaches. The proposed method is considered to be used to enumerate the factors causing the deviations from the expected control actions in the HAZOP process. For example, the situation in which the control action in the system unexpectedly stops for a certain period is extracted using one of the guide words, "NO OR NOT," and the proposed method can be used to automatically find out signal disturbance patterns that lead to that situation. HAZOP is both a top-down and bottom-up approach and can start from middle, while FTA is a top-down approach and FMEA is a bottom-up approach.

Software model checking is a technology used for applying well-established finite-state model checking algorithms to verifying programs that are inherently infinite-state because of data structures such as arrays and pointers and/or control structures such as recursion and parallelism. For example, BLAST [18] and SLAM [6] are software model checkers based on predicate abstraction and refinement. As shown below, there has been another stream, called bounded model checking (BMC), which focuses on fault or bug detection rather than correctness verification. BMC assumes a bound on the execution steps, the number of loop iterations, or the recursion depth so that a simpler logical framework can be used, e.g., satisfiability checking of propositional logic rather than temporal logic. This paper aims to apply such a fault localization method based on BMC to the safety analysis of automotive control systems.

One of the earliest papers on SAT-based BMC [7] showed that a *trace formula* was introduced to represent the behavior of a program by unfolding a loop and in-lining a recursive call within a bounded number of times. CBMC [11] is a software model checker for C and C++ programs based on Biere's method [7]. Let P be a given program and T_P be the trace formula constructed from P. In addition, let us assume that an undesirable property H for P is given. A SAT solver is used to decide whether the formula $T_P \wedge H$ is satisfiable. If $T_P \wedge H$ is satisfiable, T_P contains a fault and a satisfying assignment corresponds to an example that brings P to an undesirable condition. If T_P is unsatisfiable, T_P is fault-free as long as P runs within the assumed bound. Given that various data values in a given program must be encoded as Boolean values and variables in a trace formula in propositional logic, there is a vast increase in the formula size. To overcome the problem of size explosion and inefficiency of verification caused by this, SMT (satisfiability modulo theories such as integer, real number, and array) solvers have been used instead of SAT solvers [4], [25].

We can see that SAT-based and SMT-based approaches are applied to the problem of *fault localization*. Jose and Majumdar [22], [23] has applied a SAT-based approach to that problem. The fault localization for a program P and the property of an undesirable condition H is to compute a maximal set of statements in P that can remain unchanged for the program to become correct w.r.t. H, or equivalently, to identify a minimum set of statements in P as a fragment (or slice) of the program containing an undesirable condition defined by H. Bug-Assist [22] constructs a trace formula for the pair of a program P and an input I to P (I is a counterexample generated by CBMC for P) and a minimum set of statements is obtained by a partial Max-SAT solver. Könighofer [23] also applies Max-SMT to the fault localization problem and considers multiple inputs simultaneously. Christ [10] improved the fault localization method by a partially flow-sensitive static analysis for developing a formula. Lamraoui [26], [27] further extended the method and implemented a full flow-sensitive fault localization tool SNIPER, which can detect and locate multiple faults. Our method also uses a Max-SMT solver to locate a fault in the model of an automotive control system. The difference between the previous studies and ours is that in our setting, a fault is not just a subset of statements in a pro-

gram but a series of signals in a system. Moreover, we introduce cushion variables to concisely represent a fault in a logical formula.

Expressing a fault as an illegal assignment of values to variables in a logical formula can be found in the previous studies [5], [16]. An advantage of the method proposed in this paper compared to the previous studies is that we separate the original variables and cushion variables so that a Max-SMT solver can automatically identify the fault as soft clauses $u_j^{(i)} = u_j^{c(i)}$ (the value of an original variable equals that of the corresponding cushion variable) that are not satisfied in a maximal solution. Another advantage of the proposed method is that we can deal with real-time properties. Given a set of fault-tolerable states (or the closure) C as well as a set of fault-free states S such that $S \subseteq C$, a system is defined to be fault-tolerant if and only if whenever a system transits to a state in C, the system can eventually return to a state in S [5], [16]. For a real-time system such as an automotive control system, it is further required that if the system falls into a state in C, it must return to a state in S within a certain time or certain steps. To verify that such a requirement is met, we define a property of an undesirable condition as not a property of a single state but that of consecutive states of certain length, depending on an application.

We can see some studies on the analysis of fault trees by using model checking techniques. For example, Schäfer [34] used duration calculus with liveness and Thums [35] used clocked CTL as logical frameworks for model checking. Bozzano [9] proposed a forward and backward reachability analysis in symbolic model checking for fault tree construction. He applied a dynamic cone of influence technique to the backward analysis. However, his study is based on a finite-state model checking and does not scale when a model contains data variables. STPA regards a system as a set of control loops and identifies safety requirements and constraints at the system level. Some case studies at the system level have been reported [15], [20]. In contrast, Abdulkhaleq [1] proposed a formal verification method at the code level with respect to the requirement obtained at the system level and reported a case study on a formal verification that a cruise control system satisfies a safety requirement using a symbolic model verifier (SMV). However, the formal model constructed by Abdulkhaleq is rather small and the safety requirement is not a detailed one as in the case of the properties considered in this paper.

Automated hazard analysis in the automotive field by using the partial Max-SAT was first presented by the authors in the paper [17]. Our previous paper [33] proposes an automating safety analysis method for automotive control systems using Unsafe Control Action (UCA) in the STPA process. This paper improves and extends our previous method [33] as follows. We present UDCs as an abstraction of both UCAs in Ref. [33] and hazardous states and extend $n\text{-}UCA_F^{\leq K}$ in Ref. [33] to $n\text{-}UDC_F^{\leq K}$, so that we specify constraints on n-consecutive states of the system in a more general manner. To demonstrate that our method in this paper is applied to the existing analysis frameworks, we conducted a new case study on a system designed by FTA in addition to STPA in Ref. [33]. We presented the new experiments to show the scalability of our method and how it contrasts with other verification methods.

6. Conclusion

This paper proposes a method to automate the identification of unsuitable behavior due to signal disturbances in automotive control systems by using weighted partial Max-SMT solvers. Our method is useful, especially when dealing with intermittent multi-signal disturbances that are difficult to locate manually. The system behavior is characterized by trace formulae modified with extra variables, called *cushion variables*, to model faulty signal values. A modified trace formula with the constraints leading to an undesirable condition becomes satisfiable when the cushion variables have faulty signal values. Signal disturbances are detected as a set of negated equations between signal variables and cushion variables in value assignments to the modified trace formula. By defining these equations as soft clauses, signal disturbances are automatically detected by checking the satisfiability of the modified trace formula together with undesirable conditions using a partial Max-SMT solver, such as Yices. In applying the solver, weights assigned to soft clauses are used to control the order of detection of signal disturbances.

We apply our method to a simplified virtual automotive control system with three ECUs, including cruise control. The proposed method successfully identifies signal disturbance patterns, leading to hazardous states in the FTA and UCAs in the STPA process. In FTA, our proposed method obtains all types of signal disturbance patterns, leading to the hazardous state, which cause undesirable acceleration forces, braking forces, and transmission states. Furthermore, our method is successful in detecting intermittent multi-signal disturbances that are difficult to enumerate manually within a reasonable time. The observation of the obtained signal patterns by the proposed method can often make engineers discover key signals to prevent the system from going to the behavior satisfying the UDC property, as shown in Section 4.2. By referring the key signals to clue, the engineers can redesign the system architecture and / or build a real-time monitoring function that observes the data in the system and gives a certain action to the system before reaching the undesirable condition. Furthermore, in automotive control systems, each signal does not have an equal probability of being disturbed. The proposed technique can take these probabilities into consideration by putting different values to the weights of the soft clauses in Eq. (2).

The remaining challenges in this study are as follows. An appropriate boundary K for unrolling loops can be considered as a scalability parameter. For periodic behavior, finding an appropriate value of K is possible. Moreover, the compositional extension of the proposed method is required to deal with all automotive control systems, with an appropriate value of K that is very large.

We will consider ways to analyze practical and efficient measures for ensuring safety based on the proposed method in the future. It is very difficult to build perfect countermeasures that prevent specific signals from being disturbed, because the system has uncontrollable factors including sensor noise. We plan to use more relaxed hard constraints, such as $|u_j^{(i)} - u_j^{c(i)}| < \epsilon$ instead of

$u_j^{(i)} = u_j^{c(i)}$, to analyze the effect of introducing the countermeasures of the safety. Furthermore, we plan to investigate a method to use the time-series information from signal disturbance patterns to design sophisticated countermeasures.

References

[1] Abdulkhaleq, A. and Wagner, S.: A Software Safety Verification Method Based on System-Theoretic Process Analysis, Next Generation of System Assurance Approaches for Safety-Critical Systems, *Proc. SAFECOMP Workshop*, pp.401–412 (2014).

[2] Abdulkhaleq, A. and Wagner, S.: Experiences with Applying STPA to Software-Intensive Systems in the Automotive Domain, *Proc. STAMP Workshop* (2013).

[3] Argelich, J.: Max-SAT Formalisms with Hard and Soft Constraints, *AI Communications*, Vol.24, No.1, pp.101–103 (2011).

[4] Armando, A., Mantovani, J. and Platania, L.: Bounded Model Checking of Software Using SMT Solvers Instead of SAT Solvers, *Intl. J. Software Tools for Technology Transfer*, pp.69–83 (2009).

[5] Arora, A. and Gouda, M.G.: Closure and Convergence: A Foundation of Fault-tolerant Computing, *IEEE Trans. Software Engineering*, Vol.19, No.11, pp.1015–1027 (1993).

[6] Ball, T. and Rajamani, S.K.: Automatically Validating Temporal Safety Properties of Interfaces, *Proc. Intl. SPIN Workshop*, pp.103–122 (2001).

[7] Biere, A., Cimatti, A., Clarke, E.M. and Zhu, Y.: Symbolic Model Checking without BDDs, *Proc. Intl. Conf. Tools and Algorithms for Construction and Analysis of Systems*, pp.193–207 (1999).

[8] Biswas, S., Tatchikou, R. and Dion, F.: Vehicle-to-vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety, *IEEE Communications Magazine*, Vol.44, No.1, pp.74–82 (2006).

[9] Bozzano, M., Cimatti, A. and Tapparo, F.: Symbolic Fault Tree Analysis for Reactive Systems, *Proc. Intl. Symp. Automated Technology for Verification and Analysis*, pp.162–176 (2007).

[10] Christ, J., Ermis, E., Schäf, M. and Wies, T.: Flow Sensitive Fault Localization, *Proc. Intl. Conf. Verification, Model Checking, and Abstract Interpretation*, pp.189–208 (2013).

[11] Clarke, E., Kroening, D. and Lerda, F.: A Tool for Checking ANSI-C Programs, Tools and Algorithms for the Construction and Analysis of Systems, pp.168–176, Springer (2004).

[12] Department of Defence: *Procedure for Performing a Failure Mode Effect and Criticality Analysis*, United States Military Procedure, MIL-P-1629 (1949).

[13] Dutertre, B. and Moura, L.D.: The YICES SMT Solver (online), available from ⟨http://yices.csl.sri.com/⟩ (accessed 2019-05-07).

[14] Ericson, C.: Fault Tree Analysis–A History, *Proc. Intl. System Safety Conference*, (1999).

[15] Fleming, C.H., Spencer, M., Thomas, J., Leveson, N.G. and Wilkinson, C.: Safety Assurance in NextGen and Complex Transportation Systems, *Safety Science*, Vol.55, pp.173–187 (2013).

[16] Gärtner, F.C.: Fundamentals of Fault-tolerant Distributed Computing in Asynchronous Environments, *ACM Computing Surveys*, Vol.31, No.1, pp.1–26 (1999).

[17] Hattori, S., Yuen, S., Seki, H. and Sato, S.: Automated Hazard Analysis with pMAX-SMT for Automobile Systems, *Pre-proc. Intl. Workshop on Automated Verification of Critical Systems*, (2015).

[18] Henzinger, T., Jhala, R., Majumdar, R. and Sutre, G.: Software Verification with Blast, *Proc. Intl. SPIN Workshop*, pp.235–239 (2003).

[19] IEC 61882: Hazard and Operability Studies (HAZOP studies) - Application Guide (1992).

[20] Ishimatsu, T., Leveson, N.G., Thomas, J.P., Fleming, C.H., Katahira, M., Miyamoto, Y., Ujiie, R., Nakao, H. and Hoshino, N.: Hazard Analysis of Complex Spacecraft Using Systems-theoretic Process Analysis, *J. Spacecraft and Rockets*, Vol.51, No.2, pp.509–522 (2014).

[21] ISO 26262: Road Vehicles - Functional Safety (2011).

[22] Jose, M. and Majumdar, R.: Cause Clue Clauses: Error Localization Using Maximum Satisfiability, *ACM SIGPLAN Notices*, Vol.46, No.6, pp.437–446 (2011).

[23] Könighofer, R. and Bloem, R.: Automated Error Localization and Correction for Imperative Programs, *Proc. Intl. Conf. Formal Methods in Computer-Aided Design*, pp.91–100 (2011).

[24] Kraig, T.S.: Using FMEA to Improve Software Reliability, *Proc. Pacific Northwest Software Quality Conference* (2013).

[25] Lal, A., Qadeer, S. and Lahiri, S.K.: A Solver for Reachability Modulo Theories, *Proc. Intl. Conf. Computer Aided Verification*, pp.427–443 (2012).

[26] Lamraoui, S.M. and Nakajima, S.: A Formula-based Approach for Automatic Fault Localization of Multi-fault Programs, *J. Information Processing*, Vol.24, No.1, pp.88–98 (2016).

[27] Lamraoui, S.M. and Nakajima, S.: A Formula-based Approach for Automatic Fault Localization of Imperative Programs, *Proc. Intl. Conf. Formal Engineering Methods* (2014).

[28] Lawley, H.G.: Operability Study and Hazard Analysis, *Chemical Engineering Progress*, Vol.70, No.4, pp.45–56 (1974).

[29] Leveson, N.G.: *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press (2011).

[30] Leveson, N.G.: A Systems-Theoretic Approach to Safety in Software Intensive Systems, *IEEE Trans. Dependable and Secure Computing*, Vol.1, pp.66–86 (2004).

[31] Moura, L.D. and Bjorner, N.: *Satisfiability Modulo Theories: An Appetizer, Formal Methods: Foundations and Applications*, pp.23–36, Springer (2009).

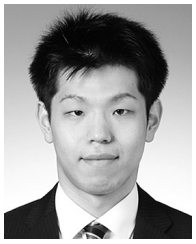
[32] Redmill, F., Chudleigh, M. and Richard, J.C.: *System Safety: HAZOP and Software HAZAOP*, Wiley-Blackwell (1999).

[33] Sato, S., Hattori, S., Seki, H., Inamori, Y. and Yuen, S.: Automating Time Series Safety Analysis for Automotive Control Systems in STPA Using Weighted Partial Max-SMT, *Proc. Intl. Workshop on Formal Techniques for Safety-Critical Systems*, pp.39–54, Springer (2016).

[34] Schäfer, A.: Combining Real-time Model-checking and Fault-tree Analysis, *Proc. Intl. Symp. Formal Methods Europe*, pp.522–541 (2003).

[35] Thums, A. and Schellhorn, G.: Model Checking FTA, *Proc. Intl. Symp. on Formal Methods Europe*, pp.739–757 (2003).

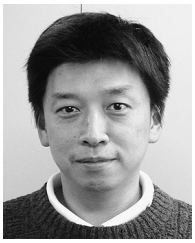
Shuichi Sato was born in 1967. He received his M.S. degree from Osaka University in 1992 and has been engaged in Toyota Central R&D Labs., Inc. He received his Ph.D. degree from Osaka University in 2005. He was a visiting professor at Nagoya University from 2014 to 2018. His research interests are system safety and reliability. He is a member of the Japan Society of Mechanical Engineers.



Shogo Hattori was born in 1992. He received his M.S. degree from Nagoya University in 2016. His research interest is system safety and reliability.

Hiroyuki Seki received his Ph.D. degree from Osaka University in 1987. He was an Assistant Professor, and later, an Associate Professor in Osaka University from 1987 to 1994. In 1994, he joined Nara Institute of Science and Technology, where he was a Professor during 1996 to 2013. Currently, he is a Professor in Nagoya University. His current research interests include formal language theory and formal approach to software development.

Yutaka Inamori was born in 1969. He received his M.S. degree from Kyoto University in 1994 and has been engaged in Toyota Central R&D Labs., Inc. His research interests are software engineering and socio-technical system engineering. He is a member of Information Processing Society of Japan.



Shoji Yuen was born in 1963. He received his Dr. degree of Engineering. from Nagoya University in 1997. He has been a Professor at the Graduate School of Informatics of Nagoya University since 2007. His research interests are theories and applications of concurrency, especially for communicating software systems. Based on the theoretical framework of communicating processes, he has been working on the additional notion of data, probability, and time to extend the application of the framework. He is a member of Information Processing Society of Japan, Japan Society for Software Science and Technology, and the Association for Computing Machinery.