LINKING OF EVENT TREES VIA ANALYSIS RESULTS

Ola Bäckström[1], Rory Gamble[1], Pavel Krcal[1], Johan Sörman[1], Wei Wang[1]

[1] Lloyd's Register, Box 1288, 17225, Stockholm, Sweden
{Ola.Backstrom, Rory.Gamble, Pavel.Krcal, Johan.Sorman, Wang.Wei}@lr.org

We introduce a new way of linking event trees - not directly by connecting sequences of father event trees to initiating events, but indirectly through analysis results. This approach greatly improves efficiency of generating minimal cutset lists. It allows analysts to modularize their calculations, run them separately and reuse previously computed results. It is also possible to fine-tune analysis settings for each calculation. An example when such modularization could be relevant is the link between level 1 and 2. PSA level 1 has already calculated the results for core damage, and the level 2 is a continuation of the sequences. It may be noticed that the calculation time for an integrated PSA level 2 is normally much longer than PSA level 1, even though the containment event trees are small. The suggested approach would resolve that problem and the results from PSA level 1 may be reused.

I. INTRODUCTION

Complexity and size of real-life PSA studies requires that event trees covering the whole sequence from the initiating event to a core damage consequence or to release fraction consequences are split into shorter event trees linked together by connections between sequences in the 'earlier' event tree and initiating events in the 'later' event tree. In some situations, such as in the case of the link between PSA level 1 and PSA level 2, the analysis presents results for both all relevant sequences of the 'earlier' event tree as well as all the relevant sequences of the 'later' event tree. In the example of PSA level 1 and level 2, we are interested in the results for core damage (PSA level 1), and also in the PSA level 2, which is a continuation of these sequences.

A typical analysis approach prepares analysis cases for both PSA level 1 consequences and also for PSA level 2 consequences. These analysis cases build and calculate a model starting with initiating events and ending with analyzed consequences. In both cases, the part for PSA level 1 is included. This normally results in calculation times for an integrated PSA level 2 that are much longer than for PSA level 1, in spite of the fact that the containment event trees are small.

We introduce a new way of linking event trees which allows reusing previously computed results as an input to an initiating event to a new analysis case. By this, we link the sequences yielding the previously calculated results with the sequences starting with the initiating event using these previously calculated results as input. This approach has a number of advantages. First, it allows analysts to reuse already calculated results, which greatly improves efficiency of generating minimal cutset lists. This is not only beneficial in the final quantification, but also during model updates and result verification, when many calculations with slightly altered logic might be executed. Second, this approach gives analysts a possibility to modularize calculations and fine-tune settings differently for each part of the analysis. We describe the method for solving analysis cases which use MCS results as inputs to initiating events. This new approach to linking of event trees has been implemented in RiskSpectrum. We also formulate a correctness criterion for these new types of analyses. Intuitively, results obtained from this analysis starting with pre-calculated results should be the same as for an integrated analysis where all event trees from initiating events to analyzed consequences are included.

We analyze properties of the proposed method with respect to this correctness criterion and we formulate certain restrictions on the analysis cases whose results are to be reused in other analysis cases. These limitations are summarized in modelling guidelines one should follow to avoid possibly unwanted effects of this method. We also illuminate sources of potential issues, mainly the effects of not-logic in presence of dependencies between analysis cases. Finally, we suggest improvements to the presented method that would resolve these issues and allow for unrestricted use of not-logic even for dependent analysis cases.

The approach presented in this paper is inspired by problems that occur in real-life models built according to the fault tree linking approach[1, 2]. The aim here is not to contribute to comparisons of the fault tree linking and event tree linking[3] or to propose any resolutions of the dichotomy.

II. EVENT TREE LINKING VIA ANALYSIS RESULTS

The proposed method links event trees by their analysis results, instead of directly connecting the event tree logic in the conventional way. Event trees are conventionally joined by a *link event*, this event is consequence of the first event tree (the *input* event tree) and is used as the initiating event of a second event tree (the *main* event tree). We term this *direct linking* of event trees. Analysis of the main ET implicitly includes analysis of the input ET because of the linking event: when building the master fault tree for a sequence in the main ET, the initiating event is replaced with the logical structure of the input ET sequences which generate the linking event. The calculation algorithm then quantifies this combined logical structure. Notice that in the case of a model with many linked event trees, an event which links the event trees may be repeated many times, which may involve repetition of this part of the calculation.

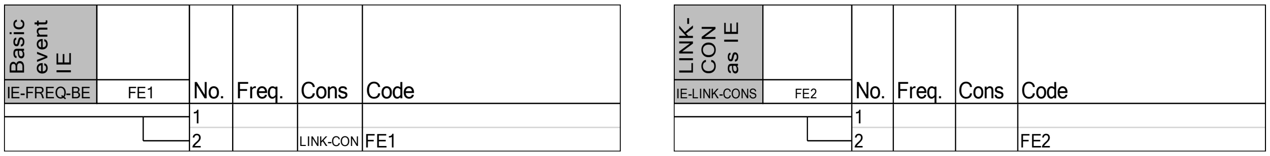
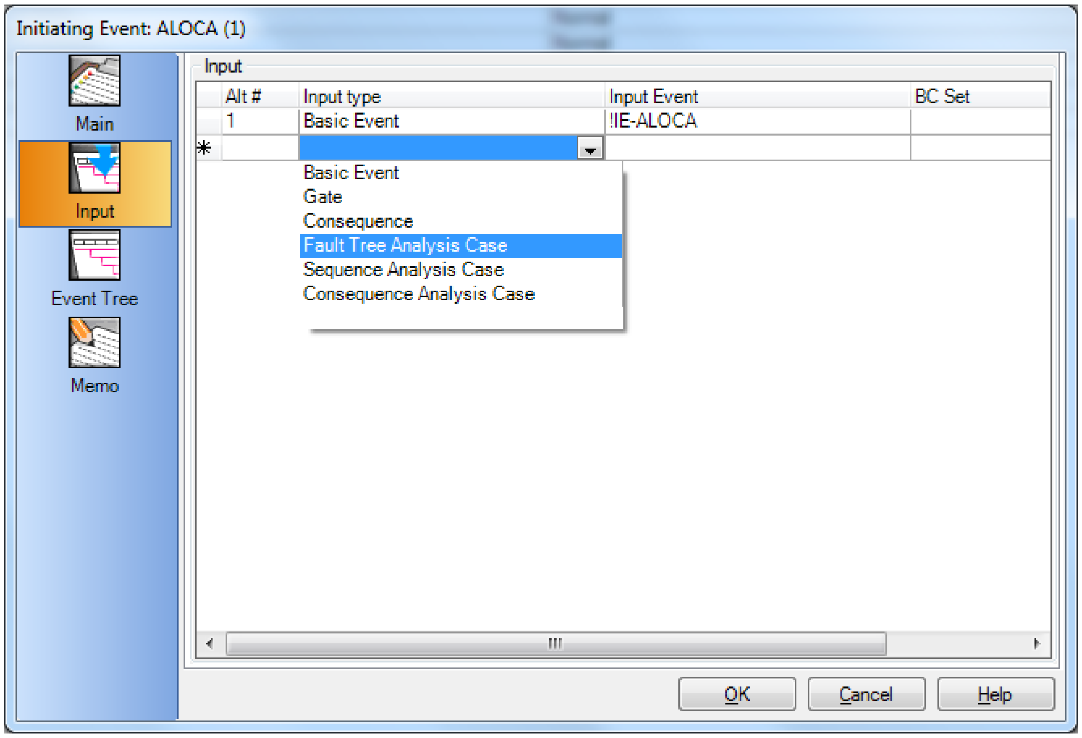


Figure 1. Sample event trees linked via the consequence LINK-CON. The second sequence in the first event tree continues in the initiating event of the second event tree.

In the new method, an MCS list may be used as an input to an initiating event. This is achieved in RiskSpectrum PSA by selecting an analysis case as the input type for the initiating event. In the current implementation, one can also select a Fault Tree Analysis Case apart from a Sequence Analysis Case or a Consequence Analysis Case. In the following, we will for simplicity consider a Fault Tree Analysis Case as a special case of a Sequence Analysis Case and still talk about the input event tree whenever suitable.



	Alt #	Input type	Input Event	BC Set
	1	Basic Event	!IE-ALOCA	
*				

Figure 2. New types of initiating event inputs as implemented in RiskSpectrum.

Logically, the main analysis case is analyzed with a placeholder initiating event. After MCS analysis has been performed, the MCS list is expanded by iteratively replacing this placeholder event with each of the cutsets from the input MCS list. A further minimization step is performed to remove any repeated or non-minimal cutsets which may have been generated. In the ideal case, this process leads to cutsets which fail both the event tree structure and the MCS list (i.e., they are a superset of a cutset from this MCS list). A complete description of the algorithm together with a description of its advantages is presented in the next section.

Some limitations exist with this formulation. Because the input MCS list is simply a list of minimal cutsets, no BC set information used to generate the input MCS list is included for use in the main event tree (in fact, the input analysis case may contain events which contradict BC sets applied in the main analysis). Also, the input analysis case and the main analysis case may both produce success modules, leading to cutsets in the final MCS list containing two success modules. These success modules must be treated correctly in order to ensure a conservative estimate of the top event value.

Assuming that the BC sets from the input event tree have no effect on the main analysis case (either because they do not affect events present in the main analysis case or because they are repeated in the main analysis case), one could expect that the result of an analysis using direct linking will be the same as the result obtained from the linking via analysis results. The MCS lists from both analyses have to be the same. Considerations for correct use of this method, i.e., leading to the equivalence between direct linking and linking via analysis results, are described in more formal detail in Section IV.

III. METHOD DETAILS

In this section, we first describe the analysis method for linking of event trees via analysis results. We also detail the way in which BC sets and success modules are treated in RiskSpectrum calculations. Finally, we discuss advantages of this algorithm.

III.A. Algorithm

Here we introduce the method implemented in RiskSpectrum. The description does not necessarily follow the actual implementation. It rather helps to understand the expected results of this new type of analysis. First, let us recall the MCS list generation process. RiskSpectrum produces cutsets from the coherent part of the model. All generated cutsets are first validated by the not-logic. If a cutset does not fail the top gate of the complete model, including the not-logic, then it is discarded. After that, each validated cutset is checked on minimality. If removing an event from this cutset results in a valid cutset then it is not minimal and it is discarded. The final MCS list contains all cutsets remaining after validation and minimization.

The proposed method splits the analysis of linked event trees into two discrete parts, which are later joined to produce the final MCS list:

1. A minimal cutset list $J = j_1, j_2, \ldots, j_m$ of the input analysis case is generated for the linking event, by MCS analysis of the top event (consequence, sequence, gate) of the input analysis case. This MCS list is stored for later use.
2. A minimal cutset list $K = k_1, k_2, \ldots, k_n$ of the main ET is generated, by performing MCS analysis on only the main ET. The master fault tree for main ET is built without the initiating event logic, i.e., only from the event trees downstream of the initiating event. These k_i are termed *partial* cutsets. Denote the top gate of this master fault tree by g.
3. The final cutset list, L, is generated by expanding the cutset list K using the MCS list J. Each partial cutset k_i is replaced with a group of cutsets, by appending it to each of the cutsets j_i. This cutset list is a Cartesian product of J and K, containing $m \times n$ entries. Each entry in L is a cutset which fails *both* the input MCS analysis case, and the main analysis case.
4. The cutset list L then is validated against not-logic[5] in the main ET and minimized to remove repeated or non-minimal entries, which may have appeared in the combination process, and then quantified as usual.

As an example, if step (1) generates the two minimal cutsets J = {A, B} and {C} and step (2) generates the three minimal cutsets K = {P}, {Q} and {R}, step (3) would generate the cutset list with 2 × 3 = 6 entries {A, B, P}, {A, B, Q}, {A, B, R}, {C, P}, {C, Q} and {C, R}.

The description above simply assumes only one MCS as input "place" in the model, but there could be many ETs with many different initiating events, each with their own different MCS lists used as input. This is a straightforward generalization of the "one input MCS list" case. For cutset generation, sequences with different initiating events can be considered independent, even as different analysis cases. Only the not-logic validation and minimization need to consider all sequences together. The implementation in RiskSpectrum can handle such situations with multiple initiating events having MCS lists as inputs.

Clearly, the result of the main analysis case can serve as an input to another initiating event. Chaining analysis cases in this way does not present any algorithmic challenges. It only requires that input analysis cases are calculated before they are used in any main analysis case.

III.B. Boundary conditions

In RiskSpectrum PSA, a boundary condition set (BC set) may be applied to any node in an event tree. In general, the BC set applied to a node in an event tree is also applied to every child node in that tree – the BC set is *inherited* by child nodes. Inheritance of BC sets is also possible between directly-linked event trees, so that a BC set on an event tree node is also inherited by each node in the child event tree (analysis must performed with the RSAT setting "Inherit BC sets between event trees" to enable this behavior). Also, global BC sets may be defined, which are applied to the entire analysis case.

This BC set behavior is not preserved when using MCS lists as input. Any BC sets are applied, as usual, when calculating the MCS list which will be used as input, but once the MCS list has been calculated, the BC set information is not saved for further use.

Similarly, BC sets applied in the child event tree (on the initiating event of that tree, or any child node) are *not* applied to the input MCS list. Calculation of the MCS list for the main analysis case is performed (using all BC sets applied within the main analysis case) to produce partial cutset list K. Expansion of the MCS list K to L in step (3) is achieved by directly appending the k_i with the cutsets j_i; BC sets have no effect at this step. It is therefore possible that an entry in the final cutset list L contains seemingly contradictory information. If an input MCS contains a particular basic event, and the BC set on the initiating event sets this basic event to FALSE, the event will be treated as false in the event tree where the BC set is applied, but not in the input MCS list. Therefore, the final MCS list could still contain this basic event.

III.C. Success modules

The MCS list generated by an analysis case that is used as input to an initiating event can include success modules. This means that the result of a consequence or a sequence analysis for event trees that have analysis cases as initiating events could include two success modules in one and the same MCS. In this case, the algorithm automatically replaces the two success modules with one success module representing a 'merge' of the original success modules with probability equal to the product of the two modules probabilities. This treatment is conservative as long as each pair of merged success modules is independent. If this is not the case then the algorithm issues an error and analysis is aborted.

III.D. Advantages

In the general case, the link event may be used as an initiating event in many places in the model. Notice that with MCS as input, step (1) only needs to be performed once, regardless of the number of places the link event appears in the main analysis case. Note also that the process of combining the two MCS lists in step (3) is less computationally demanding than generating the input MCS list in step (1). By contrast, a directly joined analysis would require that the input logic be recalculated many times. Using MCS lists as input can therefore result in a significant improvement in calculation efficiency. Furthermore, if the model needs to be re-analyzed to include a change in the main analysis, the process may be re-started from step (2), making use of existing calculation results from step (1). This can allow a model to be easily modified and re-evaluated (for example, while the model is still being constructed) without having to repeat expensive calculations in the input MCS analysis case.

IV. NOT-LOGIC AND MINIMIZATION IN GENERAL CASE

The proposed algorithm can conservatively handle simple quantitative success treatment if success modules generated by the input and main analysis cases are independent. When there are dependencies between success modules which occur in one cutset then simple quantitative treatment in the consequence analysis could be used if we knew which success function events with which BC sets are included in each success module. Then we can simply add success function events from the main analysis case. In the following, we assume that there is no simple quantitative treatment of the consequence analysis case to illustrate the challenges posed by not-logic validation {Ref BK} and minimization. Situations described in this section are generally related to the use of not-logic when there are dependencies between the input analysis case and the main analysis case.

First, we consider a situation in which the proposed algorithm might generate non-valid cutsets.

Example 1. The master fault tree for the input analysis case is shown in Figure 3. The main MCS list has only one basic event, B. The MCS list from the input analysis case contains two cutsets {A} and {C}. When merged with the main analysis, we obtain the final minimal cutsets {A, B} and {C, B}. The first cutset is not valid in the input fault tree, because it violates the not-logic on B. Therefore, the result should only be the cutset {C, B}. The top value will be over-estimated due to the unnecessary inclusion of {A, B}.

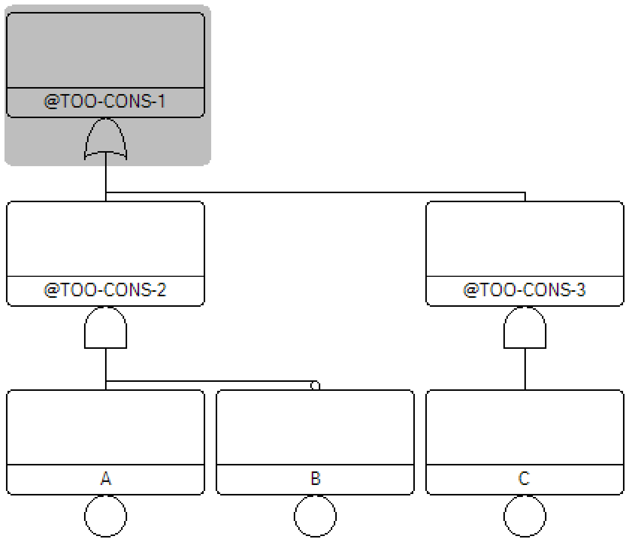


Figure 3. A master fault tree of the input analysis case leading to overly conservative results.

An initial attempt to rectify the problem in Example 1 would be to also store the master fault tree information that was used in generating the input MCS list, which would allow correct treatment of the not-logic in the input MCS list. We show here that such an approach can yield an improvement, but still miss a correct cutset.

Assume that the algorithm generates the MCS list for the input analysis case and it stores the master fault tree from this analysis. When it produces cutsets in the main analysis case, it discards all cutsets that do not fail the top gate of the master fault tree from the input analysis case. The following examples show that this approach might lead to fewer minimal cutsets than the direct linking.

Example 2. Consider the master fault tree for the input analysis case depicted in Figure 4. The main event tree MCS list contains the single basic event, C. The input MCS list contains only one cutset {A, B}. When merged with the main analysis, we obtain the cutset {A, B, C}, which is not valid in the input analysis case fault tree due to the not-logic. When retaining the MFT logic for the input MCS, this new cutset {A, B, C} could be checked to ensure that it satisfies the input MCS logic. The cutset would be tested, shown to not fail the input event tree, and be removed, resulting in an empty cutset list. However, although the incorrect cutset has been (correctly) removed, the correct result of the combined analysis is a cutset list containing the cutset {A, B, C, D}. This correct cutset is missed, because of premature minimization at the point where the two analyses are joined. Note that the correct result would be found if the analyses were directly joined in the conventional way. Note also that the model from this example would be treated correctly by the current method, because the validation step against the master fault tree from the input analysis case is missing.

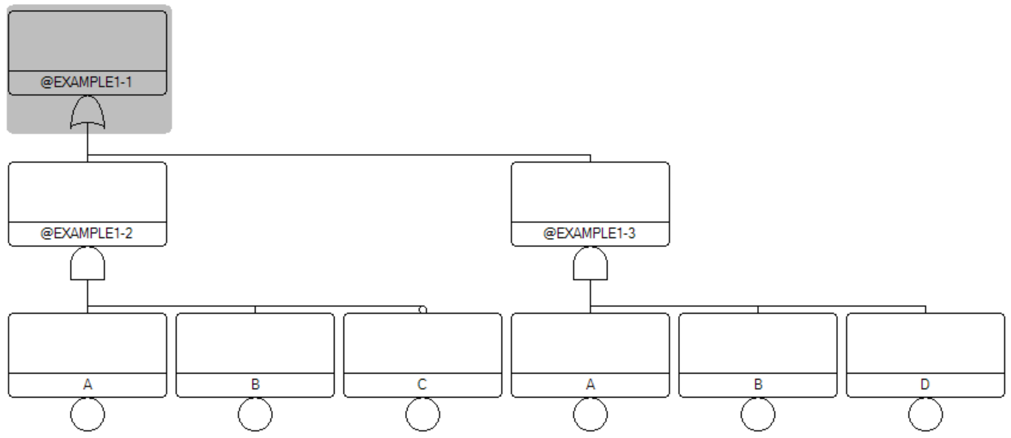


Figure 4. A master fault tree of the input analysis case illustrating minimization issues.

Example 3. Consider the master fault tree for the input analysis case depicted in Figure 5. The event tree of the main analysis case yields one basic event, C. The MCS list from the input analysis case contains only one cutset {B}, because {A} has been removed by the not-logic. Merged with the main analysis case, we obtain the cutset {B, C}, but the correct solution obtained by the direct linking also contains the cutset {A, C}.

Note that removing the cutset {A} in the input analysis case is fully in accordance with the intension of not-logic in RiskSpectrum[5]. Typically, such situations might occur in sequence analysis cases where the negated gate @FT2-3 it the top gate of a successful function event. Therefore, the cutset {A} will occur in the solution of a sequence where also this function event fails.

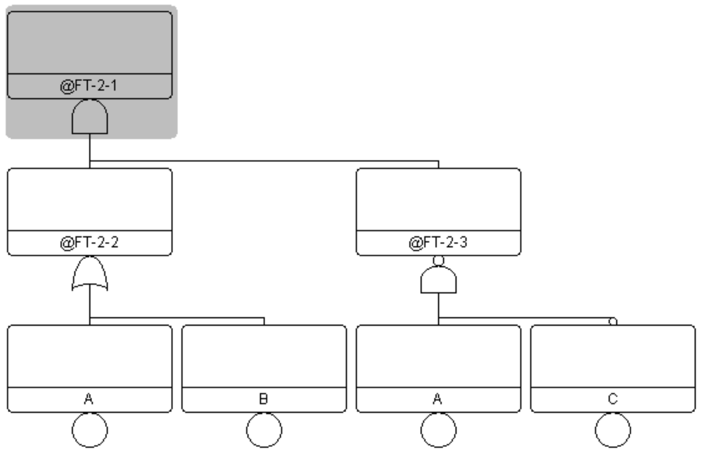


Figure 5. A master fault tree of the input analysis case illustrating not-logic validation issues.

One could discuss whether the outcome of all three examples is actually not a desired outcome, since the main analysis case models further accident progression which occurs temporally after the consequence. Should the event which has occurred in the main analysis case have an effect on removing cutsets from the input analysis case, when it only occurs after the link event? This is a complicated issue since there is no standard formalism for modelling temporal dependencies in fault/event trees.

V. AN EXTENDED ALGORITHM

Here we sketch an extension of the proposed algorithm that resolves the issues illustrated in Section IV. If the input analysis case is not coherent then we clearly need to store the information about the combinations excluded by the not-logic.

One possibility is to store the master fault tree for the input analysis case and use it each time an analysis with this analysis case in an initiating event is calculated. This remedies the over-conservatism, but the examples above show that we might still obtain results different from analysing one analysis case linked by consequences.

In Example 2, a cutset is missed because of minimization in the input analysis case part. Example 3 shows that applying not-logic to the results of the input analysis case leads to omitting a minimal cutset from the combined analysis (linked by consequences). A full remedy of both problems is to switch off minimization and the not-logic when generating cutset lists which are used as initiating event inputs. The non-minimized input cutset list can be guaranteed to include such cutsets as the prematurely-minimized cutset from Example 2. Skipping the not-logic check from the input analysis case ensures that all cutsets that can be extended to valid cutsets are kept.

This extension of the algorithm comes with a cost. Apart from tool design issues, switching off minimization and not-logic would lead to larger cutset lists which have to be stored on disk. All analyses using this cutset list as input would have to also handle non-minimal cutsets from the input consequence analysis, but this corresponds to the way in which the calculation for analyses linked by consequences works today.

Alternatively, one could switch off only minimization. Such algorithm modification would miss the cutset from the example in Figure 5, but one could argue that splitting analysis cases reflects also a temporal dependency. The input consequence models accident progression which happens first; events from the downstream event tree(s) then occur afterward. Therefore, if the event C did not fail in the input consequence analysis case then the cutset {A} has been removed correctly and cannot re-occur because C might fail in the subsequent analysis.

VI. MODELLING GUIDELINES

From the method description in Section III and the discussion in Section IV, it follows that the method described here and implemented in RiskSpectrum is correct, i.e. it produces the same results as direct linking, under the following assumptions:

- BC sets from the input analysis case should not affect the main analysis case. This can be the case if BC sets from the input analysis case assign values only to events which do not occur in the main analysis case. Another possibility is that the main analysis case repeats all assignments from the input analysis case in the initiating event.
- Success modules generated by the input analysis case and the main analysis case are independent. This condition applies only to analyses with Logical and Simple Quantitative success treatment.
- The input analysis case is exactly represented by the generated MCS list[4], which means that the master fault tree is coherent.
- If the previous condition is not satisfied then the correctness could be retained by ensuring independence of the input analysis case and the main analysis case.

The easiest modelling constraint guaranteeing correctness of the analysis is the independence of the input analysis case and the main analysis case. By this, we avoid issues with the not-logic and minimization and automatically maintain independence of success modules. Also, BC sets in the input analysis case have no effect on the main analysis case (as long as these BC sets affect the input analysis case).

If it is not possible to ensure independence of the input analysis case and the main analysis case then one has to verify the following properties:

- BC sets from the input analysis case which affect the main analysis case are repeated in the initiating event.
- Effects described in Example 1 and Example 3 in Section IV (caused by the not-logic and minimization in the input analysis case) are acceptable.

VII. CONCLUSIONS

The proposed method together with the algorithm offers new possibilities to optimize computational resources. In some cases, this approach allows analysis of a previously infeasible model. The limitations of this approach are clearly described in this paper, leading to guidelines for analysts when building and calculating their models. We also show how these limitations can be lifted by extensions of the algorithm. With careful consideration of dependencies between the input analysis case and the main analysis case and the application of not-logic in both, the use of MCS lists as input in RiskSpectrum PSA can be a powerful tool to significantly improve the construction and calculation efficiency of large models.

REFERENCES

1. H. KUMAMOTO and E. J. HENLEY, *Probabilistic risk assessment and management for engineers and scientists*. IEEE Press, New York, USA (1996).
2. NUREG/CR-2300. *A guide to performance of probabilistic risk assessments for nuclear power plants*. US NRC (1983).
3. O. NUSBAUMER and A. RAUZY. “Fault Tree Linking versus Event Tree Linking Approaches: a Reasoned Comparison.” *Risk and Reliability*, 227(3), pp. 315-326, 2013.
4. A. RAUZY, "Mathematical foundations of minimal cutsets", *IEEE Transactions on Reliability*, Vol. 50, No.4, (2001).
5. O. BÄCKSTRÖM and P. KRCAL. “A Treatment of Not Logic in Fault Tree and Event Tree Analysis.” In proc. of *PSAM 11*. Helsinki, Finland (2012).