The Emperor’s New Truncation

A Modern Fairy Tale
Once upon a time ...

Many years ago there lived an Emperor who was so exceedingly fond of fine new clothes that he spent vast sums of money on dress. To him clothes meant more than anything else in the world. He took no interest in his army, nor did he care to go to the theatre, or to drive about in his state coach, unless it was to display his new clothes. He had different robes for every single hour of the day.
We all know how the story ends. The emperor, bare-ass naked, parades himself in front of his subjects. Only the voice of a little child declares that the emperor has no clothes.

And so it is now, with BDD. Everyone is so fond of new ideas, and they hope for new breakthroughs in PSA quantification that some embrace every proposed innovation with “ooooos” and “ahhhhs”.

We, however, are like the child who asks himself, “But does he have anything on?”

We all like fairy tales. But in this time of global need for renewable energy, it is up to each of us to insure proper and correct methods for nuclear PSA, not invisible cloth.

So forgive us if this modern fairy tale we will now tell causes us to dress in less than splendor, but assuredly covering our private parts.
being too pedantic, the notions of truth tables, BDD, minterms, and minimal cut sets. For those with a more academic bent, please refer to the seminal paper, “Mathematical Foundations of Minimal Cut Sets” [Rauzy2000].

A truth table is a tabular way of representing a Boolean function. If we have a Boolean function \( F(a,b) = a \lor b \), where “\( \lor \)" means OR, the truth table for the function would be:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a \lor b</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Each row of the truth table is mutually exclusive, or disjoint.
called a directed acyclic graph, DAG) of a truth table, usually, we hope, a more compact representation, built top-down. So given the same function, \( F(a,b) = a \lor b \), we might have this BDD structure:

![BDD Structure](image)

Think of the solid lines as assigning "true" to the variable from which they emanate, the dotted lines as assigning "false", and the 1's and 0's indicating if the function is satisfied or not by the assignments. Each path in the graph is disjoint, like the rows of a truth table. We can read the graph as indicating that if \( a \) and \( b \) are true, then the function is satisfied, if \( a \) is true, but \( b \) false, the function is satisfied, and so
graph with a function called "if-then-else", usually written \( \text{ite}(\text{if}, \text{then}, \text{else}) \), which means if \( a \) is true proceed down the left branch, if false, proceed down the right branch. So given the graph from the preceding page:

Building the BDD top-down, we can represent it with this series of functions:

\[
\begin{align*}
\text{BDD} &= \text{bdd}_1 \\
\text{bdd}_1 &= \text{ite}(a, \text{bdd}_2, \text{bdd}_3) \\
\text{bdd}_2 &= \text{ite}(b, 1, 1) \\
\text{bdd}_3 &= \text{ite}(b, 1, 0)
\end{align*}
\]
Boolean function as an equation of variables “and”ed and then “or”ed together, the same as the disjoint paths of a BDD, or the rows of a truth table.

Using variable juxtaposition as “and”, “+” as “or”, and “~” as “not, we can create a minterm representation of a Boolean function, \( F(a,b,c) = ab + \sim ac \). The set of minterms for \( F \) are \( \text{MIN}(F(a,b,c)) = abc + ab\sim c + \sim abc + \sim a\sim bc \).

These representations are logically equivalent, and if you build the truth table, you will see that minterms are the “true” rows of a truth table.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>ab</td>
<td>~ac</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>
that truth tables, BDDs, and minterms are mathematically equivalent representations of Boolean functions. Something cannot be true about one representation that is not true about the others.

Perhaps one representation can be computed faster than another, or understood easier than another, or a given property proven more simply than another, but we are up against a basic fact about mathematics and equivalent representations: if I can say it with a truth table, then I can say it about a BDD, and if I can’t, then I can’t.
MCS, which are minterms without negated variables.

So given the previous function, \( F(a,b,c) = ab + \neg ac = abc + ab\neg c + \neg abc + \neg a\neg bc \), the min cut sets of \( F \) are \( \text{MCS}(F) = abc + ab + bc + c = ab + c \), which are the minterms, dropping negation.

The minterms of the min cut sets are \( \text{MIN}(	ext{MCS}(F)) = abc + ab\neg c + \neg abc + \neg a\neg bc + a\neg bc \). Notice that there is an additional term in green for \( \text{MIN}(	ext{MCS}(F)) \), \( a\neg bc \), which did not exist in the minterms.

Therefore the number of minterms in the min cut sets for a function \( F \) is always greater than, or equal to, the number of minterms of \( F \), or mathematically, \( \text{Card}(\text{MIN}(	ext{MCS}(F))) \geq \text{Card}(\text{MIN}(F)) \). This is called the upper approximation, or monotone hull, of a Boolean function.
The point of the forgoing discussions was to show one very important idea: the disjoint terms of minterms, the disjoint paths of a BDD, and the disjoint rows of a truth table are all the same thing, with the disjoint terms of the minterms of min cut sets providing an upper approximation.

But why is any of this of interest to PSA?

Because we use these representations to quantify fault trees, which are a graphical representation of Boolean equations. And what we are interested in from the PSA point of view is the probability of the top event of a fault tree.
To calculate the probability of a Boolean equation in minterms or a BDD, it suffices to assign probabilities to each variable, substitute multiplication for “and”, addition for “or”, and subtract the probability for a variable from 1 for negation. Again using F:

\[ F(a,b,c) = ab + \sim ac \]

If \( Pr(a) = Pr(b) = Pr(c) = 0.1 \), then for \( \text{MIN}(F) = abc + ab\sim c + \sim abc + \sim a\sim bc \), the probability for \( F \) is:

\[ Pr(F) = 1 \times 10^{-3} + 9 \times 10^{-3} + 9 \times 10^{-3} + 8.1 \times 10^{-2} = 1 \times 10^{-1} \]

And for the minterms of the min cut sets, \( \text{MIN}(\text{MCS}(F)) = abc + ab\sim c + \sim abc + \sim a\sim bc \):

\[ Pr(\text{MCS}(F)) = 1 \times 10^{-3} + 9 \times 10^{-3} + 9 \times 10^{-3} + 8.1 \times 10^{-2} + 9.09 \times 10^{-2} = 1.09 \times 10^{-1} \]

Notice that if we apply the same trick directly to \( F(a,b,c) = ab + \sim ac \), and convert to min cut sets, \( \text{MCS}(F) = ab + c \), we have the rare event approximation:
If this were the whole story, then we know, with proofs in hand, that we can calculate the top event probabilities in PSA accurately.

But ain’t life grand? In the actual fault trees used in PSA it is impossible, in the most important cases, to completely construct truth tables, minterms, min cut sets, or BDDs.

So we rely on truncation. We say we are only interested in values for ninterms or min cut sets or (now) BDD paths which are greater than a certain value, the truncation cutoff, C.
Now assume that you are interested only in the MCS whose probability is greater than a given cutoff $C$. Then you can remove all the minterms whose MCS probability is lower than $C$, where the MCS probability (called $\text{MCSPr}$) of a minterm (or of a product in general) is defined as the product of positive literal probabilities:

\[
\text{Pr}(\neg abc) = (1 - \text{Pr}(a)) \times \text{Pr}(b) \times \text{Pr}(c)
\]

\[
\text{MCSPr}(\neg abc) = \text{Pr}(b) \times \text{Pr}(c)
\]
Let us denote by $F/C$ the restriction of $F$ to the minterms whose MCS probability is bigger than $C$ and let $\text{MCS}(F)/C$ be the set of MCS of $F$ whose probability is bigger than $c$. Then the following theorem holds [Rauzy2000]:

$$\text{MCS}(F/C) = \text{MCS}(F)/C$$

Assume again that $\Pr(a) = \Pr(b) = \Pr(c) = 0.1$ and that $C = 0.05$. Then we have:

$$\text{MCS}(F/C(a,b,c)) = \text{abc} + \text{ab} \sim \text{c} + \sim \text{abc} + \sim \text{a} \sim \text{bc}$$ (minterms whose MCS probability is lower than $C$ are in red).

Therefore:

$$\text{MCS}(F/C) = \text{MCS} (\sim \text{a} \sim \text{bc}) = \{ \text{c} \}$$

$$= \text{MCS}(F)/C \ (= \{ \text{ab}, \text{c} \} \ )$$
Note by the way that

\[ \text{MIN}(\text{MCS}(F/C)) = \text{MIN}(c) = abc + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}\overline{c}, \text{ and in therefore:} \]

\[ \text{MIN}(\text{MCS}(F/C)) = \text{MIN}(F) \cup \{A\cdot \overline{B}\cdot C\} / A\cdot B\cdot \overline{C} \] (where U stands for the union and / stands for the set difference)

So, what you get with truncated minterms is neither an upper-approximation (for you remove \( ab\overline{c} \)), nor under-approximation (for you add \( a\overline{b}c \)), but an approximation of unknown error.

In other words, truncating minterms, truth tables or BDDs works but only from an MCS point of view, in other words, only if we include no negation.
Now the question is does probability truncation work with BDDs?

Assume that you truncate by considering the probability of branches as you build the BDD. Then you go nowhere, due to the specific structure of BDDs. Look at the function $F$:

$$F = a_1 + a_2 + \ldots + a_n$$

$$D(F) = \text{bdd}_1$$
$$d_1 = \text{ite}(a_1, 1, \text{bdd}_2)$$
$$d_2 = \text{ite}(a_2, 1, \text{bdd}_3)$$

$$d_n = \text{ite}(a_n, 1, 0)$$
by eliminating all branches whose probability is below the cutoff \( C \). Then as we build this BDD,

![Diagram](image)

and we truncate counting the probability of 0-branches, we may reach a point where \((1 - \Pr(a_1)) \times \cdots \times (1 - \Pr(a_i)) < C\), therefore eliminating some perfectly valid MCS, and in doing so, under estimate the probability of the function \( F \).
complex function, we may add some perfectly invalid MCS, for example, let $H$, $F$, and $G$ be Boolean functions such that:

$$H = \sim FG$$

$$F = a_1 + \ldots + a_n$$

$$G = a_nB$$

The truncation of $F$ may eliminate the MCS $\{a_n\}$, which is part of the function $G$, and we end up with an invalid MCS, $a_nB$, giving us an over estimation.

Together, BDD with truncation gives us an approximation of unknown error.

Notice that this is the same result as minterm truncation, which is what we would expect since they are equivalent representations. Moreover, this is the same objection which was raised concerning the Destructive Truth Table Method and Direct Probability Calculation.
No Free Lunch

As we said in CWTPRA, the rare event is fine for what it does

Proposed method is supposed to help with negation, but this is where it fails (success branches, delete-terms, recovery actions)

No mention in other papers of the limitations of the methods

No mention of previous work

What about keeping track of what is truncated?

No mention of modules or variable ordering

Takes away from the real focus which is model clarity and transportability etc.
But among the crowds a little child suddenly gasped out, "But he hasn't got anything on." And the people began to whisper to one another what the child had said. "He hasn't got anything on." "There's a little child saying he hasn't got anything on." Till everyone was saying, "But he hasn't got anything on." The Emperor himself had the uncomfortable feeling that what they were whispering was only too true.