A New Methodology to Handle Boolean Models with Loops

A. Rauzy
IML, UPR CNRS 9016
163, avenue de Luminy – Case 907
F-13288 Marseille cedex 9 – FRANCE
arauzy@iml.univ-mrs.fr

Summary & Conclusions

The main Boolean risk assessment models (Fault Trees, Event Trees, Block Diagrams, Reliability Networks) can be seen as sets of Boolean equations. In general, they are hierarchical. In some cases however, the model contains loops, because the system embeds at least two components whose states depend one another. Reliability networks are a typical example of looped models. Classical fault tree assessment methods fail to assess this kind of model, at least without a costly preprocessing. In this article, we propose a logical framework to clarify the meaning looped of sets of Boolean equations. We propose also a Binary Decision Diagrams based method to assess them. We illustrate our approach by providing experimental results on a benchmark of reliability networks.

keywords Reliability networks, Boolean reliability models with loops, Binary Decision Diagrams

1 Introduction

The main Boolean risk assessment models (Fault Trees, Event Trees, Block Diagrams, Reliability Networks) can be seen as sets (conjuncts) of Boolean equations in the form $g \Leftrightarrow F$, where g is a Boolean variable and F is a Boolean formula. These equations describe the global state of the system as a (Boolean) function of the states of its basic components. In such a model, there are two kinds of variables:

- Input variables that occur only in right members of equations. These variables describe states of basic components.
- Gate variables that occur as the left member of an equation. These variables represent functionalities of the system or states of non-basic components.

Fault trees, event trees and block diagrams are built by means of a hierarchical decomposition of the system under study. The corresponding sets of Boolean equations are hierachical and the values of gate variables are uniquely determined by the values of input variables. Sometimes components interact in such way that no hierachical description is suitable. This is for instance the case if two components A and B are such that the state of A depends on the state of B and vice versa. Reliability networks [1] examplify this general problem. The sets of Boolean equations that describe this kind of systems may contain loops, i.e. circular definitions such as $g_1 \Leftrightarrow F_1(g_2), g_2 \Leftrightarrow F_2(g_3), \ldots, g_r \Leftrightarrow F_r(g_1)$. In this case, the values of input variables do not determine uniquely the values of gate variables. However, this underterminism does not come from the physics. It comes from the model. The physics "prefers" one of the possible valuations. Consider, for instance, to an electric circuit with no power source. The equations may let the circuit components either powered or not powered. The physics tells us they are not powered. The problem comes from the fact that there is no tractable logical mean to express preferences.

In this article, we propose a formal framework to tackle this difficulty.

We show that, in some restricted cases, preferences can be expressed by logical means only, through the use of quantified formulae. This clarifies and generalizes an idea by Madre et al [2].

We establish a decomposition theorem that makes it possible to assess looped models by means of Binary Decision Diagrams (BDDs) [3, 4]. BDDs are the state-of-the-art data structure to handle boolean functions [5]. Since their introduction in the reliability field [6, 7], they have proved to be the most efficient tool to assess Boolean reliability models such as fault trees. One of the key issues to take fully advantage of BDDs is to find a good variable ordering. The size of BDDs, and therefore the efficiency of the whole methodology, depends dramatically on the chosen ordering. Several domain dependent heuristics have been proposed for circuits (see for instance [8, 9, 10]) and fault trees (see for instance [11, 12]). We propose a heuristic devoted to the assessment of sets of equations. Experimental results on a benchmark of reliability networks give evidence of the interest of the whole approach. Our algorithm gives better results than the specialized BDD method proposed recently by Kuo, Lu and Yeh [13].

The remainder of this article is organized as follows. First, we introduce quantified Boolean formulae and sets of equations section 2. The general framework we propose to deal with such models is presented section 3. In section 4, we recall basics about BDDs and we discuss the variable ordering problem. Experimental results are reported section 5. Finally, related works are examined section 6.

2 Boolean Models

2.1 Quantified Boolean Formulae

In this article, we consider Boolean formulae built over the two (Boolean) constants 0 (False) and 1 (True), a denumerable set of variables $\{v_1, v_2, \ldots\}$, and the usual logical

connectives "." (and), "+" (or), "$^-$" (not). The set of variables that occur in the formula F is denoted by $var(F)$.

For the sake of the convenience, we shall use the connectives $\Rightarrow$ and $\Leftrightarrow$ that are defined as follows.

$$F \Rightarrow G \stackrel{\text{def}}{=} \overline{F} + G \qquad F \Leftrightarrow G \stackrel{\text{def}}{=} F.G + \overline{F}.\overline{G}$$

Let F be a formula and v a variable. We denote by $F[1/v]$ (respectively $F[0/v]$) the formula obtained by substituting in F the constant 1 (respectively 0) for the variable v. Consider, for instance, the formula $F = ab + \overline{a}c$. Then, $F[1/a] = 1b + \overline{1}c = b$ and $F[0/a] = 0b + \overline{0}c = c$. We denote by $F[c_1/v_1, \ldots, c_k/v_k]$ the formula $F[c_1/v_1] \ldots [c_k/v_k]$.

In the above example, we use implicitely Boolean simplification rules (constant propagation): for any formula F, $\overline{1} = 0$, $\overline{0} = 1$, $F.0 = 0.F = 0$, $F.1 = 1.F = F$, $F+0 = 0+F = F$, $F+1 = 1+F = 1$, ... Throughout this article, we shall keep implicit such simplifications.

The treatment of Boolean models with loops introduces formulae with the universal quantifier $\forall$ and the existential quantifier $\exists$. They are defined as follows.

$$\forall v\ F \stackrel{\text{def}}{=} F[1/v].F[0/v] \qquad \exists v\ F \stackrel{\text{def}}{=} F[1/v] + F[0/v]$$

Consider, for instance, the formula $F = ab + \overline{a}c$. Then, $\forall a F = bc$ and $\exists a F = b + c$. We denote by $Qv_1, \ldots, v_k\ F$, $Q \in \{\forall, \exists\}$, the formula $Qv_1\ (Qv_2 \ldots (Qv_k\ F) \ldots)$.

2.2 Boolean Equations

A Boolean equation is a formula in the form $v \Leftrightarrow F$, where v is a variable and F is a formula. A set of Boolean equations E is assimilated with the conjunct of its elements. It is assumed that for each variable v there is at most one equation $v \Leftrightarrow F$ in E. If such a equation belongs to E, F is called the definition of v. It is moreover assumed that $v \notin var(F)$.

We call (i) output-, (ii) input- and (iii) gate-variables of E the variables that respectively (i) occur as a left member of an equation and do not occur in a right member, (ii) do not occur as a left member and (iii) occur as a left member.

In reliability models, input variables represent in general events that change the internal state of basic components. Input variables are typically the basic events of fault trees or represent failures of edges and nodes in reliability networks. Gate variables represent functionalities of the system or states of non-basic components. Gates variables are typically gates of fault trees or used to describe whether a node of a network is feeded.

We define $ancestors_E(v)$ as follows.

$$ancestors_E(v) \stackrel{\text{def}}{=} \begin{cases} var(F) \cup \bigcup_{w \in var(F)} ancestors_E(w) & \text{if } E \ni v \Leftrightarrow F \\ \emptyset & \text{otherwise} \end{cases}$$

In other words, $ancestors_E(v)$ is the set of variables w such that v depends on w.

A set of Boolean equations E is said looped if there is a variable v such that $v \in ancestors_E(v)$. It is said hierarchical otherwise.

Fault trees are hierarchical formulae. It is worth noticing that any hierarchical set of equations with a single ouput variable r can be rewritten into a equivalent equation $r \Leftrightarrow F$ by replacing bottom-up the other gate variables by their definitions.

2.3 Literals, Products, Minterms, Prime Implicants

A literal is either a variable v or its negation $\overline{v}$. v is a positive literal. $\overline{v}$ is a negative literal. They are said opposite. The opposite of a literal p is denoted by $\overline{p}$ ($\overline{\overline{p}} = p$).

A product is a set of literals that does not contain both a literal and its opposite. A product is assimilated with the conjunct of its elements.

Let $\mathcal{V}$ be a finite set of variables. A product that contains a literal built over each variable of $\mathcal{V}$ is called a minterm of $\mathcal{V}$. We denote by $minterms(\mathcal{V})$ the set of minterms that can be built over $\mathcal{V}$. Any formula can rewritten as a disjunct of minterms. This disjunct of mintems is unique (up to a permutation). E.g. $F = ab + \overline{a}c = abc + ab\overline{c} + \overline{a}bc + \overline{a}\,\overline{b}c$. It is therefore often convenient to write that a minterm π belongs to a formula F ($\pi \in F$) when F can be rewritten as $\pi + F'$. Note that for any minterm π either $\pi \in F$ or $\pi \in \overline{F}$.

A product π is an implicant of a formula F if, for any $\sigma \in minterms(var(F))$, if $\pi \subseteq \sigma$, then $\sigma \in F$. An implicant π is said prime if there is no implicant ρ of F such that $\rho \subset \pi$. We denote by $PI[F]$ the set of prime implicants of the formula F.

The formula $F = ab + \overline{a}c$ admits 7 implicants ab, abc, $ab\overline{c}$, $\overline{a}bc$, $\overline{a}c$, $\overline{a}\,\overline{b}c$ and bc and 3 prime implicants ab and $\overline{a}c$, bc.

Prime implicants play a central role in reliability studies. They represent minimal *scenarii* of failure in fault trees (minimal cutsets), minimal s-t connecting paths in reliability networks, ... Note however that there are slight differences between the the notions of prime implicants and minimal cutsets as we showed in [14]. These differences do not matter for the purpose of this article.

2.4 Order

In Boolean risk assessment models, positive literals represent in general the relevant facts (failures of components), while negative literals represent in some sense nominal situations. It is therefore natural to introduce an order among literals.

By convention, we shall consider that the negative literal is smaller than the positive literal, which is denoted by $\overline{e} \sqsubset e$. Given two literals p and q, $p \sqsubseteq q$ if either $p = q$ or $p \sqsubset q$. This order can be extended into a partial order $\sqsubseteq$ over minterms: $p_1 . \ldots . p_n \sqsubseteq q_1 . \ldots . q_n$ if $p_i \sqsubseteq q_i$ for $i = 1, \ldots, n$.

A formula F is monotone increasing if for any two minterms π and ρ such that $\pi \sqsubseteq \rho$, then $\pi \in F$ implies that $\rho \in F$. Coherent fault trees are monotone increasing formulae. A formula F is monotone decreasing if for any two minterms π and ρ such that $\pi \sqsubseteq \rho$, then $\rho \in F$ implies that $\pi \in F$.

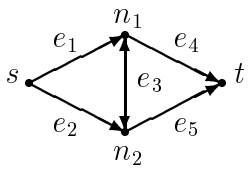

Figure 1: The bridge network.

3 A Logical Framework to Handle Looped Models

3.1 Introductory Example

Consider for instance the network pictured Fig. 1. Assume that all nodes but s are perfectly reliable and that s and all edges may fail independently. The set of Boolean equations that describe working s,t-paths of this network could be as follows.

$$\begin{aligned}
\mathsf{pwr}(n_1) &\Leftrightarrow \mathsf{pwr}(s).\mathsf{wrk}(e_1) + \mathsf{pwr}(n_2).\mathsf{wrk}(e_3) \\
\mathsf{pwr}(n_2) &\Leftrightarrow \mathsf{pwr}(s).\mathsf{wrk}(e_2) + \mathsf{pwr}(n_1).\mathsf{wrk}(e_3) \\
\mathsf{pwr}(t) &\Leftrightarrow \mathsf{pwr}(n_1).\mathsf{wrk}(e_4) + \mathsf{pwr}(n_2).\mathsf{wrk}(e_5)
\end{aligned}$$

Each $\mathsf{wrk}(e)$ represents the event "the edge e is working". Each $\mathsf{pwr}(v)$ represents the event "the vertex v is powered". $\mathsf{pwr}(s)$ and the $\mathsf{wrk}(e)$'s are input variables. $\mathsf{pwr}(n_1)$, $\mathsf{pwr}(n_2)$ and $\mathsf{pwr}(t)$ are gate variables. $\mathsf{pwr}(t)$ represents the question we are asking about the network, namely "is the target state powered ?".

The conjunct E of the above equations describes, in a natural way, the physics of the network. It is looped since $\mathsf{pwr}(n_1)$ depends on $\mathsf{pwr}(n_2)$ and vice-versa. This has two consequences. First, $\mathsf{pwr}(n_1)$ and $\mathsf{pwr}(n_2)$ are not gates in the usual sense. It follows that classical fault tree assessment techniques cannot be used to assess E, at least without a preprocessing. Second, the values of the $\mathsf{pwr}(v)$'s ($v \in \{n_1, n_2, t\}$) are not uniquely determined by the values of $\mathsf{pwr}(s)$ and the $\mathsf{wrk}(x)$'s. Consider, for instance, the case where vertex s and edges e_1 and e_2 are down while edges e_3, e_4 and e_5 are working. The formula

$$E[0/\mathsf{pwr}(s), 0/\mathsf{wrk}(e_1), 0/\mathsf{wrk}(e_2), 1/\mathsf{wrk}(e_3), 1/\mathsf{wrk}(e_4), 1/\mathsf{wrk}(e_5)]$$

can be simplified into the following set of equations.

$$\begin{aligned}
&\mathsf{pwr}(n_1) \Leftrightarrow \mathsf{pwr}(n_2) \\
&\mathsf{pwr}(t) \Leftrightarrow \mathsf{pwr}(n_1)
\end{aligned}$$

Two valuations satisfy this set: $\mathsf{pwr}(n_1) = \mathsf{pwr}(n_2) = \mathsf{pwr}(t) = 0$ which corresponds to the physics and $\mathsf{pwr}(n_1) = \mathsf{pwr}(n_2) = \mathsf{pwr}(t) = 1$.

This example illustrate a general problem: the model is looped, therefore underterministic. However, this underterminism does not come from the physics. It comes from the model. The physics has a prefered interpretation. This prefered interpretation is smallest

one with respect to the order $\sqsubset$ we defined section 2.4. The main problem when dealing with looped system is to eliminate unwanted, i.e. non-minimal, interpretations.

3.2 Framework

The problem of looped systems can be stated in the following way.

Let $E = \Pi_{i=1}^{m} (g_i \Leftrightarrow F_i)$ be a set (a conjunct) of Boolean equations that describe the physics of the system under study. E is built over two distinct sets of variables: the set $\mathcal{G} = \{g_1, \ldots, g_m\}$ of gate variables and the set $\mathcal{E} = \{e_1, \ldots, e_n\}$ of input variables.

Two generic questions may be answered about E. They can be stated as follows.

(A) What are the minimal sets of basic events that lead the system in a configuration in which a given property P is realized ?

(B) What is the probability that at least one of these configurations is realized ?

We have first to set in a formal way which valuations of the variables correspond to the physics. Consider the minterms $\pi\rho$ over $\mathcal{E} \cup \mathcal{G}$, where π is a minterm over $\mathcal{E}$ and ρ is a minterm over $\mathcal{G}$. $\pi\rho$ obeys the physics if it fulfills the following requirements.

1. $\pi\rho$ verifies the equations of the model, i.e. $\pi\rho \in E$.
2. $\pi\rho$ is minimal w.r.t. $\sqsubset$, i.e. $\forall \rho' \in minterms(\mathcal{G}),\ \rho' \sqsubset \rho \ \Rightarrow\ \pi\rho' \in \overline{E}$.

Therefore, we can defined the set M_E of the minterms that obey the physics as follows.

$$\begin{aligned} M_E \ \stackrel{\text{def}}{=}\ & \{\pi\rho, \pi \in minterms(\mathcal{E}), \rho \in minterms(\mathcal{G}); \\ & \qquad \pi\rho \in E \ \wedge\ \forall \rho' \in minterms(\mathcal{G}),\ \rho' \sqsubset \rho \ \Rightarrow\ \pi\rho' \in \overline{E}\} \end{aligned} \tag{1}$$

A set of equations E is correct if, for each configuration of the basic events, it exists a unique configuration of the other events that obeys the physics. Formally,

$$\begin{aligned} & \forall \pi \in minterms(\mathcal{E}), \\ & \qquad \exists \rho \in minterms(\mathcal{G})\ \pi\rho \in M_E \\ & \quad \wedge\ \ \forall \rho, \rho' \in minterms(\mathcal{G})\ \pi\rho \in M_E \wedge \pi\rho' \in M_E \ \Rightarrow\ \rho = \rho' \end{aligned} \tag{2}$$

In the sequel, we shall consider only correct sets of equations. It is worth noticing that M_E acts as a function from $minterms(\mathcal{E})$ to $minterms(\mathcal{G})$. We denote by $M_E(\pi)$, $\pi \in minterms(\mathcal{E})$ the unique $\rho \in minterms(\mathcal{G})$ such that $\pi\rho \in M_E$. The following lemma holds.

Lemma 1 *Let $\pi \in minterms(\mathcal{E})$ and let $\rho \in minterms(\mathcal{G})$ such that $\pi\rho \in E$. Then, $M_E(\pi) \sqsubseteq \rho$.*

This follows immediately from the uniqueness of $M_E(\pi)$. $\square$

The minterms that obeys the physics for the bridge network are given in appendix A.1.

To answer the generic questions (A) and (B), the idea is to build a formula Q over $\mathcal{E}$ such that a minterm π over $\mathcal{E}$ belongs to Q if and only if it obeys the physics and it verifies the property P. Formally, Q is defined as follows.

$$Q \stackrel{\text{def}}{=} \{\pi \in minterms(\mathcal{E}); \exists \rho \in minterms(\mathcal{G}),\ \pi\rho \in M_E.P\} \tag{3}$$

The prime implicants of Q and its probability are respectively the minimal set of events and the probability we are looking for. The problem is to make concrete the construction of such a formula Q.

3.3 The Case of Monotone Properties

In the case where P is a monotone formula, the query Q can be written in a purely logical way, as stated by the two following theorems.

Theorem 2 (Monotone decreasing property) *If P is a monotone decreasing formula then Q is equivalent to the formula Q_{MD} defined as follows.*

$$Q_{MD} \stackrel{\text{def}}{=} \{\pi \in minterms(\mathcal{E}); \exists \rho \in minterms(\mathcal{G})\ \pi\rho \in E.P\} \tag{4}$$

$Q \Rightarrow Q_{MD}$. By definition.
$Q_{MD} \Rightarrow Q$. Consider a minterm $\pi \in Q_{MD}$. Let $\rho_0 = M_E(\pi)$ and assume $\pi\rho_0 \notin P$. Let ρ be one of the minterms over $\mathcal{G}$ such that $\pi\rho \in E.P$. By lemma 1, $\rho_0 \sqsubset \rho$. Since P is monotone decreasing, $\pi\rho_0 \in P$. A contradiction. $\square$

The theorem 2 can be used to get the s,t disconnecting cuts of a reliability network. The case of the bridge network is treated in appendix A.2.

Theorem 3 (Monotone increasing property) *If P is a monotone increasing formula then Q is equivalent to the formula Q_{MI} defined as follows.*

$$Q_{MI} \stackrel{\text{def}}{=} \{\pi \in minterms(\mathcal{E}); \forall \rho \in minterms(\mathcal{G})\ \pi\rho \in E \Rightarrow \pi\rho \in P\} \tag{5}$$

$Q \Rightarrow Q_{MI}$. Let $\pi \in Q$ and let $\rho = M_E(\pi)$. By lemma 1, forall $\rho' \in minterms(\mathcal{G})$ either $\rho' \notin E$ or $\rho \sqsubseteq \rho'$. In the latter case, since P is monotone increasing, $\rho' \in P$.
$Q_{MI} \Rightarrow Q$. By definition. $\square$

The theorem 3 can be used to get the s,t-connecting paths of a reliability network. The case of the bridge network is treated in appendix A.3.

The formula Q_{MI} is, up to slight differences, the one proposed by Madre et al to compile digraphs [2]. The theorem 3 provides a formal framework to understand why Madre's method works.

3.4 Decomposition Theorem

Preferences induced by the partial order $\sqsubset$ cannot be expressed in a purely logical way, i.e. using only the usual connectives and possibly some quantifiers. We will discuss why in the next section. In other words, the formula M_E cannot be derived by purely logical means from the formula E. *A fortiori* the query Q cannot be derived from E and P if no assumption is made on P. This does not mean however that the formula M_E cannot be computed from the formula E. In this section, we show how to compute Q from E and P in the Binary Decision Diagram framework.

BDDs make an extensive use of the Shannon decomposition: Let F be a formula and let v be a variable of F, then the following equality holds.

$$F \quad = \quad v.F[1/v] + \overline{v}.F[0/v] \tag{6}$$

The Shannon decomposition can be mixed together other operations in order to produce so-called decomposition theorems. Let, for instance, $F = v.F_1 + \overline{v}.F_0$ and $G = v.G_1 + \overline{v}.G_0$ be two formulae. Then $F \odot G = v.(F_1 \odot G_1) + \overline{v}.(F_0 \odot G_0)$ for any binary connective $\odot$ ($+$, $.$, $\Rightarrow$, $\Leftrightarrow$, ...). Most of BDD algorithm rely on such decomposition theorems that make it possible to compute recursively a given quantity or function.

We will give a decomposition theorem to compute Q from E and P. The computation of Q consists in three steps:

1. One computes $Q_0 = E.P$.
2. One removes from Q_0 the minterms $\pi\rho$ for which there exists a minterm ρ' such that $\rho \sqsubset \rho'$ and $\pi\rho' \in E$. Let Q_1 be the result.
3. One quantifies existentially the $\mathcal{G}$'s in Q_1 to get Q.

As we will see, the second and third steps are actually achieved at once.

Let F and G be two formulae built over $\mathcal{E} \cup \mathcal{G}$. The formulae $F \otimes_{\mathcal{G}} G$ and $F \downarrow_{\mathcal{G}} G$ are defined as follows.

$$\begin{aligned} F \otimes_{\mathcal{G}} G \quad &\stackrel{\text{def}}{=} \quad \{\pi\rho; \pi \in minterms(\mathcal{E}), \rho \in minterms(\mathcal{G}), \\ &\qquad \pi\rho \in F \ \wedge \forall \rho' \in minterms(\mathcal{G}),\ \rho' \sqsubset \rho \ \Rightarrow \pi\rho' \in G\} \end{aligned} \tag{7}$$

$$F \downarrow_{\mathcal{G}} G \quad \stackrel{\text{def}}{=} \quad \exists \mathcal{G}\ (F \otimes_{\mathcal{G}} G) \tag{8}$$

We can now state the decomposition theorem.

Theorem 4 (Decomposition of $F \downarrow_{\mathcal{U}} G$) *Let $v \in \mathcal{E} \cup \mathcal{G}$ and let $F = v.F_1 + \overline{v}.F_0$ and $G = v.G_1 + \overline{v}.G_0$ be two formulae built over $\mathcal{E} \cup \mathcal{G}$. The following equality holds.*

$$F \downarrow_{\mathcal{G}} G \quad = \quad \begin{cases} v.(F_1 \downarrow_{\mathcal{G}} G_1) + \overline{v}.(F_0 \downarrow_{\mathcal{G}} G_0) & \text{if } v \in \mathcal{E} \\ (F_1 \downarrow_{\mathcal{G}} G_1).G_0 + (F_0 \downarrow_{\mathcal{G}} G_0) & \text{if } v \in \mathcal{G} \end{cases}$$

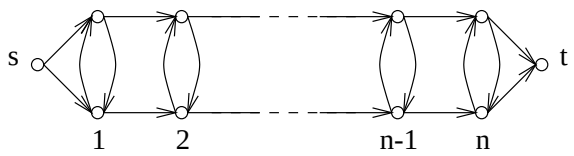

Figure 2: A network with $2n+2$ vertices, $4n+2$ edges and 2^n s,t-paths.

The proof is a straightforward application of the definitions 7 and 8.

The following corollary, that follows from the definition 3, gives a mean to compute Q.

Corollary 5 (Computation of Q) *The following equivalence holds.*

$$Q \quad = \quad E.P \downarrow_{\mathcal{G}} \overline{E}$$

3.5 Discussion

In order to remove undesirable minterms, we have used so far extra-logical means (via the partial order $\sqsubset$). It is questionable whether it is possible to handle looped descriptions by means of purely logical expressions.

In the case of reliability networks, it is possible to write a formula F that enumerates s,t-paths, therefore avoiding all difficulties. However, enumerating good configurations (s-t paths) has a serious drawback: the number of such configurations may be exponentially larger than the number of components of the system under study. The family of networks pictured Fig. 2 is a good witness of this problem.

Unfortunately a method to describe s,t-paths locally cannot exist. Let $\vec{a_i}$ denotes the nodes and edges that are adjacent to the vertex v_i. Assume there exist formulae $R_i(v_i, \vec{a_i})$ (one per vertex) such that the conjunct of the R_i's express that each vertex v_i is reachable from the source vertex s. It is clear that to determine s,t-paths one needs a formula of this kind. Now, it should be possible to standardize the R_i's into a single first-order predicate $R(\vec{v})$. The number of parameters of R depends only on the maximal degree d of the graph. By quantifying universally the conjunct of the applications of R to each vertex of the network, one gets a first order formula that is a tautology if and only if each vertex of the network is reachable from the source vertex. Moreover, R could be used for any network of maximal degree less than d. But, it is a central result of descriptive complexity theory that such a formula cannot exist. This is a consequence of the Löwenheim-Skolem theorem that asserts that any formula with a arbitrary large model has a infinite model (see for instance [15] for a simple demonstration, and [16] for a monograph on descriptive complexity).

The use of preferences is thus the price to pay to write local descriptions, i.e. descriptions such that the status of a node depends only on the status of nodes and edges of its immediate neighbourhood.

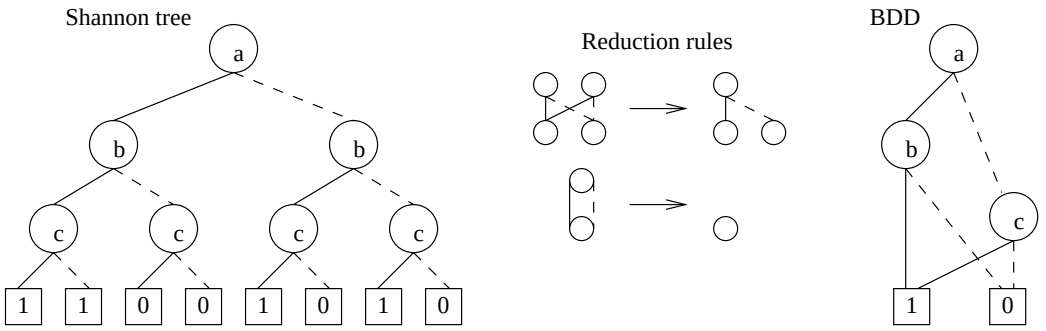

Figure 3: From the Shannon tree to the BDD encoding $ab + \bar{a}c$.

4 Binary Decision Diagrams

In this section, we recall basics about BDDs. The reader interested in a more detailed presentation should see the reference [4].

4.1 BDDs

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 (equation 6). 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. Each internal node encodes a formula F. It is labeled with a variable v and has two outedges (a *then*-outedge that points to the node that encodes $F[1/v]$, and a *else*-outedge that points 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 valuation of the variables is obtained by descending along the corresponding branch of the tree. The Shannon tree for the formula $ab + \bar{a}c$ and the lexicographic order is pictured Fig. 3 (dashed lines represent *else*-outedges).

Indeed, such a representation is 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 unique son ($v.F + \bar{v}.F = F$).

By applying these two rules as far as possible, one gets the BDD associated with the formula. A BDD is therefore a directed acyclic graph. It is unique, up to an isomorphism [3]. This process is illustrated on Fig. 3.

Logical operations (and, or, not, quantifications, ...) can be directly performed on BDDs. This results from the orthogonality of usual connectives and the Shannon decomposition. The complete binary tree is never built and then shrunk: the BDD that encodes

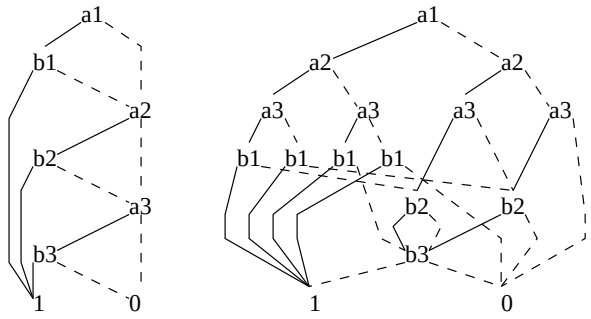

Figure 4: The BDDs encoding $a_1.b_1 + a_2.b_2 + a_3.b_3$ for two variable orderings.

a formula is obtained by composing the BDDs that encode its subformulae. Moreover, a caching principle is used to store intermediate results of computations. This makes the usual logical operations (conjunction, disjunction, quantification) polynomial in the sizes of their operands. A complete implementation of a BDD package is described in [4]. The reader interested in details should thus refer to this article.

The theorem 4 applies directly on BDDs (including caching). It can be used to compute the BDD that encodes the minterms that corresponds to the physics of a given set of equations.

4.2 The Variable Ordering Problem

One of the key issues to take fully advantage of BDDs is to find a good variable ordering. The size of BDDs, and therefore the efficiency of the whole methodology, depends dramatically on the chosen ordering. The following example, already given in [3], illustrates the problem. Let F_n be the following parametric formula $F_n = a_1.b_1 + a_2.b_2 + \ldots + a_n.b_n$. The size of the BDD that encodes F_n is linear in n for the ordering $a_1 < b_1 < \ldots < a_n < b_n$, while it is exponential for the ordering $a_1 < \ldots < a_n < b_1 < \ldots < b_n$. The BDDs for the two variable orderings and $n = 3$ are pictured on Fig. 4.

Finding a good variable ordering is a difficult problem. The only means to predict the size of a BDD is more or less to build it (see for instance [17] for theoretical insights about this question). Several domain dependent heuristics have been proposed for circuits (see for instance [8, 9, 10]) and fault trees (see for instance [11, 12]). These heuristics rely on different principles. However, they both try to put close in the order the variables that are close in the formula, as illustrated by the a_i/b_i of the previous example.

4.3 Heuristics for Sets of Equations

The problem of finding a good heuristics is therefore twofold. First, one has to define formally the notion of proximity between variables. Second, once such a formal model is found, one has to determine which ordering is optimum.

In the case of sets of equations, we can assume that right members of equations are small formulae. Therefore, we can ignore what happens inside each equation. A formal way to define the proximity is thus to associate an undirected graph G_E with the set of equations $E(\mathcal{E}, \mathcal{G})$ under study. The vertices of G are the variables of E. G contains an edge for each pair (v, w) of variables such that v is left member of an equation $v \Leftrightarrow F$ and w occurs in F. An alternative could be to create an edge for each pair of variables occurring in the equation.

An ordering I associates an index (a positive integer) $I(v)$ with each variable v. $I(v) \in [0, m+n]$ where m and n denote respectively the number of variables of $\mathcal{E}$ and $\mathcal{G}$. The weight $\omega(I)$ of an ordering I is defined as follows.

$$\omega(I) \stackrel{\text{def}}{=} \sum_{(v,w)\in G_E} |I(v) - I(w)| \tag{9}$$

It is expected that the best ordering is the one of minimum weight.

Unfortunately, the problem of determining which ordering minimizes ω is NP-Complete (it is referenced as "Optimal Linear Arrangement" in [18]). It is actually intractable to get the best solution as soon as G_E contains more than few dozens of variables.

Fortunately, approximate solutions are relatively easy to find by means of local search algorithms. We use the following method.

1. Start with a presumably good ordering. In the case of fault trees, the ordering obtained by means of a depth-first left-most traversal of the tree is often good. In the case of reliability networks, the ordering obtained by means of a breadth-first traversal of the network starting from its source is a good candidate.

2. Try to improve the current order by flipping two consecutive variables. The variable to flip is either the one that improve the most the weight or, in case of tie, it is picked at random among those that give the best weight improvement. This process is reiterated until there is no mean to improve further the current ordering or a predefined number of sideway moves is reached.

Indeed, many variations can be imagined on this scheme (see for instance [19, 20] for reviews of recent developments on local search algorithms). The main interest of the method we propose is that in practice it leads quickly to a good solution.

5 Experimental Results

In order to test our method (treatment of looped systems of equations plus variable ordering heuristics), we considered the benchmark of networks collected in literature by Kuo, Lu and Yeh. In [13], these authors propose a BDD based approach to compute the terminal-pair reliability of a network. Their method relies on a recursive decomposition of the network (and uses specific data structures in addition to BDDs). They provide a number

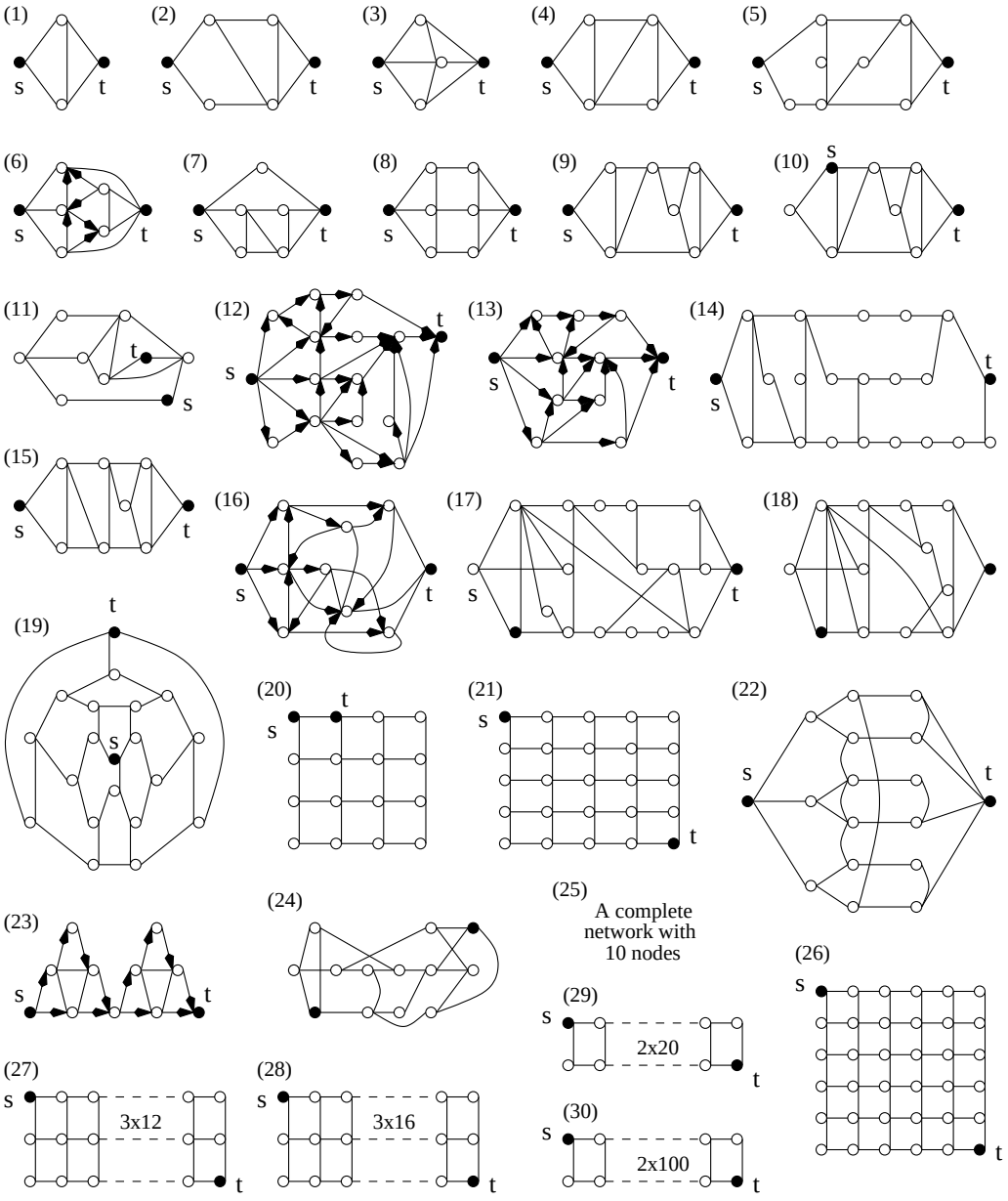

Figure 5: Benchmark networks #1-#30

of experimental results on the networks pictured on Fig. 5 (most of them are taken from [21] and [22]).

We wrote sets of equations for these networks in the same way we did for the bridge network in section 3.1. Then, we applied the variable ordering heuristics described in section 4.3. Finally, we computed the BDD that encodes the query as defined in section 3, the ZBDD that encodes the prime implicants of the query (with the algorithm proposed in [7]) and the probability there is an operating s-t paths (edge reliabilities are all set to 0.9).

The table 1 reports the results we obtained and compares them with Kuo, Lu and Yeh results. The running time for the formers are those for the computation of BDD plus the computation of the ZBDD plus the assessment of the reliability. For the latters, the table gives the running time to compute the BDD that encodes the s-t paths as well as the size of this BDD for the two best methods proposed in the article, namely EED_BFS and EED_SIFT. EED_BFS uses a breadth-first variable ordering heuristics. EED_SIFT applies the sifting post-processing to reduce the sizes of BDDs [23].

It is worth noticing that running times were measured on different processors (although roughly equivalent).

The following conclusions can be drawn from these results.

- Our method performs well both in terms of running times and in terms of BDD sizes compared to the algorithm by Kuo, Lu and Yeh.
- It is much more general for it works on any set of equations and it makes it possible to answer much more general questions about the models.

6 Related Works and Conclusions

The framework we proposed in this article generalizes and explains the work by Madre et al. [2]. The heuristics is also a formalization and a generalization of our own preliminary works on the topics [24]. Our method has three main interests:

- It is powerful since it can be applied to a large class of Boolean models that includes at least both fault trees and reliability networks.
- The models and the queries are very easy to write. They are obtained from local behavior descriptions only. Their size is linear in the size of the studied system. This makes a definitive difference with other techniques to deal with looped models that are either uncomplete or of an exponential worst case complexity (which is for instance the case of those proposed in [25, 26]).
- It can efficiently implemented by means of Binary Decision Diagrams. There is no price to pay for the generality of the method: it is efficient even compared to the specialized algorithm by Kuo, Lu and Yeh [13].

Table 1: Results on the benchmark networks

			EED_BFS		EED_SIFT		Our method		
net	#paths	reliability	$\|BDD\|$	time	$\|BDD\|$	time	$\|BDD\|$	$\|ZBDD\|$	time
01	4	0.978480	10	0.00	9	0.00	9	7	0.00
02	7	0.968425	15	0.00	15	0.02	20	13	0.00
03	9	0.997632	26	0.00	24	0.02	24	14	0.00
04	13	0.977184	22	0.00	17	0.00	17	15	0.00
05	13	0.964855	50	0.00	23	0.00	29	20	0.01
06	14	0.996664	39	0.00	32	0.02	42	20	0.00
07	25	0.997494	51	0.00	45	0.02	57	29	0.01
08	29	0.996217	66	0.00	50	0.03	53	21	0.00
09	24	0.975116	36	0.00	27	0.00	32	24	0.00
10	20	0.984068	68	0.00	27	0.02	33	22	0.01
11	18	0.969112	48	0.00	41	0.02	42	28	0.01
12	36	0.997186	548	0.03	129	0.03	112	58	0.01
13	18	0.994076	157	0.00	43	0.03	135	39	0.01
14	44	0.904577	126	0.00	64	0.05	97	51	0.01
15	44	0.974145	40	0.00	36	0.02	43	29	0.00
16	64	0.997506	407	0.02	134	0.13	135	68	0.01
17	145	0.985357	643	0.05	153	0.27	274	112	0.01
18	269	0.987310	292	0.03	134	0.15	136	84	0.01
19	780	0.997120	3591	0.35	2182	3.38	2032	450	0.20
20	98	0.987831	177	0.02	128	0.01	178	100	0.02
21	8512	0.975557	1148	0.43	1111	4.03	1148	705	0.17
22*	192	0.998171	1505	0.08	886	0.77	386	137	0.02
23	100	0.959624	46	0.00	35	0.03	45	33	0.00
24	102	0.995447	250	0.03	139	0.13	161	77	0.01
25	109601	1.000000	49785	14.10	37371	142.75	49785	5828	5.43
26	1262816	0.975645	4970	15.75	4863	148.63	4970	3153	0.35
27	538020	0.961730	317	3.65	307	5.06	316	199	0.02
28	64019918	0.956266	437	70.73	427	82.42	436	275	0.03
29	524288	0.784482	115	0.02	114	0.20	114	76	0.01
30	2^{99}	0.304293	595	2.52	594	10.25	594	396	0.27

The notion of preference has been already used in different contexts. In artificial intelligence and data bases, the close world assumption (all what was not told true should be considered as false) can be interpreted in terms of preferences. In [27], Besnard and Siegel compared several non-monotonic logics through the notion of preferential models. Our treatment of looped system can be interpreted in their framework. More recently, we used the partial order $\sqsubset$ to give a clear algebraic interpretation for the notion of minimal cutsets [14].

The present work is also related with, although different from, algorithms proposed to remove loops in sequential circuits [28, 29, 30]. It would be interesting to study whether techniques such as those proposed in [30] can be used to perform a more efficient compilation (i.e. to generate a formula that can more easily handled by means of BDD's).

References

[1] D. Shier, *Network Reliability and Algebraic Structures*. Oxford Science Publications, 1991.

[2] J.-C. Madre, O. Coudert, H. Fraïssé, and M. Bouissou, "Application of a New Logically Complete ATMS to Digraph and Network-Connectivity Analysis," in *Proceedings of the Annual Reliability and Maintainability Symposium, ARMS'94*, pp. 118–123, 1994. Annaheim, California.

[3] R. Bryant, "Graph Based Algorithms for Boolean Fonction Manipulation," *IEEE Transactions on Computers*, vol. 35, pp. 677–691, August 1986.

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

[5] R. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams," *ACM Computing Surveys*, vol. 24, pp. 293–318, September 1992.

[6] O. Coudert and J.-C. Madre, "A New Method to Compute Prime and Essential Prime Implicants of Boolean Functions," in *Advanced Research in VLSI and Parallel Systems* (T. Knight and J. Savage, eds.), pp. 113–128, March 1992.

[7] A. Rauzy, "New Algorithms for Fault Trees Analysis," *Reliability Engineering & System Safety*, vol. 05, no. 59, pp. 203–211, 1993.

[8] 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*, pp. 2–5, 1988.

[9] H. Cho, G. Hatchel, S. Jeong, B. Plessier, E. Swharz, and F. Somenzi, "ATPG Aspect of FSM Verification," in *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'90*, November 1990.

[10] K. Butler, D. Ross, R. Kapur, and M. Mercer, "Heuristics to Compute Variable Orderings for Efficient Manipulation of Ordered BDDs," in *Proceedings of the 28th Design Automation Conference, DAC'91*, June 1991. San Francisco, California.

[11] M. Bouissou, F. Bruyère, and A. Rauzy, "BDD based Fault-Tree Processing: A Comparison of Variable Ordering Heuristics," in *Proceedings of European Safety and Reliability Association Conference, ESREL'97* (C. G. Soares, ed.), vol. 3, pp. 2045–2052, Pergamon, 1997. ISBN 0-08-042835-5.

[12] J. Andrews and L. Barlett, "Efficient Basic Event Orderings for Binary Decision Diagrams," in *Proceedings of the Annual Reliability and Maintainability Symposium, ARMS'98*, pp. 61–67, 1998. ISSN 0149-144X.

[13] S.-Y. Kuo, S.-K. Lu, and F.-M. Yeh, "Determining Terminal-Pair Reliability Based on Eedge Expansion Diagrams Using OBDD," *IEEE Transactions on Reliability*, vol. 48, pp. 234–246, September 1999.

[14] A. Rauzy, "Mathematical Foundation of Minimal Cutsets," *IEEE Transactions on Reliability*, 2000. to appear.

[15] C. Papadimitriou, *Computational Complexity*. Addison Wesley, 1994. ISBN 0-201-53082-1.

[16] N. Immerman, *Descriptive Complexity*. Springer Verlag, 1998. ISBN 0387-98600-6.

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

[18] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Fransisco, 1979.

[19] C. Reeves, ed., *Modern Heuristics Techniques for Combinatorial Problems*. MacGraw Hill, 1995. ISBN 0-07-709239-2.

[20] E. Aarts and J. Lenstra, eds., *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997. ISBN 0-471-94822-5.

[21] S. Soh and S. Rai, "Experimental results on preprocessing of path/cut terms in sum of disjoint products technique," *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 24–33, 1993.

[22] L. Page and J. Perry, "A Practical Implementation of the Factoring Theorem for Newtork Reliability," *IEEE Transactions on Reliability*, vol. 37, pp. 259–267, 1988.

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

[24] Y. Dutuit, A. Rauzy, and J.-P. Signoret, "Réséda: a Reliability Network Analyser," in *Proceedings of European Safety and Reliability Association Conference, ESREL'96* (C. Cacciabue and I. Papazoglou, eds.), vol. 3, pp. 1947–1952, Springer Verlag, 1996. ISBN 3-540-76051-2.

[25] S. Lajeunesse, T. Hutinet, and J.-P. Signoret, "Automatic Fault Trees Generation on Dynamic Systems," in *Proceedings of the European Safety and Reliability Association Conference, ESREL'96*, pp. 1553–1559, European Safety and Reliability Association, 1996.

[26] E. Duhesme, J.-C. Laleuf, J.-F. Hery, and M. Bouissou, "De la modèlisation des systèmes bouclés à la génération automatique d'arbres de défaillances," in *Actes du congrès λμ10*, (Saint Malo), October 1996.

[27] P. Besnard and P. Siegel, "The preferential-models approach to non-monotonic logics," in *Non-Standard Logics for Automated Reasonning* (P. S. et al., ed.), pp. 127–161, Academic Press, 1988.

[28] S. Malik, "Analysis of cyclic combinational circuits," *IEEE Transactions on Computer-Aided Design*, vol. 13, pp. 950–956, July 1994.

[29] N. Halbwachs and F. Maraninchi, "On the symbolic analysis of combinational loops in circuits and synchronous programs," in *Proceedings Euromicro'95*, September 1995.

[30] T. Shiple, G. Berry, and H. Touati, "Constructive Analysis of Cyclic Circuits," in *Proc. International Design and Testing Conf (ITDC), Paris*, March 1996.

A The Bridge Example (continued)

The equations that describe the bridge example pictured Fig. 1 are give section 3.1. Basic events are $\mathsf{pwr}(s)$, $\mathsf{wrk}(e_1)$, ..., $\mathsf{wrk}(e_5)$. Non-basic events are $\mathsf{pwr}(n_1)$, $\mathsf{pwr}(n_2)$, $\mathsf{pwr}(t)$.

A.1 The Physics

The first problem is to determine which minterms correspond to the physics. Some of the minterms over $\mathcal{E}$ determine fully the values of non-basic events. Some other do not. The table 2 summarizes the relationship between the values of the $\mathcal{E}$'s and the values of the

Table 2: The values of the $\mathcal{G}$'s from the values of the $\mathcal{E}$'s.

$\mathsf{pwr}(s)$	$\mathsf{wrk}(e_1)$	$\mathsf{wrk}(e_2)$	$\mathsf{wrk}(e_3)$	$\mathsf{wrk}(e_4)$	$\mathsf{wrk}(e_5)$	$\mathsf{pwr}(n_1)$	$\mathsf{pwr}(n_2)$	$\mathsf{pwr}(t)$
1	1	1	$\{0,1\}$	v_4	v_5	1	1	$v_4 + v_5$
1	1	0	1	v_4	v_5	1	1	$v_4 + v_5$
1	1	0	0	v_4	$\{0,1\}$	1	0	v_4
1	0	1	1	v_4	v_5	1	1	$v_4 + v_5$
1	0	1	0	$\{0,1\}$	v_5	1	1	v_5
$\{0,1\}$	0	0	1	v_4	v_5	w	w	$w.(v_4 + v5)$
$\{0,1\}$	0	0	0	$\{0,1\}$	$\{0,1\}$	0	0	0
0	$\{0,1\}$	$\{0,1\}$	1	v_4	v_5	w	w	$w.(v_4 + v5)$
0	$\{0,1\}$	$\{0,1\}$	0	$\{0,1\}$	$\{0,1\}$	0	0	0

$\mathcal{G}$'s. The problematic situations are those where either $\mathsf{pwr}(s)$ is false or both $\mathsf{wrk}(e_1)$ and $\mathsf{wrk}(e_2)$ are false. In these cases, the physics prefers the valuation with $w = 0$.

It is worth noticing that our model for the bridge is correct: once w set to 0, the values of the $\mathcal{G}$'s are uniquely determined by the values of the $\mathcal{E}$'s.

A.2 Monotone Decreasing Properties

The theorem 2 can be used to get the s, t disconnecting cuts. In this case, the property P is reduced to $\overline{\mathsf{pwr}(t)}$ (therefore monotone decreasing) and Q_{MD} is defined as follows.

$$Q_{MD} \quad = \quad \exists\ \mathsf{pwr}(n_1), \mathsf{pwr}(n_2), \mathsf{pwr}(t)\ E.\overline{\mathsf{pwr}(t)}$$

Q_{MD} is therefore equivalent to the following formula.

$$Q_{MD} \quad \equiv \quad \sum_{v_1 \in \{0,1\}, v_2 \in \{0,1\}} E[v_1/\mathsf{pwr}(n_1), v_2/\mathsf{pwr}(n_2), 0/\mathsf{pwr}(t)]$$

The table 3 gives the terms of this disjunct.

By applying further simplifications to the formula Q_{MD}, it is easy to show its prime implicants of Q_{MD} are $\overline{\mathsf{pwr}(s)}$, $\overline{\mathsf{wrk}(e_1)}.\overline{\mathsf{wrk}(e_2)}$, $\overline{\mathsf{wrk}(e_1)}.\overline{\mathsf{wrk}(e_3)}.\overline{\mathsf{wrk}(e_5)}$, $\overline{\mathsf{wrk}(e_2)}.\overline{\mathsf{wrk}(e_3)}.\overline{\mathsf{wrk}(e_4)}$ and $\overline{\mathsf{wrk}(e_4)}.\overline{\mathsf{wrk}(e_5)}$.

A.3 Monotone Increasing Properties

The theorem 3 can be used to get the s, t connecting paths. In this case, the property P is reduced to $\mathsf{pwr}(t)$ (therefore monotone increasing) and Q_{MI} is defined as follows.

$$Q_{MI} \quad = \quad \forall\ \mathsf{pwr}(n_1), \mathsf{pwr}(n_2), \mathsf{pwr}(t)\ E \Rightarrow \mathsf{pwr}(t)$$

Q_{MI} is therefore equivalent to the following formula.

$$Q_{MI} \quad \equiv \quad \Pi_{v_1 \in \{0,1\}, v_2 \in \{0,1\}} \overline{E[v_1/\mathsf{pwr}(n_1), v_2/\mathsf{pwr}(n_2), 0/\mathsf{pwr}(t)]}$$

Table 3: Decomposition of $\exists\ \mathsf{pwr}(n_1), \mathsf{pwr}(n_2), \mathsf{pwr}(t)\ E.\overline{\mathsf{pwr}(t)}$

v_1	v_2	$E[v_1/\mathsf{pwr}(n_1), v_2/\mathsf{pwr}(n_2), 0/\mathsf{pwr}(t)]$
1	1	$(\mathsf{pwr}(s).\mathsf{wrk}(e_1) + \mathsf{wrk}(e_3)).(\mathsf{pwr}(s).\mathsf{wrk}(e_2) + \mathsf{wrk}(e_3)).\overline{\mathsf{wrk}(e_4) + \mathsf{wrk}(e_5)}$
		$= (\mathsf{pwr}(s).\mathsf{wrk}(e_1).\mathsf{wrk}(e_2) + \mathsf{wrk}(e_3)).\overline{\mathsf{wrk}(e_4)}.\overline{\mathsf{wrk}(e_5)}$
1	0	$(\mathsf{pwr}(s).\mathsf{wrk}(e_1)).\overline{\mathsf{pwr}(s).\mathsf{wrk}(e_2) + \mathsf{wrk}(e_3)}.\overline{\mathsf{wrk}(e_4)}$
		$= \mathsf{pwr}(s).\mathsf{wrk}(e_1).\overline{\mathsf{wrk}(e_2)}.\overline{\mathsf{wrk}(e_3)}.\overline{\mathsf{wrk}(e_4)}$
0	1	$\overline{\mathsf{pwr}(s).\mathsf{wrk}(e_1) + \mathsf{wrk}(e_3)}.(\mathsf{pwr}(s).\mathsf{wrk}(e_2)).\overline{\mathsf{wrk}(e_5)}$
		$= \mathsf{pwr}(s).\overline{\mathsf{wrk}(e_1)}.\mathsf{wrk}(e_2).\overline{\mathsf{wrk}(e_3)}.\overline{\mathsf{wrk}(e_5)}$
0	0	$\overline{\mathsf{pwr}(s).\mathsf{wrk}(e_1)}.\overline{\mathsf{pwr}(s).\mathsf{wrk}(e_2)}$
		$= \overline{\mathsf{pwr}(s)} + \overline{\mathsf{wrk}(e_1)}.\overline{\mathsf{wrk}(e_2)}$

The reader may verify that, by taking the disjunct of the negation of the terms given by the table 3 and then applying some simplifications, it is easy to show that the prime implicants of Q_{MI} are $\mathsf{pwr}(s).\mathsf{wrk}(e_1).\mathsf{wrk}(e_4)$, $\mathsf{pwr}(s).\mathsf{wrk}(e_1).\mathsf{wrk}(e_3).\mathsf{wrk}(e_5)$, $\mathsf{pwr}(s).\mathsf{wrk}(e_2).\mathsf{wrk}(e_2).\mathsf{wrk}(e_4)$ and $\mathsf{pwr}(s).\mathsf{wrk}(e_2).\mathsf{wrk}(e_5)$.

Antoine Rauzy is with the French National Center for Scientific Research (CNRS) and the "Institut de Mathématique de Luminy". His topics of interest are formal methods and reliability engineering. His background is in computer science (PhD, Habilitation à Diriger des Recherches).