THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
Christelle JACOB

le mardi 25 février 2014

Titre :

Uncertainty Management for Boolean Complex Systems
Application to Preventive Maintenance of Aircrafts

Management de l'incertitude pour les systèmes booléens complexes -
Application à la maintenance préventive des avions

École doctorale et discipline ou spécialité :

ED MITT : Intelligence artificielle

Unité de recherche :

Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

Mme Janette CARDOSO (directrice de thèse)
M. Didier DUBOIS (co-directeur de thèse)

Jury :

M. Terje AVEN - Rapporteur
Mme Janette CARDOSO - Directrice de thèe
M. Thierry DENOEUX - Rapporteur
M. Didier DUBOIS - Co-directeur de thèse
M. Christopher PAPADOPOULOS
M. Antoine RAUZY - Rapporteur
Mme Christelle SEGUIN

Christelle Jacob

Uncertainty management for complex Boolean systems

Application to preventive maintenance of aircrafts

Ph.D Thesis, foreseen date: 25th February 2014

This thesis was prepared by
Christelle Jacob

Supervisors
Janette Cardoso
Didier Dubois

Institut Supérieur de l'Aéronautique et de l'Espace
Département Mathématiques, Informatique et Automatique
Campus Supaéro, 10, Avenue Edouard Belin
BP 54032 - 31055 Toulouse cedex 4
France
www.isae.fr
Tel: (+33) 5 61 33 80 80
Fax: (+33) 5 61 33 83 30

Release date: 25th February 2014
Category: 1 (public)
Edition: First

Résumé

Les modèle de sûreté de fonctionnement et de maintenance de l'avion peut contenir des incertitudes notamment sur la durée des opérations de maintenance ou sur le taux de fiabilité de certains composants. Une première étude a porté sur l'analyse du type d'incertitude concernant les données : une différence doit être faite entre une incertitude provenant de la variabilité d'une variable (ou phénomène) et une incertitude épistémique provenant d'un manque d'information (ignorance). Le fait de pouvoir prendre en compte cette différence permet aux ingénieurs de calculer des indicateurs utiles à la gestion de la maintenance plus fidèles aux données réellement disponibles, et prenant en compte les incertitudes sur les paramètres du modèle de sûreté de fonctionnement. Ces indicateurs seront très utiles lors de la prise de décision dans un environnement incertain.

Le choix d'un modèle de représentation des données s'effectue tout d'abord selon la nature des données à représenter. Dans le cadre des modèles de sûreté de fonctionnement, la méthode d'analyse choisie est celle des arbres de défaillance. Un événement redouté pour un système est donc représenté par une formule Booléenne fonction des événements élémentaires (par exemple, des pannes...) pouvant survenir dans ses sous-systèmes. Ces événements élémentaires sont représentés grâce à des variables Booléennes. Dans la théorie classique de l'analyse par arbres de fautes, les arbres de fautes sont des fonctions Booléennes monotones. Cependant, pour des raisons pratiques et avec l'automatisation de la génération des arbres, cette théorie a été étendue à des arbres de fautes non monotones, où l'expression de l'événement redouté peut contenir des littéraux négatifs.

Lorsque l'on suppose l'indépendance stochastique des variables Booléennes lors de l'évaluation de la probabilité d'une formule Booléenne, nous nous retrouvons face à un problème non-linéaire de n variables, qui consiste à trouver le minimum et le maximum de l'expression, lorsque les probabilités des événements élémentaires sont indépendantes, imprécises, et connues sous la forme d'intervalles. La méthode de l'analyse par intervalles va être utilisée pour résoudre ce problème. Nous nous sommes intéressés au cas où la formule Booléenne est décomposée sous forme de Diagramme de Décision Binaire (plus connus sous l'acronyme anglophone de BDD, Binary Decision Diagram).

Une étude comparative avec l'analyse par intervalle montre que les résultats obtenus sont équivalents dans le cas des fonctions Booléennes croissantes. Nous avons ainsi démontré que dans le cas des arbres de fautes monotones, l'hypothèse de l'indépendance des sources d'information est une hypothèse suffisante pour obtenir les bornes optimales de la probabilité de l'événement redouté en utilisant une arithmétique d'intervalles naïve. Il n'est pas nécessaire de supposer l'indépendance stochastique entre les variables Booléennes.

Lorsqu'il n'est fait aucune hypothèse d'indépendance stochastique sur les variables Booléennes – mais qu'en revanche, il est supposé que les valeurs des probabilités attachées aux événements élémentaires

proviennent de différentes sources d'informations, qui elles, sont indépendantes –, les fonctions de croyance peuvent alors être utilisées afin de fusionner ces informations, ainsi que pour la modélisation d'incertitude.

Lorsque le ou les paramètres des lois de probabilités associées à un événement sont imprécis, il est alors possible de représenter l'information à propos de la distribution de probabilité cumulée de l'événement par une paire de distributions cumulées. La distribution de l'événement se situera alors quelque part entre la distribution basse et la haute. On appelle P-box cette paire de distributions de probabilités cumulées. L'utilisation d'une P-box à la place d'une distribution simple revient à propager la famille de toutes les distributions de probabilités comprises dans la P-box.

Il est donc intéressant d'étudier les P-boxes obtenues pour les différentes lois de probabilités les plus utilisées dans l'analyse de risque : les lois exponentielles, Weibull, exponentielle avec maintenance périodique... L'étude de ces différentes P-boxes permettra par la suite de modéliser les probabilités des événements élémentaires des Arbres de Fautes par des distributions de probabilités avec des paramètres imprécis, et d'en déduire l'évolution du minimum et du maximum de la probabilité de l'événement redouté. Les résultats ont été utilisés pour le développement d'un algorithme de calcul permettant de calculer l'intervalle optimal de probabilité de l'événement redouté.

Cet algorithme a été tout d'abord testé et validé sur plusieurs formules Booléennes standards, ainsi que sur des cas d'écoles tels que le modèle du Primary/Backup Switch, qui représente un Switch d'avion permettant de passer d'un composant principal à son composant de secours. Il a ensuite été testé sur le modèle du Rudder, i.e. d'une gouverne d'avion. Cet algorithme a été intégré dans un logiciel s'interfaçant avec le logiciel Cécilia OCAS, développé par Dassault, qui calcule, à partir d'un modèle dysfonctionnel d'un système, les arbres de fautes et les coupes minimales utiles aux études de sureté de fonctionnement.

L'algorithme a ensuite été étendu en utilisant les résultats des lois de probabilités imprécises, permettant ainsi de trouver les P-box optimales de la probabilité d'un événement redouté et de pouvoir étudier son évolution à travers le temps. Cette extension a également été testée sur le modéle du Primary/Backup Switch et celui du Rudder, et ce avec plusieurs loi de probabilités différentes, permettant ainsi une simulation de différents scénarios de maintenance.

Mots clefs : Management de l'incertitude, Arbres de fautes, Probabilités imprécises, Fonctions de croyance, Binary Decision Diagrams, Analyse par intervalles.

Abstract

The models used for RAMS (Reliability, Availability, Maintainability and Safety) studies of aircrafts are full of uncertainty, like on the failure rates of some components, or on the time of maintenance operations. A preliminary study was about the type of uncertainty generated by the data: there is a fundamental difference between the variability of a phenomenon, and an epistemic uncertainty induced by a lack of information (ignorance). Taking into account those two facets of uncertainty allows engineers to compute the indicators that are useful for maintenance management in a more realistic way. They will also be helpful for decision making in uncertain environment.

The choice of representation for data depends on their nature. In our context, Fault Tree Analysis has been chosen for the safety analysis. The failure condition of a system is represented by a Boolean formula, in function of the elementary events (e.g. failures, etc.) of the components and subsystems. This elementary events are represented by a Boolean variable. In classical Fault Tree Analysis, fault trees are monotonic Boolean functions. But for practical reasons, the use of models in Safety analysis required to extend this theory to non monotonic fault trees, that can contain negative literals.

When the stochastic independence of the Boolean variables is assumed, we face a non linear optimization problem with n variables. It consists in finding the minimum and maximum of the probability of the Boolean expression, when the probabilities of the elementary events are independent, imprecise and laying in an interval. In that case, we used Binary Decision Diagrams (BDD) to represent the Boolean formula, in order to improve the computation.

When no assumption is made about stochastic independence, it is possible to assume that the values of the probabilities of the elementary events are coming from independent sources of information (experts, statistical studies, etc.). In that case, Belief Functions can be used in order to aggregate these information, by means of a specific framework.

When these are the parameters of the probability law of elementary events that are imprecise, the information can be represented by a pair of cumulative distributions (upper and lower distribution), that will contain the real one. This way of representation is called a P-box. It is interesting to study the P-boxes obtained with the widely used probability laws of RAMS studies (exponential, Weibull, etc.), in order to deduce the evolution of the probability of a Failure Condition with time.

These results have been used in order to develop an algorithm that computes the optimal probability interval for a Failure Condition. It has been validated first on standard Boolean formulas. Afterwards, it has been tested on case studies like for a primary/backup switch, that illustrates a reconfiguration case for a passive redundancy, and on the model of an aircraft rudder. It has been integrated in a tool that takes in input the fault trees or minimal cut sets generated by the tool Cécilia OCAS, developed

by Dassault industry. OCAS is a model based safety tool that computes fault trees or minimal cut sets from the dysfunctional model of a system. Hence, it has been extended to imprecise probability laws in order to compute the evolution of the probability with time. It has been tested on the safety model of the rudder, with different probability laws, allowing to test several maintenance scenarios.

Key words: Uncertainty management, Fault trees, Imprecise probabilities, Belief functions, Binary Decision Diagrams, Interval analysis.

Contents

List of Figures

Acronyms and Notations

$X_1, \ldots, X_n$	Boolean variables
$\mathcal{X}$	Set of Boolean variables
$A_1, \ldots, A_n$	Boolean atoms
Ω_i	Set of possible values of a Boolean variable X_i: $\Omega_i = \{A_i, \neg A_i\}$
Ω	Set of interpretations $\mathcal{X} \to [0,1]$: $\Omega = \prod_i \Omega_i$
$\mathcal{F}$	Set of Boolean formulas
$F, G, H, \ldots$	Boolean functions (formulas)
$x_1, \ldots, x_n$	Real variables: $x \in \mathbb{R}$
$f, g, h, \ldots$	Numerical functions
ω	Minterm on Ω
$[F]$	Set of minterms of a Boolean formula F
$[\mathcal{X}]$	Set of minterms of the set $\mathcal{X}$
$P[\Omega]$	Power set of Ω
@MOST	Airbus Maintenance Operations Solutions and Technologies
ARP	Aerospace Recommended Practice
BDD	Binary Decision Diagram
BITE	Built In Test Equipment
DIANA	Dispatch Impact Analysis
FC	Failure Condition
FHA	Functional Hazard Analysis
FMEA	Failure Modes and Effects Analysis
FT	Fault Tree
FTA	Fault Tree Analysis
MCS	Minimal Cut Set
MBSA	Model Based Safety Analysis
MEL	Minimum Equipment List
MTBF	Mean Time Between Failures
MTTF	Mean Time To Failure
P-box	Probability Box
PSSA	Preliminary System Safety Assessment
RAMS	Reliability, Availability, Maintenance and Safety
RUL	Remaining Useful Life
SSA	System Safety Assessment

Chapter 1

General Introduction

Survivability from potentially harmful events is a natural desire. Every business requires measures and predictions to avoid unforeseen events that may affect it adversely. This becomes even more important for a business involving complex systems and human lives, such as aviation industry. A common practice to avoid disasters is to apply preventive maintenance policies. Hence, for Airline Companies, maintenance cost is a very important consideration when they plan to buy an aircraft. The maintenance cost can be roughly equal to the price of the aircraft.

Airbus is one of the two leading aircraft manufacturing companies of the world. In order to be competitive, Airbus is trying to offer a comprehensive package to potential buyers which include guidelines for maintenance planning during the life cycle of the aircraft. One of the main objectives of these guidelines is to decrease maintenance costs, and at the same time improve the reliability of the aircraft. From an airline operator perspective, such offers are very lucrative because downtimes for the aircraft can have significant financial repercussions.

Keeping in mind the above facts, Airbus recently started a new project named as *@MOST* (Airbus total Maintenance Operations Solutions & Technologies). The goal of this project is to significantly improve the maintenance management, which should reduce the financial impact due to unnecessary downtimes. This will be achieved by developing software that will be instrumental in decision making with regard to maintenance and scheduling issues.

@MOST is a large project involving many interdisciplinary studies and their co-ordination. @MOST is divided into seven further sub-projects. This thesis is part of one such sub-project named as the *Dispatch Impact Analysis* (DIANA). DIANA is closely linked with decision support and diagnosis sub-projects. For an airline operator, fulfilling the planed mission schedule is very important, due to its reputation and costs involved in flights delay or cancellation. This is one factor driving the study behind DIANA project. DIANA is a tool that will be used in helping the maintenance planning and to assist the pilot in decision making. The tool is based on the analysis and the simulation of a model of the aircraft or its sub-systems, by means of some stochastic computations and the use of model checking. To be able to perform this kind of analysis, several research lines had been identified in the scope of this project.

This PhD thesis describes the work done for one of such line: *Uncertainty Management* for complex Boolean systems. In order to be able to get an optimized maintenance plan at the end of the decision making process, we must deal with uncertainty at each step. There are many different kinds

of information used in the process and each information item can generate some different kind of imprecision/uncertainty. The goal will be to represent information as faithfully as possible, and then to study the propagation of this imprecise information in the system and evaluate its impact on the outputs. In the context of the project, it has been decided that the software *Cécilia OCAS* would be used for the modeling part of DIANA. The quantitative analysis of this software being mainly done by means of *fault trees*, which is a way to represent Boolean formulas in a Safety context, it was natural to study the uncertainty propagation in this representation tool.

The goal of DIANA is different from the usual Safety Analysis; hence some computed indicators will be different from the ones used for certification purpose. For example, whereas Safety is often just considering a worst case scenario, it may be useful for operational purposes to consider different variations of the parameters in order to capture more facets of the problem. In a real world scenario, the knowledge of input values to these models is not always accurate and complete. The goal of such study is to analyze the impact of the imprecision on input values on the outputs of the models. In this thesis we will discuss different ways to represent uncertainty in fault tree analysis, and evaluate the impact of some lack of knowledge on some parameters of the models. Imprecision can be better dealt with by means of intervals that represent uncertainty. Instead of using a precise probability value, we use an interval of possible values for the probability. It also allows us to represent the ignorance on the value of a probability. That is the reason why we decided to use intervals to model the probability of failures of the components, then we propagate a family of probabilities instead of propagating a single one.

Model-based Safety analysis uses a representation of the architecture of the aircraft in order to compute the probability of failure of some systems. From those models, we can extract some fault trees, representing the failure conditions of a system as function of the failures of its components and/or subsystems, by mean of *and* and *or* gates. Those fault trees can be written as Boolean formulas for analysis and model checking purposes.

When the probabilities of the failures of the components are supposed to be lying in intervals, there are several ways to deal with this information. Some of them will be discussed in this thesis. The assumptions made on input probabilities also have a significant influence on the appropriate way to manage and propagate the uncertainty in the model. Hence, according to the assumptions, different techniques will be adopted. This thesis will discuss some different types of assumptions about the dependence of the variables and/or their probabilities, and different ways to handle them.

The theoretical problem encountered in this study actually goes beyond fault tree analysis. It concerns any large non-monotonic Boolean formula, and may have applications in any domain that uses probabilistic computations on Boolean logic. It is not limited to fault trees. The context of this study mostly impacts on some of the implementation choices, and on the application part of the thesis. This thesis has been organized in 8 chapters. Appendices contain the additional material to explain in depth certain algorithms, concepts and tools. A brief overview of the thesis is given below:

Part II is dedicated to the state of the art on the different domains and theories that have been used to carry out this work.

After an introduction to RAMS (Reliability, Availability, Maintenance, Safety) domain, **Chapter** 2 explains the objectives of a Safety analysis, and the main steps of its methodologies. In this context, we will also introduce fault tree analysis, which is a technique used to carry out some of the steps of a Safety analysis. Nowadays, the trend of Safety analysis is to be more and more model based, hence a new domain called *MBSA* (Model Based Safety Analysis) is expanding. The main objective of this domain is to automatize the generation of some Safety documents and techniques, and fault trees do not escape to that trend. Automated generation of fault trees is currently being developed,

and this thesis will focus on the uncertainty management for this kind of fault trees.

In **chapter** 3, the background of fault tree analysis is presented in order to give an overview of the existing theory that is behind this technique. The basics of Boolean logic and fault tree and the probability computation from a fault tree are recalled. The use of *Binary Decision Diagrams* (BDD) to compute this probability has been a huge improvement for the handling of very large systems. The process of this probability computation is explained. It will be the basis of some of the uncertainty management algorithms developed in the context of this PhD thesis.

Chapter 4 proposes an overview of recent uncertainty management frameworks. The complexity of a faithful representation of the information requires some adapted methods, that will allow to capture different facets of the uncertainty. These representations should really be adapted to the situation to represent, hence this chapter makes an overview of representations and their application context, depending on different dependency assumptions on the probabilities. In order to be able to chose the best framework in the context of fault tree analysis, the mathematical theory behind this technique is analyzed, i.e the computation of probabilities from a complex non monotonic Boolean formula, with different assumptions.

Part III actually presents the main contributions of the thesis.

Chapter 5 describes the use of intervals in order to capture uncertainty and propagate it through a complex non monotonic Boolean formula when the probabilities attached to the Boolean variables are assumed to be independent. This approach faces a problem that is exponential in term of computation time, which is not manageable for very large Boolean formulas. The proposed solution to this complexity problem is to use a monotonicity study of the Boolean formula in order to reduce the computation time when it is possible. An algorithm is proposed in this chapter: it computes the range of the probability of a Boolean formula in function of the probability intervals of its variables, by using some properties of monotonicity of the formula in order to reduce the computation time.

In **chapter** 6, the problem of computation of the range of a probability of a Boolean formula depending on the probability intervals of its variables remains the same, but the assumption made about the dependency between the variables differs. We assume that the dependency between the variables is unknown, but that the values of their probabilities are coming from independent sources of information. In that case, *belief functions* can be used in order to compute the range of the probability of the formula. Again this computation has an exponential complexity, hence the proposed solution is to determine a formal equation from the properties of some particular ways to represent the Boolean formulas.

The results of **Chapter** 7 can be used as an input for feeding the methods developed in chapter 5 and chapter 6. It focuses on how to model the imprecise inputs and their evolution across time. This chapter describes the impact of an imprecise parameter on the probability laws that are generally used to represent the probability evolution in quantitative Safety analysis. The use of intervals for the parameters of such a probability distribution is given by a P-box, that is a pair of probability distributions defining a minimum and a maximum distribution. This pair of distributions is used to define input intervals in the previous chapters. The algorithm manages imprecise probability distributions, and is able to compute the evolution of the range of the probability of a complex Boolean formula across time.

Chapter 8 presents the implementation part of this research work. The context of the project dictates the constraint and the goals of the computations, in term of scale of the Boolean formulas to deal with, and of inputs to take into account. Hence the implementation of the algorithms presented in chapters 5 and 7 is described, along with some discussion about heuristics to reduce the

computation time. A comparison of different heuristics is elaborated, based on their efficiency and on their complexity.

Fig. 1.1 summarizes the interaction between the different main chapters: an arrow from Part II to Part III means that the theory described in the Chapter of Part II is required to understand the Chapter of Part III. An arrow inside a same Part means that the results of a Chapter are needed or useful for the comprehension of another Chapter.

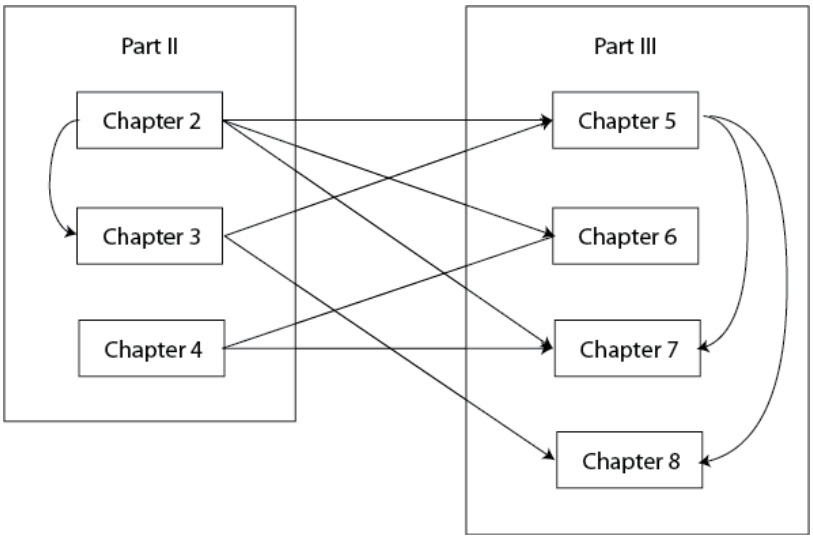


Figure 1.1: Chapters interactions

Finally, in **chapter** 9 we conclude the thesis and present different perspectives and future research directions.

Part I

Résumé en français

Introduction en français

Les coûts de maintenance constituent un paramètre important à prendre en compte par les compagnies aériennes lors de l'achat d'un avion. Durant toute son exploitation, ils peuvent s'élever quasiment au prix de l'avion lui-même. C'est pour cette raison que lors de la conception, Airbus s'efforce de diminuer significativement les coûts de maintenance, tout en conservant voir en améliorant la qualité de ses avions.

La fiabilité des différents systèmes avioniques est également un facteur crucial à prendre en compte. Les interruptions de missions et l'immobilisation de l'avion ont de grosses répercussions financières pour la Compagnie. Elles peuvent provenir des frais d'aéroport, du dédommagement des passagers lors de retards trop importants, ou en terme de rentabilité de l'avion.

C'est pour palier à tous ces problèmes et optimiser la maintenance des avions qu'Airbus a récemment lancé le projet @*MOST* (Airbus total Maintenance Operations Solutions and Technologies).

Le but du projet @MOST est d'optimiser la planification de la maintenance, afin de réduire les conséquences financières de certaines immobilisations inutiles. Bien entendu, ces économies ne doivent pas être faites aux dépends de l'opérationnabilité de l'avion, de sa fiabilité et encore moins de la sécurité des passagers. Il semble donc évident qu'une excellente analyse de risque sera une des clefs principales afin de mener à bien cet objectif. Le management de l'incertitude liée aux modèles et aux prédictions sera un facteur déterminant pour cette analyse de risque.

Le projet @MOST est un projet d'envergure Européenne constitué de plusieurs sous-projets. Cette thèse à lieu dans le cadre du sous-projet *Dispatch Impact Analysis* (DIANA), qui a pour but de développer un outil de support pour la maintenance et pour le pilotage de l'avion. L'objectif de cet outil sera de fournir à l'équipage et à l'équipe de maintenance une liste des équipements à changer ou réparer en priorité, en fonction des contraintes de sécurité, de fiabilité et d'opérationabilité.

Le but principal d'une telle liste est de permettre à l'équipe d'effectuer la maintenance de faÃ§on préventive, afin d'éviter les interruptions opérationnelles. Cela permettra également de limiter d'éventuelles diminution de performances, ou augmentations de coûts. Elle servira également de support au pilote pour la gestion du dispatch, et l'aidera à optimiser sa conduite afin de la rendre plus économique. Cet outil devra prendre en compte l'état de santé courant de l'avion, les prédictions des pannes futures, le planning de missions, ainsi que l'impact fonctionnel des pannes éventuelles. Il intégrera par exemple, des listes d'équipements telle que la *Master minimum equipment list* (MMEL), qui décrit les équipements/composants pouvant ne pas être opérationnels sans pour autant impliquer une interruption de l'avion, ainsi que les conditions et procédures éventuellement associées à leurs pannes.

Cette thèse a pour objectif principal la gestion de l'incertitude dans les systèmes Booléens complexes, qui peuvent être utilisés afin de modéliser les dysfonctionnements dans les systèmes aéronautiques. Pour pouvoir assurer une maintenance optimale, cette prise en compte de l'incertitude doit se décliner à toutes les étapes du processus de planification de la maintenance. On y rencontrera donc plusieurs types d'information différentes, et chaque information peut générer un type d'imprécision ou d'incertitude différent. Le but sera tout d'abord de représenter les incertitudes présentes de la faÃ§on la plus fidèle possible à la réalité, puis d'étudier leurs propagations et leurs impacts.

Etat de l'art sureté de fonctionnement

Dans l'industrie aéronautique, la sureté de fonctionnement regroupe les domaines de la fiabilité, la disponibilité, la maintenabilité et la sécurité. Ils sont les premiers critères de bon fonctionnement de l'avion, du bon déroulement de ses missions et de la sécurité de ses passagers et membres d'équipage. Les compagnies aériennes ont besoin d'avoir la garantie d'acheter un produit fiable, pouvant être aisément maintenu par leurs équipes de maintenance aéronautique. Un manque de fiabilité ou de maintenabilité peut avoir des répercussions économiques et/ou humaines désastreuses pour une compagnie.

C'est pour cette raison que les études de sureté de fonctionnement doivent être effectuées avec le plus grand soin dans les phases de design. Ces études prennent en compte notamment l'architecture de l'avion, le facteur humain et la fiabilité des différents composants constituant les systèmes et sous-systèmes de l'avion. Elles sont basées sur des modèles et des prédictions. Leur but principal est d'étudier les contre-temps ou dysfonctionnements éventuels, afin d'au mieux les éviter, et d'au pire, essayer d'en minimiser les conséquences.

Il est très rare qu'un désastre provienne d'un seul évènement, la plupart du temps, c'est une succession d'évènements imprévus qui conduisent à des catastrophes. Un ensemble d'évènements indésirables menant à un évènement redouté, ayant des conséquences néfastes sur le système étudié, est appelé *évènement redouté* (ER).

Identifier et analyser les risques

Des évènements redoutés peuvent provenir de différentes sources, les avions étant des systèmes très complexes. Ils peuvent être mécaniques, hydraulique, électrique, logiciel, ou liés au facteur humain. Il est souvent très difficile de prévoir les conséquences de ces derniers (Dejours, 1996)(Amalberti, 1996).

La notion de risque prend en comptes deux aspects différents : la fréquence des événements et la gravité de leurs conséquences. Ces deux aspects fonctionnent avec une certaine dualité, car plus un événement aura une probabilité d'occurrence élevée, plus ses conséquences devront être amoindries afin de réduire le risque. De même, il faudra s'assure qu'un événement avec une gravité forte ait une occurence très faible.

Des équipes d'ingénieurs seront donc en charge d'identifier les risques possibles pour les différents systèmes d'un avion : il s'agira donc de lister tous les événements redoutés possibles pour chaque composant, équipement ou sous-système, et d'étudier leurs conséquences, leur fréquence d'occurrence, leurs dépendances, etc. Cette analyse sera résumée dans un tableau récapitulatif, c'est l'Analyse des Modes de Défaillance, de leurs Effets et Criticité (AMDEC) (Stamatis, 2003). L'AMDEC sera ensuite utilisée lors de la certification de l'avion, pour s'assurer que tous les critères de fiabilité et de sécurité ont bien été respectés.

La gestion du risque passe soit par la prévention, soit par la protection (i.e. la mise en place de procédures aidant à minimiser la gravité des conséquences, comme les procédures d'urgence par exemple) ou par l'acceptation d'un risque résiduel que l'on juge acceptable. Elle passe aussi par la prise en compte de l'incertitude, qui est omniprésente dans les modèles de prévisions. C'est pour cela qu'une bonne gestion de l'incertitude est primordiale dans toute analyse de risques.

Analyse de risque basée sur les modèles

Plusieurs outils informatiques ont été développés afin de mener à bien les études de sureté de fonctionnement de systèmes complexes. Dans le cadre de cette thèse, nous utiliserons le logiciel Cécilia OCAS, développé par Dassault et basé sur le langage AltaRica(Arnold et al., 2000).

Le langage AltaRica est basé sur les automates de mode(Point and Rauzy, 1999), qui permet de décrire un système ayant plusieurs états, changeant en fonction d'événements survenant dans le système. Il est ensuite possible de connecter plusieurs automates de mode entre eux, grâce à des flux d'entrée/sortie. Par le biais de cette structure, il devient possible de modéliser le comportement d'un système, en fonction du comportement des composants ou équipements qui le composent, et de leurs interactions.

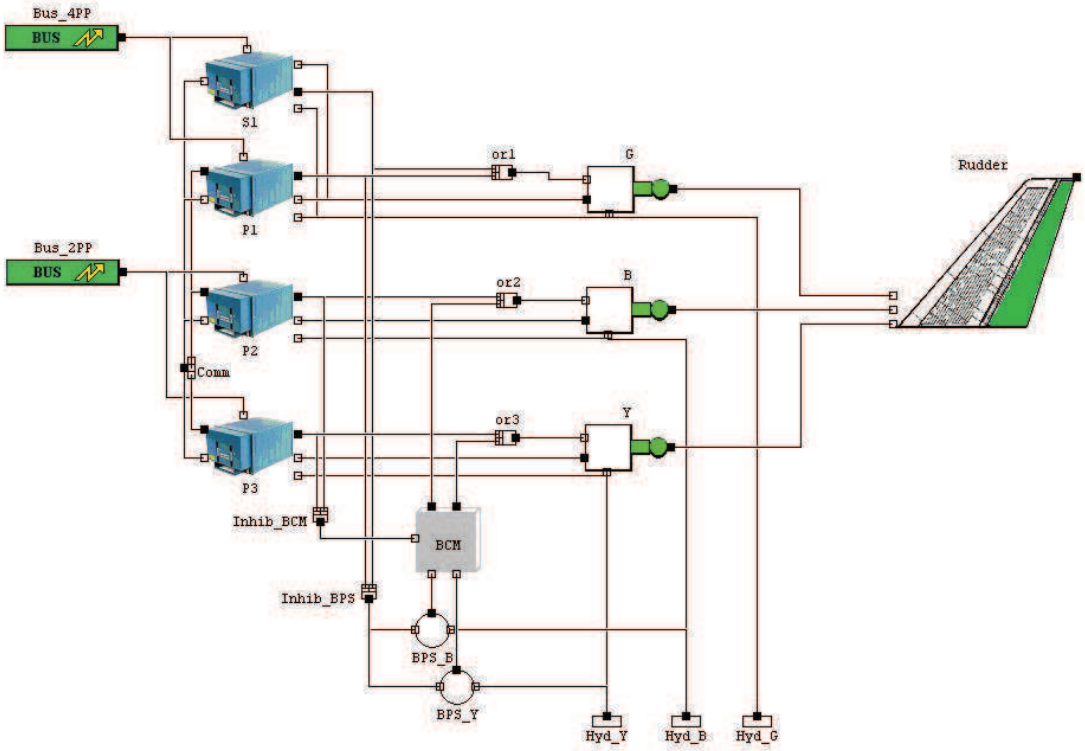


Figure 1.2: Modèle OCAS d'une gouverne d'avion

A partir du modèle hiérarchique des systèmes d'un avion, le logiciel Cécilia permet une analyse qualitative, grâce à un outil de simulation puissant, et un calculateur permettant d'extraire du modèle plusieurs indicateurs très intéressants comme l'ensemble des coupes minimales (définis dans la section 2.2.1). Il autorise également une analyse quantitative, avec des études stochastiques effectuées sur des arbres de fautes extraits du modèle (Rauzy, 2002).

Arbres de défaillances, Coupes Minimales

Les *Arbres de défaillances* (AdF) constituent un outil graphique utilisant des opérateurs Booléens afin de représenter des chaînes d'événements conduisant à un événement redouté. Cet événement redouté sera alors décrit par toutes les combinaisons d'événements élémentaires pouvant conduire à lui. Il sera représenté par le sommet de l'arbre, tandis que les événements de base constitueront les feuilles, dans la partie basse de l'arbre.

L'analyse par arbres de fautes peut être utilisée lors de la phase de conception du système, afin d'aider à l'amélioration de sa structure, ou lors de la phase de certification. Il est également possible d'utiliser les arbres de défaillances dans le but d'identifier les combinaisons d'événements les plus critiques, ou d'analyser l'impact de la panne d'un composant sur le système.

En ce qui concerne les études quantitatives à partir des arbres de fautes, des procédés utilisant des méthodes stochastiques telles que des simulation de Monte-Carlo (Rao et al., 2009) ont été largement utilisées, mais elles présupposent que toutes les probabilités des événements de base soient connues et précises. Cependant, ce n'est pas toujours le cas en pratique.

C'est pour cette raison que des travaux ont été effectués afin de prendre en compte l'imprécision ou le manque de connaissance sur les probabilités de ces événements de base : Lev Utkin travailla sur l'utilisation des probabilités imprécises dans les études de fiabilité(Utkin, 2004)(Utkin and Coolen, 2007), et Enrico Zio sur l'application d'une méthode de propagation jointe de données probabilistes et possibilistes(Baudrit et al., 2006)(Baudrit et al., 2007) à des AdF(Zio and Baraldi, 2008).

Dans le chapitre qui va suivre, nous allons nous intéresser au problème de l'utilisation de probabilités imprécises dans les arbres de défaillances.

Problème théorique

Comme nous l'avons vu dans le chapitre précédent, les arbres de fautes font partie des moyens utilisés pour l'analyse de risque stochastique. Un arbre de défaillances peut être vu comme une formule Booléenne, fonction de ses événements de base.

AdF et formules Booléennes

Definition 1.1 (Fonction Booléenne)
Une *fonction Booléenne* sur un ensemble de variables Booléennes $\mathcal{X} = \{X_1, ..., X_n\}$ est une fonction F: $\{0,1\}^n \rightarrow \{0,1\}$. Elle s'écrit à l'aide des formules Booléennes, constituées des variables $X_1, ..., X_n$ et des opérateurs $\vee$ (disjonction), $\wedge$ (conjonction) et $\neg$ (négation). $\diamondsuit$

Un littéral est soit une variable X_1, soit sa négation $\neg X_1$. X_1 est un littéral positif et $\neg X_1$ un littéral négatif. Un *minterm* ω d'un ensemble $\mathcal{X}$ de variables Booléennes est une conjonction consistante de toutes les variables de $\mathcal{X}$, apparaissant positivement ou négativement. Les événements de base d'un arbre de faute sont représentés par des variables Booléennes : elles peuvent prendre deux états. Par exemple, on peut considérer les états "en panne" (=1) ou "ok" (=0). Mais par extension, elles sont aussi utilisées pour des états plus généraux, qui pourraient représenter différents modes du composant, comme "jour" (=1) ou "nuit" (=0).

Il existe plusieurs méthodes afin de calculer la probabilité d'un événement redouté à partir de sa description sous forme de formule Booléenne, lorsque les probabilités des événements correspondant aux variables Booléennes sont toutes précises, connues et indépendantes. Malheureusement, ceci n'est pas toujours le cas en pratique, c'est pour cela que nous nous sommes intéressé aux cas où certaines de ces hypothèses ne sont pas vérifiées.

Probabilité imprécise d'une formule Booléenne

Soit $\mathcal{X}$ un ensemble de variables Booléennes $X_1, \ldots, X_n$, et F une formule Booléenne exprimée grâce à ces variables. Les variables X_i peuvent prendre les valeurs $A_i, \neg A_i$, où A_i représente un événement atomique associé aux fautes élémentaires des composants. Les modèles de F, i. e. les expressions qui la satisfont, forment un ensemble noté $[F]$ qui correspond également à l'ensemble des minterms de F. La probabilité de la formule Booléenne F peut alors être écrite de la façon suivante :

$$P(F) = \sum_{\omega \in [F]} P(\{\omega\}) \tag{1.1}$$

Dans le cas où l'indépendance stochastique des variables X_i est supposée, cette formule s'écrit sous la forme :

$$P(F) = \sum_{\omega \in [F]} [\prod_{A_i \in \mathcal{L}_\omega^+} P(A_i) \prod_{A_i \in \mathcal{L}_\omega^-} (1 - P(A_i))], \tag{1.2}$$

où $\mathcal{L}_\omega^+$ représente l'ensemble des littéraux positifs ω et $\mathcal{L}_\omega^-$ l'ensemble de ses littéraux négatifs.

Lors que les probabilités des événements élémentaires $P(A_i)$ sont mal connues, mais qu'il est possible de leur déterminer une valeur minimum et une valeur maximum, alors il va être possible de calculer une valeur minimum et maximum pour la probabilité de l'événement redouté décrit par la formule F. Ainsi, plusieurs hypothèses sont traitées dans le cadre du calcul de ces bornes minimales et maximales.

Différentes hypothèses traitées

Dans le cadre de cette thèse, différentes hypothèses d'indépendance seront discutées.

Tout d'abord, lorsqu'aucune hypothèse n'est faite à propos de l'indépendance des variables X_i : calculer les bornes $[l_F, u_F]$ de la probabilité $P(F)$ se révèle être un problème d'optimisation linéaire avec contraintes. Il peut être résolu en trouvant la solution aux problèmes d'optimisation suivants.

$$l_F = \min(\sum_{\omega \vDash F} P(\{\omega\})) \text{ et } u_F = \max(\sum_{\omega \vDash F} P(\{\omega\})) \tag{1.3}$$

sous la contrainte : $l_i \leq \sum_{\omega \vDash A_i} P(\{\omega\}) \leq u_i, i = 1 \ldots n$ et $\sum P(\{\omega\}) = 1$. C'est un cas particulier du problème de satisfaction probabilistes NP-hard, traités par de la programmation linéaire (Hansen et al., 2000).

Dans le cas où l'indépendance stochastique des variables Booléennes est supposée, la famille de probabilité induite par les probabilités élémentaires $P(A_i)$ n'est plus convexe. Ainsi, des contraintes non-linéaires sont rajoutées au problème, et les algorithmes d'optimisation linéaire s'avèrent donc inutilisables. Cependant, au lieu d'un problème à 2^n variables, nous nous retrouvons avec un problème d'optimisation non-linéaire à n variables. Nous avons donc choisi de traiter ce problème à l'aide de l'analyse par intervalles. Cette méthode sera détaillée dans le chapitre suivant I.

La seconde hypothèse d'indépendance abordée dans cette thèse, sera celle de l'indépendance des sources d'information. Dans ce cas, aucune hypothèse d'indépendance stochastique ne sera faite sur les variables Booléennes, en revanche, on supposera que les valeurs des probabilités attachées aux événements élémentaires proviendront de différentes sources d'informations, qui elles, seront supposées indépendantes. Les fonctions de croyance seront utilisées pour modéliser le problème, les détails de cette modélisation seront étudiés chapitre I.

Cas de l'indépendance stochastique

Lorsque l'on suppose l'indépendance stochastique des variables Booléennes dans le problème posé dans la section I, nous nous retrouvons donc face à un problème non-linéaire de n variables, qui consiste à trouver le minimum et le maximum de l'expression , lorsque les probabilités des événements élémentaires X_i sont indépendantes, imprécises, et connues sous la forme $P(X_i) \in [l_i; u_i]$.

L'analyse par intervalles va être utilisée afin de traiter ce problème.

Analyse par intervalles et formules Booléennes

Le principe de l'*analyse par intervalles* a été développé par des mathématiciens dans les années 60, et consiste à utiliser des intervalles de valeurs au lieu d'une valeur précise.

Une arithmétique d'intervalle naïve a été définie, très simple à mettre en pratique, cependant elle a de très fortes limitations dès lors qu'il y a répétition d'une même variable plusieurs fois dans l'expression d'une fonction. Ce cas de figure est malheureusement très courant dans les formules Booléennes, où les variables peuvent se répéter un grand nombre de fois dans l'expression. De ce fait, il est plus judicieux d'analyser la fonction mathématique décrivant la probabilité de la fonction Booléenne, afin d'en déduire l'intervalle image.

Plusieurs expressions peuvent être utilisées afin de décrire cette fonction lorsque les variables sont supposées stochastiquement indépendantes. Elle peut être déduite de la formule de Sylvester-Poincaré, aussi appelée principe d'exclusion-exclusion. Mais nous allons nous intéresser au cas où la formule Booléenne est décomposée sous forme de Diagramme de Décision Binaire (plus connus sous l'acronyme anglophone de BDD).

BDDs et probabilités de formules Booléennes

Un Diagramme de Décision Binaire est un moyen graphique de représenter une formule Booléenne. Elle correspond à une décomposition de la formule sous forme d'arbre de Shannon, en plus compacte. L'arbre de Shannon est basé sur l'itération pour toutes les variables X_i de la décomposition de Shannon d'une formule F :

$$F = (X_i \wedge F_{X_i=1}) \vee (\neg X_i \wedge F_{X=0}) \tag{1.4}$$

Suivant quelques règles de simplification graphiques, le Diagramme de Décision Binaire arrive à une représentation beaucoup plus compacte de la formule ; bien que la taille d'une BDD dépende de l'ordre des variables dans lequel la décomposition de Shannon a été effectuée. C'est un graphe comprenant au sommet l'une des variables, et à sa base, les feuilles 0 et 1.

Chaque chemin reliant le sommet et la feuille terminale 1 peut être vu comme une conjonction, et la disjonction de tous les chemins correspond exactement à la formule Booléenne représentée. Si un chemin contient le trait plein (resp. en pointillés) sortant du nœud d'une variable, alors le littéral positif (resp. négatif) de cette variable apparaitra dans la conjonction. Il est alors possible de calculer la probabilité de la formule booléenne en faisant la somme des produits des probabilités pour chaque chemin.

C'est donc cette expression de la probabilité d'une formule Booléenne qui sera étudiée, afin de déterminer son intervalle $[l_F, u_F]$.

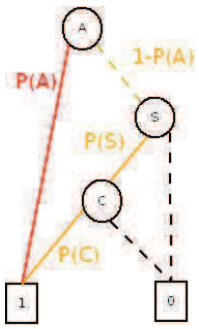


Figure 1.3: Calcul de P(F) avec $F = A \vee (S \wedge C)$

BBDs et analyse par intervalles

L'expression de la probabilité $P(F)$ d'une formule Booléenne en fonction de ses probabilités élémentaires est une fonction à n variables non monotone. Cependant, il est très important de remarquer qu'elle est *localement monotone*, c'est à dire que lorsque l'on fixe toute ses variables sauf une, alors elle est monotone par rapport à cette variable.

Ce résultat nous permet alors de déduire que les valeurs minimales et maximales de $P(F)$ seront forcément atteintes pour des valeurs minimales ou maximales des probabilités élémentaires. Lorsque la monotonicité de la formule Booléenne est connue, les bornes à utiliser pour le calcul de l'intervalle image seront également connues. Dans le cas où la monotonicité est non-triviale, il faudra alors tester les deux bornes possibles pour chaque variable, i.e. 2^n combinaisons.

Une condition suffisante pour qu'une formule Booléenne soit monotone par rapport à une variable, est que cette variable n'apparaisse que sous la forme de littéraux positifs (resp. négatifs) dans cette formule. Dans le cas échéant, la formule sera croissante (resp. décroissante) par rapport à cette variable.

Ce résultat sera utilisé pour le développement d'un algorithme de calcul permettant de calculer l'intervalle $[l_F, u_F]$ optimal I.

Approche utilisant les fonctions de croyance

Dans le problème posé dans la section I, on ne fera aucune hypothèse d'indépendance stochastique sur les variables Booléennes. En revanche, il sera supposé que les valeurs des probabilités attachées aux événements élémentaires proviennent de différentes sources d'informations, qui elles, sont indépendantes. Les fonctions de croyance peuvent être utilisées afin de fusionner des informations provenant de plusieurs sources indépendantes, ainsi que pour la modélisation d'incertitude, c'est pour cette raison que l'on va s'y intéresser dans ce chapitre.

Fonctions de croyance

La théorie des fonctions de croyance a été fondée dans les années 70 comme un cadre général de représentation des incertitudes, dont la théorie des probabilités serait un cas particulier.

Soit Ω un ensemble fini. La *masse* $m(A_i)$ d'un élément A_i de l'ensemble des parties exprime la croyance allouée par une source que l'état actuel est A_i et non un autre état, ni un sous-état de A_i.

à partir de la valeur de la masse d'un état, on peut définir un intervalle de probabilité, qui est borné par deux mesures appelées *croyance* et *plausibilité*.

Réciproquement, un intervalle $[l_i, u_i]$ définit une unique fonction de croyance sur un ensemble binaire Ω_i. Afin d'effectuer la fusion d'informations provenant de plusieurs sources indépendantes, on peut utiliser la règle de combinaison de Dempster-Shafer.

Definition 1.2 (règle de combinaison de Dempster-Shafer)
Pour deux masses m^1 et m^2, la masse jointe $m^{1,2}$ peut se calculer de la façon suivante :
- $m^{1,2}(\varnothing) = 0$
- $m^{1,2}(S) = \frac{\sum_{B \cap C = S} m^1(B) m^2(C)}{1 - \sum_{B \cap C = \varnothing} m^1(B) m^2(C)}, \forall S \subseteq \Omega$

La règle de Dempster-Shafer consiste à faire une somme conjonctive puis à renormaliser par rapport aux sous-ensembles conflictuels. Cependant, dans le cadre des formules Booléennes, il n'y a pas de conflits parmi les variables, donc le dénominateur de la formule se réduit à 1.

Plausibilité et Croyance d'une formule Booléenne

Combinaison de n fonctions de masse :
Pour n fonctions de masse $m^i, i = 1, \ldots, n$ on Ω_i, la masse jointe m^Ω sur Ω peut être calculée de la façon suivante pour chaque modèle partiel ϕ:

$$m^\Omega(\phi) = \prod_{i \in \mathcal{L}_\phi^+} l_i \prod_{i \in \mathcal{L}_\phi^-} (1 - u_i) \prod_{i \notin \mathcal{L}_\phi} (u_i - l_i) \tag{1.5}$$

La Croyance $Bel(F)$ d'une formule Booléenne F étant la somme des masses des modèles partiels satisfaisant F, son calcul requiert l'énumération des 3^n modèles partiels possibles, puis la vérification de leur satisfaction. La Plausibilité $Pl(F)$ quant-à elle, est la somme des masses de tous les modèles partiels compatibles avec F, i.e. ceux dont l'intersection avec F est non nulle. Elle peut se calculer en utilisant sa dualité avec la Croyance, qui s'exprime par $Pl(F) = 1 - Bel(\neg F)$.

Ces calculs ayant une complexité exponentielle, nous avons tenté de déterminer les cas de formules Booléennes pour lesquelles il est possible de trouver une formule explicite pour le calcul de la Plausibilité et de la Croyance. Ainsi nous avons pu prouver que les conjonctions, les disjonctions, et les formes normales disjonctives de littéraux positifs ont des expressions pour la Croyance et la Plausibilité qui se rapprochent énormément de celle de Sylvester-Poincaré :

Croyance d'une forme normale disjonctive de littéraux positifs

$$Bel(C_1 \vee \ldots \vee C_m) = \sum_{i=1}^{m} Bel(C_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} Bel(C_i \wedge C_j)$$
$$+ \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} Bel(C_i \wedge C_j \wedge C_k) - \ldots + (-1)^{m+1} Bel(C_1 \wedge \ldots \wedge C_m),$$

où les C_i sont des conjonctions de littéraux positifs.

Fonctions de croyance et Arbres de fautes

Dans la théorie classique de l'analyse par arbres de fautes, les arbres de fautes sont des fonctions Booléennes monotones, bien que cette théorie ait été étendue pour des besoins pratiques. Dans le cas des arbres de fautes monotones, il sera toujours possible d'écrire la formule Booléenne sous la forme d'une forme normale disjonctive de littéraux positifs. Les résultats précédents pourront donc être appliqués.

Dans le cas des arbres de fautes non monotone, on pourra utiliser une approximation de l'arbre à l'aide de l'ensemble des coupes minimales, qui elles, sont automatiquement générées sous une forme normale disjonctive de littéraux positifs. Une étude comparative avec l'analyse par intervalle présentée à la section I, et les résultats obtenus correspondent à l'application de l'analyse par intervalle pour les fonctions Booléennes croissantes. Nous démontrons donc ainsi que pour les arbres de fautes monotone, l'hypothèse de l'indépendance des sources d'information est *suffisante* pour obtenir les bornes optimales de la probabilité de l'événement redouté.

Lois de probabilités avec des paramètres imprécis

Les sections précédentes traitent de l'utilisation de d'intervalles afin de représenter la probabilité d'un événement élémentaire dans un arbre de faute. Cependant, on peut se demander comment cet intervalle sera lui-même déterminé. Cette section traite de la représentation de la distribution de probabilité d'un composant, puis de l'obtention des intervalles de probabilité dans le cas de lois de probabilité avec des paramètres imprécis.

Cas de paramètres précis

Les probabilités des évènements élémentaires sont représentées par des distributions de probabilités, qui suivent différentes lois selon le comportement du composant. Les lois les plus utilisées sont en général des loi exponentielles, pour modéliser les composants électroniques, ou des Weibull pour les composants mécaniques possédant des phases de jeunesse et d'usure où la probabilité de panne sera plus importante.

Les valeurs des paramètres utilisés pour ces lois proviennent d'étude statistiques établies lors des tests des composants. Un échantillon est étudié, puis des méthodes de régression linéaire sont utilisées afin de trouver les valeurs des paramètres ajustant au mieux le modèle aux observations (Morice, 1989). Lors de cette étape, les valeurs des paramètres comportent déjà une certaine imprécision, étant donné que le modèle théorique ne colle jamais parfaitement avec les observations.

Il semble donc judicieux d'essayer de prendre en compte cette incertitude sur les paramètres en compte lors de leur utilisation pour les études stochastiques des modèles. Nous nous intéresserons donc au cas où la valeur de ces paramètres sera considérée comme appartenant à un intervalle.

Cas de paramètres imprécis

Lorsque le ou les paramètres des lois de probabilités associées à un évènement sont imprécis, il est alors possible de représenter l'information à propos de la distribution de probabilité cumulée de

l'évènement par une paire de distributions cumulées. La distribution de l'évènement se situera alors quelque part entre la distribution basse et la haute. On appelle *P-box* cette paire de cumulées.

L'utilisation d'une P-box à la place d'une distribution simple revient à propager une famille de toutes les distributions de probabilités comprises dans la P-box. Elle peut être justifiée pour d'autres raisons que l'imprécision des paramètres, par exemple, si la moyenne et la variance d'une distribution sont connues, mais pas sa loi. Ou alors, elles peuvent provenir de certains intervalles de confiance.

Le cas où la loi est connue et les paramètres appartiennent à des intervalles est un cas particulier de P-box, étant donné que la famille de probabilités peut être décrite. La figure 1.4 illustre un exemple de P-box pour une distribution exponentielle dont le paramètre λ est imprécis.

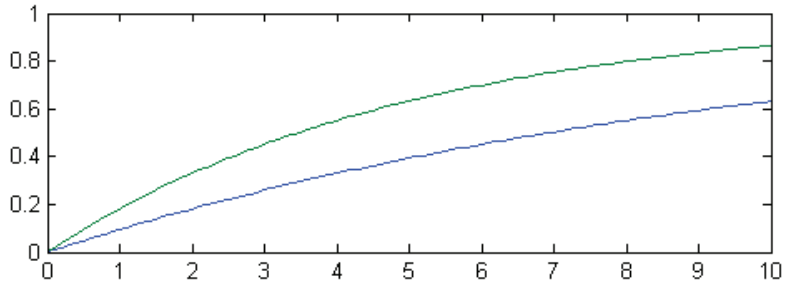


Figure 1.4: Exemple d'une P-box pour une distribution exponentielle

Il est donc intéressant d'étudier les P-boxes obtenues pour les différentes lois de probabilités les plus utilisées dans l'analyse de risque : les lois exponentielles, Weibull, exponentielle avec maintenance périodique... L'étude de ces différentes P-boxes permettra par la suite de modéliser les probabilités des événements élémentaires des Arbres de Fautes par des distributions de probabilités avec des paramètres imprécis, et d'en déduire l'évolution du minimum et du maximum de la probabilité de l'évènement redouté.

Algorithmes et applications

Dans le cadre de cette thèse, les techniques d'analyse par intervalle décrites au chapitre I ont été implémentées afin de pouvoir calculer les bornes exactes de la probabilité d'évènements redoutés décrits par des arbres de fautes. Ces arbres de fautes seront générés automatiquement par le logiciel Cécilia OCAS (I), et pris en entrée au format Aralia. Il est également possible d'utiliser des arbres de fautes partiels, décrit par l'ensemble des coupes minimales d'un certain ordre.

Les entrées de l'algorithme seront donc constituées des intervalles $[l_i, u_i]$ (ou des valeurs) de probabilités des évènements élémentaires V_i et de la formule Booléenne décrivant l'évènement redouté. Le schéma 5.6 synthétise les entrées/sorties de l'algorithme.

Le principe de l'algorithme consistera à décomposer la formule sous forme de BDD, puis de la parcourir afin de trouver des variables n'apparaissant que sous forme de littéral positif ou négatif. Cela permet d'extraire des variables dont la monotonicité est connue. Lorsque la monotonicité d'une variable est connue, on sait également déterminer l'intervalle de son image.

On aura donc en tous trois catégories de variables:
– **Type 1:** Les variables qui apparaissent seulement négativement dans la formule Booléenne
– **Type 2:** Les variables qui apparaissent seulement positivement dans la formule Booléenne
– **Type 3:** Les variables qui apparaissent positivement et négativement.

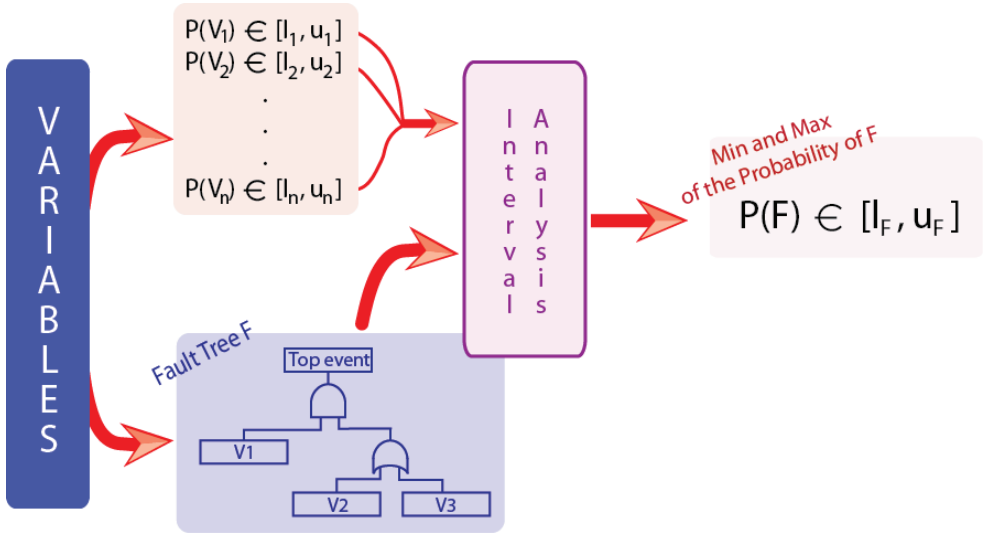


Figure 1.5: Algorithme d'analyse par intervalles pour les arbres de défaillances

Dans ce dernier cas, la monotonicité de la formule par rapport à ces variables n'est pas triviales, et il va donc falloir tester pour chaque variable les deux extrema de leur intervalle de probabilités. Nous aurons donc un algorithme en 2^k, où k est le nombre de variables de type 2.

Afin de permettre d'optimsier les temps de calcul, nous essayons de réduire au maximum le nombre de variables dans le cas 3. Dans l'idéal, notre but serait de conserver en cas 3 seulement les variables pour lesquelles la fonction est réellement non monotone. Pour se faire, nous avons donc comparé trois méthodes de calcul:
– **Cas 'A':** On utilise l'algorithme en décomposant la formule en une seule BDD
– **Cas 'B':** On utilise l'algorithme en décomposant la formule sous plusieurs BDD en intervertissant l'ordre des variables. Dans ce cas, on compare les différentes BDD et on élimine les variables de cas 3 au fur et à mesure du traitement.
– **Cas 'C':** On réduit d'abord la formule grâce à des formules mathématiques de réductions de formules Booléennes.

Cas	Nb Variables	Nb termes	Type 1 ou 2	Type 3	Temps Total
A	19	3947	3	16	4 heures
B	19	3947	16	3	<1 minute
C	19	67	13	6	1 heure

Lorsque les probabilités des évènements élémentaires ne sont pas des intervalles, mais sont décrites par des distribution de probabilités avec des paramètres imprécis, alors il est possible d'utiliser les études faites lors du chapitre I afin d'étendre l'algorithme précédent, et d'être capable d'observer les probabilités minimales et maximales de l'évènement redouté dans le temps.

La figure 7.7 schématise les entrées et les sorties de cet algorithme étendu.

Conclusion et perspectives

Le travail effectué dans le cadre de cette thèse permet d'estimer l'imprécision de la probabilité d'un événement complexe connaissant la relation booléenne qui lie l'événement redouté et les événements

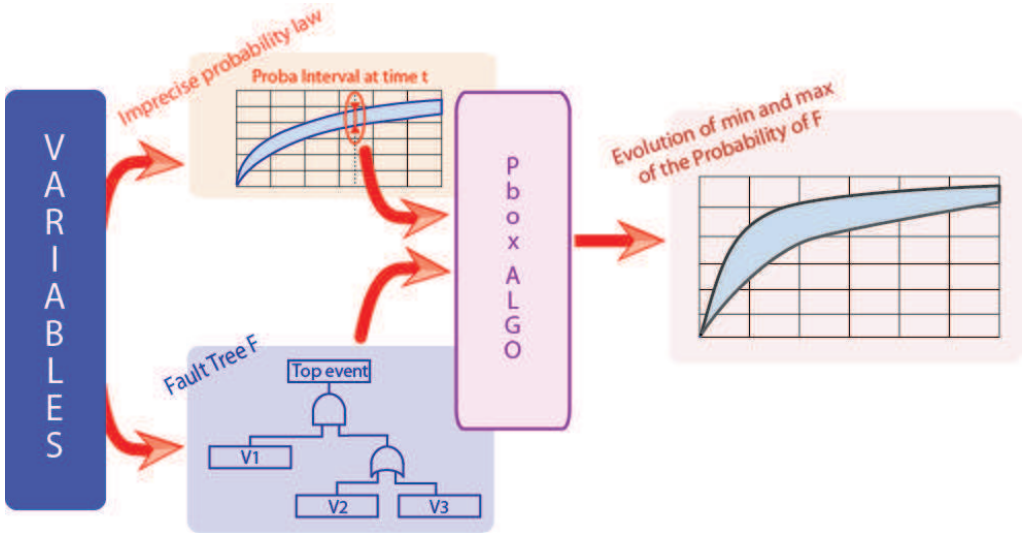


Figure 1.6: Algorithme de p-box pour les arbres de défaillances

élémentaires, ainsi que l'imprécision des probabilités des événements élémentaires. Ceci s'applique naturellement à des arbres de fautes, mais aussi à n'importe quel type de formule Booléenne dont les variables auraient des probabilités imprécises.

Le problème est abordé sous différents aspects, avec la distinction entre deux principales hypothèses :

- l'évaluation des bornes de la probabilité d'une formule Booléenne lors qu'on suppose l'indépendance forte de ses variables, on utilise l'analyse par intervalle dans ce cas,
- l'évaluation des bornes de la probabilité d'une formule Booléenne, lors qu'on suppose que les intervalles de probabilités sur les variables viennent de sources d'information indépendantes, on utilise les fonctions de croyances dans ce cas.

Une comparaison entre les résultats de ces deux hypothèses est également discutée.

Une troisième partie de la thèse est consacrée à l'évaluation des distributions de probabilités dans le cas où les paramètres de celles-ci sont imprécis. Les résultats de cette étude peuvent servir à alimenter les deux précédentes, et permettent d'étendre le calcul de l'intervalle de probabilité d'une fonction Booléenne à son évaluation dans le temps.

Plusieurs pistes de recherches peuvent être envisagées pour aller dans la continuité de ce travail. Tout d'abord il serait intéressant de comparer ces algorithmes avec les analyses de Monte-Carlo, afin d'évaluer les performances sur des cas pratiques.

Afin d'aller plus loin dans la propagation d'incertitude épistémique, il est possible d'envisager une extension des algorithmes d'analyse par intervalle aux ensembles flous. Cela est possible en utilisant le concept des α- coupes. Il serait également intéressant d'aborder les aspects dynamiques des études safety grâce aux chaines de Markovs imprécises par exemple.

Part II

State of the Art

Chapter 2

Safety analysis

Contents

In the aeronautical industry, RAMS engineering (Reliability, Availability, Maintenability and Safety) is a crucial domain. RAMS studies are important from an industrial point of view, because of the need to ensure the correct behavior of the product. The aircraft should be able to execute its missions, and guarantee the safety of passengers and crews. From an airline point of view, it is also required that a product should be properly and easily maintained, and should be reliable. The consequences of the lack of one of these requirements can be a disaster, in terms of lives and/or in terms of costs. That is why a serious RAMS study is required in the production of an aircraft. This kind of study usually takes into account, as a non exhaustive list, the architecture of the aircraft, the environment, threats, missions, the human factor and the reliability of the components (Lievens, 1976), (Pages and Gondran, 1976), (Villemur, 1988).

The main goal of RAMS studies is to ensure the compliance of the system functions and performance, and also to guarantee the safety of the people and the environment. Those studies are based on predictions. Trying to predict what could happen to the aircraft, what could cause damages, failures, or delays, and try to handle it if it is not possible to avoid it. It is impossible to ensure that an accident will never happen. But it is possible to try to make this rate of occurrence very low. Also, a good reliability and maintenance management can save huge costs for a company, that loses money for each delay of the aircraft. Hence, in addition to the safety requirements for the aircraft, we will have to deal with operational reliability, the maintenance down times and costs.

The challenge is to anticipate failures, that can happen in a very complex system, and their consequences. Illustration 2.1 shows the design process for a new system, using the experiments from the past to improve the system.

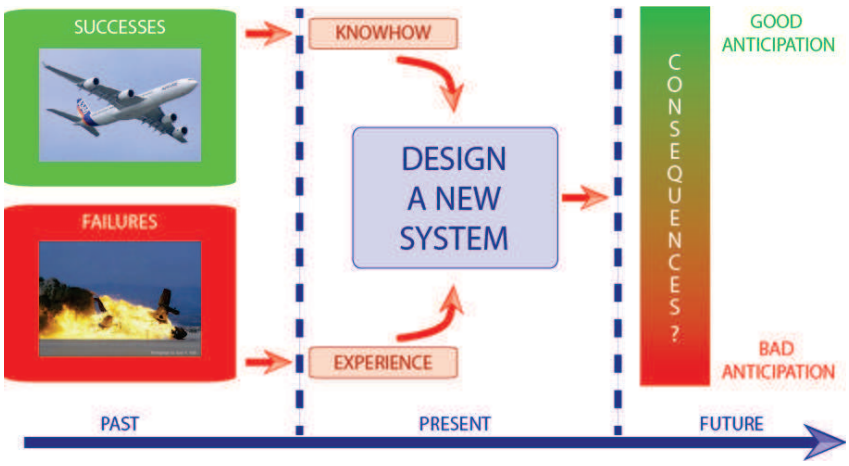


Figure 2.1: Risk mastering

As far as safety is concerned, the decision policy will always be pessimistic, and will take for granted Murphy's law: "If anything can go wrong, it will". From an operationability point of view, the policy can be different, and a bit more optimistic, as the goal is to be as much close to the real case scenarios as possible.

This chapter presents an overview of the application of safety analysis in the aeronautical domain. It is very unlikely that a disaster will be caused by only one event. Most of the time, a catastrophe happens because of several causes. The *root cause* is one of multiple factors that contributed to or created the undesired outcome. If it had been eliminated or modified, it would have prevented this outcome. A *contributing factor* is an event or a condition that may have contributed to the undesired outcome, but would not by itself have prevented its occurrence. *Failure Conditions* (FC) are a combination of events that link to an undesired event. The main goal of safety analysis is to identify and classify those failure conditions, hence to determine some requirements for the system, and verify that they are met.

2.1 Safety analysis process

An aircraft is a very large and complex system, hence, failures or dysfunctions can come from many components. Hardware, composed of electrical and mechanical devices, can either fail or break-down. But failures can also come from software, that can have unexpected bugs (Lyu, 1996)(Geffroy and Motet, 1998).They can also come from humans, as it is a man operated machine. New technologies are based on innovation, and this also implies that failures and their consequences are not always well mastered yet (Dejours, 1996)(Amalberti, 1996). The document ARP 4761 describes the recommended practices for safety analysis for a commercial aircraft. Each company has their own methodologies, but generally the steps of the ARP document are followed, and eventually improved.

The notion of *risk* includes two aspects: the occurrence of the initial cause, that is not deterministic and cannot be predicted, and the consequences of this cause. These are notions of randomness and severity. They will both be used in order to try to quantify the risk. The more the occurrence of a risk will be frequent, and the more severe it will be, the more risky the situation will be. A very low

occurrence can compensate a high risk, and vice versa (Magne and Vasseur, 2006).

It is very important to be able to allocate a maximum probability acceptable for each catastrophic failure condition at system level, depending on the gravity of its consequences. Some safety requirements are defined for each failure condition. This is the first step of safety analysis, commonly known as *Functional Hazard Analysis* or FHA.

In FHA, the functional description of a system is studied to find the impact of failure of a certain function on overall system. More precisely, when a function fails, it leads to a certain failure state of the system. Many function failures can lead to the same failure state. These failure states are also referred to as *Failure Condition*(FC). Once these failure conditions are identified, they are ranked in a table with their probability, their level of gravity and their consequences on the system. Some safety requirements can be deduced from this classification.

Once the risks are identified, the next step is to use a top-down approach to list events or combinations of events that lead to a particular failure condition. This process is known as *Preliminary System Safety Assessment* (PSSA). The role of PSSA is to check the safety of the system architecture by ensuring that the functional requirements defined by the FHA are met. This step is carried out with help of fault tree analysis, which is a top down approach consisting in describing, for each FC listed in the FHA, the combination of other events that can lead to it. From this description of the FC, we develop a tree and from this tree some probabilistic computations can be done in order to evaluate the expected frequencies of the failure conditions.

It is followed by the system safety assessment whose task is to ensure the safety of the implemented system. It consists in verifying that the behavior and the probabilities of failure of the real system meet the requirements described in the FHA and the PSSA. During the SSA, the different components of the system are listed in a table, and their failure modes, their impacts and their frequencies are studied. This is called *Failure Modes and Effects Analysis* (FMEA) (Stamatis, 2003). It is a bottom-up approach, which may look very similar to FHA, because it is also based on an analysis of failures and their impact on the system. FMEA is used for the *certification* of the aircraft, to ensure that all the safety requirements, functional and probabilistic, are met. The output of the SSA is a document containing all the results from previously described steps, necessary for the certification. Those results will also be used at different steps of the life cycle of the product, by sending feedback to the designers and system engineers, so that they improve the architecture. Some additionnal analysis, named as *Common Cause Analysis* (CCA), are done in order to ensure that the independence assumptions that are generally used for the quantitative analysis in FTA are correct.

After the classification of the failures modes and conditions, it remains to manage the risk and to define the actions that must be taken for handling such failures. In that case, several solutions can be put in practice to handle and reduce the risks. In order to manage the risk, there are two main types of actions to take: prevention and protection.

Prevention consists in taking precautions during the design phase. There can be several ways to do it: by mean of a good reliability and safety study, redundancies in the architecture, but also in having a good maintenance of the systems.

Protection consists in every action that could help reducing the gravity of the consequences, like emergency plans for example.

For every component, there will be different types of failure. The probability of each of them will be represented by a probability distribution, but those probabilities can have different meanings, depending on the type of event that we are dealing with. To each event will be attached a different probability distribution. Those probability distributions will be given by some experts, via knowledge

of failure rate. One of the widely used methods to carry out the probabilistic study of the failure conditions is Fault Tree Analysis. The following section describes the main principles, concepts and tools used in this method.

2.2 Fault tree analysis

Fault Tree Analysis (FTA) was originally developed in 1962 at Bell Laboratories by H.A. Watson. It is a particular way to analyse the failure conditions of a system. It consists in describing these conditions in a *fault tree* that can be used for the probability computations. A fault tree is a graphical design technique that uses Boolean operators to identify the chain of events leading to a specific fault or failure. An undesired event will be described as a combination of series of lower-level events. The top of the fault tree represents the undesired event, also called *top event*. The leaves represent the *elementary events*.

Fault tree analysis involves events that can be hardware wear out, material failure or malfunctions, but also combinations of deterministic contributing factors. Fig. 2.2 depicts an example of an undesired event "Brake system fails". It describes the link between the three possible causes "Actuators fail", "Sensor fail", and "Control fail", by means of the Boolean operators OR and AND.

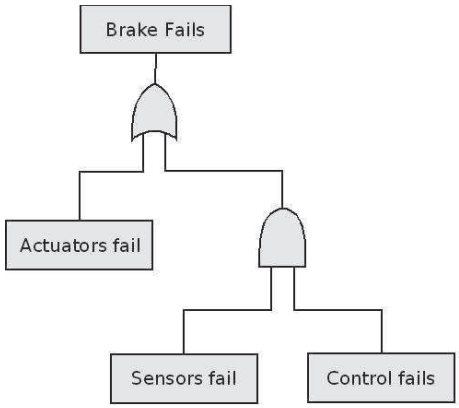


Figure 2.2: Fault tree example for the failure of a brake system

Fault trees are very efficient tools to understand the logic leading to an undesired state. It helps pointing out the compliance between the system and its safety and reliability requirements. By analyzing the fault tree, it is possible to identify and prioritize different contributors leading to the top event. The more complex and the least probable a path leading to the top event, the safer the system.

Fault tree analysis can be used as a design tool that will be helpful for creating requirements and improving the system's architecture. By studying the impact of some failures on the system, or the paths leading to some critical faults, it is possible to identify the most critical failures. Therefore, one can change the system in order to improve its reliability, e.g. by adding some redundancies of components, choosing more reliable components for a critical path, etc. In the same way, it is possible to monitor and control the safety performance of a complex system, e.g. For how long is it allowed to fly with a valve stuck closed? These results may also help in the creation of diagnosis manuals and processes.

From a fault tree, it is possible to extract several indicators that will be instrumental for the analysis of the system. They are useful for the evaluation of the impact of some particular failure or equipment

on the system, or to determine the number of failures required to reach some particular FC. Minimal Cut Sets and Importance measures are some of such indicators.

2.2.1 Minimal cut sets, importance measures

Minimal cut sets and importance measures are important indicators for fault tree analysis. They give crucial information about how the system behaves depending on its components and architecture. A *Minimal Cut Set* (MCS) is the set of all minimal conjunctions of events that are sufficient to lead to the top event. The cardinality of each combination is called the *order* of the cut. For a given order, it is possible to list all the combinations of events of this order. Such a list is instrumental for ensuring some safety requirements, or to compute an approximation of the probability of the top event. For example, a requirement for a catastrophic event is that no combination of events should have a length inferior to 3, so that a single failure cannot cause a catastrophe. So by finding no combinations of order less than 3, it ensures that the safety requirement is met. *Importance measures* allow the evaluation of the role of a component in a system's reliability. They inform about how much a system failure depends on the failure of this component. Several types of importance measures can be computed, such as:

- Conditional probabilities: the probability that the system fails if the component fails,
- Marginal importance measure: the rate, with which the availability of the system increases with the availability of the component,
- Critical importance measure: the evaluation of the relative importance of the components (highly dependent on the marginal importance factor),
- Diagnostic importance measure: the probability that the component failure has caused the failure of the system, knowing that the system has failed
- Risk rising measure: measure of the importance that the availability of a component should be ensured, to ensure the availability of the system,
- Risk decreasing measure: the maximum decrease of the risk that can be accessed by improving the reliability of a component
- Pointcaré's conditional probability: an approximation of the conditional probability that the system fails if the component fails, with the sum of all minimal cut containing the component.

We can define other importance measures depending on the system under study and the model used. Determining such indicators is part of the safety analysis; they can be used in the PSSA or the SSA.

2.2.2 Fault tree analysis tools

The probabilistic computation from a fault tree is done by mean of classical probability theory, when the basic events are independent. In that case, several tools can be used to carry out the stochastic analysis of a fault tree. Most of the existing tools used for fault tree analysis provide features to help building a fault tree manually. After the user has built the fault tree, then these tools help for its analyse by mean of some probabilistic calculators that helps computing the probability of the failure condition, of the importance measures or of the MCS. *RiskSpectrum* Ris, *Aralia SimTree* (that is

now included in *GRIF* software Sim), *CAFTA* Osys or *Isograph FaultTree+* FTp are examples of software providing such features.

But in all these software, fault trees has to be built by hand by the safety engineer, hence there is room for human mistakes. The relevance of the study and its completeness will all depend on the experience of the engineer. In order to improve and to ensure the exhaustiveness of the safety analysis, the use of some integrated tools that allows automated fault tree generation is more and more developed. In this case, the process of safety analysis is based on the use of a model of the system: it is called *Model Based Safety Analysis* (MBSA).

2.3 Model-based Safety Analysis

The research about MBSA is quite recent, but it generates more and more enthusiasm from several research teams, in industry or in academia. In this context, several tools and techniques have been developed. Different languages and methodologies have been created in order to facilitate the modeling of the failure behavior of systems. In top of the most popular ones, come the languages AltaRica (Point, 2000) and Figaro (Pfeffer, 2009). The one chosen in our project is AltaRica, it has more commercial applications than Figaro, which is a language used only in the tool *KB3* developed by EDF.

In this section, we will show how to build such a model by mean of the AltaRica language (Arnold et al., 2000), and the tool Cécilia OCAS, from Dassault industry.

2.3.1 Modeling using AltaRica

Modeling using AltaRica is based on the FHA and the FMEA described in section 2.1, and on the study of the system's architecture. It consists in modeling the behavior of each component of a system individually, and then in connecting them together following the architecture of the system, in order to simulate and evaluate the propagation and the impact of components failures on the system.

2.3.1.1 The AltaRica language

The AltaRica language was created at LaBRI in the 90's (Point and Rauzy, 1999). It is based on mode automata, that are programming constructs used to build systems with running modes. They are defined as automata, whose states are labeled by data-flow equations, and which can be composed with operators. The language allows representing generic elementary components, by describing their internal states, the events that they can receive, their inputs. Their output will be described as a function of the inputs, states and events. Fig. 2.3 depicts the structure of one component, also called *node*.

AltaRica was created to model failure propagation in a system; hence the events of the node may represent a failure or an error. But other types of event can also be represented, like an activation order for example. Each event will be associated to a failure mode or a state. Fig. 2.4 is an example of the evolution of the state of a component with the events that can occur. At the initial step, the component is in the state *Ok*, and the output is the same as the input, which means that there is no propagation of failure from this component. If the event *Error* occurs, the state of the component

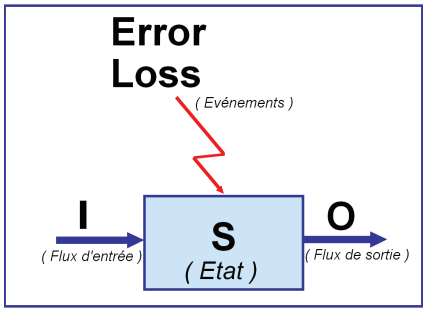


Figure 2.3: Structure of one component

becomes *Err*, and so does its output. The event *Loss* can occur when the component is in the *Ok* state, or if it is already in the *Err* state, in both cases the state will become *Lost*, so will be the output. It is interesting to note that if *Lost* can occur when the state is *Err*, the contrary is not true: the system cannot get an error if it is already down.

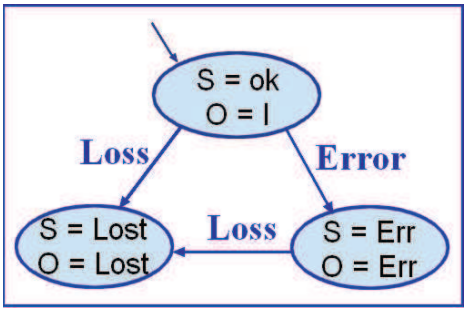


Figure 2.4: Evolution of the state of a component with events

The tool Cécilia OCAS provides a graphical interface, to help for the modeling of each component. Once the components are modeled with the language, some connections between the input and the output of several components can be made, so that they can interact with each other.

2.3.1.2 System modeling and analysis

In order to connect two different components, it is possible to link one input and one output. But their flows have to be consistent, e.g. they should be of the same type (predefined or personalized). Two components can also interact with each other by means of some synchronization of the events or the states. Some components can be used to describe some other components internally; it induces the concept of hierarchy. The graphical interface of Cécilia OCAS is very helpful for the components connections. Also, the tool helps to check the consistency of the model and to compile it. Once the model is built, a simulation process can be used in order to validate it, and to verify that it behaves as expected. At the beginning, fault trees were constructed by hand by the engineers designing the system. Now, it is possible to automatically generate fault trees from a model of the architecture of the system. The following section will describe this process.

A qualitative analysis of the model can be carried out with the tool OCAS. First, the simulation tool allows the user to select some events, and to see their effects on the system in real time. It is also possible to extract some information from the model, like the failure conditions for each failure of each part of the system. For this purpose, the user will set some *observers* in order to check his part of interest in the system. The quantitative analysis consists in computing the probability of the failure of a system (or subsystem) from the probabilities of failures of its components. Several

techniques can be used for this purpose, like fault trees (Volkanovski et al., 2009), Markov chains (Norris, 1998), or Bayesian networks (Neapolitan, 2003). In the following sections, we will focus on the techniques used in fault tree analysis.

This section describes some of the techniques where it is possible to use with the software Cécilia ARBOR, from Dassault industry. Cécilia ARBOR (System analysis by mean of fault trees) allows creating some fault trees, or to import automatically generated ones from Cécilia OCAS. It provides some algorithms for computing the probability of the top event, and to carry out sensitivity analysis on the models with Monte-Carlo simulations (Rao et al., 2009). They can be carried out on the probability of the top event, but also on importance factors, that are described in section 2.2.1. It is possible to define probability laws for representing the evolution of the failure probability of each component, and to define probability of frequencies on the parameters of those distributions. Then Monte-Carlo simulation will show the evolution of the probability of the top event. The tool also allows extracting minimal cut sets from the tree, and computing importance measures for some components. They will be introduced in the next section.

2.3.2 From architecture models to fault trees

With the help of some specialized tools, it is possible to extract some fault trees from a structural model of the system. In the case where the model is built with the AltaRica language, the tool Cécilia OCAS can compute a fault tree for any observer in the system. Different tools and methods to extract a model can be used (Majdara and Wakabayashi, 2009) (Winckell, 1998), but in this section, we will focus on how to compile mode automata into fault trees (Rauzy, 2002). The compilation of mode automata into fault trees results in a better efficiency of the assessment of the models, and a simplification of the design. But it implies the loss of the sequencing of the events: a sequence will become a conjunction of events.

The algorithm described in (Rauzy, 2002) uses a mode automaton representing a component of the system under study, and computes a set of Boolean equations from failures conditions. The algorithm is very efficient for models that are close to a block diagram representation. It is the principle use for the automated generation of fault trees from an AltaRica model of the system. It is a very efficient way to have an exhausted description of the failure conditions, and to ensure the completeness of the analysis. Such generated fault tress have different properties from the fault trees that are written by hand by safety engineer: they can contain negation of events.

Some other domains of aeronautical engineering are also closely related to safety analysis, as they study the health of the aircraft. Diagnosis and Prognosis are fields that analyze, identify and predict failures. They compute and deal with some indicators that are then used in safety analysis and more generally RAMS studies, such as MTBF (Mean Time Between Failures), MTTF (Mean Time To Failure), or RUL (Remaining Useful Life) that are defined in Appendix A.

2.4 Diagnosis and Prognosis

In simple words, diagnosis refers to a process that identifies the cause of occurrence for a given abnormal behavior or symptom. Diagnosis is being used extensively in all kind of systems and it has been under study and practice in aeronautics for several years. Different kinds of studies and research are being carried out to improve diagnosis process so that troubleshooting of systems can be done easily and more effectively. Different kind of sensors and software are added as part of the

system design to get information which helps in diagnosis.

Generally it is possible to classify embedded software pieces which help for the diagnosis into three main categories:

- The Real-Time Monitor: This type of diagnosis system (embedded software code and supporting sensors) is used primarily for the failure detection of critical functions in real time i.e as soon as they occur.
- Continuous Built-in-Test (CBIT): this diagnosis system performs the failure detection on non-flight critical functions and reports the status of real-time monitors (in background during the flight). This report is not used by the crew but by the maintenance team at the ground.
- Initiated Built-In-Test (IBIT): This kind of diagnosis consists of test sequences performed on the ground to check the system health status.

As with every measuring and detection system, the evaluation of uncertainty associated with the diagnosis process is necessary. This evaluation has many fold purposes. With the help of these evaluations we can compare different monitoring approaches and classify them according to desired criteria. We can also verify the probability of detection in case of fault. Last but not the least, these evaluations can also help in assessing the risk of false detection.

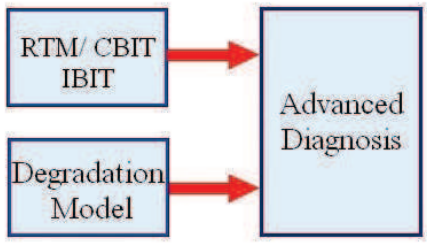


Figure 2.5: Advanced Diagnosis

This uncertainty may be contributed by sensors, sampling, imperfect/incomplete detection model, arbitrary time confirmation method to filter transient behavior or other known/unknown sources. The propagation of uncertainty is handled in the diagnosis step by building in real-time a confidence level in the detection process (e.g. the symptom really exists or not) which is propagated in the diagnosis algorithms (failure isolation and fault identification). The confidence level can be used statistically for FMEA synthesis and in real-time.

Prognosis is proactive approach as compared to reactive approach of diagnosis. Prognosis emphasizes on calculation and prediction of system's health in future, by using its past and present. It determines how the health state of a system will evolve in the future as function of the past and of the mission profile. Prognosis is the prediction of the likely occurrence of an event or outcome. The concern for prognosis in maintenance and schedule management is quite recent. Hence, it still requires the development of related methods and tools as compared to diagnosis. Research in the field of prognosis takes into account some already existing studies about mechanical degradation modeling and life consumption, and some new investigations (Ribot, 2009), for instance the computation of the RUL (Remaining Useful Life).

Existing safety models will be extended to include inputs from Diagnosis and Prognosis, along with other necessary and important inputs in order to extend them in such a way that these models can

be used dynamically for system analysis, predictions and decision making. This approach will be part of the operationability models development.

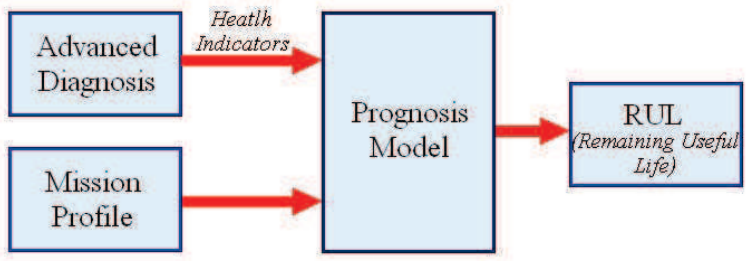


Figure 2.6: Prognosis

In the above revised approach, inputs/outputs of the model use three different levels of Prognosis namely: equipment RUL, system RUL, Aircraft RUL. Also, the inputs include the notion of degradation mode and logical assertions that can be fused with diagnostic ones. The prognosis takes its inputs from the advanced diagnosis and the mission profile as shown on fig. 2.6.

The model specifies input/output report formats such as the prognostic report format expected from aircraft systems. Some techniques of validation and verification of prognostic and health management technologies using the criteria of accuracy and precision are used in the model (Voisin et al., 2009) (Hines and Usynin, 2008). Some industrial issues raised by prognostics are also part of the model, e.g. knowledge is confined at supplier design office, then how to confirm degradation? In fig. 2.7, we can see how they use simulation for the Prognosis analysis.

First, a full simulation is done to see the real behavior of the system, then the predictions are obtained at a time t_0, where we consider that we don't know the future, and try to predict it. They compute the hypothetical behavior of the system, an associated confidence interval, and the evolution of the failure rate with time. These techniques of prognosis give some stochastic indicators that will be

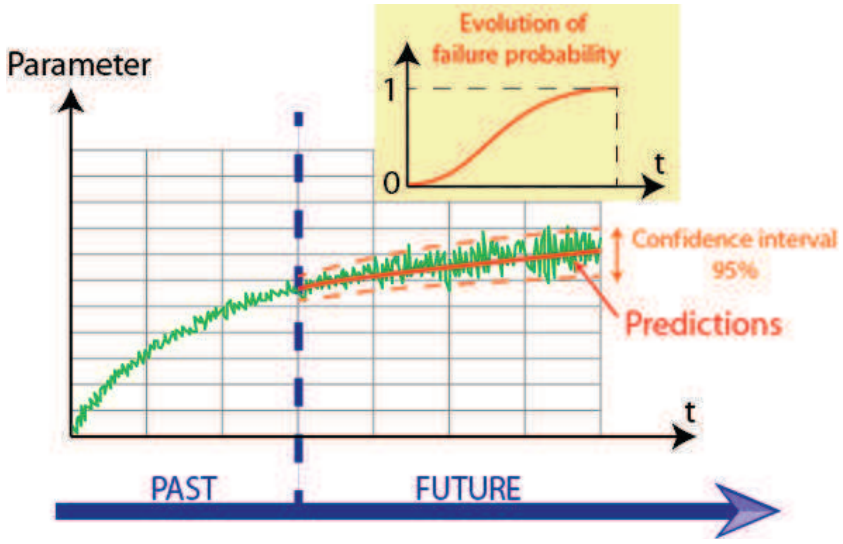


Figure 2.7: How to use simulation for Prognosis study

used in the operationability models.

Uncertainty is very present in risk management, as it is based on predictions of the future. So considering only stochastic uncertainty in predictions, models and inputs as ruled by precise probability laws is not recommended. It is better to be conscious of the uncertainty of the models, and to try to deal with it. That is the reason why uncertainty management is an important part of risk analysis. The next section describes the mathematical models that are used for representing reliability.

This Chapter gave an overview about the models that are used in the context of our studies, about the type of data that will be used in the models and also, it highlights the different sources of uncertainty present among the different computation processes.

Chapter 3

Logic and Fault Trees Analysis

Contents

As discussed in Chapter 2, fault tree analysis is one of the tools used to carry out Safety Analysis. The main objective in this thesis is to deal with imprecision in probabilistic data used for fault tree analysis. Before going into detailed discussion of how to handle this imprecision, let us first go over some basic concepts.

Basically, in fault tree analysis (FTA) theory, a fault tree can be seen as a special kind of Boolean function, with only positive literals. Each variable of this Boolean function represents the failure S of some component, that can be in either of two possible states, e.g. failed ($S = 1$) or not ($S = 0$). But this theory is used beyond this assumption in practice, for the modeling and the simulation of the systems with automatic tools like the one described in section 2.3. In this case, the Boolean variables can be used to represent some other kind of states than just failures, e.g. "day mode" ($S = 1$) and "night mode" ($S = 0$). That is the reason why, this work will consider that a fault tree is a Boolean function that can contain positive and negative literals. The next section recalls basic concepts about Boolean functions.

3.1 Basics about Boolean functions

Let us suppose that $\mathcal{X} = (X_1, ..., X_n)$ is a tuple of Boolean variables. An *assignment* of X is an element σ of $\{0, 1\}^n$.

Definition 3.1 (Boolean function)
A *Boolean function* on a set of variables $\mathcal{X} = \{X_1, ..., X_n\}$ is a function F: $\{0, 1\}^n \to \{0, 1\}$ ♢

In order to manipulate Boolean functions, we use a syntax based on *Boolean formulas*.

DEFINITION 3.2
The set $\mathcal{F}$ of **Boolean formulas** is built by structural induction from:

- A finite (or countable) set of Boolean variables $X_1, \ldots, X_n$,
- Some logic operators: $\wedge$ (and), $\neg$ (negation). $\diamondsuit$

From those operators and variables, we can induce the operator $\vee$ (or) by $A_1 \vee A_2 = \neg(\neg A_1 \wedge \neg A_2)$, and also the Boolean constants $1 : A_1 \vee \neg A_1$ and $0 : A_1 \wedge \neg A_1$.

The set $\mathcal{F}$ of Boolean formulas is the smallest set verifying that:
- 0 and 1 are formulas.
- The variables of $\mathcal{X}$ are formulas.
- If F and G are formulas, then $F \wedge G$ and $\neg F$ are formulas as well.

EXAMPLE 3.1 $F = (X_1 \wedge X_2) \vee X_3$ and $G = (X_1 \vee X_2) \wedge (X_2 \vee X_3)$ are Boolean formulas built from the set of variables $\mathcal{X} = \{X_1, X_2, X_3\}$. $\diamondsuit$

There are other logical operators constructed from the usual ones ($\wedge$, $\vee$, $\neg$):
- Implies ($\Rightarrow$): $F \Rightarrow G$ is short for $\neg F \vee G$.
- Equivalent ($\Leftrightarrow$): $F \Leftrightarrow G$ is short for $(F \Rightarrow G) \wedge (G \Rightarrow F)$.
- If-Then-Else (ite): $ite(F, G, H)$ is short for $(F \wedge G) \vee (\neg F \wedge H) \Leftrightarrow (F \Rightarrow G) \wedge (\neg F \Rightarrow H)$.

DEFINITION 3.3
A *literal* is either a variable X_1, either its negation $\neg X_1$: X_1 is a *positive literal*, and $\neg X_1$ a *negative literal*. X_1 and $\neg X_1$ are called *inverse* of each other.
A *product* is a conjunction of literals which never contains a literal and its inverse; a positive product contains only positive literals.
A *minterm* or an *interpretation* ω on $\mathcal{X}$ is a product which contain, positively or negatively, all the variables of $\mathcal{X}$. We denote by *minterm($\mathcal{X}$)* the set of all possible minterms (interpretations) of $\mathcal{X}$. $\diamondsuit$

EXAMPLE 3.2 For the set $\mathcal{X} = \{X_1, X_2\}$:
X_1X_2 and $X_1\neg X_2$ are some minterms of $\mathcal{X}$.
$minterm(\mathcal{X}) = \{X_1X_2, \neg X_1X_2, X_1\neg X_2, \neg X_1\neg X_2\}$.

A Boolean formula can always be expressed by a *disjunction of minterms*. For example the formula $F = (X_1 \wedge X_2) \vee (X_1 \wedge \neg X_3)$ can be written as $F = X_1X_2X_3 + X_1X_2\neg X_3 + \neg X_1X_2X_3 + \neg X_1\neg X_2X_3$.

DEFINITION 3.4
It is possible to represent a minterm ω by the set ω^c of its positive literals, called *Herbrand*'s representation.

EXAMPLE 3.3 For the set $\mathcal{X} = \{X_1, X_2, X_3\}$, the minterm $X_1X_2\neg X_3$ will be represented by $\{X_1, X_2\}$.

3.2 Boolean formulas and fault-tree analysis

In Boolean models of risk analysis, the variables represent the states of the elementary components, and the formulas describe the failures of the system as function of those variables (Thomas, 2002).

As seen in Section 2.2.1, it is interesting to know the minimal sets of failures of elementary components which imply a failure of the studied system, called *minimal cut set*.

Definition 3.5 (Implicants, Prime Implicants)
Suppose that F and G are two Boolean formulas built on a set $\mathcal{X}$ of variables. If all the minterms of F are minterms of G, we say that F *implies* G, and we note it $F \models G$.
An *implicant* of F is a product term π such that $\pi \models F$.
An implicant is a *prime implicant* if it contains a minimal number of literals, i.e there is no smaller product of π that implies F. We note PI[F] the *set of prime implicants* of a formula F.

Any formula is equivalent to the disjunction of its prime implicants. For example, the formula $F = (X_1 \wedge X_2) \vee (\neg X_1 \wedge X_3)$ built on $X = \{X_1, X_2, X_3\}$ admits the following implicants : $X_1X_2X_3$, $X_1X_2\neg X_3$, $\neg X_1X_2X_3$, X_1X_2, $\neg X_1X_3$ and X_2X_3. Between them, only X_1X_2, $\neg X_1X_3$ and X_2X_3 are prime implicants.

In fault tree analysis, there is a fundamental asymmetry between positive literals (that represent failures) and negative literals (that represent that everything is OK): the latter one carry less information than the first one.

In the example above, the product $\neg X_1 \wedge X_3$ contains a negative literal ($\neg X_1$) that doesn't contain any interesting information. In fact, we would like to keep only X_3 from this product, because the information given by the negative literals are not so relevant. But we have $X_2 \wedge X_3 \models X_3$ so $X_2 \wedge X_3$ is no more prime. The cut set that we want to obtain is $\{X_1 \wedge X_2, X_3\}$. A *minimal cut set* includes the positive minimal part of the prime implicants of a formula F.

Definition 3.6 (Cut, Minimal Cut)
Let $\mathcal{X}$ be a set of variables, ω a minterm and ω^c the Herbrand's representation of ω^c.

- ω^c is a *cut* of F if ω satisfies F.
- ω^c is a *minimal cut* if there is no subset of ω^c that is a cut of F. We denote MCS[F] the set of minimal cuts of F. ♢

In order to formalize the link between minimal cut sets and prime implicants, we introduce a natural order on the literals: for each variable X, we consider that $X < \neg X$. This order on literals is extended to an order on the minterms: suppose that ω and ρ are two minterms built on the same set of variables $\mathcal{X} = \{X_1, ..., X_n\}$. $\omega < \rho$ if for every X_i, $\omega[X_i] \leq \rho[X_i]$ and if it exists at least one X_i for which this inequality is strict.

A sufficient condition (but not necessary) for a formula to be monotonic is to contain only positive or only negative literals. A formula is *increasing* if for all minterms ω and ρ we have: if $\omega \models F$ and $\rho < \omega$, then $\rho \models F$.

The prime implicants of an increasing formula only contain positive literals. Indeed, if $\neg X_i \wedge \omega$ were a prime implicant of an increasing formula, then $X_i \wedge \omega$ would satisfy F (with the definition of monotonic), it means that ω would satisfy it as well so $\neg X_i \wedge \omega$ is not prime. The notions of prime implicants and of minimal cut sets are the same for monotonic formulas: if F is monotonic, PI[F]=MCS[F].

Definition 3.7 (Monotonic envelope)
If f is not monotonic, its minimal cut sets are prime implicants of a monotonic formula g. We will then define the *monotonic envelope* $\uparrow f$ of a formula f built on a set of variables $\mathcal{X}$ such that:

$$Minterm(\uparrow f) = \{\pi \in Minterms(X), \exists \rho \in Minterms(X) \mid \rho \models f \text{ and } \pi \leq \rho\} \tag{3.1}$$

♢

For any function, we have $f \models \uparrow f$ and $\text{MCS}[f] = \text{PI}[\uparrow f]$

The main drawback of these computations is that they just give an approximation of the probability. It was really necessary to improve those computation times in order to be able to have an accurate result for complex Boolean systems. That is the reason why the use of different formula representation, such as Binary Decision Diagrams (BDD) has been introduced.

3.3 Binary Decision Diagrams (BDDs)

The concept of *Binary Decision Diagrams* (BDDs) has been used in safety analysis in order to overcome the computation time problems with complex fault trees. It consists in a graphical way to represent Boolean formulas, with a view to simplify the probability computations. Before introducing the concept of BDDs, let us first talk about their origin: Shannon decomposition.

3.3.1 Shannon decomposition

Definition 3.8
Let us consider F a Boolean function on a set of variables $\mathcal{X}$, and X a variable of $\mathcal{X}$. The *Shannon decomposition* of F related to $\mathcal{X}$ is given by:

$$F = (X \wedge f_{X=1}) \vee (\neg X \wedge f_{X=0}) \tag{3.2}$$

♢

where $F_{X=1}$ (resp. $F_{X=0}$) is the function with all X replaced by the true value 1 (resp. the false value 0).

When we choose an order for the variables of a set of variables $\mathcal{X}$, and we apply recursively the Shannon decomposition for each variable of a function F built on $\mathcal{X}$, we get a binary tree; called *Shannon tree*. Each internal node of this tree can be read as an *if-then-else (ite) operator*. Each node contains a variable $\mathcal{X}$, and has two edges: one edge pointing towards the node encoded by the positive cofactor $F_{X=1}$ and the other one towards the negative cofactor $F_{X=0}$.

The leaves of the tree are the truth values of the formula 0 or 1. The expression obtained by the product of variables going down from the root to the leaf 1 is a minterm, and the sum of all those products gives the expression of the formula. But such a representation of the data is very costly in memory space.

Example 3.4 The Shannon tree for the formula $A \vee (S \wedge C)$ and the order A, S, C (the dotted edges represent the *else*) is represented on Fig. 3.3. ♢

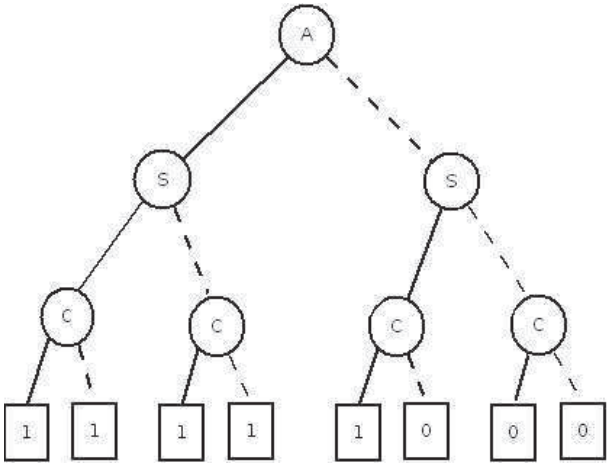


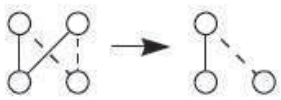
Figure 3.1: Shannon tree of $A \vee (S \wedge C)$ and the order A, S, C

3.3.2 Basics about Binary Decision Diagrams

The principle of BDDs is to reduce the size of the Shannon tree by using reduction rules.

Definition 3.9 (BDD reduction rules)
There are two rules for reducing a Shannon tree in order to get a BDD:

R1 As all isomorphic trees encode the same function, one is sufficient. We will then need only two leaves : 0 and 1.



R2 We delete all useless nodes: a node both outward edges of which point to the same node is useless.

When we apply those two rules as often as necessary, we obtain the BDD associated to the formula.

A BDD is a directed graph without cycles. For a given ranking of variables, the BDD is unique up to an isomorphism. The size of the BDD depends on the ranking that we choose for the variables. We can use some heuristics to find rankings that leads to small BDDs.

In a BDD each *path* from the root to terminal node 1 can be seen as a product, and the disjunction of all those products gives a representation of the function. A variable can be either *present* (in a positive or negative polarity) in a path, or *absent*.

- a variable is present in a positive polarity in the corresponding product if the path contains the *then-edge* of a node labeled by this variable,

Example 3.5 (From the Shannon tree to a BDD)

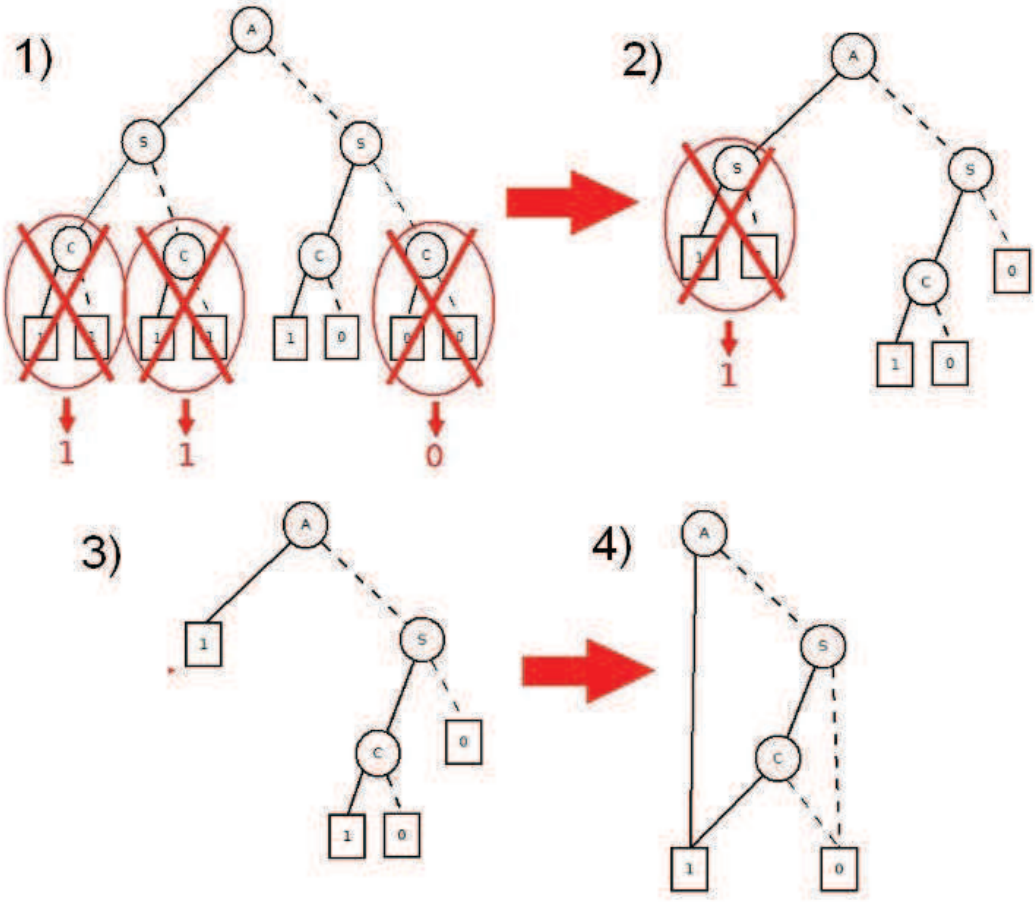


Figure 3.2: From the Shannon tree to the BDD of the formula $A \vee (S \wedge C)$ and the order A, S, C

- a variable is present in the negative polarity in the corresponding product if the path contains the *else-edge* of a node labeled by this variable,
- a variable is absent if the path doesn't contain a node labeled by this variable.

3.4 Probability of a Boolean formula

For probability computation of a Boolean formula, the experimental approach consists in approximating the probability by computing a relative frequency from tests: if we realize N experiments (where N is a very large number) in the same conditions, and we observe n times the event e, the quotient n/N give an approximation of the probability $\mathrm{P}(e)$. Indeed, this probability can be defined by: $P(e) = \lim_{N \to +\infty} (n/N)$. In practice, it is very difficult to observe events such as complex scenarios involving multiple, elementary events several times.

The analytical approach can be used when a failure condition is described as a Boolean function F of atomic events. Such dreadful event probability is thus computed from the knowledge of probabilities of atomic events, that are often given by some experts. The Boolean methods of risk analysis often make the hypothesis that **all these elementary events are independent**.

There are two methods to calculate the probability of a Boolean formula, considering:

- the Sylvester-Poincaré development, also known as inclusion-exclusion principle (letting each X_i stand for product of literals):

$$P(X_1 \vee ... \vee X_n) = \sum_{i=1}^{n} P(X_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(X_i \wedge X_j) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} P(X_i \wedge X_j \wedge X_k) - ... + (-1)^{n+1} P(X_1 \wedge ... \wedge X_n) \tag{3.3}$$

- the Shannon decomposition of the formula. This approach will be discussed in section 3.3.1.
- the formula as a sum of minterms.

As we have seen in section 3.1, a Boolean formula can be written as a sum of minterms, that are mutually exclusive by definition. So the probability of a dreadful event is equivalent to:

$$P(f) = \sum_{\pi \in Minterms(f)} P(\pi) \tag{3.4}$$

The assumption of independance between the probabilities of the variables give the following formula:

$$P(f) = \sum_{\pi \in Minterms(f)} \prod_{x \in \pi} P(x) \tag{3.5}$$

But this method requires to have all minterms of f, that is very costly in computation time.

Practically, we can compute the probability of a formula f by applying Sylvester-Poincaré theorem (3.3) to the minimal cut sets of the formula, more precisely, we compute an approximation of this probability by making some simplifications:

1. We consider the monotonic envelope of the formula instead of using the formula itself. This simplification is pessimistic because for each formula, $f \models\!\uparrow f$, means that $P(f) \leq P(\uparrow f)$.
2. We consider just the minimal cut sets of greatest probabilities. This simplification is optimistic because it ignores some of the minterms of f.
3. We consider only the k first terms of Sylvester-Poincaré development. This simplification can be optimistic if k is even, or pessimistic if k is odd.

The BDD representation of a formula can be more compact than the formula itself (sect. 3.3.2). The computation of the probability using the BDD representation of the formula is faster than using the formula or the Shannon decomposition.

We denote $\mathscr{P}$ the set of paths of the BDD, that reach the terminal leaf 1 and $\mathscr{A}_p$ the set of literals contained by path p of $\mathscr{P}$. If a literal a is present positively (resp. negatively) in the path $p \in \mathscr{P}$, we will say that $a \in \mathscr{A}_p^+$ (resp. $a \in \mathscr{A}_p^-$).

According to those notations, we have:

$$P(F) = \sum_{p \in \mathscr{P}} \prod_{a_i \in \mathscr{A}_p^+} P(a_i) \prod_{a_j \in \mathscr{A}_p^-} (1 - P(a_j)) \tag{3.6}$$

EXAMPLE 3.6 Let us consider the fault tree presented as an example in section 2.2 on fig. 2.2.

Let A be the event "Actuators fail", represented by a Boolean variable of same name, $A = 1$ means that the actuators fail, and $A = 0$ that they didn't fail. The same way, S is "Sensors fail" and C is "Control fails". We will call F the top event "Brake fails". F can be expressed as a Boolean formula:

$$F = A \vee (S \wedge C) \tag{3.7}$$

Let us compute P(F).

1. Using Sylvester-Poincaré theorem

   $P(F) = P(A \vee (S \wedge C)) \Leftrightarrow P(F) = P(A) + P(S \wedge C) - P(A \wedge S \wedge C)$ As all events are considered as independent, we have $P(S \wedge C) = P(S) \times P(C)$ and $P(A \wedge S \wedge C) = P(A) \times P(S) \times P(C)$. So we get:

   $$P(F) = P(A) + P(S) \times P(C) - P(A) \times P(S) \times P(C) \tag{3.8}$$

2. Using the formula of Shannon tree (3.2)

   The computation of $P(F)$ from the Shannon tree is obtained by summing the probabilities of all paths leading to 1:

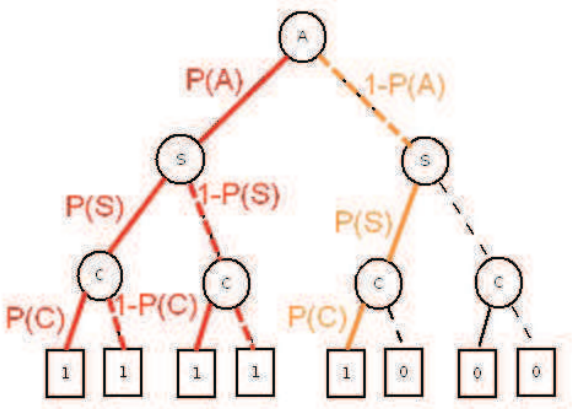


Figure 3.3: Computation of $P(F)$ with $F = A \vee (S \wedge C)$ from Shannon tree

   $P(F) = P(A) \times P(S) \times P(C) + P(A) \times P(S) \times (1 - P(C)) + P(A) \times (1 - P(S)) \times P(C) + P(A) \times (1 - P(S)) \times (1 - P(C)) + (1 - P(A)) \times P(S) \times P(C)$
   $\Leftrightarrow P(F) = P(A) \times P(S)[P(C) + (1 - P(C))] + P(A) \times (1 - P(S))[P(C) + (1 - P(C))] + (1 - P(A)) \times P(S) \times P(C)$
   We get:

   $$P(F) = P(A) + (1 - P(A))(P(S) \times P(C)) \tag{3.9}$$

3. From the BDD

   The BDD representation of the formula (3.7) is $F = A + (1 - A)SC$ (see fig. 3.2). Applying equation (3.6) to this formula, $P(F)$ is given by:

   $$P(F) = P(A) + (1 - P(A))(P(S) \times P(C)) \tag{3.10}$$

♢

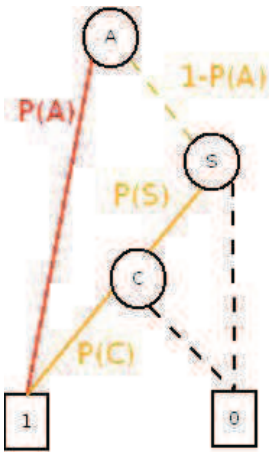


Figure 3.4: Computation of P(F) with $F = A \vee (S \wedge C)$ from BDD

As shown of this example, those three different ways to compute the exact probability of a top event are equivalent in terms of results. But in practice, the computation from the BDD is much more efficient to implement, because it is linear with the number of paths going to 1 in the BDD.

The probabilities of the elementary events that imply the top event are required to compute its probability. Those probabilities are obtained from statistical studies of the physical components and some of the mathematical concepts are described in Appendix A. With the expression of the reliability as function of the failure rate, it is possible to see the evolution of the reliability from the statistics inputs that we have on the failure rates. The shape of the failure rate will define the behavior of the reliability function.

This thesis will use the mathematical concepts about representation of Boolean formulas described in this chapter, as a base for the uncertainty management.

Chapter 4

Uncertainty theories and Reliability

Contents

All the methods in classical risk analysis describe in the previous chapters assume that all probabilities are precisely known. Moreover, it is usually assumed that a complete probabilistic information about the system behavior and component reliability is known. It is also assumed that all system components are stochastically independent. But these assumptions are not always verified in practice.

The aim of this chapter is to present some techniques and frameworks that have been already proposed in uncertainty management, and that allows to capture different facets of uncertainty. In a second step, we will expose the theoretical problem raised by the uncertainty in fault tree analysis, and discuss different approaches, depending on the assumptions made for the modeling.

4.1 Uncertainty theories

Uncertainty in risk analysis has mostly been captured from a probabilistic point of view, i.e. the uncertainty about parameters of models is generally assumed to be caused by their variability. But the uncertainty about the parameters can come from two sources (Baudrit et al., 2008). The first one is actually the variability of the parameter that comes from the randomness of the information, due to the natural variability of stochastic phenomena. The second source of uncertainty is linked to the incompleteness or the imprecision of the information, due to a lack of knowledge. It is called epistemic uncertainty.

The propagation of those two different sources of uncertainty requires different methods (Ferson and Ginzburg, 1996). Modeling the knowledge with a unique probability distribution is natural when the available information is random and precisely observed. It has some impact on the propagation of uncertainty in risk analysis models. In order to model epistemic uncertainty, different frameworks have been developed since the 80s, representing uncertainty with probability families. It is the case in Possibility theory (Dubois and Prade, 1988)(Dubois et al., 2000), where fuzzy numbers are used as a more expressive method than intervals. It is also the case for the theory of imprecise probabilities (F. Coolen, 2010). The theory of belief functions provides a way of using mathematical probability to quantify subjective judgments (G. Shafer, 1990).

4.1.1 Possibility theory

In order to handle flexible queries and to manage imprecise and uncertain information, a framework based on Fuzzy set and possibility theory can help us to provide a solution. A fuzzy number can be defined as an extension of a regular interval because a fuzzy number does not represent a single value; it refers to a connected set of possible values. Each possible value in a fuzzy number has its own weight which varies between 0 and 1. This is also known as *membership function* which describes a subpart of a universe U whose boundaries are not strictly defined and a grade of membership (valued in [0,1]) is attached to each element of U. The advantage of this kind of representation is that there is a gradual transition between full membership and exclusion which enables us to have a better representation of gradual properties, vague classes and approximate descriptions such as currently used in natural languages when we speak of the real world.

In other cases, the problem can be ”how to represent an ill-known value x?”. In such cases, possibility distribution concept is used to express a restriction on the approximate possible values of x and to represent the extent to which a given element of U is possible as being the actual value of x. We can see that fuzzy sets and possibility distributions both deal with a universe U but it should be noted that they tackle with two different issues.

Probabilistic uncertainty is based upon an additive measure and is used in cases of repeated experiments but in comparison, possibilistic uncertainty is based upon a non-additive measure and generalizes the idea of ease of attainment in a situation.

Fuzzy sets can be used in many applications, e.g. they can act as a mean of translating linguistic values into possibility distributions and the concept of approximate reasoning based on fuzzy sets can be used in simulation of a large class of human reasoning operations. As an example, the paper (Yager, 1982) introduce a measure of specificity of a possibility distribution and discusses applications of fuzzy set theory to intelligent querying of databases and multiple criteria decision making. More details about fuzzy numbers and possibility theory can be found in Appendix B.

4.1.2 Belief functions

Belief functions theory was developed by Dempster and Shafer in the 70s. It is a mathematical theory of evidence, that gives a framework that allows representing both variability and epistemic uncertainty. It is based on the idea of obtaining degrees of belief for one question from subjective probabilities for a related question, i.e. a probability derived from an individual personal judgment about whether a specific outcome is likely to occur or not. Moreover, the principle is to use some rules for the combination of information coming from different sources, in order to get a degree of belief that takes into account all the available evidence. This theory has been used for several applications in aggregation of information of different types and uncertainty management.

Let us consider Ω as a set of possible realizations (answers to a question), Ω is called *frame of discernment*. When the knowledge is not enough to determine a probability for each member of the set, it is possible to associate a *mass function* to the subsets of Ω rather than to single elements.

Definition 4.1 (Mass function)
A *mass function* m is a assignment of some non negative weights to different subsets of a set Ω such that the sum of the mass of all subsets of Ω (i.e. the power set $P[\Omega]$) is 1

$$\sum_{S \in P[\Omega]} m(S) = 1,$$

and s.t. $m(\varnothing) = 0$.

In this perspective, it is possible to represent total ignorance by assigning the mass of 1 to the set Ω.

Definition 4.2 (Focal element)
A set that receives a strictly positive mass is called *focal element*.

In order to combine two independent mass functions on a set Ω, Dempster-Shafer rule of combination should be used.

Definition 4.3 (Dempster-Shafer rule)
For two masses m^1 and m^2, the joint mass $m^{1,2}$ can be computed as follows:

- $m^{1,2}(\varnothing) = 0$
- $m^{1,2}(S) = \dfrac{\sum_{B \cap C = S} m^1(B) m^2(C)}{1 - \sum_{B \cap C = \varnothing} m^1(B) m^2(C)}, \forall S \subseteq \Omega$

Shafer's framework can represent the information about a subset of Ω by means of a pair of functions, called *Belief* and *Plausibility* functions, denoted as (Bel, Pl), such that:

$$Bel \leq Pl$$

Definition 4.4 (Belief and Plausibility)
Let Ω be a set of elements, and the E_i's the elements of $P[\Omega]$. The *belief* of a set S is the sum of the mass of all sets E_i from Ω that are included in it:

$$Bel(S) = \sum_{E_i \subseteq S} m(E_i)$$

The *plausibility* of a set S is the sum of the mass of all sets E_i of Ω that are compatible with it:

$$Pl(S) = \sum_{E_i \wedge S \neq \varnothing} m(E_i)$$

♢

The Belief and the Plausibility of a set S are non-additive measures. They can be seen as the upper and lower bounds of a probability interval. This interval contains the precise probability of the set S in the classical sense:

$$Bel(S) \leq P(S) \leq Pl(S)$$

Belief functions have several applications in uncertainty management (Dubois and Prade, 1985)(Dubois et al., 1998), particularly for the modeling of incomplete information. One of the biggest challenges of information and communication technologies is the modeling and processing of information. Different formalisms can be used, but belief function theory allows to combine imprecision and uncertainty. The information can take different forms (measures, data, knowledge...) with respect to the problems that are concerned.

Belief functions have given good results in the domains of pattern recognition and information fusion (T. Denoeux, 2012), (F. Pichon, 2012).

This theory is flexible enough to carry out the modeling of several different domains of information and communication technologies, such as data analysis, diagnosis, decision support or image processing (S. Petitrenaud, 2004)(Klir, 2006).

This framework is also used for reliability analysis of systems with data uncertainties and failure dependencies, for components with multiple states (Sallak et al., 2008) (Sallak et al., 2010). This approach consisting in evaluating reliability with belief functions through the Transferable Belief Model (TBM) has been compared to a classical Monte-Carlo approach (Aguirre et al., 2011). It turns out that this method has several advantages as compared to Monte-Carlo simulation. First of all the propagation of both types of uncertainty: variability and epistemic uncertainty, without having to select a precise probability distribution for the modeling of epistemic uncertainty. It also prevents from adding artificial information in the reliability computations, and allows the fusion of knowledge from different experts, in case of conflicting information.

Belief functions are a very powerful framework when it is necessary to deal with information that is heterogeneous, comes from different sources, or from various frames of discernment.

4.1.3 Imprecise probabilities

Classical probability requires a very high level of precision and consistency of information, and thus it is often too restrictive to cope carefully with the multi-dimensional nature of uncertainty. Perhaps the most straightforward restriction is that the quality of underlying knowledge cannot be adequately represented using a single probability measure. An increasingly popular and successful generalization is available through the use of *lower* and *upper* probabilities.

Definition 4.5 (Upper and lower probabilities)
Upper and *lower* probabilities denoted by $\underline{P}(A)$ and $\overline{P}(A)$ respectively, are s.t. $0 \leq \underline{P}(A) \leq \overline{P}(A) \leq 1$.

The special case with $\underline{P}(A) = \overline{P}(A)$ for all events A provides precise probability, while $\underline{P}(A) = 0$ and $\overline{P}(A) = 1$ represents complete ignorance about A.

Definition 4.6 (Upper and lower probability envelopes)
Let $\mathcal{P}$ be a probability family on Ω. The lower and upper probability envelopes are defined by:

$$P_*(E) = inf_{P \in \mathcal{P}} P(E)$$

$$P^*(E) = sup_{P \in \mathcal{P}} P(E)$$

♢

Upper and lower probability envelopes are the convex disclosure for a probability family. They also are referred by the name *coherent upper and lower probabilities* (Walley, 1999).

The framework of imprecise probabilities introduces the concept of *lower* and *upper previsions*, but for that, we need first to introduce the concept of *gambles* (T. Augustin, 2014).

Let Ω be a probability space.

Definition 4.7 (Gamble)
A *gamble* is a bounded mapping from Ω to $\mathbb{R}$.

In the case of probabilities, we are indifferent between betting on events or on gambles: our betting rates on events (a probability) determine our betting rates on gambles (its expectation).

Definition 4.8 (Lower and Upper previsions)
The *lower prevision* for a gamble f , $\underline{P}(f)$, is our supremum acceptable *buying* price for f , meaning that we are disposed to buy it for $\underline{P}(f) - \epsilon$ (or to accept the reward $f - (\underline{P}(f) - \epsilon)$) for any $\epsilon > 0$.

The *upper prevision* for a gamble f , $\overline{P}(f)$, is our infemum acceptable *selling* price for f , meaning that we are disposed to sell it for $\overline{P}(f) + \epsilon$ (or to accept the reward $\overline{P}(f) + \epsilon - f$) for any $\epsilon > 0$.♢

Proposition 4.1 There is a relationship between lower and upper previsions:

$$\underline{P}(-f) = -\overline{P}(f)$$

In the imprecise case, the lower and upper previsions for events do not determine the lower and upper previsions for gambles uniquely. Lower and upper previsions are more informative than lower and upper probabilities.

Imprecise probabilities provide new methods with greater flexibility than probability theory for uncertain quantification. Its advantages include the possibility to deal with conflicting evidence, to base inferences on weaker assumptions than needed for precise probabilistic methods, and to allow for simpler and more realistic relevance with subjective information (F. Coolen, 2010).

4.2 Different types of independence

Uncertainty propagation in a model with several parameters always has to take into account the dependence or independence of these variables. For variables that represent the observation of some

phenomena, there are two different levels of dependence. First, the dependence between the sources of information about the variables (coming from the observers), and second, the dependence between the variables themselves (coming from the phenomena).

In classical probability theory, the notion of independence between two events captures the fact that the frequency of occurrence of one of them has absolutely no impact on the frequency of occurrence of the other one. This idea can be formalized in the context of precise probabilities, but in the scope of other uncertainty management framework, some other definitions can be discussed.

4.2.1 Stochastic independence

The name of *stochastic independence* will refer to the independence of two events in the context of precise probability theory.

Definition 4.9 (Stochastic independence)
Let X and Y be two random variables, F_X and F_Y their marginal cumulative probability distribution, and $F_{X,Y}$ their joint cumulative probability distribution i.e. $P(Y \leq x) = F_X(x)$, $P(Y \leq y) = F_X(y)$ and $P(Y \leq y) = F_X(y)$.

X and Y are independent is equivalent to $F_{X,Y}(x, y) = F_X(x)F_Y(y) \forall (x, y)$. ♢

Proposition 4.2 Stochastic independence for two events A and B is equivalent to:

$$P(A \cap B) = P(A) \times (B)$$

Several other formalisms can be used in order to define this concept, using probability density distribution or mathematical expectation. In the case of dependency between the variables, the notion of copulas can be used in order to model those dependences (Nelsen, 1998).

As far as classical probability theory is concerned, the notion of independence is unique. But once we consider extended frameworks in uncertainty management, this notion can have several interpretations and be slightly more complex. In the following, we will present different notions of independence in imprecise probability, that are recognized and presented in the work of I. Couzo et al. (Couso et al., 2000). We will consider the random variables X and Y, and their imprecise probabilities P^X and P^Y. P^{XY} is the probability measure associated to the random vector (X, Y).

4.2.2 Strong independence

The strong independence has been introduced in the context of imprecise probabilities, described in section 4.1.3.

Definition 4.10
Let $\mathcal{P}_X$ and $\mathcal{P}_Y$ be two probabilities families for two random variables X and Y. X and Y are *strongly independent* when all the joint probabilities are under the form $P^{XY} = P^X \times P^Y$. In other terms:

$$\mathcal{P}_{(X,Y)} = \{P^X \times P^Y,\ s.t. P^X \in \mathcal{P}_X \text{ and } P^Y \in \mathcal{P}_Y =\} = \mathcal{P}_X \otimes \mathcal{P}_Y$$

This strong independence assumption has some applications in probabilistic logic for example. In (Cozman et al., 2006), F.G. Cozman et al. study inference methods based on polynomial programming are presented for strong independence for unconditional and conditional cases. The resulting model generalizes Bayesian networks, allowing probabilistic assessments and logical constraints to be mixed. It is also possible to compute lower and upper expectations under strong independence assumption (de Campos and Cozman, 2005).

4.2.3 Epistemic independence

Epistemic independence is an extension of the independence notion to probability measures. It represents the fact that the knowledge about the value of X has no impact on the knowledge about the value of Y, even if the value of X may have impact on the value of Y and vice-versa.

In that case, the probability family containing the joint probability is:

$$\mathcal{P}^{XY} = \{P^{XY} \ s.t. \ P^{X|Y} \in \mathcal{P}^X, P^{Y|X} \in \mathcal{P}^Y\},$$

where $\mathcal{P}^X$, $\mathcal{P}^Y$ and $\mathcal{P}^{XY}$ represent the probability measures modeling X, Y and the random vector (X, Y).

Some work has been done in order to determine how to compute upper and lower expectations under epistemic independence in the framework of imprecise probabilities (C.P. de Campos, 2007).

4.2.4 Independence of information sources

The independence of information sources consists in a stochastic independence between the focal elements in Dempster-Shafer theory (Couso and Moral, 2010). In this case, the probability distributions on the mass function of the focal elements are stochastically independent, but nothing is known about the relationship between X and Y inside the focal elements. It is basically an assumption about the independence of the sources where the information comes from (such as independent experts, or independent statistical studies) but not about the dependence between the variable themselves.

Some other definitions of independence exist, like *Complete independence*, *Conformational independence*, or *Kuznetsov's Independence* (Cozman, 2003). They all are defined in the scope of imprecise probabilities, and have several applications, depending on the type of model that is needed.

In reliability studies, the classical theory of probability is used in most of the cases for the quantitative analysis, even if the question of independence and dependences between the variables is already raised (Mazars, 1981). Once extended to imprecise probability, the question of the kind of (in)dependence linking the variables and/or the observers gets even more tricky, and their modeling can add complexity to the problem.

4.3 Uncertainty management in Reliability

Utkin and Coolen (Utkin and Coolen, 2007), for example, worked on imprecise reliability using imprecise probability theory, with upper and lower expectations instead of a single probability value (Guth, 1991). They studied imprecise monotonic fault trees, and also the impact of some components failure over the system under study by mean of *imprecise importance measures*. Some work has also

been done in order to apply a framework of joint propagation of probability distributions and fuzzy numbers to risk analysis in fault trees (Zio and Baraldi, 2008).

4.3.1 Interval reliability with partially known probabilities

Precise system reliability measures can always be computed if and only if it is assumed that (a) the system structure is defined precisely and (b) there exists a function linking failure time of the system and failure times of the components. In all other cases only interval reliability measures can be obtained. In reality, it is difficult to fulfill the first condition. For example, for a newly developed system or a system under development, statistical data is not sufficient or non-existent. Also in these cases system stability is not observed from a statistical point of view. In short, in such cases we only have partial information about system components.

The important question is how to trust the obtained results of reliability analysis if our assumption is based only on experts experience. In this regard, many methods have been developed to compute reliability bounds from initial incomplete information (Utkin, 2004). The idea presented in this paper is to model the imprecision of the system by sets of possible probability distributions. These probability distributions represent times to failure which are consistent with the available information. After modeling the imprecision with these sets, lower and upper reliability measures are computed.

4.3.2 Interval analysis for importance measures

In the work described in paper (Utkin and Coolen, 2007), the reliability assessments that are combined to describe systems and components may come from various sources. Some of them may be objective measures based on relative frequencies or on well-established statistical models. A part of the reliability assessments may be supplied by experts. If a system is new or exists only as a project, then there are often insufficient statistical data on which we should base precise probability distributions. Even if such data exist, we do not always observe their stability from the statistical point of view. Moreover, failure times may not be accurately observed or may even be missing. Sometimes, failures do not occur at all or occur partially; leading to censored observations of failure times, and the censoring mechanisms themselves may be complex and not precisely known. As a result, only partial information about reliability of system components may be available, for example, the MTTF or bounds for the probability of failure at a given time point.

4.3.3 Joint propagation of probability distributions and fuzzy numbers

In order to cope with different facets of uncertainty, a framework of joint propagation of probabilistic and possibilistic data has been developed by Cédric Baudrit and applied to risk analysis (Baudrit et al., 2006)(Baudrit et al., 2007). The principle of this framework is to add a subjective probability about the objectives probabilities.

It also has been applied to event trees and fault trees (Zio and Baraldi, 2008). It consists in modeling the uncertainty on the distribution parameters of each component by:

- a probability distribution, which will be a meta-probability,
- a fuzzy number if the information comes from some expert knowledge.

This joint propagation is computed with an algorithm based on Monte-Carlo simulation and gives some upper and lower bounds for the failure probability of the system. This technique allows the simultaneous propagation of both variability and epistemic uncertainty. As far as probabilistic variables are concerned, it is quite similar to a classical Monte-Carlo simulation, and the assumptions about the stochastic (in)dependence of these variables must hold. But, these variables should be independent from the one that are represented by the fuzzy numbers. A framework has also been developed in order to process the results obtained with this joint propagation, in order to be able to use it for decision making (Purba et al., 2010).

The following chapters will deal with the relaxation of these assumptions in fault tree analysis. However, some previous work has been carried out by other authors to drop these assumptions.

4.4 Fault trees under incomplete probabilistic information: general framework

Let F be a Boolean formula expressed by means of the variables X_i and $[F]$ the set of minterms of F. The probability of $F, P(F)$, can be written as the sum:

$$P(F) = \sum_{\omega \in [F]} p(\omega) \tag{4.1}$$

where $p(\omega)$ stands for $P(\{\omega\})$.

In the case where $P(A_i)$ is only known to lie in an interval, i.e. $P(A_i) \in [l_i, u_i], i = 1 \dots n$, the problem is to compute the tightest range $[l_F, u_F]$ containing the probability $P(F)$. Let $\mathscr{P}$ be the convex probability family $\{P, \forall i \ P(A_i) \in [l_i, u_i]\}$ on Ω. In the following, we shall formally express this problem under various assumptions concerning independence.

Without knowing the dependency between the variables of a fault tree $X_i, i = 1 \dots n$, finding the tightest interval $[l_F, u_F]$ of the probability $P(F)$ of the failure condition described by this fault tree boils down to a *linear optimization problem under constraints*. This goal is achieved by solving the following two problems:

$$l_F = \min(\sum_{\omega \models F} p(\omega)) \text{ and } u_F = \max(\sum_{\omega \models F} p(\omega))$$

under the constraints $l_i \leq \sum_{\omega \models A_i} p(\omega) \leq u_i, i = 1 \dots n$ and $\sum p(\omega) = 1$.

Solving each of those problems can be done by linear programming with 2^n unknown variables $p(\omega)$. It is a particular case of the probabilistic satisfiability problem studied in (Hansen et al., 2000), where known probabilities are attached to *sentences* instead of just atoms.

In the case where the independence of the $X_i, i = 1 \dots n$, is assumed,

$$p(\omega) = \prod_{i=1}^{n} P(X_i(\omega)) \tag{4.2}$$

where: $X_i(\omega) = \begin{cases} A_i \ if \ \omega \models A_i \\ \neg A_i \ otherwise \end{cases}$.

When the independence of the x_i's is assumed (i.e. A_i independent of A_j, $\forall i \neq j$), this sum becomes:

$$P(F) = \sum_{\omega \in [F]} [\prod_{A_i \in \mathcal{L}_\omega^+} P(A_i) \prod_{A_i \in \mathcal{L}_\omega^-} (1 - P(A_i))] \tag{4.3}$$

where $\mathcal{L}_\omega^+$ is the set of positive literals of ω and $\mathcal{L}_\omega^-$ the set of its negative literals.

The corresponding probability family $\mathscr{P}_I = \{\prod_{i=1}^{n} P_i \mid P_i(\{A_i\}) \in [l_i, u_i]\}$, where P_i is a probability measure on Ω_i, is not convex. Indeed, take two probability measures P, $P' \in \mathscr{P}_I$, $P = \prod_{i=1}^{n} P_i$ and $P' = \prod_{i=1}^{n} P'_i$. For $\lambda \in [0, 1]$, the sum $\lambda \prod_{i=1}^{n} P_i + (1 - \lambda) \prod_{i=1}^{n} P'_i \neq \prod_{i=1}^{n} (\lambda P_i + (1 - \lambda) P'_i)$, so it is not an element of $\mathscr{P}_I$.

This assumption introduces some non-linear constraints in the previous formulation, hence the previous method cannot be applied. Instead of a linear problem with 2^n variables, now we have a non-linear optimization problem with n variables. This problem can be solved by using Interval Analysis, it will be discussed in Chapter 5.

When there is no knowledge about the dependency between the x_i's, but the information about $P(A_i)$ comes from independent sources, *belief functions* can be used to solve the problem of probability evaluation. The information $P(A_i) \in [l_i, u_i]$ is totally linked to its source. l_i can be seen as the degree of belief of A_i and u_i as its plausibility: $l_i = Bel(A_i)$ and $u_i = Pl(A_i)$ in the sense of Shafer. In chapter 6, we will see how belief functions can be used to evaluate the probability of a Boolean formula. The application of belief functions to Boolean formulas will depend on the structure of the Boolean formula, we will see that for some particular cases, the results can be easily achieved, while the general case can be complex. A formalization of these cases is proposed.

Part III

Contribution to uncertainty management for complex Boolean systems

Chapter 5

Case of stochastic independence: interval analysis approach

Contents

In Chapter 3, we have seen the basics of fault tree analysis and how to compute probabilities from a given fault tree. These computations assume that all the probabilities are precisely known, which is not always possible in real life scenarios. The uncertainty management theories presented in Chapter 4 provides frameworks to deal with some uncertainty in such problems. The choice of a framework will be dictated by different assumptions, e.g. it is possible to have different interpretations of the independence of the Boolean variables. In this thesis, focus will mainly be on two main independence assumptions: *stochastic independence* and *independence of information sources*.

Computing the probability of a Boolean formula when the events are assumed to be stochastically independent, and when their probabilities are lying into intervals, is a non-linear problem with n variables. It consists of finding an interval by minimizing and maximizing the probability $P(F)$ of the Boolean formula. For the purpose of finding the bounds of a continuous function from the bounds of its variables, Interval analysis has been developed. The following section will present its main principles. This Chapter was the subject of the paper (Jacob et al., 2011), and inspired three other papers as well (Kreinovich et al., 2011a) (Kreinovich et al., 2011b) (Kreinovitch et al., 2012).

5.1 Interval analysis

Interval analysis is a method developed by mathematicians since the 1950s and 1960s (R.E. Moore, 2009) as an approach to put bounds on rounding errors and measurement errors in mathematical computation. It can also be used to represent some lack of information. So, instead of using a precise value for a probability P which is not exactly known, lower and upper bounds $\underline{P}$ and $\overline{P}$ can be estimated, such that $P \in [\underline{P}, \overline{P}]$.

The main idea of interval analysis is to find the upper and lower bounds, $\overline{f}$ and $\underline{f}$, of a function f of n variables $\{x_1, ..., x_n\}$, knowing the intervals of the variables: $x_1 \in [\underline{x_1}, \overline{x_1}]$, ...,$x_n \in [\underline{x_n}, \overline{x_n}]$.

5.1.1 Interval arithmetic, naive approach

The basic operations of naive interval arithmetic are, for two intervals $[\underline{a}, \overline{a}]$ and $[\underline{b}, \overline{b}]$ with $\underline{a}, \overline{a}, \underline{b}, \overline{b} \in \mathbb{R}$ and $\overline{a} \geq \underline{a}, \overline{b} \geq \underline{b}$.

$$Addition : [\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}] \tag{5.1}$$

$$Subtraction : [\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}] \tag{5.2}$$

$$Multiplication : [\underline{a}, \overline{a}] \times [\underline{b}, \overline{b}] = [min(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}), max(\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b})] \tag{5.3}$$

$$Division : \frac{[\underline{a}, \overline{a}]}{[\underline{b}, \overline{b}]} = [min(\frac{\underline{a}}{\underline{b}}, \frac{\underline{a}}{\overline{b}}, \frac{\overline{a}}{\underline{b}}, \frac{\overline{a}}{\overline{b}}), max(\frac{\underline{a}}{\underline{b}}, \frac{\underline{a}}{\overline{b}}, \frac{\overline{a}}{\underline{b}}, \frac{\overline{a}}{\overline{b}})] \tag{5.4}$$

The equation of multiplication and division can be simplified for the intervals included in [0,1]:

$$Multiplication : [a, b] \times [c, d] = [ac, bd]$$

$$Division : \frac{[\underline{a}, \overline{a}]}{[\underline{b}, \overline{b}]} = [\frac{\underline{a}}{\overline{b}}, \frac{\overline{a}}{\underline{b}}]$$

.

It is also possible to compute interval arithmetic on algebraic operations like square, square root, logarithm and exponential functions:

$$Square : [\underline{a}, \overline{a}]^2 = \begin{cases} [\underline{a}^2, \overline{a}^2] \text{ if } \underline{a} \geq 0 \\ [0, max(\underline{a}^2, \overline{a}^2)] \text{ if } \underline{a} \leq 0 \leq \overline{a} \\ [\overline{a}^2, \underline{a}^2] \text{ if } \overline{a} \leq 0 \end{cases} \tag{5.5}$$

$$Square\ root : \sqrt{[\underline{a}, \overline{a}]} = [\sqrt{\underline{a}}, \sqrt{\overline{a}}] \text{ iff } \underline{a} \geq 0 \tag{5.6}$$

$$Logarithm : \ln [\underline{a}, \overline{a}] = [\ln \underline{a}, \ln \overline{a}] \tag{5.7}$$

$$Exponential : e^{[\underline{a}, \overline{a}]} = [e^{\underline{a}}, e^{\overline{a}}] \tag{5.8}$$

The major limitation in the application of interval arithmetic to more complex functions is the *dependency problem*. Despite the similarity in the name, this is a different case then the stochastic independence assumption that is discussed in Chapter 4.

Definition 5.1 (Dependency problem)
The dependency problem comes from the repetition of a same variable in the expression of a function. It causes some difficulties to compute the exact range of the function, when it is not monotonic.

It must be pointed out that the dependency here is *logical* and not a *stochastic*, assumption made in section 4.2.1 leading to formula 3.5. It is only depending on the structure of the Boolean formula itself, and not about the relationship between the variables and/or observers, unlike the types of (in)dependences discussed in section 4.2.

Let us take a simple example in order to illustrate this tricky problem.

EXAMPLE 5.1 We consider the function $f(x) = x^2$, with $x \in [-1, 1]$. If we write the function as $f(x) = x^2$ and we apply the equations 5.5, we get that $f(x) \in [0, 1]$.

If we write the function as $f(x) = x.x$ and we apply the equation 5.4, then we get a too wide result, that is $[-1, 1]$. When using interval arithmetic, the expression of the function has influence on the obtained boundaries, while it should not be the case. Fig. 5.1 displays the difference between the two different ranges. The range obtained with the second expression is larger than the real range of the function, it adds an artificial uncertainty to the results.

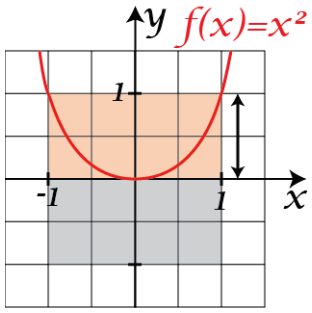


Figure 5.1: $f(x) = x^2$, with $x \in [-1, 1]$

The reason of this problem comes from the fact that this calculation considers $x^2 = x.x$ as if there were two different variables x and $y \in [-1, 1]$. It is equivalent to calculate the boundaries of a function $g(x, y) = x.y$, while f is a function of one variable and not two. Fig. 5.2 displays the difference between the two different ranges. ♢

EXAMPLE 5.2 Let us take a second example. If we consider the function $f(x) = x^2 + x$, displayed on Fig. 5.2. When we apply the formulas (5.5) and (5.1), we get:
$f(x) \in [-1, 1]^2 + [-1, 1] = [0, 1] + [-1, 1] = [-1, 2]$. But once again, this is larger than the real range of the function that is $[-\frac{1}{4}, 2]$.

In fact, f can be factorized such that variable x only appears once, f can be written as: $f(x) = (x + \frac{1}{2})^2 - \frac{1}{4}$. Then we can obtain the real range of the function. So one way to solve the dependency problem in interval arithmetic is to be able to factorize the expressions of the functions in a such a way that no variable appear more than once. But this is not always possible, or easy to do.

This dependency problem is an important issue when we want to apply interval analysis in Boolean formulas. The goal is to compute the tighter range for the probability of a formula, knowing the probability intervals of its variables.

5.1.2 Monotonicity and interval analysis

PROPOSITION 5.1 Let us consider f a continuous monotonic real function. The boundaries of f are on the boundaries of its definition domain.

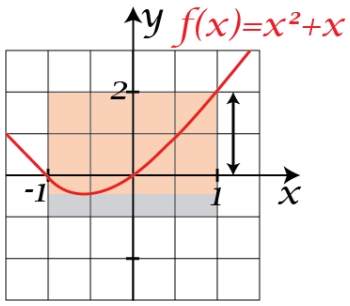


Figure 5.2: $f(x) = x^2 + x$, with $x \in [-1, 1]$

- When f is increasing on $[\underline{x}, \overline{x}]$, $f([\underline{x}, \overline{x}]) = [f(\underline{x}), f(\overline{x})]$,
- When f is decreasing on $[\underline{x}, \overline{x}]$, $f([\underline{x}, \overline{x}]) = [f(\overline{x}), f(\underline{x})]$. ♢

Proof This result can be proved from the extreme value theorem, which says that if a real-valued function f is continuous in the closed and bounded interval [a,b], then f must attain its supremum and infimum value, each at least once. That is, there exist numbers c and d in [a,b] such that: $f(c) \leq f(x) \leq f(d), x \in [a, b]$. ■

For the functions of more than one variables, different types of monotonicity need to be defined.

Definition 5.2 (Locally monotonic function)
A function f *locally monotonic* in relation to x_i, $i \in [\![1, ..., n]\!]$ if the function obtained by fixing all variables x_j but one is monotonic with respect to the remaining variable $x_i, i \neq j$ (Fortin et al., 2008).

Proposition 5.2 When a function f is locally monotonic and defined on a domain $[\underline{x_1}, \overline{x_1}] \otimes \cdots \otimes [\underline{x_n}, \overline{x_n}]$ the extrema of f are reached on the boundaries of the domain $[\underline{x_i}, \overline{x_i}]$ of the $x_i, i = 1, \ldots, n$. ♢

This particularity about locally monotonic functions induces that the range of the function can be founded by checking all the values of f at the boundaries of the domain $[\underline{x_1}, \overline{x_1}] \otimes \cdots \otimes [\underline{x_n}, \overline{x_n}]$, i.e, the images of the x_i.

Hence the range $f([\underline{x_1}, \overline{x_1}], ..., [\underline{x_n}, \overline{x_n}])$ can be obtained by testing the 2^n extreme values of the x_i's. If the function is monotonically increasing (resp. decreasing) with respect to x_j, the lower bound of this range is attained for $x_j = \underline{x_j}$ (resp. $x_j = \overline{x_j}$) and the upper bound is attained for $x_j = \overline{x_j}$ (resp. $x_j = \underline{x_j}$). The monotonicity study of a function is instrumental for interval analysis, and the concept of configuration of n intervals will be used for the cases where the monotonicity cannot be determined easily.

Definition 5.3 (Configuration of n intervals)
Given n intervals $[\underline{x_i}, \overline{x_i}], i = 1..n$, the n-tuple of values in the set $\mathcal{X} = \times_i \{\underline{x_i}, \overline{x_i}\}$ is called an *extreme configuration*.The extrema configurations $z_j, j = 1..2^n$, obtained by selecting for each value one interval end, define the set $\mathcal{H} = \times_i \{\underline{x_i}, \overline{x_i}\}$, where z_j has the form $(x_1^{c_1}, ..x_n^{c_n}), c_n \in \{0, 1\}$ with $x_i^0 = \underline{x_i}$ and $x_i^1 = \overline{x_i}$ and $\mid \mathcal{H} \mid = 2^n$.

5.2 Probability of a Boolean formula and interval analysis

In order to find the tightest range of a function, we will have to look into its structure. It is possible to determine precisely the range of a function that is monotonic from the range of its variables. Hence, the study of the monotonicity of the probability of a Boolean formula with respect to the probabilities of its variables is instrumental for its range computation.

Knowing the monotonicity of a function helps to determine its boundaries, and for some function the monotonicity can be easily found, that is the reason why in this section, we are interested in finding the monotonicity of the basics Boolean formulas. Studying the local monotonicity consists in studying the sign of the partial derivative with respect to each variable.

Example 5.3 (Monotonicity of usual Boolean formulas) The monotonicity of the probability of the most usual Boolean formulas is studied with respect to the probability of the Boolean variables.

Disjunction $\vee$

$F = A \vee B$
The probability associated to this formula is: $P(F) = P(A) + P(B) - P(A) \times P(B) = a + b - ab$
We can see P(F) is increasing with respect to a and b.

Conjunction $\wedge$

$F = A \wedge B$
The probability associated to this formula is: $P(F) = P(A) \times P(B) = ab$
The function is increasing with respect to a and b.

Implication $\Rightarrow$

$F = A \Rightarrow B = \neg A \wedge B$
The probability associated to this formula is: $P(F) = (1 - P(A)) + P(A) \times P(B) = 1 - b + ab$
$\frac{\partial}{\partial a} P(F) = b \geq 0$ because b is ≥ 0
$\frac{\partial}{\partial b} P(F) = a - 1 \leq 0$ because a is ≤ 1.
In conclusion, $P(F)$ is increasing with respect to a and decreasing with respect to b.

Equivalence $\Leftrightarrow$

$F = A \Leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$
The probability associated to this formula is: $P(F) = P(A) \times P(B) + (1 - P(A)) \times (1 - P(B)) = ab + (1 - a)(1 - b)$
Partial derivative of this function is same for a or b:
$\frac{\partial}{\partial a} P(F) = b - (1 - b) = 2b - 1 \geq 0$ when $b \geq \frac{1}{2}$. In this case, the function will be increasing with respect to a when $b \geq \frac{1}{2}$ and increasing with respect to be b when $a \geq \frac{1}{2}$.

Exclusive Or $\triangle$

$F = A \triangle B = (A \wedge \neg B) \vee (\neg A \wedge B)$
The probability associated to this formula is: $P(F) = P(A) \times (1 - P(B)) + (1 - P(A)) \times P(B) = a(1 - b) + b(1 - a)$
Partial derivative of this function is same for a or b:
$\frac{\partial}{\partial a} P(F) = 1 - 2b$ that is ≥ 0 when b is $\leq \frac{1}{2}$. Again here, the function will be increasing with respect to a when $b \leq \frac{1}{2}$ and increasing with respect to be b when $a \leq \frac{1}{2}$.

2 out of F_1, F_2, F_3

The function 2 out of F_1, F_2, F_3 is a

$F = (F_1 \wedge \neg F_2 \wedge \neg F_3) \vee (\neg F_1 \wedge F_2 \wedge \neg F_3) \vee (\neg F_1 \wedge \neg F_2 \wedge F_3)$
The probability associated to this formula is: $P(F) = P(F_1)(1 - P(F_2))(1 - P(F_3)) + P(F_2)(1 - P(F_1))(1 - P(F_3)) + P(F_3)(1 - P(F_1))(1 - P(F_2)) = f_1(1 - f_2)(1 - f_3) + f_2(1 - f_1)(1 - f_3) + f_3(1 - f_1)(1 - f_2)$

Partial derivative of this function is same for f_1, f_2 or f_3:
$\frac{\partial}{\partial f_1} P(F) = (1 - f_2)(1 - f_3) - f_2(1 - f_3) - f_3(1 - f_2) = 1 - f_2 - f_3 + 3f_2f_3$

It's more difficult to find the sign of such derivative, because we have a 2 variables function that has a saddle point, and whose range cannot be deduced with a straightforward expression.

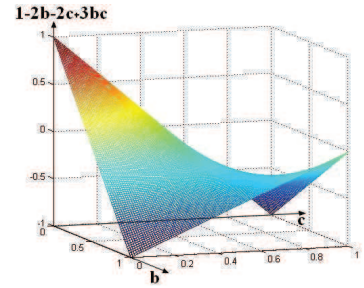


Figure 5.3: Partial derivative $1 - f_2 - f_3 + 3f_2f_3$

On the fig. 5.4 we can see the levels of the curve. The level 0 is in green, and the red part is the domain where the derivative is positive.

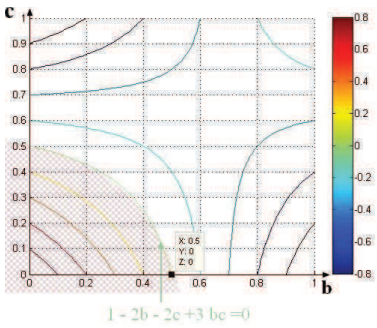


Figure 5.4: Level curve of partial derivative $1 - f_2 - f_3 + 3f_2f_3$

The derivative is null for $1 - f_2 - f_3 + 3f_2f_3 = 0 \Leftrightarrow f_2 = \frac{1-f_3}{1-3f_3}$, that is the equation of an hyperbola.

This means that the range of the function will be totally dependent on the value of the input probabilities. It will be as complex to determine and check a rule to find the range from any probability input, as to check the range of the function.

When the Boolean formula is monotone, and it is possible to prove it, the computation of its range is straightforward. But when it is not the case, and/or when the monotonicity is difficult to find, this computation can be more tricky.

5.3 BDDs and interval analysis

Applying interval analysis to BDDs is one approach for dealing with imprecise BDDs. It would permit us to be able to validate or invalidate some results even if we do not have all precise values of the probabilities in input, but we know their boundaries.

One important thing to notice about the formula (3.6) used to compute the probabilities from a BDD, is that it is *locally monotonic* 5.2. Let's consider f a function of n variables $\{x_1, ..., x_n\}$.

Let us consider the Shannon decomposition of a Boolean formula F for a variable X_i, $i \in [\![1, ..., n]\!]$. Formula (3.8)with $X = X_i$ can be written as: $P(F) = (1 - P(X_i)) \times P(F_{X_i=0}) + P(X_i) \times P(F_{X_i=1})$, where $P(X_i)$ appears twice. It can be written as:

$$P(F) = P(F_{X_i=0}) + P(X_i) \times [P(F_{X_i=1}) - P(F_{X_i=0})] \tag{5.9}$$

In order to study the local variations of the function $P(F)$ following $P(X_i)$, we fix all others X_j,$j \in [\![1, ..., n]\!]$, $j \neq i$; the partial derivative of equation 5.9 with respect to $P(X_i)$ is:

$$\frac{\partial P(F)}{\partial P(X_i)} = P(F_{X_i=1}) - P(F_{X_i=0})$$

$[P(F_{X_i=1}) - P(F_{X_i=0})]$ is a function of some $P(X_j)$, $j \neq i$, so it is constant according to $P(X_i)$, for any $i \in [\![1, ..., n]\!]$. The function $P(F)$ is locally monotonic with respect to all of its variables.

From these results, we can deduce that if $P(X_i) \in [a, b]$, $P(F)$ is in:

- $[P(f_{X_i=0})+a\times[P(f_{X_i=1})-P(f_{X_i=0}), P(f_{X_i=0})+b\times[P(f_{X_i=1})-P(f_{X_i=0})]$ if $P(f_{X_i=1}) \geq P(f_{X_i=0})$
- $[P(f_{X_i=0})+b\times[P(f_{X_i=1})-P(f_{X_i=0}), P(f_{X_i=0})+a\times[P(f_{X_i=1})-P(f_{X_i=0})]$ if $P(f_{X_i=1}) \leq P(f_{X_i=0})$

But finding the sign of $P(f_{X_i=1}) - P(f_{X_i=0})$ is not so simple, as it depends on some other X_j. If we are able to find this sign for each variable of the formula, we can find the exact interval of its values.

Another problem is that, as we saw before in section 5.1, one big issue of interval analysis is the dependency problem. And for computation of probabilities with BDDs, it will be very present. As explained in section 3.3.2, in the formula, there can be several instances of each variable, depending on the structure of the BDD.

Let us take an example to illustrate the dependency problem in a BDD. We will consider the formula $F = (A \wedge B) \vee (\neg A \wedge \neg B)$ whose BDD is represented on 5.5.

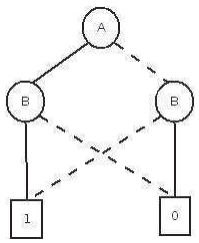


Figure 5.5: BDD associated to the formula $(A \wedge B) \vee (\neg A \wedge \neg B)$

The probability of f knowing that $P(A) = a$, $P(B) = b$, $P(\neg A) = 1 - a$ and $P(\neg B) = 1 - b$ is:

$$P(F) = a \times b + (1 - a)(1 - b) \tag{5.10}$$

In fact the values of a and b are not known, the only information available is that $a \in [\underline{a}, \overline{a}]$ and $b \in [\underline{b}, \overline{b}]$, with $[\underline{a}, \overline{a}]$ and $[\underline{b}, \overline{b}]$ included in $[0, 1]$.

The goal is to compute the interval associated to $P(f) = [\underline{f}, \overline{f}]$.

If we apply the equation of multiplication of two intervals ((5.3)) to expression, we obtain that:
$[\underline{f}, \overline{f}] = [\underline{ab}, \overline{a}\overline{b}] + (1 - [\underline{a}, \overline{a}])(1 - [\underline{b}, \overline{b}])$
$\Leftrightarrow [\underline{f}, \overline{f}] = [\underline{ab}, \overline{a}\overline{b}] + [1 - \overline{a}, 1 - \underline{a}] \times [1 - \overline{b}, 1 - \underline{b}]) \Leftrightarrow$

$$[\underline{f}, \overline{f}] = [\underline{ab} + (1 - \overline{a})(1 - \overline{b}), \overline{a}\overline{b} + (1 - \underline{a})(1 - \underline{b})] \tag{5.11}$$

Remark 5.1 This result is obviously wrong, because we use two different values for the same variable in the same expression: first we consider that $a = \underline{a}$, then that $a = \overline{a}$ and we do the computation.

Let us consider that events A and B are associated with the following intervals, respectively $[\underline{a}, \overline{a}] = [0.3, 0.8]$ and $[\underline{b}, \overline{b}] = [0.4, 0.6]$, using expression (5.11), we have that $[\underline{f}, \overline{f}] = [0.2, 0.9]$. By neglecting dependencies, we introduce a strong artificial uncertainty in the expression.

We know that the function is locally monotonic, so the boundaries of $P(F)$ will be obtained for the boundaries of the variables a and b. To find the correct bounds, we will have to explore the 2^2 configurations 5.3:

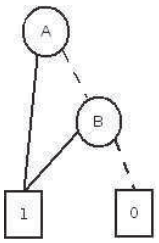
- $\underline{a} \times \underline{b} + (1 - \underline{a})(1 - \underline{b}) = 0.3 \times 0.4 + 0.7 \times 0.4 = 0.54$
- $\underline{a} \times \overline{b} + (1 - \underline{a})(1 - \overline{b}) = 0.3 \times 0.6 + 0.7 \times 0.4 = 0.46$
- $\overline{a} \times \underline{b} + (1 - \overline{a})(1 - \underline{b}) = 0.8 \times 0.4 + 0.2 \times 0.6 = 0.44$
- $\overline{a} \times \overline{b} + (1 - \overline{a})(1 - \overline{b}) = 0.8 \times 0.6 + 0.2 \times 0.4 = 0.56$

The result is $P(F) \in [0.44, 0.56]$, and we can notice that it is much tighter than the previous one. That is the reason why it is so important to find ways to optimize the results of interval analysis.

Example 5.4 (BDDs of usual formulas and range computation) As we have seen in section 3.4, there are different ways to compute the probabilities of a Boolean formula in function of the probabilities of its variables. Either we can compute it directly from the Boolean formula, or we can use the *Shannon decomposition* and compute this probability recursively, see equation (3.8).

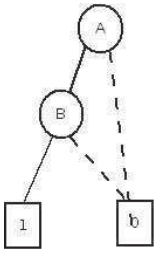
Disjunction $\vee$

$F = A \vee (\neg A \wedge B)$ from the BDD
The probability associated to this formula is:
$P(F) = P(A) + (1 - P(A))P(B) = a + (1 - a)b$
It is not obvious that $P(F)$ is increasing for a and b once the probability is computed from the BDD.



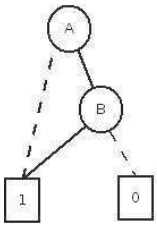
Conjunction $\wedge$

$F = A \wedge B$ from the BDD
The probability associated to this formula is:
$P(F) = P(A) \times P(B) = ab$
Here the BDD form is same than the original function, and it is obviously increasing for a and b.



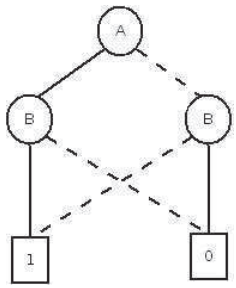
Implication $\Rightarrow$

$F = A \Rightarrow B = \neg A \vee B$
The probability associated to this formula is:
$P(F) = (1 - P(A)) + P(A) \times P(B) = 1 - b + ab$
Here the BDD form is same than the original function, hence the monotonicity determination is the same as in section 5.3.



Equivalence $\Leftrightarrow$

$F = A \Leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$
The probability associated to this formula is:
$P(F) = P(A) \times P(B) + (1 - P(A)) \times (1 - P(B)) = ab + (1 - a)(1 - b)$
Once again the BDD form is the same as the original function, hence the monotonicity determination is the same than in section 5.3.



♢

The fact that the BDD and the original form of the Boolean formula are same or not does not depend on the monotonicity of the Boolean formula. In the case of the disjunction, the addition of the negation of the variable A in the BDD makes the monotonicity of the formula less obvious than in the original formula. But for very big Boolean formulas, it is also possible for the BDD form to be much more simple than the formula itself, so in that case, it is possible that the monotonicity will be easier to study on the BDD form.

Moreover, every different expression of the same Boolean formula can make the monotonicity easier to determine for some variables and not for others. For example, if we write the disjunction as $F = A \vee (\neg A \wedge B)$, the monotonicity with respect to B is obvious, while if we write it $F = B \vee (\neg B \wedge A)$, now the monotonicity of A is trivial. For complex Boolean formulas, the more different equivalent expressions of the formula we get, the more chances there are for the determination of its monotonicity with respect to its different variables.

5.4 BDD-based interval analysis algorithm

An algorithm has been implemented in a generalised fault tree from the result of section 5.3, in order to compute the range of the probability of the top event. As seen in the previous sections, the input

of the algorithm is a Boolean function F where a probability interval $[v.lb, v.up]$ is associated to each variable X of F (the lower/upper bound of its probability interval).

The variable X of the BDD representation of F is characterized by the following additional attributes: *path.value* ($= 1$ if the variable appears as a positive literal in this path, $= 0$ otherwise) and *type* $\in \{0,1,2\}$. The output is a text file with the probability interval of F, $[P_F.lb, P_F.up]$. Fig. 5.6 summarizes the inputs and outputs of the algorithm.

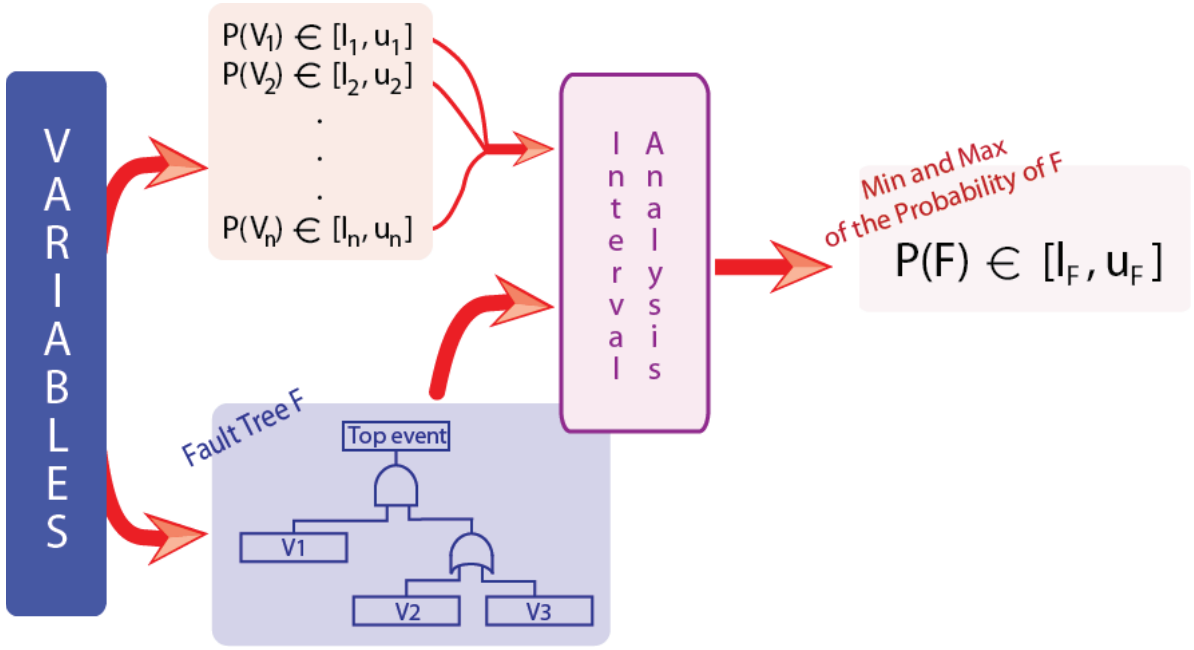


Figure 5.6: BDD-based interval analysis algo inputs/output

The BDD is represented by a set Pa of paths with the following format:

$$< X_1^{path_1} : value, \ldots, X_{n_1}^{path_1} : value > \cdots < X_1^{path_m} : value, \ldots, X_{n_m}^{path_m} : value > \quad (5.12)$$

where m is the number of paths and $n_i, i = 1, \ldots, m$ is the number of literals (positive or negative) in a path i. For example, $Pa = < A:1 >< S:0, C:1 >$ for the BDD represented in fig. 3.3.b.

In the next step, the variables are split into three categories:

– **Type 0:** Variables that only appear negatively in the Boolean formula
– **Type 1:** Variables that only appear positively in the Boolean formula
– **Type 2:** Variables present in the Boolean formula along with their negation.

We need to find the *configuration* that determines the minima and maxima for the probability of F.

We know the exact corresponding bounds of the input probabilities for the 2 first categories from section 5.3, so the optimal configuration is known for these variables:

– if $X.type = 0$, $v.ub$ is used for calculating $P_F.lb$ and $v.lb$ is used for $P_F.ub$,
– if $X.type = 1$, $v.lb$ is used for calculating $P_F.lb$ and $v.ub$ is used for $P_F.ub$,
– if $X.type = 2$, $P_F.lb$ (as well as $P_F.ub$) can be reached for $X.lb$ or $X.up$, and all possible extreme values for V must be explored. The total number of these tuples of bounds (configurations) is 2^k, where k is the number of variables classified as Type 2; hence the problem is at most NP-hard.

The last step consists of BDD-based calculations. These calculations are carried out by considering all m paths leading to leaf 1 from the top of the BDD.

Let $\vec{z^j} = (X_{1j}^{c_1}, \ldots, X_{nj}^{c_n}), j \in 1, \ldots, 2^n, c_i \in \{0,1\}$ be a configuration (section 5.3). For each configuration $\vec{z^j}$ we calculate $P_{\vec{z^j}}(F) = P(F[X_{1j}^{c_1}, \ldots, X_{nj}^{c_n}])$. The extremal values of $P(F$ are obtained by exploring all extremal configurations: $\underline{P(F)} = \min_{i=1,\ldots,2^n} P_{z_i}(F)$ and $\overline{P(F)} = \max_{i=1,\ldots,2^n} P_{z_i}(F)$.

The notation introduced in eq. (3.6) is extended to take the category of a variable into account. Let $\mathcal{X}_l$ be the set of variables belonging to the path l; $\mathcal{X}_l^{2+}$ (resp. $\mathcal{X}_l^{2-}$) the set of positive (resp. negative) literals of Type 2 in path l; $\mathcal{X}_l^{+}$ (resp. $\mathcal{X}_l^{-}$) the set of variables of Type 1 (resp. Type 0) in path l.

Let us consider now a configuration $(X_1^{c_1}, \ldots, X_k^{c_k})$ for variables of Type 2, with $X.c = X.lb$ if $X^c = 0$ and $X.c = X.ub$ if $X^c = 1$:

$$C_l(X_1^{c_1}, \ldots, X_k^{c_k}) = \prod_{X \in X_l^{2+}} X.c \cdot \prod_{X \in X_l^{2-}} (1 - X.c).$$

We have then:

$$\underline{P(F)} = \sum_{l=1,\ldots,m} \Big(\prod_{X \in \mathcal{X}_l^{+}} X.lb \cdot \prod_{X \in \mathcal{X}_l^{-}} (1 - X.ub) \cdot \min_{i=1,\ldots,2^k} C_l(X_1^{c_1}, \ldots, X_k^{c_k}) \Big) \text{ and}$$

$$\overline{P(F)} = \sum_{l=1,\ldots,m} \Big(\prod_{X \in \mathcal{X}_l^{+}} X.ub \cdot \prod_{X \in \mathcal{X}_l^{-}} (1 - X.lb) \cdot \max_{i=1,\ldots,2^k} C_l(X_1^{c_1}, \ldots, X_k^{c_k}) \Big)$$

that extends eq. (3.6) when the probability of the atomic events is given by an interval.

5.5 Example of the Primary/Backup Switch

In this case study, we consider the safety models as described in Chapter 2. We will take the example of a small reconfigureable system: a Primary/Backup Switch. It is constituted of three components:

- A primary supply
- A back-up supply
- A switch that selects the active supply between the primary or the backup one

When a fault occurs in the Primary supply, then it switches to the Back-up supply. But it may also happen that the Switch gets stuck: in this case, it will be impossible to switch to Backup supply.

The software Cécilia OCAS is used to model the architecture and the behavior of the system thanks to the AltaRica language, which is a mode-automata based language. From this description, some algorithms (Rauzy, 2002) will extract fault trees or Minimal Cut Sets for any undesired event selected by the user by means of observers. Fig.6.2 shows the OCAS model of the Primary/Backup Switch.

In the following, we will study the undesired event corresponding to the fact that the whole system is down, written as *Obs.KO*. The OCAS tool extracts the fault tree associated to this event, as displayed on Fig.6.2.

Therefore, this fault tree is equivalent to the Boolean formula:

$$Obs.KO = P.Fail \wedge (B.Fail \vee (S.stuck \wedge \neg S.activeB)),$$

where *P.Fail* stands for a failure of the Primary supplier and *B.Fail* for a failure of the Backup supplier. *S.stuck* represents the fact that the Switch is stuck and is not able to activate the Backup, and *S.activeB* is the activation order of the Switch.

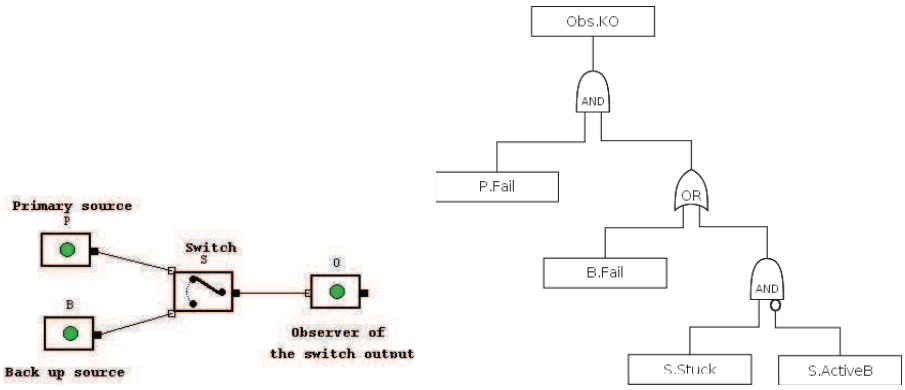


Figure 5.7: a) OCAS model of the Primary/Backup Switch b)Fault tree associated to event *Obs.KO*

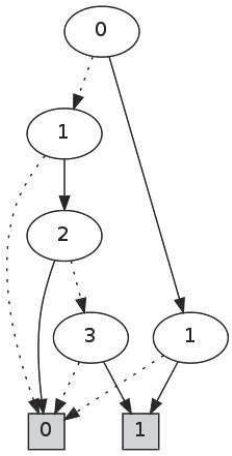


Figure 5.8: BDD of the Observer.KO for the Primary/Backup Switch

The BDD of this formula is represented on the Fig.5.8

The total paths extracted from this BDD are just 2: $< B.Fail : 0, P.Fail : 1, S.activeB : 0, S.stuck : 1 >< B.Fail : 1, P.Fail : 1 >$

In this case study, the values taken for the interval probabilities associated to the events are:

- *P.Fail* has an imprecise probability of $P(P.Fail) = 10^{-4} + / - 50\%$
- *B.Fail* has an imprecise probability of $P(B.Fail) = 10^{-4} + / - 10\%$
- *S.stuck* has a precise probability of $P(S.stuck) = 10^{-5}$
- An activation order of the Switch is given, i.e. $P(S.activeB) = 1$

The results of the computation of the range of the probability of the event $Obs.KO$ is $[4.5\times10^{-9}, 1.65\times10^{-8}]$. In this particular case, a computation with the naive interval arithmetic would lead to the same result, because the expression $Obs.KO = P.Fail \wedge (B.Fail \vee (S.stuck \wedge \neg S.activeB))$ does not include any repetition of variables, but it is not the case for general Boolean formulas.

When naive interval computations are applied directly to complex Boolean formula, it is highly probable to have several repetitions of one variable along with its negation, and the resulting interval is too imprecise and sometime totally useless. Moreover this imprecision is artificially induced by the computations, which is undesirable in the context of uncertainty management. The algorithm presented in this chapter allows to compute the exact range of the probability of any formula, without

adding any artificial uncertainty. The price to pay is an highest complexity for the computations. We pointed out that there are two cases when a variable and its negation appear in a Boolean expression: the case when there exists an equivalent expression where each variable appear with the same sign, and its probability is then monotonic in terms of the probability of atomic events; the case where such an equivalent expression does not exist. Then this probability will not be monotonic in terms of some variables, and the interval computation has NP-hard theoretical complexity.

Even if in practical fault-trees the latter situation does not prevail, the potentially exponential computation time can make it hardly applicable to very big systems, hence some heuristics will be needed to tackle this issue.

Chapter 6

Independent sources of information: Belief functions approach

Contents

When the probability information attached to the variables of a Boolean formula comes from independent sources of information, belief functions can be used in order to compute the aggregation of these information. This chapter investigates ways to compute the belief and the plausibility of a failure condition, when belief and plausibility of the contributing events are known, and are given by independent sources of information. It has been the subject of two articles, (Jacob et al., 2012a) and (F. Aguirre, 2013).

6.1 Belief and plausibility of a Boolean formula

In the case of Boolean variables, the frame of discernment introduced in section 4.1.2 contains only two elements $\Omega_i = \{A_i, \neg A_i\}$. These two elements are non trivial and disjoints, hence a probability interval $[l_i, u_i]$ is equivalent to a belief and plausibility measures (Bel, Pl) with $Bel \leq Pl$. For any event A_i, a source S_i of information gives a probabily interval $[l_i, u_i]$ that contain its probability. It is possible to find a unique atomic mass function m_i:

- $Bel(\{A_i\}) = l_i = m^i(\{A_i\})$;

- $Pl(\{A_i\}) = 1 - Bel(\{\neg A_i\}) = u_i \implies m^i(\{\neg A_i\}) = Bel(\{\neg A_i\}) = 1 - u_i$;

- The sum of masses is $m^i(\{A_i\}) + m^i(\{\neg A_i\}) + m^i(\Omega_i) = 1$, so $m^i(\Omega_i) = u_i - l_i$.

This modeling framework differs from the usual one when atomic variables are supposed to be stochastically independent. Here, the independence assumption pertains to the sources of information, not the physical variables.

When n sources of information are considered, the mass function over all Ω is : $m_\Omega = m^1 \oplus \cdots \oplus m^n$. In order to find this m_Ω for n atomic mass functions, we can use the associativity of Dempster rule of combination 4.3. Here, $A_i, i = 1, \ldots, n$ are atomic symbols, they are always compatible, i.e. $A_i \wedge A_j \neq \varnothing$ for all $A_i, A_j, i \neq j$. So the denominator is one in Dempster's equation.

In the case of a Boolean function constructed from n Boolean variables, then we need n sources. The mass function over all Ω is : $m_\Omega = m^1 \oplus \cdots \oplus m^n$.

A focal element (4.2) of m^Ω is made of a conjunction of terms of the form A_i, $\neg A_j$ and Ω_k (which is the tautology), for $i \neq j \neq k$. Hence it is a *partial model*. Let $\mathcal{P}(F)$ bet the set of partial models ϕ of a Boolean formula F, that are under the form of conjunction of elements $\lambda_i \in \{A_i, \neg A_i, \Omega_i\}$: $\phi = \bigwedge_{i=1,\ldots,n} \lambda_i$.

Then, let $\mathcal{P}(F) = \{\phi = \bigwedge_{A_i \in \mathcal{L}_\phi^+} A_i \bigwedge_{\neg A_i \in \mathcal{L}_\phi^-} \neg A_i \models F\}$, with $\mathcal{L}_\phi^+$ (resp. $\mathcal{L}_\phi^-$) the set of positive (resp. negative) literals of ϕ.

Proposition 6.1 (Combination of n atomic mass functions) For n atomic masses $m^i, i = 1, \ldots, n$ on Ω_i, the joint mass m^Ω on Ω can be computed as follows for any partial model ϕ:

$$m^\Omega(\phi) = \prod_{i \in \mathcal{L}_\phi^+} l_i \prod_{i \in \mathcal{L}_\phi^-} (1 - u_i) \prod_{i \notin \mathcal{L}_\phi} (u_i - l_i) \tag{6.1}$$

♢

The belief in a Boolean formula F can hence be written as:

$$Bel(F) = \sum_{\phi \models F} m_\Omega(\phi)$$

Its plausibility is:

$$Pl(F) = \sum_{S \wedge \phi \neq \varnothing} m_\Omega(\phi)$$

It theoretically requires 3^n computations due to the necessity of enumerating the partial models for n atomic variables. Indeed, all conjunctions $\phi = \bigwedge_{i=1,\ldots,n} \lambda_i$ must be checked for each $\lambda_i \in \{A_i, \neg A_i, \Omega_i\}$.

Verifying that a partial model implies F also requires 2^n computations. Plausibility computation requires to determine partial models not incompatible with F. From the partial models, it will need at most 2^n computations. But it can also be computed by using the duality of belief and plausibility given by:

$$Pl(F) = 1 - Bel(\neg F) \tag{6.2}$$

EXAMPLE 6.1 **Belief functions of the disjunction $F = A_1 \vee A_2$**

	A_1	$\neg A_1$	Ω_1
A_2	$A_1 \wedge A_2$ $l_1 l_2$	$\neg A_1 \wedge A_2$ $(1-u_1)l_2$	A_2 $(u_1-l_1)l_2$
$\neg A_2$	$A_1 \wedge \neg A_2$ $l_1(1-u_2)$	$\neg A_1 \wedge \neg A_2$ $(1-u_1)(1-u_2)$	$\neg A_2$ $(u_1-l_1)(1-u_2)$
Ω_2	A_1 $l_1(u_2-l_2)$	$\neg A_1$ $(1-u_1)(u_2-l_2)$	Ω $(u_1-l_1)(u_2-l_2)$

♢

Partial models that imply F are $\{A_1, A_2, A_1 \wedge \neg A_2, A_2 \wedge \neg A_1, A_1 \wedge A_2\}$, so:
$Bel(F) = (u_1-l_1)l_2 + l_1(u_2-l_2) + l_1 l_2 + l_1(1-u_2) + l_2(1-u_1) = l_1 + l_2 - l_1 l_2$, that also reads $1-(1-l_1)(1-l_2)$. Likewise, partial models that are compatible with F are $\{A_1 \wedge A_2, \Omega, A_1, A_2, \neg A_1, \neg A_2, A_1 \wedge \neg A_2, A_2 \wedge \neg A_1\}$, hence $Pl(F) = u_1 + u_2 - u_1 u_2 = 1-(1-u_1)(1-u_2)$.

In the case of a function of 3 variables, there will be 3^3 configuration to check. We can summarize them in a 3 dimension array of 3 columns, row and depth. In order to be able to represent it on the paper, it will be split in 3 arrays of dimension 2.

A_3	A_1	$\neg A_1$	Ω_1
A_2	$A_1 \wedge A_2 \wedge A_3$ $l_1 l_2 l_3$	$\neg A_1 \wedge A_2 \wedge A_3$ $(1-u_1)l_2 l_3$	$A_2 \wedge A_3$ $(u_1-l_1)l_2 l_3$
$\neg A_2$	$A_1 \wedge \neg A_2 \wedge A_3$ $l_1(1-u_2)l_3$	$\neg A_1 \wedge \neg A_2 \wedge A_3$ $(1-u_1)(1-u_2)l_3$	$\neg A_2 \wedge A_3$ $(u_1-l_1)(1-u_2)l_3$
Ω_2	$A_1 \wedge A_3$ $l_1(u_2-l_2)l_3$	$\neg A_1 \wedge A_3$ $(1-u_1)(u_2-l_2)l_3$	A_3 $(u_1-l_1)(u_2-l_2)l_3$

$\neg A_3$	A_1	$\neg A_1$	Ω_1
A_2	$A_1 \wedge A_2 \wedge \neg A_3$ $l_1 l_2(1-u_3)$	$\neg A_1 \wedge A_2 \wedge \neg A_3$ $(1-u_1)l_2(1-u_3)$	$A_2 \wedge \neg A_3$ $(u_1-l_1)l_2(1-u_3)$
$\neg A_2$	$A_1 \wedge \neg A_2 \wedge \neg A_3$ $l_1(1-u_2)(1-u_3)$	$\neg A_1 \wedge \neg A_2 \wedge \neg A_3$ $(1-u_1)(1-u_2)(1-u_3)$	$\neg A_2 \wedge \neg A_3$ $(u_1-l_1)(1-u_2)(1-u_3)$
Ω_2	$A_1 \wedge \neg A_3$ $l_1(u_2-l_2)(1-u_3)$	$\neg A_1 \wedge \neg A_3$ $(1-u_1)(u_2-l_2)(1-u_3)$	$\neg A_3$ $(u_1-l_1)(u_2-l_2)(1-u_3)$

Ω_3	A_1	$\neg A_1$	Ω_1
A_2	$A_1 \wedge A_2$ $l_1 l_2(u_3-l_3)$	$\neg A_1 \wedge A_2$ $(1-u_1)l_2(u_3-l_3)$	A_2 $(u_1-l_1)l_2(u_3-l_3)$
$\neg A_2$	$A_1 \wedge \neg A_2$ $l_1(1-u_2)(u_3-l_3)$	$\neg A_1 \wedge \neg A_2$ $(1-u_1)(1-u_2)(u_3-l_3)$	$\neg A_2$ $(u_1-l_1)(1-u_2)(u_3-l_3)$
Ω_2	A_1 $l_1(u_2-l_2)(u_3-l_3)$	$\neg A_1$ $(1-u_1)(u_2-l_2)(u_3-l_3)$	Ω $(u_1-l_1)(u_2-l_2)(u_3-l_3)$

6.1.1 Conjunctions and disjunctions of atoms

The case where Boolean formulas are only written with positive or negative atoms makes the computations easier.

PROPOSITION 6.2 (BELIEF OF A CONJUNCTION OF n ATOMS) The belief of a conjunction $C_A^+ = \bigwedge_{k\in K} A_k$ of n atoms $A_i, i \in K = [1, n]$ is given by the formula:

$$Bel(C_A) = \prod_{i=1}^{n} l_i \tag{6.3}$$

Similarly, the belief of a conjunction of negative atoms $C_A^- = \bigwedge_{k\in K} \neg A_k$ is given by the formula:

$$Bel(C_A) = \prod_{i=1}^{n} (1 - u_i) \tag{6.4}$$

♢

PROOF The conjunction of C_A atoms can be seen as one partial model, hence equation (6.1) gives:

$$\begin{aligned} Bel(C_A) &= Bel(\wedge_{k\in K} A_k) \\ &= m_\Omega(C_A) \\ &= \prod_{i=1}^{n} l_i \end{aligned}$$

PROPOSITION 6.3 (BELIEF OF A DISJUNCTION OF n ATOMS) The belief of a disjunction $D_A = \bigvee_{k\in K} A_k$ is a disjunction of n atoms $A_1 \dots A_n$ is given by:

$$Bel(D_A) = 1 - \prod_{i=1}^{n} (1 - l_i).$$

PROOF

$$\begin{aligned} Bel(D_A) &= Bel(\vee_{k\in K} A_k) \\ &= \sum_{\phi \models D_A} m_\Omega(\phi) \end{aligned}$$

The partial models that satisfies D_A are all the combinations of different conjunctions of the atoms A_k.

Making the sum of all of these combinations is very similar to the inclusion-exclusion principle Brualdi (2004), applied to the terms of the disjunctions. Hence, we get a result that is similar to a probability computation:

$$Bel(D_A) = \sum_{i=1}^{n} l_i - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} l_i l_j + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} l_i l_j l_k - \dots + (-1)^{n+1} l_1 \dots l_n \tag{6.5}$$

This formula can be factorized as:

$$Bel(D_A) = 1 - \prod_{i=1}^{n} (1 - l_i) \tag{6.6}$$

■

Once the belief of a formula F is computed, its plausibility can be deduced with the duality equation (6.2).

PROPOSITION 6.4 (PLAUSIBILITY OF A CONJUNCTION OF n ATOMS) The plausibility of a conjunction C_A of n atoms is given by:

$$Pl(C_A) = \prod_{i=1}^{n} u_i \tag{6.7}$$

♢

PROOF

$$Pl(C_A) = 1 - Bel(\neg C_A) = 1 - Bel(\neg A_1 \vee \cdots \vee \neg A_n)$$

By using equation 6.6, we get:

$$Pl(C_A) = 1 - [1 - \prod_{i=1}^{n}(1 - Bel(\neg A_i))] = \prod_{i=1}^{n} Pl(A_i)$$

■

PROPOSITION 6.5 (PLAUSIBILITY OF A DISJUNCTION OF n ATOMS) The plausibility of a disjunction D_A of n literals is given by:

$$Pl(D) = 1 - \prod_{i=1}^{n}(1 - u_i). \tag{6.8}$$

♢

PROOF

$$Pl(D) = 1 - Bel(\neg D) = 1 - Bel(\neg A_1 \wedge \cdots \wedge \neg A_n)$$

By using equation (6.4), we get:

$$Pl(D) = 1 - \prod_{i=1}^{n} Bel(\neg A_i) = 1 - \prod_{i=1}^{n}(1 - Pl(A_i))$$

■

It is also possible to generalize it to a formula written as a disjunction of conjunctions of atoms.

PROPOSITION 6.6 (BELIEF OF A DISJUNCTIVE ATOMIC NORMAL FORM)

$$Bel(C_A^1 \vee ... \vee C_A^m) = \sum_{i=1}^{m} Bel(C_i) - \sum_{i=1}^{m-1}\sum_{j=i+1}^{m} Bel(C_A^i \wedge C_A^j)$$
$$+ \sum_{i=1}^{m-2}\sum_{j=i+1}^{m-1}\sum_{k=j+1}^{m} Bel(C_A^i \wedge C_A^j \wedge C_A^k) - ... + (-1)^{m+1} Bel(C_A^1 \wedge ... \wedge C_A^m)$$

where C_A^i are conjunctions of atoms. ♢

During the computation, the conjunctions of conjunctions, such as $C_i \wedge C_j \wedge C_k$ must be simplified, deleting redundant atoms. Note that this apparent additivity of a generally non-additive function is due to the specific shape of focal elements (partial models). In general, for S and T Boolean formulas, we cannot write $Bel(S \vee T) = Bel(S) + Bel(T) - Bel(S \wedge T)$, because there are focal elements in

$S \vee T$ that are subsets of neither S nor T nor $S \wedge T$. Here due to the DANF form, all partial models of $C_1 \vee ... \vee C_m$ are conjunctions of literals appearing in the conjunctions.

A similar result holds for computing the plausibility of a DNF.

Plausibility of a disjunctive atomic normal form (DANF)

$$Pl(C_1 \vee ... \vee C_m) = \sum_{i=1}^{m} Pl(C_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} Pl(C_i \wedge C_j)$$
$$+ \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \sum_{k=j+1}^{m} Pl(C_i \wedge C_j \wedge C_k) - ... + (-1)^{m+1} Pl(C_1 \wedge ... \wedge C_m),$$

where C_i are conjunctions of atoms.

Thanks to the duality between belief and plausibility, both computations are quite similar, hence the time complexity does not increase.

6.1.2 Conjunctions and disjunctions of literals

In the more general case, we can compute the belief and plausibility of conjunctions and disjunctions of literals indexed by $K \subseteq \{1, \ldots, n\}$.

The belief of a conjunction $C = \bigwedge_{k \in \mathcal{L}_C^+} X_k \wedge \bigwedge_{k \in \mathcal{L}_C^-} \neg X_k$ of literals $X_i, i \in K$ is given by:

$$Bel(C) = \prod_{i \in \mathcal{L}_C^+} l_i \prod_{i \in \mathcal{L}_C^-} (1 - u_i) \tag{6.9}$$

Likewise the belief of a disjunction $D = \bigvee_{k \in \mathcal{L}_D^+} X_k \vee \bigvee_{k \in \mathcal{L}_D^-} \neg X_k$ is given by:

$$Bel(D) = 1 - \prod_{i \in \mathcal{L}_D^+} (1 - l_i) \prod_{i \in \mathcal{L}_D^-} u_i \tag{6.10}$$

We can deduce the plausibility of conjunctions and disjunctions of literals, noticing that

$$Pl(C) = 1 - Bel(D)$$

.

Proof

$$\begin{aligned} Pl(C) &= Pl(\wedge_{i \in \mathcal{L}^+} X_i \wedge \wedge_{i \in \mathcal{L}^-} \neg X_i) \\ &= \prod_{i \in \mathcal{L}^-} (1 - l^i) \prod_{i \in \mathcal{L}^+} u^i \\ &= 1 - (1 - \prod_{i \in \mathcal{L}^-} (1 - l_i) \prod_{i \in \mathcal{L}^+} u_i) \\ &= 1 - Bel(\vee_{i \in \mathcal{L}^-} \neg X_i \vee \vee_{i \in \mathcal{L}^+} X_i) \\ &= 1 - Bel(D) \end{aligned}$$

Hence, the plausibility of a conjunction C, and of a disjunction D of literals $X_i, i \in K$ are respectively given by:

$$Pl(C) = \prod_{i \in \mathcal{L}_C^+} (1 - l_i) \prod_{i \in \mathcal{L}_C^-} u_i;$$

$$Pl(D) = 1 - \prod_{i \in \mathcal{L}_D^+} l_i \prod_{i \in \mathcal{L}_D^-} (1 - u_i)$$

The general case of a Boolean formula with positive and negative literals is more tricky. Of course we can assume the formula is in DNF format. But it will be a disjunction of conjunctions of literals, not of atoms, and it is no longer possible to apply the propositions given in section 6.1.1. Indeed when conjunctions contain opposite literals, they have disjoint sets of models but their disjunctions may be implied by partial models (focal elements) that imply none of the conjuncts.

For instance consider $A \vee (\neg A \wedge B)$ (which is just the disjunction $A \vee B$ we know how to deal with). It does not hold that $Bel(A \vee (\neg A \wedge B) = Bel(A) + Bel(\neg A \wedge B)$, since the latter sum neglects $m(B)$, where B is a focal element that implies neither A nor $\neg A \wedge B$. However if $C_1 \vee ... \vee C_m$ are pairwise mutually inconsistent partial models such that no disjunction of C_i and C_j contains a partial model implying neither C_i nor C_j, the computation can be simplified since then $Bel(C_1 \vee ... \vee C_m) = \sum_{i=1}^{m} Bel(C_i)$. For instance, the belief in an exclusive OR $Bel((A_1 \wedge \neg A_2) \vee (A_2 \wedge \neg A_1))$ is of this form.

PROPOSITION 6.7 The set of partial models $\Phi_n = \{\phi_1, \ldots, \phi_n\}$ satisfies the inclusion/exclusion principle if and only if, for any pair ϕ_i, ϕ_j one of the two following conditions is satisfied:

- $\exists p \neq q \in \{1, \ldots, D\}$ s.t. $p, q \in (\mathcal{L}_\rangle^+ \cap \mathcal{L}_|^-) \cup (\mathcal{L}_|^+ \cap \mathcal{L}_\rangle^-)$.
- $\mathcal{L}_\rangle^+ \cap \mathcal{L}_|^- = \emptyset$ and $\mathcal{L}_|^+ \cap \mathcal{L}_\rangle^- = \emptyset$ ♢

This condition tells us that for any pair of partial models, :

- either conjunctions ϕ_i, ϕ_j contain at least two opposite literals,
- or events ϕ_i, ϕ_j have a non-empty intersection and have a common model.

PROOF Let us start with the "only if". Consider a pair $\{\phi_i, \phi_j\}$ that differs on at exactly one literal, and without loss of generality let us assume $\phi_p^i = X_p$ and $\phi_p^j = \neg X_p$. For any index $q \neq p$, consider the singletons $X_i \in \phi_i, X_j \in \phi_j$

- if $\phi_q^i = \phi_q^j = \neg A_q$, we set $X_q^i = X_q^j = \neg A_q$;
- if $\phi_q^i = \phi_q^j = A_q$, we set $X_q^i = X_q^j = A_q$;
- else if $\phi_q^i = \phi_q^j = \Omega_i$, we set (arbitrarily) $X_q^i = X_q^j$.

By construction, $X^i \in \phi^i, X^j \in \phi^j$ and X^i, X^j differs only on dimension Ω_p, meaning that $(X^i \vee X^j) \in \Omega$.

Let us now deal with the "if" part of the proof. The proof when ϕ_i, ϕ_j are disjoint and differs on two literals. The only way to have $\phi_i^q \cap \phi_j^q = \emptyset$ is for ϕ_i^q and ϕ_j^q to be two opposed literals.

Considering the second case (non-empty intersection), consider that for any event ϕ and for any dimension, we have either $\phi_q = X_q$, $\phi_q = \neg X_q$, $\phi_q = \Omega_q$. Now, as having $\phi_i \cap \phi_j = \emptyset$ for two events implies that for any dimension q, $\phi_q^i \cap \phi_q^j \neq \emptyset$, this also implies $\phi_q^i \subseteq \phi_q^j$ or $\phi_q^i \supseteq \phi_q^j$ (as if they are not the same singleton, then one of them is the whole space Ω_q) ■

These conditions allow us to check, once a formula has been put in DNF, whether or not the inclusion/exclusion principle applies. As seen in section 6.1.1, a particular case where it always applies are disjunctions of partial models having only positive (negative) atoms. More generally, the inclusion/exclusion principle applies to disjunctions of partial models which can, via a renaming, be rewritten as a disjunction of conjunctions of positive literals: namely, whenever a single variable never appears in a positive and negative form in two of the conjunctions.

Example 6.2 For instance consider the formula $(X^1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2) \vee X_3$, with $\phi_1 = X_1 \wedge \neg X_2$, $\phi_2 = \neg X_1 \wedge X_2$, $\phi_3 = X_3$.

We have $\mathcal{L}_1^+ = \{1\}, \mathcal{L}_1^- = \{2\}, \mathcal{L}_2^+ = \{2\}, \mathcal{L}_2^- = \{1\}, \mathcal{L}_3^+ = \{3\}, \mathcal{L}_2^- = \emptyset$. Thus it satisfies Proposition 6.7, and

$$
\begin{aligned}
Bel((X^1 \wedge \neg X_2) \vee (\neg X_1 \wedge X_2) \vee X_3) = \\
& Bel(X^1 \wedge \neg X_2) + Bel(\neg X_1 \wedge X_2) \\
& + Bel(X^3) - Bel(X^1 \wedge \neg X_2 \wedge X_3) - Bel(\neg X_1 \wedge X_2 \wedge X_3) \\
& \text{(other terms are 0)} \\
=& l_1(1-u_2) + (1-u_1)l_2 + l_3(1 - l_1(1-u_2) - (1-u_1)l_2)
\end{aligned}
$$

♢

Example 6.3 An example where the inclusion/exclusion principle cannot be applied, consider the formula $X_1 \vee (\neg X_1 \wedge X_2)$ (which is just the disjunction $X_1 \vee X_2$ we already considered above). It does not hold that $Bel(X_1 \vee (\neg X_1 \wedge X_2) = Bel(X_1) + Bel(\neg X_1 \wedge X_2)$, since the latter sum neglects $m(X_2)$, where X_2 is a focal set that implies neither X_1 nor $\neg_1 \wedge X_2$.

Note that this remark suggests that normal forms that are very useful to compute the probability of a Boolean formula efficiently may not be useful to speed up computations of belief and plausibility degrees. For instance, $X_1 \vee (\neg X_1 \wedge X_2)$ is a binary decision diagram for the disjunction, and this form prevents $Bel(X_1 \vee X_2)$ from being computed using the inclusion/exclusion principle. ♢

6.1.3 Inclusion exclusion principle for belief functions

This work, along with the work of F. Aguirre et al. (Sallak et al., 2012), determines the cases where inclusion exclusion results hold for more general cases other than Boolean formula. This is the main subject of work done in (). It particularly gives a framework for the case of monotonic functions, and of multi-states systems (i.e. functions that takes a finite set of values).

Although the framework that is retained in this article may seem restrictive at first, it can be applied to a number of practical situations, and that one particular application is the evaluation of system reliability is described.

Such results facilitate computations; and are particularly useful when using imprecise probabilistic models. An interesting perspective to this study is to look for conditions under which other models (e.g., general lower probabilities) satisfy the exclusion/inclusion principle. Other perspectives of the current study include looking at other problems of reliability analysis, such as importance measures used to detect critical components.

6.2 Belief functions and fault tree analysis

In this thesis, the guideline is the application of different uncertainty management theories to fault tree analysis. The results proved in section 6.1 applies here, as a fault tree can be written as a Boolean formula (monotonic, or not).

6.2.1 Belief and plausibility of a Minimal Cut Set

From a Fault-tree F, an approximation can be obtained by means of minimal cuts. For a given *order* (maximal number of atoms in conjunctions), appropriate software can find the set of all Minimal Cuts that lead to the top event. The disjunction of all those Minimal Cuts will give us a partial Fault-tree which will be an approximation of F. Fig. D.2 is an example of such a Partial Fault-tree.

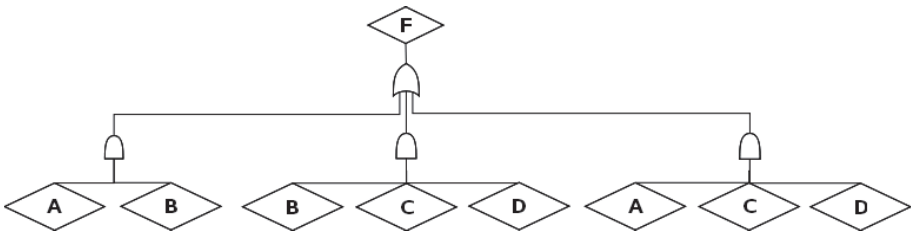


Figure 6.1: Example of Partial Fault Tree

The Boolean formula F represented by this tree will always be under the form of a disjunction of conjunctions of atoms $C_1 \vee ... \vee C_m$. The formula written in this form will be referred to as a Disjunctive Atomic Normal Form (DANF) (excluding negative literals). The results presented in section 6.1.1 makes the computation of the belief and the plausibility of the formula straightforward.

For non monotonic fault trees, as the one that can be automatically generated by Model Based Safety Analysis tools, the inclusion-exclusion principle will only hold if the Boolean formula associated to the fault tree satisfies the proposition 6.7.

6.2.2 Comparison between Interval Analysis and Dempster-Shafer Theory

Table 6.2.3 summarizes the results obtained using the two methods seen in section 5.3 and 6.1 applied to Boolean formulas: (i) the belief functions method with the assumption that the probability values come from independent sources of information, and (ii) the full-fledged interval analysis method under the assumption that all atomic events are independent. These two assumptions do not reflect the same kind of situations. In particular the independence between sources of information may be justified if elementary components in the device under study are different from one another, which often implies that the sources of information about them will be distinct. However the fact that such elementary components interact within a device tends to go against the statistical independence of their respective behaviors.

Those results are given for the basic Boolean operators with variables A, B, C and D. The probability interval used for those computations are: $P(A) \in [0.3, 0.8]$, $P(B) \in [0.4, 0.6]$, $P(C) \in [0.2, 0.4]$, and $P(D) \in [0.1, 0.5]$.

Table 6.1: Comparison between Interval Analysis and Dempster-Shafer Theory

Operator	Formula	Belief Functions (i)	Interval Analysis (ii)
OR	$A \vee B$	$l_F = l_A + l_B - l_A l_B = 0.58$ $u_F = u_A + u_B - u_A u_B = 0.92$	$l_F = l_A + l_B - l_A l_B = 0.58$ $u_F = u_A + u_B - u_A u_B = 0.92$
AND	$A \wedge B$	$l_F = l_A l_B = 0.12$ $u_F = u_A u_B = 0.48$	$l_F = l_A l_B = 0.12$ $u_F = u_A u_B = 0.48$
IMPLIES	$A \Rightarrow B$	$l_F = (1 - u_A) + l_B u_A) = 0.44$ $u_F = 1 - l_A(u_B - l_A) = 0.94$	$l_F = 1 - u_A + l_B u_A = 0.52$ $u_F = 1 - l_A + u_B l_A = 0.88$
ExOR	$A \triangle B$	$l_F = l_A(1 - u_B) + l_B(1 - u_A)$ $u_F = u_A + u_B - l_A l_B - u_A u_B$ $[l_F, u_F] = [0.2, 0.8]$	[0.44,0.56]
Fault-tree (Fig. D.2)	F	$l_F = l_A l_B + l_B l_C l_D + l_A l_C l_D - 2 l_A l_B l_C l_D$ $u_F = u_A u_B + u_B u_C u_D + u_A u_C u_D - 2 u_A u_B u_C u_D$ $[l_F, u_F] = [0.1292, 0.568]$	$l_F = l_A l_B + (1 - l_A) l_B l_C l_D + (1 - l_B) l_A l_C l_D$ $l_F = u_A u_B + (1 - u_A) u_B u_C u_D + (1 - u_B) u_A u_C u_D$ $[l_F, u_F] = [0.1292, 0.568]$

In Interval Analysis, knowing the monotonicity of a formula makes the determination of its range straightforward. A Boolean formula is *monotonic* with respect to a variable when we can find an expression of the formula where this variable appears only in a positive or negative way. It is the case for the formulas *And*, *Or*, and *Implies*.

But when the monotonicity is not easy to study, an exhaustive computation for all intervals boundaries must be carried out, like for the *Equivalence* and the *Exclusive Or*.

The difference between the results varies a lot with the formula under study. Sometimes, using the Dempster-Shafer theory give the same results as interval analysis, hence, in those cases, the dependency assumption does not have a big influence on the output value; e.g in case of conjunction and disjunction of literals, but also disjunction of conjunctions of atoms (as shown in table 6.2.3). This is not surprising as focal elements also take the form of conjunctions of literals, and their masses are products of marginals. The fact that in such cases the same results are obtained does not make the belief function analysis redundant: it shows that the results induced by the stochastic independence assumption are valid even when this assumption is relaxed, for some kinds of Boolean formulas. For more general kinds of Boolean formulas, the intervals computed by using belief functions are in contrast wider than when stochastic independence is assumed.

In general, the probability family induced by the stochastic independence assumption will be included in the probability family induced by the belief functions. This proposition can be proved using the results of Fetz (Fetz, 2001) and Couso and Moral (Couso and Moral, 2010). Any probability measure in $\mathcal{P}(m) = \{P \geq Bel\}$ dominating a belief function induced by a mass function m can be written in the form: $P = \sum_{E \subseteq \Omega} m(E) \cdot P_E$ where P_E is a probability measure such that $P_E(E) = 1$ that shares the mass $m(E)$ among elements of E. For a function of two Boolean variables x_1 and x_2, with two ill-known probability values $P_1(A_1) = p_1$ and $P_1(A_2) = p_2$, p_1 is of the form $l_1 + \alpha(u_1 - l_1)$ for some $\alpha \in [0, 1]$ and p_2 is of the form $l_2 + \beta(u_2 - l_2)$ for some $\beta \in [0, 1]$. The explicit sharing, among interpretations, of the masses $m(E)$, induced by probability intervals $[l_1, u_1]$ and $[l_2, u_2]$, that enables $P = P_1 P_2$ to be recovered is:

1. The masses on interpretations bear on singletons, hence do not need to be shared.
2. The masses on partial models are shared as follows
   - $m(A_1)\beta$ is assigned to A_1A_2, $m(A_1)(1-\beta)$ to $A_1\neg A_2$.
   - $m(A_2)\alpha$ is assigned to A_1A_2, $m(A_2)(1-\alpha)$ to $\neg A_1A_2$.
   - $m(\neg A_1)\beta$ is assigned to $\neg A_1A_2$, $m(\neg A_1)(1-\beta)$ to $\neg A_1\neg A_2$.
   - $m(\neg A_2)\alpha$ is assigned to $A_1\neg A_2$, $m(A_2)(1-\alpha)$ to $\neg A_1\neg A_2$.
3. $m(\Omega)$ is shared as follows: $\alpha\beta m(\Omega)$ to A_1A_2, $(1-\alpha)\beta m(\Omega)$ to $\neg A_1A_2$, $\alpha(1-\beta)m(\top)$ to $A_1\neg A_2$, $(1-\alpha)(1-\beta)m(\top)$ to A_1A_2.

It can be checked that the joint probability P_1P_2 is the form $\sum_{E \subseteq \Omega} m(E) \cdot P_E$ using this sharing of masses. This result can be extended to more than 2 variables. It indicates that the assumptions of source independence is weaker than the one of stochastic independence, and is of course stronger than no independence assumption at all. So the belief function approach offers a useful and tractable approach to evaluate the impact of stochastic independence assumptions on the knowledge of the probability of dreadful events in fault-tree analysis.

6.2.3 Example of the Primary/Backup Switch

Let us take again the example of the Primary/Backup Switch described in section5.5. The OCAS model and the associated fault tree are recalled:

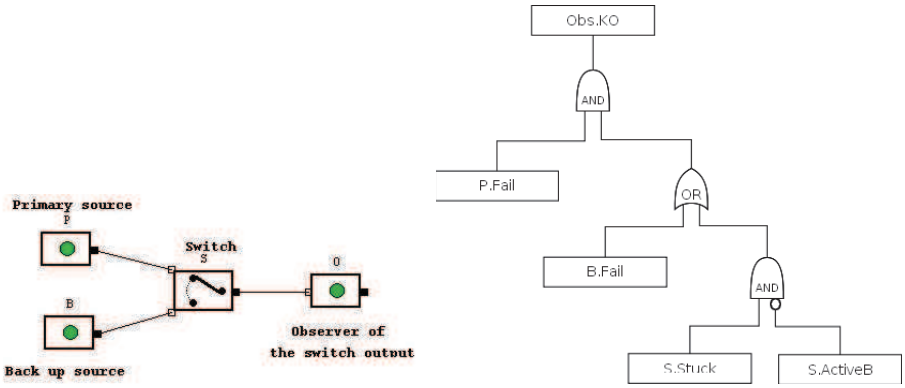


Figure 6.2: a) OCAS model of the Primary/Backup Switch b)Fault tree associated to event *Obs.KO*

The fault tree can be represented by the formula : $Obs.KO = P.Fail \wedge (B.Fail \vee (S.stuck \wedge \neg S.activeB))$, whose truth table is given on Fig. 6.3.

B	P	Sa	St	Obs.KO
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

Figure 6.3: Truth table of Obs.KO

Fig. 6.4 depicts the partial models that satisfy or are compatible with $Obs.KO$.

The same intervals values than in section 5.5 are taken for the interval probabilities associated to the events:

- $P.Fail$ has an imprecise probability of $P(P.Fail) = 10^{-4} + / - 50\%$
- $B.Fail$ has an imprecise probability of $P(B.Fail) = 10^{-4} + / - 10\%$
- $S.stuck$ has a precise probability of $P(S.stuck) = 10^{-5}$
- An activation order of the Switch is given, i.e. $P(S.activeB) = 1$

	Partial models	Bel	Pl		Partial models	Bel	Pl
1	B∧P	■	■	42	¬P∧St∧Sa		
2	B∧¬P			43	¬P∧¬St∧Sa		
3	B		■	44	P∧¬St∧Sa		■
4	¬B∧P		■	45	B∧P∧¬Sa	■	■
5	¬B∧¬P			46	B∧¬P∧¬Sa		
6	P		■	47	¬B∧P∧¬Sa		■
7	¬B		■	48	¬B∧¬P∧¬Sa		
8	¬P			49	B∧St∧¬Sa		■
9	Ω		■	50	B∧¬St∧¬Sa		■
10	B∧Sa		■	51	¬B∧St∧¬Sa		■
11	B∧¬Sa		■	52	¬B∧¬St∧¬Sa		
12	Sa		■	53	P∧St∧¬Sa	■	■
13	¬B∧Sa			54	¬P∧St∧¬Sa		
14	¬B∧¬Sa			55	¬P∧¬St∧¬Sa		
15	¬Sa		■	56	P∧¬St∧¬Sa		■
16	B∧St		■	57	B∧P∧St	■	■
17	B∧¬St		■	58	B∧¬P∧St		
18	St		■	59	¬B∧P∧St		■
19	¬B∧St		■	60	¬B∧¬P∧St		
20	¬B∧¬St			61	B∧P∧¬St	■	■
21	¬St		■	62	B∧¬P∧¬St		
22	P∧Sa		■	63	¬B∧P∧¬St		
23	P∧¬Sa		■	64	¬B∧¬P∧¬St		
24	¬P∧¬Sa			65	¬B∧¬P∧Sa∧¬St		
25	P∧St		■	66	¬P∧Sa		
26	¬P∧St			67	¬B∧¬P∧¬Sa∧¬St		
27	¬P∧¬St			68	¬B∧¬P∧¬Sa∧St		
28	Sa∧St		■	69	B∧P∧Sa∧St	■	■
29	Sa∧¬St		■	70	B∧P∧Sa∧¬St	■	■
30	¬Sa∧¬St		■	71	B∧P∧¬Sa∧St	■	■
31	¬Sa∧St		■	72	B∧P∧¬Sa∧¬St	■	■
32	P∧¬St		■	73	B∧¬P∧Sa∧St		
33	B∧P∧Sa	■	■	74	B∧¬P∧Sa∧¬St		
34	B∧¬P∧Sa			75	B∧¬P∧¬Sa∧¬St		
35	¬B∧P∧Sa			76	B∧¬P∧¬Sa∧St		
36	¬B∧¬P∧Sa			77	¬B∧P∧Sa∧St		
37	B∧St∧Sa		■	78	¬B∧P∧Sa∧¬St		
38	B∧¬St∧Sa		■	79	¬B∧P∧¬Sa∧St	■	■
39	¬B∧St∧Sa			80	¬B∧P∧¬Sa∧¬St		
40	¬B∧¬St∧Sa			81	¬B∧¬P∧Sa∧St		
41	P∧St∧Sa		■				

Figure 6.4: Partial models for the computation of Bel(Obs.KO) and Pl(Obs.KO)

Normally, the belief and the plausibility of the event $Obs.KO$ should be computed by making the exhaustive sum of the masses of the partial models described on Fig.6.4. In order to simplify the equations, we will denote $P.Fail$ by P, $B.Fail$ by B,$S.stuck$ by St and $S.activeB$ by Sa.

But the fact that the Boolean expression is constituted of four Boolean variables appearing only once each, shows it satisfies the proposition 6.7. The inclusion/exclusion principle can then be applied, by writing the formula as a DNF.

$$\begin{aligned} Obs.KO = (\neg B \wedge P \wedge \neg Sa \wedge St) \vee (B \wedge P \wedge \neg Sa \wedge \neg St) \\ \vee (B \wedge P \wedge \neg Sa \wedge St) \vee (B \wedge P \wedge Sa \wedge \neg St) \vee (B \wedge P \wedge Sa \wedge St) \end{aligned}$$

The belief will be obtain by computing:

$$\begin{aligned} Bel(Obs.KO) = Bel(\neg B \wedge P \wedge \neg Sa \wedge St) + Bel(B \wedge P \wedge \neg Sa \wedge \neg St) \\ + Bel(B \wedge P \wedge \neg Sa \wedge St) + Bel(B \wedge P \wedge Sa \wedge \neg St) + Bel(B \wedge P \wedge Sa \wedge St) \\ \text{(other terms are 0)} \end{aligned}$$

And the plausibility:

$$\begin{aligned} Pl(Obs.KO) = Pl(\neg B \wedge P \wedge \neg Sa \wedge St) + Pl(B \wedge P \wedge \neg Sa \wedge \neg St) \\ + Pl(B \wedge P \wedge \neg Sa \wedge St) + Pl(B \wedge P \wedge Sa \wedge \neg St) + Pl(B \wedge P \wedge Sa \wedge St) \\ \text{(other terms are 0)} \end{aligned}$$

With the above values, we obtain the same result than in section 5.5, i.e. $[4.5 \times 10^{-9}, 1.65 \times 10^{-8}]$.

The use of belief functions in the general case of Boolean formulas that does not satisfy the proposition 6.7 remains difficult because of the computation time required. But this chapter gives a straightforward way to handle some particular cases of Boolean formulas. The demonstration of the equivalence with the interval arithmetic in the case of monotonic fault trees is a result that can be interesting for the Common Cause Analysis introduced in 2.1. It shows that it is sufficient to prove the independence of information sources instead of the stochastic independence, which can be easier to verify in practice.

Chapter 7

Probability laws with imprecise parameters

Contents

In this chapter, we will see the origin of the intervals used in interval analysis as discussed in the chapters 5 and 6. The probabilities that are attached to the variables comes from statistical studies on the different components of the system. Some uncertainty comes from these statistical studies, and can be handled by different means. This chapter will focus on the use of P-boxes to cope with the imprecision on certain parameters of probability laws. This study has been subject of one article (Jacob et al., 2012b).

7.1 Probability intervals generation

In the previous chapters, we assumed that the probability attached to every Boolean variable was contained in an interval. In order to understand the source of this imprecision, we will first explain the statistical analysis from where these values are extracted.

7.1.1 Using linear regression

The probabilities of the Boolean variables in a fault tree generally represent the failure of a component or equipment of the system from which the fault tree was generated. These probabilities are provided by the equipment suppliers or manufacturers. Generally, they calculate such probabilities based on statistical studies of the component or the system. Such studies involve test of components and collection of their results. In the end, techniques such as *regression analysis* are used to determine the probability laws that may fit the failure probability of the components.

In statistics, regression analysis is a statistical technique used for estimating some relationships between some variables, when the focus is on the relationship between a dependent variable and one or more independent variables. It includes several techniques for modeling and analyzing several variables. It is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. The analysis begins with the construction of a regression model. It may be important to confirm the goodness of fit of the model and the statistical significance of the estimated parameters. The estimation target is a function of the independent variables called *regression function*. When the sample tend to be representing a straight line, i.e. the relationship between two variables is under the form $Y = a + bX$, where X is an explanatory variable and Y is a dependent variable; the regression analysis is called *linear regression*. Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.

In reliability studies, the components failure is generally following exponential or Weibull probability laws, as explained in appendix A. Those laws are represented by a straight line when depicted on a logarithmic scale. Fig. 7.1 represents a data sample of some measures following a Weibull distribution.

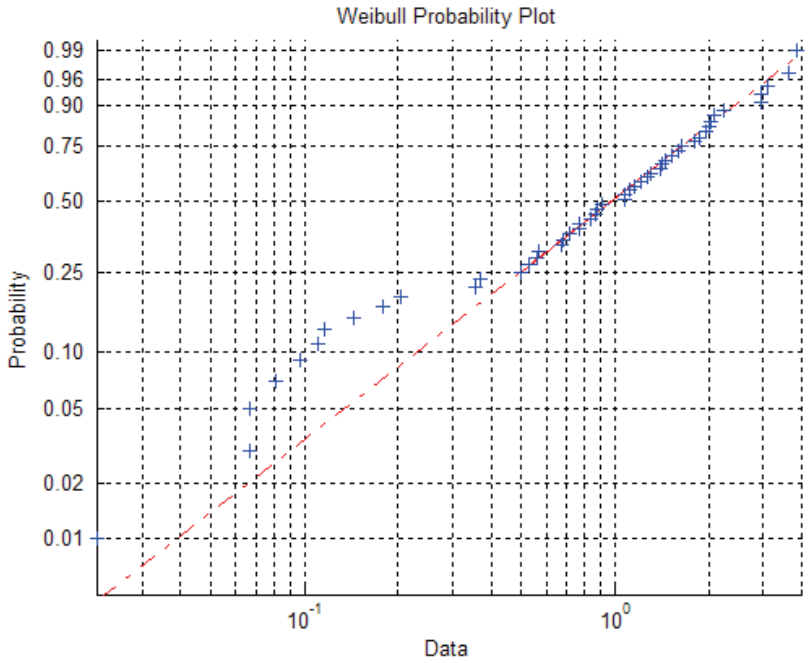


Figure 7.1: Weibull sample on a logarithmic scale

The statistical studies consist in finding the equation of the straight line that fits best the sample, in order to get the probability distribution modeling the data. Hence, the uncertainty on the parameters computation seems obvious, as it is not possible to fit to all values of a real sample, the model will try to fit as much as possible. Some values will be left over, it is not possible to cope with all the

sample in a straight line. That is the main reason why the imprecision of probability distributions used for describing the probability of failure of components and equipments is not taken into account in the model when considering a precise parameter. It is only possible to find a *confidence interval*, whose width depends on the number of samples. The following sections give some indications about the way to capture more faithful information in those probability distributions.

7.1.2 Definition of a P-box

A cumulative distribution allows to define a probability measure on R. A natural way to give an approximation of a probability that is not well known is to consider a pair $[\underline{F_T}, \overline{F_T}]$ of cumulative distributions in order to represent this uncertainty. Those cumulative distributions generalize the notion of interval, and are called *p-boxes*(Destercke et al., 2008) (Utkin and Destercke, 2007).

A p-box represent the class of probability measures whose distribution function is bounded by $\underline{F}$ and $\overline{F}$:

$$\underline{F_T}(x) \leq F_T(x) \leq \overline{F_T}(x) \forall x \in R \tag{7.1}$$

Conversely, a p-box can be induced by a probability family $\mathcal{P}$ by:

$$\forall x \in R, \underline{F_T}(x) = \underline{P}([0, x]), \overline{F_T}(x) = \overline{P}([0, x]) \tag{7.2}$$

P-boxes may be used in many different kinds of situations involving incomplete information about a quantity. There are different ways to obtain p-boxes from data and analytical analysis.

Sometimes a probability distribution is known to have a particular shape while its parameters are not known precisely but only as intervals. Such distribution is known as a *distributional p-box* or *parametric p-box* (Montgomery, 2009). Distributional p-box or parametric p-box is easy to obtain by enveloping extreme distributions given the possible parameters.

If we have precise knowledge about the parameters such as mean and variance of a distribution but we do not know the type of the distribution family, the distribution cannot be specified precisely due to missing family type. In such cases, envelopes of all distributions matching given moments can be constructed from inequalities that enclose all distribution functions having specified parameters. These are basically *distribution-free p-boxes* because they make no assumption about the family or shape of the uncertain distribution.

An *empirical distribution* is often used when all members of a population can be measured, or when random sample data are abundant. This empirical distribution may be generalized to a p-box when those data have non negligible measurement uncertainty represented by interval ranges about each sample value. Such a p-box can be specified by cumulating the lower endpoints of all the interval measurements into a cumulative distribution forming the left edge of the p-box, and cumulating the upper endpoints to form the right edge. The broader the measurement uncertainty, the wider the resulting p-box.

Uncertainty about the shape of a probability distribution may arise when the sample size of the empirical data characterizing it is small. In traditional statistics several methods have been proposed to account for this sampling uncertainty about the distribution shape, which are distribution-free in the sense that they make no assumption about the shape of the underlying distribution. While at the same time, there are related confidence-band methods that include assumptions about the shape or family of the underlying distribution, which can often result in tighter confidence bands.

A p-box can be constructed as the envelope of the various cumulative distributions when there are multiple possible probability distributions that might describe a variable, and an analyst cannot select any of them based on available information. With a sensitivity study, it is also possible to account for the uncertainty about the correct distribution but such studies become more complex as the number of possible distributions grows; and combinatorially more complex as the number of variables, for which there could be multiple distributions, increases.

7.2 Probability laws with imprecise parameters

This section will focus on the cases where the parameters of the widely used reliability distributions are imprecise. It is the case of *distributional p-box*, as the probability distributions are known, but the not the parameters.

7.2.1 Exponential distribution

Recall that the reliability analysis of an aircraft takes into account, among others, the electronic components. Their probabilities of failure are modeled with constant failure rates $\lambda(t) = \lambda$, because they do not have any burn-in nor any wear-out periods, respectively at their beginning and their end of life. Moreover, when the failure rate is constant, equation (A.7) becomes $R(t) = e^{-\lambda t}$, i.e., an exponential distribution.

The probability density function is given by:

$$f_T(t) = \lambda e^{-\lambda t} \tag{7.3}$$

And is represented in Fig. 7.2.a). Its cumulative distribution, depicted in Fig. 7.2.b) is given by:

$$F_T(t) = 1 - e^{-\lambda t} \tag{7.4}$$

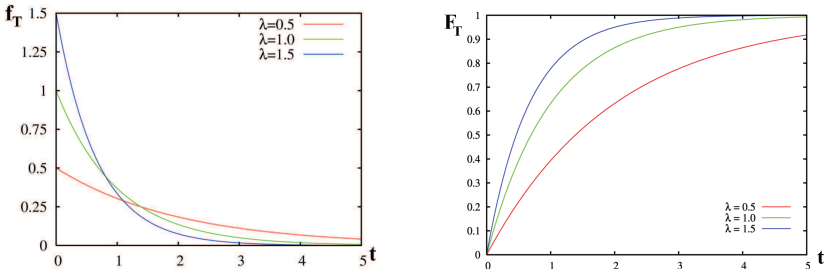


Figure 7.2: a) Exponential density function b) Cumulative distribution

7.2.2 Exponential law with imprecise failure rate

If the only information available about the failure rate λ is an interval containing it, then there are different probability distributions representing the failure of the component, as will be presented in the sequel.

The goal is to find the range of the cumulative distribution, when the failure rate is imprecise: $\lambda \in [\underline{\lambda}, \overline{\lambda}]$. In interval analysis, knowing the monotonicity of a function makes the determination of its range straightforward.

The function $1 - e^{-\lambda t}$ is strictly increasing with λ, hence the range of the cumulative distribution, when λ is varying, for every $t > 0$ and $\lambda > 0$, is given by the expression:

$$F_T(t) = \{1 - e^{-\lambda t}, s.t.\ \lambda \in [\underline{\lambda}, \overline{\lambda}]\} = [1 - e^{-\underline{\lambda} t}, 1 - e^{-\overline{\lambda} t}] \tag{7.5}$$

The range of the cumulative distribution with respect to some values of λ and in time interval $t = [0, 10]$ is represented in Fig. 7.3.a).

For the probability density function, it is a little bit more complex. The derivative with respect to λ of the function $f_T(t)$ is:

$$\frac{\partial}{\partial \lambda} f_T(t) = (1 - \lambda t) e^{-\lambda t} \tag{7.6}$$

This means that the function will be increasing with respect to λ when $\lambda t < 1$, and decreasing otherwise. The range of the function will depend on λ and t, as illustrated on Fig. 7.3.b).

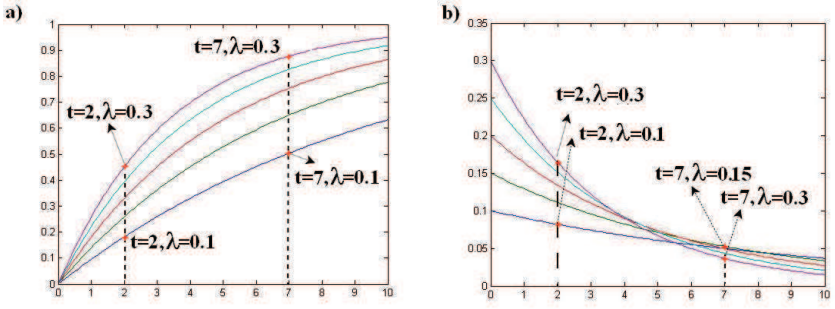


Figure 7.3: a) Range of the cumulative distribution b) Range of the probability distribution ($0.1 < \lambda < 0.3$, $0 < t < 10$)

In the following, we give different interpretations of the probability of failure of a component, as used in fault tree analysis.

7.2.2.1 Occurrence of an atomic failure before time t

In the quantitative analysis of a safety model, each component (or type of component) of this model will have its own failure rate, and its own failure probability. The main goal of this analysis is to ensure that, at each time t, the probability that the system has failed remains below a certain value. We are interested in the probability of failure of a component or system before time t, hence the cumulative distribution will be used for our computations.

When the parameter λ is imprecise, and its possible values are known to lie within the interval $[\underline{\lambda}, \overline{\lambda}]$, the probability distribution will be contained in the *p-box* (Ferson et al., 2003):

$$\{P, P(T < t) \in [1 - e^{-\underline{\lambda} t}, 1 - e^{-\overline{\lambda} t}]\}, \tag{7.7}$$

where a p-box is an ordered pair of cumulative distributions, representing the probability family. This family (7.7) contains more probability distributions than those with an exponential distribution. However it is enough to use the p-box when computing probability bounds of events of the form $T < t$. But it will not be the case for computing other indicators, e.g. the variance.

7.2.2.2 Occurrence of an atomic failure between t_1 and t_2

In some cases, it can also be useful to compute the probability that the event will occur between two dates t_1 and t_2. This can be expressed as the conditional probability $t_1 < T < t_2$ given that T does not occur before t_1:

$$P(T < t_2 | T \geq t_1) = \frac{e^{-\lambda t_1} - e^{-\lambda t_2}}{1 - e^{-\lambda t_1}} \tag{7.8}$$

When $\lambda \in [\underline{\lambda}, \overline{\lambda}]$, the partial derivative of $P(T < t_2 | T \geq t_1)$ with respect to λ must be computed in order to find the p-box of the probability distribution.

$$\frac{\partial}{\partial \lambda} P(T < t_2 | T \geq t_1) = \frac{t_2 e^{-\lambda t_2} - t_1 e^{-\lambda t_1}}{(1 - e^{-\lambda t_1})^2} \tag{7.9}$$

By noticing that the function $xe^{-\lambda x}$ is decreasing with λ when x is fixed, we can deduce that $\frac{\partial}{\partial \lambda} P(T < t_2 | T \geq t_1)$ is strictly negative. Hence, $P(T < t_2 | T \geq t_1)$ is decreasing with respect to λ, and the p-box containing the probability that the event occurs between t_1 and t_2 is:

$$P(T < t_2 | T \geq t_1) \in \left[\frac{e^{-\overline{\lambda} t_1} - e^{-\overline{\lambda} t_2}}{1 - e^{-\overline{\lambda} t_1}}, \frac{e^{-\underline{\lambda} t_1} - e^{-\underline{\lambda} t_2}}{1 - e^{-\underline{\lambda} t_1}} \right]. \tag{7.10}$$

7.2.3 Exponential distribution with periodic maintenance

It is also possible to represent schedules of preventive maintenance by means of probability distributions. Indeed, some components are preventively replaced or repaired with a period of length θ: this maintenance task will reset the probability of failure to 0 after θ flight hours (FH). The cumulative distribution representing this probability of failure in this case is a periodic function that can be written as:

$$\text{for } k \in \mathbb{N}, P(T < t) = 1 - e^{-\lambda(t - k\theta)}, \text{ if } t \in [k\theta, (k+1)\theta] \tag{7.11}$$

If the failure rate λ is imprecise, then the probability of failure is the same as in the section 7.2.2.1 on the interval $[0, \theta]$:

$$\text{for } k \in \mathbb{N}, P(T < t) \in [1 - e^{-\overline{\lambda}(t - k\theta)}, 1 - e^{-\underline{\lambda}(t - k\theta)}], \text{ if } t \in [k\theta, (k+1)\theta] \tag{7.12}$$

If both the failure rate and the period are imprecise, then it is still possible to compute the range of the resulting cumulative distribution: we can consider that the period θ can be any value in the interval of time $[\theta_1, \theta_2]$. In this case, the size of the interval probability will grow very quickly with the size of the interval $[\theta_1, \theta_2]$. The minimum and maximum cumulative distributions, denoted by $\overline{P}(F < t)$ and $\overline{P}(F < t)$, are given for $k \in \mathbb{N}$ by the following expressions:

$$\underline{P}(T < t) = \begin{cases} 0 \text{ for } k\theta_1 < t < k\theta_2 \\ 1 - e^{-\underline{\lambda}(t - k\theta_1)}, \text{ for } t \in [k\theta_2, (k+1)\theta_1] \end{cases}$$

$$\overline{P}(T < t) = 1 - e^{-\overline{\lambda}(t - k\theta_1)}, \text{ for } t \in [k\theta_2, (k+1)\theta_2]$$

An example of those p-boxes for $\lambda \in [0.5, 0.6]$ and $T \in [\theta_1, \theta_2]$ is shown on Fig. 7.4.

7.2.4 Weibull distribution

In the case of a hardware component, it can be useful to model its *burn-in* period (i.e. the fact that the failure rate is high at the beginning but will decrease after some time) and its *wear-out* phase

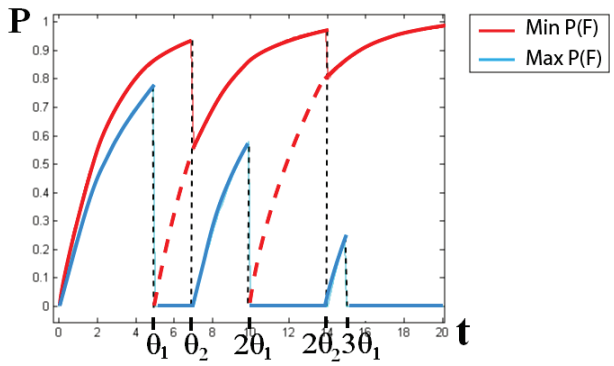


Figure 7.4: An example of periodic maintenance with $\theta \in [\theta_1, \theta_2]$ and $\lambda \in [0.5, 0.6]$

(i.e. the fact that after some time, the failure rate of the component increases). Therefore, the failure rate has the shape of a *bathtub curve*, as shown on Fig.7.5.

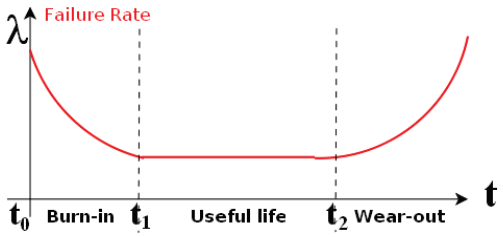


Figure 7.5: Bathtub Curve

In order to model the reliability in this case, the *Weibull* law is used. It is a two parameters law, described by the formula:

$$R(t) = e^{-(\frac{t}{\eta})^{\beta}} \tag{7.13}$$

where η is the *scale parameter* and β the *shape parameter*.

The probability density function of a Weibull law is given by the expression:

$$f_T(t) = \frac{\beta}{\eta}(\frac{t}{\eta})^{\beta-1} e^{-(\frac{t}{\eta})^{\beta}} \tag{7.14}$$

And its cumulative distribution is:

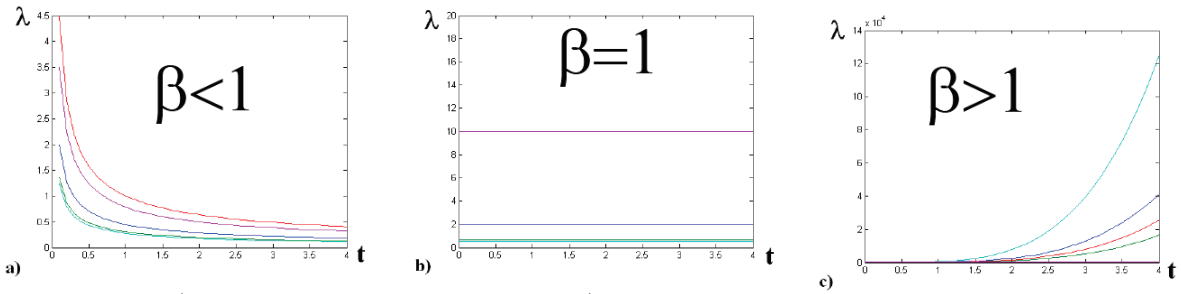
$$F_T(t) = 1 - e^{-(\frac{t}{\eta})^{\beta}} \tag{7.15}$$

From equation (A.5), the expression of the failure rate as a function of t is:

$$\lambda(t) = \beta.\frac{1}{\eta^{\beta}}.t^{\beta-1} \tag{7.16}$$

In order to get a bathtub curve, we will chose:

- a value $\beta_1 < 1$ for the burn-in phase (t_0 to t_1),
- $\beta = 1$ for the useful life (t_1 to t_2),
- a value $\beta_2 > 1$ for the wear-out phase ($t > t_2$).



. a) $\beta_1 < 1$. b) $\beta = 1$. c) $\beta_2 > 1$

Figure 7.6: Variation of the Weibull distribution with η

In the wear-out phase, the reference origin of the failure rate and the cumulative function is not 0, hence in order to be able to shift the distribution to starting time t_2, a *location parameter* γ should be added:

$$F_T(t) = 1 - e^{-(\frac{t-\gamma}{\eta})^{\beta}} \tag{7.17}$$

Despite the fact that the parameter β is different for each phase of the bathtub curve, the failure rate is a continuous function. Therefore, there will be a constraint for each change of phase, that will ensure the continuity. When the scale parameter η remains the same for all the phases, this constraint is expressed as below:

$$\left\{ \begin{array}{l} \beta_1.\frac{1}{\eta^{\beta_1}}.t_1^{\beta_1-1} = \frac{1}{\eta} \\ \beta_2.\frac{1}{\eta^{\beta_2}}.(t_2-\gamma)^{\beta_2-1} = \frac{1}{\eta} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} \beta_1.(\frac{t_1}{\eta})^{\beta_1-1} = 1 \\ \beta_2.(\frac{(t_2-\gamma)}{\eta})^{\beta_2-1} = 1 \end{array} \right. \tag{7.18}$$

Like the failure rate curve, the global cumulative distribution will be composed of three pieces of cumulative distributions with different parameters. To ensure the continuity of the global one, the cumulative distribution of each new phase should start from the last value of the previous phase.

When the parameters of a Weibull law are imprecise, they should still verify the constraints of β for each phase, and the ones expressing the continuity of $\lambda(t)$ (equation 7.18). Fig. 7.6 shows the variation of the failure rate with the variation of η for the three different phases of the bathtub curve.

The imprecision pervading the parameters β and η of the Weibull law affects the value of the time points where the phases change in the bathtub curve (t_1 and t_2), due to equation (7.18). These time points become themselves intervals.

In order to find the range of the cumulative distribution with the different parameters, the monotonicity study of the function will also be required, as in section 7.2.2. In this case, we have a two parameter function, hence we compute its gradient.

$$\overrightarrow{\nabla} P(T<t) = \left| \begin{array}{l} \frac{\partial}{\partial\eta}P(T<t) \\ \frac{\partial}{\partial\beta}P(T<t) \end{array} \right. = \left| \begin{array}{l} \beta.\frac{t^{\beta}}{\eta^{\beta+1}}e^{-(\frac{t}{\eta})^{\beta}} \\ ln(\frac{t}{\eta}).(\frac{t}{\eta})^{\beta}e^{-(\frac{t}{\eta})^{\beta}} \end{array} \right. \tag{7.19}$$

By noticing that t, η, β and $e^{-(\frac{t}{\eta})^{\beta}}$ are always positive, we can conclude that the partial derivative $\frac{\partial}{\partial\eta}P(T<t)$ is positive. But the partial derivative $\frac{\partial}{\partial\beta}P(T<t)$ is positive when $\eta > t$ and negative otherwise, because of the term $ln(\frac{t}{\eta})$. Equation (7.18) implies that for $\eta > t_1$, $P(T<t)$ is decreasing

with respect to β for $t < t_1$. This means that the p-box of a Weibull distribution will be:

$$[1 - e^{-(\frac{t}{\overline{\eta}})^{\overline{\beta}}}, 1 - e^{-(\frac{t}{\underline{\eta}})^{\underline{\beta}}}], \text{ for } t \in [0, t_1] \tag{7.20}$$

Between t_1 and t_2, β is fixed to 1, hence the bounds for the cumulative distribution are:

$$[1 - e^{-\frac{t-t_1}{\overline{\eta}}} + P(T < t_1), 1 - e^{-\frac{t-t_1}{\underline{\eta}}} + P(T < t_1)], \text{ for } t \in [t_1, t_2] \tag{7.21}$$

When $t > t_2$, the quantity $(t_2 - \gamma)$ is computed through equation (7.18). Now the partial derivatives are similar to the ones in equation (7.19), replacing t by $t - \gamma$. Hence the condition for $\frac{\partial}{\partial\beta}P(T < t)$ being positive is that $t < \gamma + \eta$, so we get the range of the cumulative distribution:

$$[1 - e^{-(\frac{t-\gamma}{\overline{\eta}})^{\underline{\beta}}} + P(T < t_2), 1 - e^{-(\frac{t-t_2}{\overline{\eta}})^{\overline{\beta}}} + P(T < t_2)], \text{ for } t < \gamma + \eta \tag{7.22}$$

$$[1 - e^{-(\frac{t-\gamma}{\underline{\eta}})^{\overline{\beta}}} + P(T < t_2), 1 - e^{-(\frac{t-t_2}{\overline{\eta}})^{\underline{\beta}}} + P(T < t_2)], \text{ for } t > \gamma + \eta \tag{7.23}$$

Once we know the impact of imprecise parameters on one probability distribution, we can use it in the computation of the probability distribution of an undesired event described by a fault tree.

7.2.5 Extension to probability laws with imprecise parameters

In the case of fault tree analysis, the probability of a undesired event is described with a Boolean formula F, function of N Boolean variables $V_i, i = 1 \ldots N$ representing the failure (or states) of its components. When the probability of V_i is represented by a probability distribution with an imprecise parameter, we have a p-box for the cumulative probability that undesired event occurs before a given date. The variables V_i are supposed to be stochastically independent, and they can follow different probability distributions. Also, their parameters can be of different types: some can be precise, when the information is available and well known, some can be imprecise.

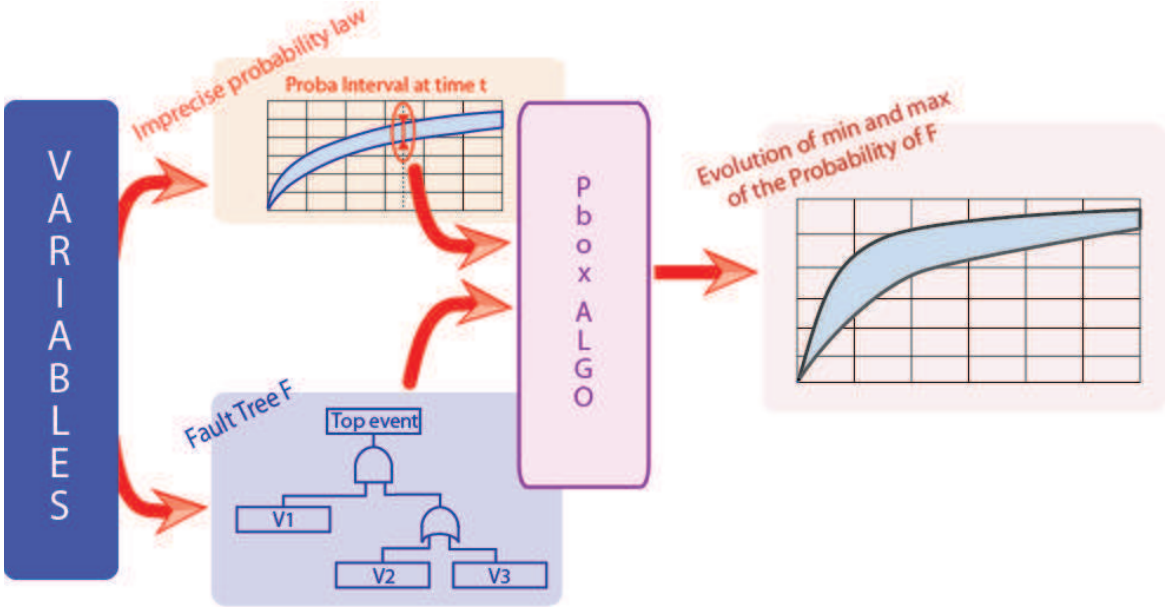


Figure 7.7: P-box algorithm

The goal will be to find the p-box describing the undesired event probability across time from the p-boxes of the variables V_i. The best way to carry out this computation is discretization of the time, and to find for each t and for each V_i, the associated interval $I(t, V_i)$. Of course, if all input probability distributions are precise, the probability of the undesired event will be precise. When T_i is a random variable representing the failure time of the component V_i, we have that:

$$I_i(t, V_i) = [\underline{P(T_i < t)}, \overline{P(T_i < t)}]$$

Let us consider two variables V_1 and V_2 with exponential laws, and respective imprecise parameters $\lambda_1 \in [0.1, 0.14]$ and $\lambda_2 \in [0.1, 0.2]$. The two first graphs of Fig. 7.8 depicts the intervals $I_i(t = 6, V_i)$ associated to these variables.

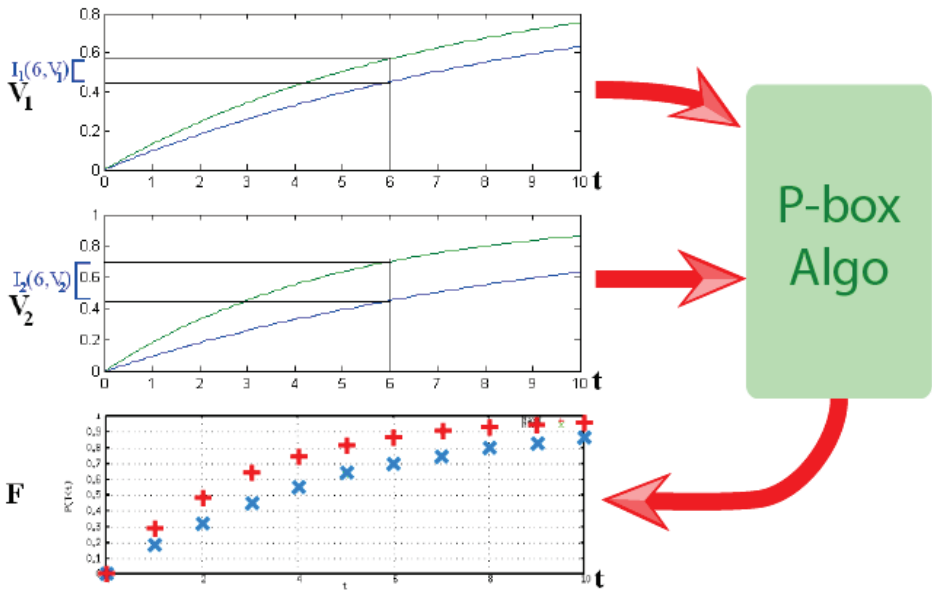


Figure 7.8: Example of aggregation of two exponential p-boxes with $F = V_1 \vee V_2$

For the same time point, the range of the probability of variable V_i is given by the interval $I_i(t, V_i)$. So, *for this time* t, the algorithm presented in 5.4 can be used to compute the probability of the undesired event.

In order to compute the range of the cumulative distribution of the undesired event for *all time instants*, we apply the algorithm for all k time instants, $k = \frac{T_o}{T_s}$, where T_o is the observation interval and T_s is the time step. An example of the result given by the algorithm for an undesired event described by the Boolean formula $F = V_1 \vee V_2$, and for a time step of 1, is depicted on the last graph of Fig. 7.8.

At this point, we can use the approach described in section 5.4, where the available probability information are intervals (I_i) associated to each Boolean variable V_i. Hence, the same algorithm can be applied. Following this method, the algorithm has been extended to take into account several probability distributions.

7.3 Example of the Primary/Backup switch

The sensitivity analysis algorithm is applied to the fault tree, with the following imprecise parameters for the distributions:

- *P.Fail* possesses an exponential distribution with an imprecise failure rate $\lambda = 10^{-4} + / - 50\%$ and a precise periodic maintenance of period $\theta = 30$ FH
- *B.Fail* possesses an exponential distribution with an imprecise failure rate $\lambda = 10^{-4} + / - 10\%$ and a precise periodic maintenance of period $\theta = 35$ FH
- *S.stuck* possesses an exponential distribution with a precise $\lambda = 10^{-5}$
- An activation order of the Switch occurs after $t = 80$ FH

On Fig. 7.9, we can observe the minimum and maximum cumulative distributions of the event *Obs.KO*, for a duration of 100 flight hours (around three or four months for a commercial aircraft). The picture lays bare the effect of periodic maintenance on those distributions.

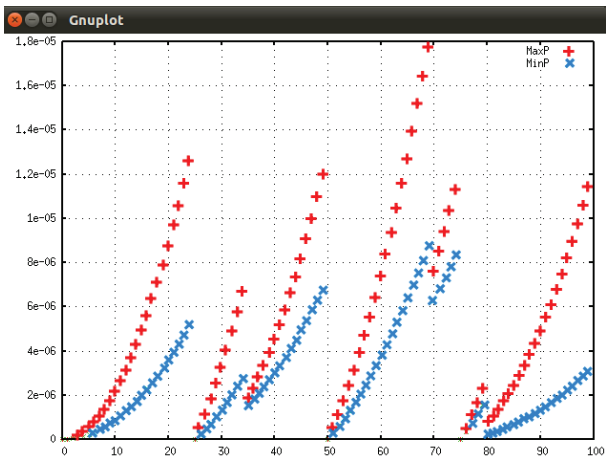


Figure 7.9: Evolution of the p-box of the event *Obs.KO* for a duration of 100 FHs

The study of this p-box can give crucial information about the probability of undesired events, such as *Obs.KO*: when the area between the minimum curve and the maximum curve is tight, computations are reliable. The larger it is, the more uncertainty we will get. But even under uncertainty, it can still be possible to ensure safety, if the upper probability of the undesired event is below a legal threshold. For instance, in our case study, the maximum probability is always less than 1.8×10^{-5} for 100 flight hours, with this maintenance schedule.

In safety analysis, the requirements to meet for each failure condition are described in the PSSA, mentioned in section 2.1. They are classified with respect to their probability of occurrence, their severity and some other features. This classification defines the legal probability threshold to be met. In the example, the event *Obs.KO* meets a requirement of an event occurring less than 10^{-4} over the 100 first flight hours, but not the threshold of 10^{-5}. In case a threshold of 10^{-5} is required by the PSSA for this event, then we must change the maintenance schedule in order to meet this requirement. The algorithm allows to test easily several scenarios of maintenance, in order to find one that ensures the threshold of 10^{-5}. With a periodic maintenance of the primary supplier every 19 flight hours instead of 25, and of the backup supplier every 22 flight hours instead of 35, this requirement can be met despite the uncertainty about the inputs, as shown on Fig. 7.10.

Being able to model the impact of incomplete information on probabilistic safety analysis is very useful for maintenance management. It allows the user to select the best representation for available data, in order to get a faithful advice. Precise data can be used when they are available, but they do not need to be assumed so when they are not. Consequently, more facets of uncertainty can be

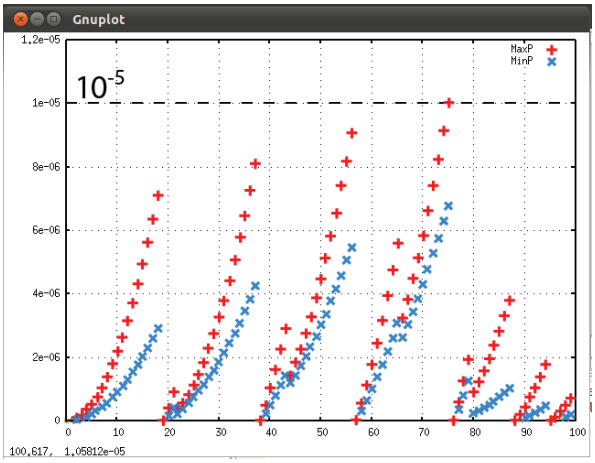


Figure 7.10: A different scenario of maintenance schedule

taken into account, and especially the difference between the variability of failure times and the lack of knowledge on distribution parameters.This difference can be very crucial in a decision process, where confidence about the computations plays a decisive role.

The computation time of this algorithm is exponential with respect to logical variables that appears both in positive and negative forms in the fault-tree (in practice there are very few of them), but the complexity of the Boolean functions makes the monotonicity difficult to determine. Hence, some efforts must be done in the implementation of the algorithm in order to be able to reduce computation time.

Chapter 8

Implementation

Contents

An aircraft must fulfill some operational requirements in order to achieve its mission. For example, it must satisfy certain predefined conditions prior to taking off, called dispatch requirements. Dispatch requirements must be met before aircraft is allowed to fly, otherwise, the mission has to be adapted to the current state of the aircraft, or canceled. These conditions concern maintenance facilities at the current airport and at the next destinations, the current functional state of the aircraft and the intended mission. Therefore, missions and maintenance activities must be adjusted regularly, based up on the current status of the above mentioned parameters. This leads us to the core issue of developing evaluation methods which assess operational reliability and can suggest a suitable mission under given circumstances.

DIANA is a tool that will be used in helping the maintenance planning and to assist the pilot in decision making. The crew and maintenance team will be able to use it for maintenance support. Indeed, DIANA should be able to provide the list of priorities for the equipments to maintain, in terms of potential failures, future missions, cost and reliability constraints. It will also take into account the list of equipments that do not work properly and have some influence on the efficiency or the energy consumption. Also each airport has different maintenance facilities, therefore DIANA must take into account the availability of maintenance facilities at airports and those at the aircraft

operator's base. Moreover, DIANA must also consider unexpected events, such as failure of some equipments or changes in maintenance facilities at the next destination of the aircraft.

In order to compute the priorities mentioned above, DIANA uses safety models extended to support operationability requirements. Safety models are architectural and/or functional representations of the equipments of aircraft to capture the health status of aircraft. These models are further extended to incorporate environment variables such as a change in aircraft mission.

At the moment, Airbus has a good expertise in safety models. The goal in the @MOST project is to design an Operational model of the aircraft, based on the already existing safety models. In our point of view, and from the discussion in the @MOST meetings, the main difference between the existing Safety model and the *foreseen* Operational model is that in the safety model only the dreaded events are considered, whereas in the operational model it is necessary to also model the events whose occurrence *impacts* on maintenance even if they do not impact on safety.

So, the extension of the safety models will take into account more issues, because some failure can have minimal impact on the safety, but have impact on the operationability, or decrease the performances of the aircraft. Also, the model will now take into account the different phases of the flight, and the missions of the aircraft, in order to improve the pertinence indicators for the Decision Support. As in the safety analysis, the goal of the operational analysis is to build a model of the system, with their sub-systems and components in a hierarchical way, and to study the failure propagation. From this new operational model we can also generate *fault trees* for some prescribed events to be avoided, as they can have a negative impact on the safety or the operationability.

DIANA will also help from the pilot support point of view, by providing some input to help for the dispatch. It will provide the list of items that could be operationally relevant to help the pilot in his decision, and some indications and advice for the most economic use of the aircraft. In order to be able to make the best possible decisions at the end of the process, we must deal with uncertainty at each step. There are many different kinds of information used in the process and each information item has some kind of imprecision/uncertainty in it. The goal will be to represent information as faithfully as possible, then to study the propagation of this imprecise information in the system and evaluate its impact.

We first studied the impact of uncertainty pervading the available input values on already existing models. The goal was to exploit the results this study at the decision making step. Hence, the indicators used will be different from the ones used for certification purpose. In a real world scenario, the knowledge of input values to these models is not always accurate and complete. The goal of such study is to analyze the impact of the imprecision on input values on the outputs of the models. That is the reason why efficient computation algorithms are needed, in order to deal with the complexity of the models.

After presenting all the theoretical background in Part II, developed algorithms in Chapters 5 and 7, and deducing relevant conclusions, it is also important to implement them in a software to use in a real life application. In this chapter we will describe the implementation of previously established concepts in a computer software and apply it on a case study.

8.1 Implementation of the algorithm

In this section, we will present implementation of the algorithm corresponding to the method of imprecise probability computations for BDDs described in the section 5.3. This algorithm is able to

compute the exact bounds of the probability of a Boolean function, given the interval ranges of its atomic probabilities. It has been coded in C++ language, and is based on BDD packages named CUDD and BuDDy along with other open source libraries, details of which can be seen in Appendix D. We call this implementation as BDD based interval analysis.

8.1.1 Monotonicity and computational complexity

In Fault-Tree Analysis theory, modeling conventions are generally such that all variables appear only positively in the Boolean formula of the top event: no variable appears with a negation, so the probabilistic formula is monotonic and increasing. But in practice, there are several cases where some variables can appear negatively, and sometimes even both negatively and positively, so that the top formula can be non-monotonic. This can be due to:

- Fact that some negations are introduced due to compilation: this is clear in BDDs and also in fault-trees obtained from so-called Mode Automata (Rauzy, 2002). In this case, the expression is still monotonic as long as the Boolean formula could also be expressed without negative literals (e.g. the connective OR).
- State modeling: in some systems, it is necessary to use variables that model some special states, or modes, which no longer represent a failure, and the global formula may depend on such variables and their negation. It is not necessary increasing with respect to these variables.
- Exclusive Failures: sometimes, failures cannot physically occur simultaneously, they are then represented by mutually exclusive events or failure modes. Mutual exclusion implies non-monotonicity.

In practice, those kind of variables are very few compared to ”usual” failures; hence, the algorithm will only have NP-hard complexity for them, and be linear for all other variables. Then there are some variables for which we don’t know the monotonicity of top formula. For these variables of unknown monotonicity, we must compute all possible cases to get exact bounds of the probability of the top formula, given the interval ranges of its atomic probabilities. Therefore, it is very important to reduce the number of variables for which top formula’s monotonicity is unknown to reduce computation times. The computational complexity is exponential with respect to variables of unknown monotonicity. Different techniques were explored to reduce the number of such variables. In sections below we will discuss comparison of these techniques.

8.1.2 Flow chart of the implemented software

Before going into detail of different techniques to determine monotonicity of top formula with respect to its variables, it is relevant to briefly discuss top level implementation of the software code. This will help in understanding different techniques to determine monotonicity, described afterwards.

In order to compute exact bounds of the probability of the top formula, given the interval ranges of its atomic probabilities, we have two approaches implemented in the software. We can do this either by using Minimal cut sets or by using fault trees. Flow chart of the implementation is shown in Appendix D. In order to perform computations on the given input files, we parse the input file. For example, in the flow chart we can see that we read input file in aralia format and parse it to convert it into boolean formula. Then, we parse Boolean formula to construct a BDD representing this boolean formula and so on. All these steps use different algorithms, which are described below.

This section presents some of the algorithms used in the implementation of the software. The algorithm 1, is the top level algorithm of the software and describes the computation process from input to output of results. Some processes used in this algorithm such as $SplitVariables$, $ComparisonOfCat$ and $CombinationI$ are further explained in algorithms 2, 3 and 4 respectively.

Algorithm 1 Interval Analysis for BDD

Parse input file with formula F
Create a list V of variables of F with attributs lb, ub (lower/upper bound), $type$, $path.value$
N_V= length(V)
Step1:
Pa = BuDDy(CUDD(formula F)) {create a BDD from F using the format of eq. E.1}
$\mathcal{P}$ = set of paths of Pa
Step2: Split the variables in 3 types
Create 2 tables $Type(F)$ and $Type(BDD\_F)$ of size N_V
SplitVariables(F) {Split the variables in 3 types from the Formula F}
SplitVariables(BDD_F) {Split the variables in 3 types from the BDD of F}
ComparisonOfCat(F, BDD_F) {To keep as few variables from type 2 as possible}
Step3: Compute the interval [Pmin,Pmax] of $P(Pa)$
Product1_2InPath()
for each $i = 1$ to 2^{t2} **do**
  CombinationI(i)
  Pmin =min(Pmin,[i].lb)
  Pmax =max(Pmax,[i].ub)
end for

Store the interval $[P_F.lb, P_F.ub]$ of $P(Pa)$, $P_F.lb$ =Pmin and $P_F.ub$=Pmax in the output file

Algorithm 2 Split the variables in 3 types

SplitVariables(Boolean_formula){Split the variables in 3 types from any Boolean Formula}
Iterator $i = 0$
for each variable $v \in V$ **do**
  if Only $\neg v$ appears in F **then**
    set $Type(Boolean\_formula)[i] = 0$ {v is type 0}
  else if Only v appears in F **then**
    set $Type(Boolean\_formula)[i] = 1$ {v is type 1}
  else if v and $\neg v$ appears in F **then**
    set $Type(Boolean\_formula)[i] = 2$ {v is type 2}
  end if
  $i = i + 1$
end for
ComparisonOfCat(Boolean_formula1, Boolean_formula2) {To keep as few variables from Type 2 as possible}
Iterator i=0
for Each variable $v \in V$ **do**
  if Type(Boolean_formula1)[i]= 0 or Type(Boolean_formula2)[i]= 0 **then**
    set $v.type = 0${v is type 0}
  else if Type(Boolean_formula1)[i]= 1 or Type(Boolean_formula2)[i]= 1 **then**
    set $v.type = 1${v is type 1}
  else
    $v.type = 2${v is type 2}
    $t2 = 2 + 1$
  end if
  $i = i + 1$
end for

8.1.3 BDD-based fault tree analysis

In the context of fault tree analysis, we take the fault tree generated by Cécilia OCAS software, i.e. a complex Boolean formula, and parse it to develop a BDD representing this fault tree. Afterwards, we generate all paths of this BDD leading to 1 i.e over which top formula evaluates to true. Then we traverse these paths to do the computations presented in 5.4. As discussed earlier, these computations have exponential complexity with respect to number of variables whose monotonicity is unknown, therefore before performing these computations, we try to reduce number of unknown monotonicity

Algorithm 3 Product1_2InPath

Product1_2InPath() {Compute for each path the products that are constant (Type 0, Type 1)}
for each $path \in \mathcal{P}$ **do**
$V_{path} = \{v \mid v \in V \wedge v \in path\}$ {v belongs to this path}
$V^0_{path} = \{v \mid v \in V_{path} \wedge v.type = 0\}$, $V^1_{path} = \{v \mid v \in V_{path} \wedge v.type = 1\}$
$V^{1+}_{path} = \{v \mid v \in V^1_{path} \wedge v.value = 1\}$, $V^{1-}_{path} = \{v \mid v \in V^1_{path} \wedge v.type = 0\}$
$V^{0+}_{path} = \{v \mid v \in V^0_{path} \wedge v.value = 1\}$, $V^{0-}_{path} = \{v \mid v \in V^0_{path} \wedge v.type = 0\}$
$V^2_{path} = \{v \mid v \in V_{path} \wedge v.type = 2\}$
if length(V^2_{path})=length(V_{path}) **then**
$path.c1\_2min = 1$, $path.c1\_2max = 1$ {There is no variable of type 0 or type 1 in this path}
else
$path.c1\_2min = \prod_{v \in V^{1+}_{path}} v.lb * \prod_{v \in V^{1-}_{path}} (1 - v.lb) * \prod_{v \in V^{0+}_{path}} v.ub * \prod_{v \in V^{0-}_{path}} (1 - v.ub)$
$path.c1\_2max = \prod_{v \in V^{1+}_{path}} v.ub * \prod_{v \in V^{1-}_{path}} (1 - v.ub) * \prod_{v \in V^{0+}_{path}} v.lb * \prod_{v \in V^{0-}_{path}} (1 - v.lb)$
end if
end for

Algorithm 4 CombinationI

{Compute one element of the vector [i], $i \in \{1, ..2^{t2}\}$}
$[i].lb = 0$, $[i].ub = 0$
$(z_1, z_2, \ldots z_{t2}) \leftarrow binary(i)$
for $j = 1$ to $t2$ **do**
if $z_j = 0$ **then**
$v_j.c = v_j.lb$ {The lower bound is used for the corresponding $v \in V^2$}
else
$v_j.c = v_j.ub$ {The upper bound is used for the corresponding $v \in V^2$}
end if
end for
for each $path \in \mathcal{P}$ **do**
$V^{2+}_{path} = \{v \mid v \in V^2_{path} \wedge v.path.value = 0\}$ {positive literals of Cat3 in the path}
$V^{2-}_{path} = \{v \mid v \in V^2_{path} \wedge v.path.value = 1\}$ {negative literals of Cat3 in this path}
$path.t2 = \prod_{v \in V^{2+}_{path}} v.c * \prod_{v \in V^{2-}_{path}} (1 - v.c)$ {A same value of bound is used for all paths of the BDD for this combination}
{Compute the min and max value of this path, using the values calculated by Product1_2}
$path.t2\_min = path.c1\_2min * path.t2$
$path.t2\_max = path.c1\_2max * path.t2$
$[i].lb = [i].lb + path.t2\_min$ {Add the lower bound of this path for the combination i}
$[i].ub = [i].ub + path.t2\_max$ {Add the upper bound of this path for the combination i}
end for

variables. For this purpose we developed few different methods. These methods are described in sections below.

8.1.3.1 Single BBD-based analysis using fixed variable order

Since we use BuDDy package to develop BDD, BuDDy explores the order of variables of top formula in order to optimize the size of the BDD. This optimization is not rigorous and is very basic one. This results in a selection of one variable order and its corresponding BDD.

In order to determine the monotonicity of top formula with respect to each variable, we use paths of the BDD to check if a given variable appears positively, negatively or both. If a variable appears only positively, the top formula is monotonically increasing, if the variable appears only negatively then top formula is monotonically decreasing and if the variable appears both positively and negatively, then we declare this variable as unknown type of monotonicity. This algorithm is shown in 1. As the Boolean formulas to deal with are very large, the default order of variables chosen by BuDDy package is not efficient enough to reduce variables of unknown type of monotonicity. But still it does effectively calculates monotonicity of some variables.

8.1.3.2 Multiple BBD-based analysis using different variable orders

In order to determine monotonicity of maximum number of variables, we tried to override the default variable order of BuDDy package and introduced our own variable orders. In this approach we use different variable orders and generate corresponding BDDs. Theoretically we could use all possible variable orders and corresponding BDDs but it requires lot of computational time and resources. We adapted a single heuristic. We take the order of variables as they appear in top formula and then shift each variable to right by one. We repeat this process for the number of total variables present in the top formula. This heuristic generates the number of BDDs equal to the number of variables in the top formula.

The principle consist in the fact that everytime that a BDD is parsed and the variables splited in three categories, chances to find information about the monotonicity of the variables increases. It is not possible to estimate in advance the number of variables whose monotonicity will be determined at each parsing, but it is a certain way to find out more information than with a single-BDD approach. Then, the best order and its corresponding BDD is chosen among these BDDs as the one which has least number of paths: these paths are used for the rest of the probability computations.

We have found this heuristic to be very efficient in our results, as we will discuss in the case study section later on. Though we did not explore many heuristics but other kind of heuristics can also be used with this software.

8.1.3.3 Simplification of top formula using binary mathematics

Another quite different approach to reduce the number of variables of unknown monotonicity is to simplify the top formula, which is a boolean formula, by using boolean arithmetic. In order to achieve this simplification, we adapted an open source *Python* library called *simbool*. Basic idea behind this approach is to reduce number of literals in the top formula by boolean simplification. After obtaining the simplified formula, we use this simplified formula to generate BDD and corresponding paths instead of default top formula obtained from fault tree generated by Altarica.

This technique gives mixed results. Even though we were able to reduce the number of laterals by a factor approaching 60, in certain cases, after boolean simplification but resulting BDDs were not able to decrease more number of variables of unknown monotonicity type as compared to multiple BBD-based analysis using different variable orders.

8.1.4 Minimal cut sets based analysis

In this approach we use minimal cut sets to evaluate exact bounds of probability. Minimal cut sets are generated by OCAS software in Aralia format as shown in Appendix C. In this case top formula is always monotonic. Hence, this approach is not affected by monotonicity and related computational complexity. For example, in the case study presented in next section, the loss of the rudder is considered as a catastrophic event, so following the safety requirements, the order of minimal cut sets of this loss should be at least 3. We can verify it with the OCAS tool of sequence generation. For example, the sequence ('B.loss' & 'Bus_4PP.loss' & 'Hyd _Y.loss') leads to the loss of the rudder. All sequences generated by OCAS are presented in Appendix C.

8.2 Case Study

In this section we will present the result of our implemented software and its application to a real life example. The example chosen as case study is taken from aeronautics domain. It is the safety model of the working of Rudder of an aircraft. It is a pertinent model in term of safety because it takes into account some reconfiguration and supports many type of redundancies such as active redundancy, passive redundancy, and some which implies priorities as well.

8.2.1 Rudder System - the composition

The Rudder system is composed of:

1. Calculators
   - 3 primary calculators (P1, P2, P3)
   - 1 secondary calculator (S1)
   - 1 emergency autonomous equipment, constituted by a Back-up Control Module (BCM) and 2 Back-up Power Supply (BPS_B, BPS_Y)

2. Servo-commands
   - 3 servo-commands: S/C Green (G), S/C Blue (B), S/C Yellow (Y)

3. External power supply
   - 2 electric sources (Bus 2PP, Bus 4PP)
   - 3 hydraulic sources (Hyd Green, Hyd Blue, Hyd Yellow)

The OCAS model of this system is represented in fig. 8.1.

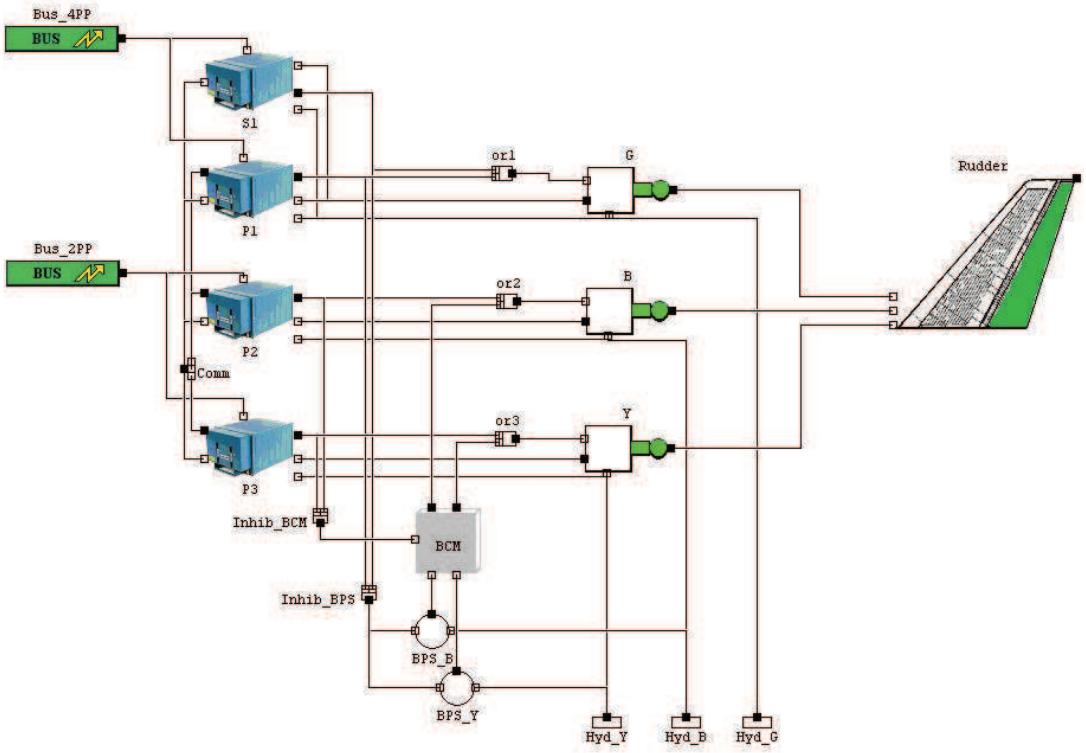


Figure 8.1: Rudder's OCAS model

8.2.2 System behavior

In absence of failures, the rudder will use the 3 hydraulic sources, all servo-commands will be active, the secondary calculator will be inactive and the emergency autonomous equipment as well. Some hidden failures can appear on the inactive components, and active failures on the active one.

When a calculator detect that there is failure on itself on or the servo-command that it controls, it will disengage. If the 3 calculators disengage, then the secondary calculator will command the Green servo-command.
If the secondary calculator fails as well, then the BCM will command in priority the Yellow servo-command, and the Blue in case of failure of the Yellow one. The components *Inhin_BCM, Inhib_BFS, Comm* and or_i are virtual components that has been created to make some logic between input conditions. For example, the output of or_3 is activated if the primary calculator P3 *or* the BCM are working properly. Then the servo-command Yellow (Y) will be activated if the Hyd_Y is Ok.

8.2.3 Fault tree analysis of Rudder system

In this section we will use our software to analyze the rudder system presented in previous section. We will also compare the result of different techniques we implemented in order to reduce number of variables of unknown type of monotonicity and hence the computational complexity.

In the first step, we will take the fault tree generated by the Altarica software and parse it get the top formula as a boolean formula of its variables. This process is explained in the flow chart in Appendix D. Let us designate each technique as a case: case 'A' represents the method where we use single BBD-based analysis using fixed variable order, case 'B' represents the method where we use

multiple BBD-based analysis using different variable orders and case 'C' represents the method where we use simplification of top formula using binary mathematics. The Rudder system has 19 variables in its top formula obtained from the fault tree (without boolean simplification), and consists of 3947 number of literals.

This model is used for safety analysis and it is has also been used as part of the operational reliability analysis. It is genuinely non-monotonic because of the explicit use of states that do not refer to failures. The failures (elementary events) taken into account are: loss of a component (e.g. B.loss means loss of the Y servo-command), a hidden failure and an active failure (that can occur in S1 and BCM). An excerpt from the detailed generated fault tree in the Aralia format, for the loss of the first primary calculator P1 is given on Appendix C.3. Sub-trees are automatically generated and given names (DTN3578, etc.), one per line in the figure.

The BDD representation of F, with alphabetical order of the variables has 640 paths and are displayed in Appendix E.1. Fig. E.1 depicts the BDD associated to these paths.

The probabilities of some variables are known: P(Hyd_i.loss)= $e^{-0.004t}$, P(Bus_iPP)= $e^{-0.0003}$. But for some others, only an interval containing the probability values is known: i.[lb,ub] =$e^{-[0.0015,0.0025]t}$, $i \in \{Y, B, G\}$, P$_i$[lb,ub]=$e^{-[0.0004,0.0005]t}$, $i = 1, \ldots, 3$,
S1.Active_failure[lb,ub]=[0.015, 0.0345], Hidden_failure =[0.005, 0.007] .

Using the algorithm described in section 7.2.5 on the whole fault-tree (that is non-monotonic) we for instance obtain the interval $I_1 = [0.00742951, 0.00965619]$ after 20 flighthours. With the extended algorithm described in section 7.2.5, it is possible to see the evolution of the probability thought time, as shown on the Fig. 8.2.

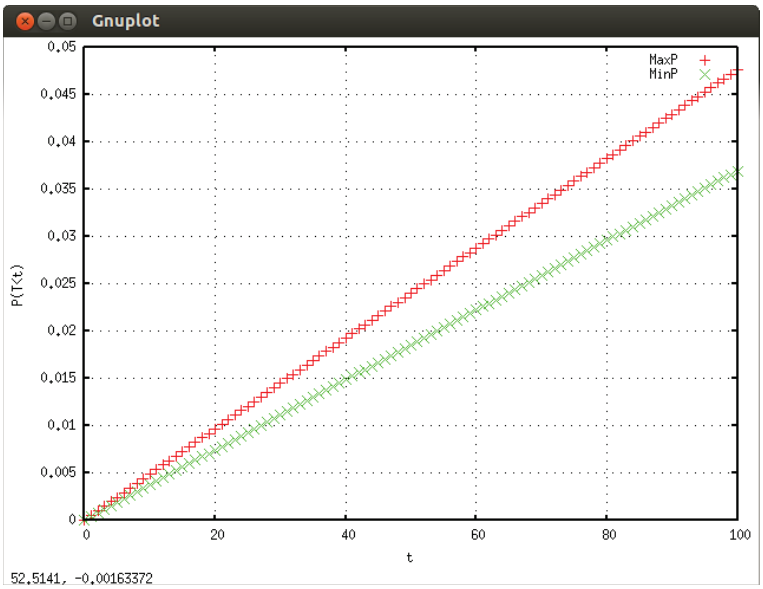


Figure 8.2: Evolution of the probability of P1.hs

8.2.4 Comparison of the different cases

In this section we will present the results of the case study of Rudder system. The table 8.2.4 summarizes the results of three different methods i.e case 'A', 'B' and 'C' evaluated on the Rudder system.

In case 'A', the default order of variables and resulting BDD is able to determine only 3 variables for their monotonicity, so there remain 16 variables as unknown type of monotonicity out of 19 total variables. This leads us to 2^{16} computations.

In case 'B', we use 19 different variable orders, as described by the heuristics presented in section 8.1.3.2. Among these 19 orders we chose the one which gives us minimum number of paths. This approach is able to resolve 16 variables for their monotonicity. 3 variables were still unknown for their monotonicity. This leads us to 2^3 number of cases for computations. This is drastically less than that of case 'A' but we spend some extra computations in evaluation of best BDD and variable order. Compared to case 'A', this time is much less than the time consumed in computing 2^{16} computations of case 'A'.

In case 'C', boolean simplification also requires considerable amount of computations. For the Rudder system example, top formula simplification was huge. We were able to reduce number of literals to 67 from 3947 of the original formula, which is of the order of 60. But it took almost 1 hour to achieve this simplification. After the simplification, the formula was used to generate BDD and resulting paths. This approach was able to find monotonicity for 13 variables, leaving behind 6 variables for unknown type of monotonicity.

Comparing the results shown in 8.2.4, we can see that computational complexity is exponential with respect to variables of unknown monotonicity. As discussed in section 8.1.1, the algorithm to find monotonicity of variables requires paths of the BDD representing the top formula. Then, the aim is to find the BDD which results in paths able to resolve monotonicity of maximum number of variables. Such BDD, in turn, depends on order of the variables. Hence, the main objective is to find the order of variables that leads us to minimum number of variables of unknown monotonicity. At present we still need more insight on what can lead us to the right variable order. For the given example of Rudder system, clearly the case 'B' takes least amount of time over all, and hence is the most efficient method.

Case	No. of Variables	No. of Literals	Variables resolved	Total Time
A	19	3947	3	4 hours
B	19	3947	16	<1 minute
C	19	67	13	1 hour

All of these methods are not always able to determine the monotonicity of all the variables, but they improve significantly the computation time. They can be more or less efficient, depending on the Boolean formula that they have to deal with. The advantage of the Boolean formula reduction is that once that it is done, it can be used to deal with all the scenarios done from this formula. But a reduction of the formula size do not imply any size reduction in the BDD, when the variables are taken in the same order.

The multiple-BDD principle is based on randomness, hence the improvement in computation time is fluctuating. But in the case of the rudder, it gives a very impressive reduction of computation time. It is possible to add different orders and respective associated BDDs in order to have more chances to determine the monotonicity of maximum possible variables. Any additional knowledge from the user about the monotonicity of variables can also be taken into account in the model.

Chapter 9

Conclusion

This thesis work was initiated with the main objective to improve the uncertainty management in existing risk analysis techniques for the main application to preventive maintenance of aircraft. Uncertainty can have many different kinds of forms and is always present in risk management. Precise probability laws, mostly used in existing techniques of quantitative safety quantitative analysis, do not cover all forms of uncertainty that may exist in models. To overcome this limitation, this work was done in order to be able to cope more facets of uncertainty and propagate them into the models used. This thesis proposed the use of different uncertainty management techniques in order to propagate probability intervals in Fault Tree Analysis, which is a well known approach for safety analysis. In this work, Boolean formulas are used as the basis of models for uncertainty management studies, as they can be used for representing a fault tree.

9.1 Contributions

The first part of this work was to formalize the problem of computing the probability interval of a Boolean formula representing a faulty event in terms of probability intervals of failures of components, under the assumption of stochastic independence. When using the theory of imprecise probabilities instead of classical theory of probability, the problem of computing the probability of Boolean formulas becomes tricky and taking into account partial ignorance can add complexity to the computations.

Applying naive interval arithmetic directly to Boolean formula may result in a very imprecise interval due to repetitions of a variable and its negation. In Chapter 5, we have developed an efficient algorithm based on monotonicity analysis to reduce this imprecision. Our algorithm can compute the exact range of the probability of any formula without adding any *artificial* uncertainty (i.e. uncertainty induced by the computation itself, not by the inputs). Hence, we could prove that this range is always smaller than the one computed applying naive interval arithmetic. The downside of this algorithm is that the computations remains exponential with the number of variables that are found appearing both positively and negatively in the expression. Some heuristics trying to decrease it (restricting exponential complexity to variables variables with respect to whom the monotonicity of the formula is hard to determine) are discussed in Chapter 8. The intervals used in this approach may be coming from probability laws with imprecise parameters, as described in Chapter 7.

Our second contribution is alternative formulation in the theory of belief functions under assumption of information source independence, and determination of cases where the import export formula can be used (Chapter 6). Belief functions can be helpful, because it proposes a way to propagate probability intervals, without making any assumption about the dependence of the variables, but only about the sources of information that the probability values come from (statistical studies, experts, etc.). Using Belief functions for computing the resulting probability interval of a Boolean formula is also an NP-hard problem. In this thesis, cases where the computation can be straightforward are described.

The third contribution of this thesis is the comparison of the two approaches mentioned before, and the application to examples from aircraft maintenance (Chapter 7 and 8). Both approaches can be applied for uncertainty management in aircraft maintenance in order to cover more facets of uncertainty with better results as compared to existing techniques. Depending on the context, those two approaches can be equivalent, or complementary.

9.2 Perspectives

The work done during this thesis can be used in the scope of a decision making process, *i.e.* the intervals of probabilities for a failure condition computed by our developed algorithms can be used as an indicator for supporting decisions. For example, the width of an interval (or of a p-box) can be related to the quality of the result. A wider interval suggests lesser confidence about the result and vice versa. Similarly, other indicators exist as well. Further research should be carried out to figure out all the indicators with varying degrees of influence on the final decision. It will also be interesting to propagate the information contained in these imprecise probabilities along the different stages of decision making process in order to be able to make the best possible decision. Intervals and p-boxes are also a way to indicate a part of ignorance in the results, hence this aspect will have to be taken into account. An extension to fuzzy numbers would also be interesting in order to model and propagate epistemic uncertainty.

In the belief function approach, we demonstrated that when a Boolean formula satisfies certain properties, it is possible to use computations that are very similar to the classical probability computations in order to compute the belief and the plausibility of the formula. Some more investigations can be done in order to find practical methods to prove that the formula satisfies those properties. This could lead to a huge improvement of the computation time, and a better scalability of this framework for complex systems. It would also be very interesting to try to compute probability intervals of failure conditions under weaker independence assumptions, and to compare the results with those of the approaches described in this thesis.

The last point is that this thesis focuses on the uncertainty management in Fault Tree Analysis. But one of the major drawback of this approach is that it is static, and it does not cope with the dynamic part of safety analysis. The dynamic part of safety analysis can be handled by the algorithms developed by ONERA in the context of @MOST project (Teichteil-Königsbuch et al., 2011). Those algorithms are based on Markov chains. In order to cope with uncertainty in dynamic part of Safety analysis, a good research direction can be to study the use of imprecise Markov chains and their application to dynamic safety analysis (Cooman et al., 2007).

Bibliography

http://www.isograph-software.com/2011/software/reliability-workbench/fault-tree-analysis/.

http://www.riskspectrum.com/en/risk/meny_2/riskspectrum_fta/.

http://grif-workshop.com/grif/tree-module/.

F. Aguirre, M. Sallak, and W. Schön. Prise en compte des incertitudes dans les études de fiabilité: Comparaison entre le modèle de croyances transférables et les méthodes de simulations de monté-carlo. In *Qualita 2011*, pages 80–80, 2011.

R. Amalberti. *La conduite des systèmes à risques*. PUF, 1996.

A. Arnold, A. Griffault, G. Point, and A. Rauzy. The altarica language and its semantics. *Fundamenta Informaticae*, 34:109–124, 2000.

R.E. Barlow and F. Proschan. *Statistical theory of reliability and life testing: probability models*. Holt, Rinehart and Winston, 1975.

R.E. Barlow and F. Proschan. *Mathematical theory of Reliability*. John Wiley and sons, 65.

C. Baudrit, D. Dubois, and D. Guyonnet. Joint propagation and exploitation of probabilistic and possibilistic information in risk assessment. *IEEE T. Fuzzy Systems*, 14(5):593–608, 2006.

C. Baudrit, I. Couso, and D. Dubois. Joint propagation of probability and possibility in risk analysis: Towards a formal framework. *Int. J. Approx. Reasoning*, 45(1):82–105, 2007.

C. Baudrit, D. Dubois, and N. Perrot. Representing parametric probabilistic models tainted with imprecision. *Fuzzy Sets and Systems*, 159(15):1913–1928, 2008.

R.A. Brualdi. *Introductory Combinatorics*. Pearson/Prentice Hall, 2004.

G. De Cooman, F. Hermans, and E. Quaeghebeur. Limit behaviour for imprecise markov chains. pages 16:14–16:14, 2007.

I. Couso and S. Moral. Independence concepts in evidence theory. *Int. J. Approx. Reasoning*, 51(7): 748–758, September 2010. ISSN 0888-613X.

I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5(02), 2000.

F. G. Cozman. Computing lower expectations with kuznetsov's independence condition. In *ISIPTA*, pages 177–187, 2003.

F. G. Cozman, C. P. de Campos, and J. C. Ferreira da Rocha. Probabilistic logic with strong independence. In *IBERAMIA-SBIA*, pages 612–621, 2006.

F. G. Cozman C.P. de Campos. Computing lower and upper expectations under epistemic independence. *Int. J. Approx. Reasoning*, 44(3):244–260, 2007.

C. P. de Campos and F. G. Cozman. Computing lower and upper expectations under epistemic independence. In *ISIPTA*, pages 78–87, 2005.

C. Dejours. *Le facteur humain*. PUF, 1996.

S. Destercke, D. Dubois, and E. Chojnacki. Unifying practical uncertainty representations: I. generalized p-boxes. *International Journal of Approximate Reasoning*, 49:649–663, 2008.

D. Dubois and H. Prade. Combination and propagation of uncertainty with belief functions - a reexamination. In *IJCAI*, pages 111–113, 1985.

D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, 1988.

D. Dubois, H. Fargier, and H. Prade. Comparative uncertainty, belief functions and accepted beliefs. In *UAI*, pages 113–120, 1998.

D. Dubois, H. Prade, and H.T. Nguyen. Possibility theory, probability and fuzzy sets: misunderstandings, bridges and gaps. *Fundamentals of Fuzzy Sets*, pages 343–438, 2000.

M. Sallak C. Jacob D. Dubois F. Aguirre, S. Destercke. Inclusion/exclusion principle for belief functions. In *ISIPTA'13*, 2013.

T. Augustin F. Coolen, M. Troffaes. Imprecise orobability. *International Encyclopedia of Statistical Science*, 2010.

T. Denoeux F. Pichon, D. Dubois. Relevance and truthfulness in information correction and fusion. *International Journal of Approximate Reasoning*, 53(2):159–175, 2012.

S. Ferson and L.R. Ginzburg. Different methods are needed to propagate ignorance and variability. *Reliability Engineering and Systems Safety*, (54):133–144, 1996.

S. Ferson, V. Kreinovich, L. Ginzburg, D.S. Myers, and K. Sentz. Constructing probability boxes and dempster-shafer structures. *SAND2002-4015*, (January):1–143, 2003.

T. Fetz. Sets of joint probability measures generated by weighted marginal focal sets. In *ISIPTA'01*, pages 171–178, 2001.

J. Fortin, D. Dubois, and H. Fargier. Gradual numbers and their application to fuzzy interval analysis. *Trans. Fuz Sys.*, 16(2):388–402, April 2008.

J. Pearl G. Shafer, editor. *Readings in uncertain reasoning*. Morgan Kaufmann Publishers Inc., 1990.

J.-C. Geffroy and G. Motet. *Sûreté de fonctionnement des systèmes informatiques*. Dunod, 1998.

M.A.S. Guth. A probabilistic foundation for vagueness & imprecision in fault-tree analysis. *IEEE Transactions on Reliability*, 40:563–571, 1991.

P. Hansen, B. Jaumard, M. Poggi de Aragão, F. Chauny, and S. Perron. Probabilistic satisfiability with imprecise probabilities. *Int. J. Approx. Reasoning*, pages 171–189, 2000.

J.W. Hines and A. Usynin. Current computational trends in equipment prognostics. *International Journal of Computational Intelligence Systems*, 1(1):94–102, 2008.

C. Jacob, D. Dubois, and J. Cardoso. Uncertainty handling in quantitative bdd-based fault-tree analysis by interval computation. In *SUM*, pages 205–218, 2011.

C. Jacob, D. Dubois, and J. Cardoso. Evaluating the uncertainty of a boolean formula with belief functions. In *IPMU (3)*, pages 521–531, 2012a.

C. Jacob, D. Dubois, and J. Cardoso. From imprecise probability laws to fault tree analysis. In *SUM*, pages 525–538, 2012b.

G.J. Klir. *UNCERTAINTY AND INFORMATION - Foundations of Generalized Information Theory.* JOHN WILEY and SONS, 2006.

V. Kreinovich, C. Jacob, D. Dubois, J. Cardoso, and M. Ceberio. Estimating probability of failure of a complex system based on partial information about subsystems and components, with potential applications to aircraft maintenance. In *SKAD'11*, pages 30–41, 2011a.

V. Kreinovich, C. Jacob, D. Dubois, J. Cardoso, M. Ceberio, and I. Z. Batyrshin. Estimating probability of failure of a complex system based on inexact information about subsystems and components, with potential applications to aircraft maintenance. In *MICAI (2)*, pages 70–81, 2011b.

V. Kreinovitch, C. Jacob, D. Dubois, J. Cardoso, and M. Ceberio. Failure analysis of a complex system based on partial information about subsystems, with potential applications to aircraft maintenance. *Applied and Computational Mathematics, Artificial Intelligence and Soft Computing*, 11(2):165–179, 2012.

C. Lievens. *Sécurité des systèmes.* Cépaduès-Éditions, 1976.

M.R. Lyu. *Handbook of Software Reliability Engineering.* Mc Graw Hill, 1996.

L. Magne and D. Vasseur. *Risques industriels.* Tec et Doc, 2006.

A. Majdara and T. Wakabayashi. Component-based modeling of systems for automated fault tree generation. *Rel. Eng. & Sys. Safety*, 94(6):1076–1086, 2009.

N. Mazars. *De l'indépendance et des dépendances stochastiques en théorie de la fiabilité.* PhD thesis, Université Paul Sabatier, 1981.

Victoria Montgomery. *New statistical methods in risk assessment by probability bounds.* PhD thesis, Durham University, 2009.

E. Morice. Détermination des paramètres du modèle de weibull à partir de la méthode de l'actuariat. *Revue de statistique appliquée*, 37(4):5–39, 1989.

R. E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, 2003.

R.B. Nelsen. *An Introduction to Copulas (Lecture Notes in Statistics)*. Springer, 1998.

J. R. Norris. *Markov chains*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.

Osys. http://www.o-sys.com/products-services/fault-tree-analysis/.

A. Pages and M. Gondran. *Fiabilité de systèmes*. Eyrolles, 1976.

A. Pfeffer. Figaro: An object-oriented probabilistic programming language. 2009.

G. Point and A. Rauzy. AltaRica: Constraint automata as a description language. *Journal Européen des Systèmes Automatisés*, 33(8–9):1033–1052, 1999.

Gérald Point. *AltaRica : Contribution à l'unification des méthodes formelles et la sureté de fonctionnement*. PhD thesis, UNIVERSITÉ BORDEAUX I, 2000.

J.H. Purba, J. Lu, D. Ruan, and G. Zhang. A hybrid approach for fault tree analysis combining probabilistic method with fuzzy numbers. In *Proceedings of the 10th international conference on Artificial intelligence and soft computing: Part I*, ICAISC'10, pages 194–201. Springer-Verlag, 2010.

K. Durga Rao, V. Gopika, V. V. S. Sanyasi Rao, H. S. Kushwaha, A. K. Verma, and Ajit Srividya. Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment. *Rel. Eng. & Sys. Safety*, 94(4):872–883, 2009.

A. Rauzy. Modes automata and their compilation into fault trees. *Reliability Engineering and System Safety*, 78:1–12, 2002.

M.J. Cloud R.E. Moore, R.Baker Kearfott. *Introduction to interval analysis*. Cambridge University Press, 2009.

P. Ribot. *Vers l'intégration diagnostic/prognostic pour la maintenance des systèmes complexes*. PhD thesis, LAAS, 2009.

T. Denoeux S. Petitrenaud. Non-parametric regression analysis of uncertain and imprecise data using belief functions. *International Journal of Approximate Reasoning*, 35(1):1–28, 2004.

M. Sallak, C. Simon, and J.-F. Aubry. A fuzzy probabilistic approach for determining safety integrity level. *Trans. Fuz Sys.*, 16(1):239–248, 2008. ISSN 1063-6706.

M. Sallak, W. Schon, and F. Aguirre. The transferable belief model for reliability analysis of systems with data uncertainties and failure dependencies. *Proceedings of the Institution of Mechanical Engineers Part O Journal of Risk and Reliability*, 4/2010(224):266–278, Aout 2010.

M. Sallak, W. Schön, F. Aguirre, and Sébastien Destercke. Principe d'inclusion-exclusion pour fonctions de croyance : application Ã la fiabilite. inclusion-exclusion principle for belief functions : application to reliability. In *LFA 2012 - Logique floue et ses applications*, 2012.

D.H. Stamatis. *Failure Mode and Effect Analysis: Fmea from Theory to Execution*. ASQ Quality Press, 2003.

F. Coolen G. de Cooman T. Augustin, M. Troffaes. *Introduction to imprecise probabilities*. Wiley-Blackwell, 2014.

M-H. Masson T. Denoeux, editor. *Belief Functions: Theory and Applications*, volume 164 of *Advances in Soft Computing*, 2012. Springer.

F. Teichteil-Königsbuch, V. Vidal, and G. Infantes. Extending classical planning heuristics to probabilistic planning with dead-ends. In *AAAI*, 2011.

P. Thomas. *Contribution à l'approche booléenne de la sûreté de fonctionnement: L'atelier logiciel Aralia Workshop*. PhD thesis, Bordeaux 1, 2002. Th.: automatique, productique.

L. Utkin and S. Destercke. Computing expectations with p-boxes: two views of the same problem. In Gert De Cooman, Jirina Vejnarova, and Marco Zaffalon, editors, *International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Prague, 16/07/07-19/07/07*, pages 435–445. The Society for Imprecise Probability: Theories and Applications (SIPTA), juillet 2007.

L.V. Utkin. Interval reliability of typical systems with partially known probabilities. *European Journal of Operational Research*, 153(3):790–802, 2004.

L.V. Utkin and F.P.A. Coolen. Imprecise reliability: An introductory overview. In *Intelligence in Reliability Engineering*, volume 40 of *Studies in Computational Intelligence*, pages 261–306. Springer, 2007.

A. Villemur. *Sûreté de fonctionnement des systèmes industriels, Fiabilité - Facteurs Humains - Informatisation*. Eyrolles, 1988.

A. Voisin, E. Levrat, P. Cocheteux, and B. Iung. Generic prognosis model for proactive maintenance decision support: application to pre-industrial e-maintenance test bed. *Journal of Intelligent Manufacturing*, 2009.

A. Volkanovski, M. Cepin, and B. Mavko. Application of the fault tree analysis for assessment of power system reliability. *Rel. Eng. & Sys. Safety*, 94(6):1116–1127, 2009.

P. Walley. Towards a unified theory of imprecise probability. In *International Symposium on Imprecise Probabilities and Their Applications*, 1999.

R. Winckell. Compilation du langage altarica vers les fonctions booléennes, 1998.

R. Yager. An introduction to applications of possibility theory. *Human Systems Management*, 3(4), 1982.

E. Zio and P. Baraldi. A combined monte carlo and possibilistic approach to uncertainty propagation in event tree analysis. *Risk Analysis*, 28(5):1309–1326, 2008.

Appendix A

Mathematical definitions for RAMS

The aim of this section is to recall some basics about the mathematical modeling used for the quantitative studies of reliability (Barlow and Proschan, 65).

Definition A.1 (Reliability)
The *reliability* $R(t)$ of a system, also called the *survival function*, is the probability that the system does not fail before time t. It can be expressed as:

$$R(t) = P(T > t) \tag{A.1}$$

where T is a random variable representing the *failure date*. ♢

The reliability expresses the fact that a system is performing the required mission under given conditions for a given time interval [0,t].

Definition A.2 (Probability of failure)
The *probability of failure* of a system before time t, called *failure distribution* $F_T(t) = P(T \leq t)$, is the complement of its reliability:

$$F_T(t) = 1 - R(t) \tag{A.2}$$

♢

Definition A.3 (Failure density function)
The *failure density function* $f_T(t)$ expresses the probability that the system fails between t and $t + dt$:

$$f_T(t)dt = P(t \leq T < t + dt) \tag{A.3}$$

♢

Definition A.4 (Failure rate)
The *failure rate* λ of a system is the frequency of its failure. λ is often considered as proportional to the probability that a failure occurs at a specified time point t, given that no failure occurred before this time:

$$\lambda(t)dt = P(t \leq T \leq t + dt \mid T > t) \tag{A.4}$$

♢

The failure rate is a function of the system health state, and in general it is time dependent. It is a value for which statistical studies can be carried out, because it is a frequency (Barlow and Proschan, 1975).

The conditional probability of the failure rate can be written as:

$$\lambda(t)dt = \frac{f_T(t)dt}{R(t)} = \frac{-R'(t)}{R(t)}, \tag{A.5}$$

where $R'(t)$ is the derivative of $R(t)$ with respect to the time.

The solution of this differential equation is:

$$\ln(R(t)) = \int_0^t \lambda(u)du + c, \text{where } c \text{ is a constant.} \tag{A.6}$$

Hence, the reliability expressed in terms of the failure rate has the expression:

$$R(t) = e^{-\int_0^t \lambda(u)du} \tag{A.7}$$

With the expression of the reliability as function of the failure rate, it is possible to see the evolution of the reliability from the statistics inputs that we have on the failure rates. The shape of the failure rate will define the behavior of the reliability function.

From the probability distribution of the reliability function, it is possible to compute some useful indicators about the failure times of a component.

Definition A.5 (Mean Time To Failure (MTTF))
Mean time to failure (MTTF) measures average time to failures with the modeling assumption that the failed system is not repaired. ♢

Definition A.6 (Mean Time Between Failure (MTBF))
Mean Time Between Failures (MTBF) is the predicted elapsed time between inherent failures of a system during operation. MTBF can be calculated as the arithmetic mean time between failures of a system. The MTBF is typically part of a model that assumes the failed system is immediately repaired, as a part of a renewal process. ♢

Definition A.7 (Remaining Useful Life (RUL))
The Remaining Useful Life (RUL) of a component/equipment/system, is defined as the expected time after which it will fail. Fig. A.1 depicts how the predictions are used in order to evaluate this remaining time. ♢

MTTF, MTBF and RUL are values that are often used in Prognosis 2.4. They can be computed from the probability distribution of the component, but the Prognostic analysis consists in reassessing them with the evolution of the state of the system.

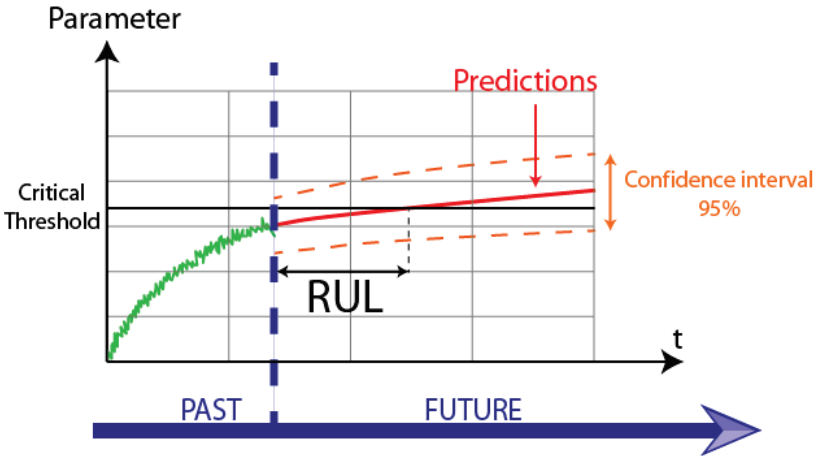


Figure A.1: Remaining Useful Life (RUL)

Appendix B

Possibility Theory

In the Possibility Theory, information is represented by a possibility distribution. It is a way to represent the knowledge of one agent related to the state of the World. The formal definition of a possibility distribution is that for:

U: Set of states of the world
x: Ill-known variable
L: Possibility scale ([0,1], finite interval, ...)

$$\left.\begin{array}{l}\\ \\ \\ \end{array}\right\}\pi_x : \begin{array}{l} U \to L \\ u \mapsto \pi_x(u) \end{array} \quad \text{such that:}$$

$$\pi_x(u) \in L, and\ \exists u\ s.t. \pi_x(u) = 1$$

By convention:
$\pi_x(u) = 0$ iff x = u is impossible
$\pi_x(u) = 1$ iff x = u is possible
$\pi_x(u) < \pi_x(u') \Leftrightarrow$ x = u' is more plausible than x = u

Let's take an example: we take the concept of "teenager", and we want to represent it on an age scale.
We consider a teenager:
- If he is more than 20 and less than 12, he is definitely not a teenager. The possibility distribution will be null for those values.
- If he is called teenager, than it is very possible that he is between 15 and 18, so the possibility for those values will be 1, it is the core of the distribution.
- If he is in [12,15] (resp. [18,20]), the more he will be close to 15 (resp. 18), the more chances are that he will be considered as a teenager.

We obtain the possibility distribution depicted in fig. B.1:

Some particular cases of fuzzy sets:

- **Precise information:** x = u_0 : $\pi_x(u) = 1$ if $u = u_0$, and 0 otherwise (fig. B.2.a)
- **Incomplete but clear information:** x ∈ [a,b] : $\pi_x(u) = 1$ if u ∈ [a,b], and 0 otherwise (fig. B.2.b)

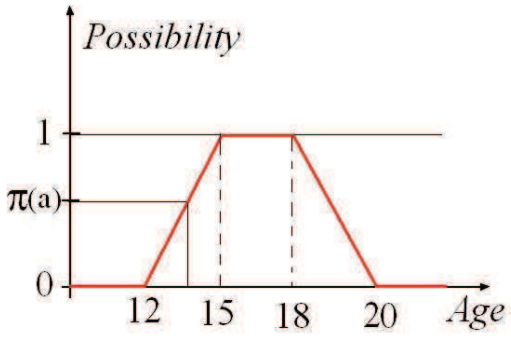


Figure B.1: Fuzzy set representing the concept of "Teenage"

- **Total ignorance:** : $\pi_x(u) = 1 \ \forall \ u$ (fig. B.2.c)

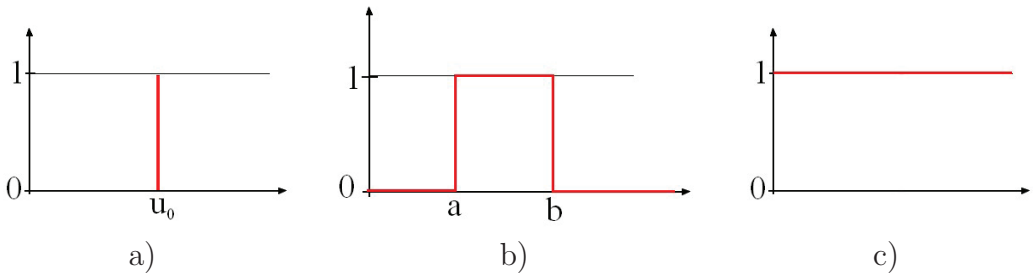


Figure B.2: Representation of: a) precise information, b) interval information, c) total ignorance

Such a distribution can be interpreted as a set of nested confidence intervals; it is called α-cuts, where α is the possibility degree. An α-cut will be define by: $[a_*, a^*] = \{a, \pi(a) \geq \alpha\}$
The figure B.3 shows an α-cut. By taking different values for α, we get different associated intervals. For α= 1, we obtain the kernel of the possibility distribution.

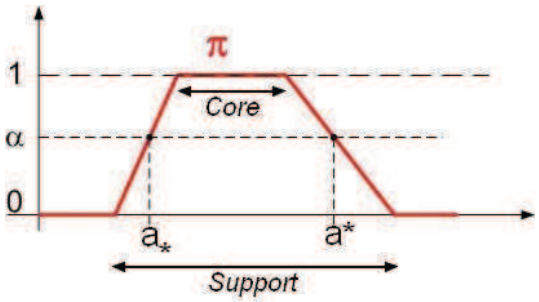


Figure B.3: α-cut

For an event A of U, two measures are defined:

- **Possibility measure:** $\Pi(A) = \sup_{a \in A} \{\pi(a)\}$
- **Necessity measure:** $\mathrm{N(A)} = \inf_{a \in \overline{A}} \{1 - \pi(a)\}$

Those measures express respectively the degree of plausibility and the degree of certainty of A.

Appendix C

Aralia format

This appendix illustrates the format of the fault trees that are taken in input of the algorithms descrided in chapters 5 and 7.

C.1 The syntax

The Aralia syntax is a way to write Boolean formulas. This syntax is, for $F_1, ..., F_n$ formulas, and $X_1, ..., X_n$ variables:

$-F_1$	Negation ($\neg$)
$F_1\|...\|F_n$	Disjunction ($\vee$)
$F_1\&...\&F_n$	Conjunction ($\wedge$)
$F_1 = ... = F_n$	Equivalence ($\Leftrightarrow$)
$F_1\#...\#\|F_n$	Exclusive or ($\triangle$)
$F_1 => F_2$	Implication ($\Rightarrow$)
$F_1?F_2 : F_3$	If-Then-Else (ite)
$@(k, [F_1, ..., F_n])$	k out of $F_1, ..., F_n$

C.2 Example of a partial fault tree in Aralia format for the Rudder model

There are 21 minimal cut of order 3.

```
/* Order of products :
3 21
*/


g.1 := ('B.loss' & 'Bus_4PP.loss' & 'Hyd_Y.loss');
g.2 := ('B.loss' & 'Bus_4PP.loss' & 'Y.loss');
```

g.3 := ('B.loss' & 'G.loss' & 'Hyd_Y.loss');
g.4 := ('B.loss' & 'G.loss' & 'Y.loss');
g.5 := ('B.loss' & 'Hyd_G.loss' & 'Hyd_Y.loss');
g.6 := ('B.loss' & 'Hyd_G.loss' & 'Y.loss');
g.7 := ('BCM.hidden_failure' & 'Bus_2PP.loss' & 'Bus_4PP.loss');
g.8 := ('BCM.hidden_failure' & 'Bus_2PP.loss' & 'G.loss');
g.9 := ('BCM.hidden_failure' & 'Bus_2PP.loss' & 'Hyd_G.loss');
g.10 := ('Bus_2PP.loss' & 'Bus_4PP.loss' & 'Y.loss');
g.11 := ('Bus_2PP.loss' & 'G.loss' & 'Y.loss');
g.12 := ('Bus_2PP.loss' & 'Hyd_G.loss' & 'Y.loss');
g.13 := ('Bus_4PP.loss' & 'Hyd_B.loss' & 'Hyd_Y.loss');
g.14 := ('Bus_4PP.loss' & 'Hyd_B.loss' & 'Y.loss');
g.15 := ('Bus_4PP.loss' & 'P2.loss' & 'Y.loss');
g.16 := ('G.loss' & 'Hyd_B.loss' & 'Hyd_Y.loss');
g.17 := ('G.loss' & 'Hyd_B.loss' & 'Y.loss');
g.18 := ('G.loss' & 'P2.loss' & 'Y.loss');
g.19 := ('Hyd_B.loss' & 'Hyd_G.loss' & 'Hyd_Y.loss');
g.20 := ('Hyd_B.loss' & 'Hyd_G.loss' & 'Y.loss');
g.21 := ('Hyd_G.loss' & 'P2.loss' & 'Y.loss');
'Rudder.Output.false' := (g.1 | g.2 | g.3 | g.4 | g.5 | g.6 | g.7 | g.8 | g.9 | g.10 | g.11 | g.12 | g.13 | g.14 | g.15 | g.16 | g.17 | g.18 | g.19 | g.20 | g.21);

C.3 Example of a fault tree in Aralia format

P1.Status.hs := ((-B.loss & DTN3578) | (B.loss & DTN3617));
DTN3578 := ((-BCM.active_failure & DTN3503) | (BCM.active_failure & DTN3577));
DTN3503 := ((-BPS_B.active_failure & DTN3500) | (BPS_B.active_failure & DTN3502));
DTN3500 := ((-BPS_Y.active_failure & DTN3483) | (BPS_Y.active_failure & DTN3499));
DTN3483 := ((-Bus_2PP.loss & DTN3482) | (Bus_2PP.loss & DTN3480));
DTN3482 := ((-Hyd_B.loss & DTN3478) | (Hyd_B.loss & DTN3481));
DTN3478 := ((-Hyd_Y.loss & DTN3475) | (Hyd_Y.loss & DTN3477));
DTN3475 := (P1.loss & DTN3474);
DTN3474 := ((-P2.loss & -S1.active_failure) | (P2.loss & DTN3473));
DTN3473 := ((-P3.loss & DTN3471) | (P3.loss & DTN3472));
DTN3471 := (-S1.active_failure | (S1.active_failure & DTN3470));
DTN3470 := (-S1.hidden_failure & Y.loss);
DTN3472 := (-S1.active_failure | (S1.active_failure & -S1.hidden_failure));
DTN3477 := (P1.loss & DTN3476);
DTN3476 := ((-P2.loss & -S1.active_failure) | (P2.loss & DTN3472));
DTN3481 := ((-Hyd_Y.loss & DTN3479) | (Hyd_Y.loss & DTN3480));
DTN3479 := (P1.loss & DTN3473);
DTN3480 := (P1.loss & DTN3472);
DTN3499 := (-BPS_Y.hidden_failure & DTN3498);
DTN3498 := ((-Bus_2PP.loss & DTN3494) | (Bus_2PP.loss & DTN3497));
DTN3494 := ((-Bus_4PP.loss & DTN3493) | (Bus_4PP.loss & DTN3482));
DTN3493 := ((-G.loss & DTN3492) | (G.loss & DTN3482));

DTN3492 := ((-Hyd_B.loss & DTN3487) | (Hyd_B.loss & DTN3491));
DTN3487 := ((-Hyd_G.loss & DTN3486) | (Hyd_G.loss & DTN3478));
DTN3486 := ((-Hyd_Y.loss & DTN3484) | (Hyd_Y.loss & DTN3485));
DTN3484 := (P1.loss & DTN3474);
DTN3485 := (P1.loss & DTN3476);
DTN3491 := ((-Hyd_G.loss & DTN3490) | (Hyd_G.loss & DTN3481));
DTN3490 := ((-Hyd_Y.loss & DTN3488) | (Hyd_Y.loss & DTN3489));
DTN3488 := (P1.loss & DTN3473);
DTN3489 := (P1.loss & DTN3472);
DTN3497 := ((-Bus_4PP.loss & DTN3496) | (Bus_4PP.loss & DTN3480));
DTN3496 := ((-G.loss & DTN3495) | (G.loss & DTN3480));
DTN3495 := ((-Hyd_G.loss & DTN3489) | (Hyd_G.loss & DTN3480));
DTN3502 := (-BPS_B.hidden_failure & DTN3501);
DTN3501 := ((-BPS_Y.active_failure & DTN3498) | (BPS_Y.active_failure & DTN3499));
DTN3577 := (-BCM.hidden_failure & DTN3576);
DTN3576 := ((-BPS_B.active_failure & DTN3574) | (BPS_B.active_failure & DTN3575));
DTN3574 := ((-BPS_B.hidden_failure & DTN3549) | (BPS_B.hidden_failure & DTN3573));
DTN3549 := ((-BPS_Y.active_failure & DTN3547) | (BPS_Y.active_failure & DTN3548));
DTN3547 := ((-BPS_Y.hidden_failure & DTN3538) | (BPS_Y.hidden_failure & DTN3546));
DTN3538 := ((-Bus_2PP.loss & DTN3533) | (Bus_2PP.loss & DTN3537));
DTN3533 := ((-Bus_4PP.loss & DTN3532) | (Bus_4PP.loss & DTN3531));
DTN3532 := ((-G.loss & DTN3530) | (G.loss & DTN3531));
DTN3530 := ((-Hyd_B.loss & DTN3520) | (Hyd_B.loss & DTN3529));
DTN3520 := ((-Hyd_G.loss & DTN3512) | (Hyd_G.loss & DTN3519));
DTN3512 := ((-Hyd_Y.loss & DTN3509) | (Hyd_Y.loss & DTN3511));
DTN3509 := (P1.loss & DTN3508);
DTN3508 := (P2.loss & DTN3507);
DTN3507 := ((-P3.loss & DTN3505) | (P3.loss & DTN3506));
DTN3505 := ((-S1.active_failure & DTN3504) | (S1.active_failure & DTN3470));
DTN3504 := (S1.hidden_failure & Y.loss);
DTN3506 := ((-S1.active_failure & S1.hidden_failure) | (S1.active_failure & -S1.hidden_failure));
DTN3511 := (P1.loss & DTN3510);
DTN3510 := (P2.loss & DTN3506);
DTN3519 := ((-Hyd_Y.loss & DTN3516) | (Hyd_Y.loss & DTN3518));
DTN3516 := (P1.loss & DTN3515);
DTN3515 := (P2.loss & DTN3514);
DTN3514 := ((-P3.loss & DTN3513) | (P3.loss & DTN3472));
DTN3513 := ((-S1.active_failure & Y.loss) | (S1.active_failure & DTN3470));
DTN3518 := (P1.loss & DTN3517);
DTN3517 := (P2.loss & DTN3472);
DTN3529 := ((-Hyd_G.loss & DTN3524) | (Hyd_G.loss & DTN3528));
DTN3524 := ((-Hyd_Y.loss & DTN3521) | (Hyd_Y.loss & DTN3523));
DTN3521 := (P1.loss & DTN3507);
DTN3523 := (P1.loss & DTN3522);
DTN3522 := ((-P2.loss & DTN3507) | (P2.loss & DTN3506));
DTN3528 := ((-Hyd_Y.loss & DTN3525) | (Hyd_Y.loss & DTN3527));
DTN3525 := (P1.loss & DTN3514);
DTN3527 := (P1.loss & DTN3526);
DTN3526 := ((-P2.loss & DTN3514) | (P2.loss & DTN3472));

DTN3531 := ((-Hyd_B.loss & DTN3519) | (Hyd_B.loss & DTN3528));
DTN3537 := ((-Bus_4PP.loss & DTN3536) | (Bus_4PP.loss & DTN3480));
DTN3536 := ((-G.loss & DTN3535) | (G.loss & DTN3480));
DTN3535 := ((-Hyd_G.loss & DTN3534) | (Hyd_G.loss & DTN3480));
DTN3534 := (P1.loss & DTN3506);
DTN3546 := ((-Bus_2PP.loss & DTN3545) | (Bus_2PP.loss & DTN3537));
DTN3545 := ((-Bus_4PP.loss & DTN3544) | (Bus_4PP.loss & DTN3543));
DTN3544 := ((-G.loss & DTN3542) | (G.loss & DTN3543));
DTN3542 := ((-Hyd_B.loss & DTN3520) | (Hyd_B.loss & DTN3541));
DTN3541 := ((-Hyd_G.loss & DTN3539) | (Hyd_G.loss & DTN3540));
DTN3539 := ((-Hyd_Y.loss & DTN3509) | (Hyd_Y.loss & DTN3523));
DTN3540 := ((-Hyd_Y.loss & DTN3516) | (Hyd_Y.loss & DTN3527));
DTN3543 := ((-Hyd_B.loss & DTN3519) | (Hyd_B.loss & DTN3540));
DTN3548 := (-BPS_Y.hidden_failure & DTN3538);
DTN3573 := ((-BPS_Y.active_failure & DTN3571) | (BPS_Y.active_failure & DTN3572));
DTN3571 := ((-BPS_Y.hidden_failure & DTN3555) | (BPS_Y.hidden_failure & DTN3570));
DTN3555 := ((-Bus_2PP.loss & DTN3554) | (Bus_2PP.loss & DTN3537));
DTN3554 := ((-Bus_4PP.loss & DTN3553) | (Bus_4PP.loss & DTN3552));
DTN3553 := ((-G.loss & DTN3551) | (G.loss & DTN3552));
DTN3551 := ((-Hyd_B.loss & DTN3550) | (Hyd_B.loss & DTN3529));
DTN3550 := ((-Hyd_G.loss & DTN3509) | (Hyd_G.loss & DTN3516));
DTN3552 := ((-Hyd_B.loss & DTN3516) | (Hyd_B.loss & DTN3528));
DTN3570 := ((-Bus_2PP.loss & DTN3562) | (Bus_2PP.loss & DTN3569));
DTN3562 := ((-Bus_4PP.loss & DTN3561) | (Bus_4PP.loss & DTN3560));
DTN3561 := ((-G.loss & DTN3559) | (G.loss & DTN3560));
DTN3559 := ((-Hyd_B.loss & DTN3558) | (Hyd_B.loss & DTN3541));
DTN3558 := ((-Hyd_G.loss & DTN3556) | (Hyd_G.loss & DTN3557));
DTN3556 := (Hyd_Y.loss & DTN3509);
DTN3557 := (Hyd_Y.loss & DTN3516);
DTN3560 := ((-Hyd_B.loss & DTN3557) | (Hyd_B.loss & DTN3540));
DTN3569 := ((-Bus_4PP.loss & DTN3568) | (Bus_4PP.loss & DTN3567));
DTN3568 := ((-G.loss & DTN3566) | (G.loss & DTN3567));
DTN3566 := ((-Hyd_B.loss & DTN3565) | (Hyd_B.loss & DTN3535));
DTN3565 := ((-Hyd_G.loss & DTN3563) | (Hyd_G.loss & DTN3564));
DTN3563 := (Hyd_Y.loss & DTN3534);
DTN3564 := (Hyd_Y.loss & DTN3480);
DTN3567 := ((-Hyd_B.loss & DTN3564) | (Hyd_B.loss & DTN3480));
DTN3572 := (-BPS_Y.hidden_failure & DTN3555);
DTN3575 := (-BPS_B.hidden_failure & DTN3549);
DTN3617 := ((-BCM.active_failure & DTN3587) | (BCM.active_failure & DTN3616));
DTN3587 := ((-BPS_B.active_failure & DTN3584) | (BPS_B.active_failure & DTN3586));
DTN3584 := ((-BPS_Y.active_failure & DTN3579) | (BPS_Y.active_failure & DTN3583));
DTN3579 := ((-Bus_2PP.loss & DTN3481) | (Bus_2PP.loss & DTN3480));
DTN3583 := (-BPS_Y.hidden_failure & DTN3582);
DTN3582 := ((-Bus_2PP.loss & DTN3581) | (Bus_2PP.loss & DTN3497));
DTN3581 := ((-Bus_4PP.loss & DTN3580) | (Bus_4PP.loss & DTN3481));
DTN3580 := ((-G.loss & DTN3491) | (G.loss & DTN3481));
DTN3586 := (-BPS_B.hidden_failure & DTN3585);
DTN3585 := ((-BPS_Y.active_failure & DTN3582) | (BPS_Y.active_failure & DTN3583));

DTN3616 := (-BCM.hidden_failure & DTN3615);
DTN3615 := ((-BPS_B.active_failure & DTN3613) | (BPS_B.active_failure & DTN3614));
DTN3613 := ((-BPS_B.hidden_failure & DTN3595) | (BPS_B.hidden_failure & DTN3612));
DTN3595 := ((-BPS_Y.active_failure & DTN3593) | (BPS_Y.active_failure & DTN3594));
DTN3593 := ((-Bus_2PP.loss & DTN3592) | (Bus_2PP.loss & DTN3537));
DTN3592 := ((-Bus_4PP.loss & DTN3591) | (Bus_4PP.loss & DTN3589));
DTN3591 := ((-G.loss & DTN3590) | (G.loss & DTN3589));
DTN3590 := ((-Hyd_G.loss & DTN3588) | (Hyd_G.loss & DTN3589));
DTN3588 := ((-Hyd_Y.loss & DTN3521) | (Hyd_Y.loss & DTN3534));
DTN3589 := ((-Hyd_Y.loss & DTN3525) | (Hyd_Y.loss & DTN3480));
DTN3594 := (-BPS_Y.hidden_failure & DTN3593);
DTN3612 := ((-BPS_Y.active_failure & DTN3610) | (BPS_Y.active_failure & DTN3611));
DTN3610 := ((-BPS_Y.hidden_failure & DTN3601) | (BPS_Y.hidden_failure & DTN3609));
DTN3601 := ((-Bus_2PP.loss & DTN3600) | (Bus_2PP.loss & DTN3537));
DTN3600 := ((-Bus_4PP.loss & DTN3599) | (Bus_4PP.loss & DTN3598));
DTN3599 := ((-G.loss & DTN3597) | (G.loss & DTN3598));
DTN3597 := ((-Hyd_B.loss & DTN3596) | (Hyd_B.loss & DTN3590));
DTN3596 := ((-Hyd_G.loss & DTN3521) | (Hyd_G.loss & DTN3525));
DTN3598 := ((-Hyd_B.loss & DTN3525) | (Hyd_B.loss & DTN3589));
DTN3609 := ((-Bus_2PP.loss & DTN3608) | (Bus_2PP.loss & DTN3569));
DTN3608 := ((-Bus_4PP.loss & DTN3607) | (Bus_4PP.loss & DTN3606));
DTN3607 := ((-G.loss & DTN3605) | (G.loss & DTN3606));
DTN3605 := ((-Hyd_B.loss & DTN3604) | (Hyd_B.loss & DTN3590));
DTN3604 := ((-Hyd_G.loss & DTN3602) | (Hyd_G.loss & DTN3603));
DTN3602 := (Hyd_Y.loss & DTN3521);
DTN3603 := (Hyd_Y.loss & DTN3525);
DTN3606 := ((-Hyd_B.loss & DTN3603) | (Hyd_B.loss & DTN3589));
DTN3611 := (-BPS_Y.hidden_failure & DTN3601);
DTN3614 := (-BPS_B.hidden_failure & DTN3595);

Appendix D

Implementation

D.1 CUDD

The CUDD package provides functions to manipulate Binary Decision Diagrams (BDDs), Algebraic Decision Diagrams (ADDs), and Zero-suppressed Binary Decision Diagrams (ZDDs).

The CUDD package can be used in three ways:

- As a black box. In this case, the application program that needs to manipulate decision diagrams only uses the exported functions of the package. The rich set of functions included in the CUDD package allows many applications to be written in this way. An application written in terms of the exported functions of the package needs not concern itself with the details of variable reordering, which may take place behind the scenes.
- As a clear box. When writing a sophisticated application based on decision diagrams, efficiency often dictates that some functions be implemented as direct recursive manipulation of the diagrams, instead of being written in terms of existing primitive functions.
- Through an interface. Object-oriented languages like C++ and Perl5 can free the programmer from the burden of memory management. A C++ interface is included in the distribution of CUDD. It automatically frees decision diagrams that are no longer used by the application and overloads operators. Almost all the functionality provided by the CUDD exported functions is available through the C++ interface, which is especially recommended for fast prototyping.

(cite http://vlsi.colorado.edu/ fabio/CUDD/)

D.2 BuDDy

BuDDy is a Binary Decision Diagram library, with :many highly efficient vectorized BDD operations,dynamic variable reordering,automated garbage collection, a C++ interface with automatic

reference counting. Although it does not have the GNU public license, the code is open for modifications and is in public domain. BuDDy is implemented in C but has a wrapping C++ interface.

BuDDy has an internal garbage collection mechanism optimizing its memory usage, totally transparent, if desired, to use the C++ interface. It also has a cache that saves the last results of operations. Almost everything performance related is configurable through the interface, and the library begins runs with default values, so you can start using it without being familiar with all the possible configurations.

BuDDy has a rich set of operations, which can help you implement very complex operations with only a few code lines. Among the operations in the interface are the abilities to define the variables (inputs) of the expressions, refer a large vector of binary variables as a vector of encoded integers, and create a vector of expressions.

D.3 Boost C++ library

Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use. Boost works on almost any modern operating system, including UNIX and Windows variants.

The libraries are aimed at a wide range of C++ users and application domains. They range from general-purpose libraries like the smart pointer library, to operating system abstractions like Boost FileSystem, to libraries primarily aimed at other library developers and advanced C++ users, like the template metaprogramming (MPL) and domain-specific language (DSL) creation (Proto).

In order to ensure efficiency and flexibility, Boost makes extensive use of templates. Boost has been a source of extensive work and research into generic programming and metaprogramming in C++.

Use of high-quality libraries like Boost speeds initial development, results in fewer bugs, reduces reinvention-of-the-wheel, and cuts long-term maintenance costs. And since Boost libraries tend to become de facto or de jure standards, many programmers are already familiar with them. Ten of the Boost libraries are included in the C++ Standard Library's TR1, and so are slated for later full standardization. More Boost libraries are in the pipeline for TR2. Using Boost libraries gives an organization a head-start in adopting new technologies.

(cite http://www.boost.org/)

D.4 wxWidget

wxWidgets (formerly wxWindows) is a widget toolkit and tools library for creating graphical user interfaces (GUIs) for cross-platform applications. wxWidgets enables a program's GUI code to compile and run on several computer platforms with minimal or no code changes. It covers systems such as Microsoft Windows, OS X (Carbon and Cocoa), iOS (Cocoa Touch), Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2 and AmigaOS. A version for embedded systems is under development.

wxWidgets is used across many industry sectors, most notably by Xerox, Advanced Micro Devices (AMD), Lockheed Martin, NASA and the Center for Naval Analyses. It is also used in the public sector and education by, for example, Dartmouth Medical School, National Human Genome Research Institute, National Center for Biotechnology Information, and many others. wxWidgets is used in many open source projects, and by individual developers. A wide choice of compilers and other tools to use with wxWidgets, allows development of highly sophisticated applications on a tight budget.

It is free and open source software, distributed under the terms of the wxWidgets License, which satisfies those who wish to produce for GPL and proprietary software.

D.5 RPN format

Reverse Polish notation (RPN) is a mathematical notation where we put operands first and the operator after the operands. RPN is also known as Postfix Notation. The basic feature of RPN is that it does not require parenthesis to write mathematical formula or expression. Word ”Polish” refers to the nationality of logician Jan Lukasiewicz, who invented (prefix) Polish notation in the 1920s. RPN was developed to reduce memory access in computer implementation of mathematical formula and expressions. It also helps in using computer stack memory to evaluate expressions.

During the thesis, while developing a software to parse Aralia file and to build a BDD from the file, it was very complex and tedious to parse parenthesis of final expression of the boolean formula of the Aralia file. To overcome this issue, RPN notation is used in the Aralia parser. First the Aralia file is read and parsed to construct the final boolean formula from given variable expressions. Then this final boolean formula is converted to RPN notation. Finally, a BDD is constructed from this RPN notation of the final boolean formula.

The working principal of this parser is explained in the following section; first the conversion to RPN is explained with help of an algorithm which not only takes into account the precedence among operators but also the parentheses. After, the process of creation of BDD from this RPN expression is explained.

The algorithm of conversion to RPN format is shown in 5. This algorithm is explained with a simple example as shown in Figure D.1. The formula in normal mathematical expression with parenthesis is read and is conversed to RPN notation following the steps of the algorithm.

For example, consider the formula $A + (B - C)/D$. According to the algorithm, we start with an empty RPN string, an empty operators stack and the normal mathematical expression string as shown in step 1 of Figure D.1. We then tokenize the normal mathematical expression string. We read the tokens of the string in a while loop. If the token is a variable, in this case ’A’, we add it to the RPN string as show in step 2 of Figure D.1.

If the token is an operator, in this case ’+’, and the stack is empty, or the top value of stack is starting parenthesis ’(’ or the precedence of the current operator is greater than the top operator on the stack, then we store this operator on the stack as show in step 3 of Figure D.1 (because in this case stack was empty). If the token is starting parenthesis ’(’, we add it to the stack as show in step 4 of Figure D.1.

If the token is closing parenthesis ’)’, as show in step 8 of Figure D.1, we remove items from operator stack till the starting parenthesis ’(’ and add the operators removed from stack to RPN string. We then discard ’(’ from stack and ’)’ from expression string. When there are no more tokens left, we add all the operators in the stack to the end of RPN string as show in step 10 of Figure D.1.

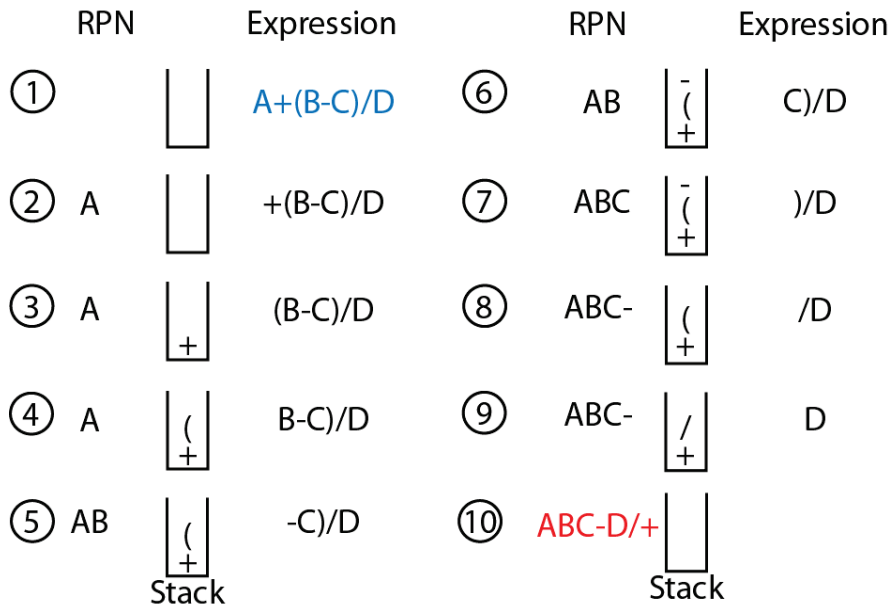


Figure D.1: Example of RPN algorithm for the formula $A + (B - C)/D$

Algorithm 5 Conversion to RPN

```
Parse Aralia file and get final boolean formula expression with parentheses and precedences.
Tokenize the boolean formula expression
Initialize Operator stack and RPN string
while there are tokens remaining in the expression do
    T = next token from expression
    if T == '(' then
        push T onto the stack
    else if T is a variable or numeric literal then
        add T to the RPN output string
    else if T is '&', '|' or '!' then
        if (operator stack is empty) or (the value at the top of the stack is '(') or (precedence(operator at top of stack) < precedence(T))
        then
            push T onto the operator stack
        else
            add the operator at the top of the stack to the RPN string
            pop the stack
            while (the stack is not empty) and (the top of the stack is not '(') and (precedence(T) < precedence(operator at top of stack))
            do
                push T onto the stack
            end while
        end if
    else if T is ')' then
        while top of operator stack is not '(' do
            add the operator at the top of the stack to the RPN string
            pop the stack
        end while
    else
        report error
    end if
    while operator stack is not empty do
        add the operator at the top of the stack to the RPN string
        pop the stack;
    end while
    return  RPN string

end while
```

D.6 Aralia file Parser

The algorithm in Figure 6 explains the Aralia parser functionality. Aralia file has a format where last line is the Formula and lines before that are variables of the formula with their boolean expressions. The format is simple where variable name is followed by ":=" symbol and then the boolean expression for the variable is given. The end of line is indicated by ';'. The software opens the Aralia file from disk and reads the whole file and counts the total number of lines. Then it reads the last line which is Formula and split it into two strings: top event name on left side of ":=" and boolean equation on right side. Finally, each line of file is read, and the variables in boolean equation of Formula are replaced with expressions read from each line.

Algorithm 6 Aralia file parser

```
Open Aralia file for reading and count total number of lines.
Split last line of Aralia file (Formula) into two parts.
store top event name in "expression" string and formula in "variable" string
for each line in Aralia file do
    read variable name from the line
    search "variable" string for this variable name
    replace variable name in "variable" string with expression on right hand side of the read line
end for
return "variable" string as final boolean formula
```

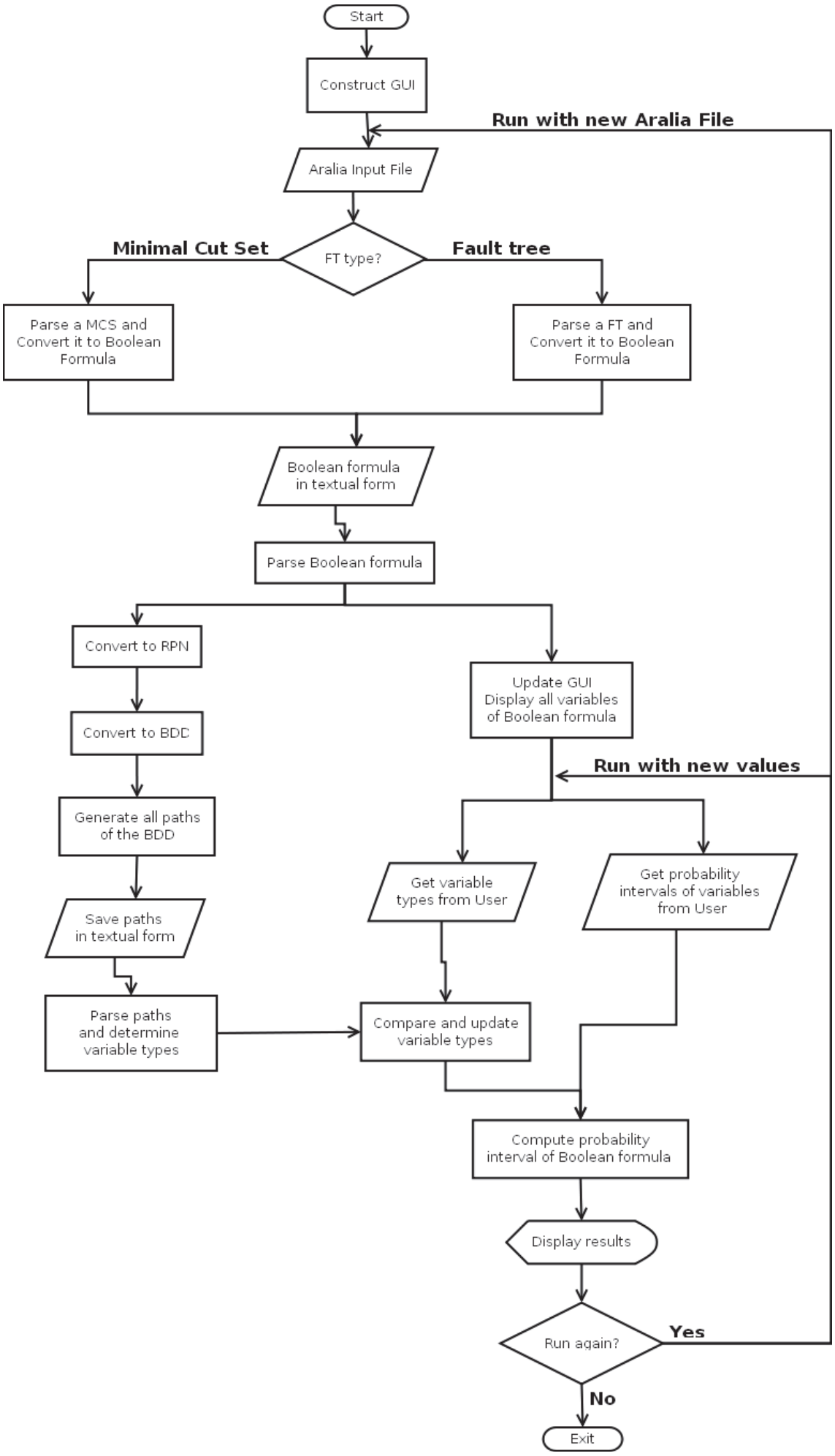


Figure D.2: Simplified Flowchart of the software

Appendix E

BDDs with different orders

This appendex illustrates the BDDs that are used in Chapter 8. They are the files that the algorithm has to parse in order to compute the range of the probability of the Boolean formula.

E.1 Paths of the BDD of P1.hs

A BDD can be represented by a set Pa of paths with the following format:

$$< V_1^{path_1} : value, \ldots, V_{n1}^{path_1} : value > \cdots < v_1^{path_m} : value, \ldots, V_{nm}^{path_m} : value > \tag{E.1}$$

where m is the number of paths and $n_i, i = 1, \ldots, m$ is the number of literals (positive or negative) in a path i.

Figure E.1: BDD given by CUDD for $P1\_hs$

Figure E.2: Best BDD given for $P1\_hs$ and order

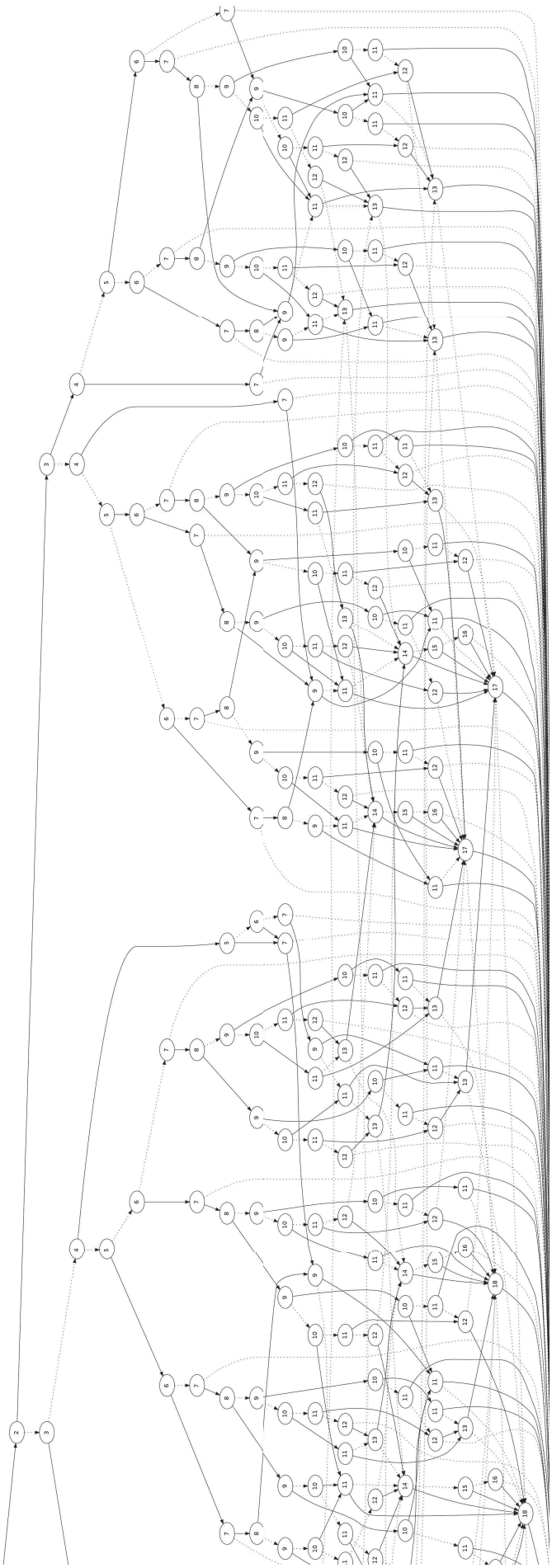


Figure E.3: Best BDD given for $P1\_hs$ and order (Part 1)

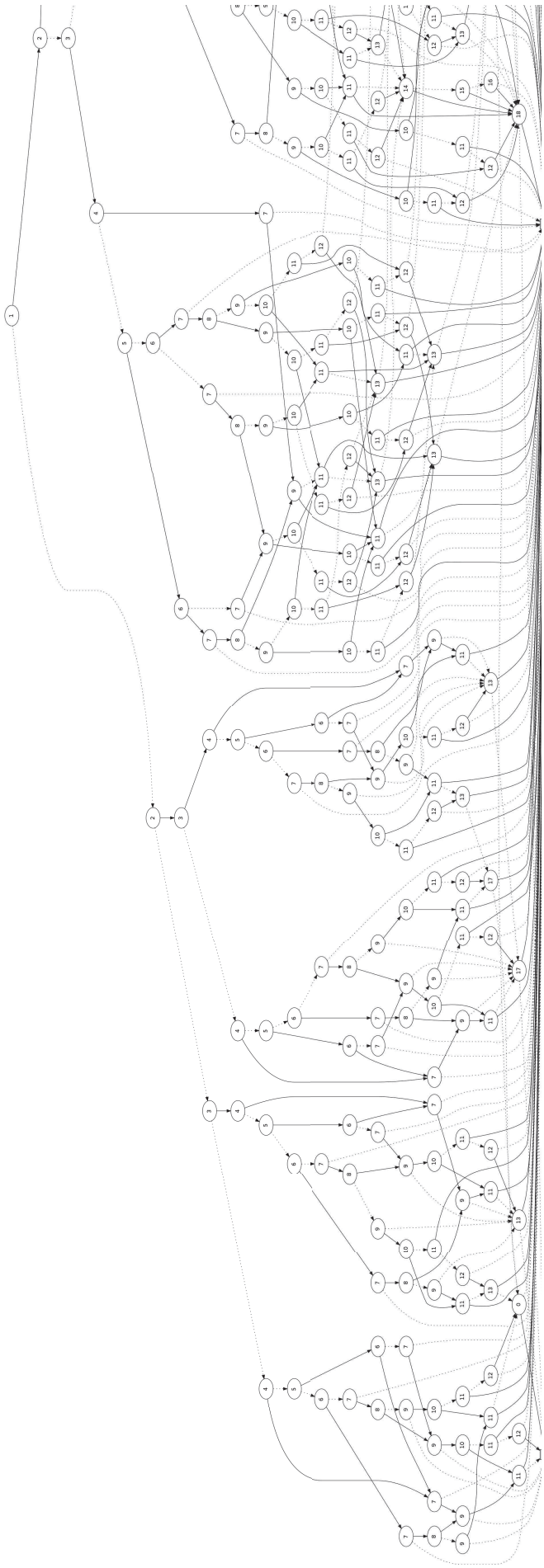


Figure E.4: Best BDD given for $P1\_hs$ and order (Part 2)

Index