Development of a Multi-Module Event Sequence Modeling Tool for High-Temperature Gas-Cooled Reactors

Pengcheng Peng[1,2,3], Jun Zhao[1,2,3], Jiejuan Tong[1,2,3,*], Hongru Zhao[1,2,3]

[1] *Institute of Nuclear and New Energy Technology, Tsinghua University, Beijing, 100084, China*
[2] *Collaborative Innovation Center of Advanced Nuclear Energy Technology, Beijing 100084, China*
[3] *The Key Laboratory of Advanced Reactor Engineering and Safety, Ministry of Education, Beijing, 100084, China*
[*] *Corresponding author: J. Tong (tongjj@tsinghua.edu.cn)*

ABSTRACT

The high-temperature gas-cooled reactor (HTGR) in China adopts a modular design of multiple reactor modules serving one single unit. For the relationships within one unit even among different units, accident scenario analysis involving multiple reactor modules becomes an essential consideration in probabilistic safety assessment in HTGRs. It is still applicable for modeling event sequences in HTGR by using traditional event tree analysis, and the multi-module event trees are typically modeled by combining event trees of single modules. In this way, however, the sequences of multi-module event trees will increase exponentially with growth in the number of reactors affected by the accident, which will consume not only a huge manpower but also make it difficult to ensure the absence of errors during the combining process so that there exist challenges in the establishment and display of those event trees. This paper designs and develops a program tool that can generate multi-module event trees automatically based on the combination of single-module event trees. This tool has a browser/server architecture, and is capable of achieving unified-format import and export of event trees, combining event trees by setting rules, and automatically generating and displaying multi-module event trees.

Keywords: Multi-module event tree, event tree combination, probabilistic safety assessment, high-temperature gas-cooled reactor

I. INTRODUCTION

The modularisation of small reactors becomes a significant trend in the nuclear power industry and a typical example is the high-temperature gas-cooled reactors (HTGR) where multiple reactor modules will service a single turbine at the same time [1]. That design strengthens the interdependencies between modules within the unit of HTGR, which requires the event sequence analysis, as the most fundamental part of probabilistic safety analysis (PSA), to address accident scenarios involving multiple modules. For this reason, Zhang et al. [2-6] established a multi-module PSA framework for HTGR and developed a tool for multi-module event sequence modelling. Under this framework, the event sequences of multi-module event trees (METs) are defined as combinations of ones from the single-module event trees (SETs). In order to realize the combination of event trees automatically, it is necessary to model the elements used in the whole combination process. According to the engineering experience in practice, the following three issues should be considered in that process.

1. Graphical representation of METs. To facilitate the combination and screening of event sequences, the framework mentioned above provides a vector representation of METs [5]. However, it will only reserve valid combinations so that MET may not retain a complete tree structure, which means that it could not be converted into graphical representation. Since graphical representation will display a clear logic of the development of event sequences to PSA specialists, it is an indispensable form of expression in practice. We need a suitable method to enable mutual conversion between these two representations.
2. Classification of multi-module release categories. The probabilistic safety objective of HTGRs in China focuses on the type of radioactive release (RC50) that cause individual effective doses exceeding 50mSv on the plant boundary [7]. In practice, as the number of modules in HTGRs increases, say a six-module design in [8], combinations of the releases (non-RC50) from single module that will not lead to RC50 may result in RC50 consequences, thereby the

non-RC50 release combinations should be considered as well. It imposes higher requirements on the combination of event trees.

3. The scalability of modelling tools. To facilitate the exchange and application of PSA model, a standard interactive format of MET and its related elements is needed, such as the XML representation developed by the Open-PSA project [9].

To address the issues above, this paper presents the theory of event tree combination where some technical elements in the combination process are revised, compared to the framework proposed in [5], say the conversion between different representations of event tree and the automatic classification of multi-module releases. Based on this theory and a modelling tool of MUETBASE in [4], we will develop a more comprehensive event sequence modelling tool of HTGRs, which achieves automatic modelling and improves the scalability of the programme tool.

The remainder content is organized as followings. Chapter II provides the theory of event tree combination from three aspects: definition, representation methods, and combination rules. Chapter III introduces an event sequence modelling tool of HTGRs that is developed on the basis of that theory. The conclusion is presented at last chapter.

II. THEORY OF EVENT TREE COMBINATION

II.A. Definition of Multi-Module Event Tree

II.A.1. Initiating Event

An initiating event involving multiple modules is defined as an initiating event that affects two or more reactor modules simultaneously. Such events are often identified on the basis of initiating events from PSA model for single module and involve shared structures, systems, or components. Taking the HTR-PM with two modules as an example, loss of coolant that can be isolated will include breaks at the pipe sections shared by two modules, which should be separately classified as two-module initiating events, while the breaks at other positions are still the single-module initiating events. Also, loss of off-site power will simultaneously affect both modules and should be directly regarded as a two-module initiating event.

II.A.2. Function Event

The function events (hereinafter referred to as headers) of a MET are obtained by combining headers from SETs in order, where any shared headers (if any) will be merged to the first occurrence. For example, if the headers of two SETs are $\{F_1, F_2, F_3\}$ and $\{F_4, F_2\}$ respectively, then the headers of the two-module event tree generated by these two SETs will be $\{F_1, F_2, F_3, F_4\}$.

II.A.3. Event Sequence

Similarly, multi-module event sequences could be obtained by combining sequences from single modules in order, where the states of the shared headers (if any) are determined by the constraints of the shared systems that correspond to these headers. Let two sequences from single modules that corresponds to headers of $\{F_1, F_2\}$ and $\{F_3, F_2, F_4\}$ respectively, be $\{F_1$ success, F_2 success$\}$ and $\{F_3$ success, F_2 success, F_4 failure$\}$, where the shared header of F_2 have success states in both sequences. If the shared system corresponding to F_2 is available for use by both modules at the same time, these two sequences can be combined to generate a two-module event sequence of $\{F_1$ success, F_2 success, F_3 success, F_4 failure$\}$. If the shared system can only be called by one module at a time, the states of F_2 in these two sequences are contradictory, which means that the combination is invalid, and it should be screened out. How to select valid combinations will be illustrated in Section II.C.1.

II.A.4. Release Category

The consequences (also refereed as release categories) of multi-module event sequences, can be roughly divided into three types: RC50, non-RC50, and OK. Among them, RC50 corresponds to situations where the consequences could reach RC50 level through direct superposition of the consequences from single module; OK applies to combinations where none of the consequences from single module result in radioactive release; and all the other cases correspond to non-RC50. Take HTR-PM with two modules as an example. Since the direct superposition of any non-RC50 consequences from single module will not reach the RC50 level, for two-module event sequences with RC50 consequences, the combinations of consequences from single module include three cases, i.e., {RC50, RC50}, {RC50, non-RC50}, and {RC50, OK}.

II.B. Event Tree Representation

II.B.1. Graphical Representation

Graphical representation is the most conventional form of event trees. It will display all possible developments in a sequence of events with a tree structure. That is to say, starting from the initiating event (root node), two branches representing success and failure are generated at each header (branch node), which will deduce different sequences (paths) and the corresponding consequences (leaf nodes). In Figure 1, N_1 is the root node, N_2 and N_3 are branch nodes, and N_4, N_5, and N_6 are leaf nodes. The three paths from the root node to the leaf nodes correspond to the event sequences with consequences of C_1, C_2, and C_3 respectively. If the header does not generate a branch, it means that neither success or failure of it could influence the consequence of that sequence.

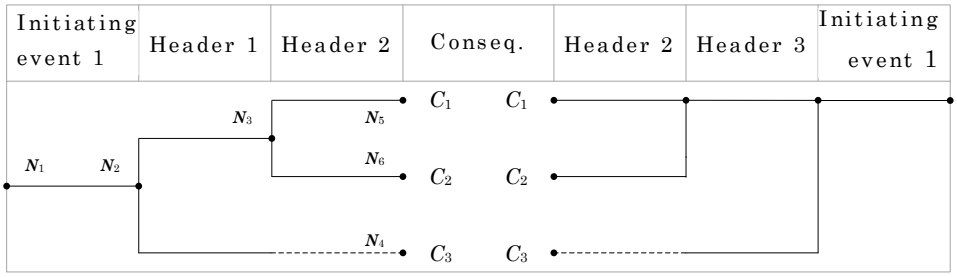

FIGURE 1. Two Conventional Ways of Drawing Event Trees

II.B.2. Vector Representation

The vector representation [5] could facilitate the combination of event trees from the perspective of program implementation. Each event sequence is denoted by a vector of state values of the headers, which could be further converted into an integer for storage. Let us consider the event tree on the left side of Figure 1. All the possible states of the header are failure, success, and inapplicable, denoted by 0, 1, and 2 respectively. For instance, the second sequence of the event tree can then be written as $\boldsymbol{S}_2 = (1, 0)^{\mathrm{T}}$. It corresponds to the ternary number of 10, equaling to the integer $3 = 3^1 \times 1 + 3^0 \times 0$. Hence, those three sequences of the event tree can be converted to 4, 3, and 2 respectively, and the following vector representation is obtained by

$$\mathrm{ET} = [\, a\,;\ n_f, \boldsymbol{D}_f\,;\ n_s, \boldsymbol{D}_c, \boldsymbol{L}\,] = \left[1;\quad 2, \begin{pmatrix}1\\2\end{pmatrix};\quad 3, \begin{pmatrix}1\\2\\3\end{pmatrix}, \begin{pmatrix}4\\3\\2\end{pmatrix}\right],$$

where a is the index of the initiating event; n_f is the number of headers; $\boldsymbol{D}_f$ is the vector composed of indexes of the headers; n_s is the number of event sequences; $\boldsymbol{D}_c$ is the vector composed of indexes of the consequences; and $\boldsymbol{L}$ is the vector of integers corresponding to the sequences. METs can also be written in this manner (see Section II.C.1).

II.B.3. Mutual Conversion

The vector representation is characterized by the simplicity in form, ease of storage, and programmability. It is effective in addressing the modeling challenges posed by METs, and thereby leading to a substantial reduction in manual modeling expenses. However, the graphical representation is commonly used in most PSA modeling tools for its readability, and it is essential to convert vector representation of event trees into graphical representation. The followings in this section will present a conversion method.

For the sake of illustration, we first introduce an intermediate form of event trees, referred to as the event tree table. It records all event sequences of the event tree in a tabular form. The first two rows record the initiating event and the headers respectively, with each subsequent row recording an event sequence; the last column of each row records the consequences, and the remaining columns record the state values of the headers. In Figure 2, there is an example of an event tree table with a scale of 3 rows and 3 columns, and the relationships between the three representations are demonstrated as well, where the

graphical representation, tabular representation and vector representation are denoted by ETGraph, ETTable and ETVector respectively.

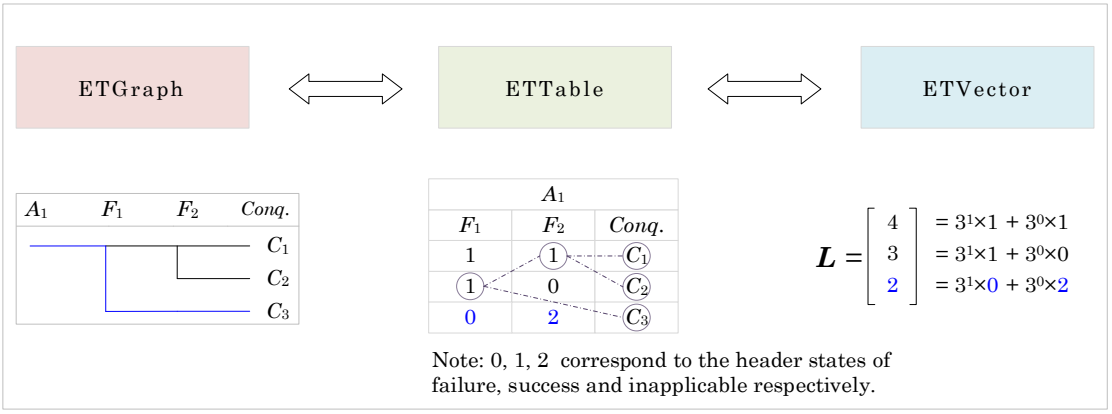

FIGURE 2. Relationships Between Graphical Representation and Vector Representation of Event Trees

The conversion of event trees from graphical representation to vector representation is a natural process. In ETGraph, every event sequence is shown as a path in a binary tree, and it corresponds a single row in ETTable where the state values of 0 and 1 in the table represent the failure and success branches generated by the header of event tree. If there is no branch for the header in a path, the state value is designated as 2. For any row in the table, the state values of the headers from left to right correspond to a ternary number from high bits to low bits. Then, the element of the vector in ETVector could be obtained by converting that number to an integer.

However, the ETVector of a MET with shared header(s) may not have a complete tree structure when converted to ETGraph. According to the definition of event sequences in Section II.A.3, when combining sequences from single modules to generate MET, we have to merge shared headers, and those combinations that do not satisfy the constrains of the shard systems corresponding to the headers will be screened out. That will affect the tree structure of this MET. A two-module ETTable generated by two event trees from Figure 1 is shown on the left side of Figure 3 to illustrate this situation.

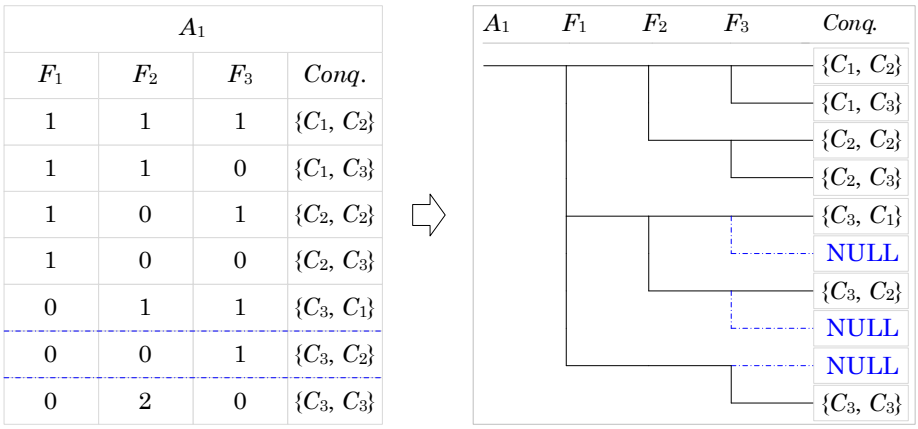

FIGURE 3. The First Way of Converting an Event Tree Table into Graphical Representation

It is obvious that this event tree requires supplementation with event sequences (denoted in blue in Figure 3) to establish a complete tree structure, when converting the table to graphical representation. To address this issue, we provide two distinct approaches below.

1. Complete the tree structure with minimal branches (MBs). The following algorithm is designed to verify whether the event sequences in the event tree table have a complete tree structure, and to automatically add the missing branches with minimal times.

TABLE I. The Algorithm for Completing the Tree Structure With Minimal Branches

Function	$\boldsymbol{S}$:= MBs({$\boldsymbol{S}_1$, ..., $\boldsymbol{S}_{ns}$})
Input	{$\boldsymbol{S}_1$, ..., $\boldsymbol{S}_{ns}$}, the vector representation of all event sequences in an event tree.
Output	$\boldsymbol{S}$, the collection of event sequences that could be converted into graphical representation after completing the tree structure.
1.	Verify the applicability of this algorithm for these event sequences. Define $\boldsymbol{S}$ = { } as the collection of the valid sequences. For any two sequences $\boldsymbol{S}_i = (f_{i1}, ..., f_{im})^T$, $\boldsymbol{S}_j = (f_{j1}, ..., f_{jm})^T$, perform the following process.
	1.1. A contradiction of having success (or failure) and inapplicable states at the same time in local branches. If $1_{\{2\}}(f_{i1}) + 1_{\{2\}}(f_{j1}) = 1$, or, $f_{ik} = f_{jk} = 1$ and $1_{\{2\}}(f_{i(k+1)}) + 1_{\{2\}}(f_{j(k+1)}) = 1$, then go to Step 5. Note that $1_{\{2\}}(\ )$ is the indicator function for the set of {2}. 1.2. Otherwise, go to Step 2.
2.	Initialization. Define two sequences $\boldsymbol{S}_{\text{this}} = (f_1, ..., f_m)^T$, $\boldsymbol{S}_{\text{next}} = (h_1, ..., h_m)^T$, where f_i, h_i = 0, 1 或 2, i = 1, ..., m, and t = 1. Set $\boldsymbol{S}_{\text{this}}$ to be the first sequence, i.e., $\boldsymbol{S}_{\text{this}} = \boldsymbol{S}_1$. Construct a sequence $\boldsymbol{S}_{\text{fit}}$ where its i-th element $(\boldsymbol{S}_{\text{fit}})_i = 1 + 1_{\{2\}}(f_i)$.
	2.1. If there exists any zero element in $\boldsymbol{S}_{\text{this}}$, then $\boldsymbol{S}_{\text{this}} = \boldsymbol{S}_{\text{fit}}$, $\boldsymbol{S} = \boldsymbol{S} \cup \boldsymbol{S}_{\text{fit}}$, t = 0 and go to Step 3. 2.2. Otherwise, $\boldsymbol{S} = \boldsymbol{S} \cup \boldsymbol{S}_1$, and go to Step 3.
3.	Check the current sequence. Denote the set of indices of elements with a value of 1 in $\boldsymbol{S}_{\text{this}} = (f_1, ..., f_m)^T$, by $K = \{k_1, ..., k_p\}$, which means that for any $k \in K$, we have f_k = 1.
	3.1. If K equals to empty set and $\boldsymbol{S}_{\text{this}}$ to the last sequence (or say $t = n_s$), then go to Step 5. 3.2. If K equals to empty set but $t < n_s$, then let K = {1} and go to Step 4. 3.3. Otherwise, go to Step 4.
4.	Match the adjacent sequence. Let $\boldsymbol{S}_{\text{next}} = \boldsymbol{S}_{t+1}$ and $\boldsymbol{S}_{\text{fit}} = (f_1, ..., f_{kp-1}, 0, 2, ..., 2)^T$ where the first k_p - 1 elements of $\boldsymbol{S}_{\text{fit}}$ are the same as those of $\boldsymbol{S}_{\text{this}}$, the k_p-th element is 0, and the left elements (if any) are 2s.
	4.1. If $\boldsymbol{S}_{\text{next}} = \boldsymbol{S}_{\text{fit}}$, then $\boldsymbol{S}_{\text{this}} = \boldsymbol{S}_{\text{next}}$, $\boldsymbol{S} = \boldsymbol{S} \cup \boldsymbol{S}_{\text{next}}$, $t = t$ +1 and go to Step 3. 4.2. Otherwise, let $\boldsymbol{S}_{\text{fit}} = (f_1, ..., f_{kp-1}, 0, 1 + 1_{\{2\}}(h_{kp+1}), ..., 1 + 1_{\{2\}}(h_m))^T$. 4.2.1. If $\boldsymbol{S}_{\text{next}} = \boldsymbol{S}_{\text{fit}}$, then $\boldsymbol{S}_{\text{this}} = \boldsymbol{S}_{\text{next}}$, $\boldsymbol{S} = \boldsymbol{S} \cup \boldsymbol{S}_{\text{next}}$, $t = t$ +1, and go to Step 3. 4.2.2. Otherwise, $\boldsymbol{S}_{\text{this}} = \boldsymbol{S}_{\text{fit}}$, $\boldsymbol{S} = \boldsymbol{S} \cup \boldsymbol{S}_{\text{fit}}$, and go to Step 3.
5.	Return $\boldsymbol{S}$.

2. Establish the minimum cut sets with respect to function event (FE-MCSs). The main idea of this approach is as follows: for a certain multi-module consequence, take the headers with failure state in each sequence that leads to the interest consequence, as cut sets, have the FE-MCSs by eliminating all redundant ones, and then establish a MET with a single one header defined by these FE-MCSs [6]. Figure 4 shows an illustrative example.

A_1			
F_1	F_2	F_3	*Conq.*
1	1	1	{C_1, C_2}
1	1	0	{C_1, C_3}
1	0	1	{C_2, C_2}
1	0	0	{C_2, C_3}
0	1	1	{C_3, C_1}
0	0	1	{C_3, C_2}
0	2	0	{C_3, C_3}

FE-CS with RC50 (C3)
{F_3}
{F_2, F_3}
{F_1}
{F_1, F_2}
{F_1, F_3}

FE-MCS
{F_3}
{F_1}

Graphical representation of two-module event tree

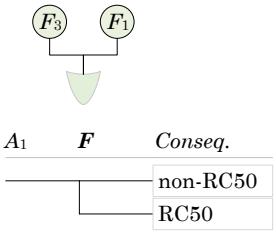

FIGURE 4. The Second Way of Converting an Event Tree Table into Graphical Representation

Define C_3 as the consequence of RC50 for a two-module HTGR. According to the definition of two-module consequence in Section II.A.4, the combinations of consequences from single modules could result in RC50 only when these two consequences contain RC50. The event tree table has five multi-module event sequences with RC50 consequence, and thus we

have five cut sets of {F_3}, {F_2, F_3}, {F_1}, {F_1, F_2}, {F_1, F_3}. The FE-MCSs of these cut sets are {F_1}, {F_3}. Hence, we could build the two-module event tree shown on the right side in Figure 4, where its header is named as "no RC50 release" and its failure criteria is defined by the union of FE-MCSs, i.e., {F_1} OR {F_3} in logic.

II.C. Combination of Event Trees

II.C.1. Screening Criterion

A multi-module event sequence could be obtained by combining sequences from single modules. If there is a shared header in those sequences, the states of that header in all the sequences need to satisfy the constrains of the shared system corresponding to the header. These constrains set the screening criterion for event sequence combinations. Table II presents a set of general screening criterion based on individual header [5], which can be applied to each header in the combinations of sequences from single module for screening the valid ones as the multi-module event sequence.

TABLE II. Screening Criterion for Event Sequence Combinations

Code	Classification characteristics		Screening rules	Scenario analysis
	A shared header?	MAU		
I	No	0	No need to build rules and reserve all combinations.	The scenario is unique.
II	Yes	$= n_t$	The states of the header are the same in all single-module event sequences.	The state of the header is the state in the sequence.
III	Yes	$< n_t$	Any single-module event sequence in the combination adhere to one of the following criteria: 1. The number of sequences with the header having a success state must equal MAU.	The sequences with the header having a failure state means that the corresponding module cannot call the shared system due to the occupation of it by other modules. The state of the header is success.
			2. There are no sequences with the header having a failure state, and the number of sequences with the header having a success state is greater than 0 and less than MAU.	The state of the header is success.
			3. There are no sequences with the header having a success state.	Assume that there exist k sequences with the header having a failure state. If $k = 0$, then the stare of the header is inapplicable. If $0 < k \leq$ MAU, the stare of the header is failure. If MAU $< k \leq n_t$, then there are C_k^{MAU} scenarios, corresponding to all the combinations of cases where MAU of k modules called the shared system and thereby it cannot be called by the remaining k - MAU modules. In such cases, the stare of the header is failure.
Notes: (1) nt is the number of SETs used for combination, which often equals to the total number of modules in HTGRs; (2) MAU is the maximum allowable users.				

Denote these two event trees from left to right in Figure 1 by ET_1, ET_2 and their event sequences by $\boldsymbol{S}_{1i}$, $\boldsymbol{S}_{2j}$, respectively, where $i = 1, 2, 3$, and $j = 1, ..., 5$. Assume that the only one shared system corresponding to F_2 allows at most one module to call. Let us consider the combination of $\boldsymbol{S}_{12} = (1, 0)^T$ and $\boldsymbol{S}_{22} = (1, 0)^T$. We first apply Criteria I to F_1 and F_3 in order, and reserve it. Then, apply Criteria III to F_2. Since F_2 has a state of 0 in both sequences, it satisfies Criteria III.3. Therefore, this combination is valid and written as a multi-module sequence of $(1, 0, 1)^T$. Note that this multi-module sequence actually comprises two scenarios: (1) module 1 calls the shared system, precluding module 2 to call it, and (2) module 1 is unable to call it due to a prior call by module 2. Sequences generated by these two scenarios are identical, and we could select one of them as a

representative. In fact, duplicate sequences will be eliminated in logic. Similarly, we could have the remaining combinations, and further obtain the two-module event tree shown in Figure 3.

II.C.2. Release Classification

As the number of HTGR modules increases, say a six-module design of HTR-PM600 [8], the combinations of consequences of non-RC50 from single module may lead to RC50. It is necessary to further refine the classification of multi-module release categories. The release categories from single module in HTGRs are primarily classified according to three characteristics: the magnitude of radioactive material release (determined by accident type), the integrity of the primary-loop pressure boundary, and the filtration of radioactive material controlled by the ventilation system. If the function of f quantifying the relationships between single-module release categories and their corresponding individual effective doses at the plant boundary is known, then we could classify the combinations of consequences from single modules into different multi-module release categories by directly summing these doses caused by single-module event sequences.

For instance, given any m-module event sequence, the combination of consequences from single modules is written as $\{C_1, ..., C_m\}$. The individual effective dose for this sequence could be defined as follows:

$$y = \sum_{i=1}^{m} f(C_i).$$

For the sake of illustration, we still use RC50 to denote the dose of 50 mSv, and the interval of the dose could be divided into different subintervals by a integer z: {0}, (0, RC50/z), [RC50/z, 2*RC50/z), ..., [(z-1)*RC50/z, RC50), [RC50, +∞). These intervals will define the multi-module release categories of OK, RC50_1, ..., RC50_z and RC50 respectively. If y = 0, the consequence is OK. If 0 < y < RC50, then the consequence is RC50_[1+y/z], where the symbol of [ ] is rounding function. Otherwise, the consequence is RC50.

II.C.3. Elimination Rules

In the process to have FE-MCSs, the key procedure is to perform elimination between these cut sets, as it will simplify the scale of METs at a certain extent. The utilization of vector representation could facilitate the establishment of elimination rules.

For an event sequence, the state values of the headers arranged from left to right, correspond to a ternary number from high bits to low bits. The elimination rules between sequences can be defined through the implementation of bit operations between ternary numbers. As illustrated in Figure 4, the sequences of $\boldsymbol{S}_2 = (1, 1, 0)^T$, $\boldsymbol{S}_4 = (1, 0, 0)^T$ correspond to the ternary numbers of z_2 = 110 and z_4 = 100, respectively. $\boldsymbol{S}_2$ eliminate $\boldsymbol{S}_4$, is equivalent to that $z_2|z_4$ = 110|100 = 110 = z_2, where the symbol of | is the bitwise operation of OR in logic. In fact, for sequences with any header having a state value of 2, their elimination relationships with others are equivalent to the one between the sequences revised by replacing all 2s with 1s and the others. Hence, we can directly use the binary logical operation to swiftly determine the elimination relationship between sequences. Say $\boldsymbol{S}_7 = (0, 2, 0)^T$ corresponds to the binary number of z_7 = 010. Obviously, $\boldsymbol{S}_2$ could also eliminate $\boldsymbol{S}_7$ for $z_2|z_7 = z_2$. Let any two sequences of $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ have the corresponding binary numbers z_i and z_j respectively, then $\boldsymbol{S}_i$ can eliminate $\boldsymbol{S}_j$ if and only if

$$z_i|z_j = z_i.$$

III. THE MULTI-MODULE EVENT SEQUENCE MODELING TOOL

III.A. Functional Framework

The Multi-Module Event Sequence Modeling Tool for HTGRs (MUETBASE 2.0) has six types of function shown in Figure 5.

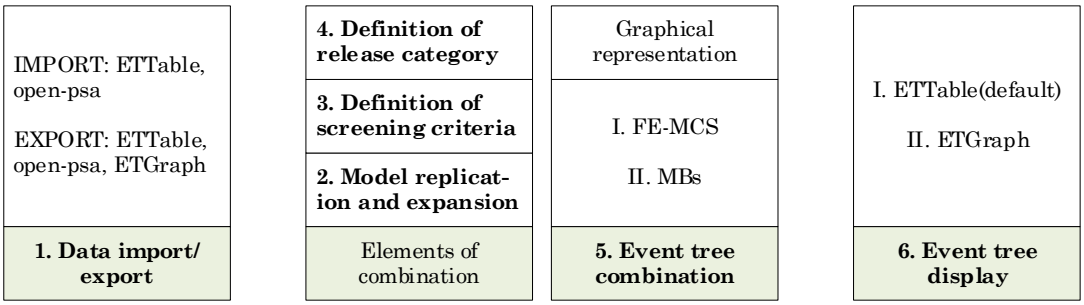

FIGURE 5. The Functional Framework of Multi-Module Event Sequence Modeling Tool for HTGRs

1. Data import/export. This tool supports both importing and exporting event trees in two distinct formats of event tree table and Open-PSA, and it also enables the export of event tree graphics. Both SETs and METs have four data elements: initiating events, function events, event sequences, and release categories.
2. Model replication and expansion. After having a SET and the total number of modules, the SETs of other modules can be replicated automatically to support the combination of SETs.
3. Definition of screening criteria. The screening criterion for the combination of event sequences from single modules can be assigned by selecting a specific function event and its screening rule presented in Section II.C.1.
4. Definition of multi-module release categories. The combinations of consequences from single modules can be automatically categorized by specifying the mapping from single-module release categories to the corresponding individual effective doses.
5. Event tree combination. Specify SETs and screening criterion for combination, and then a MET could be generated automatically. Two ways of graphical generation are provided: (1) according to the release categories, generate the FE-MCSs by eliminating sequences with the same specific release category, and (2) complete the tree structure with minimal branches.
6. Event tree display. Two display modes are available: (1) the default mode of event tree table, and (2) the event tree graphic.

Compared to the previous version of MUETBASE, this tool has improved the functions of graphical display and data exchange, including (1) the import and export of event tree data in the format of Open-PSA, (2) model replication, (3) the classification of multi-module release categories, and (4) two ways of graphical generation.

III.B. Program Implementation

The implementation of this tool uses a browser/server (B/S) network architecture, allowing users to directly access the operating interface of it via a web browser. All the data of event trees is stored and managed by using SQL Server Express, and algorithms of all the calculations involved in the process of event tree combination are still implemented by using C/C++.

IV. CONCLUSION

This paper presents a theory of event tree combination for HTGRs and develops a corresponding modeling tool of MUETBASE 2.0. In this theory, all the elements of METs, including initiating events, function events, event sequences, and release categories are clearly defined; two representations of event trees and their relationships for mutual conversion are presented; and three key technical elements of event tree combination are identified, including screening criterion, classification of release categories, and elimination rules of the minimum cut sets with respect to function events. Accordingly, an update tool is developed based on this theory and MUETBASE.

The future work will concern the following two aspects for optimizing and upgrading this modeling tool.

1. Try to model the classification of the plant operating states (POSs) and the identification of initiating events for HTGRs in this tool. Under the different POSs, the plant configurations and response requirements may vary significantly. Therefore, the identification of initiating events should be conducted on the basis of the classified POSs. Liu et al. [3] proposed a POS analysis method for multi-unit nuclear power plants. Incorporating the technical elements of this method into the program would enable a more automatic modeling process.
2. Develop a simplified fault tree solver by combining the approximate algorithms and numerical truncation, to achieve the quantification of event trees.

ACKNOWLEDGMENTS

This work is supported by the National Science and Technology Major Project of People's Republic of China (Grant No.2018ZX06902015).

REFERENCES

[1] IAEA. Nuclear Energy Series No. NR-T-1.18: Technology Roadmap for Small Modular Reactor Deployment. 2021.
[2] Sai Zhang, Jiejuan Tong, Jun Zhao. An integrated modeling approach for event sequence development in multi-unit probabilistic risk assessment. Reliability Engineering and System Safety, Volume 155, 2016.
[3] Ao Liu, Pengcheng Peng, et al. A plant operating state analysis method in probabilistic safety assessment for multi-unit nuclear power plant. Annals of Nuclear Energy, Volume 169, 2022.
[4] Ao Liu, Pengcheng Peng, et al. An event sequence modeling method in multi-unit probabilistic risk assessment for high temperature gas-cooled reactor. Annals of Nuclear Energy, Volume 182, 2023.
[5] Pengcheng Peng, Jiejuan Tong, Jun Zhao. A rapid approach to generate multi-unit event trees based on tree combinatorial calculation. Progress in Nuclear Energy, Volume 162, 2023.
[6] Hongru Zhao, Jiejuan Tong, et al. An FE-MCS based modeling method for event sequence development for the multi-module probabilistic risk assessment. Progress in Nuclear Energy, Volume 170, 2024.
[7] National Nuclear Safety Administration of P.R. China. Regulatory Safety Review Principles for HTR-PM. 2008.
[8] Zuoyi Zhang, Yujie. Dong, et al. 600-MWe high-temperature gas-cooled reactor nuclear power plant HTR-PM600. Nuclear Science and Techniques, Volume 33, 2022.
[9] Steven Epstein, Rauzy Antoine. Open-PSA Model Exchange Format. 2017.