A new algorithm for computer-aided fault tree synthesis

Y. Wang a, T. Teague a, H. West a, S. Mannan a,∗

a Mary Kay O’Connor Process Safety Center, Chemical Engineering Department, Texas A&M University System, College Station, TX 77843-3122, USA

Received 3 October 2001; received in revised form 24 January 2002; accepted 24 January 2002

Abstract

Fault tree analysis (FTA) has been used in the chemical process industry (CPI) for systematic safety and reliability analysis during the past decades. Conventional manual construction of fault trees can be extremely time consuming and vulnerable to human error. A computer-aided fault tree synthesis methodology can be an initial step, or as an independent check to assist or supplement manual FTA. However, no entirely satisfactory algorithm has been published for fault tree synthesis, especially when control loops are encountered. A potential methodology to construct fault trees automatically is proposed in this paper. This algorithm works directly form the system block diagram, thus avoids the tedious work of generating digraphs, transition tables, decision tables, and knowledge-based rules. Mini cause-and-effect trees are used to model the cause and effect logics around each item of equipment. Control loops are treated by special cause-and-effect unit models — logical combinations of the unit models of their constituent components. Multiple or complex control loops can be easily taken into account by providing their corresponding cause-and-effect unit models. In particular, the fault tree construction algorithm presented here is based on a component-by-component basis instead of a loop-by-loop or node-by-node basis. The tree structure is much more concise and easier to read. An example is embedded in the description of the methodology for better understanding. Analysis shows that the fault tree generated here is equivalent to the published result. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Fault tree synthesis; Cause and effect model; System block agram; Control loop; System boundary

1. Introduction and background

Pressure from regulation and society is increasing demands for more rigorous safety and reliability analyses of chemical plants. An essential goal of a Chemical Process Quantitative Risk Analysis (CPQRA) is to estimate the frequency and the consequence of specific incidents. As a powerful tool for risk assessment, Fault Tree Analysis (FTA) has long been successfully applied in CPQRA to predict the likelihood of hazardous accidents and identify major risk contributors.

While well-developed computerized codes are available for the evaluation and analysis of fault trees, the synthesis of fault trees remains the weakest link of the whole procedure. The conventional manual construction of fault trees is a difficult and time-consuming task and susceptible to human errors and omissions. Realizing this problem, many researchers have been focusing on computerized fault tree synthesis for years. Also, some computer-aided fault tree synthesis techniques have been developed.

Computerized fault tree synthesis is similar to conventional fault tree construction in the sense that it starts with the TOP Event, through all of the identified intermediate gates to the primary failures of the components or boundary conditions. A variety of modeling techniques have been employed to model fault propagation in chemical processes. Fussell (1973) developed a formal fault tree synthesis methodology for electrical systems. He used failure transfer functions to model failure modes of electrical equipment. In his synthesis tree model, the system-independent component failure transfer functions were used together with the system schematic diagram and associated system boundary conditions to construct the final fault tree. Fussell’s (1973) algorithm was later modified and improved by Taylor (1982) to deal with loops in the system. Taylor used the equation bigraph,
the signal flow graph, and the state transition table to model the fault propagation. Lapp & Powers (1977, 1979) proposed their famous algorithm based on digraphs with "operators" to cope with control loops. A digraph gives an explicit description of the qualitative relationships between the process variables, human errors, and equipment failures. Camarda, Corsi, and Trentadue (1978) proposed an efficient simple algorithm for fault tree synthesis from reliability graphs. In their method, reliability graphs were employed to identify all the minimal path sets, and then transformed into equivalent fault trees. Shafaghi, Lees and Andow (1984) presented a systematic approach to construct fault trees by decomposing the plant into a set of control loops. In their method digraphs were created manually for each control loop and for the plant as a whole. Control loops are treated as the skeleton of the plant. The interconnection and fault propagation between loops were used to construct intermediate trees, and then final fault trees. The fault tree synthesis algorithm of Kelly and Lees, 1986a, 1986b) is based on a mini-fault tree model. Their mini-fault trees are generated from propagation equations, event statements, and decision tables. Bossche, 1985, 1991a, 1991b) used the "relations" between process variables and component states to model the fault propagation in chemical industry. The final fault tree is extracted from an intermediate causal tree. De Vries (1990) developed a quantitative methodology to generate fault tree from a schematic diagram for electrical circuits. Wang and Liu (1993) employed an extended decision table and a virtual transfer component to construct fault trees. Elliott (1994) presented a knowledge-based approach to build a fault tree from a reliability block diagram. User interactions were required to input the knowledge-based rules and the rule interpreter was used to execute fault tree construction rules.

However, some of the above methods are not applicable to complex systems. Some involve generation of a complex system model, such as digraphs, decision tables, transition tables and connection tables between control loops. In particular, control loops are integral parts of a chemical plant. It is crucial in fault tree synthesis, since a control system is designed to prevent process deviations. Fault tree logics to model the behavior of process loops and control loops are one of the most challenging parts and have been the subject of many literature debates for years. No entirely-satisfactory algorithm has been published for fault tree synthesis, especially when control loops are encountered.

In this paper, a potential algorithm for computer-aided fault tree synthesis is developed. This methodology starts directly from a process block diagram. Since mini-fault trees avoid the difficult task of generating digraphs and are more suitable for the automatic construction of fault trees, the method presented here is based on a mini cause-and-effect tree approach. Mini-fault tree cause-and-effect unit models are developed for unit operations such as heat exchangers, compressors, pumps, tanks, etc.
The basic idea is to capture the cause and effect logic between equipment behaviors, human responses, and environment factors around each item of equipment directly into mini cause-and-effect trees. Special unit models are developed to treat control loops. Users can tailor the generalized structures of this special unit model if necessary. Details of the methodology are discussed in the following sections. An example is embedded in the description of the methodology for better understanding. The fault tree generated here is compared to the published result in the analysis section.

2. Methodology

2.1. Overview

The first step of FTA is to identify the undesired accidents. This is normally achieved by hazard identification analysis. In this proposed method, a checklist will be used to select concerned accidents. Users can also input their perceived incidents into the TOP event library.

In order to computerize the whole procedure, system Pipeline & Instrument Diagram (P&ID) and certain configuration and initiating information must be transformed to computer readable formats. The block diagram is used to represent the system topology. In a system block diagram, a solid line between components indicates the physical connection between them, and a dotted line represents the signal transmission between components (typically in control loops, trip systems, and alarm systems). Configuration information, initiating states, and scope of study are incorporated into the system boundary. Once the TOP Events have been identified, the algorithm proceeds to find the unit models in the TOP Event library to trace the TOP Event down to deviations in the process variables in the library, or prompt for user interaction if it does not exist in the library. Then intermediate events are selected and processed in turn iteratively until system boundary or primary events are reached. The algorithm will detect events not existing in the cause-and-effect unit model library and prompt for user interaction. After the user responses, the algorithm will resume. Serial consistency is checked during this procedure. Parallel consistency is performed after the initial tree is constructed. After that, a simplification procedure will be performed to make the fault tree more concise.

Fig. 1 is a simplified flow chart of the algorithm. Some steps are described in detail in the following sections. An example will be used to illustrate the working procedure of the algorithm. To simplify the problem, environment factors are not considered. To limit the space, failures of all the pipes and flanges are not considered as well. However, it is straightforward to incorporate these factors and failures.

Fig. 1. A simplified flow chart of the fault tree synthesis algorithm.

2.2. System boundary

Chemical plants are usually very large and complex. It is impractical and unnecessary to analyze the entire plants at one time, but chemical plants can be easily divided into functional units. To gain a reasonable and sufficient scope of study, the first task is to divide the plant into a set of units and then examine the specified units one by one. Therefore, before applying the methodology, the concerned system and its physical boundary must be well defined. System boundary defines the initial states.
Fig. 2. Nitric acid cooler with temperature feedback and pump-shutdown feedforward (Lapp & Powers, 1977).

of components and the environmental factors to be considered as well.

2.3. System block diagram

In order to automate fault tree synthesis, an appropriate modeling technique must be specified. Before applying the modeling technique, the computer should be able to interpret the system P&ID and other necessary information. In this algorithm, a system block diagram is used to represent the system topology. In a system block diagram, a solid line between components indicates the physical connection between them, and a dotted line represents the information transmission between components (typically in control loops, trip systems, and alarm systems). Each block in a system block diagram represents one item of equipment as illustrated by the simple example presented in Fig. 2, which is adapted from Lapp and Powers (1977). This process is to cool a hot nitric acid stream before entering the reactor, where it reacts with benzene to produce nitrobenzene.

The corresponding block diagram is shown below in Fig. 3. The enclosed dotted rectangle represents the physical boundary of our study. The trip valve and the control valve here are both air-to-open which will close upon loss of instrument air. The signals of temperature sensor and controller will increase with increasing inputs.

2.4. TOP event (accident) library

TOP Event is defined as the undesirable event or incident. The identification of TOP Events is critical but often underestimated. Undesirable accidents can be categorized into three classes as human impacts, environmental impacts, and economic impacts (CCPS, 1992). Typical TOP Events include release of toxic materials, fires, explosions, human injury, environment contamination, property damage, poor product yield/quality, legal liability, etc. Each of these accidents can be subdivided by their characters and consequences. For instance, a fire can be a flash fire, pool fire, jet fire, or a BLEVE.

Industry has used checklists, safety audit, What-if Analysis, HAZOP, and Cause and Consequence Analysis to identify major hazard accidents (CCPS, 1992). It is rather difficult to automate TOP Event identification. In this paper, a checklist will be used to assist identifying undesired events. A checklist is a list of questions about material information, process operations and plant management. Based on the information in the answers to the checklist, the algorithm will generate a list of “possible” TOP Events for the specific process. Users can then examine the list and decide which event(s) to study further. Users can also input perceived TOP Events directly.
Fig. 5. Some unit models for the nitric acid cooling process.
TOP Events are normally undesired accidents such as fires or explosions. Therefore they must be traced to deviations of some process physical variables to start the automatic fault tree construction procedure. At this moment, user interaction is required to do this. However, this process might be assisted by providing generalized cause-and-effect unit models for typical TOP Events such as fire and explosion.

For the above nitric acid cooler, one of the identified accidents can be that runaway reaction occurs in the benzene reactor. Many factors such as large external fire outside the reactor and feeding too fast can cause a runaway reaction. In our system boundary, the only cause of this accident is hot HNO₃ to the reactor. The cause-and-effect unit model for this TOP Event is shown in Fig. 4.

2.5. Cause-and-effect unit model library

Each chemical plant is unique, however, they have much identical equipment, materials, and even processes. For these common components, we have sufficient knowledge of their failure modes. If we can utilize this knowledge and automate this part, it will reduce
Runaway Reaction in the reactor

TS Tout (+1)

TS Tin (+1)

HE Tout (+1)

Fig. 9. The subtree for the TOP Event “runaway reaction in the reactor”.

the labor needed significantly. There is one cause-and-effect unit model in the unit model library corresponding to each item of equipment. Each cause-and-effect unit model may consist of several mini cause-and-effect trees depending on the specific component. When constructing the fault tree, the algorithm will search for appropriate mini-fault trees in the model and apply them.

To save space, we list some cause-and-effect unit models to be used later in the nitric acid cooler example. These figures are only for illustration purpose. The actual library contains more mini cause-and-effect trees, and they may be more complex. In particular, the heat exchanger in the above system is special in that there will be an exothermic reaction if there is an internal leakage. This failure mode is neglected here to simplify the problem. Following the notation of Lapp & Powers (1977, 1979), 1 indicates a moderate disturbance and 10 indicates a large disturbance. The sign of the disturbance represents the relative direction of the deviation, with + meaning increasing and − meaning decreasing. See Fig. 5.

2.6. Cause-and-effect models for control loops

Control loops are integral parts of a chemical plant. It is crucial in fault tree synthesis, since a control system is designed to prevent process deviations. The cause-and-effect unit models for control loops are basically logical combinations of the unit models of their constituent components. Different types of control systems have different unit models. It is not easy to identify and classify control loops automatically. In this algorithm, user interaction is required to input certain information such as control types and configurations for control systems.

There are three circumstances that a controlled process variable can deviate beyond its normal range (Lapp & Powers, 1977). Firstly, uncontrollable disturbances can drive the controlled variable to abnormal states. Secondly, a deviation can be caused by a controllable disturbance while the control loop is inactive. Finally, sometimes the control loop itself can cause process deviations. Based on this, a generalized cause-and-effect unit model is proposed for control loops, see Fig. 6.

Different types of control systems have different mechanisms of failures. Feedback control, feedforward control, nested feedback control, and flow ratio control loops are typically used in chemical processes. Among them, negative feedback control and negative feedforward control are the most popular control schemes. Computer control, alarm system, and manual control systems can be viewed as special forms of feedback or feedforward systems. In this algorithm, each type of control loop has its corresponding cause-and-effect model, which has three mini cause-and-effect trees for “Uncontrollable disturbances”, “Controllable disturbances and control loop inactive”, and “Control loop causes the deviation” events. In the nitric acid cooling process, the temperature control system is a typical negative feedback control loop. The pump shutdown trip system is a negative feedforward control loop. However, upon loss of instrument air, both the control valve and trip valve will close. Therefore “loss of instrument air” will cause
shutdown of the nitric acid cooling process. This is viewed as a negative feedforward control loop. Fig. 7 and Fig. 8 display the cause-and-effect unit models for the negative feedback control loop and the feedforward control loop. User interaction is required for the cause-and-effect model of “control loop causes the deviation” event in feedforward control loops.

### 2.7. Consistency checking

It is essential to check the consistency among tree events to remove inconsistent and unreasonable events. There are two types of consistency — serial and parallel consistency. Serial consistency is the consistency of events with its upper level events. Parallel consistency is the consistency of events between two branches of the same AND gate. During the fault tree construction, the program will store all the upper level parents, check for serial consistency, and delete the inconsistent events whenever encountered. However, parallel consistency checking cannot be performed during construction. It is also necessary to ensure that events in the fault tree do not violate or go beyond the system boundary. This step can be done during fault tree construction.

### 2.8. Initial fault tree synthesis

As stated in section 2.4, one of the identified accidents can be runaway reaction in the reactor in the nitric acid cooling process. Within our system boundary, the only cause is hot HNO₃ to the reactor. According to the system block diagram, this is equivalent to “TS Tout (+1)”. Applying the cause-and-effect model of temperature sensor, “TS Tout(+1)” can be caused by “TS Tin(+1)”.

---

**Fig. 11.** The subtree for “HE Tout (+1)”. 

![Fault Tree Diagram]

The temperature sensor is connected to the heat exchanger in the block diagram, so this is trace to “HE Thot-out(+1)”. See Fig. 9.

HE Tout (+1) is the controlled variable of the temperature control system — a negative feedback control loop. In Fig. 10 we applied the generalized unit model for the control loops.

After applying the cause-and-effect unit model of negative control loops, the unit models for “Th-out (+10)” and “Th-out (+1)” of the heat exchanger are used separately. Thus the fault tree is traced to the output variable deviations of the trip valve, control valve, and pump subsequently. This procedure continues until the system boundary or primary failures are met, as shown in Fig. 11.

When intermediate event “TS Sig (-1)” is traced, one of its child events is “HE Thot-out (-1)”. Serial consistency checking reports that this event is inconsistent with its parent node “HE Thot-out (+1)”. Therefore this event is deleted as shown in the following subtree (Fig. 12).

Following a similar deductive way, we obtain the subtree for the intermediate event “HE Thot-out (+10)” in Fig. 13. As in the previous figure (Fig. 12), “TS Sig (-10)” only has one input — “sensor fails very low” because of serial inconsistency.

If there is no emergency shutdown system, the above fault tree is correct. However, “Pump stops” is the starting point of the trip system (negative feedforward control). The compensating path for this “Pump stops”
event is the signal line and trip valve. Therefore the “Pump stops” event must be replaced by the “Pump stops and the trip system inactive” gate below to take the negative feedforward system into account. The process will also be shutdown by “Loss of instrument air”. “Loss of instrument air” is viewed as the starting deviation of a feedforward control loop as well. The corresponding compensation is the closure of the trip valve upon loss of instrument air. Therefore, in the fault tree, these two events are replaced by the unit model of negative feedforward control loops, see Figs. 14 and 15.

After the initial fault tree has been developed, the algorithm will run parallel consistency checking to remove any inconsistent events between the two arms of AND gates. This example does not have parallel inconsistency.

2.9. Simplification

Fault trees drawn directly from a computer algorithm are normally opaque. A simplification procedure can make them concise and understandable to humans. Two kinds of simplification will be applied --- algebraic simplification and tree simplification. When a certain event (with the probability of 1) is under an OR gate, algebraic simplification is performed to remove the parent gates...
until an AND gate is encountered. If an impossible event (with the probability of 0) is under an AND gate, removal will continue until an OR gate is met. Tree simplification mainly refers to tree suppression. All intermediate events with only one child are removed during tree suppression. After algebraic simplification and tree simplification, identical events under a gate are identified and removed automatically.

From section 2.8, we already have the initial fault tree for the runaway reaction top event. Fig. 16 is the final tree after simplification.

2.10. User interaction

Though a computer will take out the dull part of the methodology, human–machine interfaces are required in many places to select TOP Events, input loop configuration information, and incorporate the cause-and-effect unit models for special process components, etc. Users are responsible for verifying the final fault tree structure. User decision will override a computer most of the time during or after the tree construction and consistency checking stages.

3. Analysis

Many failures in the fault tree of Fig. 16 are not considered in the published fault tree in Lapp and Powers (1977). In order to compare these two fault trees, the analysis resolution should be consistent. By assigning those failures that are not considered in Lapp & Powers (1977, 1979)’ paper a probability of 0 and remove them from OR gates, we can obtain the simplified fault tree shown in Fig. 17.

Calculation of the TOP Event probability involves Minimal Cut Set (MCS) analysis. MCSs are the minimal combinations of basic events that can result in the TOP event. The procedure of MCS analysis transforms the fault tree into an equivalent two-level tree, which is an OR gate of all the possible MCSs. After certain algebraic analysis, the MCSs of the above simplified fault tree are:

1. \{Inlet HNO₃ T (+10)\}
2. \{Inlet HNO₃ F (+10)\}
3. \{Large external fire\}
4. \{CL reversed\}
5. \{CV reversed\}
6. \{CW P(−10)\}
7. \{Loss of instrument air, TV reversed\}
8. \{Pump shutdown, TV reversed\}
9. \{Pump shutdown, Signal line broken\}
10.\{Low instrument air pressure, CL stuck\}
11.\{Low instrument air pressure, TS stuck\}
12.\{Inlet HNO₃ T(+1), CL stuck\}
13.\{Inlet HNO₃ T(+1), TS stuck\}
14.\{Inlet HNO₃ F(+1), CL stuck\}
15.\{Inlet HNO₃ F(+1), TS stuck\}
16.\{External fire, CL stuck\}
17.\{External fire, TS stuck\}
Fig. 16. The final fault tree for the nitric acid cooler.

Fig. 17. The simplified fault tree for the nitric acid cooler.

18. \{\text{CW P}(1), \text{CL stuck}\}
19. \{\text{CW P}(1), \text{TS stuck}\}

As pointed by Lapp and Powers (1979), the probability of the event “Low air pressure of the cooling water control valve” and the event “control valve reversed” occurring simultaneously is negligible. We can simply replace the EOR (exclusive or) gate as OR gate.
The MCSs for the published fault tree in Lapp and Powers (1977) are:

1. \{Inlet HNO\textsubscript{3} T (+10)\}
2. \{Inlet HNO\textsubscript{3} P (+10)\}
3. \{Large external fire\}
4. \{CL reversed\}
5. \{CV reversed\}
6. \{CW P(−10)\}
7. \{Loss of instrument air\}
8. \{Pump shutdown, TV reversed\}
9. \{Pump shutdown, Signal line plugged\}
10. \{Low air pressure (controller), CL stuck\}
11. \{Low air pressure (controller), TS stuck\}
12. \{Inlet HNO\textsubscript{3} T(+1), CL stuck\}
13. \{Inlet HNO\textsubscript{3} T(+1), TS stuck\}
14. \{Inlet HNO\textsubscript{3} P(+1), CL stuck\}
15. \{Inlet HNO\textsubscript{3} P(+1), TS stuck\}
16. \{External fire, CL stuck\}
17. \{External fire, TS stuck\}
18. \{CW P(−1), CL stuck\}
19. \{CW P(−1), TS stuck\}

The minimum cut set \{Loss of instrument air, TV reversed\} is different from that in the published paper \{Loss of instrument air\} because Lapp & Powers, 1977, 1979 did not consider the fact that the trip valve will close upon loss of instrument air. Except for this, the two minimum cut sets are consistent.

4. Conclusion

A new potential methodology to construct fault trees automatically is proposed in this paper. System block diagram and cause-and-effect unit models are employed to model chemical processes, and a simple example is shown. From the analysis section, we can see the fault tree developed by this new algorithm is equivalent to the published result. This algorithm is advantageous in many aspects. The algorithm works directly from the system block diagram and avoids the tedious working of generating digraphs, transition tables, decision tables, and knowledge-based rules. Control loops are considered and treated by special cause-and-effect unit models — logical combinations of the unit models of their constituent components. Multiple or complex control loops can be taken into account by providing the cause-and-effect unit models. In particular, the fault tree construction algorithm presented here is based on a component-by-component basis instead of a loop-by-loop or node-by-node basis. The tree structure is much more concise and easier to read.

To be programmed in computer codes and applied in a real case, the algorithm described above must be tested against substantial examples. Any computer codes must be examined carefully and extensively verified before used in a real application. It is not recommended to use the computer-aided approach alone, since many of the problems in the system design can be discovered during study of the process. However, the automatic fault tree synthesis can be an initial step or as an independent check to assist or supplement a manual FTA.

Acknowledgements

This project is under the support of Mary Kay O’Connor Process Safety Center, Chemical Engineering Department, Texas A&M University.

References