Automatic Generation Method of Temporal Fault Tree Based on AltaRica3.0

Qin Zhang(✉), Lisong Wang, and Jun Hu

Nanjing University of Aeronautics and Astronautics, Nanjing, China
Zhangqin1611@nuaa.edu.cn

Abstract. Altarica3.0 is a high-level modeling language for security analysis, and its basic mathematical form is Guardian Transformation System (GTS). At present, GTS (or Altarica3.0 model) can be compiled into a fault tree or critical event sequence. There are several reasons for turning GTS model into a fault tree: automatically generating a fault tree from a high-level model is easier and less time consuming than creating one from scratch; Second, the high level model greatly improves the model design, sharing and maintenance; Finally, the evaluation tools of the Boolean model are more effective than the state/transition model. In general, however, the cost is the loss of sequence between events: the sequence of events is compiled into the connection of events, so there may be potential omissions throughout the system reliability analysis process and analysis results. To remedy this problem, in this paper we outline an analysis approach that converts failure behavioural models (GTS) to temporal fault trees (TFTs), which can then be analysed using Pandora a recent technique for introducing temporal logic to fault trees. The approach is compositional and potentially more scalable, as it relies on the synthesis of large system TFTs from smaller component TFTs. We verify, by using a redundant system, the effectiveness of the proposed method.

Keywords: AltaRica3.0 · GTS · Accessibility diagram · Pandora · Temporal fault tree

1 Introduction

With the wide application of critical safety system, its complexity is increasing. How to ensure the safety of the system and avoid the occurrence of catastrophic accidents, such as property loss or even casualties, has become a research hotspot in the field of system safety analysis. The Altarica modeling language is designed to solve this problem. Altarica [1–3] is a high-level safety analysis modeling language, which has a good layered system structure and dynamic description mechanism, and can provide fault behavior modeling elements based on fault. As a result, it is beginning to be widely used in safety-critical fields such as aerospace [4]. In research [5], an algorithm for compiling Altarica data streams (or pattern automata) into fault trees is proposed. However, because the event sequence is compiled into the connection of events, the timing relationship between the events is lost, which leads to the potential error in the analysis of the system dynamic timing fault. Therefore, considering the timing state of system component

Q. Liang et al. (Eds.): CSPS 2021, LNEE 878, pp. 892–900, 2022.
https://doi.org/10.1007/978-981-19-0390-8_113

failure events, Pandora timing gate is introduced to describe the timing relationship. In this paper, an automatic compilation and generation method of Pandora timing fault tree based on Altarica3.0 modeling language is proposed.

2 Background

2.1 Guardian Transformation System (GTS)

Altarica3.0 is a new version of the Altarica language [6]. Its new underlying mathematical model is the guard transformation system (GTS) [7], which is a state/transition form specifically designed for security 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:

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 postcondition. 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.

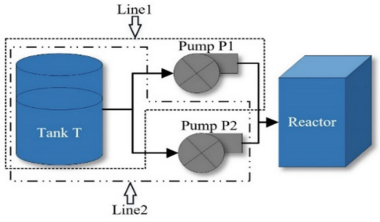

Fig. 1. Cooling system

Concrete components in the cooling system are abstracted into classes. Tank, Pump and Reactor are 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 part of GTS model obtained by flattening the hierarchical information of the cooling system is shown below (Table 1).

Table 1. Part of the GTS model of the cooling system

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

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 [8]. The temporal fault tree symbols of the three gates are shown in Fig. 2. PAND gates are not new and have been used in FTA [9] since the 1970s. 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 'I'), 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 ocurrence 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 ⇔ 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 full details about Pandora's temporal laws we refer the reader to [12], but four laws in particular will be useful:

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

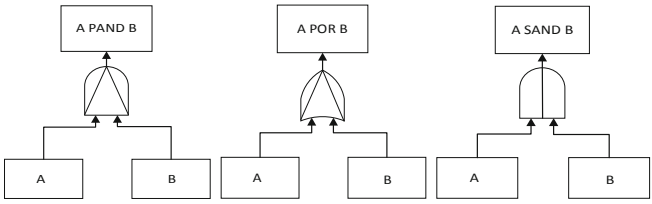

Fig. 2. Pandora gates

3 A Framework for Automatic Generation of Pandora Temporal Fault Tree Based on GTS and Core Algorithm Set

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. The algorithm GTS to Pandora Temporal Fault Tree (GTS2PTFTA) design involved in each step is then explained in detail. The GTS2PTFTA method framework is generally divided into the following five main steps:

design involved in each step is then explained in detail. The GTS2PTFTA method framework is generally divided into the following five main steps:

(I) Read the flattened GTS model description file and construct the corresponding GTS object. (II) The input GTS object is divided into a set of independent GTS and an independent assertion. (III) 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. (IV) 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 Sect. 3.3. The AND gate operator is further annotated as: search all paths from state s_0 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 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, Generates a sequence of events with a temporal relationship. (V) Synchronize the independent assertions obtained by partition with the formulas obtained by compilation of each independent GTS reachable graph, and obtain the set of timing relational event sequences of the whole model, which is composed of the Pandora timing fault tree of this GTS model.

3.1 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.

3.2 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.

3.3 The Accessibility Graph of the Sub-GTS Model Is Compiled to Generate a Temporal Expression

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.

Definition 2: Reachability graph (RG)
A Reachability graph is a quadruple (S, Σ, δ, s0) 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 fifinal states
Let F be the set of the fifinal 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:

Input: Reachability graph; **Output:** An expression with a time sequence

1: P ∈ reachability graph; F ∈ S ; let Φ = ∅ ;
2: **for each** s∈F **do**
3: get Ps from P where Ps = {$\pi \in P | s_0 \rightarrow s(\pi)$}
4: let n = |Ps| ; let πi∈Ps 1≤i≤n ; let leni=length(πi) 1≤i≤n
5: let Seqπi={s_{i0} , s_{i1} , . . . , s_{ileni}} ; let φs = W1≤i≤n(V1≤j≤leni eij)
6: **for each** sij in Seqπi 1≤i≤n 1≤j≤leni s.t. |Σsij I|>1 **do** let e = eij
7: **for each** e'=eij in Σsij I **do** let t∈S s.t. sij e' →t
8: **if** ((e'∈{ei(j 1) , ei(j 2) , . . . , ei1 }) **or** (∃ (y∈Σ, π_0∈P, f∈S, v∈S in Seqπ$_0$, w∈S in Seqπ$_0$) s.t. t π$_0$ and y∈{eij , ei(j-1) , . . . , ei1 })) **then** let e = e|e'
9: **end if**
10: **end for**
11: replace eij with e in φs
12: **end for**
13: let Φ = Φ ∪ {φs}
14: **end for**

Generates a set Φ of Pandora formulae: Φ = {φs|s ∈ F} one formula φs for each final state s. These expressions can then be analysed 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.

4 Case Studey

To better illustrate the effectiveness of our approach, we use a generic three-module redundancy (GTR) system, as shown in Fig. 3. Elements X1, X2, and X3 are abstract representations of any input, control, or drive, arranged in redundant series, with X1 as the primary element, X2 as the secondary element, and X3 as the third element. Two monitoring sensors S1 and S2 detect leakage from X1 and X2 outputs, respectively, activating the next backup in the series. O is just an abstraction of the GTR output, and I represents the input to the system.

System failure events are Omission of input at I, all of X1, X2, and X3 fail, Both X1 and S1 fail (X2 will not be activated) and all of X1, X2, S2 fail (X3 is not activated) and failure of O [11]. If there is no input to the system, then it cannot operate; If all three main components fail, then the system will likewise fail as well. However, the GTR system exhibits dynamic behaviour: its true failure behaviour depends on the chronology of events. Different sequences of failure events can lead to the system failing in different ways, and not all of them are correctly represented by the results. For example, suppose

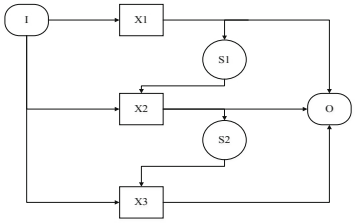

Fig. 3. GTR system

X2 fails first, then X1 fails again; In this case, X2 cannot be activated when sensor S1 detects the output leakage of X1, and sensor S2 cannot activate X3 because X2 has never been activated. Sensor S2 can detect the output leakage only when the monitored component is activated. So, failure of sequence X2 before sequence X1 will result in system failure, regardless of the status of X3 and S2.

In this experiment, Altarica3.0 was used to convert the model file built by the system into the GTS model of the flattened system, and the description file was taken as the input file of this experiment. Because the system contains many variables, transformations and assertions, it generates many accessibility graphs. This paper presents only partial resultsd. For component X3 (reachability graph of Fig. 4), there are two final states, both with omission of output from X3 (O-X3) as common effect. The state permanently OFF means that X3 will never be activated and is caused by O-S2 (i.e. S2 has prematurely failed and both X1 and X2 have failed in sequence). The other final state is a state where X1 and X2 have failed in sequence and where X3 has also failed (caused by the event X3 fails). Omission from X3 is the third cause of failure for output component O, leading to Omission O and system failure. Note that X1 and O are each initially in state ON, i.e. the GTR system is initially working with its primary component, and the output is being delivered by the system. Backup components have their initial states set to OFF and sensors are initially set to Monitoring.

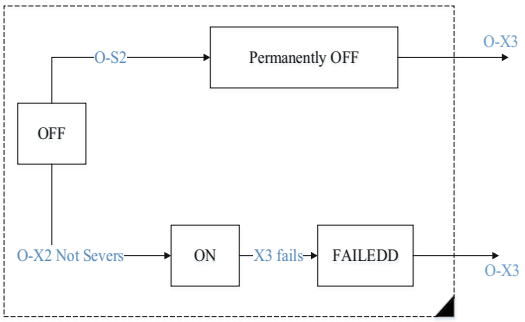

Fig. 4. Accessibility diagram of X3

Compile the accessibility graph into an expression with a time series relationship, and generate the corresponding Pandora timing fault tree based on the expression, as

shown in Fig. 5. for the sake of clarity, any component symbol (e.g. X1) abbreviates the failure (e.g. X1 fails) of the corresponding component.

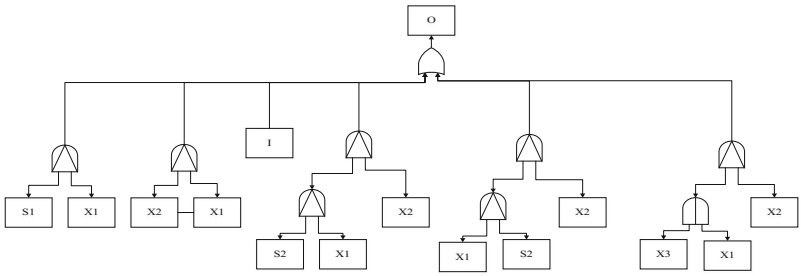

Fig. 5. TFT of the GTR system

5 Conclusion

This paper proposes a method to transform the GTS model into a Pandora temporal fault tree, which allows more detailed analysis of different sequences of events to better capture dynamic behavior. In the following work, we will further combine the previous work with the present work to build a complete system security modeling and analysis tool for Altarica3.0, and conduct a large scale application case study in the typical aviation field.

References

1. Jun, H., Song, C., Mingming, W.: Transformation method for AltaRica 3.0 to Promela and its verification. Comput. Eng. Sci. **39**(4), 708–716 (2017)
2. Prosvirnova, T.: AltaRica 3.0: a model-based approach for safety analyses. Ph.D. diss., Ecole Polytechnique (2014)
3. Prosvirnova, T.: The AltaRica 3.0 project for model-based safety assessment. In: IEEE International Conference on Industrial Informatics, IEEE (2013)
4. Latif-Shabgahi, G., Bass, J.M., Be Nnett, S.: A taxonomy for software voting algorithms used in safety-critical systems. IEEE Trans. Reliab. **53**(3), 319–328 (2004)
5. Zhipeng, W.U., Jun, H.U., Chen, S., Shi, J.: Safety verification methodology of embedded system based on altarica model. J. Front. Comput. Sci. Technol. **11**(1), 24–36 (2017)
6. Batteux, M., Prosvirnova, T., Rauzy, A.: Advances in the simplification of fault trees automatically generated from AltaRica 3.0 models (2018)
7. Rauzy, A.B.: 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)
8. Walker, M., Bottaci, L., Papadopoulos, Y.: Compositional temporal fault tree analysis. In: Saglietti, F., Oster, N. (eds.) SAFECOMP 2007. LNCS, vol. 4680, pp. 106–119. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75101-4_12

9. Rauzy, A.: Mode automata and their compilation into fault trees. Reliab. Eng. Syst. Saf. **78**(1), 1–12 (2002)
10. Mahmud, N.: Dynamic model-based safety analysis: from state machines to temporal fault trees. University of Hull (2012)
11. Walker, M.D.: Pandora: a logic for the qualitative analysis of temporal fault trees. Computer Science the University of Hull (2009)
12. Walker, M.D.: Pandora: a logic for the qualitative analysis of temporal fault trees. Ph.D. diss., The University of Hull (2009)