Efficient Algorithms to Assess Components and Gates Importances in Fault Tree Analysis

Y. Dutuit
LAP
Université Bordeaux I,
33405 Talence Cedex, FRANCE
dutuit@hse.iuta.u-bordeaux.fr

A. Rauzy
LaBRI, CNRS,
351, cours de la Libération,
33405 Talence Cedex, FRANCE
rauzy@labri.u-bordeaux.fr

Abstract

One of the principal activities of risk assessment is expected to be either the ranking or the categorization of structures, systems and components with respect to their risk-significance or their safety-significiance. Several measures, so-called importance factors, of such a significiance have been proposed for the case where the support model is a fault tree. In this article, we show how Binary Decision Diagrams can be use to assess efficiently a number of classical importance factors. This work completes the preliminary results obtained recently by Andrews and Sinnamon, and the authors. It deals also with the notion of joint reliability importance.

Keywords: Fault Trees, Importance Factors, Binary Decision Diagrams

1 Introduction

One of the principal activities of risk assessment is expected to be either the ranking or the categorization of structures, systems and components with respect to their risk-significance or their safety-significiance. Several measures of such a significiance have been proposed for the case where the support model is a fault tree. These measures are grouped under the generic designation of "importance factors". Many articles have been devoted to their mathematical expressions, their physical interpretations and the various ways they can be evaluated by computer programs.

Importance factors can be grouped into two categories:

- Importance factors that are computed at one point in the time. In this article, we study importance factors that belong to this category, namely the marginal importance factor (MIF), the critical importance factor (CIF), the diagnostic importance

factor (DIF), the risk achievement worth (RAW), and the risk reduction worth (RRW).

- Importance factors that average such quantities over a time period, in order to avoid the difficult task of selecting points of the time to be studied (e.g. [BP75, Nat79, Nat85]). These importance factors are indeed much more costly to assess for, precisely, they require integrations over a time period. We shall not consider them it this article.

Some authors studied extensions of these importance factors to assess the interaction between components in contributing to system reliability (e.g. [HL93, Arm95, HKL00, CPS98]).

Most of these works are based on classical methods to assess the top event probability, i.e. methods that work with the well-known Sylvester-Poincare development applied to minimal cutsets. As pointed out in [Fle96] and [Ves96], this kind of methods may induce misleading conclusions because they make a lot of approximations. These approximations are in general safe for what concerns the assessment of the top event probability, but they are often too rough to rank components properly.

It is now known that these classical methods are outperformed by the so-called Binary Decision Diagrams (BDD) technique (see for instance [Rau93]). With BDDs, not only the assessment of the top event probability is more efficient than with classical methods, but also the obtained results are exact (for no approximation is needed).

It was therefore of a great interest to revisit importance factors from the BDD point of view. J. Andrews and R. Sinnamon initiated such a work in [SA96, SA97] and further studied by the authors in [DR99]. The present article completes their work. Our results are twofold.

- First, we show that the definitions of the main importance factors work even in the case where the component under study is not reduced to a terminal event and/or the structure function is not monotone.

- Second, we proposed efficient algorithms to assess these importance factors (that improve those proposed in [SA96, SA97] for MIF and CIF measures).

- Third, we show by means of an example that the algorithms we propose makes it possible to compute easily the joint reliability importance [HL93, Arm95, HKL00] of two gates in a fault tree.

It is worth noticing that the algorithms we propose give exact results, therefore avoiding the pitfalls pointed out in [Fle96, Ves96].

The remaining of this article is organized as follows. In section 3, we give the mathematical definitions of the importance factors we consider here. In section 4, we recall basics about BDDs. In section 5 we give basic algorithms to assess probabilistic quantities of interest. In section 6, we show how these basic algorithms can be used to compute all of the importance factors. Finally, some experimental results are presented section 7.

2 Definitions and Notations

In this article, we shall use the following conventions.

- Boolean variables (terminal events) are denoted by lower case letters, e.g. a, e, ...
- Boolean functions (fault trees) are denoted by upper case letters, e.g. S, F, ...
- The disjunct of two Boolean functions F and G is denoted by $F+G$. Their conjunct is denoted by $F.G$, or even by FG. The negation of F is denoted by $\overline{F}$.
- The probability of a Boolean function F is denoted by $p(F)$.
- The conditional probability of a Boolean function F given a function G is denoted by $p(F|G)$. The fundamental equality of conditional probabilities is as follows.

$$p(F.G) \quad = \quad p(F|G).p(G) \tag{1}$$

A product is a conjunct of literals (i.e. variables or negation of variables) that does not contain both a literal and its opposite (e and $\overline{e}$). Let V be a finite set of variables. A product that contains a literal built over each variable of V is called a *minterm* of V. We denote by $minterms(V)$ the set of minterms that can be built over V. From a probabilistic point of view, minterms are always pairwisely independent.

Any Boolean function F can be assimilated to a disjunct of minterms. It is often convenient to write that the minterm π belongs to the function F, when $F = \pi + \ldots$, and that the literal l belongs to the product (or the minterm) π, when $\pi = l.\ldots$.

We denote by $F[v/e]$, where F is a Boolean function, e is a Boolean variable and $v \in \{0, 1\}$, the function F in which the value v has been substituted for the variable e. $F[1/e]$ and $F[0/e]$ are called respectively the positive and negative cofactors of F w.r.t. e.

Most of the importance factors were defined with the implicit assumption that the fault tree under study is coherent. A function F is monotone (or coherent) w.r.t. a variable e if for all minterm $\pi = \overline{e}.\pi'$ that belongs to F, the minterm $\rho = e.\pi'$ belongs to F as well. A function is (globally) monotone if it is monotone w.r.t. all of its variable.

3 Importance Factors

In this section, we present the six importance factors we consider throughout the paper. We recall their mathematical definitions as well as their possible physical interpretations. Detailed presentations of these measures can be found in textbooks (for instance [HR94, KKP97, MKK99]).

It is worth noticing that all of these importance factors depend on the time t at which the system and its components are observed. In what follows, the time t is omitted, although it is implicitely present in all of the definitions.

Conditional Probability The conditional probability $p(S|e)$, where S denotes the structure function of the fault tree under study and e denotes one of its basic events, is not strictly speaking an importance factor. However, the assessment of this quantity is one of the keypoints of the assessment of the other importance factors.

It is easy to show that the following equality holds.

$$p(S|e) \quad = \quad p(S[1/e]) \tag{2}$$

Marginal Importance Factor The marginal importance factor, denoted by $MIF(S,e)$, is defined as follows.

$$MIF(S,e) \quad \stackrel{\text{def}}{=} \quad \frac{\partial[p(S)]}{\partial[p(e)]} \tag{3}$$

MIF is often denoted by I_B and called Birnbaum importance factor in the literature for it has been introduced in [Bir69]. As we shall see, it can be interpreted, when S is a monotone function, as the conditional probability that, given e, the system S is failed and e is critical, i.e. a repair of e makes the system working.

Formally, let V be the set of variables of S and let us denote $crit(S,e)$ the set of critical states of S w.r.t. e. $crit(S,e)$ is defined as follows.

$$crit(S,e) \quad \stackrel{\text{def}}{=} \quad \{e.\pi \in minterms(V); e.\pi \in S \ \wedge \ \overline{e}.\pi \notin S\} \tag{4}$$

$crit(S,e)$ is a Boolean function. Another way to write it is as follows.

$$crit(S,e) \quad = \quad e.(S[1/e].\overline{S[0/e]}) \tag{5}$$

Now, the following equalities hold.

$$\begin{aligned} MIF(S,e) \quad &= \quad \frac{\partial p(S)}{\partial p(e)} = \frac{\partial[p(e).[p(S[1/e]) - p(S[0/e])] + p(S[0/e])]}{\partial[p(e)]} \\ &= \quad p(S[1/e]) - p(S[0/e]) \qquad (6) \\ &= \quad p(S|e) - p(S|\overline{e}) \qquad (7) \end{aligned}$$

If S is monotone w.r.t. e, then any minterm of $S[0/e]$ is also a minterm of $S[1/e]$. Therefore, $p(S[1/e]) - p(S[0/e]) = p(S[1/e].\overline{S[0/e]})$. It follows that:

$$MIF(S,e) \quad = \quad p(crit(S,e)|e) \tag{8}$$

It is worth noticing that $MIF(S,e)$ does not depend on $p(e)$. Therefore, MIF can be used only to rank the components according to their criticity.

Equation 6 is often taken as the definition of the Birnbaum importance factor. In order to generalize this measure to non-basic components, the equation 7 should be taken as the definition. This has however the drawback to make impossible an interpretation of MIF in term of critical states.

Note that in the case where S is not a monotone function, this interpretation is anyway no longer valid. Consider for instance $S = ab + \overline{a}c$. Then, $p(S|a) = p(S[1/a]) = p(b)$, $p(S|\overline{a}) = p(S[0/a]) = p(c)$, $crit(S,a) = ab\overline{c}$. It follows that $p(S|a) - p(S|\overline{a}) = p(b\overline{c}) - p(\overline{b}c) \neq p(crit(S,a)|a) = p(b\overline{c})$. It is worth noticing that $p(S|a) - p(S|\overline{a})$ may be negative, while, by definition, $p(crit(S,e)|e)$ is non-negative for all S and e. Therefore, to extend the interpretation of MIF in terms of critical states to non-monotone functions (and basic components), the equation 8 could be used as a definition.

Critical Importance Factor The criticality of a component is related to the potential improvement of the system reliability resulting from the improvement of the component reliability. It is clear that it would be more difficult and costly to improve the more reliable components than to improve the less reliable ones. However, the marginal importance factor does not depend on the component reliability. The critical importance factor, denoted by $CIF(S,e)$, is another measure of component criticality that does depend on component reliability. It is defined as follows.

$$CIF(S,e) \stackrel{\text{def}}{=} \frac{p(e)}{p(S)} \times MIF(S,e) \tag{9}$$

$CIF(S,e)$ has been introduced in [Lam75]. The following equality holds, by definition 9 and equality 8.

$$CIF(S,e) = p(crit(S,e)|S) \tag{10}$$

Therefore, in the case of monotone systems, $CIF(S,e)$ can be interpreted as the conditional probability that the system is in a critical state w.r.t. e, given that the system is failed. For the same reasons as previously, if S is not monotone and/or if e is not a terminal event, this interpretation is not valid.

Diagnostic Importance Factor The diagnostic importance factor, denoted by $DIF(S,e)$, is defined as follows.

$$DIF(S,e) \stackrel{\text{def}}{=} p(e|S) \tag{11}$$

DIF is often denoted by $I_{VF}(S)$ and called Vesely-Fussel Importance factor, for it has been introduced by Vesely and Fussel in [Fus75]. Since, by equation 1, $p(e|S) = \frac{p(S.e)}{p(S)}$, $DIF(S,e)$ is the fraction of the system unavailability (or risk) that involves the component failure. It is worth noticing that this interpretation still works in the cases where S is not monotone and/or e is not a terminal event.

Risk Achievement Worth The risk achievement worth, denoted by $RAW(S,e)$, is defined as follows.

$$RAW(S,e) \stackrel{\text{def}}{=} \frac{p(S|e)}{p(S)} \tag{12}$$

$RAW(S,e)$ is also called risk increase factor. It measures the increase in system failure probability assuming the worst case of failing component. It is an indicator of the importance of maintaining the current level of reliability for the component [CPS98]. In reference [WW96], it is argued that RAW should be used with cared, for it is rather rough.

Risk Reduction Worth The risk reduction worth, denoted by $RRW(S,e)$, is defined as follows.

$$RRW(S,e) \stackrel{\text{def}}{=} \frac{p(S)}{p(S|\overline{e})} \tag{13}$$

$RRW(S,e)$ is also called risk decrease factor. It represents the maximum decreasing of the risk it may be expected by increasing the reliability of the component. Therefore this quantity may be used to select components that are the best candidates for efforts leading to improving system reliability.

4 Binary Decision Diagrams

Bryant's Binary Decision Diagrams (BDDs) [BRB90] are the state-of-the-art data structure to encode and to manipulate Boolean functions. Since their introduction in the reliability analysis framework [Rau93, CM94], BDDs have proved to be the most efficient technique to assess fault trees.

The BDD associated with a formulae is a compact encoding of the truth table of this formula. This representation is based on the Shannon decomposition: Let F be a Boolean formula that depends on the variable v, then $F = v.F[1/v] + \overline{v}.F[0/v]$. By choosing a total order over the variables and applying recursively the Shannon decomposition, the truth table of any formula can be graphically represented as a binary tree. The nodes are labeled with variables and have two outedges (a *then*-outedge, pointing to the node that encodes $F[1/v]$, and a *else*-outedge, pointing to the node that encodes $F[0/v]$). The leaves are labeled with either 0 or 1. The value of the formula for a given variable assignment is obtained by descending along the corresponding branch of the tree. The Shannon tree for the formula $ab + \overline{a}c$ and the lexicographic order is pictured Fig. 1 (dashed lines represent *else*-outedges).

Indeed such a representation is very space consuming. It is however possible to shrink it by means of the following two reduction rules.

- Isomorphic subtrees merging. Since two isomorphic subtrees encode the same formula, at least one is useless.
- Useless nodes deletion. A node with two equal sons is useless since it is equivalent to its son ($v.F + \overline{v}.F = F$).

By applying these two rules as far as possible, one get the BDD associated with the formula. A BDD is therefore a directed acyclic graph. It is unique, up to an isomorphism. This process is illustrated on Fig. 1.

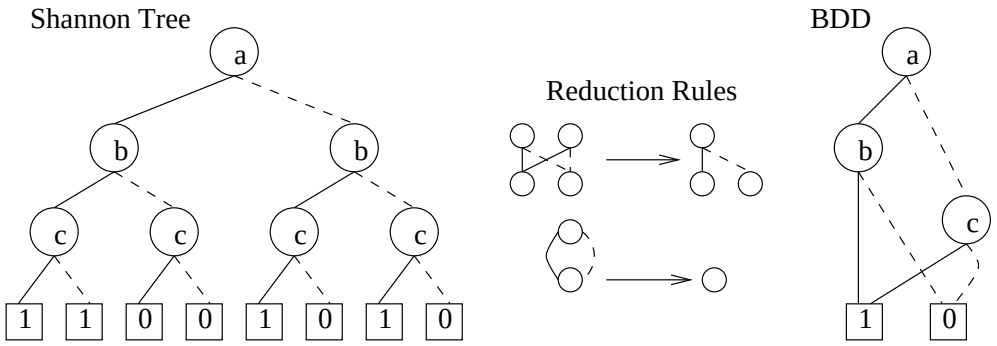

Figure 1: From the Shannon tree to the BDD encoding $ab + \overline{a}c$.

Logical operations (and, or, xor, ...) can be directly performed on BDDs. This results from the orthogonality of usual connectives and the Shannon decomposition:

$$(v.F_1 + \overline{v}.F_0) \odot (v.G_1 + \overline{v}.G_0) \quad = \quad v.(F_1 \odot G_1) + \overline{v}.(F_0 \odot G_0) \tag{14}$$

where $\odot$ is any binary connective.

Among other consequences, this means that the complete binary tree is never built and then shrunk: the BDD encoding a formula is obtained by composing the BDDs encoding its subformulae. Moreover, a caching principle is used to store intermediate results of computations. This makes the usual logical operations (conjunction, disjunction) polynomial in the sizes of their operands. A complete implementation of a BDD package is described in [BRB90]. The reader interested in more details should thus refer to this article.

Discussion It is widely known, since the very first uses of BDDs, that the chosen variable ordering has a great impact on the size of BDDs, and therefore on the efficiency of the whole methodology. Finding the best ordering (or even a reasonnably good one) is a very hard problem (see [Weg00] for a recent survey on this topics). Two kinds of heuristics are used to determine which variable ordering to apply. Static heuristics are based on topoligical considerations and select the variable ordering once for all (see for instance [FFK88]). Dynamic heuristics change the variable ordering at some points of the computation. They are thus more versatile than the formers, but the price to pay is a serious increase of running times. Sifting is the most widely used dynamic heuristics [Rud93].

The algorithms proposed in this article are independent of the choice of a variable ordering, although this choice influences their efficiency. For the experiments we report, a simple depth-first left-most ordering was used. This heuristics gives often very good result.

5 Basic Algorithms

In this section, we propose different BDD algorithms to compute $p(S)$, $p(S|C)$ and $MIF(S, C)$, where S and C are Boolean functions (C is in general reduced to a terminal event e). These

```
Pr(S)
    if S = 0 return 0
    if S = 1 return 1
    if S = v.S_1 + v̄.S_0
        if cache has entry ⟨S, p⟩ return p
        p_1 ← Pr(S_1)
        p_0 ← Pr(S_0)
        p ← p(v).p_1 + [1 − p(v)].p_0
        add to cache ⟨S, p⟩
        return p
```

Figure 2: The pseudo-code for the `Pr` algorithm.

algorithms will be used in the next section to assess the other importance factors.

Algorithms are named using this `font` in order to distinguish them from the mathematical quantities they compute. E.g. $\texttt{MIF}(S, C)$ denotes the algorithm that computes $MIF(S, C)$. Moreover, in the expression $\texttt{MIF}(S, C)$, S and C denote the BDDs that encode the corresponding functions.

The algorithm to compute $p(S)$. The first basic algorithm that is needed to assess importance factors is indeed the computation of the top event probability from the basic events probabilities. In the case of Binary Decision Diagrams, this algorithm is based on the Shannon decomposition. It is fully described by the following equations (that recurse over BDD nodes).

$$\begin{array}{lcl} \texttt{Pr}(1) & = & 1 \\ \texttt{Pr}(0) & = & 0 \\ \texttt{Pr}(v.S_1 + \overline{v}.S_0) & = & p(v).\texttt{Pr}(S_1) + [1 - p(v)].\texttt{Pr}(S_0) \end{array} \tag{15}$$

The algorithm Pr computes the exact probability. Moreover, thanks to the memorization of intermediate results, the algorithm is called only once on each BDD node. Its complexity is therefore of linear in the size of the BDD [Rau93].

The pseudo-code for actual algorithm is given Fig. 2. This pseudo-code makes explicit the way the memorization of intermediate results works. To be efficient, a BDD algorithm must compute the quantity it assesses bottom-up in order to make it possible to use this caching mechanism.

All of the algorithms presented in what follows do work bottom-up. We shall describe them only by means of set of recursive equations as equations 15, but the use of a caching mechanism will be implicitely assumed.

Algorithms to compute $p(S|C)$. The second quantity of interest for our purpose is the conditional probability $p(S|C)$. A first way to assess $p(S|C)$ is to apply the fundamental

theorem of conditional probabilities (equation 1).

$$\texttt{CPr}_1(S,C) \;=\; \frac{\texttt{Pr}(\textsc{BddAnd}(S,C))}{\texttt{Pr}(C)} \tag{16}$$

The algorithm $\texttt{CPr}_1$ works in two steps. First, one computes the BDD that encodes the function $S.C$. Second, one computes the quotient $\frac{p(S.C)}{p(C)}$ (using Pr).

If C is reduced to a basic event e, the computation may be simplified by computing the BDD that encodes the cofactor $S[1/e]$ and then the probability from this BDD (by equation 2). The algorithm to compute $S[v/e]$, $v \in \{0,1\}$, is described by the following equations (recall that BDDs assume a total order over the variables [BRB90]).

$$\begin{array}{lcl}
\texttt{cofactor}(1,v,e) & = & 1 \\
\texttt{cofactor}(0,v,e) & = & 0 \\
\texttt{cofactor}(x.S_1 + \overline{x}.S_0, v, e) & = & \\
\multicolumn{3}{c}{\left\{\begin{array}{lll}
x.\texttt{cofactor}(S_1,v,e) + \overline{x}.\texttt{cofactor}(S_0,v,e) & \text{if} & x < e \\
S_1 & \text{if} & x = e \wedge v = 1 \\
S_0 & \text{if} & x = e \wedge v = 0 \\
x.S_1 + \overline{x}.S_0 & \text{if} & x > e
\end{array}\right.}
\end{array} \tag{17}$$

$\texttt{CPr}_2(S,e)$ is defined as follows.

$$\texttt{CPr}_2(S,e) \;=\; \texttt{Pr}(\texttt{cofactor}(S,1,e)) \tag{18}$$

The computation of $\textsc{BddAnd}(S,C)$ is in $\mathcal{O}(|S|.|C|)$ (where $|S|$ denotes the number of nodes of the BDD S) [BRB90]. It is clear from equations 17 that the computation $S[v/e]$ is in $\mathcal{O}(|S|)$. Since $\texttt{Pr}(S)$ is in $\mathcal{O}(|S|)$, $\texttt{CPr}_1(S,C)$ and $\texttt{CPr}_2(S,e)$ are respectively in $\mathcal{O}(|S|.|C|)$ and $\mathcal{O}(|S|)$, which is indeed very good.

It remains that the construction of the BDDs that encode respectively $S.C$ and $S[1/e]$ may be space consuming (if these BDDs are otherwise useless). It is therefore questionable whether these constructions can be avoided. The answer is positive.

In the case where C is reduced to the terminal event e, the idea is to performed the (virtual) computation of the cofactor together with the (actual) computation of the probability. The algorithm $\texttt{CPr}_3$ is defined by the following equations.

$$\begin{array}{lcl}
\texttt{CPr}_3(1,v,e) & = & 1 \\
\texttt{CPr}_3(0,v,e) & = & 0 \\
\texttt{CPr}_3(x.S_1 + \overline{x}.S_0, v, e) & = & \\
\multicolumn{3}{c}{\left\{\begin{array}{lll}
p(x).\texttt{CPr}_3(S_1,v,e) + [1-p(x)].\texttt{CPr}_3(S_0,v,e) & \text{if} & x < e \\
\texttt{Pr}(S_1) & \text{if} & x = e \wedge v = 1 \\
\texttt{Pr}(S_0) & \text{if} & x = e \wedge v = 0 \\
\texttt{Pr}(x.S_1 + \overline{x}.S_0) & \text{if} & x > e
\end{array}\right.}
\end{array} \tag{19}$$

Finally, a similar idea applies to compute directly $p(S|C)$. One combines the (virtual) computation of the conjunct of S and C with the (actual) computation of the probability

of this conjunct. The algorithm $\texttt{CPr}_4$ is defined by the following equations.

$$\texttt{CPr}_4(S,C) \;=\; \frac{\texttt{PrAnd}(S,C)}{\texttt{Pr}(C)} \tag{20}$$

$$\begin{array}{lcl}
\texttt{PrAnd}(0,C) & = & 0 \\
\texttt{PrAnd}(F,0) & = & 0 \\
\texttt{PrAnd}(1,1) & = & 1 \\
\texttt{PrAnd}(x.S_1+\overline{x}.S_0,1) & = & \texttt{Pr}(x.S_1+\overline{x}.S_0) \\
\texttt{PrAnd}(1,y.C_1+\overline{y}.C_0) & = & \texttt{Pr}(y.C_1+\overline{y}.C_0) \\
\texttt{PrAnd}(x.S_1+\overline{x}.S_0,y.C_1+\overline{y}.C_0) = & & \\
\quad \left\{ \begin{array}{ll}
p(x).\texttt{PrAnd}(S_1,C)+[1-p(x)].\texttt{PrAnd}(S_0,C) & \text{if} \quad x<y \\
p(x).\texttt{PrAnd}(S_1,C_1)+[1-p(x)].\texttt{PrAnd}(S_0,C_0) & \text{if} \quad x=y \\
p(y).\texttt{PrAnd}(S,C_1)+[1-p(y)].\texttt{PrAnd}(S,C_0) & \text{if} \quad x>y
\end{array} \right. &
\end{array} \tag{21}$$

Thanks to the caching principle and from equations 19, 20 and 21, it is clear that $\texttt{CPr}_3(S,v,e)$ and $\texttt{CPr}_4(S,C)$ are respectively in $\mathcal{O}(|S|)$ and $\mathcal{O}(|S|.|C|)$.

Algorithms to compute $MIF(S,C)$. The last quantity of interest for our purpose is the marginal importance factor $MIF(S,C)$. Equation 7 gives a mean to compute $MIF(S,C)$ from $p(S|C)$ and $p(S|\overline{C})$. If C is reduced to a terminal event any of the $\texttt{CPr}_i$ above defined works. Otherwise, one should use either $\texttt{CPr}_1$ or $\texttt{CPr}_4$. Note that equation 7 induces two conditional probability computations. It is possible to assess $MIF(S,C)$ with only one call at $\texttt{CPr}_i$ and two calls at $\texttt{Pr}$.

$$MIF(S,C) \;\overset{eq.\ 7}{=}\; p(S|C)-p(S|\overline{C}) = \frac{p(S|C)-p(S)}{1-p(C)}$$

However, there exists much more direct means to assess $MIF(S,C)$.

If C is reduced to a terminal event e, a first method consists in computing a partial derivative. First, the value of $p(S)$ is assessed for $p(e)$, then it is assessed for $p(e)+\delta(e)$, finally the following quotient is computed.

$$\texttt{MIF}_1(S,e) \;\approx\; \frac{\texttt{Pr}(S)_{p(e)+\delta p(e)} - \texttt{Pr}(S)_{p(e)}}{\delta p(e)} \tag{22}$$

The second method consists in computing a BDD that encodes the critical state of S w.r.t. e, and then to assess $MIF(S,e)$ from this BDD. The algorithm that computes the critical states of S w.r.t. e looks like `cofactor`. It is defined by the following equations.

$$\begin{array}{lcl}
\texttt{crit}(1,e) & = & 0 \\
\texttt{crit}(0,e) & = & 0 \\
\texttt{crit}(x.S_1+\overline{x}.S_0,e) & = & \\
\quad \left\{ \begin{array}{ll}
x.\texttt{crit}(S_1,e)+\overline{x}.\texttt{crit}(S_0,e) & \text{if} \quad x<e \\
\texttt{AND}(S_1,\texttt{NOT}(S_0)) & \text{if} \quad x=e \\
0 & \text{if} \quad x>e
\end{array} \right. &
\end{array} \tag{23}$$

Note that the BDD built by $\texttt{crit}(S,e)$ does not depend on e (conversely to the definition 4). This makes it possible to compute $MIF(S,e)$ without divising the result by $p(e)$.

$$\texttt{MIF}_2(S,e) \;=\; \texttt{Pr}(\texttt{crit}(S,e)) \tag{24}$$

A third method consists in computing $MIF(S,e)$ directly from the BDD that encodes S. This is achieved by means of a recursive algorithm that looks like $\texttt{CPr}_3$. $\texttt{MIF}_3$ is defined by the following recursive equations.

$$\begin{array}{lcl}
\texttt{MIF}_3(1,e) & = & 0 \\
\texttt{MIF}_3(0,e) & = & 0 \\
\texttt{MIF}_3(x.S_1+\overline{x}.S_0,e) & = & \\
\multicolumn{3}{c}{\left\{\begin{array}{lll}
p(x).\texttt{MIF}_3(S_1,e)+[1-p(x)].\texttt{MIF}_3(S_0,e) & \text{if} & x<e \\
\texttt{Pr}(S_1)-\texttt{Pr}(S_0) & \text{if} & x=e \\
0 & \text{if} & x>e
\end{array}\right.}
\end{array} \tag{25}$$

$\texttt{MIF}_1(S,e)$ requires two calls to $\texttt{Pr}(S)$. It is therefore in $\mathcal{O}(|S|)$. $\texttt{MIF}_3(S,e)$ requires only traversal of S, since for each node either $\texttt{CPr}_3$ is called or $\texttt{Pr}$ is called, but not both. It is therefore also in $\mathcal{O}(|S|)$. The computation of $\texttt{crit}(S,e)$ is quadratic in the worst case because $\texttt{AND}$ is so. Therefore $\texttt{MIF}_2(S,e)$ is in $\mathcal{O}(|S|^2)$. Note however that this worst case complexity is seldom reached in practice.

In the case where C is not reduced to a terminal event, it is still possible to assess $p(S|C)-p(S|\overline{C})$ in one pass over these BDDs. The idea is as follows.

$$\begin{array}{rcl}
p(S|C)-p(S|\overline{C}) & = & \dfrac{p(S.C)}{p(C)}-\dfrac{p(S.\overline{C})}{p(\overline{C})} \\
& = & \displaystyle\sum_{\pi\in S}\left\{\begin{array}{lll}
\frac{1}{p(C)}.p(\pi) & \text{if} & \pi\in C \\
-\frac{1}{1-p(C)}.p(\pi) & \text{if} & \pi\notin C
\end{array}\right.
\end{array}$$

The resulting algorithm looks like $\texttt{PrAnd}$. The trick is to pass $p(C)$ as a parameter.

$$\texttt{MIF}_4(S,C) \;=\; \texttt{MIF}_4^\star(S,C,\texttt{Pr}(C))$$

$$\begin{array}{lcl}
\texttt{MIF}_4^\star(0,C,\rho) & = & 0 \\
\texttt{MIF}_4^\star(1,C,\rho) & = & \\
\multicolumn{3}{c}{\left\{\begin{array}{lll}
-\frac{1}{1-\rho} & \text{if} & C=0 \\
\frac{1}{\rho} & \text{if} & C=1 \\
p(y).\texttt{MIF}_4^\star(1,C_1,\rho)+[1-p(y)].\texttt{MIF}_4^\star(1,C_0,\rho) & \text{if} & C=y.C_1+\overline{y}.C_0
\end{array}\right.} \\
\texttt{MIF}_4^\star(S,0,\rho) & = & -\frac{1}{1-\rho}.\texttt{Pr}(S) \\
\texttt{MIF}_4^\star(S,1,\rho) & = & \frac{1}{\rho}.\texttt{Pr}(S) \\
\texttt{MIF}_4^\star(x.S_1+\overline{x}.S_0,y.C_1+\overline{y}.C_0,\rho) & = & \\
\multicolumn{3}{c}{\left\{\begin{array}{lll}
p(x).\texttt{MIF}_4^\star(S_1,C,\rho)+[1-p(x)].\texttt{MIF}_4^\star(S_0,C,\rho) & \text{if} & x<y \\
p(x).\texttt{MIF}_4^\star(S_1,C_1,\rho)+[1-p(x)].\texttt{MIF}_4^\star(S_0,C_0,\rho) & \text{if} & x=y \\
p(y).\texttt{MIF}_4^\star(S,C_1,\rho)+[1-p(y)].\texttt{MIF}_4^\star(S,C_0,\rho) & \text{if} & x>y
\end{array}\right.}
\end{array} \tag{26}$$

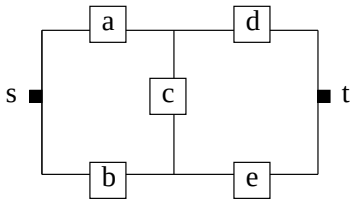

Figure 3: The bridge.

It is clear from equations 26 that $\texttt{MIF}_4(S,C)$ is in $\mathcal{O}(|S|.|C|)$.

J. Andrews and R. Sinnamon proposed in [SA96, SA97] an algorithm to assess $MIF(S,e)$ in the case where e is a terminal event. This algorithm relies on different principles and is not as efficient as those presented above. The algorithms $\texttt{CPr}_i$ and $\texttt{MIF}_i$ we proposed here are thus entirely new.

6 Algorithms to assess importance factors

The importance factors defined in section 3 can be assessed using only the basic algorithms presented in the previous section. The game consists in rewriting their definitions using $p(S)$, $p(C)$ and either and $p(S.C)$ or $p(S|C)$ or $MIF(S,C)$. The table 1 provides such a rewriting. The negation can be performed in constant time over BDDs [BRB90]. Therefore we didn't consider them as a basic algorithm.

7 Experiments

As an illustration, we give in this section some experimental results.

The bridge. The first example is the well-known bridge network pictured Fig. 3. The function that encodes s-t cuts is $S = C_1 + C_2 + C_3 + C_4$; where $C_1 = ab$, $C_2 = de$, $C_3 = ace$ and $C_4 = bcd$.

The following table gives the values of the main importance factors for different com-

	Basic algorithms		
	$\texttt{Pr}, \textsc{BddAnd}$	$\texttt{Pr}, \texttt{CPr}$	$\texttt{Pr}, \texttt{MIF}$
$p(S\|C)$	$\frac{\texttt{Pr}(\textsc{BddAnd}(S,C))}{\texttt{Pr}(C)}$	$\texttt{CPr}(S,C)$	$\texttt{Pr}(S) + (1 - \texttt{Pr}(C)).\texttt{MIF}(S,C)$
$MIF(S,C)$	$\frac{\frac{\texttt{Pr}(\textsc{BddAnd}(S,C))}{\texttt{Pr}(C)} - \texttt{Pr}(S)}{1 - \texttt{Pr}(C)}$	$\frac{\texttt{CPr}(S,C) - \texttt{Pr}(S)}{1 - \texttt{Pr}(C)}$	$\texttt{MIF}(S,C)$
$CIF(S,C)$	$1 - \frac{\texttt{Pr}(\textsc{BddAnd}(S, \textsc{BddNot}(C)))}{(1 - \texttt{Pr}(C)).\texttt{Pr}(S)}$	$1 - \frac{\texttt{CPr}(S, \textsc{BddNot}(C))}{\texttt{Pr}(S)}$	$\frac{\texttt{Pr}(C)}{\texttt{Pr}(S)} \times \texttt{MIF}(S,C)$
$DIF(S,C)$	$\frac{\texttt{Pr}(\textsc{BddAnd}(S,C))}{\texttt{Pr}(S)}$	$\frac{\texttt{Pr}(C).\texttt{CPr}(S,C)}{\texttt{Pr}(S)}$	$\texttt{Pr}(C) + \frac{\texttt{Pr}(C).[1 - \texttt{Pr}(C)].\texttt{MIF}(S,C)}{\texttt{Pr}(S)}$
$RAW(S,C)$	$\frac{\texttt{Pr}(\textsc{BddAnd}(S,C))}{\texttt{Pr}(C) \times \texttt{Pr}(S)}$	$\frac{\texttt{CPr}(S,C)}{\texttt{Pr}(S)}$	$1 + \frac{[1 - \texttt{Pr}(C)].\texttt{MIF}(S,C)}{\texttt{Pr}(S)}$
$RRW(S,C)$	$\frac{[1 - \texttt{Pr}(C)].\texttt{Pr}(S)}{\texttt{Pr}(\textsc{BddAnd}(S, \textsc{BddNot}(C)))}$	$\frac{\texttt{Pr}(S)}{\texttt{CPr}(S, \textsc{BddNot}(C))}$	$\frac{\texttt{Pr}(S)}{\texttt{Pr}(S) - \texttt{Pr}(C).\texttt{MIF}(S,C)}$

Table 1: Algorithms to compute importance factors.

ponents. The probability of S, not indicated in the table, is $p(S) = 0.234$.

C	Pr	CPr	MIF	CIF	DIF	RAW	RRW
a	0.1	0.432	0.22	0.0940171	0.184615	1.84615	1.10377
b	0.2	0.334	0.125	0.106838	0.28547	1.42735	1.11962
c	0.3	0.276	0.06	0.0769231	0.353846	1.17949	1.08333
d	0.4	0.537	0.505	0.863248	0.917949	2.29487	7.3125
e	0.5	0.4264	0.3848	0.822222	0.911111	1.82222	5.625
C_1	0.02	1	0.781633	0.0668062	0.0854701	4.2735	1.07159
C_2	0.2	1	0.9575	0.818376	0.854701	4.2735	5.50588
C_3	0.015	1	0.777665	0.0498503	0.0641026	4.2735	1.05247
C_4	0.024	1	0.784836	0.080496	0.102564	4.2735	1.08754
$C_1 + C_2$	0.216	1	0.977041	0.901884	0.923077	4.2735	10.192
$C_3 + C_4$	0.0378	1	0.796092	0.1286	0.161538	4.2735	1.14758

The main purpose of the above table is to show that BDDs make it possible to rank the influence of not only the basic components of the network, but also of paths, groups of paths, cuts, groups of cuts, ...

Joint Importance Reliability. The joint importance reliability (JRI) has been introduced in [HL93] and further studied in [Arm95, HKL00]. It measures the interaction of two components in contributing to the system reliability. Its definition is as follows.

$$JRI(S, C_1, C_2) \stackrel{\text{def}}{=} p(S|C_1.C_2) + p(S|\overline{C_1}.\overline{C_2}) - p(S|\overline{C_1}.C_2) - p(S|C_1.\overline{C_2}) \quad (27)$$

Negative JRI's indicate that one component becomes less important when the other is functionning [HL93].

In the case where both C_1 and C_2 are reduced to two basic events e_1 and e_2, it is not difficult to verify that the following equalities hold.

$$JRI(S, e_1, e_2) = MIF(S[1/e_1], e_2) - MIF(S[0/e_1], \overline{e_2})$$

The above equation can be used to design a one pass algorithm to assess $JRI(S, e_1, e_2)$.

The following table gives the values of the JRI for all of the pairs of s-t cuts of the bridge example. The basic events have the same failure probability p. In the table, we indicate for each pair (C_i, C_j) the conditional probability $p(S|\overline{C_i}.\overline{C_j})$ (the others conditional

probabilities involved in the definition of JRI are obviously equal to 1).

	$p = 0.5$		$p = 0.9$	
i, j	$CPr(\overline{C_i}.\overline{C_j})$	$JRI(C_i, C_j)$	$CPr(\overline{C_i}.\overline{C_j})$	$JRI(C_i, C_j)$
$1, 2$	0.111111	-0.888888	0.403878	-0.596122
$1, 3$	0.272727	-0.727272	0.816225	-0.183775
$1, 4$	0.272727	-0.727272	0.816225	-0.183775
$2, 3$	0.272727	-0.727272	0.816225	-0.183775
$2, 4$	0.272727	-0.727272	0.816225	-0.183775
$3, 4$	0.36	-0.64	0.837573	-0.162427

The above values are in accordance with those given in [HKL00] for $p = 0.5$ (but differ for $p = 0.9$).

A consecutive k-within-m-out-of-n system As a third example, we consider a consecutive k-within-m-out-of-n system. A consecutive k-within-m-out-of-n system consists of n linearly ordered components such that the system fails iff there are m consecutive components which include among them, at least k failed components. Consecutive k-within-m-out-of-n systems were introduced in [Gri86] as a generalization of both k-out-of$n$ and consecutive k-out-of-n systems.

Formally, the formula $S_{k,m,n}$ is defined as follows (assuming $1 \leq k \leq m \leq n$).

$$S_{k,m,n} \stackrel{\text{def}}{=} \sum_{i=1}^{n-m+1} k/m(e_i, e_{i+1}, \ldots, e_{i+m-1})$$

For the sake of the simplicity, we consider that all of the variables have the same probability p. The probability of a minimal solution is therefore p^k.

We did the following experiments for $k = 8$, $m = 16$ and $n = 64$.

First, we computed the BDD that encodes $S_{8,16,64}$. This computation took 3.06s on a pentium III (cadenced at 733 MegaHerz). The BDD is made of 442147 nodes ($S_{8,16,64}$ admits 321750 minimal cuts).

Second, we computed the marginal importance factor of $S_{k,m,n}$ and e_1, e_{16}, e_{32}, e_{48} and e_{64}, for different values of p. We used the algorithms $\texttt{MIF}_1$, $\texttt{MIF}_2$ and $\texttt{MIF}_3$. It makes sense to examine several variables for they are located at different levels in the BDD.

The running times are independent of p. Moreover, the three algorithms give exactly the same result for all $p > 10^{-3}$ (which was not that obvious, due to rounding errors). The following table gives the size of the auxiliary BDDs as well as the average running times of each algorithm. The computation of the top event probability took 0.79s on average.

event	e_1	e_{16}	e_{32}	e_{48}	e_{64}
time $\texttt{MIF}_1$	1.56s	1.56s	1.57s	1.58s	1.58s
$\lvert\texttt{crit}(S_{8,16,64}, e_i)\rvert$	350627	430707	430707	430707	430707
time `crit`	0.04s	0.42s	3.46s	25.96s	95.60s
time `Pr`	0.62s	0.80s	0.80s	0.77s	0.61s
time $\texttt{MIF}_3$	0.79s	0.79s	0.81s	0.85s	0.85s

At least one thing is clear: the running time of $\lvert\texttt{crit}(S_{8,16,64}, e_i)\rvert$ depends on the variable location in the BDD !

The above table shows that the situation is rather clear, at least when the BDD that encodes the structure function is large (which is the case here).

First, it is not interesting to build auxiliary BDDs, even in the perspective of many computations in a row (for instance to perform sensitivity analyses). Second, the numerical differentiation is always about twice as costly as the direct assessment (for it performs two BDD traversals). Therefore, $\texttt{MIF}_3$ should be prefered.

Other experiments have shown that the above remarks apply in general, even in the cases where the component is not reduced to a terminal event.

Sylvester-Poincaré versus Binary Decision Diagrams Binary Decision Diagrams give exact results. Their are also sometimes much more efficient that classical fault tree assessment algorithms such as MOCUS [FV72].

As an illustration, we consider now a classical test case so-called `baobab1` in the litterature. This fault tree is made of 61 terminal events and 84 gates (16 and-gates, 59 or-gates and 9 3-out-of-4-gates). We apply on this tree BDDs as well as our (highly optmized) version of MOCUS [Rau00]. For this latter algorithm, we consider 4 relative cutoffs. The minimal cutsets C such that the quotient $p(C)/\sum p(C)$ is below the relative cutoff are discarded. This makes it possible to reduce dramatically the computational effort.

The table 2 gives the results we obtained on the `baobab1` test case. These results illustrate that BDD are sometimes by order of magnitude more efficient than MOCUS like algorithms.

The latters make approximations that lead to rather unaccurate results. More exactly, either one accepts to pay a high computational cost (as with the relative cutoff 10^{-5} in our example) or the result may be subject to strong deviations.

Moreover, as soon as there is a rather important number of minimal cutsets, it is not possible to compute the Sylvester-Poincaré development for orders beyond two (within reasonable time constraints).

Finally, some minimal cutsets of low order may be missed anyway (as the cutset of order 3 in the above example), which is indeed problematic when computing importance factors.

Table 2: Results on the `baobab1` test case.

Algorithms	BDD	MOCUS with a relative cutoff of			
		10^{-2}	10^{-3}	10^{-4}	10^{-5}
#cutsets of order 2	1			1	1
#cutsets of order 3	1				
#cutsets of order 4	70		8	52	60
#cutsets of order 5	400		12	64	196
#cutsets of order 6	2212	36	220	644	1268
#cutsets of order 7	14748		4	528	4620
#cutsets of order 8	8460				
#cutsets of order 9	10624				
#cutsets of order 10	6600				
#cutsets of order 11	3072				
Total	46188	36	244	1289	6145
Running times	0.08s	10.28s	20.27s	57.54s	161.57s
Sylvester-Poincaré order 1		$3.59\ 10^{-7}$	$9.39\ 10^{-7}$	$1.38\ 10^{-6}$	$1.62\ 10^{-6}$
Running times		0.00s	0.00s	0.00s	0.00s
Sylvester-Poincaré order 2		$3.02\ 10^{-7}$	$7.07\ 10^{-7}$	$9.18\ 10^{-7}$	$9.64\ 10^{-7}$
Running times		0.00s	0.04s	1.31s	44.18s
Sylvester-Poincaré order 3		$3.20\ 10^{-7}$	$8.34\ 10^{-7}$	$1.38\ 10^{-6}$	
Running times		0.02s	3.50s	530.93s	
Exact probability	$1.28\ 10^{-6}$				
Running times	0.00s				

8 Conclusion

In this paper, we presented a complete picture of the BDD based algorithms to compute importance factors CPr, MIF, CIF, DIF, RAW, RRW and JRI. Most of these algorithms are entirely new. It is worth mentioning that:

- The algorithms we proposed give exact results (no approximation is performed).
- For each importance factor, we proposed at least one algorithm that works in the case where the component is not reduced to a terminal event. This opens new perspectives in the ranking of systems, structures and components with respect to their risk-significiance and safety-significiance. References [HL93, Arm95, CPS98] contain preliminary discussions on this topics.
- The algorithms we proposed are very efficient for they are linear in the size of the BDD that encodes the system if the component is reduced to a terminal event, and in the product of the sizes of the BDDs that encode the system and the component otherwise. We reported experiments that illustrate their efficiency.
- The algorithms we proposed are therefore good candidates to assess importance factors defined over a time period by means of a numerical integration.

The results presented here are a part of a study on BDD algorithms to assess Boolean risk assessment models. This study includes also on-going works on importance factors defined over minimal cutsets and time dependent analyses.

References

[Arm95] M.J. Armstrong. Joint reliability importance of components. *IEEE Transactions on Reliability*, 44:408–412, 1995.

[Bir69] Z.W. Birnbaum. On the importance of different components and a multicomponent system. In P.R. Korishnaiah, editor, *Multivariable analysis II*. Academic Press, New York, 1969.

[BP75] R.E. Barlow and F. Proschan. Importance of system components and fault tree events. *Stochastic Processes and their Applications*, 3:153–173, 1975.

[BRB90] K. Brace, R. Rudell, and R. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 40–45. IEEE 0738, 1990.

[CM94] O. Coudert and J.-C. Madre. MetaPrime: an Iteractive Fault Tree Analyser. *IEEE Transactions on Reliability*, 43(1):121–127, March 1994.

[CPS98] M.C. Cheok, G.W. Parry, and R.R. Sherry. Use of importance measures in risk informed regulatory applications. *Reliability Engineering and System Safety*, 60:213–226, 1998.

[DR99] Y. Dutuit and A. Rauzy. New algorithms to compute importance factors CPr, MIF, CIF, DIF, RAW and RRW. In *Proceedings of the European Safety and Reliability Association Conference, ESREL'99*, volume 2, pages 1015–1020. A.A. Balkema, 1999. ISBN 90 5809 111 2.

[FFK88] M. Fujita, H. Fujisawa, and N. Kawato. Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'88*, pages 2–5, 1988.

[Fle96] Flemming. Developping useful insights and avoiding misleading conclusion from risk importance measures in psa applications. In *Proceedings of the Probabilistic Safety Assessment conference, PSA'96*, 1996.

[Fus75] J.B. Fussel. How to hand-calculate system reliability characteristics. *IEEE Transactions on Reliability*, R-24(3), 1975.

[FV72] J.B. Fussel and W.E. Vesely. A New Methodology for Obtaining Cut Sets for Fault Trees. *Trans. Am. Nucl. Soc.*, 15:262–263, June 1972.

[Gri86] W.S. Griffith. On consecutive k-out-of-n failure systems and their generalizations. *Reliability and Quality Control*, pages 157–165, 1986.

[HKL00] J.S. Hong, H.Y. Koo, and C.H. Lie. Computation of joint reliability importance of two gates in a fault tree. *Reliability Engineering and System Safety*, 68(1):1–5, 2000.

[HL93] J.S. Hong and C.H. Lie. Joint reliability importance of two edges in an undirected network. *IEEE Transaction on Reliability*, 42:17–23, 1993.

[HR94] A. Høyland and M. Rausand. *System Reliability Theory.* John Wiley & Sons, 1994. ISBN 0-471-59397.

[KKP97] I.N. Kovalenko, N.Y. Kuznetsov, and P.A. Pegg. *Mathematical Theory of Reliability of Time Dependent Systems with Practical Applications.* Wiley Series in Probability and Statistics. John Wiley & Sons, 1997. ISBN 0-471-95060-2.

[Lam75] H.E. Lambert. Measures of importance of events and cut sets in fault trees. In R.E. Barlow, J.B. Fussel, and N.D. Singpurwalla, editors, *Reliability and Fault Tree Analysis*, pages 77–100. SIAM Press, 1975.

[MKK99] M. Modarres, M. Kaminsky, and V. Krivstov. *Reliability Engineering and Risk Analysis.* Marcel Dekker, 1999. ISBN 0-8247-2000-8.

[Nat79] B. Natvig. A suggestion of a new measure of importance of system components. *Stochastic Processes and their Applications*, 9:319–330, 1979.

[Nat85] B. Natvig. New light on measures of importance of system components. *Scandinavian Journal of Statistics*, 12:43–52, 1985.

[Rau93] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, 05(59):203–211, 1993.

[Rau00] A. Rauzy. Towards an Efficient Implementation of Mocus, 2000. Technical note LaBRI/MVTSI/Aralia/NT00-9, submitted to IEEE Transactions on Reliability.

[Rud93] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'93*, pages 42–47, November 1993.

[SA96] R.M. Sinnamon and J.D. Andrews. Quantitative Fault Tree Analysis Using Binary Decision Diagrams. *Journal Européen des Systèmes Automatisés, RAIRO-APII-JESA*, 30:1051–1072, 1996. Special Issue on Binary Decision Diagrams.

[SA97] R.M. Sinnamon and J.D. Andrews. Improved Accuracy in Qualitative Fault Tree Analysis. *Quality and Reliability Engineering International*, 13:285–292, 1997.

[Ves96] W.E. Vesely. The use of risk importances for risk-based applications and risk-based regulation. In *Proceedings of the Probabilistic Safety Assessment conference, PSA'96*, 1996.

[Weg00] I. Wegener. *Branching Programs and Binary Decision Diagrams - Theory and Applications*. SIAM Monographs on Discrete Mathematics and Applications, 2000. ISBN 0-89871-458-3.

[WW96] I.B. Wall and D.H. Worledge. Some perspectives on risk importance measures. In *Proceedings of the international conference on Probabilistic Safety Assessment, PSA'96*, pages 203–207, 1996.