Quantification of MCS with BDD, Accuracy and Inclusion of Success in the Calculation – the RiskSpectrum MCS BDD Algorithm

Wei Wang[a], Ola Bäckström[a], and Pavel Krcal[ab]
[a] Lloyd's Register Consulting, Stockholm, Sweden
[b] Uppsala University, Uppsala, Sweden

Abstract: A quantification of a PSA can be performed through different techniques, of which the Minimal Cut Set (MCS) generation technique and Binary Decision Diagrams (BDD) are the most well known. There is only one advantage with the MCS approach compared to the BDD approach - calculation time, or rather, the capability to always solve the problem. In most cases the MCS approach is fully sufficient. But as the number of high probability events increases, e.g. due to seismic risk assessments, more accurate methods may be necessary. In some applications, a relevant numerical treatment of success in event trees may also be required to avoid overly conservative results.

We discuss the quantification algorithm in RiskSpectrum MCS BDD, especially with regard to success in event trees. A BDD for both the failure and success parts of a sequence can be generated separately - and thereafter, the BDD structures can be combined. Under some conditions, this calculation will yield exactly the same result as if a complete BDD for both the failure and success parts was generated. Properties of the algorithm are also demonstrated on several examples including a large size PSA.

Keywords: PSA, Binary Decision Diagrams, Minimal Cut Set List Quantification

1. INTRODUCTION

The PSA models are of increasing size and complexity, and they are used for an increasing number of applications. In most cases the standard calculation methods (e.g. minimal cut set upper bound) are fully sufficient. But as the number of high probability events is increasing, e.g. due to seismic risk assessments, more accurate methods may be necessary. In some of the applications, a relevant numerical treatment of success in event trees may also be required to avoid overly conservative results.

Bryant's Binary Decision Diagrams [9] present an efficient data structure for encoding of Boolean functions. Their main industrial applications lie in the area of computer aided design of electronic circuits and formal verification. Since their introduction to fault tree analysis [5,6] they stand as a challenge to traditional cut set based approaches. The BDD approach would yield the exact result unaffected by cutoff truncation, first order approximation and imperfect negation treatment. In spite of a large body of work, see e.g. [7,10,11], it has not been possible to solve current PSA models with BDD because of their size – it is therefore not a realistic approach today.

A less ambitious approach uses a variant of BDDs, ZBDDs [4,8], for a partial solution of the problem, for quantification of a coherent structure (e.g., MCS lists produced by traditional cut set generation methods). ZBDDs serve as a compact representation of the cut set list in question, but the quantification procedure results in the same value as rare event approximation.

The calculation algorithm in RiskSpectrum MCS BDD was briefly described at PSAM9 [1], and this paper will discuss the quantification in more detail, especially with regard to success in event trees. We apply BDD technique to quantify cut set lists with two aims:

- Avoid conservative results of the first order approximation for important events
- Allow exact negation treatment for success in sequences

A characteristic of a BDD is the possibility to calculate a structure given the event combinations that have, or have not, happened. This characteristic is discussed and its use within the algorithm is presented. A BDD for both the failure and success parts of a sequence can be generated separately - and thereafter, the BDD structures can be combined. Under some conditions, this calculation will yield exactly the same result as if a complete BDD for both failure and success part was generated.

The structure of the paper is as follows. First, we briefly introduce BDD background. Section 3 presents the MCS BDD algorithm together with some heuristics applied during BDD generation. Then we describe properties of the algorithm and argue for its correctness. Finally, some of the properties are demonstrated in Section 5 on example problems (including a full scale PSA study).

2. BACKGROUND

The classic MCS quantification methods like minimal cut set upper bound or 1st, 2nd or 3rd approximation are effective and efficient, but they also have some obvious drawbacks, such as overestimation of the top results, especially with some high probability events. Another problem is the treatment of success in sequence and consequence analysis.

The MCS BDD algorithm has therefore been developed as a part of the RiskSpectrum software package, which gives a better approach to quantify top results from MCS list.

The MCS BDD algorithm is derived from Shannon non-intersect decomposition,

$$f = xf_1 + \bar{x}f_0,$$

where x is one of decision variables. The functions f_1 and f_0 are Boolean functions evaluated at $x=1$ and $x=0$ respectively.

Minimal cut sets can be converted to a group of mutually exclusive variables by non-intersect operation, then the top result is obtained by the summation of the probabilities of these variables.

Example 1: MCS list {A, BC, D}.

By no-intersect decomposition, the Boolean expression of the MCS list can be converted as follows:

$$Top = A + BC + D = A + \bar{A}D + \bar{A}\bar{D}BC$$

The top event probability can be calculated directly:

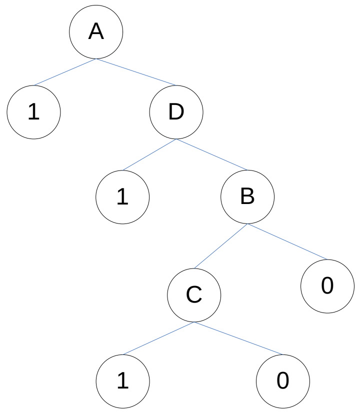
$$P(Top) = P(A) + P(\bar{A}D) + P(\bar{A}\bar{D}BC)$$

BDD is a natural non-intersect structure, in which all the paths from the root to the terminal "1" (failure state) nodes make the whole set of mutually exclusive minimal cut set variables. If we succeed to build the BDD structure for the target MCS list, it means that we can get a non-intersect form of the MCS list, from which the exact top result could be retrieved easily. The disjoint paths through the BDD structure in Figure 1 are: A, $\bar{A}D$, $\bar{A}\bar{D}BC$.

3. GENERATING BDD FROM A MCS LIST

The MCS BDD algorithm uses a pivotal decomposition method to construct the BDD structure from the MCS list. By continually picking up pivotal elements (decision variables) from the MCS list, both failure and success branches are being further developed. When all the branches have reached "1" (failure state) or "0" (success state), the BDD structure is generated completely.

Figure 1. BDD structure in Example 1



The key process of building the BDD structure is to select the pivotal element from the remaining MCS list and append it to the branch of the current pivotal node under construction. A backtracking has to be performed before selecting the next pivotal element in order to get all the elements along the backward path; the MCS list can then be minimized against the states of these pivotal elements.

The purpose of minimization is to reduce the size of the cut set list that has to be maintained during run-time. However, instead of doing a complete cut set minimization for each pivotal element, which is actually time consuming, we do a limited minimization to simply remove those events and cut sets with fixed states. If any cut set that contains the current pivotal element has got a fixed state after minimization or there is no cut set remaining in the current MCS list, it indicates that the branch has reached a terminal state ("1" or "0"); otherwise a new pivotal element shall be picked up from those events in the current MCS list whose states have not been decided yet, and appended to the target branch of the current pivotal node.

Take Example 1 to illuminate the BDD building process. First we choose A as a pivotal element, its failure branch leads directly to terminal state "1" after minimization, while its success branch is still to be developed. Then we need to select the next pivotal element from the remaining MCS list (with A succeeded) to extend the success branch of A; followed by selecting D as the new pivotal element, and the state of the failure branch can be decided to terminal state "1" by a propagation of the remaining MCS list (with D failed, A succeeded), while its success branch is still left to be further developed. In the same way, we can add another two pivotal events B and C to the structure. When all paths are terminated by "1" or "0", the building process has come to an end.

3.1 Function Event Success in a MCS List

First, we describe the way in which we store information about success in minimal cut set lists. We assume that the cut set list is obtained from analyzing one or more sequences of one or more linked event trees. All function events that succeed along a sequence are analyzed separately in the following way. We generate a minimal cut set list where each cut set fails at least one function event which succeeded. This means that these cut sets cannot be a part of the sequence solution. We label this minimal cut set list with the set of successful function events which were used to generate it and call it, together with the label, a *success module*.

We note that a success module uniquely identifies a sequence. This is obvious from the fact that each sequence for one or more linked event trees is uniquely defined by enumerating function events that succeed (or, that fail).

The algorithm for cut set generation will add the corresponding success module to each cut set generated from a sequence with successful function events.

3.2 MCS Cutoff Term

In order to reduce the amount of computation to a practical scale, in the MCS BDD algorithm it is possible to define a threshold on the cut set list so that those parts of cut sets that are below the threshold shall be evaluated with ordinary MCS quantification. The algorithm then would focus on the other part of the MCS list to quantify the result exactly, because this part will make the greatest contribution to the accuracy of the top result.

The results of the two parts shall be summed up with the minimum cut set upper bound method to ensure a conservative top result. If the threshold is set to 0%, the algorithm will build the BDD structure based on the whole MCS list.

3.2 MCS Grouping

Frequency events, usually as initiating events, and success modules are considered naturally mutually exclusive. It is possible to split the MCS list into different groups which are mutually exclusive to each other. This can greatly reduce the scale of the problem: the top result then can be calculated by a direct accumulation of the results of each separate group.

In the MCS BDD algorithm we first group the MCS list with different frequency events, and for each such group, we then split the cut sets into sub-groups with different success modules. The input to the algorithm for BDD generation is a cut set list in which all cut sets contain the same frequency event and the same success module. We assume that the frequency event and the success module are already removed from the cut sets.

In the MCS BDD algorithm, instead of building a complete BDD for the whole sequence, we build one main BDD structure for the sequence MCS list without success modules, as well as another BDD for the success module itself. In a later stage, the success module BDD is appended and merged to the main BDD to get a full representation of the logic of the sequence.

3.3 Pivotal element ordering

The BDD is generated from the MCS list by successively adding events from the MCS list. The order of pivotal elements makes a great impact on the size of the BDD structure. The problem of deciding whether there is an ordering of pivotal elements resulting in a BDD of a given size is NP-complete [3]. Therefore, it is unlikely that a polynomial algorithm for finding the best order of pivotal elements in order to obtain the optimal BDD structure exists. In the MCS BDD algorithm, instead of using a predefined fixed order of these events, we use a dynamic method to select new pivotal elements from the remaining MCS list with some basic principles, e.g. number of event occurrences, cut set order, which allow a more concise BDD structure to be obtained.

3.4 BDD modules

The MCS-BDD algorithm will also try to group events together in modules to simplify the BDD structure. The modules are detected and created during runtime based on the actual MCS list. Each time when selecting a new pivotal element, it will check whether it is possible to find any modules. Modules always have a higher priority when selecting a pivotal element.

3.5 Approximate treatment

When a pivotal element is chosen the MCS list has to be treated and minimized based on that failure or success. There are basically two different ways of generating the BDD structure in the MCS BDD algorithm:

- Exact treatment
- Approximate treatment

The exact treatment represents the normal way of building a BDD structure from an MCS list. This means that an event A is considered to be either failed or successful, and then apply that on the MCS list.

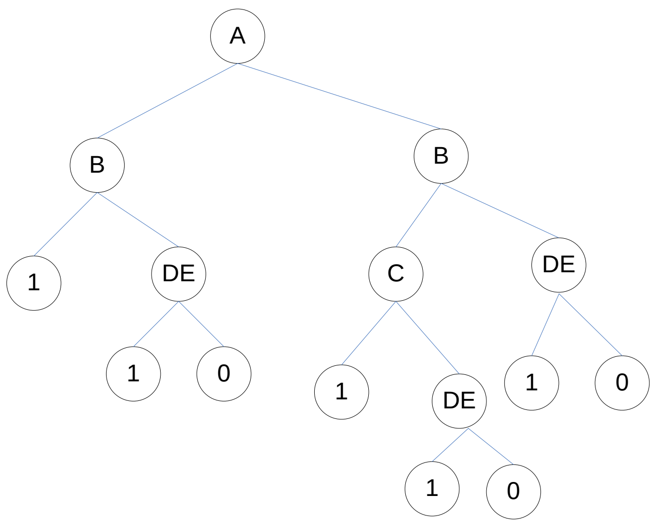
The approximate treatment means that when the MCS is analyzed and an event is failed then only MCSs where the event is included are considered further (in principle, the same way as ZBBDs [3,4] are built). The reason for this is that all the MCSs not containing the event will be a part of the success branch. Hence, with a slight modification of the quantification of the structure it is possible to have a very good approximation of the solution that normally is very efficient in reducing the structure.

Take Example 2 to illuminate the approximate treatment:

Example 2: MCS list {AB, BC, DE}

The complete BDD structure from exact treatment is shown in Figure 2 (the pivotal elements in the order of A, B, C, DE as a module).

Figure 2. BDD of Example 2

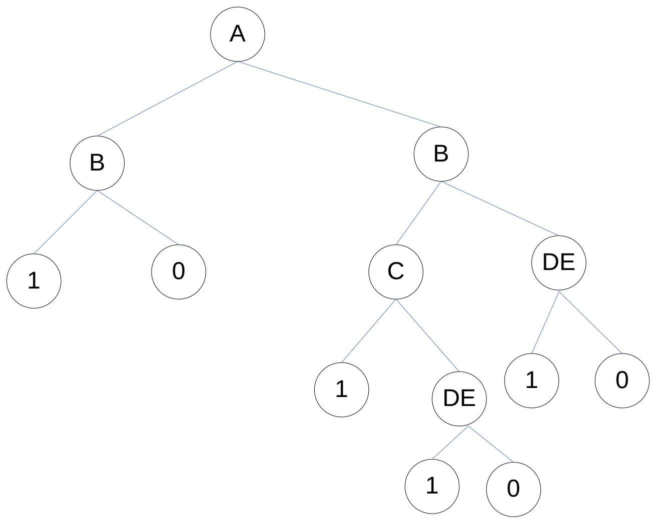


Because the pivotal element A is only included in the cut set {AB}, the remaining two cut sets should be in the success branch of A. With approximate treatment, the above BDD structure can be simplified as shown in Figure 3.

The new BDD structure still keeps the information of all the minimal cut sets, and the failed branch of A is simplified. As can be understood from the above the quantification of the approximate structure must be different. The quantification of the BDD structure will in this case be:

P(Top) = P(A) * [P(MCS including A) + P(MCS not including A) - P(MCS including A) * P(MCS not including A)] + (1 - P(A)) * P(MCS not including A)

Figure 3. BDD of Example 2 with approximate treatment



This can be simplified as:
P(Top) = P(A) * P(MCS including A) + (1- P(A) * P(MCS including A)) * P(MCS not including A).

When this approximate treatment is used the following conditions apply:

- The approximate treatment can never be used when the event is negated in the cut set list.
- The approximate treatment is conservative for coherent structures, since dependencies between MCS are not treated exact. The conservative error is however very small if the pivotal element has low probability.
- When the two parts of the cut set list both contain events that occur in the success module, exact treatment has to be used or we could switch to other more conservative quantification methods, e.g. ZBDD.

The use of the approximate treatment is optional and is actuated by two different triggers:

- Fussel-Vesely importance above a specified level
- Unavailability above a specified level

Fussel-Vesely importance is defined as a measure of the contribution a basic event makes to the top event probability or frequency. It is calculated using the formula:

$$FV_E = \frac{\sum Q_{\bar{A}_E}}{Q_{Top}},$$

where each $Q_{\bar{A}_E}$ is the probability value of the cut set $\bar{A}_E$ containing the element E.

4. CORRECTNESS

First, note that the MCS list (without success modules) and success modules are both generated from coherent fault trees. Fault tree models in RiskSpectrum can contain negated gates. These gates are not interpreted according to standard rules of Boolean algebra, expressing that a failure occurs if some subsystems succeed. Not-logic is treated as a syntactic shortcut for cut set removal during the fault tree analysis [2].

4.1 Correctness of Quantification

We build a BDD from the minimal cut set list in the standard way. We use events from success modules as pivot elements first before all other events. These events are treated exactly. When we have processed all of these basic events then we can use both exact and approximate way of building BDD for the remaining basic events.

Finally, we merge this structure with the success module BDD. Because we have built nodes for basic events from success modules in the exact way, we can do this in a way that allows for exact quantification of success modules.

Claim 1. The MCS BDD algorithm quantifies success modules exactly.

If all nodes in the BDD are treated exactly then the quantification procedure returns the exact probability of the MCS list, including exact quantification of the success modules. This means that the whole MCS BDD algorithm handles also the non-coherent part, as if cut sets with success modules were expanded into prime implicants.

The approximate treatment might lead to overestimating the actual probability of the failure part of the BDD. Both BDD substructures of a node which is treated approximately represent coherent structures (note that success modules and these substructures are disjoint). Because of this, the term "- P(MCS including A) * P(MCS not including A) " in the quantification formula for approximate treatment will never lead to an under-approximation. On the other hand, it will give a more precise result than the standard ZBDD quantification.

Claim 2. The MCS BDD algorithm is always conservative and it is exact if all nodes are treated exactly.

The conservatism of the top result will be determined by the way we select basic events for exact or approximate treatment. Because we select most important events for exact treatment, the over-estimation will be negligible in most cases.

4.1 Effects of Cutoff

The whole event/fault tree analysis algorithm applies cutoffs to keep the problem in a manageable size. There are three types of cutoff:

- Cutoff during cut set generation of the failure part
- Cutoff during generation of success modules
- Cutoff before building the MCS BDD structure

The first cutoff type has the same effect as for the standard MCS based event/fault tree analysis. It decreases the top value probability and potentially makes the result non-conservative. It is up to the PSA analyst to set this cutoff correctly so that the cutoff error remains acceptably small (in the best case, negligible with respect to the calculated top value).

The second type of cutoff is always conservative. It removes cut sets from success modules. By doing so, the actual probability of the success module in question grows. It might also result in removing all cut sets containing a particular basic event. Then this event might be treated approximately in the MCS BDD algorithm. But, this leads again only to an over-approximation of the final probability. This means that we can use greater cutoff values for success modules without the risk of underestimating the top probability.

The last cutoff type is also conservative. It excludes some cut sets from the MCS BDD treatment and quantifies them by the minimal cut set upper bound method. This results in an over-approximation and the total probability for these cut sets is always greater then (or, equal to if all BDD nodes are treated approximately) the result of the MCS BDD algorithm without any cutoff would be.

This brings us to the following conclusion.

Claim 3. If the analysis algorithm does not apply any cutoff and if the MCS BDD generation process always uses the exact treatment of basic events then it returns the exact top value by the means of MCS BDD.

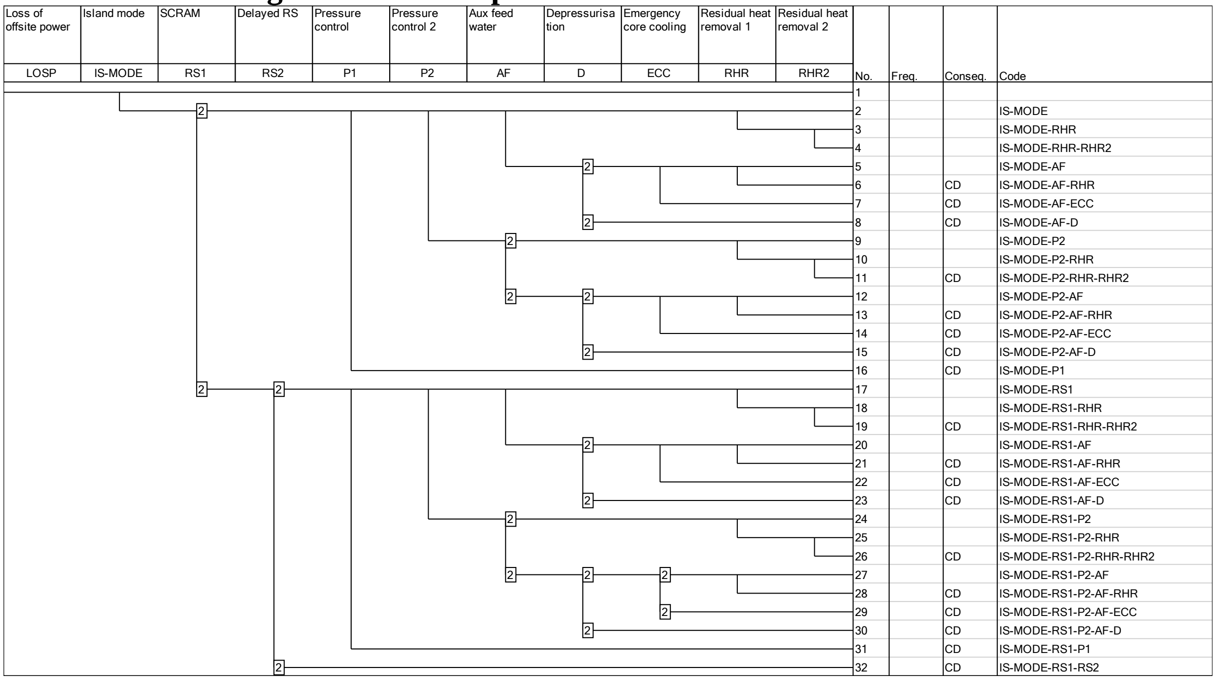
4.4. Importance with MCS BDD

The importance factor that suffers most from over-approximations in the standard MCS list quantification is the Risk Increase Factor (RIF). The main problem is that simply setting the probability of a basic event to one might leave the MCS list with many non-minimal cut sets. The minimal cut set upper bound algorithm counts in also all of these non-minimal cut sets. This results in overly high RIF values. If the MCS BDD algorithm treats the basic event for which we calculate RIF exactly then there will be no over-approximation coming from what would correspond to non-minimal cut sets in the regular cut set list quantification. If we use approximate treatment then we always get a result which is at least as good as from the standard MCS list quantification with minimal cut set upper bound.

5. EXAMPLES

The use of the MCS BDD algorithm is demonstrated in the following examples. The first example uses a full scale PSA model, which is slightly modified so that it is not producing results that are applicable for any plant. The problem, size and numbers, are still representative of the type of problem we want to demonstrate. Figure 4 shows the analyzed event tree.

Figure 4. An example event tree from a full scale PSA



No.	Freq.	Conseq.	Code
1			
2			IS-MODE
3			IS-MODE-RHR
4			IS-MODE-RHR-RHR2
5			IS-MODE-AF
6		CD	IS-MODE-AF-RHR
7		CD	IS-MODE-AF-ECC
8		CD	IS-MODE-AF-D
9			IS-MODE-P2
10			IS-MODE-P2-RHR
11		CD	IS-MODE-P2-RHR-RHR2
12			IS-MODE-P2-AF
13		CD	IS-MODE-P2-AF-RHR
14		CD	IS-MODE-P2-AF-ECC
15		CD	IS-MODE-P2-AF-D
16		CD	IS-MODE-P1
17			IS-MODE-RS1
18			IS-MODE-RS1-RHR
19		CD	IS-MODE-RS1-RHR-RHR2
20			IS-MODE-RS1-AF
21		CD	IS-MODE-RS1-AF-RHR
22		CD	IS-MODE-RS1-AF-ECC
23		CD	IS-MODE-RS1-AF-D
24			IS-MODE-RS1-P2
25			IS-MODE-RS1-P2-RHR
26		CD	IS-MODE-RS1-P2-RHR-RHR2
27			IS-MODE-RS1-P2-AF
28		CD	IS-MODE-RS1-P2-AF-RHR
29		CD	IS-MODE-RS1-P2-AF-ECC
30		CD	IS-MODE-RS1-P2-AF-D
31		CD	IS-MODE-RS1-P1
32		CD	IS-MODE-RS1-RS2

The initiating event is loss of offsite power and its frequency is 0.204/a. This is a PSA level 1, using large fault trees and small event trees. Since it is a PSA level 1, the probabilities for the top events (function events) in the event tree are typically low.

In RiskSpectrum PSA, there are several quantification methods that may be used. In this example, we are going to compare Logical ET Success, Logical and Simple Quantitative and MCS BDD (including success modules).

In short,

- Logical ET success means to logically ensure that the MCS that are generated cannot be in two different sequences. There is no consideration of the success path.
- The term *logical and simple quantitative* means, in addition to the above, that the successes are quantified as a FT analysis case ($P_{failure}$), and then the success probability is 1-$P_{failure}$.
- MCS BDD including success modules generates the MCS using the logical and simple quantitative method, but the success modules are appended to the BDDs that are generated for the failed part.

The example is run to illustrate the results from the different methods and to demonstrate that the MCS BDD method is scalable and applicable to real size problems. The number of MCS generated for the sequences spans from a few MCSs up to around 100 000 MCSs, including around 4000 events.

The example is, for each sequence, using 99,95% of the top frequency as basis for the BDD, and exact BDD treatment for events that have a FV larger than 1E-2 or a probability greater than 1E-2. The results are generated based on an MCS cut off defined at 1E-12. The core damage frequency is approximately 1,3E-5. This means that individual sequences close to cutoff will not be correct – but they will on the other hand have no impact on the overall results.

The comparison of individual sequences is shown in Table 1.

Table 1. A comparison of three different quantification algorithms

	MCS BDD	Log Simp Q	Logical ET Success
LOSP:0001	1,51E-01	1,51E-01	2,04E-01
LOSP:0002	5,13E-02	5,13E-02	5,31E-02
LOSP:0003	1,24E-06	1,26E-06	1,45E-06
LOSP:0004	1,33E-07	1,35E-07	1,40E-07
LOSP:0005	1,03E-05	1,06E-05	1,10E-05
LOSP:0006	4,18E-09	4,28E-09	4,44E-09
LOSP:0007	8,25E-06	1,11E-05	1,15E-05
LOSP:0008	2,12E-06	2,14E-06	2,22E-06
LOSP:0009	1,77E-03	1,83E-03	1,84E-03
LOSP:0010	3,54E-08	3,63E-08	4,05E-08
LOSP:0011	3,71E-09	3,78E-09	3,79E-09
LOSP:0012	4,99E-06	5,56E-06	5,59E-06
LOSP:0013	5,17E-09	5,50E-09	5,52E-09
LOSP:0014	5,81E-07	8,04E-07	8,06E-07
LOSP:0015	1,46E-07	1,52E-07	1,52E-07
LOSP:0016	5,49E-10	5,69E-10	5,70E-10
LOSP:0017	1,13E-04	1,15E-04	1,78E-04
LOSP:0018	4,86E-09	5,00E-09	8,70E-09
LOSP:0019	5,06E-10	5,07E-10	7,89E-10
LOSP:0020	5,85E-05	7,17E-05	7,42E-05
LOSP:0021	6,07E-08	6,36E-08	6,57E-08

LOSP:0022	2,36E-08	3,75E-08	3,88E-08
LOSP:0023	1,29E-07	1,42E-07	1,47E-07
LOSP:0024	3,58E-06	3,67E-06	5,55E-06
LOSP:0025	8,64E-11	8,75E-11	1,60E-10
LOSP:0026			
LOSP:0027	2,11E-06	2,55E-06	2,55E-06
LOSP:0028	6,10E-11	6,18E-11	6,19E-11
LOSP:0029	1,90E-09	2,34E-09	2,34E-09
LOSP:0030	1,45E-09	1,73E-09	1,73E-09
LOSP:0031			
LOSP:0032	1,26E-06	1,36E-06	1,36E-06
	2,04E-01	2,04E-01	2,59E-01

It can be seen from the table above, which presents all sequences, that both MCS BDD and logical and simple quantitative methods produce results that sum up to the initiating event frequency. It can also be noticed that MCS BDD always produces lower results than logical and simple quantitative, which in turn produces lower results than logical ET success.

The requirement to produce results that sum up to the initiating event frequency is that success treatment is included in the evaluation. There are some sequences that differ significantly between MCS BDD and logical and simple quantitative. The results of the MCS BDD are always lower (or equal), which is expected since the quantification of the success module is conservative in the simple quantitative approach.

If an analysis is set up to quantify the results for a group of sequences, the results can simply be summed for the MCS BDD and the simple quantitative results (or to define a consequence analysis). For logical ET success, the situation is different, since success is not included and hence some MCSs may be the same, or non-minimal between sequences. Therefore, the logical ET success result has to be calculated as a consequence analysis case.

The results for the CD with the different settings are:

- Logical ET success: 1,31E-5
- Logical and simple quantitative: 1,58E-5
- MCS BDD: 1,26E-5

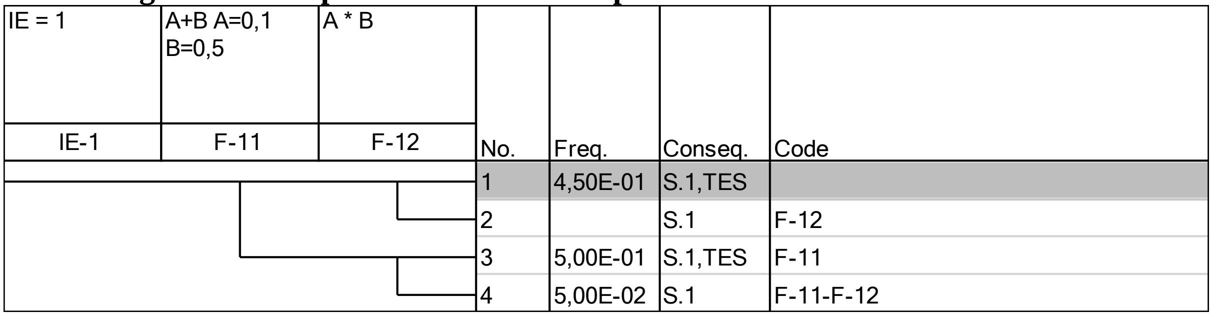
This example demonstrates that:

- The simple quantitative approach produces slightly conservative results, but it does give a reasonable estimate of individual sequence results.
- The logical ET success will produce conservative estimates for sequences in which success probability has a significant impact, and it does produce good approximations for normal cases where core damage is studied.
- The MCS BDD approach produces very good estimates of sequence results, including successes and it does also produce results that are accurate for a group of sequences.

The advantage with the MCS BDD is both the accuracy in the quantification of high probability events and the calculation of success. The other example illustrates the impact of success treatment and it is compared to the logical and simple quantitative and logical ET success approaches.

The example is illustrated in Figure 5, together with the correct results. Sequence 1 is the combination -(A+B), Sequence 2 will not produce any results. Sequence 3 is represented by (A-B) and (B-A) and Sequence 4 is (AB).

Figure 5. A simple event tree with dependencies between function events



No.	Freq.	Conseq.	Code
1	4,50E-01	S.1,TES	
2		S.1	F-12
3	5,00E-01	S.1,TES	F-11
4	5,00E-02	S.1	F-11-F-12

Table 2 presents the results for the different quantification methods:

Table 2. Comparison results for the simple event tree

	Log ET	Log Simple Quant	MCS BDD
Seq1	1	0,45	0,45
Seq2	0	0	0
Seq3	0,55	0,5249	0,5
Seq4	0,05	0,05	0,05

As can be seen, the logical and simple quantitative is a good approximation, but when there are dependencies between the failed part and the successful part of the MCS the results are overestimated. This is illustrated by Sequence 3, where the logical and simple quantitative produce following MCSs:
A –F12
B –F12

F12 is represented by the MCS
AB

Hence, the probability P(-F12) is calculated as $1\text{-}P_{F12} = 1 – P_A P_B = 0{,}95$

The problem is that there are dependencies between the failed part and the success part. For example, the MCS A-(AB) is always conservative, since this should be reduced to A(-B). The success module should be quantified depending on each MCS, but this would be very time consuming.

The MCS BDD, where the BDD for the success is appended/merged with the failed structure will be considered. In the MCS BDD the correct results will therefore be retrieved.

6. CONCLUSIONS

The MCS BDD algorithm presents a new way to quantify MCS lists in RiskSpectrum. Its main advantages are improved accuracy over the first or third order approximation and exact quantification of the part of the MCS list which represents event tree success. Especially, the new treatment of success enables a more accurate quantification in presence of function events with high probability. Heuristics implemented in the algorithm, e.g., combination of exact and approximate treatment of BDD nodes, make the algorithm scalable while always giving a conservative result. On the other hand, the algorithm has capability of returning the exact top event probability/frequency. If there are no cutoffs used and all BDD nodes are treated exactly then the new method returns the same result as a complete BDD algorithm.

References

[1] O. Bäckström and D. Ying, "*A presentation of the MCS BDD algorithm in the RiskSpectrum Software Package*", In Proc. of PSAM9, Hong Kong, 2008.

[2] O. Bäckström and P. Krcal, "*A Treatment of Not Logic in Fault Tree and Event Tree Analysis*", In Proc. of PSAM11, Helsinki, 2012.

[3] B. Bollig and I. Wegener, "*Improving the Variable Ordering of OBDDs is NP-Complete*", IEEE Trans. on Computers, Vol. 45(9), pp. 993-1002, 1996.

[4] S.-I. Minato, "*Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems*", In Proc. of DAC'93, 1993.

[5] A. Rauzy, "*New Algorithm for Fault Tree Analysis*", Reliability Engineering and System Safety, Vol. 40, pp. 203-211, 1993.

[6] O. Coudert and J.-C. Madre, "*Fault Tree Analysis: 10^{20} Prime Implicants and Beyond*", In Proc. of Annual Rel. and Maint. Symp. 1993, Atlanta, 1993.

[7] A. Rauzy, "*Binary Decision Diagrams for Reliability Studies*", Handbook of Performability Engineering, pp. 381-396, 2008. ISBN 978-1-84800-130-5.

[8] W. S. Jung et al., "*A Fast BDD Algorithm for Large Coherent Fault Trees Analysis*", Reliability Engineering and System Safety, Vol. 83, pp. 369-374, 2004.

[9] R. Bryant. "*Graph Based Algorithms for Boolean Function Manipulation*", IEEE Transactions on Computers, Vol. 35(8), pp. 677-691, 1986.

[10] O. Nusbaumer, "*Analytical Solutions of Linked Fault Tree Probabilistic Risk Assessments using Binary Decision Diagrams with Emphasis on Nuclear Safety Applications*", PhD Thesis, ETH No. 17286, 2007.

[11] R. Remenyte and J. D. Andrews, *"Qualitative analysis of complex modularized fault trees using binary decision diagrams"*. Journal of Risk and Reliability, Vol. 220 (1), pp. 45-53, 2006.