AUTOMATYKA/ AUTOMATICS • 2013 • Vol. 17 • No. 2

http://dx.doi.org/10.7494/automat.2013.17.2.219

Paweł Skrobanek*

Minimal Cut Sets with Time Dependencies Analysis

1. Introduction

Safety system analysis is an important aspect and as such requires the use of formal methods. One of the most commonly used techniques is the Fault Tree Analysis (FTA) [1, 2].

FTA is a top-down approach. The initial step is to identify potentially dangerous situations, so-called faults. Then, for each fault the fault tree (FT) is constructed – the fault is the top-event. The next levels of the tree are contracted by the identification causes of the top-event. The events (causes and top-event) are connected with an appropriate logical gate (AND, OR, etc.). Going down (identifying the cause of the causes) we are building the next levels of the tree, up to a certain level of detail or up to event for which we cannot or do not want to detail the reasons.

For example, if "melt down of the reactor core" fault occurs as a result of the co-occurrence of events "nuclear reaction takes" and "insufficient cooling of the reactor" – that events are connected connect with an AND gate. the next step is to identify the causes of events "nuclear reaction takes" and "insufficient cooling of the reactor".

The results of the analysis are Minimal Cut Sets (MCSs) [1] usually with defined probability of occurrence of the event. The events which form the MCS must occur together to cause the top-event (fault). In the given example this could be MCS: {"insufficient cooling of the reactor", „nuclear reaction continues"}. Note: if one of them would not occur, for example, cooling would be appropriate or would be no nuclear reaction (reactor would be switched off and cooled), respectively then the top-event "insufficient cooling of the reactor" would not occur. Therefore, the MCS is the set of events, which occurrence is a necessary and sufficient condition for the occurrence of a fault (the top-event).

A standard fault tree cannot express time dependencies between events. Hence, using FTA we cannot analyze an aspect such as for example, the minimum and maximum time in which the lack of cooling would not lead to a danger – reactor melt or how fast (at what time) we can stop the nuclear reaction. We can analyze that using other methods or using

* Wrocław University of Technology, The Institute of Computer Engineering, Control and Robotics, Wrocław, Poland

Fault Trees with Time Dependencies (FTsTD) [3, 4]. The time dependencies (e.g. event duration, delay between causes and its effect) are determined by specifying parameters for the event and for certain types of gates (see Section 2).

The result of the Fault Tree with Time Dependencies Analysis (FTTDA) and FTA are MCS. The only difference is that the MCS obtained by FTTDA contain additional information about the temporal relationships between events. These relationships are described by time conditions of the start and end of the event. In some cases MCS may contain additional inequalities, for example, describing the minimal and maximal duration time of any event or of co-occurrence of events, which is necessary and sufficient to cause the hazard. As in the described earlier example of the reactor, we would have time conditions related to the event "insufficient cooling of the reactor." Intuitively, if this event ends up quickly enough (for example, is inclusion of an emergency cooling) then fault "insufficient cooling of the reactor" will not occur.

Due to the time dependencies in MCS, we obtain more such sets as the result of the FTTDA. However, many of this MCS contain the same events but with different time parameters (some examples are shown in sub-chapter 4). This chapter presents a method of ordering MCS with time and reduction of the time conditions for specific cases. The main goal is to reduce the number MCS that after the transition should be consistent with the number of MCS obtained in standard FTA.

2. Fault tree with time dependencies

Fault trees are presented in [3, 5, 6]. In the paper [6] a prototype version of the tool that allows the analysis of FTsTD was discussed.

Construction of the FTTD initially runs as a classical fault tree construction. An additional step is to determine the time parameters for the event (describing the minimum and maximum duration time) and some types of gates. The parameters are determined by experts, although the process may be done partly automatically [7]. The parameter values are often the result of the construction of the system (used equipment, cables, etc.), physical laws (such as the speed of sound in the air), the set of human values (e.g. the value of the variable in the program – the period of time that is testing every component system). Examples of applications the FTTDA for real systems can be found in [8, 9].

The notation of gates and events in FTTD used in this chapter is in accordance with the proposal presented in [6] and is presented in Figure 1. The events are described by a unique number *No*, the parameters that specify the minimum and maximum duration of the event <*tmin*, *tmax*> (in particular, they can be, respectively, 0, ∞). In addition, each event excluding the top-event contains information about the gate and the entrance to which it is connected (*L* – left, *R* – right). The final element is a verbal description *comment*. For the output events of generalization gates, the time parameters are not specified, because they are calculated from parameters given for input events into the analysis process.

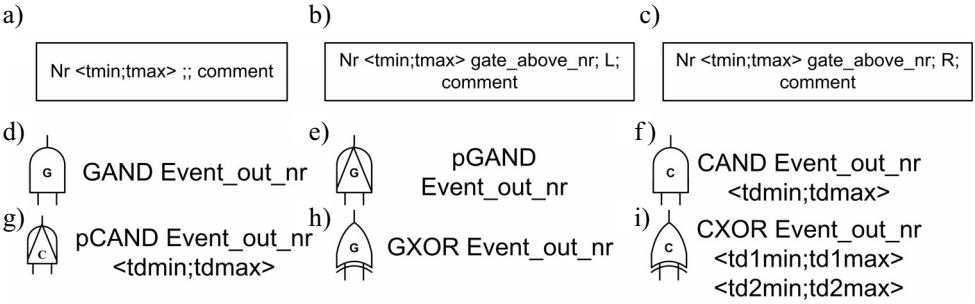

Fig. 1. Notation of the evens: a) top-event (fault), b) left-input event, c) right-input event. Notation of the gates: d) generalization AND, e) generalization priority AND, f) causal AND, g) causal priority AND, h) generalization XOR, i) causal XOR

The graphical notation of the gates and events indicating the type of gates is as follows: AND, XOR with additional information about the priority (priority or not) and type (generalizing or causal). In addition to the graphical notation, because of the tool (the ability to read data from any file with a specific structure [6], the information is also stored as a string (by the gate) and contains the output event number. Additionally for the causal gates time parameters that describe the delay between cause (XOR gate) or causes (AND gate) and an effect (an output event) are given. An example of notation MCS by XML file is given in Figure 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <FAULT_TREE_ANALYSE>
  - <Tree>
    - <MCS>
      - <Event_6>
          <MinStart>0.0-min(0.0;min(0.0;niesk))-min(tsamax;niesk)</MinStart>
          <MaxStart>0.0-tsamin</MaxStart>
          <MinEnd>0.0</MinEnd>
          <MaxEnd>0.0-tsamin+niesk</MaxEnd>
          <![CDATA[0.0<=min{niesk;0.0} AND tsamin<=niesk]]>
        </Event_6>
      - <Event_5>
          <MinStart>0.0-niesk-min(tamax;min(niesk;niesk))</MinStart>
          <MaxStart>0.0-tamin</MaxStart>
          <MinEnd>0.0-niesk</MinEnd>
          <MaxEnd>0.0-tamin+niesk</MaxEnd>
          <![CDATA[0.0<=min{niesk;0.0} AND tamin<=min{niesk;niesk}]]>
        </Event_5>
      - <Event_4>
          <MinStart>0.0-niesk-niesk</MinStart>
          <MaxStart>0.0-tamin</MaxStart>
          <MinEnd>0.0-niesk</MinEnd>
          <MaxEnd>0.0-tamin+niesk</MaxEnd>
          <![CDATA[0.0<=min{niesk;0.0} AND tamin<=min{niesk;niesk}]]>
        </Event_4>
      </MCS>
    </Tree>
</FAULT_TREE_ANALYSE>
```

Fig. 2. An example of MCS notation

The MCS given in Figure 2 contains three events: 4, 5 and 6. For each we have two time intervals, respectively, for start and end. The time parameters are refered to the "0" time instant which is connected to the start of the top-event. Because some time parameters in FTTD given in Figure 3 are given parametrically such as minimal and maximal time of SYN-flood attack <*tamin*, *tamax*>, additionally into the process of analysis we obtain some condition labeled as *CDATA[…]*. The minimal time at which the attack SYN-flood (if the server does not have appropriate safeguards) can be effective and the maximal time after which they will surely be effective, respectively *tamin* and *tamax*, are given parametrically, because they depend on a particular server. More information can be found in [10].

As an alternative to recording in the embodiment shown in Figure 2 is a graphical notation, an example of which is shown in Figure 4.

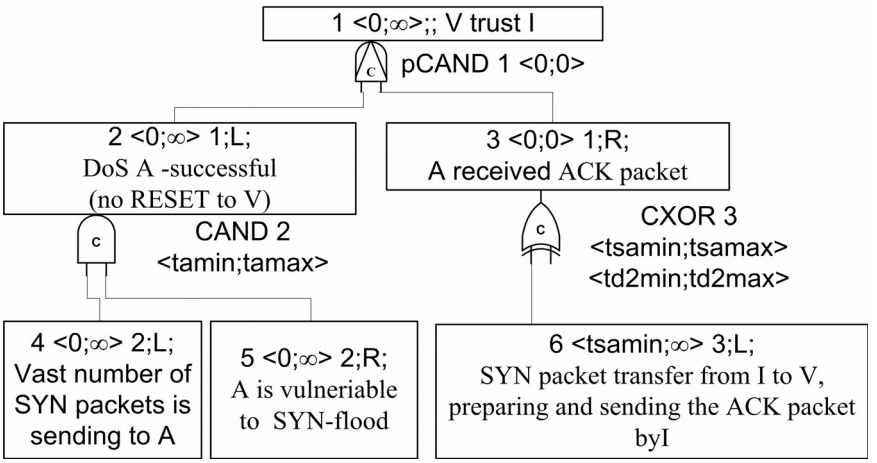

Fig. 3. Example of FTTD, source [6]

$$4 < -\infty, -ta\ min > < -\infty, \infty >$$
$$5 < -\infty, -ta\ min > < -\infty, \infty >$$
$$6 < -tsa\ max, -tsa\ min > < 0, \infty >$$

Fig. 4. Graphical notation of MCS from Figure 2
(the notation of the conditions given into CDATA[…]
is not present, because of all conditions are satisfying)

3. The process of organizing MCS and identification of safety conditions

Analysis performed by the human is a difficult process due to the large number of MCS. Additionally, the number of such sets is much greater in FTTDA than in standard FTA.

Facilitating of the analysis can thus be carried out in two levels: by simply ordering and a partial analysis of the results, such logical analysis. Hence, we have two types of activities:

– ordering activities (discussed in Section 3.1 and 3.2),
– suporting analysis (Section 3.3).

While ordering activity may be useful in the analysis of any FTTD, supporting analysis will be useful to support the analysis only for a specific type of FTTD or by checking certain dependencies. In particular it is possible to develop different algorithms for the study of certain properties, but this is beyond the scope of this paper.

3.1. Ordering MCS in view of events

MCS with time dependencies contain collections of events together with the given parameters specifying the conditions for the start times (start events) and the moments of their completion.For example, the notation form: *zx* <*tsmin*, *tsmax*> <*temin*,*temax*> means, among other things, that an event *zx* must begin no earlier than *tsmin* and no later than the *tsmax* time units, related to the contractual point of time "0" – which is typically the start of the top-event (fault). In general, the time parameters define the relationship between the events themselves. There are therefore cases where the certain minimal cut sets would contain the same event but with different time parameters. The first proposal concerns the appropriate ordering of such sets, and in fact the transition to a single set of events with the time domain parameters – in short, with the time domain.

The idea of MCS ordering is shown in Figure 5. It is based on a combination of all MCS with the same set of events together into one MCS with the time domain composed of all connected sets. The Minimal Cut Sets with Time Domain (MCSTD) obtained in this way corresponds to MCS, which could be obtained in standard FTA. More specifically, one MCSTD would correspond one MCS. The number of MCSTD could be less than the number of MCS – some sets could not be ruled out as a likely to cause fault, due to time dependencies.

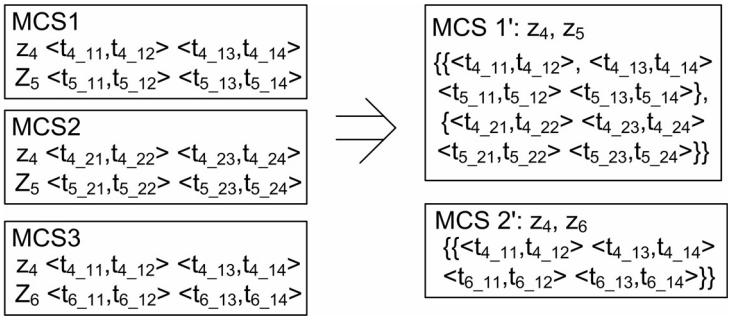

Fig. 5. Example of transformations MCS to MCSTD

3.2. Simplifying conditions

The second area in which it is possible to partially automate an analysis of the conditions in CDATA sets [...] – mainly from the reduction of inequalities containing the symbol "∞" and by comparing the numerical values ??(if numerical value occur or we receive them as a result of ordering). Expressions transformed in this way then would be subjected to logical analysis, as form "TRUE / FALSE" AND "TRUE / FALSE" [AND ....], or form "CONDITION AND TRUE / FALSE [AND ...]". The implementation of the first part requires the use of a set of rules, as follows ("sth." – is both parametric and numerical value):

1) R1: max{sth., ∞} = ∞,
2) R2: min {sth., ∞} = sth.,
3) R3: min { sth., coś} = sth.,
4) R4: min {∞, ∞} = ∞,
5) R5: max { sth., sth.} = sth.,
6) R6: max {∞, ∞} = ∞,
7) R7: sth.<= sth. = TRUE,
8) R8: sth.<= ∞ = TRUE
9) R9: number1<= number2 = TRUE/FALSE (depending on the relationship between numbers).

The principle of conduct can be summarized as follows:

K1: determine the minimum and maximum value for all files containing infinite elements – according to the rules R1–R6,

K2: an analysis of the conditions – in accordance with the rules R7–R9,

K3: the logical analysis of expressions.

Carrying out this type of operation does not ensure the elimination of all the conditions, but in many cases will simplify their writing, thus improving the readability. Note: it is worth to remember that the infinity should be coded unequivocally.

The result of this process for MCS given in in Figure 2 is shown in Figure 6.

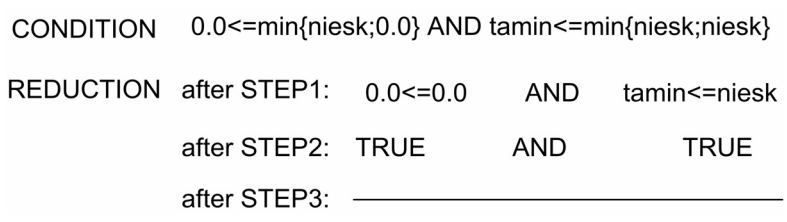

Fig. 6. Example of reduction (analysis) of condition in MCS ("niesk" coded infinity)

A reduction of "sth. <= niesk" requires a comment. According to the definition (see, e.g. [5, 6]) the minimum duration time of an event can be zero or an arbitrarily large real number, but it must be less than infinity. So inequality as "sth. <= niesk" (such as "tamin <= niesk") will always be true, because it is not possible to have the situation "$\infty = \infty$".

3.3. Analysis of dependencies for MCS of FTTD with event synchronization

Event synchronization is specific for FTTD. It is based on the analysis of temporal properties of a system for a specific event, such as the appearance of the train (control system outside the city gates), start an attack on a computer system, start a failure. Synchronization allows, among other, for the analysis or identification of time dependencies for sequences of events, such as sequence of action for securing a system. Examples can be found in [3, 11]. In the case of synchronization, MCSTD may contain the same event with different conditions for the moment of start and end. Assume that the MCSTD contains two single events *zx* with the following time parameters:

zx <mins, maxs> <mine, maxe>

zx <min2s, max2s> <min2e, max2e>

Let us consider the intervals defining constraints on the start of the event *zx*: <*mins, maxs*> and <*min2s*, *max2s*>. Knowing that the event zx will be able to occur only when they have a common part and that the corresponding volume (mins – min2s and maxs – max2s) may be the same – in the case of an equal number, if parameters – saved in the same way, we obtain the algorithm:

IF mins=min2s THEN <mins,min{maxs,max2s}>

ELSE (IF maxs=max2s THEN <max{mins,min2s},maxs>

ELSE <max{mins,min2s},min{maxs,max2s} /COND: maxs≥min2sAND max2s≥min2s/ >)

The fulfillment of the condition given in the last part of the case is necessary for intervals to have a common part. Therefore, non-fulfillment of the condition prevents the occurrence of fault. An example is shown in the next section. In the first two parts, the intervals always have a common part – in particular, one "point" (time instant).

Similarly, following the <*mine*,*maxe*> and <*min2e*,*max2e*> the conditions associated with the completion of the event will be obtained.

These are not the single property of MCSTD that may be of interest for experts, and of which determination can be automated through the implementation of the tools. The most

interesting, which are to be the next stage of the research are for example: the maximal time of some failure co-occurring than cannot a fault, minimal time distance between particular failure and top-event.

An analysis of these aspects is of interest not only from the point of view of studying the results by the human and implementation of system security, but can be used for example for monitoring of the systems using tools based on temporal logic [12].

4. Case study

Example 1. GAS BURNER SYSTEM – description of the system, analysis and system security proposal is given in [3].

A hazardous situation "dangerous gas concentration" may occur when a gas valve is open for too long and there is no flame. The valve could have been opened by the controller (the event "t") or by accident (event "u"). The lack of flame could have been caused by the failure of ignition device (event, "w") or the lack of ignition signal from controller (event "v"). MCS and the result of their transformation into MCSTD is given in Figure 7.

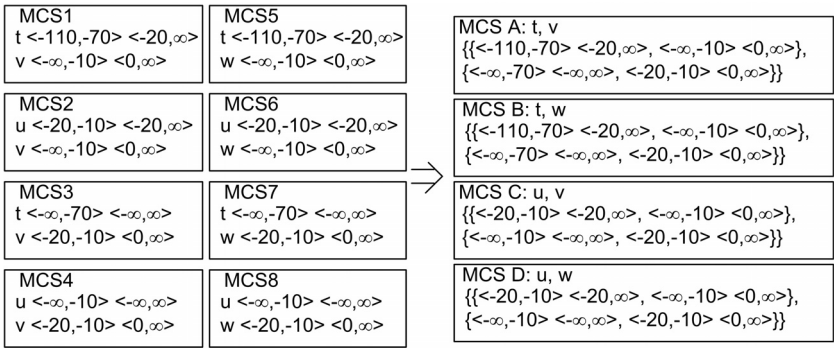

Fig. 7. Transformation MCS for the gas burner system

Example 2. SESSION HIJACKING BY [11].

MCS are shown in Figure 8a. Event 8 "SYN packet sending from Intruder to Victim" must occur when another computer (e.g. server) cannot reset the connection (send information that he had not sent the SYN packet). Double-occurrence of events 7 ("initializing attack") is used to synchronize events: "vulnerability to SYN-flood attack and the "an attack on the computer – sending too many packages". Note: the vulnerability to attack has to occur at a specific time. If it is not synchronized with the attack, the attack is not dangerous (for more details see [11]). By proceeding in accordance with the algorithm given in Section 3.3 we obtain the target MCSTD given in Figure 8c.

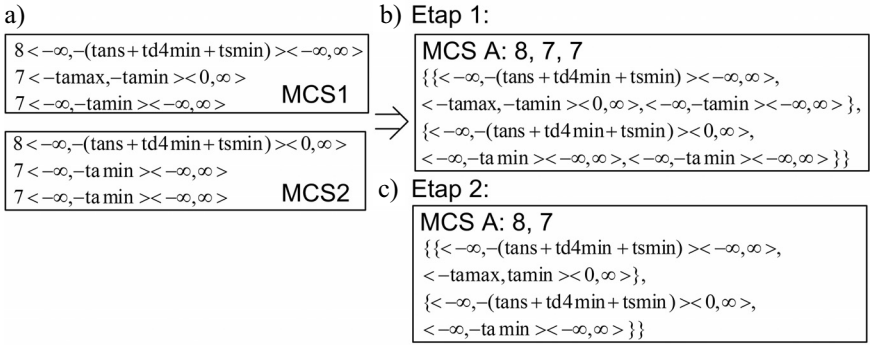

Fig. 8. MCS with time dependencies (a), ordering (b), analysis for synchronized event (c)

5. Summary

A large number of MCS during the analysis of the fault tree with time dependencies analysis can arouse problems with their subsequent use and understanding. Two main areas of activity in order to support such work through their implementation in the target tools are proposed is this chapter.

The first area is transformation of many MCS with time dependencies into one MCS with time domain (as was shown in sections 3.2 and 3.3). Obtained in this way MCSTD correspons to the MCS obtained by the standard FTA. The combination of the results of two methods of analysis allows to assess both the probability of fault (standard analysis) and also to know the time relationship between events. Such knowledge would provide greater possibilities for system securing, for example, security and event monitoring and predicting the possibility of fault.

The second area is defined by actions related to the analysis of certain properties of MCSTD. An example of such actions related to the FTTD modeling synchronization of events is discussed in section 3.3 and illustrated by example (Section 4).

References

[1] Fault Tree Analysis (FTA), International Standard, IEC 61025:2006.

[2] Fault Tree Handbook, NUREG-0492, U.S. Nuclear Regulatory Commission, Washington D.C. 2055, 1981. (http://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/sr0492.pdf).

[3] Magott J., Skrobanek P., *A method of analysis of fault tree with time dependencies*. in: Proc. SAFECOMP 2000, Rotterdam, The Netherlands, LNCS, vol. 1943, Springer-Verlag, 2000, pp. 176–186.

[4] Górski J., Magott J., Wardziński A., *Modeling Fault Trees Using Petri Nets*. SAFECOMP'95, Belgirate (Italy), 1995.

[5] Skrobanek P., *Analiza zależności czasowych w drzewach niezdatności systemów związanych z bezpieczeństwem*. Praca doktorska, Politechnika Wrocławska, raport: PRE. 24/2005.
[6] Jureczko M., Skrobanek P., *Tool for analysis of the fault tree with time dependencies*. Przegląd Elektrotechniczny, R. 86, No. 9, 2010, pp. 179–183.
[7] Magott J., Skrobanek P., *Timing analysis of safety properties using fault trees with time dependencies and timed state-charts*. Reliability Engineering & Systems Safety, vol. 97, No. 1, 2012, pp. 14–26.
[8] Magott J., Nowakowski T., Skrobanek P., Werbinska-Wojciechowska S., *Logistic system modeling using fault trees with time dependencies – example of tram network, Reliability, risk and safety: theory and applications*. Vol. 3/eds. Radim Briš, C. Guedes Soares, Sebastián Martorell. London, Taylor & Francis, cop. 2010, pp. 2293–2300.
[9] Lukowicz M., Magott J., Skrobanek P., *Selection of minimal tripping times for distance protection using fault trees with time dependencies*. Electric Power Systems Research, vol. 81, No. 7, 2011, pp. 1556–1571.
[10] Jureczko M., Skrobanek P., *Oprogramowanie wspomagające analizę drzew niezdatności z zależnościami czasowymi, Systemy czasu rzeczywistego.* in: *Postępy badań i zastosowania*. Z. Zieliński (Ed.), Warszawa, Wydawnictwa Komunikacji i Łączności, 2009, pp. 255–267.
[11] Skrobanek P., Woda M., *Analysis of timing requirements for intrusion detection and prevention using fault tree with time dependencies*. in: Intrusion Detection Systems, Pawel Skrobanek (Ed.), Rijeka, InTech, 2011, pp. 307–324.
[12] Głuchowski P., *Languages of CTL and RTCTL calculi in real-time analysis of a system described by a fault tree with time dependencies*. Dependability of Computer Systems, DepDoS-RELCOMEX'09, W. Zamojski et.al. (Ed.), IEEE 2009