Wang *et al.*
EURASIP Journal on Advances in Signal Processing (2022) 2022:30
https://doi.org/10.1186/s13634-022-00853-8

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

A framework of safety analysis with temporal feature based on MBSA and case study for ACC system

Lisong Wang*, Qin Zhang and Jun Hu

*Correspondence:
wangls@nuaa.edu.cn
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 29 Jiangjun Avenue, Nanjing 211106, China

Abstract

The safety of automotive Adaptive Cruise Control (ACC) system is of great significance to prevent fatigue driving, improve driving comfort, reduce accident rate and promote the development of intelligent transportation and autonomous driving technology. However, the current safety analysis of ACC lacks consideration of the temporal dynamic property, so it is necessary to establish a set of safety analysis methods to consider the temporal characteristics. This paper proposes a new safety analysis method based on MBSA framework and introduces temporal features. Altarica3.0 is a high-level modeling language for safety analysis, and its basic mathematical form is Guardian Transformation System (GTS). In this paper, we outline an analysis approach that converts failure behavioral models (GTS) to temporal fault trees (TFTs), which can be analyzed using Pandora a recent technique for introducing temporal logic to fault trees. However, like classical fault tree analysis, TFT analysis requires a lot of manual effort, which makes it time consuming and expensive. In order to improve the safety of the system, the proposal extends Bayesian Networks with Pandora and results to dependability analysis with temporal relationships to provide more reliable basis for safety design. As a typical case study, the safety analysis method proposed in this paper is applied to the safety analysis of adaptive cruise system, and the results show the effectiveness of the proposed method. Furthermore, it also provides new technologies for the automation and intelligence of safety analysis for smart internet of vehicle.

Keywords: ACC, GTS, Reachable graph, Pandora temporal fault tree, BNs

1 Introduction

With the development of automatic driving technology, the application of adaptive cruise control reduces driving intensity to a certain extent, which is an important part of vehicle assistance technology. It is a new generation of advanced driving assistance system developed on the basis of cruise control system. As a safety–critical system [1], the functional dependence relationship is complex and the fault propagation cause relationship is difficult to determine, so it is urgent to analyze its safety reliability. Systems in different fields have their own strict safety standards in the design and production process, examples include HAD 102/16 (computer-based Safety Critical Systems software for Nuclear Power Plants) for nuclear power and HB for aviation 6781-1993 (Standard

for Basic Requirements of Software Safety Assurance), GB/T34590.10-201 (Functional Safety of Road Vehicles), ARP4761 (Criteria and Methods for safety Assessment process of civil airborne systems and equipment) [2], etc. Strict compliance with safety standards at all stages of system design can greatly reduce the probability of system failure. In the process of System Safety Assessment, the main steps include Functional Hazard Analysis (FHA) [3], Preliminary System Safety Assessment (PSSA) [4], and System Safety Assessment (SSA) [5]. Traditional safety analysis methods mainly include fault tree analysis (FTA) [6], Markov analysis (MA) [7], failure mode and effect analysis (FMEA) [8] and common cause analysis (CCA), etc. But with the passage of time, the traditional methods of risk assessment of excessive dependence on low level form of modeling, system specification and related models between cannot match very well, when very small changes in the system specification, the relevant safety model also need to conduct a comprehensive assessment, a lot of manpower and material resources, at the same time. When the analyst converts the actual physical model into the corresponding fault tree, there is a great subjective influence, and there may be the consequence of inaccurate modeling.

To solve these problems, a model-based safety analysis (MBSA) method is proposed. It is actually the reliability Engineering branch of Model-based System Engineering (MBSE) [9, 10]. Model-based system safety analysis can be divided into two steps: system modeling and related safety analysis for the model. To be specific, the first step is to write the system model in the high-level modeling form, which can completely express the function and structure of the original system. The second step is to evaluate the model directly, or compile the model into a low-level model for evaluation, such as FTA, MA, etc.

AltaRica is a kind of safety for fault modeling language, to the guard conversion system (GTS) as the core [11–13]; the use of reusable object-oriented language description node and fault through the node connected to the interface between embedded systems describes the interaction between the node and system information, including data flow elements support interaction between components, there are dependencies between components.

AltaRica3.0 is a new version of the AltaRica language [14]. Its new underlying mathematical model is the guardian transformation system (GTS) [15], which improves the expression ability of previous versions by introducing a fixed point mechanism to stabilize the theological value after each transition. This mechanism allows the design of causal components and the handling of systems with immediate cycles. In the framework of AltaRica3.0, this paper proposes an automatic construction method of Pandora timing fault tree based on extended fault tree model. The method is verified by modeling and simulation of typical structural model of ACC system. After that, quantitative analysis of Pandora timing fault tree was carried out. Since traditional analysis methods are prone to state explosion and fault events are interdependent, reliability analysis of Pandora temporal fault tree was carried out in this paper based on Bayesian network. The Pandora temporal fault tree analysis and probability calculation provide a more reliable basis for the safety of system and give system designers an idea where to invest more design efforts to improve safety. Finally, the design defects of the system design model are tracked by combining the verification results and traceable information model.

2 Background

2.1 A guardian transformation system (GTS)

Altarica3.0 is a new version of the Altarica language. Its new underlying mathematical model is the guard transformation system (GTS), which is a state/transition form specifically designed for safety analysis. The guardian transition system (GTS) is an automaton whose state is represented by a variable assignment, that is, the variable and its value. A change in state is represented by an event-triggered transition. It can also represent the flow through the network and synchronize events to describe remote interactions between components of the system under study. The formal definition of flattened GTS is as follows [16]:

Definition 1 GTS is composed of a quintuple $< V, E, T, A, \iota >$, where V is the set of variables, divided into two groups of disjoint state variable S and flow variable F, that is, $V = S \cup F$; E is a set of events; T is a set of transformations, which are represented as a triple $< e, G, P >$, where e is an event in E; G is a guard condition (a Boolean expression) built on variable V, and P is an instruction built on variable V, known as an action or post condition. Thus, in general, the transformation T is expressed as e: $G \rightarrow P$; A is a set of assertions, assertions are instructions built on the variable V; ι is the assignment of the variable V, which is the initial or default assignment.

To elaborate on the composition of the GTS, Fig. 1 shows a graphical representation of a cooling system and its flattened model representation.

Concrete components in the cooling system are abstracted into classes: tank, pump and reactor, respectively. These classes interact with each other through variables. Tank instantiates object T outflow coolant as outflow variable of the whole system, and two instantiated objects P1 and P2 of pump have two variables, respectively: outflow from tank, pump and reactor. Inflow of input variable, inflow of reactor is the same as outflow of P1 and P2. The system has to keep at least one pump running so that the coolant can be used to cool the reactor properly, but the pump could fail or the tank could be empty, which could cause the entire system to fail.

The GTS model obtained by flattening the hierarchical information of the cooling system is shown in Fig. 2.

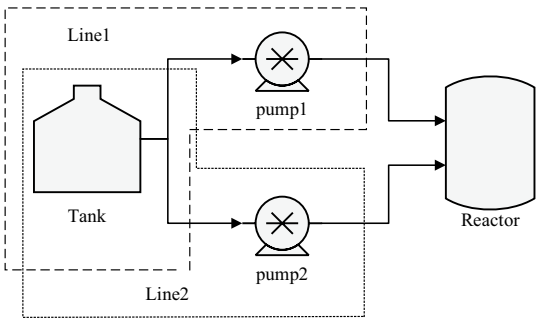


Fig. 1 A cooling system

```
block CoolingSystem
    Boolean T.isEmpty (init = false);
    Boolean T.outFlow (reset = false);
    RepairableState LineOne.POne.s,
    LineTwo.PTwo.s (init = WORKING);
    Boolean LineOne.POne.inFlow,
    LineOne.POne.outFlow (reset = false);
    Boolean LineTwo.PTwo.inFlow,
    LineTwo.PTwo.outFlow (reset = false);
    Boolean Reactor.inFlow (reset = false);
    event T.getEmpty;
    event LineOne.POne.failure, LineOne.POne.repair;
    event LineTwo.PTwo.failure, LineTwo.PTwo.repair;
transition
    T.getEmpty: not T.isEmpty -> T.isEmpty := true;
    LineOne.POne.failure: LineOne.POne.s == WORKING -> LineOne.POne.s := FAILED;
    LineOne.POne.repair: LineOne.POne.s == FAILED -> LineOne.POne.s := WORKING;
    LineTwo.PTwo.failure: LineTwo.PTwo.s == WORKING -> LineTwo.PTwo.s := FAILED;
    LineTwo.PTwo.repair: LineTwo.PTwo.s == FAILED -> LineTwo.PTwo.s := WORKING;
assertion
    T.outFlow := if not T.isEmpty then true else false;
    LineOne.POne.inFlow := T.outFlow;
    LineTwo.PTwo.inFlow := T.outFlow;
    LineOne.POne.outFlow := if LineOne.POne.s == WORKING then LineOne.POne.inFlow else false;
    LineTwo.PTwo.outFlow := if LineTwo.PTwo.s == WORKING then LineTwo.PTwo.inFlow else false;
    Reactor.inFlow := LineOne.POne.outFlow orLineTwo.PTwo.outFlow;
end
```

Fig. 2 The GTS model of the cooling system

2.2 Pandora temporal fault tree

Pandora defines three time gates: Priority-AND gate (PAND), Priority-OR gate (POR) and Simultaneous-AND gate (SAND) to extend the classic fault tree [17]. The temporal fault tree symbols of the three gates are shown in Fig. 3. PAND gates are not new and have been used in FTA since the 1970s [18]. The symbol "$<$" used to represent the PAND gate in the logical expression, such as $A<B$ represents A PAND Y, where both A and B are failure events. the Priority-OR (POR, symbol '|'), which as described earlier models a priority situation where one event must occur first and other events may or may not occur subsequently. The Simultaneous-AND (SAND, '&') gate, which represents simultaneous occurrence of events, e.g., as a result of a common trigger. In this article, we use "$+$" for OR gate and "." for the AND gate.

Pandora allows more than one form of reduction. As well as removal of redundancies, e.g., $A<B+A|B \Leftrightarrow A|B$ (where A and B are fault tree events), Pandora also allows reduction via the recognition and removal of contradictions and by means of 'completion'—conversion of temporal expressions into static Boolean expressions. For

- (A|B).B ⇔ A<B;
- A|B ⇔ A|B;
- (A|B). (A|C) ⇔ A|B|C;
- A|(B+C) ⇔ A|B|C.

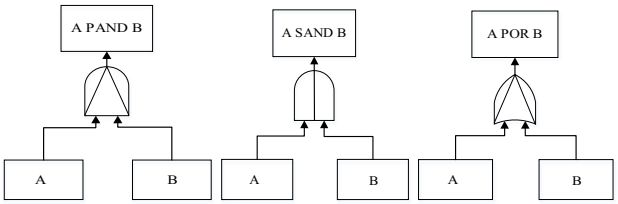


Fig. 3 Pandora gates

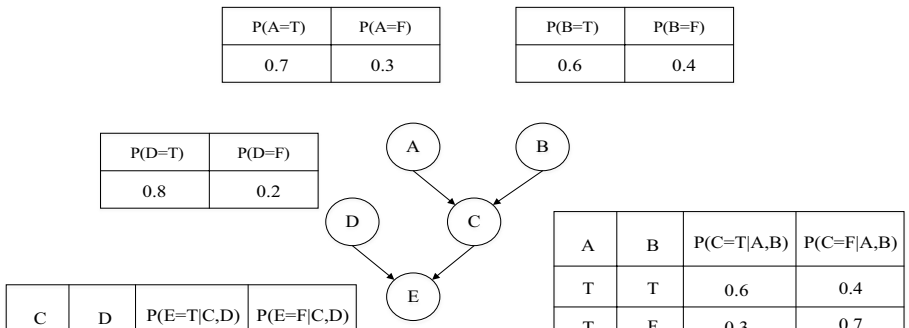
full details about Pandora's temporal laws, we refer the reader to [18], but four laws in particular will be useful:

- $(A|B).B \Leftrightarrow A<B$;
- $A|B \Leftrightarrow A|B$;
- $(A|B).\ (A|C) \Leftrightarrow A|B|C$;
- $A|(B+C) \Leftrightarrow A|B|C$.

2.3 Bayesian network

Bayesian network (BN) is a directed acyclic graph, which is composed of nodes and directed edges. Nodes represent variables and directed edges represent relationships between nodes. A BN model contains a qualitative and a quantitative part (see Fig. 4). Bayesian network is a directed acyclic graph representing a group of random variables and their conditional correlation [19]. The node of BN represents the random variable, and the directed arc represents the dependence or causality between nodes. In the Bayesian network, if there is an arc from node X and node Y, node X is said to be the parent node of another node Y. Has direct influence to the subsidiary to the parent node, it is defined as a Pr {Xi | parent (Xi)} to quantify the influence of the subsidiary to the parent node. A node is a root node if it has no parents, and a leaf node if it has no children.

Discrete-time Bayesian network (DTBN) is a BN with N nodes is generally represented $N=<<X, T_n, E>, P>$. The nodes $X=\{X_1, \ldots, X_N\}$ in the figure represent variables, and all phenomena such as component state values and personnel operations can be abstracted into the node variable. $T_n=\{[t_0, t_1), \ldots, [t_{n-1}, t_n), [t_n, \infty\}$ is a division of the mission, its elements represent the interval where the variable fails at the time. Directed edge E characterizes the causal relationship between node variables. In (X_i, X_j), X_i is the parent of X_j, and X_j is the child of X_i. A node without a parent is a root node, and a node without children is a leaf node. Digraphs imply conditional independence. P represents conditional probability distribution (CPD). It can be seen from the implied independence assumption of the Bayesian network that the joint probability distribution including all nodes can be obtained on



P(A=T)	P(A=F)
0.7	0.3

P(B=T)	P(B=F)
0.6	0.4

P(D=T)	P(D=F)
0.8	0.2

A	B	P(C=T|A,B)	P(C=F|A,B)
T	T	0.6	0.4
T	F	0.3	0.7
F	T	0.8	0.2
F	F	0.01	0.99

C	D	P(E=T|C,D)	P(E=F|C,D)
T	T	0.9	0.1
T	F	0.2	0.8
F	T	0.7	0.3
F	F	0.05	0.95

Fig. 4 BN model

the premise of knowing the prior probability distribution of the root node and the conditional probability distribution of the non-root node. The joint probability distribution function of all nodes is:

$$\begin{aligned} P(X1,X2,X3,X4,X5,X6) &= \prod_{i=1}^{6} P(X_i|\text{parent}(X_i)) \\ &= P(X6|X5)P(X5|X3,X4)P(X4|X1)P(X3|X1,X2)P(X2)P(X1) \end{aligned} \tag{1}$$

If the top event E_T occurs within task time T, the time when E_T occurs must be within an interval of [0,△], [△,2△], ..., [$(n-1)$△, n△]. Therefore, the probability of E_T occurring within task time T is

$$P(T) = \sum_{0<x\le n} P(E_T = [(x-1)\ \triangle, x\ \triangle)] \tag{2}$$

$P(E_T)$ can be calculated directly by a joint probability distribution, namely:

$$\begin{aligned} P(E_T) &= P([(x-1)\ \triangle, x\ \triangle] \\ &= \sum_{E_1,\ldots,E_{M-1}} P(E_1 = e_1,\ldots,E_{M-1} = e_{M-1}, E_T = [(x-1)\ \triangle, x\ \triangle]) \end{aligned} \tag{3}$$

In Eq. (3), ($1<i\le M-1$) corresponds to non-leaf nodes in DTBN (i.e., intermediate and base events in the dynamic fault tree), M is the number of nodes, $e_i \in \{[0, \triangle], [\triangle, 2\triangle], \ldots, [(n-1)\triangle, n\triangle], [T, +\infty]\}$were used to characterize the failure interval of E_i.

According to Eqs. (2) and (3), the probability of E occurring within task time T can be obtained, namely:

$$P(T) = \sum_{0<x<n} \sum_{E_1,\ldots,E_M} P(E_1 = e_1, E_2 = e_2, \ldots, E_{M-1} = e_{M-1}, E_T = [(x-1)\ \triangle, x\ \triangle]) \tag{4}$$

3 Method

3.1 A framework for automatic generation of Pandora temporal fault tree based on GTS

In general, this section gives the method framework and several steps of how to automatically generate the Pandora temporal fault tree based on the flattened GTS model of Altarica3.0 as shown in Fig. 5. The algorithm GTS to Pandora temporal fault tree (GTS2PTFTA) design involved in each step is then explained in detail:

(1) Read the flattened GTS model description file and construct the corresponding GTS object.
(2) The input GTS object is divided into a set of independent GTS and an independent assertion.
(3) Iterate the independent GTS set to obtain the corresponding reachable graph (state machine graph) of each independent GTS, so as to facilitate the subsequent generation of event sequences with timing characteristics according to the state machine graph design algorithm.
(4) Compile each reachable graph and obtain the logical formula of the independent GTS with the temporal relationship. Find all possible paths between two nodes in the reachable graph and design the algorithm to update the possible priority relationship into the path. The algorithm described in Step 4 is corresponding to the logical formula containing time sequence relation of the 3.3 sub-GTS model reachable graph. The AND gate operator is further annotated as: search all paths from state S0 to the other states of the graph. Events occurring along path π are transformed into the relationship of events. Each state of the graph is first associated

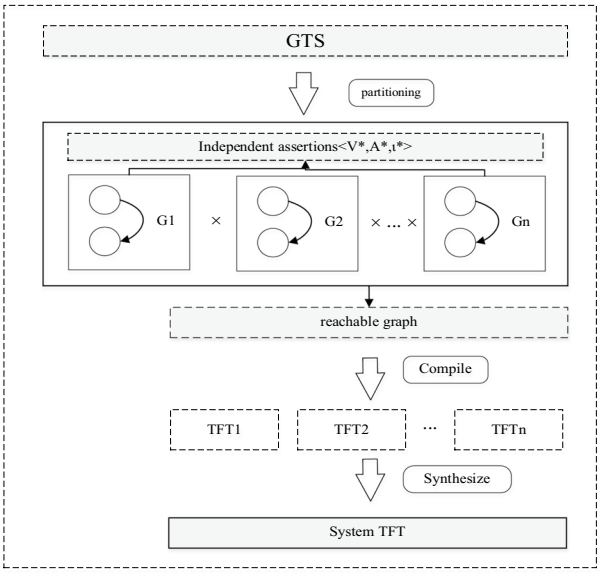


Fig. 5 Compilation of GTS into temporal fault trees or event sequences

with a sequence disjunction obtained through the compile path, then each pair (variable values) associated with a sequence of disjunction, including variable uses this value, secondly, traverse to search path, looking for the same end node status on different paths to share events, update priority or temporal sequence in the original sequence of disjunction, after exhaustive recursion and, at the same time, and preference or conversion between matching rules, generating a sequence of events with a temporal relationship.

(5) Synchronize the independent assertions obtained by partition with the formulas obtained by compilation of each independent GTS reachable graph and obtain the set of temporal relational event sequences of the whole model, which is composed of the Pandora temporal fault tree of this GTS model.

3.2 GTS model preprocessing and partitioning

A partition operation on a GTS instance object. Considering the large scale of the system model, after the GTS instance object is obtained, in order to simplify the subsequent steps, a more efficient partition algorithm is adopted here to improve the efficiency of the algorithm, so as to cope with the large scale of the system model. The partitioning algorithm divides a model into multiple components and modules, and then processes each component and module individually. When the fault tree with time sequence relationship is finally obtained, it is processed together with the data results, which simplifies the intermediate steps of the whole algorithm framework process.

3.3 Construction of the accessibility diagram of the sub-GTS model

The corresponding reachable graphs of each independent GTS were obtained. In this step, we mainly process each independent GTS to obtain the relevant reachable chart. A reachable graph contains a set of nodes, and a node exists in the form of a set of variables, and the current system state is represented by the values of the current variables.

3.4 Algorithm for compiling reachable graph to generate temporal expression of sub-GTS model

An accessibility graph is a state machine with a finite number of states. It may change state as an event occurs, but at each moment, it is only in one state. In the previous step, the algorithm obtained the reachable graph of each GTS. To obtain a logical formula containing temporal relationships, in short, find all possible paths between two nodes in the reachable graph and design an algorithm to update the possible priority relationships to the paths. The algorithm mainly uses priority or gate (POR) to distinguish fault sequences. A gate can represent a situation in which one event takes precedence over others and must occur first, but does not specify that other events must also occur, for example, if A occurs and B does not occur, or if A occurs and B occurs but A occurs first, then POR B is true. In this step, operator "." is defined as a sequence of "and" door

and operator "+" is defined as a sequence of "or" door and operator "<" is defined as a sequence of "priority-AND gate," operator "|" is defined as "priority-OR" gate and the operator "ø" represents a collection of sequences. The AND gate operator is further labeled as search all paths from state S0 to other states of the graph. Events occurring along path π are converted to and relationships of events. Each state of the graph is first associated with the sequence disjunction obtained through the compile path, then each pair (variable values) associated with a sequence of disjunction, including variable uses this value, secondly, traverse to search path, looking for the same end node status on different paths to share events, update priority or temporal sequence in the original sequence of disjunction, after exhaustive recursion and, at the same time, and preference or conversion between matching rules, Generates a sequence of events with timing relationships.

Definition 2 Reachable Graph (RG).

A reachable graph is a quadruple (S, Σ, δ, s_0) where:

- S is a finite set of states.
- Σ is a finite set of events, such that $S \cap \Sigma = \emptyset$.
- δ is a partial function: $S \times \Sigma \to S$, s.t. for $(u, u') \in S^2$, and $e \in \Sigma$, $u' = \delta(u, e)$ *iff* e is incident from u to u', and we write it as: $u \to u'(e)$.
- s_0 is the initial state.

Definition 3 Paths Set.

Let P be the set of all paths in the RG,

$P = \{\pi | u \to u'(\pi), (u, u') \in S^2\}$,We write $u \to u'$ iff $\exists \pi \in P$ s.t. $u \to u'(\pi)$.

Definition 4 Forward and backward incidence sets.

For any state $u \in S$, let ΣuI (resp. ΣuJ) be the set of events incident from u (resp. incident to u),

$\Sigma uI = \{e \in \Sigma | \ \exists u' \in S \ s.t. \ u \to u'(e)\}$, $\Sigma uJ = \{(e, u') \in \Sigma \times S | \ u' \to u(e)\}$.

Definition 5 Set of final states.

Let F be the set of the final states, $F = \{f \in S | \ \Sigma fI = \emptyset\}$.

The algorithm and pseudocode of compiling the accessible graph of independent GTS to obtain the time-series relation formula are as follows:

Algorithm 1 . **Input:** Reachable graph; **Output:** Expression of a fault state

1: $P \in$ reachable graph; $F \in S$; let $\Phi = \emptyset$;

2: **for each** $s \in F$ **do**

3: get P_s from P where $P_s = \{\pi \in P |\ s_0 \longrightarrow s\ (\pi)\}$

4: let $n = |P_s|$; let $\pi_i \in P_s$ $(1 \le i \le n)$; let len_i=length(π_i) $(1 \le i \le n)$

5: let $Seq\pi_i = \{s_{i0}, s_{i1}, \ldots, s_{ileni}\}$; let $\varphi_s = \bigvee_{1 \le i \le n} (\wedge_{1 \le j \le leni} e_{ij})$

6: **for each** s_{ij} in $Seq\pi_i$ $1 \le i \le n$, $1 \le j \le leni$, s.t. $|\Sigma s_{ij}\ I| > 1$ **do** let $e = e_{ij}$

7: **for each** $e' = e_{ij}$ in Σs_{ij} I **do** let $t \in S$ s.t. $s_{ij}\ e' \rightarrow t$

8: **if** (($e' \in \{e_{i(j1)}, e_{i(j2)}, \ldots, e_{i1}\}$) **or** ($\exists (y \in \Sigma, \pi_0 \in P, f \in S, v \in S$ in $Seq\pi_0$, $w \in S$ in Seqπ_0) s.t. t $\rightarrow f(\pi_0)$ and $y \in \{e_{ij}, e_{i(j-1)}, \ldots, e_{i1}\}$)) **then** let $e = e|e'$

9: **end if**

10: **end for**

11: replace e_{ij} with e in φ_s

12: **end for**

13: let $\Phi = \Phi \cup \{\varphi_s\}$

14: **end for**

Generates a set Φ of Pandora formula: $\Phi = \{\phi_s | s \in F\}$ one formula ϕ_s for each final state s. These expressions can then be analyzed by Pandora. Transformation algorithms are biased toward increasingly dynamic systems. The best case complexity for checking the necessity of chronological order is $O(n)$, and the worst case complexity is $O(n^2)$, where n is the number of paths from the initial state to the final state in the reachable graph. In the best case, for each divergent path, there exists a shareable event that is related to the direct reachable state of the connection state when the path diverges. In this case, the cost of adding j to each state is the order time of an $O(m^2)$ operation, where m is the degree of j.

3.5 Safety analysis of Pandora temporal fault tree based on Bayesian network

To perform quantitative analysis of temporal failure behavior, Bayesian Network has been integrated with the Pandora temporal fault Tree, a compilation of the AltaRica model-based dependability analysis process.

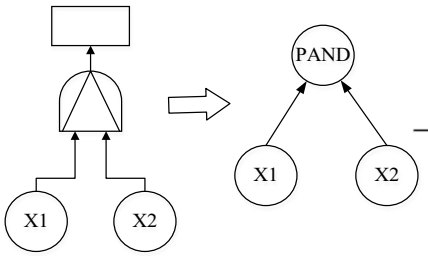
The primary goal of the Bayesian network-based approach is to transform the Pandora TFT into a Bayesian network for quantitative analysis of dynamic systems using BN models. The transformation from TFT to BN is done in two steps. In the first step, the TFT to BN graphic mapping is achieved by converting the base event to the root

node, the intermediate event (logic gate) to the intermediate node, and the top event to the leaf node. In the second step, the prior probability value of the root node is filled according to the failure probability of the basic event, and the conditional probability table of the other nodes is filled according to the type of logical gate they represent to achieve the numerical mapping. Pandora TFT has Boolean gates and time gates. The results of time gates depend on the order in which their input events occur, which is represented by sequence values. On the other hand, the outcomes of Boolean gate do not depend on the order of input events. However, in TFT, the output of a temporal gate can be connected to the input of a Boolean gate and vice versa. Therefore, it is necessary to define temporal behavior for Boolean gates.

To represent sequence values, which are used in Pandora to represent sequencing among events, in this paper, the mission time T is divided into N equal intervals from $t=0$ to $t=T$, where each interval represents a possible non-zero sequence value, during which an event occur. Initially, all components are assumed to be fully operational and therefore all events are given an initial sequence value of 0 (i.e., the component has not yet failed). This value holds until the occurrence of a component failure, i.e., if the component can survive till the end of the mission time then it will continue carrying the sequence value 0. If a component fails in interval 1, then it will have the sequence value 1, but if the component fails in any other interval i where $1<i<N$ then it can have any sequence value in between 1 and i based on its relative position to its immediate predecessor. An event having a sequence value i is considered to be in State i and it has an associated probability value representing the probability of the event being in State i. As the root nodes in the BN represent different basic events, we need to define prior probability tables for the root nodes, where each entry in a prior probability table of a node represents the probability of the respective event being in a particular state. For exponentially distributed failure rate, the probability of a component being failed in the interval $[t1, t2]$ (e.g., in State i) can be obtained by integrating the probability density function of exponential distribution, λe, λt, in the following way:

$$P = \int_{t1}^{t2} \lambda e^{-\lambda t} \mathrm{d}t \tag{5}$$

Once prior probability values are assigned to each root node, conditional probability values are generated for all the intermediate nodes. Remember that Pandora represents the outcome of every gate with a sequence value, showing not only whether that gate is true or false but also the relative order in which the gates become true. This is purely deterministic: the outcome of each gate depends solely on the sequence values of its input events. Consequently, the probability of an intermediate BN node (representing a gate) being in a certain state can either be 0 or 1, depending on the state of its parent nodes. In [20], the authors have shown the translation and CPT generation process for the PAND gate, as seen in Fig. 6. The mapping of a POR gate by diving the mission time into two intervals is shown in Fig. 7. The CPT of the PAND gate resembles its temporal truth table. As seen in the CPT of the PAND gate, the PAND outcome becomes true only in one scenario when the first input (X) is in State 1 and the second input (Y) is in State 2, i.e., they occur in a sequence from left to right. In this case, the state associated

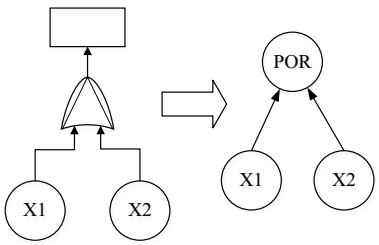


X1	X2	P{X1 PAND X2}		
		state1	state2	state3
State1	State1	1	0	0
State1	State2	0	1	0
State1	State3	0	0	1
State2	State1	1	0	1
State2	State2	0	1	0
State2	State3	0	0	1
State3	State1	0	0	1
State3	State2	0	0	1
State3	State3	0	0	1

X2	state1	state2	state3
Pr{X2}	p0	p1	p2

X1	state1	state2	state3
Pr{X1}	p0	p1	p2

Fig. 6 BN's CPT of PAND gate



X1	X2	P{X1 POR X2}		
		state1	state2	state3
State1	State1	1	0	0
State1	State2	1	0	0
State1	State3	1	0	0
State2	State1	0	1	0
State2	State2	1	0	0
State2	State3	0	1	0
State3	State1	0	0	1
State3	State2	1	0	0
State3	State3	1	0	0

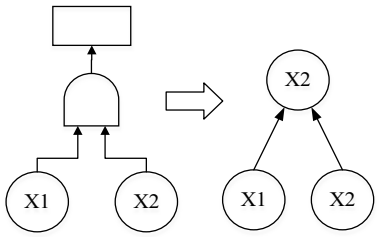
X2	state1	state2	state3
Pr{X2}	p0	p1	p2

X1	state1	state2	state3
Pr{X1}	p0	p1	p2

Fig. 7 BN's CPT of POR gate

with the PAND outcome is State 2 because this is the state when the last input of the PAND becomes true.

Bayesian Network of the AND and OR gates are shown in Figs. 8 and 9, respectively. From the CPT of the AND gate, we can see that the AND outcome becomes true (1s in column State 1 or State 2) when the input events are either in State 1 or State 2. If any of the input event is in State 0 (logical false), then the outcome of the gate is false (1s in the column State 0). So the CPT of the POR, AND and the OR gates are of the same pattern and entries in the table are either 0 or 1, but the positions of the 0s and 1s change according to the logical specification of the gates. Once we have the Pandora TFT of the failure behavior of a system, we can translate the TFT to BN model and subsequently perform predictive reasoning on the Bayesian Network to obtain system unreliability. This is done by following the direction of the BN arcs from the root nodes toward the leaf node. In this process, failure probability data of the root nodes are used to obtain the probability of system failure, i.e., data about causes are used to obtain new belief about the effects. Using the facility of observing the status of a node, we can also perform

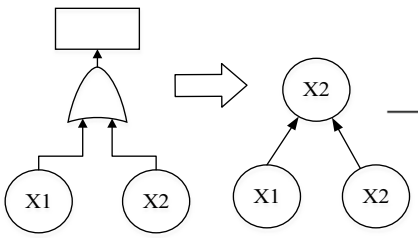


X1	X2	P{X1 AND X2}		
		state1	state2	state3
State1	State1	1	0	0
State1	State2	0	1	0
State1	State3	0	0	1
State2	State1	0	1	0
State2	State2	0	1	0
State2	State3	0	0	1
State3	State1	0	0	1
State3	State2	0	0	1
State3	State3	0	0	1

X2	state1	state2	state3
Pr{X2}	p0	p1	p2

X1	state1	state2	state3
Pr{X1}	p0	p1	p2

Fig. 8 BN's CPT of AND gate



X1	X2	P{X1 OR X2}		
		state1	state2	state3
State1	State1	1	0	0
State1	State2	1	0	0
State1	State3	1	0	0
State2	State1	1	0	0
State2	State2	0	1	0
State2	State3	0	1	0
State3	State1	1	0	0
State3	State2	0	1	0
State3	State3	0	0	1

X2	state1	state2	state3
Pr{X2}	p0	p1	p2

X1	state1	state2	state3
Pr{X1}	p0	p1	p2

Fig. 9 BN's CPT of OR gate

diagnostic analysis on the BN, i.e., reasoning from effects back to their causes. If the analysts have the evidence that the system has failed, then based on this evidence the analysts belief about the failure probability of the components can be updated. That means we now have to put an observation on the leaf node of the BN and work backwards (in the opposite direction of the BN arcs) toward the root nodes to update the probability of root nodes.

In order to facilitate diagnostic analysis, Bayes' theorem in Eq. (1) is used.

4 ACC system design experiment based on method framework

4.1 Description of ACC system

Typical ACC system of automobile is mainly composed of information perception unit, electronic control unit (ECU), execution unit and human–computer interaction

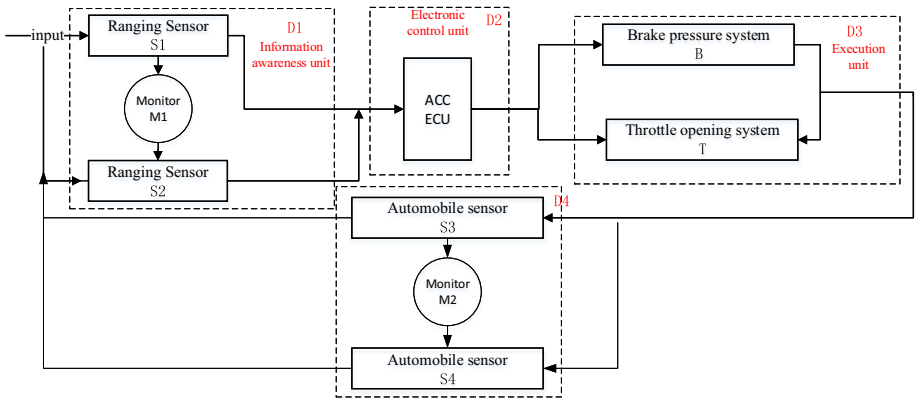


Fig.10 Schematic diagram of ACC system

interface, etc. This paper selects some of these structures as examples for analysis, and the basic composition is shown in Fig. 10.

In the ACC system, ranging radar used to measure the vehicle and the vehicle in front of the car distance, relative velocity and relative acceleration, is one of the key equipment of adaptive cruise control system, is to determine the main components of the cost of the system, its main components including transmitting antenna and receiving antenna, DPS (digital signal processing (DSP)) processing unit, data cables, etc. The electronic control unit in ACC module is composed of large-scale integrated circuits, such as microprocessor (CPU) memory (ROM RAM) input/output interface (I/O) analog-to-digital converter (A/D) and shaping driver, as well as common single-chip microcomputer, according to its memory program and data on the air flow meter and a variety of sensor input information processing judgment, and then output instructions.

The main working principle of the ACC system is Lord radar sensors to detect the target vehicle in front of the car, and to provide the main vehicle and target vehicle electronic control unit (ECU) the relative speed between the relative distance information such as the relative azimuth Angle Electric control unit based on the pilot set the safety car distance and cruising speed, combined with radar transmit information to determine the main vehicle driving state, issue instructions to adjust throttle opening and brake control of main vehicle. The arrow in the figure indicates the direction of signal transmission.

4.2 System modeling and analysis of ACC system

Model-based analysis of system is performed in a compositional fashion. For this reason, the ACC system is divided into smaller subsystems and the subsystems are further divided into smaller subsystems to the level of components. Components are annotated with their failure behavior and their behavior are combined to obtain the behavior of the subsystem possessing them. We will use this system structure as an example to describe the different steps involved in compiling the Guard transformation system into a Pandora temporal fault tree.

```
domain RangingSensorState { ON, OFF, FAILED }

 class RangingSensor

  RangingSensor s (init = ON); // The state variable of the ranging sensor

  Boolean demanded , outFlow , inFlow (reset = false);

  Boolean fail(reset = false);

  event start (delay = 0, expectation = 1 - gamma); // The component started
successfully from the standby state

  event failureOnDemand (delay = 0, expectation = gamma); // Failed to start
component from standby state

  event stop(delay = 0);// The distance sensor itself fails and stops working

  parameter Real gamma = 0.001; // Parameter for standby startup failure

  transition

   start: s == OFF and demanded -> s := ON;

   failureOnDemand: s == OFF and demanded -> s := FAILED;

   stop: s == ON and not demanded -> s :=OFF;

  assertion

   outFlow := s == ON and inFlow;

   fail := if s == FAILED then true else false ;

end
```

Fig. 11 GTS representing a ranging sensor

```
domain ComponentState { WORKING, FAILED}

  class NonRepairableComponent

   ComponentState s (init = WORKING);

   event failure (delay = exponential(lambda)); // Probability of failure

    parameter Real lambda = 0.001;

    transition

        failure: s == WORKING -> s := FAILED;

   end

   class Monitor extends NonRepairableComponent

    Boolean inflow (reset=true);

     Boolean demanded, outflow (reset=false);

     Boolean fail(reset = false);

     event  failure;

     transition

        failure: s == WORKING and demanded -> s := FAILED;

     assertion

        outFlow := s == WORKING and inFlow;

        fail := if s == FAILED then true else false ;

 end
```

Fig. 12 GTS representing a monitor

```
class ACC extends NonRepairableComponent
  ComponentState  s  (init = WORKING);
  Boolean inflow (reset=false);
  Boolean outflow (reset=false);
  Boolean fail(reset = false);
 assertion
  outflow : = if s == WORKING then inflow else false;
  fail := if s == FAILED then true else false ;
end
```

Fig. 13 GTS representing an electronic control unit

The GTS, describing a ranging sensor, is given in Fig. 11 (Set as secondary redundancy component, S1 as the primary component and S2 as the backup component), automobile sensor is similar to:

Monitor component modeling: The monitor is mainly used to monitor the output of the main component. If any omission or failure is detected, the backup component will be activated, as shown in Fig. 12.

The GTS, describing an electronic control unit, is given in Fig. 13.

Brake pressure module and throttle opening module are the same as above. Modeling of the whole adaptive cruise system is shown in Fig. 14.

The whole system model was analyzed: First, the AltaRica3.0 model obtained was flattened; second, the reachable graph of each independent GTS was obtained according to the correlation partition algorithm, as follows:

The reachable graph for the remaining components is similar to the figure above. "O-x" represents the output omission or fault of a component, "X_Failure" represents the failure event of a component, the box represents the status of the component at this time, and the line between the boxes represents the transmission of failure events. For descriptive purposes, any part symbol abbreviates the failure of the corresponding part (e.g., "M1" is denoted as "M1 failure").

The TFT of component S1 is the simplest: $O-A=A$ (i.e., failure of A causes omission of A), as shown in Fig. 15. For the reachable graph of M1 in Fig. 16, the initial state is "WORKING," which means the sensor is ready to activate B. If a failure occurs in the output of S1 (for example, an output omission represented as "O-S1" in the reachable graph) and is detected, the state of M1 changes to "omission detected" (that is, B needs to be activated). However, if M1 fails after waking up S2 (because S2 takes over the functions of S1), the system can continue to operate, so the failure sequence (O-S1 first, M1 second failure) will cause the sensor to enter the "safety failure" state. Conversely, if M1 fails first (into a "premature failure" state), it will not be able to detect subsequent output omissions from S1, so the second sequence will result in a "serious failure" state (since B will not be activated, which means the system is completely faulty). The latter state represents an output omission from M1 (" O-M1 "), where errors propagate through the system (in this particular case leading to complete failure). Therefore, as you can see, from the situation in Fig. 15, we have two different sequences of the same two failures, each of which leads to different failure states

```
Block ACCSystem
 RangingSensor S1 (s.init = WORKING);
 RangingSensor S2 (s.init = STANDBY);
 Sensor S3(s.init = WORKING);
 Sensor S4(s.init = STANDBY);
 Monitor M1, M2(s.init=WORKING);
 ACC acc(s.init=WORKING);
 Braking B (s.init=WORKING);
 Throttle T (s.init=WORKING);
 Boolean failure=D1_failure or D2_failure or D3_failure or D4_failure (reset=false)
 assertion
 // Four parts failure modes
      D1_failure:=S1.fail and S2.fail;
      D2_failure:=acc.fail;
      D3_failure:=B.fail or T.fail;
      D4_failure:=S3.fail and S4.fail;
   // Flags that each component is enabled on demand
      acc.demanded := not D1_failure ;
      B.demanded := not D1_failure and not D2_failure ;
      T.demanded := not D1_failure and not D2_failure ;
      S3. demanded := not D1_failure and not D2_failure and not D3_failure ;
      S4. demanded := not D1_failure and not D2_failure and not D3_failure
and not D3_failure and S3.fail;
      S1. demanded := not D4_failure ;
      S2. demanded := not D4_failure and S1.fail ;
      S1.inFlow:=true;// (The rangeing sensor input stream variable is true)
      // System data flow direction
      S2.input:=M1.output;
      acc.input:=S1.output or S2.output;
      B.input:=acc.output; T.input:=acc.output;
      S3.input:=B.output and T.output;
      S4.input:=M2.output;
end
```

Fig. 14 GTS of ACC system

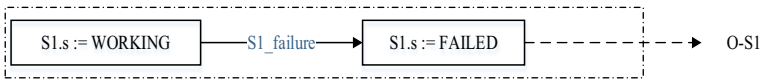


Fig. 15 Reachable graph of S1

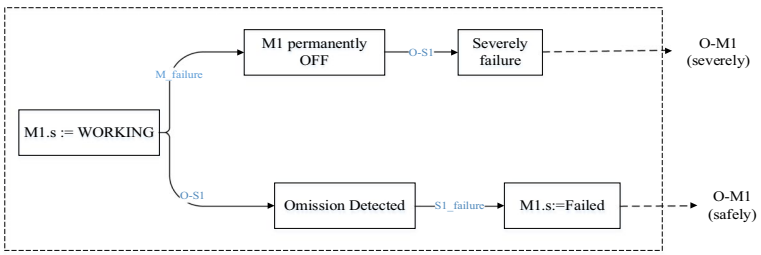


Fig. 16 Reachable graph of M1

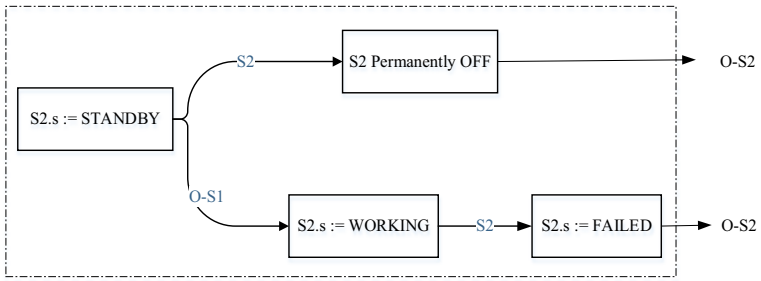


Fig. 17 Reachable graph of S2

of the sensor. One sequence can cause the system to fail, while the other gives it a chance to continue working during backup. Therefore, we will introduce Pandora gates to represent the sequence of events in the fault tree, and then the reachable graph can be transformed into a Pandora temporal fault tree without sacrificing the relevant event sequencing, which affects the outcome. So the above two sequences are converted to:

$$\mathrm{O{-}M1(Safely\ failed)} = (\mathrm{O} - \mathrm{S1}) < \mathrm{M1};\ \mathrm{O{-}M2(Severely\ failed)} = \mathrm{M1} < (\mathrm{O} - \mathrm{S1}) \tag{6}$$

If we apply this algorithm to the reachable graph of component S2, as shown in Fig. 17, we get the following fault state expression:

$$\mathrm{O} - \mathrm{S2} = \mathrm{O} - \mathrm{M1} + \mathrm{S2} < \mathrm{O} - \mathrm{S1} \tag{7}$$

5 Results and discussion

5.1 Generate Pandora temporal fault tree

According to the automatic generation algorithm introduced in Sect. 3, the sequence of timing fault tree is as follows:

$$\mathrm{Failed} = \mathrm{O} - \mathrm{I} + \mathrm{M1} < \mathrm{S1} + \mathrm{S2} < \mathrm{S1} + \mathrm{ACC} + \mathrm{B} + \mathrm{O} + \mathrm{M2} < \mathrm{S3} + \mathrm{S4} < \mathrm{S3} \tag{8}$$

Therefore, the resulting Pandora timing fault tree is shown in Fig. 18.

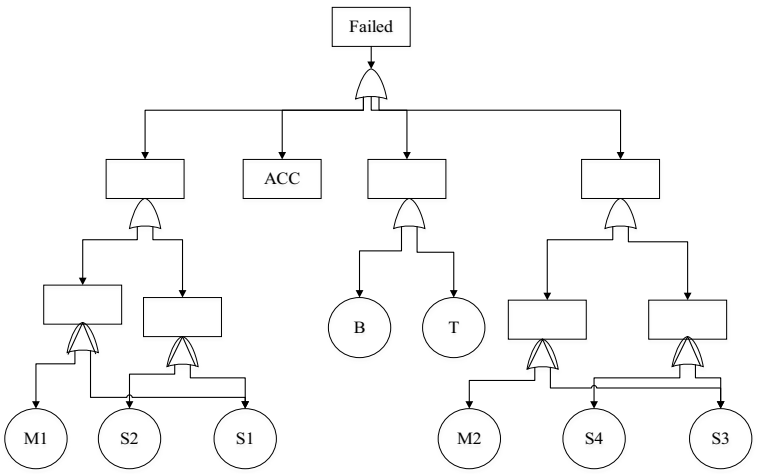


Fig. 18 Pandora TFT of ACC system

5.2 Model-based analysis of the ACC system

The ACC system is analyzed based on the model to obtain the cause of failure. The temporal fault tree obtained from the analysis is shown in Fig. 18. As you can see from TFT, some basic events are shared with each other, which makes them statistically dependent. If we use an analytical approach to quantify TFT, statistical dependencies between events are ignored. However, the proposed method of converting to a Bayesian network can quantify TFT by taking these dependencies into account, resulting in more realistic results.

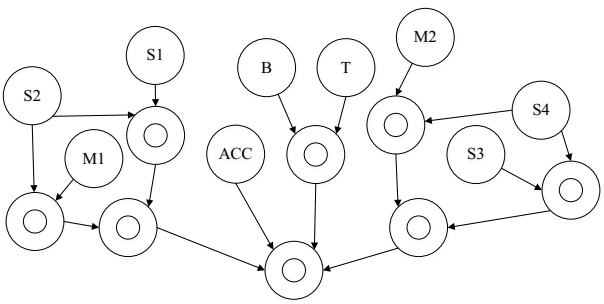


Fig. 19 BN model of ACC system

Table 1 Failure rate of the basic events

Basic events	Failure rate (1 h)
M1, M2	1.0×10^{-4}
S1, S2, S3, S4	1.93×10^{-4}
B	2.0×10^{-4}
T	2.0×10^{-4}
ACC	3.0×10^{-4}

To conduct quantitative analysis of the TFT, the Pandora TFT is mapped to the Bayesian network model, as shown in Fig. 19. The failure rate of Pandora TFT basic events is shown in Table 1, and time changes are characterized by the failure rate of basic events.

The prior probability table of each root node of BN is filled according to the failure rate of the corresponding component. As mentioned in Sect. 4, in order to generate the prior probability value of the event, we need to define a task time and interval number (N), and divide the task time into discrete intervals. Assuming that the task time is 5000 h, the task time is divided into four intervals ($N=4$), and the prior probability value of the root node of BN is calculated. The conditional probability table for each intermediate node in BN is filled according to the type of TFT gate represented.

The reliability analysis of the ACC system is carried out based on the DTBN model. Compared with the results of the Markov chain method, the results are shown in Table 2. It can be seen that the DTBN method has a high solution accuracy and Markov chains cause state explosion problems. Reliability analysis of ACC system is done by running queries on BN models. Meanwhile, using BN-based method, it can also be concluded that ACC is the most critical basic event of TFT, followed by Si ($i=1,\ldots,4$). This gives the system designer an idea of where to put more design effort to improve the reliability of the system. For example, an event ACC corresponds to a failure of the core electronic control within the system, while Si corresponds to a failure of the ranging sensor in the system. Therefore, if the analyst wants to improve the reliability of the system, then they can consider replacing the above key components with more reliable components, or they can consider introducing redundant components along with the key components. By observing the BN node endpoint in Fig. 19, in the failed state, the system was diagnosed and analyzed. This describes a scenario in which the analyst has evidence that the most significant event of the TFT has occurred. Based on this evidence, analysts' beliefs about prior probabilities of fundamental events have been updated. Based on these updated beliefs about the failure probability of basic events, a whole new predictive analysis can be performed.

By observing the leaf node of the BN model in Fig. 19, in the failed state, the system was diagnosed and analyzed. This describes a scenario in which the analyst has evidence that the most significant event of the TFT has occurred. Based on this evidence, the belief of the analyst on the prior probability of the basic event has been updated, and the posterior failure probability of the basic event is shown in Fig. 20. Based on these updated beliefs about the failure probability of basic events, a whole new predictive analysis can be performed. A posterior failure probability curve explains the dynamics of

Table 2 TE probability estimated by BN-based method and MC-based

Markov chain		DTBN	
TE probability	**Number of Markov chain states**	**Mission time (/h)**	**TE probability**
0.67×10^{-4}	9,864,101	1000	0.68923
		2000	0.89058
		3000	0.94459
		4000	0.97175
		5000	0.98487

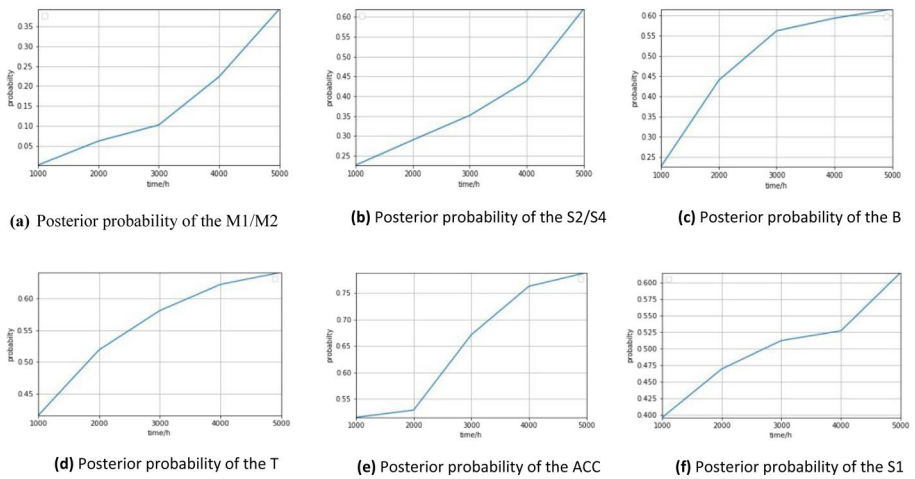


Fig. 20 Dynamically update the posterior probability of the base events

system failure and provides updated causality between variables that may change over time.

6 Conclusion

In recent years, more and more attention has been paid to the research of automatic fault tree generation. However, considering the timing fault characteristics of the system, AltaRica language modeling and automatic generation of timing fault tree algorithm in the field of the Internet of vehicles has not made much progress. This paper is based on the AltaRica model and studies how to automatically generate Pandora timing fault tree according to its flattening results GTS. The algorithm in this paper is a comprehensive design and implementation. First, the independent GTS of the whole model is obtained by dividing the algorithm, and then it is stored in two ways of edge node set and variable state node set in the module to obtain the reachable graph, and then expressed by adjacency matrix. The benefit of this approach is that it is fault-tolerant and uses different storage methods to store the information in the diagram, so that it can be verified in many aspects later when facing large system models. In compiling the accessibility graph algorithm, each TFT obtained is aggregated upward and unified into a system TFT, and assertions are propagated when multiple state results have been obtained, which is much more efficient than directly analyzing a complete GTS to generate a fault tree. Then, Pandora TFTs are generated and combined with Bayesian networks for reliability analysis, so as to realize effective automation of reliability analysis of complex dynamic systems.

In the following work, we will further combine the previous work with the current to build a complete system safety analysis for AltaRica3.0, and carry out a large-scale application case study in the typical field of smart Internet of vehicles.

Abbreviations

ACC: Adaptive cruise control; GTS: Guardian transformation system; TFTs: Temporal fault trees; MBSA: Model-based safety analysis.

Acknowledgements

We are very grateful to Dr. Qingfan Gu for the mail exchanges on the MBSA.

Authors' contributions
LW conceived and proposed the framework, the ideas of algorithms. QZ performed the experiments and analyzed the results. JH designed the case study. All authors read and approved the final manuscript.

Funding
The research work supported by National Basic Research Program of China (973 Prgram) No. 2014CB744900.

Availability of data and materials
The algorithms in this paper are available online at: https://github.com/jeneyzhang/GTS2TFTA/tree/master

Declaration

Competing interests
The authors declare that they have no competing interests.

Received: 1 December 2021 Accepted: 16 February 2022
Published online: 04 April 2022

References

1. G. Latif-Shabgahi, J.M. Bass, N.S. Be, A taxonomy for software voting algorithms used in safety-critical systems. IEEE Trans. Reliab. **53**(3), 319–328 (2004)
2. J. Delange, P. Feiler, Supporting the ARP4761 safety assessment process with AADL
3. D. Kritzinger, Functional hazard analysis, Aircraft Syst Saf (2017)
4. N. Xiao, W. Peng, T. Yi, et al, Research and application of preliminary system safety assessment on civil airborne systems, IEEE (2011)
5. Civil Aircraft Electrical Power System Safety Assessment || Abbreviations and Acronyms[J]. 2017:xix-xxi.
6. K.S. Trivedi, A. Bobbio, J. Muppala, Reliability and availability engineering: modeling, analysis, and applications. Non-State-Space (Combin.) Models (2017). https://doi.org/10.1017/9781316163047
7. S. Distefano, F. Longo, K.S. Trivedi, Investigating dynamic reliability and availability through state–space models. Comput. Math. Appl. **64**(12), 3701–3716 (2012)
8. D.J. Reifer, Software failure modes and effects analysis. IEEE Trans. Reliab. **R-28**(3), 247–249 (2009)
9. F. Ortmeier, A. Rauzy, *Model-Based Safety and Assessment* (Springer, Berlin, 2014)
10. A. Gomes, A. Mota, A. Sampaio et al., *Systematic Model-Based Safety Assessment Via Probabilistic Model Checking* (Springer, Berlin, 2010)
11. H.U. Jun, S. Chen, M.M. Wang, A transformation method for AltaRica3.0 to Promela and its verification. Comput. Eng. Sci. **39**(4), 708G716 (2017)
12. F. Belmonte, E. Soubiran, A model based approach for safety analysis, in International Conference on Computer Safety. Springer, Berlin, Heidelberg (2012)
13. J. Song, B. Chen, X. Li et al., The software fault prediction model based on the AltaRica language, in 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, pp. 2549–2552 (2019)
14. M. Batteux, T. Prosvirnova, A. Rauzy, Advances in the simplification of Fault Trees automatically generated from AltaRica 3.0 models (2018)
15. A.B. Rauzy, Guarded transition systems: a new states/events formalism for reliability studies. Proc. Inst. Mech. Eng. Part O J. Risk Reliab. **222**(4), 495–505 (2008)
16. J. Vidalie, M.S. Kendel, F. Mhenni et al., State machines consistency between model based system engineering and safety assessment models, in 2021 IEEE International Symposium on Systems Engineering (ISSE). IEEE, pp. 1–8
17. F. Chiacchio, J.I. Aizpurua, D. D'Urso et al., Coherence region of the Priority-AND gate: analytical and numerical examples. Qual. Reliab. Eng. Int. **34**(1), 107–115 (2018)
18. J.B. Fussell, E.F. Aber, R.G. Rahl, On the quantitative analysis of priority-AND failure logic. IEEE Trans. Reliab. **25**(5), 324–326 (1976)
19. C. Wang, L. Wang, H. Chen et al., Fault diagnosis of train network control management system based on dynamic fault tree and Bayesian network. IEEE Access **9**, 2618–2632 (2020)
20. S. Kabir, M. Walker, Y. Papadopoulos, Reliability analysis of dynamic systems by translating temporal fault trees into Bayesian networks, in Model-Based Safety and Assessment (2014)

Publisher's Note