



Évaluation de la fiabilité des systèmes modélisés par arbres de défaillances grâce aux techniques de satisfiabilité

Margaux Duroeulx

► To cite this version:

Margaux Duroeulx. Évaluation de la fiabilité des systèmes modélisés par arbres de défaillances grâce aux techniques de satisfiabilité. Performance et fiabilité [cs.PF]. Université de Lorraine, 2020. Français. NNT : 2020LORR0026 . tel-02881242

HAL Id: tel-02881242

<https://hal.univ-lorraine.fr/tel-02881242>

Submitted on 25 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4

Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

http://www.cfcopies.com/V2/leg/leg_droi.php

<http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm>

Évaluation de la fiabilité des systèmes modélisés par arbres de défaillances grâce aux techniques de satisfiabilité

THÈSE

Présentée et soutenue publiquement le 5 mars 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

Mention Informatique

par

Margaux DUROEULX

Composition du jury :

Président :

Jean-Marc FAURE Professeur, LURPA, Université Paris-Sud

Rapporteurs :

Daniel LE BERRE Professeur, CRIL, Université d'Artois

Zineb SIMEU-ABAZI Professeure, G-SCOP, Université de Grenoble

Examineurs :

Sylvain CONCHON Professeur, LRI, Université Paris-Sud

Marie DUFLLOT Maîtresse de conférence, LORIA, Université de Lorraine

Marine MINIER Professeure, LORIA, Université de Lorraine

Directeurs de thèse :

Stephan MERZ Directeur de recherche, LORIA, Université de Lorraine

Nicolae BRÎNZEI Maître de conférence, CRAN, Université de Lorraine

Remerciements

La particularité de cette thèse réside dans le fait qu'elle a été réalisée à la fois aux laboratoires du Loria et du Cran. Deux bureaux différents, sur deux campus différents. Ce fut souvent une force, me permettant de m'entourer de nombreuses personnes pour m'accompagner durant ces trois années. Ce fut parfois une faiblesse, lorsqu'il fallut courir d'un bâtiment à l'autre dans la même journée. Au-delà de ce détail géographique, ces trois années et demi furent extrêmement enrichissantes scientifiquement et humainement et je quitte Nancy avec un petit peu de Lorraine en moi.

Je tiens tout d'abord à remercier les membres de mon jury de thèse d'avoir accepté d'examiner le fruit de mes travaux de recherche. Je remercie tout particulièrement Zineb Simeu-Abazi et Daniel Le Berre d'avoir accepté de rapporter mon travail et de m'avoir fait des retours très pertinents. Je remercie également Marine Minier de m'avoir accompagnée avec beaucoup d'intérêt tout au long de ma thèse. Enfin, je tiens à remercier Jean-Marc Faure de m'avoir fait l'honneur de présider mon jury de thèse, après m'avoir accompagnée à Supméca et à l'ENS Cachan.

Je souhaite remercier Stephan Merz et Nicolae Brinzei d'avoir permis à cette thèse de se réaliser, de m'avoir fait confiance et d'avoir toujours cru en moi. J'ai énormément appris à vos côtés et cette thèse n'aurait pas été possible sans vous. Stephan, merci pour ta patience et ta disponibilité tout au long de la thèse. Nicolae, merci de m'avoir toujours encouragée à me surpasser.

Cette thèse fut un travail d'équipe et le quatrième membre de cette équipe fut un pilier essentiel à mon équilibre. Marie, je te remercie d'avoir été si présente dans ma thèse et dans ma vie. Tu as eu un rôle capital et je t'en suis infiniment reconnaissante.

Je remercie Coralie Fritsch d'avoir pris le temps de répondre à mes questions, de m'avoir conseillée et d'avoir fait et refait mille calculs avec moi.

Merci à Véronique Poirel, Olivia Brener, Annabelle Arena, Anne Chrétien et Christine Pierson vers qui j'ai toujours pu me tourner pour trouver une oreille attentive et bienveillante face aux embûches de l'administration française et aux aléas de la thèse.

Je tiens à remercier Martin Riener et Simon Cruanes d'avoir insufflé tant de joie dans ma première année de thèse. *Thank you Martin for the helper gnomes, dancing and cheering me up every hard night of work!*

Pierre, Hans, Daniel, Antoine, j'ai adoré partager ce bureau avec vous. Alexis, Imane, Hamid, Igor, Etienne, Yann, Nicolas, merci pour tous ces bons moments passés ensemble. Quentin, merci pour toutes ces pauses cafés qui furent de vraies échappatoires. Athénaïs, Justine, Lin, merci pour votre joie de vivre et votre soutien.

Prisca, merci de vouloir changer le monde avec moi. Clarice, je te remercie pour nos joyeux déjeuners et pour ton accueil à chacun de mes assauts à ton bureau. Merci à mon co-bureau Matthieu. Merci à mon accolyte d'enseignement Pierre. Merci à Clémence et à Sara pour nos soirées ensemble. Merci à Jérémie d'avoir découvert Nancy avec moi.

Merci à Benjamin de m'avoir fait découvrir Metz.

Merci à Clément et Mathieu de l'OSUL, j'ai aimé jouer à vos côtés.

Merci à Lucie, Amélie et Isaline!

J'ai la chance de pouvoir compter sur des amies chères et fidèles. Merci à Audrey, Marie, Marine, Florence, Lamia, Marine, Anaïs, Marion et Fanny d'être là pour moi.

Je ne serais pas là sans ma famille. Merci à mes parents d'avoir cru en moi, de m'avoir soutenue et de m'avoir accompagnée dans mes études, mes déménagements et à chaque étape de mon éducation dont cette thèse est l'aboutissement. Je remercie mes grands-parents pour tout leur amour. Merci à mes cousins de faire partie de ma vie.

Je remercie Sylvie, Bruno, Aurélie et Cédric de m'avoir accueillie à bras ouverts dans leur famille.

Enfin, je souhaite tout particulièrement remercier la femme que j'aime pour son soutien et son amour qui m'ont portée ces deux dernières années. Laëtitia, les moments de doute furent nombreux mais tu as toujours été là pour sécher mes larmes et éclairer ma réflexion sous un nouveau jour.

PS : Si une coquille persistait dans ce manuscrit, seul Tsuku en serait tenu responsable. Merci à petit chat de m'avoir offert des câlins nocturnes lorsque je rédigeais et de m'avoir appris à verrouiller mon clavier plus vite que mon ombre!

"Que la force me soit donnée de supporter ce qui ne peut être changé et le courage de changer ce qui peut l'être, mais aussi la sagesse de distinguer l'un de l'autre."

Marc Aurèle

Table des matières

Glossaire	xv
Abbréviations	xvii
Introduction générale	1
1 État de l'art	5
1.1 Introduction	5
1.2 Sûreté de fonctionnement	6
1.2.1 Indicateurs RAMS	6
1.2.2 Analyses qualitatives et quantitatives	8
1.3 Modélisation des systèmes	9
1.3.1 Diagramme de fiabilité	10
1.3.2 Arbres de défaillances	10
1.3.2.1 Arbres de défaillances statiques	11
1.3.2.2 Arbres de défaillances dynamiques	12
Porte FDEP	13
Porte PAND	13
Porte SPARE	14
Porte SEQ	14
1.3.2.3 Arbres de défaillances flous	14
1.3.2.4 Arbres de défaillances réparables	15
1.3.2.5 (G)BDMP	15
1.3.3 Chaînes de Markov	16
1.3.4 Réseaux de Petri stochastiques	16
1.4 Étude de formules propositionnelles	16
1.4.1 Coupes, liens et implicants premiers	16
1.4.2 Formules propositionnelles	17
1.4.3 Satisfiabilité	18
1.4.3.1 SAT	18
1.4.3.2 SMT	20

1.5	Évaluation probabiliste	21
1.5.1	Fiabilité des composants	21
1.5.2	Fiabilité du système	23
1.5.2.1	Approches analytiques	23
	Principe d'inclusion-exclusion	23
	Model-checking de systèmes probabilistes : méthodes numé- riques	24
	Diagrammes de décision binaire	24
	Diagramme de Hasse	26
	Analyse de fiabilité par calcul différentiel	27
1.5.2.2	Approches par simulation (simulation de Monte Carlo) . .	28
	Arbres de défaillances dynamiques	28
	Réseaux de Petri stochastiques	29
	Model-checking de systèmes probabilistes : méthodes statis- tiques	29
1.6	Conclusion	30
2	Étude des systèmes modélisés par arbres de défaillances statiques grâce aux techniques de satisfiabilité	33
2.1	Introduction	33
2.2	Définitions	34
2.3	Fonction de structure	37
2.3.1	Expression booléenne des portes	37
	Porte ET	37
	Porte OU	37
	Porte KooN	38
2.3.2	Dualisation de la fonction de structure	39
2.4	Diagramme de Hasse d'un système statique	40
2.5	Calcul des liens minimaux	42
2.5.1	Calcul des liens minimaux à partir d'un format DNF	42
2.5.2	Calcul des liens minimaux à partir d'un format CNF	43
	Preuve	44
2.5.3	Calcul des liens minimaux à partir des coupes minimales	46
	Preuve	47
2.6	Calcul de la fiabilité du système	48
2.6.1	Probabilité d'un monôme	49
2.6.2	Attribution des poids dans un diagramme de Hasse	49
2.6.3	Obtention du polynôme de fiabilité	50
2.7	Application	52

2.7.1	Application au système de secours d'un escalier mécanique	52
2.7.2	Application à une bibliothèque de systèmes industriels	55
2.7.2.1	Bibliothèque	56
2.7.2.2	Comparaison des résultats	57
2.7.2.3	Résumé	58
2.8	Conclusion	59

3	Étude des systèmes modélisés par arbres de défaillances dynamiques grâce aux techniques de satisfiabilité	61
3.1	Introduction	61
3.2	Définitions	63
3.3	Fonction de structure d'un système dynamique	66
3.3.1	Portes statiques	66
3.3.2	Portes dynamiques	67
3.3.2.1	Porte FDEP	67
3.3.2.2	Porte PAND	68
3.3.2.3	Porte SPARE	69
	Porte SPARE sans composant partagé	70
	Portes SPARE avec un composant redondant partagé	70
	Portes SPARE avec entrées partagée et non partagée	71
	Généralisation des portes Spare avec composants partagés	72
	Redondance froide	76
3.3.3	Interactions entre les portes dynamiques	76
3.3.4	Obtention d'une fonction de structure booléenne	77
3.3.5	Gestion des équations conditionnelles	77
3.3.6	Suppression des opérateurs minimum et maximum	78
3.3.7	Suppression des variables non booléennes	79
3.3.8	Dualisation de la fonction de structure	80
3.4	Calcul des séquences de lien minimales	82
3.4.1	Définition des séquences de lien	82
3.4.2	Calcul des séquences de lien minimales	83
3.5	Calcul de la fiabilité	86
3.5.1	Diagramme de Hasse d'un système dynamique	87
	Probabilité et redondance froide	88
	Diagramme de Hasse et porte FDEP	89
	Construction du diagramme de Hasse	90
3.5.2	Attribution d'un poids à chaque noeud	91
3.5.3	Obtention du polynôme de fiabilité	92
3.5.4	Calcul de probabilité d'un monôme	95

3.5.4.1	Calcul de probabilité d'un monôme sans variable de temps	95
3.5.4.2	Calcul de probabilité d'un monôme avec une variable de temps	95
3.5.4.3	Calcul de probabilité d'un monôme avec plusieurs variables de temps	96
3.5.5	Calcul de probabilité des variables de temps	97
3.6	Application	98
3.6.1	Fonction de structure	99
	CPU	99
	Gâchette	99
	Moteurs	99
	Pompes	99
3.6.2	Analyse de la fonction de structure	101
3.6.3	Obtention du polynôme de fiabilité	102
3.6.3.1	Analyse de l'unité CPU	103
3.6.3.2	Analyse de l'unité des moteurs	104
3.6.3.3	Analyse de l'unité des pompes	106
3.7	Conclusion	110

Table des figures

1.1	Indicateurs de sûreté de fonctionnement et de sécurité-immunité.	7
1.2	Diagramme de fiabilité d'un système de 3 pompes en redondance 2-parmi-3.	10
1.3	Portes statiques.	11
1.4	Installation industrielle dans deux états différents.	12
1.5	Portes dynamiques.	13
1.6	Courbes de la fonction de répartition exponentielle.	21
1.7	Courbe en baignoire.	22
1.8	Courbes de la fonction de répartition de Weibull.	22
1.9	BDD réduit d'un système de trois pompes en redondance 2oo3.	25
1.10	BDD et BDD réduit d'un même système.	26
1.11	Diagramme de Hasse d'un système de trois pompes en redondance 2oo3.	27
2.1	Arbre de défaillances statique.	34
2.2	Portes statiques.	38
2.3	Portes 2oo3.	41
2.4	Diagramme de Hasse.	41
2.5	Diagramme de Hasse.	41
2.6	Noeud G_A	49
2.8	Courbe d'évolution de la fiabilité du système de l'exemple 2.1 sur une année.	50
2.7	Diagramme de Hasse pondéré de l'exemple 2.1	51
2.9	Arbre de défaillances du système de secours d'un escalier mécanique.	52
2.10	Diagramme de Hasse restreint du système 2.9.	53
2.12	Évolution de la fiabilité du système au cours du temps.	55
2.13	Fichier cp <i>Test</i>	56
2.14	Arbre de défaillances de <i>Test</i>	56
3.1	Exemple d'arbre de défaillances dynamique.	62
3.2	Porte PAND à deux entrées.	65
3.3	Portes statiques.	66
3.4	Porte FDEP.	67
3.5	Porte FDEP généralisée.	67

3.6	Chronogrammes de la porte FDEP de la Figure 3.4.	68
3.7	Porte PAND.	68
3.8	Chronogrammes de la porte PAND.	69
3.9	Porte SPARE sp_0	70
3.10	Portes SPARE sp_1 et sp_2	70
3.11	Chronogrammes de la porte SPARE de la Figure 3.10.	71
3.12	Portes SPARE avec un composant partagé et un composant non partagé. .	71
3.13	Chronogrammes de la porte SPARE de la Figure 3.12.	72
3.14	Arbre de défaillances.	73
3.15	Graphe de dépendance.	73
3.16	Arbre de défaillances.	74
3.17	Graphe de dépendance.	74
3.18	Portes FDEP et PAND.	77
3.19	Portes FDEP et SPARE.	77
3.20	Portes PAND et SPARE avec un composant partagé et un non partagé. . .	78
3.21	Principe de l'algorithme d'obtention des MTSSs.	85
3.22	Porte PAND.	87
3.23	Diagramme de Hasse.	87
3.24	Diagramme de Hasse d'un système à trois composants et une variable tem- porelle considérée $t_{1,2}$	88
3.25	Diagramme de Hasse d'un système à trois composants et deux variables de temps considérées $t_{1,2}$ et $t_{1,3}$	88
3.26	DFT.	89
3.27	Diagramme de Hasse.	89
3.28	Arbre de défaillances dynamique.	89
3.29	Diagramme de Hasse.	90
3.30	Construction du diagramme de Hasse de l'exemple 3.41.	90
3.31	Diagramme de Hasse d'un système à trois composants et deux variables de temps considérées $t_{1,2}$ et $t_{1,3}$	91
3.32	Porte PAND et portes SPARE partageant un composant.	92
3.33	Arbre de défaillances.	92
3.34	Diagramme de Hasse.	92
3.35	Diagramme de Hasse du système de la Figure 3.32.	93
3.36	Diagramme de Hasse après la première suppression.	94
3.37	Diagramme de Hasse après la deuxième suppression.	94
3.38	MTSS avec contrainte de temps.	96
3.39	Arbre de défaillances dynamique du système CAS.	98
3.40	Unité CPU du système CAS.	103
3.41	Courbe de fiabilité de l'unité des processeurs du système CAS.	104

3.42	Unité moteurs du système CAS.	105
3.43	Diagramme de Hasse de l'unité moteurs.	105
3.44	Courbe de fiabilité de l'unité des moteurs du système CAS.	106
3.45	Unité des pompes du système CAS.	106
3.46	Diagramme de Hasse de l'unité constituée des pompes du système CAS. . .	107
3.47	Courbe de fiabilité de l'unité des pompes du système CAS.	109
3.48	Courbe de fiabilité du système CAS.	110
3.49	Conversion des tokens vers smt2s	117
3.50	Arbre de défaillance d'un système <i>Test</i>	118
3.51	Fichier cp du système <i>Test</i>	118
3.52	Fichier smt2 du système <i>Test</i>	118
3.53	Calcul des coupes minimales du système <i>Test</i>	118
3.54	Coupes minimales du système <i>Test</i>	118

Glossaire

Ce manuscrit propose de faire converger les travaux sur l'évaluation probabiliste avec ceux sur les techniques symboliques de vérification. Par conséquent, il se situe à la frontière entre les domaines de la satisfiabilité et de la sûreté de fonctionnement. Afin d'éviter toute confusion sur le vocabulaire employé dans ce manuscrit, ce glossaire contient les éléments de langage qui pourraient être à l'origine d'une confusion.

L'arbre de défaillances est la modélisation la plus répandue pour les études de sûreté de fonctionnement dans le domaine industriel. Il y est souvent fait référence à un arbre de défaillance par le terme *modèle* puisqu'il permet de modéliser le comportement d'un système. Ce terme est également utilisé pour faire référence à une représentation d'un système par diagramme de fiabilité, chaîne de Markov ou tout autre représentation graphique d'un système. En informatique, un modèle est une instance générée par un solveur (SAT en l'occurrence) pour une formule donnée, qui rend la formule vraie ; c'est donc une *solution* au problème.

Puisque nous considérons des systèmes industriels, les événements de base sont principalement des composants. Par abus de langage, nous nommerons *composants* les événements de base du système dans ce manuscrit.

Un système ou un composant ne pouvant plus assurer sa fonction est dit *défaillant* ; mais on dira d'une porte ou d'une entrée qu'elle est *activée* ou *enclenchée*.

	Logique	Sûreté de fonctionnement
	Littéral	Variable
	Modèle	Solution
Arbre de défaillances		Modèle
	Clause	Maxterme
	Cube	Minterme
Formule logique modélisant le fonctionnement d'un système		Fonction de structure

Tableau 1 – Vocabulaire de la logique et de la sûreté de fonctionnement.

Abbreviations

BDD : diagrammes de décision binaire (Binary Decision Diagram)
BDMP : processus de Markov à logique booléenne
(Boolean logic Driven Markov Process)
CCF : défaillance de cause commune (Common Cause Failure)
Cold SPARE : porte SPARE à redondance froide
CS : cut (Cut Set)
CSS : séquence de coupe (Cut Set Sequence)
ER : évènement redouté
FT : arbre de défaillances (Fault Tree)
DFT : arbre de défaillances dynamique (Dynamic Fault Tree)
FDEP : dépendance fonctionnelle (Functional-DEPendency)
Hot SPARE : porte SPARE à redondance chaude
MC : chaînes de Markov (Markov Chain)
MCS : coupe minimale (Minimal Cut Set)
MCSS : séquence de coupe minimale (Minimal Cut Set Sequence)
MTS : lien minimal (Minimal Tie Set)
MTSS : séquence de lien minimale (Minimal Tie Set Sequence)
PAND : ET prioritaire (Priority-AND)
RBD : bloc-diagramme de fiabilité (Reliability Block Diagram)
SEQ : (Sequence Enforcing)
SDD : diagramme de décision séquentielle (Sequence Decision Diagram)
SFT : arbre de défaillances statique (Static Fault Tree)
TS : lien (Tie Set)
TSS : séquence de lien (Tie Set Sequence)
Warm SPARE : porte SPARE à redondance tiède

Introduction générale

La conception d'un système industriel a pour objectif de proposer une architecture physique et logicielle permettant de réaliser le service attendu par le cahier des charges. Le système est conçu afin qu'il soit apte à délivrer ce service à tout moment, c'est l'étude de la *disponibilité*. Toutefois, que ce système soit un moyen de transport (ferroviaire, aéronautique, aérospatial, etc.) ou une centrale de production d'énergie (nucléaire, thermique, hydraulique, etc.), il faut garantir qu'il est capable de fonctionner sans défaillir pendant un intervalle de temps donné, c'est l'étude de la *fiabilité*. Traduit de l'anglais *reliability* et entré au dictionnaire français en 1962, le mot fiabilité est construit à partir du mot *fiable* : "en qui on peut se fier". Un système fiable est alors un système auquel nous pouvons faire confiance. Si une défaillance du système peut avoir lieu, il est nécessaire d'étudier sa probabilité et sa criticité, afin de prévenir les défaillances dangereuses, c'est l'étude de la *sécurité*. Il faut ensuite définir la périodicité des maintenances, c'est-à-dire la durée pendant laquelle le système peut être utilisé sans nécessiter de maintenance ou de réparation, c'est l'étude de la *maintenabilité*. Des systèmes non réparables existent, comme les lanceurs de fusée aérospatiale. Lorsqu'une fusée est lancée dans l'espace, il est inconcevable de la réparer durant la phase de lancement et elle est supposée non réparable. L'étude de la fiabilité, disponibilité, maintenabilité et sécurité du système est appelée la *sûreté de fonctionnement* (SdF).

Un des principes de base de la sûreté de fonctionnement est la redondance. Cela consiste à prévoir plusieurs entités¹ permettant d'effectuer la même fonction. De cette façon, si une entité défaille, une autre la remplace afin de continuer à réaliser la fonction attendue. Les calculs de trajectoire de la fusée Ariane sont réalisés par des ordinateurs, pour lesquels une quintuple redondance a été prévue, quatre ordinateurs peuvent donc défaillir sans que cela n'altère le système. Dans les centrales nucléaires, les capteurs des générateurs de vapeurs font l'objet d'une septuple redondance. La physique des générateurs de vapeur est très complexe, car le système est multi-phase (mélange eau-vapeur, à des pressions et températures très élevées). La redondance permet d'avoir un plus grand nombre de mesures et d'interpréter ces données afin d'obtenir la meilleure estimation possible du niveau de l'eau. En outre, si le processus physique n'est pas correctement maîtrisé,

1. Une entité est un composant ou un ensemble de composants.

il y a un risque d'accident dont la criticité est élevée puisqu'il s'agit d'une centrale nucléaire; cela a un impact sur la sûreté du système.

Un second principe consiste à garantir l'indépendance de traitements dans les différentes chaînes de calcul et d'utiliser des technologies différentes dans les chaînes redondées. Le Rafale de Dassault Aviation est conçu avec une quadruple redondance, dont trois chaînes numériques et une chaîne analogique séparée, et ne comporte aucune liaison mécanique entre les commandes et les gouvernes. L'indépendance entre les chaînes garantit qu'aucune anomalie ne peut affecter plusieurs chaînes simultanément. De plus, l'utilisation de chaînes numériques et analogique (diversité matérielle) permet de réduire le risque de défaillance de cause commune.

Néanmoins, la redondance est une solution qui a ses limites. Redonder chaque composant une ou plusieurs fois est un processus coûteux, puisque les entités de rechange sont aussi coûteuses que les entités principales. La redondance entraîne également une augmentation du volume et de la masse du système. Cette augmentation peut avoir une incidence sur les performances du système. Dans le cas de la conception d'un train, si la masse du train est augmentée, alors il est nécessaire d'accroître la puissance des moteurs afin que le train puisse rouler à la même vitesse.

Des normes internationales (dont la norme IEC 61508 pour les systèmes de sécurité électriques, électroniques ou électroniques programmables) exigent un niveau minimum de fiabilité afin de garantir un usage sûr des systèmes développés. Ces normes concernent uniquement les aspects de sécurité fonctionnelle assurés par les systèmes instrumentés de sécurité (SIS) et ne régissent pas les questions de fiabilité, disponibilité et maintenabilité.

Afin de satisfaire les normes en vigueur, la probabilité de défaillance du système doit être en-dessous d'un certain seuil. Pour obtenir un système suffisamment fiable, tout en ayant un coût et un dimensionnement (masse, taille, etc.) acceptables, il devient essentiel d'optimiser le système. De plus, lorsque des défaillances surviennent, elles sont également coûteuses, car elles causent des arrêts de service (qui causent une perte financière) et nécessitent la réparation et le remplacement d'entités (qui ont aussi un coût).

Des recherches approfondies en automatique et en informatique ont été consacrées à l'élaboration d'approches, fondées sur des théories mathématiques, qui abordent les propriétés de sûreté. Parmi ces approches, des techniques de vérification formelle permettent de démontrer que les systèmes numériques sont exempts d'erreurs critiques et des techniques probabilistes permettent d'évaluer le niveau de confiance en ces systèmes. Il est primordial, vis à vis de la confiance que nous plaçons dans ces systèmes, qu'ils assurent des propriétés essentielles de sûreté de fonctionnement.

La sûreté de fonctionnement est apparue au XIX^{ème} siècle afin de réglementer la conception des machines à vapeur. Analysant le comportement dysfonctionnel du système, la sûreté de fonctionnement est considérée comme la *science des défaillances*. Une étude de SdF consiste à identifier les défaillances de la manière la plus exhaustive possible, évaluer

leur importance, prévoir les défaillances et maîtriser les défaillances par la réduction de leur occurrence, leur tolérance et la prévention contre leur conséquences.

Ainsi, la SdF consiste à la fois à évaluer des indicateurs globaux de la SdF : les attributs RAMS (Fiabilité, Maintenabilité, Disponibilité, Sécurité) ou des indicateurs de temps moyens tels que MTTF (Mean Time To Failure), MTBF (Mean Time Between Failure) ; et à analyser le système afin de comprendre son comportement (qualitativement) et de l'évaluer (quantitativement). Depuis une cinquantaine d'années, de nombreuses méthodes d'analyse des risques ont été proposées pour répondre à la diversité des besoins et des objectifs des études de la SdF, telles que les AMDEC (Analyse des Modes de Défaillance, de leurs Effets et de leurs Criticité (Reifer (1979))), HAZOP (HAZard and OPerability study (McDermid et al. (1995))) ou les arbres de défaillance (Fault Tree Analysis (Stamatelatos et al. (2002))).

Un arbre de défaillances est une représentation graphique des combinaisons possibles d'événements conduisant à la réalisation de l'événement redouté. Ce formalisme est très répandu dans les études de SdF, car il permet de représenter la composition d'un système et la propagation des défaillances. Ils permettent en particulier d'obtenir la *fonction de structure*, qui est une formule décrivant le comportement dysfonctionnel du système.

Les méthodes formelles sont des techniques, des outils et des langages fondées sur des théories mathématiques afin de spécifier et de vérifier des systèmes physiques et logiciels (Clarke et Wing (1996)). Les méthodes formelles comprennent plusieurs approches, dont la satisfiabilité, la démonstration de théorèmes et les diagrammes de décision binaire (Binary Decision Diagrams ou BDDs). De très nombreux travaux ont d'ores et déjà été réalisés en sûreté de fonctionnement grâce à une modélisation de la fonction de structure par BDDs. Les techniques de satisfiabilité étudient des formules mathématiques et permettent de déterminer, pour chaque formule, s'il existe une solution qui rende la formule vraie, ou si elle ne possède pas de solution. Les techniques de satisfiabilité pourraient s'appliquer à une formule comme la fonction de structure et n'ont pas encore été utilisées pour des études de fiabilité.

La problématique de ce manuscrit est la suivante :

Comment évaluer le niveau de confiance d'un système en s'appuyant sur la vérification formelle symbolique ?

Pour répondre à cette problématique, ce mémoire a pour objectif de proposer une approche employant les techniques de satisfiabilité en logique propositionnelle afin d'analyser les propriétés de sûreté des systèmes et de déterminer l'ensemble de liens² minimaux d'un système. Il est primordial d'identifier l'ensemble des liens minimaux afin de réaliser

2. Un lien est une combinaison de composants dont le fonctionnement garantit le fonctionnement du système.

l'analyse probabiliste (quantitative). Pour ce faire, nous proposerons une méthodologie globale permettant de vérifier formellement les propriétés de sûreté et d'évaluer le niveau de confiance effectif d'un système. Cette méthodologie a pour objectif de générer les liens minimaux afin de permettre l'évaluation de la fiabilité.

Pour satisfaire ces objectifs, ce manuscrit est organisé en trois chapitres, comme suit :

Le premier chapitre réalise un état de l'art des techniques de modélisation des systèmes dans le but d'effectuer une étude de fiabilité, puis il présente les techniques et outils des méthodes formelles. Il présente ensuite les principales techniques d'évaluation probabiliste des systèmes et conclut par une synthèse argumentée.

Le deuxième chapitre propose d'analyser des systèmes modélisés par des arbres de défaillances statiques. À partir de ces arbres de défaillances, nous déduisons la fonction de structure du système afin d'employer les techniques de satisfiabilité pour déterminer les liens minimaux d'un système. Les liens minimaux représentent les configurations minimales de fonctionnement à partir desquelles toute défaillance supplémentaire entraîne la perte du service attendue. Les liens minimaux permettent d'effectuer une étude qualitative du système, mais également de construire le diagramme de Hasse du système afin de déterminer le polynôme de fiabilité du système. Enfin, ce chapitre propose plusieurs applications.

Le troisième chapitre développe une analyse des systèmes modélisés par des arbres de défaillances dynamiques. Il propose une méthodologie permettant d'écrire une forme propositionnelle de la fonction de structure d'un système dit dynamique. Cette formule est analysée grâce aux techniques de satisfiabilité afin d'identifier les séquences de lien minimales, qui est l'extension des liens minimaux aux systèmes dynamiques. Ces séquences de lien minimales permettent d'une part de réaliser une étude qualitative et, d'autre part, de construire une extension du diagramme de Hasse aux systèmes dynamiques. À partir du diagramme de Hasse, nous proposons de déterminer le polynôme de fiabilité du système afin d'évaluer sa fiabilité sur un intervalle de temps. Nous proposons une application de cette approche à un cas d'étude.

Ce manuscrit s'achève par des conclusions et des propositions de perspectives de recherche dans la continuité de nos travaux.

Chapitre 1

État de l'art

L'objectif de ce manuscrit est de proposer une contribution aux techniques pour l'évaluation du niveau de confiance des systèmes, grâce à l'utilisation de méthodes formelles. Dans ce chapitre, nous introduisons tout d'abord la Sûreté de Fonctionnement et ses notions élémentaires. La deuxième partie dresse un état de l'art des différentes modélisations permettant de décrire le comportement d'un système, sans avoir recours au langage naturel. Dans la troisième partie, nous présentons une synthèse bibliographique dans le domaine des méthodes formelles. Enfin, le chapitre se conclut par une synthèse argumentée qui justifie le choix des arbres de défaillance et des techniques de satisfiabilité dans le cadre de notre étude.

1.1 Introduction

Une *entité* est définie par Villemeur (1988) comme tout élément, composant, unité fonctionnelle, équipement, système ou sous système, que l'on peut considérer individuellement. Elle peut être constituée d'éléments matériels (technologies de toute nature), d'éléments immatériels (logiciels, calculs...), d'éléments vivants (plantes, bactéries...) et bien sûr d'hommes (opérateurs, utilisateurs...) ou de toute combinaison de ces éléments. Le système et ses composants sont des entités et un ensemble d'entités peut aussi être considéré comme une entité.

Avizienis et al. (2004) définit une *défaillance* du service comme étant un évènement qui survient lorsque le service délivré dévie du service correct, parce qu'il n'est plus conforme à la spécification. Un système est considéré défaillant lorsqu'il ne peut pas assurer sa mission. Les *modes de défaillance* sont donc les différentes manières selon lesquelles une entité peut défaillir, classées selon leur gravité. La sûreté de fonctionnement, ou *science des défaillances*, cherche à les identifier de la manière la plus exhaustive possible.

Au sens le plus strict, la Sûreté de Fonctionnement est l'aptitude d'une entité à assumer une ou plusieurs fonctions requises dans des conditions données (IEC (2000)), ce sont les

fonctions dont l’accomplissement est considéré nécessaire afin de fournir le service attendu. Elle consiste à étudier le comportement dysfonctionnel d’un système.

1.2 Sûreté de fonctionnement

Un principe fondamental de la sûreté de fonctionnement est la *redondance*. Elle consiste à mettre à disposition plusieurs ressources pour réaliser un même service, afin que le système puisse tolérer certaines défaillances, qui sont considérées comme fréquentes et dangereuses. Deux types de redondance sont à distinguer. D’une part, les redondances *matérielles* consistent à utiliser plusieurs composants similaires pouvant réaliser le même service. D’autre part, les redondances *fonctionnelles* font appel à plusieurs solutions techniques permettant d’assurer le même service. Ces redondances peuvent être *actives* et opérer en parallèle, ou *passives* et proposer un composant de rechange lorsque le composant en service défaille. Toutefois, la redondance engendre un surcoût et augmente la masse et le volume du système.

Avizienis et al. (2004) enrichit la définition de la sûreté de fonctionnement en la considérant comme l’aptitude à délivrer un service de *confiance justifiée*. Elle désigne un ensemble de concepts, méthodes et outils permettant d’évaluer le degré de confiance qu’il est légitime d’accorder à un système pour accomplir la mission qui lui a été attribuée. Cette connaissance permet aux utilisateurs du système de placer une confiance justifiée dans le service que le système leur fournit.

Cette notion de confiance est fondamentale puisque tous les éléments matériels, immatériels, humains et vivants peuvent être dysfonctionnels et impacter cette entité. Afin d’évaluer la confiance accordée à un système, des indicateurs nommés RAMS sont définis par Villemeur (1988) et présentés sur la Figure 1.1. Ces quatre indicateurs sont :

- Fiabilité (Reliability),
- Disponibilité (Availability),
- Maintenabilité (Maintainability) et
- Sécurité-innocuité (Safety).

1.2.1 Indicateurs RAMS

La *fiabilité* d’un système est l’aptitude de ce système à accomplir une fonction requise dans des conditions données, pendant une durée donnée. La fiabilité se mesure par la probabilité que le système accomplisse la fonction requise pendant l’intervalle de temps donné. L’aptitude contraire représente la probabilité de défaillance, nommée parfois la *défiabilité*. À l’origine en 1960, la fiabilité était à elle seule la sciences des défaillances. L’histoire de l’analyse de risques et de la fiabilité est présentée dans Zio (2013). Le terme

fiabilité permet également de désigner la valeur de la fiabilité. Ainsi, la fiabilité est la probabilité qu'une entité puisse accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné.

La *disponibilité* d'un système est l'aptitude de ce système à accomplir la fonction requise à un instant donné et dans des conditions données. La mesure de la disponibilité est donnée par la probabilité qu'une entité soit en état d'accomplir une fonction requise dans des conditions données à l'instant t . L'inverse de la disponibilité est l'indisponibilité.

La *maintenabilité* d'un système est l'aptitude d'un système à être maintenu et réparé dans un état dans lequel il peut accomplir la fonction requise. Elle est mesurée par la probabilité que la maintenance de l'entité s'achève à l'instant t , sachant que l'entité est défaillante à l'instant initial $t_0 = 0$. L'évaluation de cette probabilité est liée à la manière dont la remise en état de fonctionnement est effectuée.

La *sécurité-innocuité* est l'aptitude d'un système à accomplir sa fonction sans causer de lésions ou d'atteinte à la santé. Elle est définie comme l'absence de risque inacceptable. C'est la sécurité des personnes. Par extension, elle est définie comme l'aptitude d'une entité à éviter de faire apparaître des événements critiques ou catastrophiques, c'est-à-dire pouvant affecter les personnels et les équipements. L'étude de la sécurité-innocuité est essentielle lorsqu'il s'agit de systèmes critiques tels que les systèmes nucléaires, aérospatiaux, aéronautiques, ferroviaires et automobiles. On constate une certaine ambiguïté entre les mots sécurité et sûreté en français d'une part ; et les mots safety et security d'autre part. Elle est nommée "sécurité innocuité" (safety) afin de la distinguer de la "sécurité-immunité" qui est l'absence d'accès ou de manipulations non-autorisés de l'état du système (qui regroupe la disponibilité, la confidentialité et l'intégrité du système).

La *confidentialité* d'un système est l'aptitude du système à assurer l'absence de divulgations non autorisées de l'information.

L'*intégrité* d'un système est la capacité du système à assurer de l'absence d'altérations inappropriées du système.

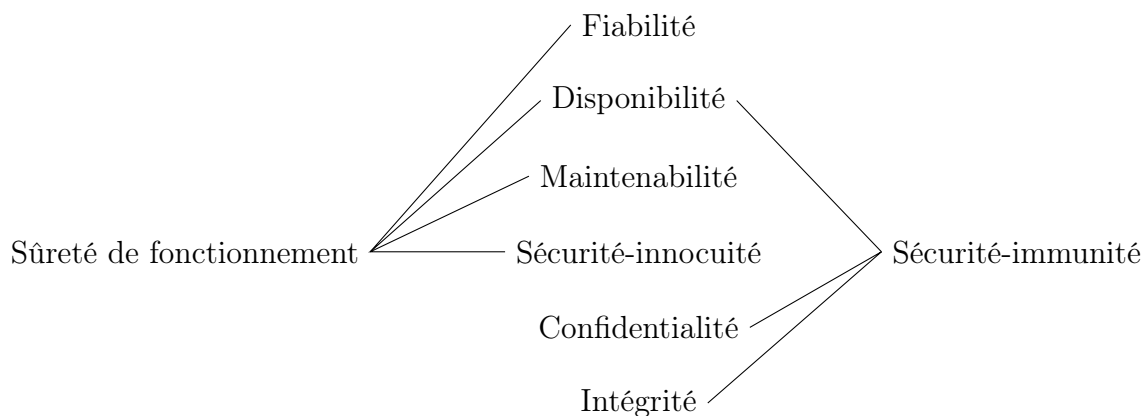


FIGURE 1.1 – Indicateurs de sûreté de fonctionnement et de sécurité-immunité.

Cette thèse s'intéresse tout particulièrement à la fiabilité. Néanmoins, étant donné que la fiabilité analyse toutes les défaillances du système et que la sécurité-innocuité concerne uniquement les défaillances dangereuses, les moyens mis en oeuvre pour réaliser une étude de fiabilité peuvent également être utilisés pour une étude de sécurité-innocuité.

Pour garantir qu'un système soit fiable, disponible, maintenu et sûr, les moyens déployés en sûreté de fonctionnement sont : la prévention de fautes, la tolérance aux fautes, l'élimination des fautes et la prévision des fautes.

1.2.2 Analyses qualitatives et quantitatives

Afin d'estimer la fiabilité d'un système, l'analyse du système se déroule en deux phases :

- l'analyse préliminaire des risques, et
- l'analyse du comportement dysfonctionnel.

La première phase d'analyse préliminaire des risques permet de formaliser la connaissance du système sur les risques de dysfonctionnement des composants.

La méthode la plus fréquemment utilisée est l'AMDEC (Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité) qui fait l'objet de normes internationales, comme le standard IEC (812). L'AMDEC vise à répertorier les différents modes de défaillance de chaque composant, les causes associées à chaque mode de défaillance et leur criticité. La seconde méthode d'analyse qualitative la plus répandue est l'Analyse Préliminaire des Risques (APR). En parallèle de cette analyse qualitative du système, il est nécessaire de quantifier les probabilités de défaillances des composants, qui sont le plus souvent estimées grâce à des retours d'expérience (REX) ou des tests et essais spécifiques dans des laboratoires de test.

La seconde phase d'analyse du comportement dysfonctionnel vise à exploiter la connaissance produite par la première, afin de comprendre le comportement dysfonctionnel du système. D'un point de vue qualitatif, elle cherche à déterminer les scénarios d'évènements conduisant à la défaillance du système ; tandis que, d'un point de vue quantitatif, il s'agit d'estimer les indicateurs probabilistes de la sûreté de fonctionnement (fiabilité, disponibilité, temps moyen avant la première défaillance (MTTF), etc.).

L'analyse quantitative passe aussi par le calcul du λ équivalent du système. La valeur associée à un tel paramètre n'a de signification que pour la durée considérée et l'utiliser dans un modèle non exponentiel pour prédire la fiabilité au bout d'un temps différent n'a a priori aucun sens (Ringler (1988)).

Les défaillances peuvent être classifiées selon : la rapidité et l'importance de la défaillance (catalectique ou par dégradation) ; le type de manifestation (systématique ou aléatoire) ; les effets : (mineure ou bénigne, significative, critique ou catastrophique) ; les causes ; la date d'apparition : défaillance précoce ou par vieillissement (usure). La norme IEC 61508 (Bell (2006)) s'applique aux systèmes de sécurité électriques, électroniques

ou électroniques programmables et utilisant des techniques connexes dans le domaine industriel, destinés à exécuter des fonctions de sécurité. Elle définit le niveau de sécurité attendu pour le système (Safety Integrity Level, SIL).

L'analyse qualitative cherche à comprendre le comportement du système tandis que l'analyse quantitative vise à évaluer le système. Ainsi, l'analyse qualitative étudie les mécanismes, les risques et les combinaisons ou séquences d'événements conduisant le système à un événement redouté, et l'analyse quantitative calcule de manière formelle des indicateurs probabilistes et des indicateurs de temps moyens. Une analyse qualitative des systèmes critiques s'appuie sur l'identification des combinaisons de composants dont le fonctionnement (resp. la défaillance) garantit que le système fonctionne (resp. défaille), nommées liens (resp. coupes).

Afin d'estimer le niveau de confiance que nous pouvons accorder aux systèmes, la section suivante présente les techniques usuelles de modélisation des systèmes binaires ¹.

1.3 Modélisation des systèmes

Les spécifications des systèmes industriels étaient écrites en langue naturelle par le passé. Toutefois, la langue naturelle ne permet pas d'automatiser la vérification et la validation de la cohérence entre les spécifications des systèmes. De plus, l'ambiguïté inhérente des textes en langue naturelle complexifie l'inspection du modèle. Afin que la conception des systèmes industriels ne dépendent plus uniquement de documents textuels, des approches sont développées pour fournir des outils réactifs et adaptés au contexte industriel, ce qui explique l'essor de l'ingénierie basée sur les modèles (Lebeaupin (2017)).

Le modèle d'un système est défini par IEEE (1990) comme étant une abstraction de celui-ci, visant à représenter un ou plusieurs aspects de sa structure et/ou de son comportement, sous une forme idéalisée et approximative. Les approches basées sur des modèles peuvent être divisées en deux catégories (Piètre-Cambacédès (2010)). D'une part, les modèles structurels font l'hypothèse de l'indépendance des composants et les principaux modèles structurels sont les diagrammes de fiabilité et les arbres de défaillances statiques. D'autre part, les modèles comportementaux offrent la possibilité de saisir l'influence du temps sur le système et les principaux modèles comportementaux sont les arbres de défaillances dynamiques, les chaînes de Markov et les réseaux de Petri. Les capacités de modélisation des modèles comportementaux sont plus puissantes que celles des modèles statiques, mais ces modèles sont également plus complexes à construire et analyser.

Un système est défini comme statique par Birnbaum et al. (1961) lorsque son état chaque instant est uniquement déterminée par l'état de ses composants au même instant.

1. Un système est dit binaire lorsqu'il peut être décrit par une variable booléenne (fonctionnement ou défaillance), ainsi que ses composants.

Birnbaum et al. (1961) définit également la *fonction de structure*, qui est la relation entre la défaillance du système et la défaillance de ses composants.

D’après le livre de Birolini (1999), un système est dit *cohérent* si :

- La fonction de structure du système dépend de l’état de tous les composants et
- La fonction de structure est monotone. Dans ce cas, il n’existe aucun état du système tel qu’à partir de cet état, la défaillance d’un composant puisse réparer le système.

Dans la suite de nos travaux, les capacités et limites des modèles sont analysés en considérant leur aptitude à fournir la fonction de structure du système.

1.3.1 Diagramme de fiabilité

Le bloc-diagramme de fiabilité (BDF/RBD) est un modèle graphique utilisé en sûreté de fonctionnement pour analyser les systèmes et estimer leur fiabilité (Čepin (2011)). Il permet de déterminer l’état du système en fonction des états de ses composants. C’est une approche statique puisqu’elle modélise des structures logiques indépendantes du temps, et booléenne car elle permet uniquement de représenter des composants à deux états. Elle permet à la fois une étude qualitative, en déterminant la fonction de structure puis les coupes minimales du système, et quantitative, en calculant la fiabilité du système. Il existe des extensions couplant les diagrammes de fiabilité et d’autres modèles comme les réseaux de Petri dans des modèles hiérarchiques à deux niveaux. Pour un couplage avec les réseaux de Petri par exemple, à chaque bloc du diagramme de fiabilité est associé un modèle par réseau de Petri. Les diagrammes de fiabilité modélisent le fonctionnement du système et chaque chemin allant d’une extrémité à l’autre est une combinaison de composants dont le fonctionnement garantit le fonctionnement du système (Figure 1.2).

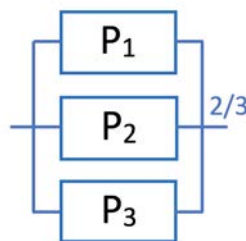


FIGURE 1.2 – Diagramme de fiabilité d’un système de 3 pompes en redondance 2-parmi-3.

1.3.2 Arbres de défaillances

Les arbres de défaillance (fault trees ou FT) ont été initialement proposés par Watson et al. (1961), puis décrits comme représentation graphique de la fonction de structure définie par Birnbaum et al. (1961) et Fussell et al. (1976). Le choix des FTs par la NASA

à travers le Fault Tree Handbook (Stamatelatos et al. (2002)) a permis une large diffusion de cette modélisation dans le milieu industriel. Elle est d'ailleurs parmi les méthodes les plus largement utilisées à l'heure actuelle (Chaux (2013)). Stamatelatos et al. (2002), Villemeur (1988) et Limnios et al. (2005) sont des livres de référence en matière d'arbres de défaillances.

L'arbre de défaillances partage les mêmes bases booléennes et probabiliste que le diagramme de fiabilité. Toutefois, contrairement au diagramme de fiabilité qui modélise le fonctionnement du système, l'arbre de défaillances modélise la défaillance du système.

Un arbre de défaillances (fault tree, FT) est un graphe direct acyclique. C'est une représentation graphique de la propagation des défaillances dans un système, représentant la manière dont la défaillance des composants conduit à la défaillance du système. La défaillance du système est communément nommée *événement redouté*. Un arbre de défaillances est constitué d'événements représentant la défaillance des composants, qui sont supposés indépendants, et de portes logiques.

1.3.2.1 Arbres de défaillances statiques

Les arbres de défaillances standards, ou statiques, permettent de modéliser les systèmes pour lesquels l'ordre des défaillances et l'instant de temps auquel les défaillances se produisent n'impactent pas le fonctionnement du système. Un système est dit statique lorsqu'il peut être modélisé par un arbre de défaillances statiques, i.e. dont toutes les portes sont statiques. Représentées sur la Figure 1.3, les principales portes *statiques* sont AND, OR et KooN.

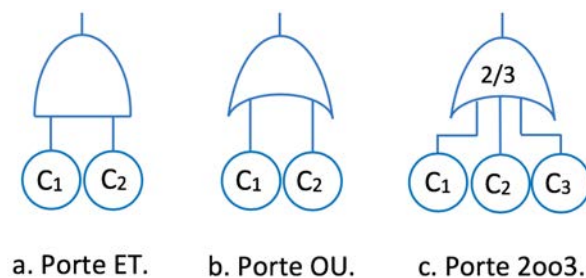


FIGURE 1.3 – Portes statiques.

La porte *ET* s'active lorsque toutes ses entrées s'activent.²

La porte *OU* (son dual) s'active lorsqu'au moins une de ses entrées s'active.

La porte *K-parmi-N* (K-out-of-N, KooN) s'active lorsqu'au moins K des N entrées s'activent. Bien qu'elle puisse être définie par une combinaison de portes ET et OU, il est pratique de la considérer comme une porte à part entière. L'exemple d'une porte 2oo3 est représentée sur la Figure 1.3.c.

2. Un composant et un système défontent, tandis qu'une porte et une entrée s'activent.

La porte NON, également statique, a une seule entrée et s'active lorsque son entrée fonctionne. Les portes XOR, NAND, NOR sont mentionnées dans la littérature. Bennetts (1975) proposent les expressions des portes NAND et NOR en fonctions des portes statiques ET, OU et NON grâce au théorème de Morgan. Han et al. (2005) proposent des modèles de fiabilité pour les portes logiques NAND, NOR, XOR, NOT, Majority, Interconnect et une table de vérité pour la porte Majority; la porte Majority étant un cas particulier de la porte KooN.

Nos travaux se concentrent sur les systèmes cohérents. Or, pour représenter un système cohérent, il existe au moins une expression de la fonction de structure qui s'écrit sans opérateur de négation \neg . La porte NON et ses extensions (XOR, NOR, NAND) ne seront donc pas traitées dans ce manuscrit.

Il existe de nombreuses extensions des arbres de fautes standards (ou statiques). Nous présentons certaines d'entre elles, dont les arbres de défaillances dynamiques.

1.3.2.2 Arbres de défaillances dynamiques

La volonté d'améliorer la fiabilité des systèmes industriels a conduit à la conception de systèmes dont l'évolution dépend du temps et à une interdépendance entre les états des composants (certains composants pouvant être actifs ou dormants, car non sollicités). De tels comportements de défaillance ne pouvant pas être modélisés par les portes logiques définies initialement, de nouvelles portes dites dynamiques ont été créées.

La Figure 1.4, proposée par Ionescu (2016), met en évidence le fait qu'« une séquence d'événements peut conduire le système dans un état dangereux alors que les mêmes événements se produisant dans un ordre différent n'y conduisent pas ».

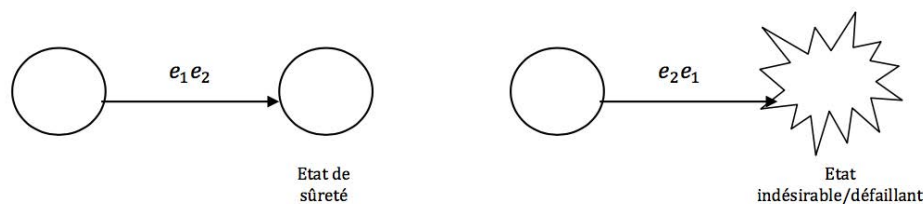


FIGURE 1.4 – Installation industrielle dans deux états différents.

L'évaluation de la fiabilité des systèmes dynamiques et complexes est souvent réalisée par une analyse quantitative par arbres de défaillances dynamiques, qui modélisent la défaillance du système en fonction des défaillances des composants. Les arbres de défaillances dynamiques (Dynamic Fault Tree, DFT) prennent en considération les relations séquentielles entre les événements. À partir des probabilités de défaillance des composants, l'analyse quantitative permet d'évaluer la probabilité de défaillance du système.

Merle (2010) propose un cadre algébrique afin de déterminer une forme canonique de la fonction de structure d'un système à partir de son DFT, dans le but d'identifier

les séquence de coupe (CSS) du système. Merle et al. (2011) proposent de déterminer la fonction de structure d'un système à partir de son arbre de défaillances dynamique. Pour ce faire, il introduit trois nouveaux opérateurs représentant la simultanéité des défaillances, la précédence d'une défaillance par rapport à une autre et "la simultanéité ou la précédence".

La Figure 1.5 présente les trois principales portes dynamiques.

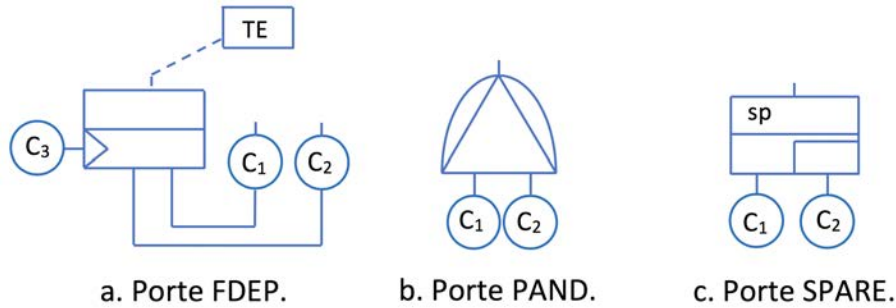


FIGURE 1.5 – Portes dynamiques.

Porte FDEP

Proposée par Dugan et al. (1990), utilisée par Boyd (1992), la porte FDEP (Functional-DEPendency) modélise une dépendance fonctionnelle entre une *gâchette* (située à gauche de la porte) et des composants en entrée (Figure 1.5.a).

Lorsque la gâchette s'active, elle provoque l'activation immédiate des entrées. Si l'alimentation électrique d'un système est assurée par un générateur électrique, alors la défaillance de ce générateur entraîne la défaillance des autres composants nécessitant du courant, qui ne peuvent plus assurer leur mission.

La porte FDEP est assimilée à une porte OU par Merle et al. (2010) (et donc à une porte statique) lorsqu'il n'y a pas d'interdépendance entre porte FDEP. Néanmoins, une telle modélisation omet la relation logique (et donc le fait que le composant ne peut plus défaillir après l'activation de la gâchette) :

$$\text{activation de la gâchette} \Rightarrow \text{activation des entrées.}$$

Porte PAND

La porte PAND (Priority-AND) est introduite par Fussell et al. (1976) (Figure 1.5.b). Elle modélise l'effet de l'ordre d'occurrence des défaillances sur le système. La porte PAND s'active si tous les entrées s'activent dans un ordre précis, de la gauche vers la droite, suivant leur connexion à la porte. Les entrées des portes PAND peuvent être des composants, des portes statiques, PAND et SPARE.

Ruijters et Stoelinga (2015) considèrent que l'ajout d'une porte PAND altère la cohérence générale du système d'un point de vue probabiliste, car une augmentation du taux de défaillance de l'entrée droite d'une porte PAND entraîne une augmentation de la fiabilité du système.

Porte SPARE

La porte SPARE (littéralement : *de rechange*) est définie par Dugan et al. (1990) pour modéliser l'utilisation des composants de rechange. Elle possède un composant principal et un (ou plusieurs) composant de rechange. Les composants de rechange peuvent être partagés, comme le composant C_2 de la Figure 1.5.c, ou réservés à une porte. Lorsque le composant principal défaille, le composant qui est sollicité en premier est celui situé le plus à gauche dans l'ordre de connexion à la porte, à condition qu'il ne soit pas déjà sollicité par une autre porte, auquel cas, le composant suivant serait sollicité.

Le type de redondance d'une porte SPARE indique le type d'utilisation des composants de secours avant leur sollicitation. Nous différencions ceux qui ne défont pas avant sollicitation (cold), ceux qui défont de la même façon avant et après la sollicitation (hot) et ceux qui défont différemment avant et après la sollicitation (warm).

Les composants de rechange sont parfois modélisés par un comportement multi-états afin de savoir si le composant est sollicité ou non. Puisque nous nous focalisons sur le fait que le composant soit fonctionnel et apte à remplacer un autre composant, nous ne représentons pas le fait que le composant soit sollicité ou non par une variable et nous modélisons les composants de rechange par une variable booléenne.

Porte SEQ

Une quatrième porte dynamique a été définie par Dugan et al. (1992), nommée SEQ (Sequence Enforcing) qui signifie "séquence forcée". La porte SEQ impose que les entrées s'activent dans un ordre spécifique. D'après Clarhaut (2009), cette porte est utile dans la modélisation des redondances passives. En effet, le composant redondant démarre seulement lorsque le composant principal n'assure plus le service. De ce fait, l'occurrence de la défaillance du composant redondant ne peut se produire qu'après l'occurrence de la défaillance du composant principal. Lorsque les composants ne sont pas sollicités par d'autres portes SEQ, la porte SEQ peut être exprimée au moyen d'une porte SPARE en redondance froide, comme évoqué par Ruijters et Stoelinga (2015). Par conséquent, elle ne sera donc pas traitée dans ce manuscrit.

1.3.2.3 Arbres de défaillances flous

Une extension des arbres de défaillances utilisant des nombres *flous* (fuzzy) est proposée par Tanaka et al. (1983) pour atténuer le problème des probabilités de défaillance qui sont souvent inconnues. Les nombres flous représentent l'incertitude qui existe sur les valeurs des paramètres et permettent d'utiliser une fourchette qui contient la valeur exacte. Mahmood et al. (2013) réalisent un état des l'art sur les arbres de défaillances flous. L'utilisation d'arbres de défaillances flous n'a pas d'incidence sur la fonction de structure, cela impacte uniquement l'évaluation probabiliste.

1.3.2.4 Arbres de défaillances réparables

Afin d'analyser un système sur une longue durée, il peut s'avérer utile de considérer les réparations que ce système peut subir. Celles-ci peuvent allonger la durée avant la première défaillance et survenir alors que le système n'a pas encore défailli en réparant ou remplaçant des composants défaillants en redondance. Elles peuvent également réparer le système défaillant pour le remettre en service (Ruijters et Stoelinga (2015)).

Ruijters et al. (2019a,b) proposent FFORT (Fault tree FOResT), une collection de FTs, statiques, dynamiques et réparables, conçue pour être variée, extensible, en libre accès et facile d'utilisation, afin de fournir aux chercheurs un panel varié d'arbres de défaillances pour tester et valider leurs méthodes et outils.

1.3.2.5 (G)BDMP

Pour répondre au besoin de modélisation des systèmes de production d'énergie, Bouissou et Bon (2003) proposent les BDMPs (Boolean logic Driven Markov Processes) pour effectuer des analyses quantitatives des systèmes dynamiques réparables. Les BDMPs permettent de modéliser des systèmes dynamiques et réparables en alliant les avantages des théories booléennes et markoviennes.

Il s'agit d'un arbre de défaillance simple, dont les feuilles ne sont plus associées à des variables booléennes mais à des processus de Markov afin de représenter un comportement dynamique localisé précis et de considérer des réparations et des défaillances à la sollicitation. De plus, le formalisme a la possibilité de décrire des mécanismes de commutation basiques puisqu'il utilise la primitive de gâchette dans les arbres de défaillance, aussi utilisée pour les portes FDEP des arbres de défaillances dynamiques.

Ce modèle permet d'une part la représentation de la relation entre la sécurité-innocuité du système et la fiabilité de ses composants et d'autre part la modélisation des mécanismes de redondance complexes du système. L'implémentation de ce modèle sur la plateforme KB3 permet l'automatisation des études de sûreté de fonctionnement (Chaux (2013)). Ce formalisme a aussi été utilisé pour des applications à la sécurité informatique par Piètre-Cambacédès et al. (2011) et Kriaa (2016).

Enfin, Piriou (2015) propose une extension nommée BDMP Généralisé (GBDMP) permettant de considérer les systèmes dynamiques, réparables et reconfigurables, et présente un prototype d'outil logiciel, SAGE, à l'usage de l'ingénieur expert en sûreté de fonctionnement. Piriou et al. (2019) utilisent la modélisation par GBDMP pour déterminer les séquences de coupe minimales des systèmes dynamiques, réparables et reconfigurables.

De la même façon que la modélisation par arbres de défaillances réparables, la modélisation par BDMPs ne permet pas de représenter le comportement des systèmes par une modélisation propositionnelle comme la fonction de structure.

1.3.3 Chaînes de Markov

Une autre approche consiste à modéliser le système par une chaîne de Markov (Coccozza-Thivent (2018)). Cet outil de modélisation de type système à événements discrets permet de modéliser le comportement des systèmes incluant des réparations ou des reconfigurations. Cependant, les chaînes de Markov ne permettent pas la génération d’une expression propositionnelle décrivant ce comportement.

1.3.4 Réseaux de Petri stochastiques

Proposés par Petri (1962), les réseaux de Petri modélisent des systèmes qui évoluent sous l’occurrence des événements. Les réseaux de Petri stochastiques sont une extension des chaînes de Markov couramment utilisée dans les études de sûreté de fonctionnement. À la différence des chaînes de Markov qui nécessitent une modélisation globale du système, les réseaux de Petri stochastiques modélisent les comportements locaux du système. L’état global du système résulte des comportements locaux lors de l’évolution de réseau de Petri. Les réseaux de Petri permettent de modéliser le partage de ressources et la synchronisation entre des systèmes. Toutefois, cette modélisation ne nous intéresse pas, car elle ne permet pas d’identifier une fonction représentant le comportement du système.

Par conséquent, seules les modélisations par diagrammes de fiabilité et les arbres de défaillances permettent de générer la fonction de structure du système. Parmi les méthodes utilisées en vérification formelle symbolique, nous cherchons une approche qui permette d’analyser la fonction de structure des systèmes étudiés.

1.4 Étude de formules propositionnelles

Dans la suite de ce manuscrit, le terme *modèle* ne fera plus référence à une modélisation d’un système, mais à une interprétation de variables rendant une formule vraie.

1.4.1 Coupes, liens et implicants premiers

À partir de la fonction de structure, Mortureux (2001) définit une *coupe* comme une combinaison sans lien de causalité d’événements et de conditions suffisantes pour provoquer l’événement indésirable. Une coupe est dite minimale si elle ne contient aucune autre coupe du système. Les coupes minimales permettent d’une part de décrire l’ensemble des défaillances du système de la façon la plus concise possible et d’autre part de faciliter la réalisation des études quantitatives.

Une conjonction logique de littéraux³ est un *implicant* de la fonction de structure f s’il satisfait la fonction de structure (Rauzy et Dutuit (1997); Coudert et Madre (1993)).

3. Un littéral est une variable c de la fonction de structure f ou son complément $\neg c$.

Un implicant est dit *premier* s'il ne contient aucun autre implicant de f .

Lorsqu'un système est cohérent, les implicants premiers sont les coupes minimales du système. Lorsque le système est non cohérent, l'ensemble des coupes minimales est contenu dans l'ensemble des implicants premiers.

Le dual d'une coupe est appelé lien ou chemin de succès. Un *lien* est une combinaison sans lien de causalité d'événements et de conditions suffisantes garantissant le fonctionnement du système. Dans ce manuscrit, le terme lien (plutôt que chemin) est utilisé pour éviter toute confusion avec les chemins du diagramme de Hasse. Kaufmann et al. (1977) a formellement démontré que les concepts de coupes et de liens sont dérivés de ceux de coupes et chemins de la théorie des graphes.

Lorsque l'état d'un système ne dépend plus uniquement des états de ses composants, mais également de l'ordre dans lequel ils défaillent, alors ce système peut être modélisé par un arbre de défaillances dynamique. Les scénarios de défaillances des composants qui conduisent à la défaillance du système sont définis par des séquences de coupe, dont le plus petit sous-ensemble constitué des séquences de coupe les plus courtes possibles est appelé l'ensemble des séquences de coupe minimales. L'intérêt d'utiliser des scénarios plutôt que des coupes est souligné par Clarhaut (2009). Il vient du fait que les scénarios permettent d'améliorer la modélisation du comportement dysfonctionnel d'un système complexe et d'évaluer le niveau de sûreté de fonctionnement de ce système de manière plus précise. En effet, un scénario (ou une séquence d'événements) peut conduire à un événement redouté alors que les mêmes événements se produisant dans un ordre différents n'y conduisent pas. Ce comportement est notamment dû au comportement du système et à l'agencement des composants dans l'architecture.

1.4.2 Formules propositionnelles

Une *formule de la logique propositionnelle* contient des variables propositionnelles (booléennes) et des opérateurs logiques (et \wedge , ou \vee , non \neg , implique \Rightarrow). Une formule propositionnelle est construite à partir de propositions simples, telle que :

$$(l \wedge \neg m \wedge n) \Rightarrow (p \vee q)$$

Une formule prend sa valeur dans l'ensemble $\{\text{vrai}, \text{faux}\}$. Elle est dite *satisfiable* (SAT) lorsqu'il est possible de trouver une valuation des variables qui rende la formule vraie. On dit de cette interprétation que c'est un *modèle* de la formule.

Lorsque la formule utilise des variables, des opérateurs logiques, des prédicats et des quantificateurs, on parle alors de *formule du premier ordre*, par opposition aux *logiques d'ordre supérieur*, où il est possible de quantifier sur les prédicats ou les fonctions.

Les deux formes normales les plus couramment employées sont la CNF (forme normale

conjonctive), qui est une conjonction de disjonctions de littéraux :

$$(l \vee \neg m \vee n) \wedge (p \vee q) \wedge (\neg n \vee q),$$

et la DNF (forme normale disjonctive) qui est une disjonction de conjonctions de littéraux :

$$(l \wedge \neg m \wedge n) \vee (p \wedge q).$$

Toute formule de la logique propositionnelle peut être convertie en une formule CNF ou DNF équivalente, cette formule est dite équi-satisfiable.

Dans le cas d’une étude de fiabilité, la formule étudiée représente la fonction de structure d’un système.

1.4.3 Satisfiabilité

Le modèle généré permet de déterminer un lien ou une coupe du système.

Si nous considérons la formule suivante :

$$(u \wedge w) \vee (v \wedge \neg w),$$

cette formule est satisfiable et u, w est un modèle qui satisfait cette formule.

À présent, si nous considérons la formule

$$u \wedge \neg v \wedge w \wedge (v \vee \neg u),$$

cette formule n’est pas satisfiable, car il n’existe aucun modèle qui la satisfasse.

Un problème est dit NP-complet (non-deterministic polynomial) (Karp (1972)) lorsque :

- une éventuelle solution peut être vérifiée efficacement (complexité polynomiale), et
- tous les problèmes NP-complets peuvent se ramener à celui-ci via une réduction polynomiale (le problème est au moins aussi difficile que les autres problèmes de la classe NP). Un problème qui remplit cette condition est dit NP-difficile.

Un exemple classique de problème NP-complet est le problème de décision du voyageur de commerce, qui cherche à déterminer le plus court chemin du voyageur pour une liste de villes données (Garey et al. (1974)). Historiquement, le problème SAT est l’un des premiers problèmes de décision NP-complet étudié (Cook (1971)).

1.4.3.1 SAT

Le problème SAT est le suivant :

Étant donnée une formule propositionnelle à k variables au format CNF,
existe-t-il une valuation des k variables qui satisfait la formule ?

Présentée dans le manuel de la satisfiabilité de Biere et al. (2009b), la satisfiabilité est un problème des méthodes formelles qui se focalise sur l'assignation de valeurs booléennes afin que la formule de logique propositionnelle soit vraie. Dans le cas d'une formule qu'aucun modèle ne peut satisfaire, on dit de cette formule qu'elle est insatisfiable (UNSAT).

Les solveurs SAT emploient des algorithmes pour calculer un modèle s'il en existe un, un modèle étant une assignation de valeurs qui rendent la formule vraie. Étant donné qu'il s'agit d'un problème NP-complet, tous les algorithmes connus à ce jour ont, au pire des cas, une complexité exponentielle.

L'algorithme DPLL (Davis-Putnam-Logemann-Loveland) (Davis et al. (1961)) est une procédure de décision de SAT. Il alterne entre propagations (assignations de valeurs aux variables, forcées par une clause) et décisions (choix arbitraire d'une valeur pour une variable sans assignation préalable). Nous l'illustrons sur la formule suivante :

$$t \wedge (w \vee u) \wedge (\neg t \vee \neg u \vee \neg w) \wedge (\neg w \vee u)$$

D'abord, nous propageons en assignant t à vrai : t .

$$t \wedge (w \vee u) \wedge (\neg t \vee \neg u \vee \neg w) \wedge (\neg w \vee u)$$

Puis, nous décidons d'assigner w à vrai afin que la clause $w \vee u$ soit vraie : w .

$$t \wedge (w \vee u) \wedge (\neg t \vee \neg u \vee \neg w) \wedge (\neg w \vee u)$$

Nous propageons et assignons u à faux afin que la clause $\neg t \vee \neg u \vee \neg w$ soit vraie : $\neg u$.

$$t \wedge (w \vee u) \wedge (\neg t \vee \neg u \vee \neg w) \wedge (\neg w \vee u)$$

Nous aboutissons à un conflit entre les clauses, puisque la clause $(\neg w \vee u)$ ne peut être vraie si w est assigné à vrai et si u à faux. Nous remontons donc à la dernière décision prise et nous la modifions. Nous propageons donc : $\neg w$.

$$t \wedge (w \vee u) \wedge (\neg t \vee \neg u \vee \neg w) \wedge (\neg w \vee u)$$

Nous propageons et assignons u à vrai. Nous obtenons le modèle $\{t, \neg w, u\}$.

Un second algorithme, CDCL (conflict-driven clause learning), a été proposé comme une amélioration de DPPL pour la résolution de problèmes SAT (Silva et Sakallah (2003), Biere et al. (2009a)). L'algorithme CDCL génère des "clauses de conflit" qui représentent l'essence des conflits générés de manière concise. La différence principale entre DPLL et CDCL réside dans le fait que les sauts en arrière de CDCL ne sont pas réalisés par ordre chronologique, puisque l'algorithme CDCL se base sur ces clauses de conflit lorsqu'un saut

en arrière est nécessaire.

Pour utiliser les solveurs SAT, il est nécessaire que la formule soit au format CNF. Deux méthodes permettent de transformer une formule en une formule au format CNF. D'une part, il est possible de développer la formule jusqu'à obtenir une conjonction de disjonctions ; d'autre part, il est possible d'utiliser la transformation de Tseitin (1983), qui permet de transformer une formule comme la formule suivante :

$$(x \vee \neg y) \Rightarrow \neg z,$$

en une formule CNF équi-satisfiable, par l'introduction de nouvelles variables :

$$\begin{aligned} & (a \Leftrightarrow x \vee \neg y) \wedge (a \Rightarrow \neg z) \\ \Rightarrow & (\neg a \vee x \vee \neg y) \wedge (\neg x \vee a) \wedge (y \vee a) \wedge (\neg a \vee \neg z). \end{aligned}$$

La transformation de Tseitin est linéaire en la taille de l'entrée. Il peut sembler risqué d'alourdir la résolution SAT ultérieure car le problème est NP-complet en le nombre de variables. Dans les faits, cela demeure plus rapide que de développer la formule. De plus, la transformation de Tseitin n'est pas sujette à l'explosion combinatoire, comme le prouve les travaux de Heule et al. (2016).

Une simplification due à Plaisted et Greenbaum (1986) consiste à remplacer l'équivalence entre la nouvelle variable et la sous-formule qu'elle nomme par une implication :

$$\begin{aligned} & (a \Rightarrow (x \vee \neg y)) \wedge (a \Rightarrow \neg z) \\ \Rightarrow & (\neg a \vee x \vee \neg y) \wedge (\neg a \vee \neg z). \end{aligned}$$

Une formule propositionnelle peut toujours être transformée en CNF, ou DNF, en préservant l'équi-satisfiabilité. Une conversion au format CNF est proposée par Boy de la Tour (1992) et améliorée par Nonnengart et al. (1998).

Parmi les nombreux solveurs SAT existant, nous pouvons citer par exemple les solveurs MiniSAT (Eén et Sörensson (2003)), SAT4J (Le Berre et Parrain (2010)), Glucose (Audemard et Simon (2009)) et Lingeling (Biere (2011)).

1.4.3.2 SMT

La satisfiabilité modulo théories (SMT) permet d'une part de considérer d'autres théories que la logique booléenne et d'autre part, de s'intéresser à des formules où interviennent plusieurs théories combinées (souvent sans quantificateurs). Elle est utilisée en informatique et en logique mathématique et elle inclut notamment l'arithmétique linéaire sur les entiers, l'arithmétique polynomiale sur les réels et diverses structures de données comme

les listes.

Si on considère la formule

$$(u \wedge x > 1) \vee (y < 0 \wedge x < 1),$$

cette formule est satisfiable et $u = \text{vrai}$ ⁴, $x = 3$ est un modèle qui satisfait cette formule.

En considérant comme formule la fonction de structure, le modèle obtenu est une configuration qui garantit que le système défaille. L'étude d'un système par des techniques de satisfiabilité permettra d'identifier les coupes des systèmes que nous considérons. Afin d'identifier les liens, il est nécessaire de considérer la négation de la fonction de structure, ce qui permet d'obtenir les configurations garantissant que le système fonctionne.

1.5 Évaluation probabiliste

1.5.1 Fiabilité des composants

La défaillance des composants d'un système peut être représentée par différentes lois de distributions et leurs paramètres.

La loi de probabilité *exponentielle* modélise correctement la distribution des durées de vie lorsque le taux de défaillance λ est constant et que les défaillances sont indépendantes, imprévisibles, et dont la génération obéit à un processus de Poisson. Ce modèle est le plus utilisé car il correspond à la période de vie utile, représenté par le palier de la courbe en baignoire de la Figure 1.7.

La Figure 1.6 représente la fonction de répartition $F(t)$ qui est égale à la probabilité de défaillance, donnée par l'équation (1.1).

La fiabilité $R(t)$ est calculée à partir de la fonction de répartition : $R(t) = 1 - F(t)$.

$$1 - F(t) = e^{-\lambda \cdot t} \quad (1.1)$$

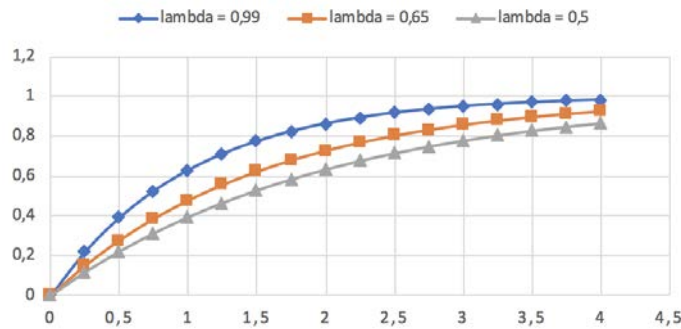


FIGURE 1.6 – Courbes de la fonction de répartition exponentielle.

4. Par la suite, la notation u impliquera que u est vrai et $\neg u$ désignera le cas où u est faux.

La fonction exponentielle permet de construire une courbe en baignoire modélisant la vie d'un système, comprenant sa jeunesse, sa période utile et sa vieillesse. On observe sur la Figure 1.7 que la probabilité de défaillir du système est plus élevée durant la jeunesse et la vieillesse du système que durant sa période utile.

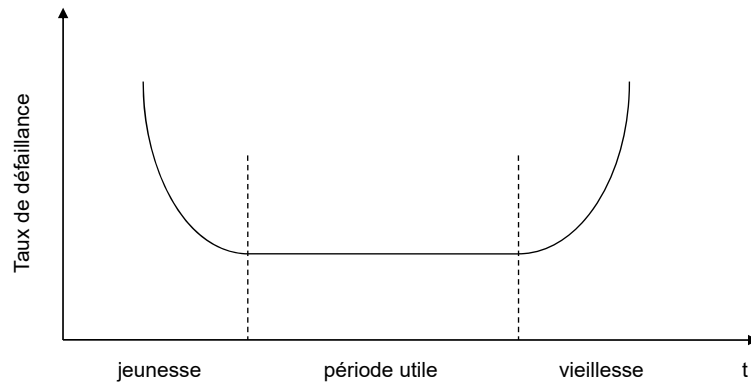


FIGURE 1.7 – Courbe en baignoire.

La distribution de Weibull (Weibull et al. (1951)) est fréquemment utilisée pour les composants dont les taux de défaillance varient beaucoup, notamment pour les composants mécaniques.

La fiabilité $1 - F(t)$ du composant est donnée par l'équation (1.2), où k est le paramètre de forme, λ est le paramètre d'échelle de la distribution et θ est le paramètre de position (Stone et Van Heeswijk (1977)).

$$1 - F(t) = e^{-\left(\frac{t-\theta}{\lambda}\right)^k} \quad (1.2)$$

Les courbes de la loi de Weibull où varient k et λ sont données par la Figure 1.8, θ étant fixé à zéro.

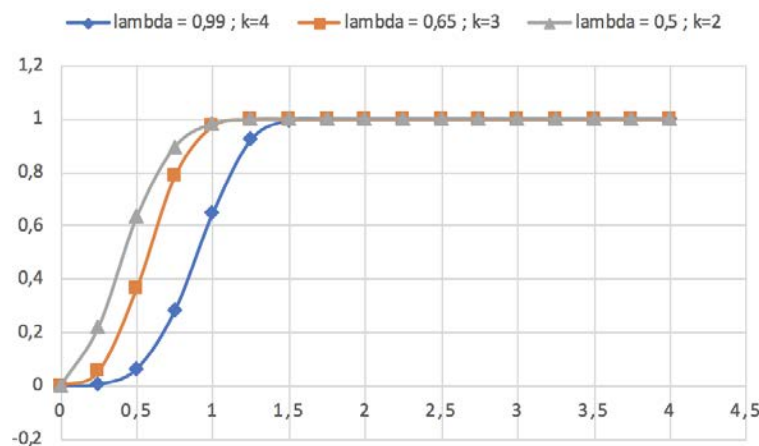


FIGURE 1.8 – Courbes de la fonction de répartition de Weibull.

La loi de probabilité *normale* s'applique aux composants dont le taux de défaillance augmente avec le temps, tels que les batteries qui subissent un grand nombre de charges et décharges.

La loi de probabilité *log-normale* s'applique aux systèmes dont les caractéristiques fonctionnelles sont influencées par un nombre importants de facteurs indépendants agissant de manière multiplicative. Elle convient parfaitement aux matériaux travaillant en fatigue sous l'action répétée de contraintes mécaniques. Elle est notamment utilisée pour les composants optoélectroniques comme les diodes laser.

1.5.2 Fiabilité du système

Les approches permettant une évaluation probabiliste peuvent être classées en deux familles : les approches analytiques et les approches par simulation.

1.5.2.1 Approches analytiques

Les approches analytiques les plus utilisées pour quantifier la fiabilité sont le principe d'inclusion-exclusion, le model checking et les diagrammes de décision binaire.

Principe d'inclusion-exclusion

La première approche est le calcul de la probabilité d'occurrence de l'évènement indésirable basée sur le principe d'inclusion-exclusion, ou théorème de Sylvester-Poincaré (Narushima (1974, 1982)).

La probabilité d'occurrence de l'évènement indésirable est égale à la probabilité qu'une coupe au moins soit vraie parmi toutes les coupes. Si le système contient m coupes, alors la probabilité d'occurrence de l'évènement indésirable est :

$$P(S) = P(C_1 \vee \dots \vee C_m)$$

La probabilité d'une coupe est égale au produit des probabilités de chacun de ses évènements élémentaires, lorsque les défaillances des composants sont supposées indépendantes. Les coupes étant des combinaisons d'évènements de base, deux coupes sont corrélées lorsqu'elles ont un ou plusieurs évènements de base en commun.

Le théorème de Sylvester-Poincaré donne l'équation suivante :

$$P(S) = \sum_{i=1}^m P(C_i) - \sum_{j=2}^m \sum_{i=1}^{j-1} P(C_i \wedge C_j) + \dots + (-1)^m \cdot P(C_1 \wedge \dots \wedge C_m)$$

Cette somme est une somme alternée dont les valeurs absolues sont décroissantes. Il est courant de réaliser une approximation sur le calcul de la probabilité de l'évènement indésirable, afin d'obtenir un résultat avec la précision souhaitée. Néanmoins, la volonté

d'une plus grande précision nécessitera de prendre en compte un plus grand nombre de termes de la somme alternée.

La probabilité est encadrée par la somme des k premiers termes et celle des $k + 1$ premiers termes comme ceci, cette inégalité est connue sous le nom de l'inégalité de Bonferroni :

$$\sum_{i=1}^m P(C_i) - \sum_{j=2}^m \sum_{i=1}^{j-1} P(C_i \wedge C_j) \leq P(S) \leq \sum_{i=1}^m P(C_i).$$

Pour le cas particulier de $k=0$, l'inégalité devient $0 \leq P(S) \leq \sum_{i=1}^m P(C_i)$.

Les diagrammes de décision binaires permettent de rechercher une expression du polynôme représentant l'évènement indésirable afin de déterminer les coupes.

Model-checking de systèmes probabilistes : méthodes numériques

Le model-checking (Emerson et Clarke (1980); Clarke (1997)) est une technique de vérification formelle qui consiste à construire un modèle formel du système à étudier avant de le confronter à des propriétés, exprimées en général sous la forme de formules logiques. Des algorithmes permettent alors de déterminer automatiquement si le système satisfait ou non la propriété, et de générer un contre-exemple lorsqu'il ne la satisfait pas.

Vue l'importance des probabilités pour le calcul de fiabilité des systèmes, nous allons plus particulièrement parler de méthodes prenant en compte cet aspect majeur de nos systèmes. Les méthodes de model-checking probabiliste se divisent en deux catégories : méthodes numériques et méthodes statistiques.

Les méthodes numériques (Reibman et Trivedi (1988)) consistent à construire un modèle complet du système, sur lequel des calculs vont être faits afin de, suivant les cas, s'assurer qu'une propriété est satisfaite ou évaluer la probabilité d'un évènement donné. Leur avantage majeur est la fiabilité sur le résultat donné (ce sont des méthodes dites "exactes"). Leur applicabilité est en revanche limitée par deux aspects. Comme elles nécessitent de construire l'espace d'états du système, elles sont de fait impraticables sur de très grands systèmes. D'autre part elles imposent des contraintes sur le système (notamment sur le type de distribution) sans quoi le problème de la vérification n'est pas décidable.

Diagrammes de décision binaire

Les diagrammes de décision binaire (BDDs) sont des représentations graphiques des fonctions booléennes introduites par Lee (1959) comme une structure de données basée sur l'expansion de Shannon, puis diffusés par Akers (1978). Bryant (1986) en propose une variante plus efficace avec l'introduction des BDDs réduits. Biere et al. (2009b) décrit les BDDs comme une représentation par un graphe direct acyclique de la table de vérité d'une fonction booléenne. La fonction de structure n'est alors plus représentée par une fonction

booléenne mais par un diagramme de décision binaire (BDD) (Rauzy et Dutuit (1997); Dutuit et Rauzy (2005); Ibáñez-Llano et al. (2010)).

La Figure 1.9 représente le diagramme de décision binaire réduit d'un système de trois composants en redondance 2oo3. Un chemin allant du noeud situé en haut jusqu'à un noeud terminal du diagramme permet de déterminer si une assignation de valeurs aux variables rend la fonction de structure vraie (ou non).

De chaque noeud partent un arc annoté 1 et un arc annoté 0.

Les BDDs sans zéro (ZBDD) sont des BDDs basés sur une nouvelle règle de réduction afin de créer une représentation unique et compacte des ensembles qui apparaissent dans de nombreux problèmes combinatoires. Décrits par Minato (1993, 2001), l'utilisation des ZBDDs est ensuite illustrée par Tang et Dugan (2004). Kvassay et al. (2016) explique qu'il est compliqué d'étudier la fiabilité de systèmes complexes constitués d'un grand nombre de composants et propose d'utiliser les BDDs afin de sauvegarder des informations à propos de la topologie du système de manière efficace.

Tang et Dugan (2004); Rauzy (2011) proposent le diagramme de décision séquentiel (SDD). Cette extension des BDDs considère les séquences plutôt que les coupes (ou liens), afin de traiter les systèmes pour lesquels l'ordre des défaillances influe.

Néanmoins, il ne faut pas négliger l'importance de l'ordre utilisé pour les composants. La taille d'un BDD est très dépendante de l'ordre dans lequel les composants sont placés dans le BDD. La Figure 1.10 contient deux BDDs représentant le même système, dont seul l'ordre des variables change, et dont la fonction de structure est la suivante :

$$f = x_3 \vee (x_1 \wedge x_2).$$

En résumé, les BDDs sont très dépendants de l'ordre dans lequel on prend les variables et des chercheurs travaillent à l'obtention d'heuristiques pour déterminer l'ordre optimal dans lequel prendre les variables.

L'obtention d'un ordre optimal est un problème NP-complet (Costa et al. (2000)) et il n'a pas de sens de trouver un tel ordre avant de déterminer la satisfiabilité.

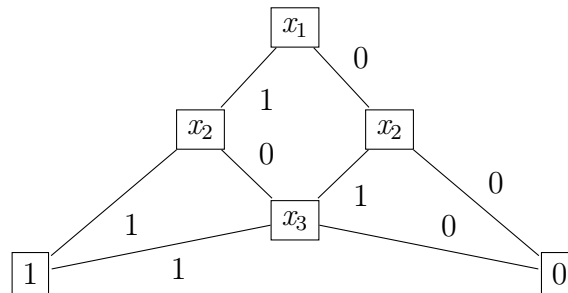


FIGURE 1.9 – BDD réduit d'un système de trois pompes en redondance 2oo3.

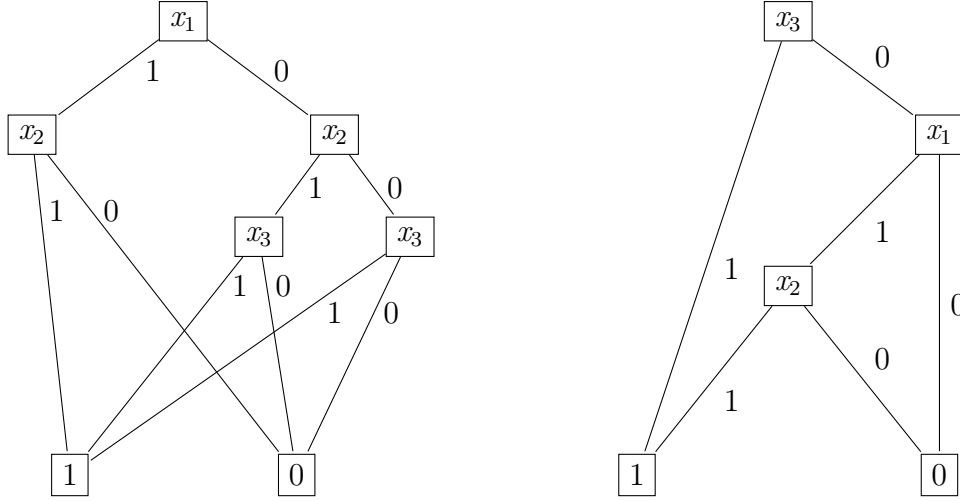


FIGURE 1.10 – BDD et BDD réduit d'un même système.

Diagramme de Hasse

Une quatrième approche a été proposée par Brânzei et Aubry (2015), basée sur le diagramme de Hasse, du nom du mathématicien allemand Helmut Hasse. Un diagramme de Hasse est un treillis, c'est-à-dire une structure algébrique, permettant de représenter un ordre fini. Il est introduit dans le domaine de la fiabilité par Matousek et Nesetril (1998)) et il se compose d'éléments ordonnés, nommés les *noeuds* du diagramme. La relation entre deux éléments est représentée par un segment entre deux noeuds qui est appelé *arc*.

Deux noeuds G_1 et G_2 sont comparables si chaque élément de G_1 est inférieur (ou égal) à l'élément correspondant de G_2 : $g_{1i} \leq g_{2i}$. Cette relation d'ordre entre les noeuds est partielle car tous les noeuds ne sont pas comparables.

Brânzei et Aubry (2015) proposent de mettre à profit les diagrammes de Hasse pour représenter les systèmes statiques afin de déterminer le polynôme de fiabilité. Pour ce faire, les liens minimaux sont nécessaires pour enrichir le diagramme de Hasse avec l'état du système et le transformer ainsi dans un graphe d'états. Cela se fait en exploitant la relation d'ordre entre les noeuds et l'hypothèse de la cohérence du système. Par cette transformation du diagramme de Hasse en graphe d'états⁵, nous uniformisons également la représentation des systèmes combinatoires (classiquement représentés par des diagrammes de fiabilité et arbres de défaillances) sous forme d'automates à états finis comme pour les systèmes séquentiels.

Un noeud contient autant de booléens que le système contient de variables ; un "1" représentant un composant fonctionnel et un "0" un composant défaillant. L'ordre d'un noeud est le nombre de "1" qui le composent. Il correspond à la position verticale du noeud dans le diagramme de Hasse.

La Figure 1.11 représente le diagramme de Hasse d'un système de trois composants

5. Par simplicité, nous ferons toujours référence en le nommant diagramme de Hasse.

en redondance 2oo3.

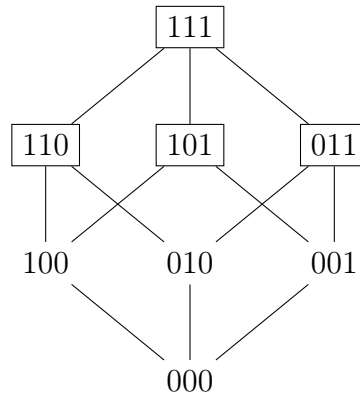


FIGURE 1.11 – Diagramme de Hasse d'un système de trois pompes en redondance 2oo3.

Un *chemin* est une succession verticale d'arcs (vers le bas) composé d'un ou plusieurs arcs, allant d'un noeud à un autre. Sur la Figure 1.11, il existe deux chemin entre les noeuds $\langle 111 \rangle$ et $\langle 001 \rangle$ qui correspond à la défaillance des composants C_1 et C_2 .

Pour différencier les liens des coupes, les liens sont représentés par des noeuds encadrés et les coupes par des noeuds non encadrés.

Les avantages de cette approche résident dans le fait qu'elle ne nécessite pas d'approximation lors de l'évaluation probabiliste d'un système comme le font les simulations de Monte Carlo, ni de déterminer un ordre dans lequel utiliser les composants comme c'est le cas pour les BDDs. L'inconvénient de cette approche est qu'elle n'est pas concise, le nombre d'états du diagramme de Hasse croissant de façon exponentielle par rapport au nombre de composants du système. Toutefois, le diagramme de Hasse peut être obtenu de manière automatisée dès lors que le nombre de composants du système est connu.

Brameret (2015) propose d'automatiser la génération de chaînes de Markov partielles avec un haut niveau de description. Pour ce faire, ils s'appuient sur une variation de l'algorithme de Dijkstra afin d'identifier les liens minimaux. Cette méthode a pour objectif de solutionner deux limites des chaînes de Markov. Le premier consiste à éviter la construction manuelle des chaînes de Markov, qui est un travail fastidieux et source d'erreur. Le second est de dépasser l'explosion combinatoire de la taille de la chaîne de Markov obtenue. La chaîne de Markov partielle représente une fraction de la chaîne de Markov. Grâce à celle-ci, Brameret et al. (2015) réalisent une estimation de la défiabilité avec une erreur relative de moins de 0,25%. Ainsi, ces chaînes de Markov partielles deviennent des outils pertinents pour la modélisation de systèmes avec un nombre important de composants lorsque l'on souhaite une estimation de la fiabilité ou défiabilité.

Analyse de fiabilité par calcul différentiel

Enfin, Zaitseva et Levashenko (2013); Kvassay et al. (2014) proposent l'utilisation

du calcul logique différentiel pour calculer les coupes minimales. Ils nomment *vecteurs de coupes minimales* le mot booléen constitué des états de chaque composant, ils correspondent aux noeuds des diagrammes de Hasse. Puisque ces travaux considèrent des systèmes multi-états, ils adaptent l'approche par BDDs et utilisent des diagramme de décision multi-état (Multiple-Valued Decision Diagrams ou MDDs) qui permettent de représenter la fonction de structure du système multi-état de manière efficace, y compris pour les systèmes de grande dimension.

L'objectif de l'utilisation du calcul différentiel est de déterminer l'importance individuelle de chaque composant du système et d'identifier quelles défaillances de composant conduisent à la défaillance du système. Proposée par Tapia et al. (1991), la dérivation logique partielle directe (Direct Partial Logic Derivatives ou DPLD) utilisée permet d'évaluer l'influence sur le système du changement d'état de chaque composant.

1.5.2.2 Approches par simulation (simulation de Monte Carlo)

L'évaluation probabiliste de la fiabilité d'un système est également effectuée en utilisant des approches par simulation.

Le model checking statistique utilise des simulations de Monte Carlo. La simulation de Monte Carlo est une méthode de résolution de problèmes d'ingénierie, utilisée dans de nombreux domaines industriels. Elle permet d'obtenir des estimations aux problèmes mathématiques grâce à des simulations stochastiques. Elle est très utilisée pour l'analyse quantitative des arbres de défaillances dynamiques, tout particulièrement pour les systèmes à grande échelle. La simulation de Monte-Carlo demeure chronophage, mais cela reste moindre comparé au model checking numérique.

Elle permet d'estimer des indicateurs de fiabilité en simulant leur processus actuel et des comportements aléatoires dans un modèle. Les informations nécessaires sont :

- la fonction de densité de probabilité de la défaillance de chaque composant et la valeur de leurs paramètres,
- la durée de la mission du système, et
- les modes de défaillance du système.

Arbres de défaillances dynamiques

Gascard et Simeu-Abazi (2018) proposent de simuler les arbres de défaillances dynamiques par un simulateur d'événements (*event-driven*) basé sur la simulation de Monte-Carlo, qui permet de traiter toutes les distributions de défaillance et n'est pas limitée à la représentation DFT : elle considère les DFTs avec des événements répétés et partagés, en prenant en compte les portes dynamiques PAND, SEQ, FDEP, et SPARE.

Merle et al. (2016) proposent une approche combinant les avantages de deux approches : par la fonction de structure (Merle et al. (2014)) et par simulation de Monte

Carlo. Ils réalisent des simulations de Monte Carlo sur les séquences de coupes minimales extraites de la fonction de structure et non sur le DFT, en utilisant la fonction de structure comme entrée pour la simulation de Monte Carlo. Ils s'appuient sur ses travaux précédents (Merle et al. (2014)) où ils calculent la forme canonique minimale de la fonction de structure à partir d'un DFT, et à partir de laquelle il détermine les séquences de coupes minimales. La simulation de Monte Carlo est ensuite utilisée pour générer des scénarios aléatoires.

Réseaux de Petri stochastiques

Les réseaux de Petri (RdP) sont des modèles états-transitions, avec un pouvoir d'expression plus riche que les automates à états finis (Murata (1989)). Les réseaux de Petri stochastiques (RdPS) sont des réseaux de Petri dont les temps de franchissement des transitions sont générés par des variables aléatoires de distributions quelconques dans $[0, \infty[$ (Leveson et Stolzy (1987); Chiola et al. (1993); Aubry et al. (2016)). Leur utilisation permet de modéliser des tâches dont les temps d'exécution sont non-déterministes ou des événements dont leur occurrence nécessite une durée aléatoire, elle permet également d'évaluer quantitativement les séquences d'événements dans les études de sûreté de fonctionnement. Par ailleurs, cette modélisation permet également d'évaluer des indicateurs plus classiques, tels que la fiabilité, la disponibilité, MTTF, MUT, etc. Bien que les RdPS soient équivalents à des processus stochastiques tels que les chaînes de Markov, les processus semi-markoviens et leurs extensions et que, par conséquent, une solution exacte existe, lorsque l'hypothèse markovienne est vérifiée, dans la plupart des cas, la simulation de Monte Carlo est utilisée pour évaluer les performances de sûreté de fonctionnement, car celle-ci permet de s'affranchir de l'hypothèse markovienne. Néanmoins, les RdPS ne permettent pas de représenter le fonctionnement (ou la défaillance) du système par une fonction propositionnelle.

Model-checking de systèmes probabilistes : méthodes statistiques

Les méthodes statistiques (Legay et al. (2010)) se basent sur des simulations de Monte Carlo pour obtenir une estimation statistique du résultat souhaité. Leur avantage majeur est la bien plus grande liberté sur la richesse du modèle considéré et sa taille. L'inconvénient est que les résultats ne sont plus exacts, mais donnés avec un certain degré de confiance.

L'outil le plus connu de model-checking probabiliste utilisant des méthodes numérique est PRISM créé par Kwiatkowska et al. (2009). Il propose aussi un module de model checking statistique. Le model-checker temporisé UppAal a également un module de model-checking statistique, UppAal SMC, qui donne une sémantique probabiliste aux systèmes temporisés (David et al. (2015)).

Certains outils de model checking numérique, comme PRISM, ont été étendus à des

fonctionnalités de vérification statistique. Toutefois, afin de traiter le même type de modèles avec les deux méthodes de vérification, deux restrictions majeures s'imposent : la limitation à des systèmes markoviens et l'expressivité restreinte de la logique utilisée pour exprimer l'affirmation à vérifier. Le model checker statistique Cosmos évalue des indicateurs du système basés sur des automates hybrides linéaires et des formules stochastiques (Ballarini et al. (2011)).

Younes et al. (2006) proposent de comparer les méthodes numériques et statistiques.

1.6 Conclusion

Étant donné que la représentation par arbres de défaillances est couramment utilisée et qu'elle permet de générer la fonction de structure représentant le comportement du système, nous modélisons les systèmes par des arbres de défaillances et nous considérons les portes statiques ET, OU et KooN et dynamiques FDEP, PAND et SPARE.

Nous nous intéressons aux systèmes dont les composants sont supposés indépendants et avec des distributions continues pour la durée de vie avant la défaillance, de telle sorte que la probabilité de défaillances simultanées de composants élémentaires est nulle. Certaines défaillances de composants peuvent néanmoins se produire simultanément lorsqu'un arbre de défaillances contient une porte FDEP qui provoque des défaillances simultanées.

Les systèmes et les composants sont supposés non réparables. Une seule occurrence est donc attribuée à chaque système et composant, correspondant à sa défaillance et il est possible d'attribuer un temps unique d'occurrence à chacun (Stamatelatos et al. (2002); Merle et al. (2014)).

Notre approche est basée sur l'usage de la fonction de structure, obtenue à partir de l'arbre de défaillances, à laquelle nous appliquons des techniques de satisfiabilité. L'objectif est de déterminer l'état du système en fonction de l'état de ses composants (fonction qui est introduite par la suite dans le cas d'un système dynamique). Dans le cas d'un système statique, nous identifions les liens minimaux du système; tandis que dans le cas d'un système dynamique, nous déterminons les séquences de liens minimales, qui correspondent à l'extension des liens minimaux aux systèmes dynamiques (prenant en compte les états des composants et l'ordre des défaillances).

Nous cherchons à évaluer la fiabilité d'un système de manière exacte, sans avoir recours à des simulations de Monte Carlo. Les diagrammes de Hasse ont l'avantage de permettre une évaluation probabiliste de la fiabilité du système sans nécessiter d'approximations comme les simulations de Monte Carlo. De plus il ne nécessitent pas d'optimisation comme c'est le cas pour les BDDs. Le diagramme de Hasse est un outil conceptuel et son utilisation nécessite de le déterminer, mais il n'est pas nécessaire de le représenter (au moins dans le cas de systèmes statiques). L'identification des liens minimaux du système au préalable

est cependant nécessaire pour utiliser un diagramme de Hasse.

D'une part, les liens minimaux obtenus grâce aux techniques de satisfiabilité donnent lieu une analyse qualitative du système. D'autre part, ils sont nécessaires à la construction du diagramme de Hasse afin de procéder à l'obtention du polynôme de fiabilité du système, dans le cadre d'une analyse quantitative.

Nous considérons des systèmes statiques dans un premier temps et nous étendons par la suite nos travaux aux systèmes dynamiques, pour lesquels l'instant de défaillances des composants impacte le système, ainsi que les autres composants.

Chapitre 2

Étude des systèmes modélisés par arbres de défaillances statiques grâce aux techniques de satisfiabilité

2.1 Introduction

Les arbres de défaillances sont des outils de représentation graphique permettant de représenter les systèmes de manière concise et dont est extraite la fonction de structure, qui modélise le système par une formule logique.

Le premier objectif de ce chapitre est de proposer une approche permettant de déterminer une représentation propositionnelle de la fonction de structure d'un arbre de défaillance statique. Le second objectif est de faire appel à un solveur de satisfiabilité auquel est envoyé la fonction de structure et de déduire des modèles générés les liens minimaux, qui permettent de réaliser une analyse qualitative du système. Nous en déduisons également la position des liens minimaux dans le diagramme de Hasse, ce qui nous permet de déterminer le polynôme de fiabilité du système en vue d'une analyse quantitative du système. Nous proposons d'illustrer la suite de notre propos par un exemple.

Exemple 2.1. Considérons le système, dont la Figure 2.1 est l'arbre de défaillances.

L'acronyme ER signifie "*évènement redouté*" et représente la défaillance du système. Il constitue le sommet de l'arbre de défaillances. L'arbre de défaillances qui modélise le système contient trois portes et cinq composants :

- les composants C_1 et C_2 sont les entrées d'une porte OU ;
- les composants C_3 , C_4 et C_5 sont les entrées d'une porte 2-parmi-3 (2oo3) qui s'active lorsqu'au moins 2 entrées s'activent parmi 3 ; et
- les portes OU et 2oo3 sont les entrées d'une porte ET.

L'évènement redouté se produit lorsqu'un composant parmi C_1 et C_2 défaille et lorsque deux composants défaillent parmi les C_3 , C_4 et C_5 .

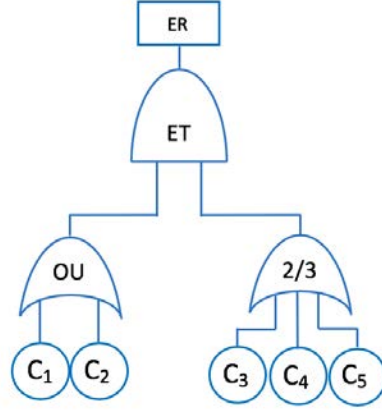


FIGURE 2.1 – Arbre de défaillances statique.

Les travaux présentés dans ce chapitre sont décomposés de la façon suivante :

- La section 2.2 définit les notions sur lesquelles s’appuie notre travail.
- La section 2.3 présente les portes statiques et propose une méthodologie permettant d’écrire la fonction de structure, en fonction des portes de l’arbre de défaillances.
- La section 2.4 présente le diagramme de Hasse et met en évidence le lien entre ce diagramme et la relation d’ordre entre les configurations.
- Dans la section 2.5, nous présentons trois algorithmes permettant d’obtenir les liens minimaux selon la représentation syntaxique de la fonction de structure, dont un algorithme utilisant les techniques de satisfiabilité.
- La section 2.6 développe la méthode d’obtention du polynôme de fiabilité grâce au diagramme de Hasse. Nous présentons d’abord la construction du diagramme de Hasse, puis l’obtention du polynôme de fiabilité et enfin le calcul de la fiabilité.
- La section 2.7 illustre notre approche avec un système d’assistance cardiaque et réalise ensuite une comparaison de nos résultats en vue de leur validation.
- Enfin, la section 2.8 conclut et propose d’étendre cette approche.

2.2 Définitions

Cette partie définit les notions utilisées dans notre approche pour modéliser un système statique avec un comportement binaire.

Définition 2.2. Un *système* est une entité constituée de n composants C_i , $i \in \{1, 2, \dots, n\}$.

Le comportement d’un système est dit *binaire* si l’état du système et l’état de ses composants n’ont que deux valeurs (marche et panne) et peuvent être représentés par une variable booléenne.

Définition 2.3. L’état d’un composant C_i est représenté par la variable booléenne c_i , $c_i \in \mathbb{B} = \{0, 1\}$, et l’état du système est représenté par la variable booléenne. Leur état est assignée à 0 (respectivement 1) lorsqu’ils fonctionnent (resp. défaillent).

Par la suite nous identifions 0 à faux et 1 à vrai et nous admettons que les booléens sont ordonnés par la relation d'ordre tel que $0 < 1$.

Définition 2.4. Une *configuration* est un n -uplet représenté par le mot booléen

$$G = \langle c_1, \dots, c_n \rangle \in \mathbb{B}^n. \quad (2.1)$$

L'ensemble des configurations du système est doté d'une relation d'ordre, dont la définition est donnée par Kaufmann et al. (1977).

Définition 2.5. Les configurations sont ordonnées par une relation d'ordre partiel représentée par le symbole \preceq . Considérons deux configurations G et H telles que

$$G = \langle G_1, \dots, G_n \rangle \text{ et } H = \langle H_1, \dots, H_n \rangle. \quad (2.2)$$

La configuration G est dominée par la configuration H , i.e. $G \preceq H$, si et seulement si

$$\forall i \in \{1, 2, \dots, n\}, G_i \leq H_i. \quad (2.3)$$

La relation d'ordre entre les configurations est *réflexive* car $G \preceq G$, *antisymétrique* car $G \preceq G' \wedge G' \preceq G \Rightarrow G = G'$, *transitive* car $G \preceq G' \wedge G' \preceq G'' \Rightarrow G \preceq G''$ et *partielle* car toutes les configurations ne peuvent pas être ordonnées deux à deux.

Définition 2.6. Un système est dit *statique* si l'état du système dépend uniquement de l'état de ses composants.

Remarque 2.7. L'état d'un système statique ne dépend ni du temps, ni de l'enchaînement des défaillances. Une notion plus fine de systèmes dynamiques sera vue au chapitre suivant.

L'état du système est déterminé à partir de l'état de ses composants grâce à la fonction de structure, qui est l'expression logique de l'évènement redouté.

Définition 2.8. La *fonction de structure* $f : \zeta \rightarrow \mathbb{B}$, où ζ est l'ensemble des configurations, associe un état du système à chaque configuration.

La fonction de structure peut être représentée par une *expression booléenne* qui se construit à partir des expressions booléennes c_γ des portes γ . Les expressions booléennes des portes ET, OU et KooN sont formulées dans la section 2.3.

Exemple 2.9. L'expression de l'évènement redouté de l'exemple 2.1 est : $f = c_{OU} \wedge c_{2oo3}$.

Définition 2.10. Une fonction de structure f est dite *monotone* si et seulement si

$$\forall G, G' \in \zeta, G \preceq G' \Rightarrow f(G) \leq f(G'), \quad (2.4)$$

où G et G' sont des configurations du système (Gnedenko et al. (1995)).

Par conséquent, la défaillance d'un composant ne peut pas remettre en fonctionnement un système défaillant. Ainsi, une défaillance peut provoquer une défaillance du système ou ne pas impacter le système.

Définition 2.11. Un système est dit *cohérent* (Biolini (1999)) si :

- La fonction de structure du système dépend de toutes les variables c_i et
- La fonction de structure est monotone.

Lorsqu'une défaillance peut remettre en fonctionnement un système, le système est dit *non cohérent*. Dans ce cas d'un système cohérent, il n'existe aucun état du système tel, qu'à partir de cet état, la défaillance d'un composant puisse réparer le système.

L'étude de la fonction de structure permet d'identifier les liens et les coupes du système.

Définition 2.12. Un *lien* est un ensemble de composants dont le bon fonctionnement garantit que le système fonctionne.

Si une configuration représente un état de fonctionnement, alors le lien représentant cette configuration est l'ensemble des composants dont la variable booléenne vaut 0 dans le n-uplet de la configuration.

Exemple 2.13. Dans l'exemple 2.1, si le composant C_1 a défailli et que les composants C_2 , C_3 , C_4 et C_5 fonctionnent, alors le système fonctionne. Par conséquent, la configuration $\langle 1, 0, 0, 0, 0 \rangle$ correspond à un état de fonctionnement du système et $\{C_2, C_3, C_4, C_5\}$ est un lien.

Définition 2.14. Un lien est *minimal* lorsqu'il n'existe aucun lien qui correspond un sous-ensemble de celui-ci.

Exemple 2.15. Dans l'exemple 2.1, les liens du système sont : $\{C_1, C_2, C_3, C_4, C_5\}$, $\{C_2, C_3, C_4, C_5\}$, $\{C_1, C_3, C_4, C_5\}$, $\{C_1, C_2, C_4, C_5\}$, $\{C_1, C_2, C_3, C_5\}$, $\{C_1, C_2, C_3, C_4\}$, $\{C_3, C_4, C_5\}$, $\{C_3, C_4\}$, $\{C_3, C_5\}$, $\{C_4, C_5\}$ et $\{C_1, C_2\}$.

Or, $\{C_1, C_2\} \preceq \{C_1, C_2, C_4, C_5\}$; $\{C_3, C_5\} \preceq \{C_3, C_4, C_5\}$; $\{C_4, C_5\} \preceq \{C_3, C_4, C_5\}$ et $\{C_3, C_4\} \preceq \{C_3, C_4, C_5\} \preceq \{C_1, C_2, C_4, C_5\} \preceq \{C_1, C_2, C_3, C_4, C_5\}$.

Les liens minimaux du système sont donc $\{C_3, C_4\}$, $\{C_3, C_5\}$, $\{C_4, C_5\}$ et $\{C_1, C_2\}$. \square

Nous définissons ensuite la coupe comme le dual du lien.

Définition 2.16. Une *coupe* est un ensemble de composants dont la défaillance assure que le système défaille.

Exemple 2.17. Dans l'exemple 2.1, les coupes du système sont : $\{C_1, C_2, C_3, C_4, C_5\}$, $\{C_1, C_2, C_3, C_4\}$, $\{C_1, C_2, C_3, C_5\}$, $\{C_1, C_2, C_4, C_5\}$, $\{C_1, C_3, C_4, C_5\}$, $\{C_2, C_3, C_4, C_5\}$, $\{C_1, C_3, C_4\}$, $\{C_1, C_3, C_5\}$, $\{C_1, C_4, C_5\}$, $\{C_2, C_3, C_4\}$, $\{C_2, C_3, C_5\}$, $\{C_2, C_4, C_5\}$. \square

Définition 2.18. Un coupe est *minimale* lorsqu'il n'existe aucune coupe qui soit un sous-ensemble de celle-ci.

Exemple 2.19. Les coupes minimales du système de l'exemple 2.1 sont donc $\{C_1, C_3, C_4\}$, $\{C_2, C_3, C_4\}$, $\{C_1, C_3, C_5\}$, $\{C_2, C_3, C_5\}$, $\{C_1, C_4, C_5\}$, $\{C_2, C_4, C_5\}$. \square

La suppression d'un seul composant d'une configuration représentant un lien (resp. une coupe) minimal(e) crée une configuration représentant une coupe (resp. un lien).

Cette relation d'ordre entre liens et entre coupes peut être illustrée par un treillis afin d'en faciliter la compréhension. Le diagramme de Hasse, que nous introduisons dans la section 2.4, est une représentation graphique d'un ordre fini qui permet de mettre en évidence cette relation d'ordre.

Par abus de notation, nous utiliserons la notation $x \preceq t$, où t est un lien et x est une conjonction de littéraux, sous-entendu le lien ou la coupe qui correspond aux composants dont les littéraux sont dans la conjonction x .

Par la suite, nous considérons uniquement des systèmes cohérents, non réparables et dont le comportement est binaire. Notre objectif étant de déterminer la fonction de structure du système, grâce à l'arbre de défaillances, dans la section 2.3, nous considérons les portes ET, OU et KooN et modélisons leur activation par une expression booléenne.

2.3 Fonction de structure

2.3.1 Expression booléenne des portes

Nous présentons les trois portes statiques permettant de construire un arbre de défaillances statique. Elles correspondent à la conjonction, à la disjonction et à K-parmi-N (K -out-of- N). La Figure 2.2 donne une représentation graphique de ces trois portes.

Porte ET Illustrée sur la Figure 2.2.a, la porte *ET* s'active lorsque toutes ses entrées s'activent et son expression booléenne est donnée par l'équation (2.5). Nous généralisons cette expression à k entrées par l'équation (2.6).

$$c_{ET} = c_1 \wedge c_2 \quad (2.5)$$

$$c_{ET} = \bigwedge_{i \in \{1, \dots, k\}} c_i, \quad (2.6)$$

Porte OU La porte *OU* s'active lorsqu'au moins une de ses entrées s'active. L'équation (2.7) fournit l'expression booléenne de la porte OU de la Figure 2.2.b. La généralisation à k entrées est donnée par l'équation (2.8).

$$c_{OU} = c_1 \vee c_2 \quad (2.7)$$

$$c_{OU} = \bigvee_{i \in \{1, \dots, k\}} c_i, \quad (2.8)$$

Porte KooN La porte *KooN* s'active lorsqu'au moins K entrées parmi N s'activent. Illustrée par la Figure 2.2.c, la porte *2-parmi-3* (2oo3) s'active lorsqu'au moins 2 entrées s'activent parmi 3. Son expression booléenne est une disjonction de conjonctions, elle est donnée par l'équation (2.9).

$$c_{2oo3} = (c_1 \wedge c_2) \vee (c_1 \wedge c_3) \vee (c_2 \wedge c_3) \quad (2.9)$$

Nous proposons de généraliser l'expression de cette porte à des instances à N entrées.

Pour définir formellement l'expression booléenne d'une porte KooN, nous considérons l'ensemble $\mathcal{N} \subset \mathcal{J}$, de tous les composants en entrée de la porte KooN, où \mathcal{J} désigne l'ensemble des composants du système et c_i est la variable booléenne du composant C_i .

La cardinalité de l'ensemble \mathcal{N} est notée $N = |\mathcal{N}|$. $[\mathcal{N}]^K$ est l'ensemble des sous-ensembles de cardinalité K et \mathcal{K} est l'un de ces sous-ensembles.

Afin d'étudier la porte KooN, nous considérons l'ensemble des sous-ensembles de \mathcal{N} avec une cardinalité égale à K .

L'expression booléenne de la porte KooN est donnée par l'équation (2.10).

$$c_{KooN} = \bigvee_{\mathcal{K} \in [\mathcal{N}]^K} \bigwedge_{i \in \mathcal{K}} c_i, \quad (2.10)$$

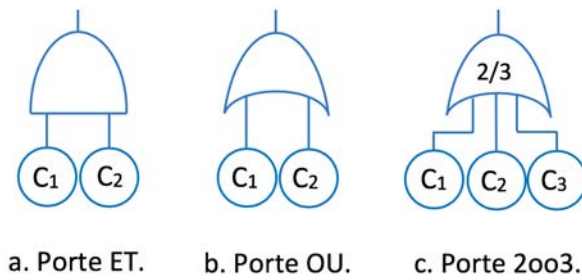


FIGURE 2.2 – Portes statiques.

Exemple 2.20. Reprenons l'exemple 2.1. En utilisant les équations (2.5), (2.7) et (2.9), nous obtenons les expressions booléennes des trois portes (équations (2.11)-(2.13)).

Nous déterminons ensuite la fonction de structure en combinant les expressions booléennes des portes de l'arbre de défaillances et nous obtenons l'équation (2.14).

$$f = c_{OU} \wedge c_{2oo3} \quad (2.11)$$

$$c_{OU} = c_1 \vee c_2 \quad (2.12)$$

$$c_{2oo3} = (c_3 \wedge c_4) \vee (c_3 \wedge c_5) \vee (c_4 \wedge c_5) \quad (2.13)$$

$$f = (c_1 \vee c_2) \wedge ((c_3 \wedge c_4) \vee (c_3 \wedge c_5) \vee (c_4 \wedge c_5)) \quad (2.14)$$

2.3.2 Dualisation de la fonction de structure

La définition de la fonction de structure associe, à une configuration, une variable booléenne c qui correspond à l'occurrence de l'évènement redouté. Étant donné que la fonction de structure exprime la défaillance du système, une assignation de valeurs aux variables c_i qui rende vraie la fonction de structure serait une coupe.

Afin de réaliser le diagramme de Hasse, nous cherchons à obtenir les liens du système. Puisque le lien est le dual de la coupe, l'obtention des liens nécessite de considérer le dual de la fonction de structure, qui correspond au complément de la fonction de structure :

$$\bar{f} = \neg f.$$

Les littéraux de la fonction de structure construits à partir de c_i sont donc remplacés par leurs compléments, que nous nommerons x_i :

$$x_i = \neg c_i$$

où le composant C_i fonctionne lorsque $x_i = 1$ et défaille lorsque $x_i = 0$.

Remarque 2.21. La relation d'ordre définie précédemment s'applique sur les variables booléennes et non sur leur interprétation (marche ou panne). Avant la dualisation, la supériorité d'une configuration par rapport à une autre correspond au fait que plus de composants aient défailli. Après la dualisation, la supériorité d'une configuration par rapport à une autre correspond au fait que plus de composants soient fonctionnels.

Exemple 2.22. Pour illustrer la dualisation de la fonction de structure, nous considérons le système de l'exemple 2.20, dont l'expression booléenne est donnée par l'équation (2.14). L'expression booléenne dualisée de ce système est donnée par l'équation (2.15).

$$\neg f = (x_1 \wedge x_2) \vee ((x_3 \vee x_4) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5)). \quad (2.15)$$

Nous remarquons que la porte 2oo3 est "symétrique" puisqu'elle s'active si deux entrées sur trois s'activent et elle ne s'active pas si deux composants sur trois ne s'activent pas.

□

Dans la suite de ce chapitre, nous utilisons uniquement la fonction de structure dualisée et la variable x_i plutôt que son complément c_i .

2.4 Diagramme de Hasse d'un système statique

Pour représenter les configurations et la relation d'ordre qui les régit, nous modélisons le système par un diagramme de Hasse, qui permet de tirer profit de l'ordre partiel régissant l'ensemble des configurations et de la monotonie de la fonction de structure, qui cause par la cohérence du système.

Le diagramme de Hasse se construit à partir des variables x_i .

Définition 2.23. Le *diagramme de Hasse* d'un système à n composants est un graphe ordonné contenant des éléments ordonnés, nommés *noeuds*, et la relation entre deux éléments est représentée par un segment entre deux points, nommé *arc*.

Un noeud représente un état du système et le diagramme de Hasse contient 2^n noeuds.

Exemple 2.24. Un exemple d'arbre de défaillances est représenté sur la Figure 2.3 et son diagramme de Hasse est représenté sur la Figure 2.4 adjacente.

Remarque 2.25. Certains noeuds du diagramme de Hasse sont encadrés afin de différencier les états de fonctionnement et de défaillance du système. Un noeud encadré représente un état de fonctionnement et un noeud non encadré représente un état de défaillance.

Un *chemin* est une succession verticale d'arcs (vers le bas) composé d'un ou plusieurs arcs, allant d'un noeud à un autre. Lorsque d'un noeud à un autre, il n'existe ni arc, ni chemin, alors les deux configurations associées à ces noeuds ne sont pas comparables.

Définition 2.26. Un noeud est dit *supérieur* (resp. *inférieur*) à un autre s'il existe un chemin allant de l'un à l'autre.

Quant à un arc, d'un noeud vers un noeud inférieur, il représente la défaillance d'un composant.

Définition 2.27. Un noeud G_1 est dit « couvert » par un noeud G_2 si G_1 est supérieur à G_2 . On dit de G_2 qu'il couvre G_1 ($G_2 \preceq G_1$).

Exemple 2.28. Sur la Figure 2.4, le noeud $\langle 010 \rangle$ couvre le noeud $\langle 011 \rangle$.

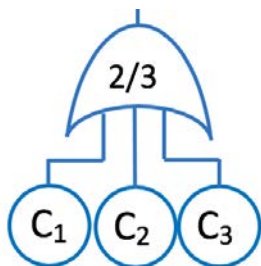


FIGURE 2.3 – Portes 2oo3.

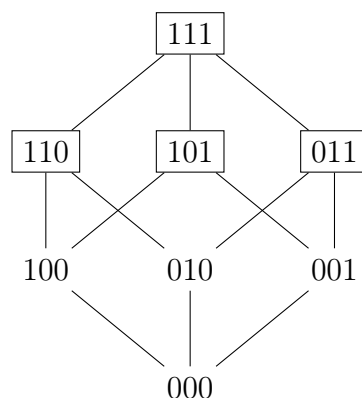


FIGURE 2.4 – Diagramme de Hasse.

Remarque 2.29. Il y a autant d'arcs allant d'un noeud A vers des noeuds inférieurs que de composants fonctionnels dans la configuration du noeud A .

Définition 2.30. L'*ordre* d'un noeud est le nombre de 1 qui composent le mot booléen de cette configuration.

Exemple 2.31. Sur la Figure 2.4, l'ordre du noeud $\langle 110 \rangle$ vaut 2.

Définition 2.32. Un *maximum* d'un graphe ordonné est un noeud dont le n-uplet domine tous les n-uplets du graphe. Un *minimum* d'un graphe ordonné est un noeud dont le n-uplet ne domine aucun n-uplet du graphe.

Le maximum d'un diagramme de Hasse est le sommet de ce diagramme de Hasse, qui correspond à l'état du système où tous les composants sont fonctionnels.

Définition 2.33. Le *diagramme de Hasse restreint aux états de fonctionnement du système* contient tous les noeuds du diagramme de Hasse dont la configuration représente un état de fonctionnement du système.

Le diagramme de Hasse restreint aux états de fonctionnement du système possède un maximum et plusieurs minima, qui sont les noeuds des liens minimaux du système.

Exemple 2.34. Si nous considérons le diagramme de Hasse restreint aux états de fonctionnement de la Figure 2.4, nous obtenons la Figure 2.5. Le noeud $\langle 111 \rangle$ est un maximum et les noeuds $\langle 110 \rangle$, $\langle 101 \rangle$ et $\langle 011 \rangle$ sont des minima.

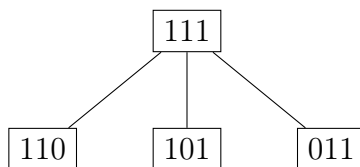


FIGURE 2.5 – Diagramme de Hasse.

2.5 Calcul des liens minimaux

Afin de réaliser des études qualitative et quantitative des systèmes, nous souhaitons calculer les liens minimaux de ces systèmes à partir de leur fonction de structure. Pour ce faire, nous présentons trois algorithmes qui permettent de calculer les liens minimaux d'un système à partir de sa fonction de structure, selon sa syntaxe, ou à partir des coupes minimales du système.

2.5.1 Calcul des liens minimaux à partir d'un format DNF

La forme normale disjonctive (DNF) est une disjonction de conjonctions de littéraux. Lorsque la fonction de structure f est au format DNF, $f = \bigvee_{i=1}^p \bigwedge_{j=1}^{q_i} l_{ij}$,¹ chaque terme $\bigwedge_{j=1}^{q_i} l_{ij}$ correspond à un lien $\langle C_1, \dots, C_{q-1} \rangle$ où $l_{ij} = c_k$ implique que le composant C_k fonctionne. Pour tout lien t du système, il existe une conjonction de littéraux x dans la représentation DNF, telle que x couvre t : $x \preceq t$.

Il suffit ensuite de déterminer les liens minimaux parmi les liens obtenus. Pour cela, nous créons un ensemble Tie auquel nous ajoutons les liens obtenus à partir de la fonction de structure, afin que chaque lien $t \in Tie$ soit comparé aux autres liens de l'ensemble Tie . S'il existe un lien $t' \in Tie$ tel que $t' \preceq t$, alors le lien t n'est pas minimal et nous le supprimons de l'ensemble Tie . Au contraire, si nous identifions un lien $t' \in Tie$ tel que $t \preceq t'$, alors le lien t' n'est pas minimal et nous le supprimons de l'ensemble Tie . Enfin, il est possible que certains liens ne soient pas comparables selon la relation d'ordre \preceq et cela n'entraîne aucune suppression de lien dans l'ensemble Tie .

Nous procédons de cette façon avec tous les liens de l'ensemble Tie afin qu'il contienne uniquement les liens minimaux du système. Notre démarche est illustrée par l'algorithme 1.

Algorithm 1 Calcul des liens minimaux à partir de la fonction de structure dualisée au format DNF

Require: f : fonction de structure dualisée au format DNF

- 1: $Tie \leftarrow \{ \bigwedge_{j=1}^{q_i} l_{ij} : f = \bigvee_{i=1}^p \bigwedge_{j=1}^{q_i} l_{ij} \}$ {Initialisation}
 - 2: **for** $t, t' \in Tie$ **do**
 - 3: **if** $t' \preceq t$ **then**
 - 4: $Tie \leftarrow Tie \setminus \{t\}$
 - 5: **else if** $t \preceq t'$ **then**
 - 6: $Tie \leftarrow Tie \setminus \{t'\}$
 - 7: **end if**
 - 8: **end for**
 - 9: $MinTie \leftarrow Tie$
 - 10: **return** $MinTie$: ensemble des liens minimaux
-

1. Puisque nous considérons des systèmes cohérents, nous pouvons supposer que la fonction de structure contient uniquement des littéraux positifs.

Exemple 2.35. Prenons la fonction de structure dualisée suivante :

$$f = (c_1 \wedge c_2 \wedge c_4) \vee (c_3 \wedge c_4 \wedge c_5) \vee (c_3 \wedge c_5) \vee (c_4 \wedge c_5)$$

Cette fonction de structure dualisée est exprimée comme une disjonction de conjonctions et nous en déduisons les liens suivants :

$$Tie = \{\{C_1, C_2, C_4\}, \{C_3, C_4, C_5\}, \{C_3, C_5\}, \{C_4, C_5\}\}.$$

Nous comparons ensuite ces liens entre eux afin de trouver lesquels sont minimaux.

$$\{C_1, C_2, C_4\} \not\preceq \{C_3, C_4, C_5\}; \{C_1, C_2, C_4\} \not\preceq \{C_3, C_5\}; \{C_1, C_2, C_4\} \not\preceq \{C_4, C_5\}.$$

$$\{C_3, C_4, C_5\} \not\preceq \{C_1, C_2, C_4\}; \{C_3, C_5\} \not\preceq \{C_1, C_2, C_4\}; \{C_4, C_5\} \not\preceq \{C_1, C_2, C_4\}.$$

Puisque $\{C_1, C_2, C_4\}$ n'est comparable à aucun autre lien, alors c'est un lien minimal.

Par conséquent, $MinTie = \{\{C_1, C_2, C_4\}\}$ et $Tie = \{\{C_3, C_4, C_5\}, \{C_3, C_5\}, \{C_4, C_5\}\}$.

$$\{C_3, C_5\} \preceq \{C_3, C_4, C_5\}; \{C_4, C_5\} \preceq \{C_3, C_4, C_5\}.$$

Puisque $\{C_3, C_4, C_5\}$ est supérieur à $\{C_3, C_5\}$ et $\{C_4, C_5\}$, ce n'est pas un lien minimal.

Par conséquent, $MinTie = \{\{C_1, C_2, C_4\}\}$ et $Tie = \{\{C_3, C_5\}, \{C_4, C_5\}\}$.

$$\{C_3, C_5\} \not\preceq \{C_4, C_5\}; \{C_4, C_5\} \not\preceq \{C_3, C_5\}.$$

Puisque $\{C_3, C_5\}$ et $\{C_4, C_5\}$ sont non comparables entre eux, ce sont des liens minimaux.

Par conséquent, $MinTie = \{\{C_1, C_2, C_4\}, \{C_3, C_5\}, \{C_4, C_5\}\}$ et $Tie = \emptyset$.

Les liens minimaux de ce système sont donc : $\{C_1, C_2, C_4\}$, $\{C_3, C_5\}$ et $\{C_4, C_5\}$.

2.5.2 Calcul des liens minimaux à partir d'un format CNF

Bien que l'algorithme de la section 2.5.1 soit simple et efficace, il requiert que la fonction de structure soit au format DNF. Convertir une formule propositionnelle quelconque au format DNF mène en général à une explosion combinatoire (exponentielle) et il est préférable de ne pas y avoir recours.

Le dual de la DNF est la forme normale conjonctive (CNF), qui est une conjonction de disjonctions. Le format CNF est celui qu'attendent généralement les solveurs SAT et des algorithmes (Sheridan (2004)) permettent de convertir n'importe quelle formule en une formule CNF, dont la taille est linéaire en la taille de l'entrée, au coût d'ajout de variables supplémentaires. Pour une formule donnée au format CNF, les solveurs de satisfiabilité (SAT) emploient des algorithmes afin de déterminer un modèle (une assignation de valeurs) qui rend la formule vraie (si un modèle existe). Lorsqu'il n'existe pas de modèle qui rende vraie la formule, la formule n'est pas satisfiable et ils renvoient *UNSAT*.

Nous proposons d'employer un solveur SAT, comme MiniSat (Eén et Sörensson (2003)), pour produire un ensemble de liens Tie , afin que chaque lien x du système soit « couvert » par un lien de l'ensemble Tie : \forall un lien x , $\exists y \in Tie$ tel que $y \preceq x$.

L'algorithme de la section 2.5.1 est ensuite appliqué à l'ensemble Tie pour obtenir l'ensemble des liens minimaux $MinTie$, la disjonction des liens minimaux étant une DNF.

Dans le cadre d'une étude de fiabilité, la formule considérée est la fonction de structure du système. La fonction de structure f est envoyée au format CNF au solveur SAT afin qu'il détermine si f est satisfiable. Si tel est le cas, le solveur SAT produit un modèle de f , c'est-à-dire un ensemble de littéraux dont la conjonction implique f . À partir de ce modèle, nous déduisons un lien du système, en ne considérant que les littéraux positifs (puisque le système est supposé cohérent).

Lorsqu'un lien a été calculé, relancer le même calcul génèrerait le même modèle, et donc le même lien. Afin que le solveur génère un nouveau lien (s'il en existe un), nous modifions l'entrée envoyée² au solveur SAT, en lui ajoutant une clause. La clause ajoutée est la négation du lien précédemment obtenu, ce qui correspond à la disjonction de la négation des littéraux du modèle. La procédure est répétée jusqu'à ce que le solveur SAT établisse que la formule augmentée est insatisfiable.

Nous définissons quatre fonctions nécessaires à l'écriture de l'algorithme.

- La fonction *callsolver* fait appel au solveur SAT et renvoie le couple (R, M) où $R \in \{sat, unsat\}$ et M est le modèle généré si $R = sat$, $m = \emptyset$ sinon.
- La fonction *negate* prend un ensemble de littéraux en entrée et renvoie l'ensemble des littéraux complémentaires.
- La fonction *map* applique une fonction (*negate*) à chaque élément de l'ensemble nommé après cette fonction (M').
- La fonction *DNFalgorithm* applique l'algorithme 1 et renvoie les liens minimaux.
- La fonction *cleantie* détermine un lien à partir du modèle M . Ce lien n'est constitué que des composants fonctionnels du modèle. Nous nommons M' le lien obtenu en enlevant les littéraux négatifs de M .

Remarque 2.36. La clause ajoutée à la fonction de structure est la clause obtenue de $\neg M'$ en faisant passer la négation à l'intérieur de la formule.

Preuve Nous souhaitons prouver que les modèles produits par le solveur SAT "couvrent" tous les liens et par conséquent, que la minimisation produit tous les liens minimaux.

Soit $L = \{x_1, \dots, x_n\}$ un lien.

Nous voulons montrer qu'il est « couvert » par l'ensemble Tie , i.e. $\exists M \in Tie, M \subseteq L$.

Puisque le solveur SAT renvoie UNSAT lorsqu'il a généré tous les liens de Tie , alors la fonction booléenne associée à cette formule renvoie faux pour n'importe quelle valuation des variables qui y apparaissent, ce qui se traduit par l'équation (2.16).

$$f \wedge \neg M_1 \wedge \dots \wedge \neg M_m = \perp \quad (2.16)$$

2. La formule augmentée.

Algorithm 2 Calcul des liens minimaux à partir de la fonction de structure dualisée au format CNF

Require: f : fonction de structure dualisée représentée par un ensemble d'ensembles

```

1:  $Tie = \emptyset$ 
2:  $(R, M) \leftarrow callsolver(f)$ 
3: while  $R = SAT$  do
4:    $M' \leftarrow cleantie(M)$ 
5:    $Tie \leftarrow Tie \cup \{M'\}$ 
6:    $f \leftarrow f \cup \{\text{map negate } M'\}$ 
7:    $(R, M) \leftarrow callsolver(f)$ 
8: end while
9:  $MinTie \leftarrow DNFAalgorithm(Tie)$ 
10: return  $MinTie$  : ensemble de liens minimaux
```

Comme L est un lien de la fonction de structure f , alors la valuation associée à L , v_L , vérifie l'expression $f(v_L) = \top$. Ainsi, si nous appliquons l'équation (2.16) à la valuation de L , nous obtenons l'équation (2.17).

$$f(v_L) \wedge \neg M_1(v_L) \wedge \cdots \wedge \neg M_m(v_L) = \perp \quad (2.17)$$

Par conséquent, $\exists i$ tel que $\neg M_i(v_L) = \perp$ et par conséquent, $M_i(v_L) = \top$.

Soit $M_i = x_1 \wedge \cdots \wedge x_k$. Puisque $M_i(v_L) = \top$, alors $\forall x_j \in M_i, x_j \in L$.

Pour que $M_i(v_L)$ soit vrai, il faut que $\{x_j/j \in \{1, \dots, k\}\} \subseteq \{x_i \text{ tel que } v_L \text{ met } x_i \text{ à vrai}\}$, ce qui signifie que $M_i \subseteq L$.

Par conséquent, chaque lien du système est couvert par un lien de Tie . □

Exemple 2.37. Reprenons l'équation (2.15) de l'exemple 2.1. Nous utilisons la transformation de Plaisted Greenbaum puis développons la formule afin de la transformer en CNF.

$$\begin{aligned}
\neg f &= (x_1 \wedge x_2) \vee ((x_3 \vee x_4) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5)) \\
&= a \vee b \\
&\quad \wedge (\neg a \vee (x_1 \wedge x_2)) \\
&\quad \wedge (\neg b \vee ((x_3 \vee x_4) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5))) \quad (2.18)
\end{aligned}$$

$$\begin{aligned}
&= a \vee b \\
&\quad \wedge (\neg a \vee x_1) \\
&\quad \wedge (\neg a \vee x_2) \\
&\quad \wedge (\neg b \vee x_3 \vee x_4) \\
&\quad \wedge (\neg b \vee x_3 \vee x_5) \\
&\quad \wedge (\neg b \vee x_4 \vee x_5) \quad (2.19)
\end{aligned}$$

En appliquant l'algorithme 2, le premier modèle généré est $\{\neg C_1, C_2, C_3, \neg C_4, C_5, \neg a, b\}$. En supprimant les littéraux négatifs et les littéraux contenant a et b , nous obtenons le lien $\{C_2, C_3, C_5\}$.

Après modification de la fonction envoyée au solveur SAT, cette opération est répétée et nous obtenons l'ensemble des liens *Tie* :

$$\begin{aligned} Tie = \{ & \{C_2, C_3, C_5\}, \{C_1, C_2\}, \{C_1, C_2, C_3\}, \{C_1, C_2, C_4\}, \{C_1, C_2, C_3, C_4\}, \\ & \{C_1, C_2, C_5\}, \{C_1, C_2, C_4, C_5\}, \{C_1, C_2, C_3, C_5\}, \{C_1, C_2, C_3, C_4, C_5\}, \\ & \{C_3, C_5\}, \{C_1, C_3, C_5\}, \{C_4, C_5\}, \{C_2, C_4, C_5\}, \{C_3, C_4\}, \{C_3, C_4, C_5\}, \\ & \{C_2, C_3, C_4\}, \{C_2, C_3, C_4, C_5\}, \{C_1, C_3, C_4\}, \{C_1, C_4, C_5\}, \{C_1, C_3, C_4, C_5\} \} \end{aligned}$$

L'algorithme 1 détermine : $MinTie = \{\{C_1, C_2\}, \{C_3, C_4\}, \{C_3, C_5\}, \{C_4, C_5\}\}$. \square

2.5.3 Calcul des liens minimaux à partir des coupes minimales

Pour calculer la fiabilité d'un système, il est fréquent de construire un arbre de défaillances pour déterminer la fonction de structure et obtenir les coupes minimales. De la même façon qu'une représentation de la fonction de structure au format DNF est un ensemble de liens, la représentation au format CNF est un ensemble de coupes, auquel nous pouvons appliquer l'algorithme 1 afin d'identifier les coupes minimales.

Nous présentons un algorithme permettant de calculer les liens minimaux à partir des coupes minimales. Les entrées de l'algorithme sont la fonction de structure f du système et l'ensemble *MinCut* des coupes minimales. Il retourne l'ensemble des liens minimaux.

Nous définissons deux fonctions nécessaires à l'écriture de cet algorithme.

- La fonction *fathers* prend en entrée un noeud et renvoie l'ensemble des noeuds qui lui sont supérieurs.
- La fonction *sons* prend en entrée un noeud et renvoie l'ensemble des noeuds qui lui sont inférieurs.

L'algorithme 3 est basé sur la traversée du diagramme de Hasse, grâce au suivi des arcs. Nous observons que les « pères » des coupes minimales sont des liens (pas forcément minimaux). À partir de ces liens, l'algorithme détermine s'ils sont minimaux et sinon, il poursuit la traversée du diagramme de Hasse jusqu'à atteindre les liens minimaux. Il paraît intuitif que les liens minimaux soient (majoritairement) à une distance relativement faible des coupes minimales, et qu'il y ait peu d'arcs à suivre pour les atteindre.

Remarque 2.38. L'algorithme pourrait fonctionner en ne connaissant que l'ensemble des coupes minimales *MinCut* et déterminer, à partir de celui-ci, si un élément est un lien ou une coupe en vérifiant ceci : $\nexists c \in MinCut, s \preceq c$ (plutôt que de vérifier que $f(s) = 1$). Toutefois, il semble plus efficace d'utiliser la fonction de structure lorsque celle-ci est connue.

Algorithm 3 Calcul des liens minimaux à partir des coupes minimales

Require: f : fonction de structure

Require: $MinCut$: ensemble des coupes minimales

```
1:  $Tie \leftarrow \{fathers(c) : c \in MinCut\}$ 
2:  $MinTie \leftarrow \emptyset$ 
3: while  $Tie \neq \emptyset$  do
4:   take  $t \in Tie$ 
5:    $Tie \leftarrow Tie \setminus \{t\}$ 
6:    $ts \leftarrow \{s \in sons(t) : f(s) = 1\}$ 
7:   if  $ts = \emptyset$  then
8:      $MinTie \leftarrow MinTie \cup \{t\}$ 
9:   else
10:     $Tie \leftarrow Tie \cup ts$ 
11:   end if
12: end while
13: return  $MinTie$  : ensemble des liens minimaux
```

Théorème 2.39. *En supposant que f est la fonction de structure d'un système cohérent et non trivial, dont l'ensemble des coupes minimales est donné par $MinCut$, l'algorithme 3 calcule l'ensemble des liens minimaux correspondant.*

Preuve L'algorithme conserve :

1. Les ensembles Tie et $MinTie$ contiennent des liens, resp. des liens minimaux.
2. Pour tout lien minimal x du système décrit par la fonction de structure f , il existe un élément $y \in Tie \cup MinTie$ tel que $x \preceq y$.

L'invariant implique l'affirmation du théorème : lorsque $Tie = \emptyset$, l'ensemble $MinTie$ contient uniquement des liens minimaux (1) et tous les liens minimaux (2).

Nous montrons maintenant que l'invariant est satisfait lorsque la condition de la boucle est évaluée pour la première fois. Notons que le « père » d'une coupe minimale est nécessairement un lien ; et le fait que l'ensemble $MinTie = \emptyset$ lors de la première entrée dans la boucle établit la condition (1).

Pour la condition (2), nous supposons que $x = \langle x_1, \dots, x_n \rangle$ est un lien minimal. Puisque le système n'est pas trivial, il existe au moins un j tel que $x_j = 1$, alors x a un fils $x' = \langle x_1, \dots, x_{j-1}, 0, x_{j+1}, \dots, x_n \rangle$. Puisque x est un lien minimal, alors x' est une coupe. Par conséquent $MinCut$ contient une coupe minimale $y' = \langle y'_1, \dots, y'_n \rangle$ telle que $x' \preceq y'$. De plus, nous devons avoir $y'_j = 0$, sinon nous aurions $x \preceq y'$ mais $f(x) = 1$ et $f(y') = 0$, puisque x and y' sont respectivement un lien et une coupe. Cela entrerait en contradiction avec l'hypothèse de cohérence du système. Considérons à présent le père $y = \langle y'_1, \dots, y'_{j-1}, 1, y'_{j+1}, \dots, y'_n \rangle$ de y' : par construction nous avons $x \preceq y$ et $y \in Tie$, établissant la condition (2).

Il reste à prouver que l'invariant est préservé par toutes les exécutions de la boucle,

donc nous supposons qu'il est satisfait, que $Tie \neq \emptyset$ et que t soit l'élément de Tie choisi pour exécuter la boucle. Nous allons désigner par Tie' et $MinTie'$ les valeurs de ces variables à la fin de la boucle.

Grâce à la condition (1), nous savons que t est un lien et ts est défini comme un ensemble de liens. De plus, t est un lien minimal si et seulement si $ts = \emptyset$, et cela montre que la condition (1) est toujours satisfaite après l'exécution de la première boucle.

Pour la condition (2), supposons que x soit un lien minimal. Si $x \preceq y$ est satisfait pour un élément $y \in (Tie \setminus \{t\}) \cup MinTie$ lors de l'entrée dans la boucle, nous avons toujours $y \in Tie' \cup MinTie'$, et la condition (2) est trivialement préservée. Supposons maintenant que $x \preceq t$. Si $ts = \emptyset$, et nous avons $t \in MinTie'$, établissant la condition (2) à la fin du corps de la boucle. Sinon, t n'est pas minimal et est donc différent de x . Puisque $x \preceq t$, il existe un composant C_k qui fonctionne dans le n-uplet $t = \langle t_1, \dots, t_{k-1}, 1, t_{k+1}, \dots, t_n \rangle$ mais pas dans x . Le n-uplet $t' = \langle t_1, \dots, t_{k-1}, 0, t_{k+1}, \dots, t_n \rangle$ est donc un fils de t et il satisfait $x \preceq t'$. Comme le système est cohérent, t' est un lien et donc $t' \in ts \subseteq Tie'$. Cela termine la preuve. \square

Remarque 2.40. L'algorithme 3 peut manipuler le même lien de façon répétée. Cela peut être évité en ajoutant une variable $TieSeen$ qui contient tous les liens déjà traités par l'algorithme. La variable $TieSeen$ est initialisée à Tie . Dans la branche **else** du corps de la boucle, $(ts \setminus TieSeen)$ est ajouté à Tie et l'ensemble ts est ajouté à $TieSeen$.

Un exemple de cette approche sera présenté dans la section 2.7.1.

2.6 Calcul de la fiabilité du système

Dans cette section, nous utilisons le diagramme de Hasse restreint aux état de fonctionnement du système, construit à partir des liens minimaux obtenus grâce aux techniques de satisfiabilité. Afin de calculer la fiabilité des systèmes, nous cherchons à déterminer le polynôme de fiabilité, que l'approche de Brânzei et Aubry (2018) permet de déterminer à partir du diagramme de Hasse. À partir du diagramme de Hasse restreint aux état de fonctionnement, cette méthode attribue un *poids* à chaque noeud, afin de compter une seule et unique fois la probabilité de chaque état de fonctionnement du système.

Définition 2.41. Un *cube* est une expression logique contenant uniquement des opérateurs *conjonction* \wedge et des opérateurs *complément* \neg appliqués aux seules variables.

Définition 2.42. Par exemple, l'expression $a \wedge b \wedge \neg c$ est un cube.

Un cube correspond à une configuration.

Définition 2.43. Le monôme d'un cube est le produit des probabilités des composants en état de fonctionnement du cube.

Définition 2.44. L'expression $P(a) \cdot P(b)$ est le monôme du cube $a \wedge b \wedge \neg c$.

Un monôme représente la probabilité d'un noeud et de tous les noeuds qui lui sont supérieurs dans le diagramme de Hasse. Par conséquent, la probabilité d'un noeud peut-être comptée plusieurs fois dans le polynôme de fiabilité.

2.6.1 Probabilité d'un monôme

La défaillance des composants est modélisée par une loi de probabilité (section 1.5.1). Le polynôme de fiabilité exprime la fiabilité du système en fonction de la fiabilité des composants qui le composent. En remplaçant les probabilités de fonctionnement des composants par leur loi de distribution, nous pouvons calculer la valeur de la fiabilité du système. La fiabilité R_i d'un composant C_i est déterminée par sa loi de probabilité et la valeur de ses paramètres. Les variables R_i permettent d'écrire le polynôme de fiabilité, elles sont ensuite remplacées par l'expression de la probabilité de fonctionnement de C_i .

On considère un système dont l'ensemble des composants est \mathbb{C} et dont le noeud G_A , de la Figure 2.6, représente un état où les composants C_m et C_n fonctionnent et les composants de l'ensemble $\mathbb{C} \setminus \{C_m, C_n\}$ sont défaillant. La probabilité du monôme G_A est : $P(G_A) = R_m \times R_n$.

$$\begin{array}{cc} \text{n} & \text{m} \\ \uparrow & \uparrow \\ G_A = 0010001 \end{array}$$

FIGURE 2.6 – Noeud G_A .

Nous généralisons cette expression pour un monôme G , où les composants $C_i \in \mathbb{C}_1$ fonctionnent et les composants $C_i \in \mathbb{C}_2$ défaillent (\mathbb{C}_1 et \mathbb{C}_2 forment une partition de \mathbb{C}), la probabilité du monôme G est : $P(G) = \prod_{C_i \in \mathbb{C}_1} R_i$.

Exemple 2.45. Reprenons l'exemple 2.1 dont le diagramme de Hasse est la Figure 2.7.

La probabilité de la configuration $\langle 00111 \rangle$ est $P(00111) = R_3 \cdot R_4 \cdot R_5$.

La fiabilité de C_1 et C_2 est identique ($\lambda_1 = \lambda_2 = 10^{-2}h^{-1}$) et donnée par $R_1(t)$ et de C_3 , C_4 et C_5 est identique ($\lambda_3 = \lambda_4 = \lambda_5 = 10^{-3}h^{-1}$) et donnée par $R_3(t)$.

$$R_1(t) = e^{-\lambda_1 \cdot t}, \quad R_3(t) = e^{-\lambda_3 \cdot t} \quad (2.20)$$

2.6.2 Attribution des poids dans un diagramme de Hasse

Un poids est attribué à chaque noeud du diagramme de Hasse restreint. Un poids de 1 est attribué à chaque lien minimal et les poids des noeuds qu'il couvre sont incrémentés de 1. Ainsi, au départ, le poids d'un lien minimal vaut 1 et d'un non minimal au moins 1.

Exemple 2.46. Le diagramme de Hasse de l'exemple 2.1 est illustré grâce à la Figure 2.7, où l'ordre des composants est $\langle C_1, C_2, C_3, C_4, C_5 \rangle$ et où apparaît l'attribution initiale des poids. Par exemple, le noeud $\langle 0, 0, 1, 1, 1 \rangle$ est « couvert » par trois liens minimaux (les noeuds $\langle 0, 0, 0, 1, 1 \rangle$, $\langle 0, 0, 1, 0, 1 \rangle$ et $\langle 0, 0, 1, 1, 0 \rangle$), donc un poids de 3 lui est attribué.

2.6.3 Obtention du polynôme de fiabilité

En ramenant tous les poids du diagramme de Hasse restreint à 1, cette approche garantit que chaque lien soit compté une seule et unique fois dans le polynôme de fiabilité.

Premièrement, les monômes des liens minimaux sont ajoutés au polynôme de fiabilité.

Deuxièmement, la méthode consiste à choisir un minimum du diagramme :

- Si son poids est supérieur (resp. inférieur) à 1, son poids (et celui des noeuds qu'il couvre) est diminué (resp. augmenté) pour le ramener à 1 et le monôme équivalent est ajouté (resp. soustrait) au polynôme de fiabilité autant de fois que le poids a été augmenté ou diminué ;
- Si son poids vaut 1, ce noeud est supprimé.

Lors de cette étape, il est possible que le poids d'un noeud soit inférieur à 1. Dans l'exemple 2.47, c'est le cas du noeud $\langle 11111 \rangle$ de la Figure 2.7, dont le poids est augmenté de 2, à la fin, pour le ramener à 1.

Lorsque tous les poids valent 1, le polynôme de fiabilité est obtenu.

Exemple 2.47. Reprenons l'exemple 2.1 dont le diagramme de Hasse est la Figure 2.7.

Le polynôme de fiabilité du système est donné par l'équation (2.21). En supposant que $\lambda_1 = 1,00 \cdot 10^{-3}h^{-1}$, $\lambda_2 = 5,00 \cdot 10^{-4}h^{-1}$ et $\lambda_3 = \lambda_4 = \lambda_5 = 2,00 \cdot 10^{-3}h^{-1}$, la fiabilité de ce système est de $1,97 \cdot 10^{-6}$ au bout d'un an (8760 heures), illustrée par la Figure 2.8.

$$\begin{aligned}
R = & R_1 \times R_2 + R_3 \times R_4 + R_3 \times R_5 + R_4 \times R_5 \\
& - R_1 \times R_2 \times R_3 \times R_4 - R_1 \times R_2 \times R_3 \times R_5 - R_1 \times R_2 \times R_4 \times R_5 \\
& - 2 \times R_3 \times R_4 \times R_5 + 2 \times R_1 \times R_2 \times R_3 \times R_4 \times R_5
\end{aligned} \tag{2.21}$$

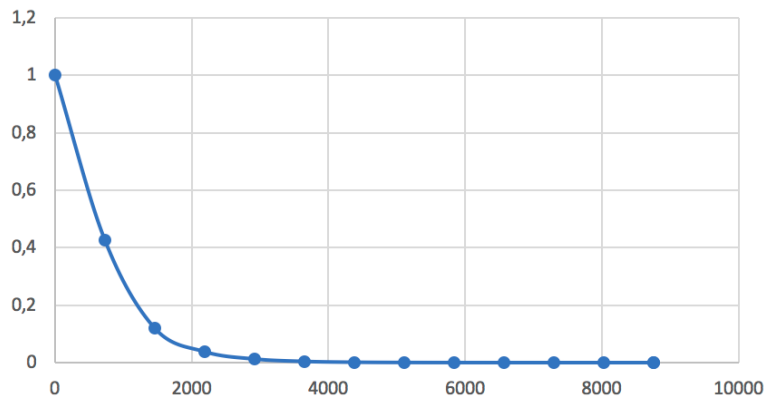


FIGURE 2.8 – Courbe d'évolution de la fiabilité du système de l'exemple 2.1 sur une année.

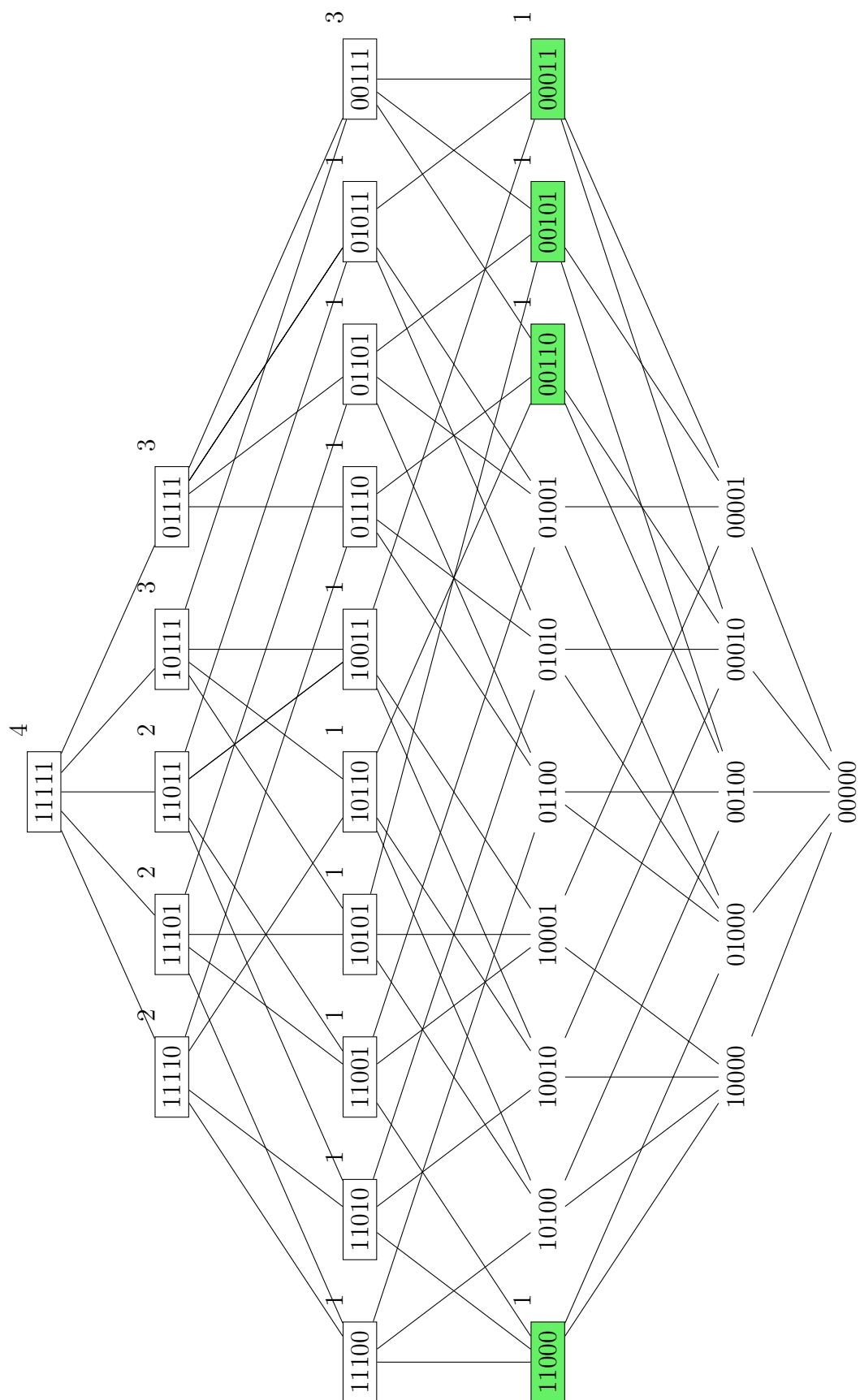


FIGURE 2.7 – Diagramme de Hasse pondéré de l'exemple 2.1 .

2.7 Application

Dans cette section, nous souhaitons mettre notre approche en application. Nous proposons dans un premier temps de considérer le système de secours d'un escalier mécanique. Dans un second temps, nous considérons des systèmes de tailles variables et comparons nos résultats à ceux du logiciel Grif (Satodev (2018)).

2.7.1 Application au système de secours d'un escalier mécanique

Dans cette sous-section, nous appliquons l'approche (algorithme 3) déterminant les liens minimaux à partir des coupes minimales, au système de secours d'un escalier mécanique proposé par Rogova et Lodewijks (2015). Il est modélisé par l'arbre de défaillances statique de la Figure 2.9 et contient :

- Un contrôleur principal, ou main controller (MC)
- Deux canaux constitués chacun :
 - D'un contrôleur de freinage, ou braking controller (BC)
 - D'un capteur, ou sensor (S)
 - D'un système de freinage, ou braking system (BS)

Le système fonctionne lorsqu'au moins le contrôleur principal et l'un des canaux fonctionnent. L'ensemble des composants est $\{MC, BC_1, S_1, BS_1, BC_2, S_2, BS_2\}$.

L'équation (2.22) donne la fonction de structure, d'après la méthode de la section 2.3.

$$f = c_{MC} \vee ((c_{BC_1} \vee c_{S_1} \vee c_{BS_1}) \wedge (c_{BC_2} \vee c_{S_2} \vee c_{BS_2})) \quad (2.22)$$

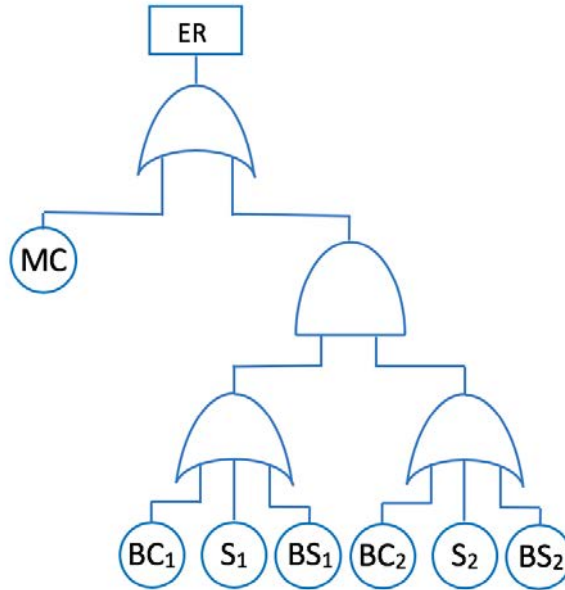


FIGURE 2.9 – Arbre de défaillances du système de secours d'un escalier mécanique.

Nous dualisons cette expression et l'exprimons comme une conjonction de disjonctions. Nous obtenons la fonction de structure dualisée $\neg f$ au format CNF (équation (2.23)).

$$\begin{aligned}
& \wedge \quad x_{MC} \\
& \wedge \quad (x_{BC_1} \vee x_{BC_2}) \quad \wedge \quad (x_{BC_1} \vee x_{S_2}) \quad \wedge \quad (x_{BC_1} \vee x_{BS_2}) \\
& \wedge \quad (x_{S_1} \vee x_{BC_2}) \quad \wedge \quad (x_{S_1} \vee x_{S_2}) \quad \wedge \quad (x_{S_1} \vee x_{BS_2}) \\
& \wedge \quad (x_{BS_1} \vee x_{BC_2}) \quad \wedge \quad (x_{BS_1} \vee x_{S_2}) \quad \wedge \quad (x_{BS_1} \vee x_{BS_2}).
\end{aligned} \tag{2.23}$$

À partir de la fonction de structure, nous déduisons certaines coupes (équation (2.24)).

$$\begin{aligned}
& \{MC\}, \{BC_1, BC_2\}, \{BC_1, S_2\}, \{BC_1, BS_2\}, \{S_1, BC_2\}, \\
& \{S_1, S_2\}, \{S_1, BS_2\}, \{BS_1, BC_2\}, \{BS_1, S_2\}, \{BS_1, BS_2\}.
\end{aligned} \tag{2.24}$$

À partir de ces coupes, il est nécessaire de déterminer les coupes minimales. Comme l'équation (2.24) contient l'ensemble des coupes minimales du système (et aucune coupe non minimale), nous appliquons l'algorithme 3.

L'ensemble des liens minimaux générés est le suivant :

$$Tie = \{\{MC, BC_1, S_1, BS_1\}, \{MC, BC_2, S_2, BS_2\}\}.$$

À partir des liens minimaux, nous construisons le diagramme de Hasse restreint aux états de fonctionnement du système (Figure 2.10), où l'ordre des composants est le suivant : $\{MC, BC_1, S_1, BS_1, BC_2, S_2, BS_2\}$.

Afin d'illustrer le principe de l'algorithme 3, la Figure 2.11 présente le diagramme de Hasse avec l'ensemble des liens (minimaux et non minimaux) et l'ensemble des coupes minimales, où les liens sont encadrés. Les liens sont encadrés, les liens minimaux sont encadrés en verts et les coupes ne sont pas encadrées.

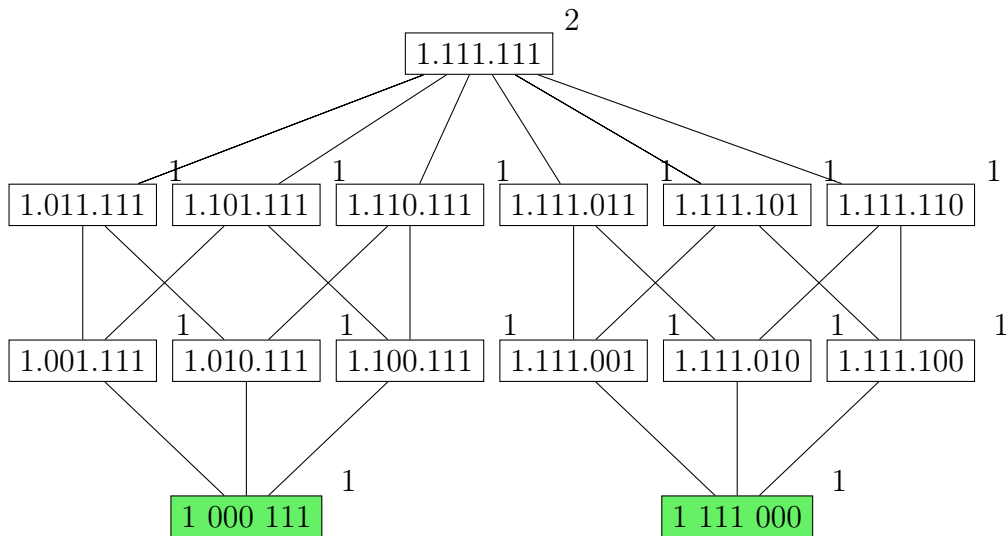


FIGURE 2.10 – Diagramme de Hasse restreint du système 2.9.

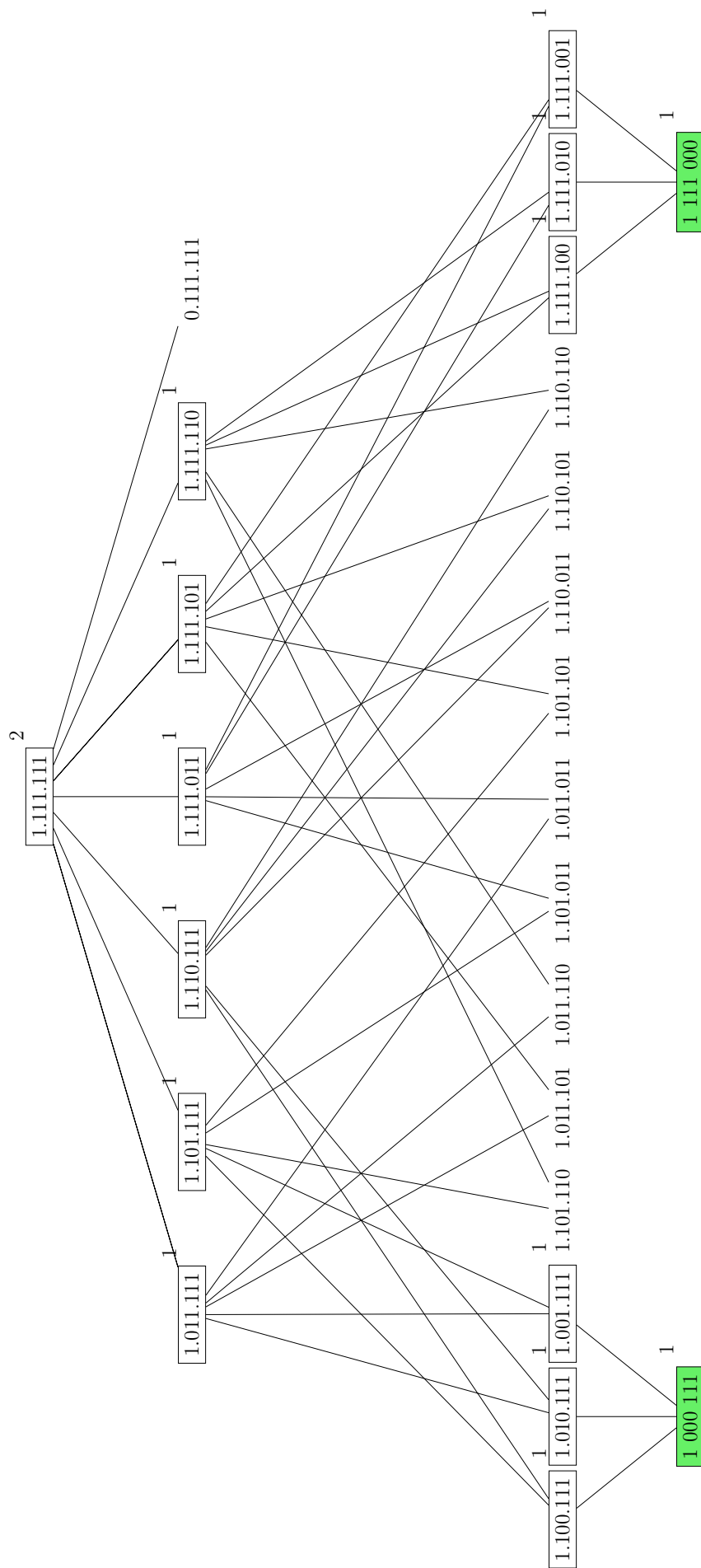


FIGURE 2.11 – Diagramme de Hasse du système 2.9 contenant l'ensemble des noeuds des coupes minimales et des liens minimaux.

La méthode de la section 2.6 permet de déterminer le polynôme de fiabilité :

$$\begin{aligned} R &= R_{MC} \cdot R_{BC1} \cdot R_{S1} \cdot R_{BS1} \\ &+ R_{MC} \cdot R_{BC2} \cdot R_{S2} \cdot R_{BS2} \\ &- R_{MC} \cdot R_{BC1} \cdot R_{S1} \cdot R_{BS1} \cdot R_{BC2} \cdot R_{S2} \cdot R_{BS2}. \end{aligned}$$

La fiabilité des contrôleurs (*MC* et *BC*) est supposée suivre une distribution exponentielle avec un taux de défaillance $\lambda = 5.5 \cdot 10^{-8} h^{-1}$, les capteurs (*S*) suivent une distribution exponentielle avec un taux de défaillance $\lambda = 1.09 \cdot 10^{-8} h^{-1}$, et les systèmes de freinage (*BS*) suivent une distribution de Weibull avec un paramètre d'échelle $\eta = 8.2942 \cdot 10^4 h$ et un paramètre de forme $\tau = 1.77459$.

La Figure 2.12 illustre l'évolution de la fiabilité du système durant vingt ans. La courbe bleue représente le résultat généré par Grif et la courbe rouge représente le résultat obtenu par notre approche.

La fiabilité du système après vingt ans (i.e. $1.752 \cdot 10^5$ heures) est estimée à 0.0483 par Grif et 0.0446 par notre approche. Une différence de moins de 1% est observée entre les deux valeurs qui est imputable aux approximations de calcul.

En outre, puisque nous supposons que le système n'est pas réparable, il est cohérent d'obtenir une fiabilité de 0.99 après un an et 0.04 après vingt ans.

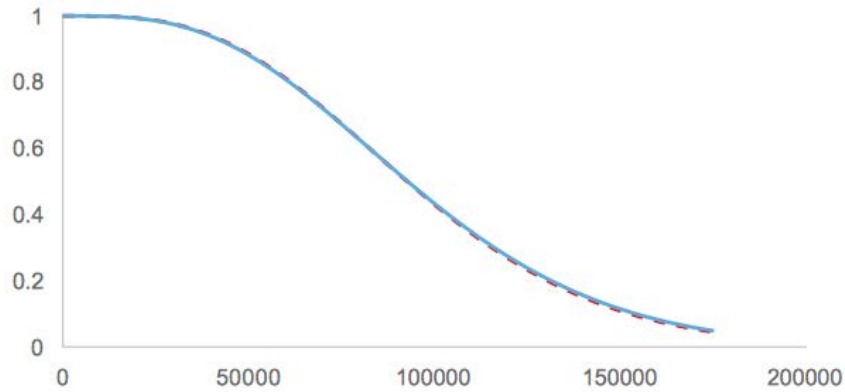


FIGURE 2.12 – Évolution de la fiabilité du système au cours du temps.

2.7.2 Application à une bibliothèque de systèmes industriels

Le logiciel Grif permet, entre autres, de modéliser les systèmes au moyen de diagrammes de fiabilité, d'arbres de défaillances, de chaînes de Markov et de réseau de Petri. Nous nous intéressons tout particulièrement à sa capacité à calculer les coupes minimales et la fiabilité d'un système à partir d'un arbre de défaillances.

2.7.2.1 Bibliothèque

Nous proposons de considérer une bibliothèque de systèmes industriels. Ces systèmes sont représentés par leur fonction de structure, obtenue à partir d'un arbre de défaillances statique. Les composants de ces systèmes sont anonymisés et numérotés pour préserver la confidentialité des infrastructures analysées. Cette bibliothèque de fichiers a fait l'objet d'une étude par Rauzy et Dutuit (1997).

La bibliothèque est constituée de 46 systèmes provenant d'entreprises industrielles. Les systèmes modélisés contiennent entre 25 composants (pour un système qui contient 35 portes) et 1567 composants (pour un système qui contient 1621 portes). La fonction de structure de certains systèmes contient uniquement des portes ET et OU et d'autres utilisent également les portes KooN.

Les fichiers de cette bibliothèque sont sauvegardés au format cp. Ce format textuel contient une forme compacte et explicite de la fonction de structure. Il se compose d'une première partie permettant la déclaration des variables et d'une seconde partie répertoriant les règles associés à l'arbre de défaillances.

Les variables sont répertoriables en trois catégories :

- r : l'évènement redouté (ou top),
- e_i : un composant de l'arbre de défaillance (ou event), et
- g_i : une porte de l'arbre de défaillance (ou gate).

Les règles (rules) énoncées sont les expressions de chaque porte de l'arbre de défaillance, exprimées en fonction de leurs entrées.

Le format cp n'étant pas supporté par les solveurs de satisfiabilité, les fichiers doivent être convertis vers un format compatible, afin d'utiliser les techniques de satisfiabilité pour analyser la fonction de structure et déterminer les coupes minimales. Une conversion vers le format smt2 permet d'utiliser les techniques de satisfiabilité en évitant la transformation de la formule à la forme CNF.

La Figure 2.13 présente le fichier cp de l'exemple nommé *Test* et la Figure 2.14 présente l'arbre de défaillances de l'exemple *Test*.

```

1  variable
2    bool r1,g1,g2,e1,e2,e3,e4,e5;
3
4  rule
5    (r1==(g1 && g2));
6    (g1==(e1 || e2 || e3));
7    (g2==(e4 || e5));

```

FIGURE 2.13 – Fichier cp *Test*.

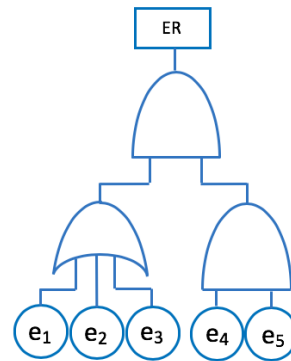


FIGURE 2.14 – Arbre de défaillances de *Test*.

2.7.2.2 Comparaison des résultats

Dans l'étude de l'escalier mécanique, nous avons comparé la fiabilité obtenue par Grif et celle calculée par notre approche. Nous souhaitons à présent nous intéresser à la génération des liens ou des coupes. Les liens ne sont pas générés par Grif, seules les coupes sont déterminées.

Afin qu'une comparaison soit possible, il est nécessaire de déterminer les coupes minimales par notre approche. Puisque la coupe est le dual du lien, notre approche peut être adaptée pour permettre de calculer les coupes minimales (MCSs) (plutôt que les liens minimaux (MTSs)). Pour chaque système considéré, nous utilisons notre approche basée sur l'utilisation d'un solveur SAT (algorithme 2) pour déterminer les coupes minimales et nous les comparons à celles obtenues par Grif.

Un lexer et un parser sont présentés dans l'Annexe 1 afin de transformer le fichier cp en fichier smt2. Pour calculer les coupes au lieu des liens, il suffit de ne pas dualiser la fonction de structure. Nous appliquons ensuite l'algorithme 2 au fichier généré par le lexer et le parser pour identifier les coupes minimales. \square

Le fichier *Chinese* est l'un des plus petits fichiers de la bibliothèque. Il est composé de 25 composants : e_1 à e_{25} . Nous calculons 730 coupes et déterminons que 392 sont minimales, en un temps de 32 secondes. Grif génère également 392 coupes minimales, en 4 secondes, qui sont identiques à celles trouvées par notre approche.

De la même façon, le fichier *Baobab2* est composé de 32 composants (e_1 à e_{32}) et 39 portes, dont des portes ET, OU et KooN. Nous identifions 5848 coupes dont 4806 sont minimales, en un temps de 19 minutes. Grif génère les mêmes coupes minimales, en 6 secondes.

Le tableau 2.1 présente les résultats de notre approche, dont le nombre de composants de chaque système, le nombre de coupes minimales et le temps de calcul T des coupes minimales (Minimal Cut Sets ou MCS) par notre approche.

Système	Nombre de composants	Temps T	Nombre de MCSs
<i>Test</i>	5	0,06 sec	3
<i>Chinese</i>	25	32 sec	392
<i>Baobab2</i>	32	1152 sec	4806

Tableau 2.1 – Nombre de MCSs et temps de calcul selon notre approche.

Nous obtenons bien les mêmes résultats par notre approche que le logiciel Grif. Toutefois, il serait pertinent d'optimiser notre implémentation afin d'améliorer ses performances.

2.7.2.3 Résumé

Puisque notre approche se focalise sur les liens minimaux, nous avons appliqué notre approche de calcul des liens minimaux basée sur l'utilisation d'un solveur SAT à plusieurs systèmes de la bibliothèque. Le Tableau 2.2 présente nos résultats.

Le temps de calcul des liens minimaux dépend du nombre de composants du système et du nombre de portes (de l'arbre de défaillances), qui déterminent le nombre de variables de la fonction de structure envoyée au solveur SAT. Cependant, le temps de calcul dépend également de la porte la plus haute de l'arbre de défaillances (la fonction de structure est-elle une disjonction ou une conjonction ?) et de l'architecture du système.

Notre approche fournit les liens minimaux des systèmes de 60 à 84 variables en moins de 4 minutes. Pour les systèmes avec 129 et 186 variables, le solveur SAT trouve plusieurs dizaines de milliers de liens, mais n'avait pas terminé après plus d'une journée de calcul et nous avons interrompu le calcul. Plus le nombre de liens calculés est grand, plus le temps nécessaire au solveur SAT est grand pour déterminer un nouveau modèle.

Par conséquent, notre approche semble déterminer les coupes minimales des systèmes et sous-systèmes industriels en un temps raisonnable dès lors que la somme du nombre de composants et de portes ne dépasse pas une centaine de variables.

Ce temps de réponse pourrait être amélioré en filtrant les liens non minimaux durant la boucle d'interaction avec le solveur SAT. Cela réduirait notamment la taille de l'entrée fournie au solveur SAT. Après plusieurs dizaines de milliers de modèles générés, le temps de calcul des liens minimaux serait ainsi réduit.

Nombre de variables	Nombre de composants	Nombre de portes (internes)	Nombre de TSs calculés	Nombre de MTSs	Temps (s)
60	25	35	15	12	0.6
71	32	39	1993	960	218
80	51	29	5	5	0.2
82	53	29	131	53	5.1
84	49	35	31	19	1.1
129	89	40	> 37 000		> 1 jour
186	80	106	> 25 000		> 1 jour

Tableau 2.2 – Nombre de TSs calculés, nombre de MTSs et temps de calcul.

2.8 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux arbres de défaillances statiques par le biais de leur fonction de structure. Afin d'utiliser les techniques de satisfiabilité, nous avons écrit la fonction de structure comme une formule propositionnelle et considéré les différentes formes nomales qu'elle peut prendre, conjonctive ou disjonctive.

Nous avons proposé une méthode et plusieurs algorithmes pour déterminer les liens minimaux des systèmes statiques grâce aux techniques de satisfiabilité et quelque soit la forme de la fonction de structure obtenue à partir de l'arbre de défaillances. Cela a permis de mettre en avant le potentiel des techniques de satisfiabilité appliquées à la fiabilité.

Néanmoins, la modélisation par arbres de défaillances statiques est très restreinte quant aux systèmes qu'elle est capable de représenter. Il existe de nombreuses extensions des arbres de défaillances, comme présenté dans Ruijters et Stoelinga (2015), qui offrent un plus vaste choix de modélisation que les arbres de défaillances statiques.

La modélisation par arbres de défaillances dynamiques est l'une de ces extensions des arbres de défaillances. Elle est plus riche et plus précise que la modélisation par arbres de défaillances statiques, ce qui permet d'élargir le champ des systèmes qui peuvent être modélisés. Les arbres de défaillances dynamiques permettent de prendre en compte l'ordre d'occurrence des défaillances pour déterminer l'état du système, les dépendances fonctionnelles entre les composants principaux et les composants redondants partagés par plusieurs portes.

Le chapitre suivant porte sur l'analyse de la fiabilité de systèmes modélisés par arbres de défaillances dynamiques en se basant sur les techniques de satisfiabilité.

Chapitre 3

Étude des systèmes modélisés par arbres de défaillances dynamiques grâce aux techniques de satisfiabilité

3.1 Introduction

Le chapitre 2 propose une approche permettant d'étudier les systèmes modélisés par arbre de défaillances statique. Cependant, la modélisation par arbre de défaillances statique restreint fortement les possibilités de modélisation, ne permettant pas de modéliser les systèmes où l'ordre de défaillance influe sur l'état du système, comme les systèmes contenant un commutateur ayant pour but de basculer d'un composant principal vers un composant redondant, lorsque le composant principal défaille. Si le commutateur défaille avant que le composant principal défaille, alors le basculement vers le composant redondant ne peut pas se produire et le système ne peut pas se servir de ce composant de rechange. Au contraire, si le commutateur défaille après le composant principal, alors cela ne provoque pas de défaillance du système, car le basculement a déjà eu lieu. Lorsque le commutateur et le composant principal ont défailli, si l'ordre d'occurrence des défaillances n'est pas connu, nous ne pouvons pas différencier ces deux scénarios et l'état du système ne peut pas être déterminé. Un tel système peut être modélisé par un arbre de défaillances dynamique (AdDD ou DFT) qui prend en considération l'ordre d'occurrence des défaillances. En outre, cette modélisation permet de modéliser le fait que la défaillance d'un composant provoque la défaillance instantanée d'autres composants du système, c'est la dépendance dysfonctionnelle. Enfin, cette modélisation permet de modéliser la redondance en représentant les composants fonctionnels en attente de sollicitation. Ils sont sollicités lorsque le composant en fonctionnement défaille. Un composant en attente peut être assigné à plusieurs portes et la première porte à le solliciter en obtient l'usage, il devient alors indisponible pour les autres portes.

L'aptitude des arbres de défaillances dynamiques à modéliser ces comportements est rendue possible grâce aux trois portes que l'on qualifie de dynamiques, chacune correspondant à une des trois fonctionnalités présentées ci-avant.

Nous illustrerons notre propos par l'exemple d'un système à neuf composants, dont la Figure 3.1 présente l'arbre de défaillances.

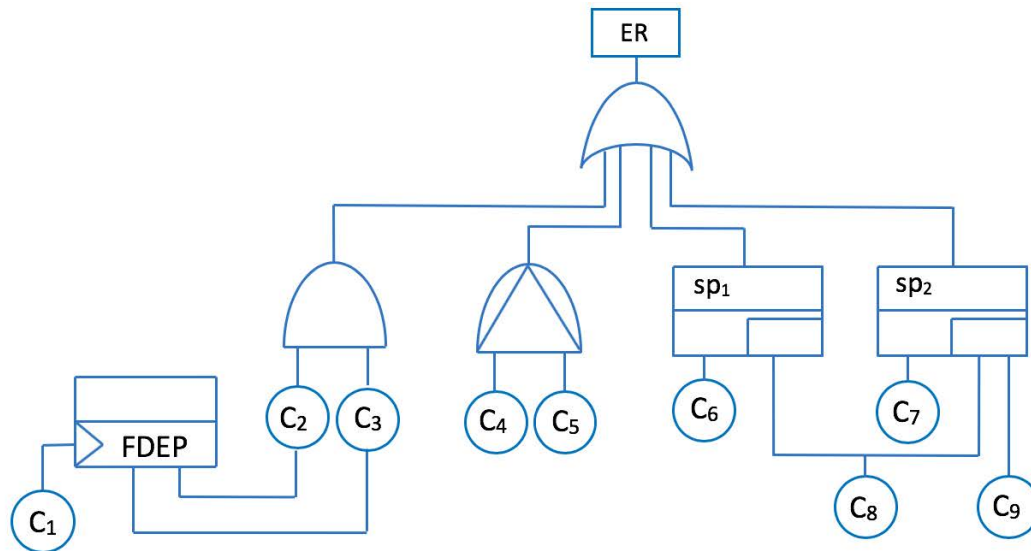


FIGURE 3.1 – Exemple d'arbre de défaillances dynamique.

Exemple 3.1. L'acronyme ER signifie "évènement redouté". Il constitue le sommet de l'arbre de défaillances et représente la défaillance du système, c'est-à-dire l'impossibilité de réaliser le service attribué au système. Cet arbre de défaillance contient des portes statiques et dynamiques :

- une porte statique OU dont les entrées sont les portes ET, PAND et les deux portes SPARE sp_1 et sp_2 en redondance froide ;
- une porte statique ET dont les entrées sont les composants C_2 et C_3 ;
- une porte dynamique FDEP représentant une dépendance fonctionnelle qui propage instantanément la défaillance du composant C_1 aux composants C_2 et C_3 ;
- une porte dynamique PAND qui s'active si C_5 défaille après que C_4 ait défailli ; et
- deux portes dynamiques SPARE sp_1 et sp_2 dont les composants principaux C_6 et C_7 peuvent chacun être remplacé par C_8 (ce composant est dit partagé). Quant à C_9 , seule la porte sp_2 peut le solliciter.

Les travaux présentés dans ce chapitre sont décomposés de la façon suivante :

- La section 3.2 rassemble les définitions des notions nécessaires à nos travaux.
- La section 3.3 propose une méthodologie, détaillée pour chaque porte, permettant d'obtenir la fonction de structure, afin de déterminer une expression de la fonction de structure dualisée. Les interactions entre les portes FDEP et les autres portes

dynamiques sont évoquées. En effet, les portes FDEP conduisent à des défaillances simultanées de composants et les portes PAND et SPARE ont besoin de connaître l'ordre des défaillances, soit pour savoir si la porte s'est activée dans le cas d'une porte PAND ; soit pour savoir à quelle porte associer un composant de rechange dans le cas de plusieurs portes SPARE.

- Dans la section 3.4, les séquences de lien (minimales) sont définies comme l'extension de la notion de lien (minimal) aux arbres de défaillances dynamiques. Une séquence de lien minimale (MTSS) est un ensemble minimal de composants garantissant le fonctionnement du système en intégrant l'ordre de défaillance des composants ayant défailli précédemment. Nous proposons une approche qui permette d'obtenir les séquences de lien minimales, pour tout système dynamique, à partir de la fonction de structure de ce système grâce aux techniques de satisfiabilité.
- Dans la section 3.5 nous proposons une adaptation du diagramme de Hasse aux arbres de défaillances dynamiques pour hiérarchiser les séquences de lien obtenues. Le polynôme de fiabilité du système permettant de calculer la fiabilité du système en fonction de la fiabilité des composants, les résultats présentés dans cette section visent à calculer la probabilité des cubes¹ du polynôme de fiabilité.
- La section 3.6 illustre l'approche présentée par une application à un système d'assistance cardiaque.
- La section 3.7 conclut ce chapitre et présente des perspectives.

3.2 Définitions

Dans cette partie, nous présentons les concepts et les notions que nous utilisons dans ce chapitre. Nous définissons, pour les systèmes dynamiques, la notion d'état, de scénario, de configuration, de fonction de structure et la relation d'ordre entre les configurations.

Définition 3.2. Un *système* est une entité constituée de n composants C_i , $i \in \{1, \dots, n\}$.

On dit d'un système qu'il est *dynamique* si l'arbre de défaillances qui le représente contient au moins une des trois portes : FDEP, PAND et SPARE, ce qui signifie que les états des composants ne suffisent pas à déterminer l'état du système. Un arbre de défaillances dynamique peut comporter des portes dynamiques et des portes statiques.

Définition 3.3. L'*état* d'un composant C_i est représenté par la variable booléenne c_i , $c_i \in \mathbb{B} = \{0, 1\}$, et l'état du système est représenté par la variable booléenne, où 0 (resp. 1) signifie que l'entité associée, composant ou système, fonctionne (resp. défaille).

L'étude d'un système débute à l'instant initial $t = 0$. Puisque nous effectuons une étude de fiabilité, nous considérons que tous les composants sont fonctionnels au début de

1. Un cube contient des opérateurs *conjonction* \wedge et *complément* \neg (appliqués aux seules variables).

l'étude. Par conséquent, l'instant de défaillance t_i d'un composant C_i est un réel positif. Tant qu'un composant C_i n'a pas défailli, la valeur de la variable de temps qui lui est associée, t_i , est fixée à $+\infty$, car elle est considérée comme non pertinente.

Définition 3.4. Un *scénario* correspond à une séquence de défaillances de composants.

Au contraire des liens et des coupes, les scénarios prennent en considération l'ordre des défaillances, ils sont donc plus adaptés à l'étude des systèmes dynamiques.

Exemple 3.5. Pour le système de la Figure 3.1, considérons la porte PAND et les composants C_4 et C_5 . Le scénario constitué des défaillances de C_5 puis C_4 mène à un état de fonctionnement du système, tandis que le scénario constitué des défaillances de C_4 puis C_5 mène à un état de défaillance du système.

Dans le cas d'un système statique, une *configuration* est définie comme étant un n -uplet représenté par le mot booléen $G = (c_1, \dots, c_n) \in \mathbb{B}^n$. Dans le cas d'un système dynamique, nous proposons une nouvelle définition de la configuration afin qu'elle prenne en compte l'instant d'occurrence des défaillances.

Définition 3.6. Pour un système dynamique, une *configuration* est un $2n$ -uplet constitué d'une variable booléenne c_i et d'un instant de défaillance t_i pour chaque composant du système. Elle est représentée par $G = ((c_1, \dots, c_n), (t_1, \dots, t_n)) \in \mathbb{B}^n \times (\mathbb{R}^+)^n$.

Pour les systèmes dynamiques, une configuration est donc composée de l'état et de l'instant de défaillance de chaque composant.

Définition 3.7. Deux configurations G et G' :

$$G = ((c_1, \dots, c_n), (t_1, \dots, t_n)) \text{ et } G' = ((c'_1, \dots, c'_n), (t'_1, \dots, t'_n)), \quad (3.1)$$

sont ordonnées telles que $G \preceq G'$ si et seulement si

$$\forall i \in \{1, \dots, n\}, c_i \leq c'_i, \text{ et } \forall k \in \{1, \dots, n\}, c_k = 1 \Rightarrow t'_k = t_k. \quad (3.2)$$

Ainsi, une configuration G est *inférieure* à une configuration G' si chacun des booléens c_i est inférieur ou égal au booléen c'_i , et si les relations d'ordres portant sur les composants défaillants de G sont aussi des relations d'ordres de G' .

Nous définissons ensuite l'équivalence entre deux configurations.

Définition 3.8. Deux configurations G et G' sont équivalentes si et seulement si :

$$G \preceq G' \text{ et } G' \preceq G.$$

Pour déterminer l'état du système à partir de l'état des composants et de l'instant de défaillance des composants, nous proposons une définition de la fonction de structure des systèmes dynamiques.

Dans la suite de ce manuscrit, l'appellation *fonction de structure* fera référence à la fonction de structure étendue aux aspects dynamiques des arbres de défaillances.

Définition 3.9. La *fonction de structure* d'un système dynamique est une fonction f

$$\begin{aligned} f : \mathbb{B}^n \times (\mathbb{R}^+)^n &\rightarrow \mathbb{B} \times \mathbb{R}^+ \\ G &\mapsto (f, t) \end{aligned} \quad (3.3)$$

qui décrit l'état du système et l'instant auquel il a défailli, s'il a défailli. La fonction de structure associe, à une configuration $G = ((c_1, \dots, c_n), (t_1, \dots, t_n))$, un couple (f, t) , où la variable f indique l'état du système et la variable t indique l'instant de temps auquel la défaillance du système est survenue.

Afin de déterminer l'expression de la fonction de structure d'un système, nous cherchons à déterminer les expressions c_γ et t_γ de chaque porte γ de l'arbre de défaillances, exprimant respectivement l'activation et l'instant d'activation de la porte. De la même façon que pour les composants, la variable de temps t_γ est fixée à $+\infty$ tant que la porte γ ne s'est pas activée.

Exemple 3.10. Soit une porte PAND dont les entrées sont les composants C_1 et C_2 , modélisée par la Figure 3.2. Elle est décrite par les équations (3.4) et (3.5).

$$c_\gamma = c_1 \wedge c_2 \wedge t_1 < t_2 \quad (3.4)$$

$$t_\gamma = \begin{cases} t_2 & \text{si } c_1 \wedge c_2 \wedge t_1 < t_2 \\ +\infty & \text{sinon} \end{cases} \quad (3.5)$$

Les configurations sont du type $((c_1, c_2), (t_1, t_2))$. La configuration $((1, 1), (6, 8))$ active la porte PAND puisqu'elle rend vrai le booléen c_γ ; tandis que les configurations $((0, 1), (+\infty, 7))$ et $((1, 1), (6, 2))$ rendent faux le booléen c_γ et n'activent pas la porte. \square



FIGURE 3.2 – Porte PAND à deux entrées.

Afin d'établir l'expression de la fonction de structure des systèmes dynamiques en fonction des portes de leur arbre de défaillances, nous proposons dans la section suivante d'exprimer les variables c_γ et t_γ pour chaque porte statique et dynamique.

3.3 Fonction de structure d'un système dynamique

Dans cette section, nous considérons dans un premier temps les portes statiques ET, OU et KooN et dans un second temps les portes dynamiques FDEP, PAND et SPARE.

Dans la suite de ce chapitre, les systèmes sont supposés dynamiques, cohérents, non réparables et ayant un comportement binaire.

3.3.1 Portes statiques

Nous ajoutons une équation à chacune des portes statiques afin d'exprimer l'instant d'activation des portes statiques. Nous obtenons donc deux expressions pour chaque porte.

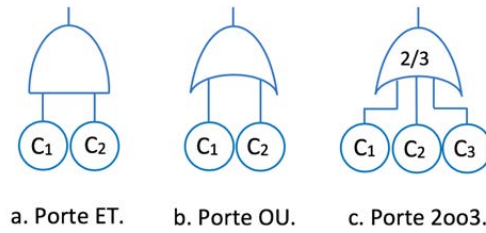


FIGURE 3.3 – Portes statiques.

Nous modélisons la porte ET par l'équation (3.6) qui exprime l'activation de la porte et l'équation (3.7) qui définit l'instant d'activation de la porte.

Tant que les deux composants C_1 et C_2 n'ont pas défailli, $t_{ET} = +\infty$. Dès qu'ils ont défailli tous les deux, alors l'instant d'activation de la porte est instancié à la valeur de l'instant de défaillance du dernier à défailir, soit le maximum des deux valeurs.

$$c_{ET} = c_1 \wedge c_2 \quad (3.6)$$

$$t_{ET} = \max(t_1, t_2) \quad (3.7)$$

Nous modélisons la porte OU, son dual, par les équations (3.8) et (3.9).

Dès qu'un composant parmi C_1 et C_2 défaille, alors l'instant d'activation de la porte est instancié à la valeur de l'instant de défaillance de ce composant. Qu'un ou deux composants aient défailli, cette valeur est le minimum des deux valeurs.

$$c_{OU} = c_1 \vee c_2 \quad (3.8)$$

$$t_{OU} = \begin{cases} \min(t_1, t_2) & \text{si } c_1 \vee c_2 \\ +\infty & \text{sinon} \end{cases} \quad (3.9)$$

Nous modélisons la porte 2oo3 par les équations (3.10) et (3.11).

Dès que deux composants parmi C_1 , C_2 et C_3 ont défailli, alors l'instant d'activation de la porte est instancié à la valeur de l'instant de la deuxième défaillance. Les équations (3.10) et (3.11) seraient à adapter pour d'autres valeurs de K et N .

$$c_{2003} = (c_1 \wedge c_2) \vee (c_1 \wedge c_3) \vee (c_2 \wedge c_3) \quad (3.10)$$

$$t_{2003} = \begin{cases} \max(t_1, t_2) & \text{si } c_1 \wedge c_2 \wedge \neg c_3 \\ \max(t_1, t_3) & \text{si } c_1 \wedge \neg c_2 \wedge c_3 \\ \max(t_2, t_3) & \text{si } \neg c_1 \wedge c_2 \wedge c_3 \\ \min(\max(t_1, t_2), \max(t_1, t_3), \max(t_2, t_3)) & \text{sinon} \end{cases} \quad (3.11)$$

3.3.2 Portes dynamiques

Afin d'obtenir la fonction de structure d'un système dynamique à partir de son arbre de défaillances, nous considérons à présent les portes dynamiques FDEP, PAND et SPARE, qui permettent de modéliser la dépendance fonctionnelle, la priorité à la défaillance et la redondance.

Afin de souligner l'importance de l'ordre de défaillance des composants, nous proposons l'utilisation de chronogrammes, où un front montant correspond à une défaillance.

3.3.2.1 Porte FDEP

La porte FDEP (Functional-Dependency) proposée par Dugan et al. (1990) permet de modéliser une dépendance fonctionnelle entre une *gâchette* et des composants en entrée^{2 3}.

La gâchette se situe sur le côté gauche de la porte FDEP et peut être constituée d'un composant, d'une porte statique, d'une porte PAND ou d'une porte SPARE. Lorsque la gâchette est activée, cela provoque l'activation des entrées de la porte FDEP.

La Figure 3.4 représente une porte FDEP à deux entrées (C_1 et C_2) dont la gâchette est le composant C_3 . Si le composant C_3 défaille, alors cela conduit à la défaillance des composants C_1 et C_2 .

Deux comportements sont possibles. Sur le chronogramme de gauche de la Figure 3.6, les composants C_1 et C_2 défaillent avant le composant C_3 . Dans ce cas, la défaillance de C_3 n'a pas d'incidence sur C_1 et C_2 . Sur le chronogramme de droite, C_1 défaille puis C_3 défaille. L'activation de la gâchette entraîne la défaillance instantanée de C_2 .

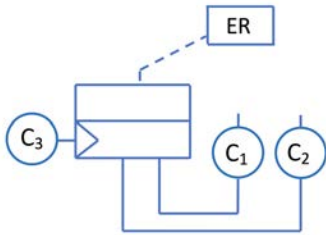


FIGURE 3.4 – Porte FDEP.

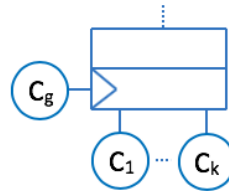


FIGURE 3.5 – Porte FDEP généralisée.

-
2. Seuls des composants peuvent être des entrées d'une porte FDEP.
 3. Une porte FDEP n'a pas de sortie, ce qui est représenté par un trait en pointillés à son sommet.

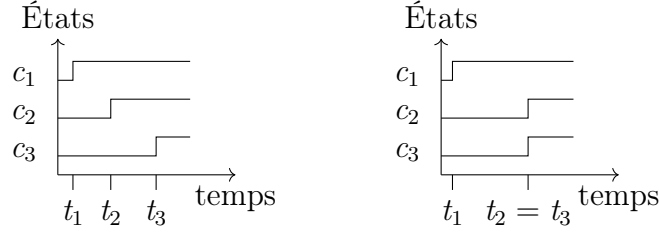


FIGURE 3.6 – Chronogrammes de la porte FDEP de la Figure 3.4.

La dépendance des défaillances est représentée formellement par des contraintes sous la forme d'implications concernant les états des composants (3.12) et sous la forme d'inégalités pour les instants de défaillance (3.13). Ces contraintes sont l'expression formelle de la dépendance des défaillances.

$$c_3 \Rightarrow c_1 \quad c_3 \Rightarrow c_2 \quad (3.12)$$

$$t_1 \leq t_3 \quad t_2 \leq t_3 \quad (3.13)$$

La porte FDEP n'ayant pas de sortie, les expressions c_γ ou t_γ ne sont pas exprimées.

Nous proposons de généraliser la porte FDEP à k entrées, modélisée par l'arbre de défaillances de la Figure 3.5, grâce aux équations (3.14).

$$c_g \Rightarrow c_i \quad t_i \leq t_g, \quad \forall i \in \{1, \dots, k\}. \quad (3.14)$$

3.3.2.2 Porte PAND

La porte PAND (Priority-AND), introduite par Fussell et al. (1976), permet de modéliser un système pour lequel l'ordre d'occurrence des défaillances influence le fonctionnement du système. La porte PAND s'active si toutes ses entrées s'activent dans un ordre précis, de la gauche vers la droite suivant leur ordre de connexion à la porte.

Pour modéliser la porte PAND, Merle et Roussel (2007) utilisent une inégalité large. En accord avec notre hypothèse selon laquelle les défaillances ne peuvent pas se produire simultanément, nous choisissons de modéliser ces inégalités par une inégalité stricte. L'ordre d'occurrence de deux défaillances devient binaire.

Nous illustrons la porte PAND avec un exemple à 2 entrées, chaque entrée étant un composant. Cette porte est représentée par la Figure 3.7.



FIGURE 3.7 – Porte PAND.

Sur le chronogramme de droite de la Figure 3.8, la porte s'active lorsque le composant C_2 défaille après le composant C_1 . Sur le chronogramme de gauche de la Figure 3.8, lorsque C_2 défaille avant C_1 , alors la porte ne s'active pas.

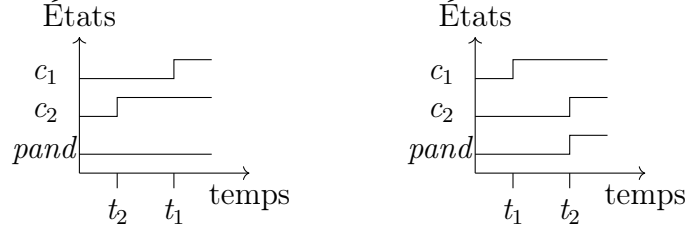


FIGURE 3.8 – Chronogrammes de la porte PAND.

Nous modélisons la porte PAND de la Figure 3.7 par les expressions (3.15) et (3.92). Puisque la porte s'active seulement si le second composant défaille après le premier, alors l'instant d'activation de la porte est l'instant de défaillance du second composant.

$$c_{pand} = c_1 \wedge c_2 \wedge t_1 < t_2 \quad (3.15)$$

$$t_{pand} = \begin{cases} t_2 & \text{si } c_1 \wedge c_2 \wedge t_1 < t_2 \\ +\infty & \text{sinon} \end{cases} \quad (3.16)$$

Nous généralisons la porte PAND à n entrées et la modélisons par les équations (3.17) et (3.18). L'instant d'activation de la porte est déterminé par l'activation de la *dernière* entrée de la porte (la plus à droite).

$$\begin{aligned} c_{pand} &= c_1 \wedge \dots \wedge c_n \\ &\wedge t_1 < t_2 \\ &\wedge \dots \\ &\wedge t_{n-1} < t_n \end{aligned} \quad (3.17)$$

$$t_{pand} = \begin{cases} t_n & \text{si } c_1 \wedge \forall i \in \{2 \dots n\}, c_i \wedge t_{i-1} < t_i \\ +\infty & \text{sinon} \end{cases} \quad (3.18)$$

3.3.2.3 Porte SPARE

La troisième porte dynamique est la porte SPARE proposée par Dugan et al. (1992). Elle modélise des entités de rechange permettant de remplacer une entité défailante. La porte SPARE s'active si chacune de ses entrées a défailli ou est utilisée par une autre porte SPARE, une entrée pouvant être un composant ou une porte. Lorsque l'entrée utilisée s'active, soit une seule entrée peut être sollicitée, soit plusieurs entrées peuvent être sollicitées et l'entrée sollicitée est celle située *la plus à gauche* parmi les entrées disponibles, selon les connexions à la porte. Lorsqu'une entrée peut être utilisée par plusieurs portes

SPARE, elle est dite *partagée* et elle est utilisée par la première porte à la solliciter.

Plusieurs modes de redondance existent pour les portes SPARE. Nous considérerons des redondances chaudes par défaut, ce qui signifie que les entrées peuvent défaillir avant d'être sollicitées. Le cas des redondances froides sera traité par la suite et nécessite l'ajout d'une contrainte supplémentaire concernant l'ordre d'occurrence des défaillances.

Porte SPARE sans composant partagé

Une porte SPARE dont aucun composant n'est partagé avec d'autres portes SPARE se comporte comme une porte ET (les composants étant en redondance chaude).

Le comportement de la porte sp_0 (Figure 3.9) est décrit par les équations (3.19)-(3.20).

$$c_{sp_0} = c_1 \wedge c_2 \wedge \dots \wedge c_k \quad (3.19)$$

$$t_{sp_0} = \max(t_1, \dots, t_k) \quad (3.20)$$

Portes SPARE avec un composant redondant partagé

Considérons une porte SPARE avec une entrée partagée avec une autre porte SPARE. Sur la Figure 3.10, la porte sp_1 partage le composant C_3 avec la porte sp_2 . La porte sp_1 s'active si C_1 a défailli et si C_3 ne peut pas être sollicité, soit parce qu'il a défailli, soit parce qu'il est déjà utilisé par la porte sp_2 . Le comportement de la porte sp_1 est décrit par les équations (3.21)-(3.22). Par symétrie, l'expression de la porte sp_2 peut être déterminée.

$$\begin{aligned} c_{sp_1} &= (c_1 \wedge c_3) \vee (c_1 \wedge c_2 \wedge t_2 < t_1) \\ &= c_1 \wedge (c_3 \vee (c_2 \wedge t_2 < t_1)) \end{aligned} \quad (3.21)$$

$$t_{sp_1} = \begin{cases} \max(t_1, t_3) & \text{si } t_1 < t_2 \\ t_1 & \text{sinon} \end{cases} \quad (3.22)$$

L'activation des portes sp_1 et sp_2 est illustrée par les chronogrammes de la Figure 3.11. Sur le chronogramme de gauche, le composant C_2 défaille en premier et le composant C_3 est sollicité pour le remplacer. Lorsque le composant C_1 défaille, cela provoque l'activation de la porte sp_1 puisqu'il n'y a plus d'autres composants à solliciter. La défaillance de C_3 active la porte sp_2 . Le chronogramme de droite présente la situation opposée.

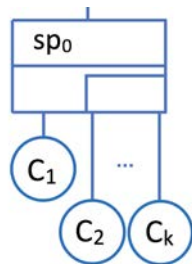


FIGURE 3.9 – Porte SPARE sp_0 .

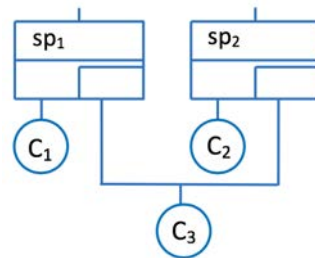


FIGURE 3.10 – Portes SPARE sp_1 et sp_2 .

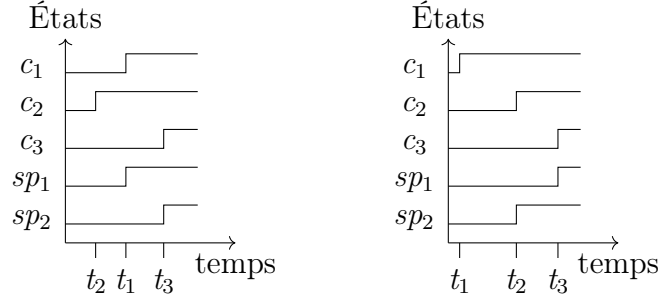


FIGURE 3.11 – Chronogrammes de la porte SPARE de la Figure 3.10.

Portes SPARE avec entrées partagée et non partagée

La Figure 3.12 contient deux portes SPARE. La porte sp_3 peut solliciter C_3 et la porte sp_4 peut solliciter C_3 (qui est partagé) et C_4 . L'enjeu réside dans l'attribution de C_3 , car l'attribution de C_3 à la porte sp_4 prive la porte sp_3 de composant de rechange.

La Figure 3.13 présente les chronogrammes de deux scénarios possibles. Sur le chronogramme de gauche, lorsque le composant C_2 défaille, la porte sp_4 sollicite le composant C_3 . Puisque le composant C_3 est déjà utilisé, la défaillance de C_1 provoque l'activation de la porte sp_3 . Lorsque le composant C_3 défaille, la porte sp_4 sollicite le composant C_4 , dont la défaillance provoque ensuite l'activation de la porte sp_4 . Le chronogramme de droite illustre un scénario où le composant C_3 est sollicité par la porte sp_3 après la défaillance de C_1 . La porte sp_4 sollicite C_4 lorsque C_2 défaille. Les portes sp_3 et sp_4 sont modélisées par les équations (3.23)-(3.24), et respectivement (3.25)-(3.26).

$$c_{sp_3} = c_1 \wedge (c_3 \vee (c_2 \wedge t_2 < t_1)) \quad (3.23)$$

$$t_{sp_3} = \begin{cases} t_1 & \text{si } t_2 < t_1 \\ \max(t_1, t_3) & \text{sinon} \end{cases} \quad (3.24)$$

$$c_{sp_4} = c_2 \wedge c_4 \wedge c_3 \vee (c_1 \wedge t_1 < t_2) \quad (3.25)$$

$$t_{sp_4} = \begin{cases} \max(t_2, t_3, t_4) & \text{si } t_2 < t_1 \\ \max(t_2, t_4) & \text{sinon} \end{cases} \quad (3.26)$$

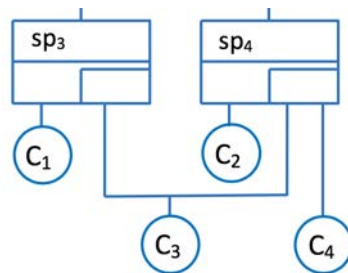


FIGURE 3.12 – Portes SPARE avec un composant partagé et un composant non partagé.

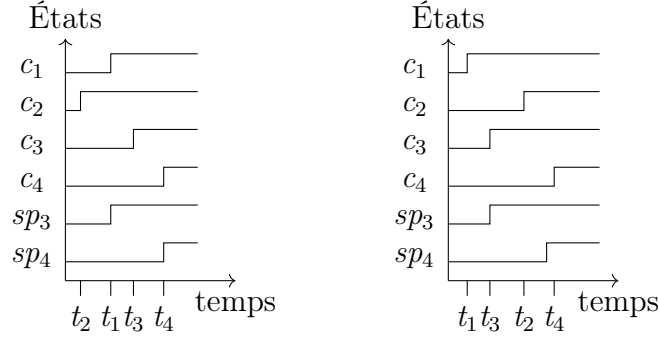


FIGURE 3.13 – Chronogrammes de la porte SPARE de la Figure 3.12.

Pour obtenir ces expressions, il est nécessaire de déterminer les conditions auxquelles un composant ne peut plus être sollicité par une porte. Les composants C_2 et C_4 ne pouvant être sollicités que par la porte sp_4 , ils ne peuvent plus être sollicités dès lors qu'ils ont défailli. Quant à C_3 , il ne peut plus être sollicité par la porte sp_4 s'il a défailli ou s'il est déjà utilisé par la porte sp_3 . Le second cas tient si le composant C_1 défaille avant que le composant C_2 défaille, et cela justifie l'équation (3.25).

Puisqu'il n'existe pas de restriction concernant le partage de composants par des portes SPARE, l'affectation des composants de rechange dépend de l'ordre d'occurrence des défaillances. Nous proposons de généraliser l'obtention de l'expression des portes SPARE.

Généralisation des portes Spare avec composants partagés

Afin de déterminer de manière systématique l'expression d'une porte SPARE γ , nous proposons l'introduction d'une fonction de répartition des composants de rechange $f(\gamma, i)$, où i est l'identifiant du composant C_i . Cette fonction permet de déterminer les conditions selon lesquelles le composant C_i est sollicité par la porte γ .

Nous définissons la fonction de répartition des composants de rechange par récurrence. Pour pouvoir l'exprimer, il est nécessaire qu'il n'existe pas d'interdépendance entre les composants. Pour nous en assurer, nous réalisons un graphe de dépendance pour les entrées des portes SPARE, dans lequel il est nécessaire qu'il n'y ait pas de boucle.

Définition 3.11. Un *graphe de dépendance* est un graphe orienté dont les nœuds sont les composants reliés à des portes SPARE. Deux composants C_i et C_j sont reliés par un arc allant de C_i vers C_j s'ils sont reliés à une même porte et si C_j suit C_i dans l'ordre de sollicitation par cette porte SPARE.

Exemple 3.12. Considérons l'arbre de défaillances de la Figure 3.14, dont le graphe de dépendance est la Figure 3.15. Le graphe de dépendance de ce système contient un cycle entre les nœuds représentant les composants C_3 et C_4 , ce qui ne permet pas d'utiliser la fonction de répartition des composants de rechange, car la fonction $f(sp_5, 3)$ serait définie en fonction de $f(sp_5, 4)$ et la fonction $f(sp_6, 4)$ serait définie en fonction de $f(sp_6, 3)$.

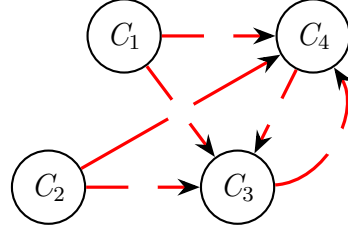
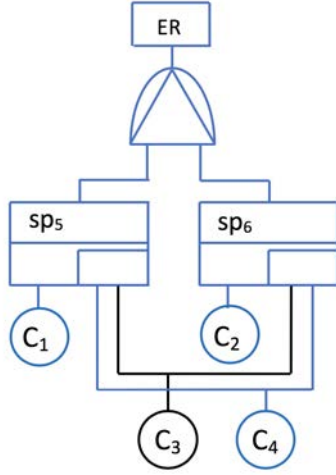


FIGURE 3.14 – Arbre de défaillances. FIGURE 3.15 – Graphe de dépendance.

Définition 3.13. La fonction de répartition des composants de rechange $f(\gamma, i)$ associe à l'identifiant i d'un composant C_i et à la porte γ , l'expression déterminant les conditions auxquelles la porte γ sollicite le composant C_i .

Si le composant C_i est un composant non-partagé de la porte γ , alors la porte γ est la seule pouvant solliciter ce composant. On obtient donc $f(\gamma, i) = 1$.

Pour traiter les composants partagés, nous introduisons la notation Δ_i afin de désigner l'ensemble des portes SPARE qui peuvent solliciter le composant C_i . Pour les portes sollicitant plusieurs composants, nous introduisons la notation \mathcal{C}_γ qui contient l'ensemble des (identifiants de) composants que peut solliciter la porte γ . Enfin, nous introduisons l'opérateur $<_\gamma$ pour déterminer l'ordre de sollicitation des composants par la porte γ . L'expression $j <_\gamma i$ signifie que C_j précède C_i dans l'ordre de sollicitation des composants par la porte γ et cela peut être déterminé grâce au graphe de dépendance.

La fonction de répartition des composants de rechange $f(\gamma, i)$ est déterminée par l'équation (3.27).

$$f(\gamma, i) = \left(\bigwedge_{j <_\gamma i} (c_j \vee \bigvee_{\delta \in \Delta_j \setminus \gamma} f(\delta, j)) \right) \wedge \bigwedge_{j <_\gamma i} \neg f(\gamma, j) \vee \bigwedge_{\delta \in \Delta_j \setminus \gamma} \bigvee_{l <_\delta i} (f(\delta, l) \wedge t_j < t_l) \quad (3.27)$$

La première ligne de l'équation (3.27) signifie que la porte γ sollicite le composant C_i si les composants que la porte γ peut solliciter avant C_i ont défailli ou ont été utilisés par une autre porte notée δ .

Lorsqu'un composant de rechange peut être sollicité par plusieurs portes, nous nous focalisons sur le composant qui défaille le premier parmi les composants actuellement utilisés. La porte γ utilise le composant C_i si le composant C_j actuellement utilisé par la porte γ défaille avant les composants utilisés par les portes δ (pouvant solliciter C_i), ce

qui correspond à la seconde ligne. Le terme $\bigvee_{l <_{\delta} i} (f(\delta, l) \wedge t_j < t_l)$ signifie que le composant C_j (utilisé par la porte γ) a défailli avant le composant C_l (utilisé par la porte δ). Par conséquent, le composant C_i est sollicité par la porte γ avant que la porte δ ne le sollicite. Le composant C_i est donc utilisé par la porte γ à condition que celle-ci sollicite ce composant avant les autres portes qui pourraient le solliciter.

Par conséquent, la porte γ s'active si et seulement si tous ses composants ont défailli ou ont été utilisés par une autre porte, ce qui est modélisé par l'équation (3.28).

L'instant d'activation de la porte γ est calculé par l'équation (3.29).

$$c_{\gamma} = \bigwedge_{i \in \mathcal{C}_{\gamma}} (c_i \vee \bigvee_{\delta \in \Delta_i \setminus \gamma} f(\delta, i)) \quad (3.28)$$

$$t_{\gamma} = \max_{i \in \mathcal{C}_{\gamma}} t(\gamma, i) \quad (3.29)$$

Pour déterminer l'instant auquel la porte γ est activée, il faut considérer les instants de défaillance des composants que la porte γ a sollicités. L'équation (3.30) exprime l'instant $t(\gamma, i)$, où t_i est l'instant de défaillance du composant C_i s'il est utilisé par la porte γ .

$$t(\gamma, i) = \begin{cases} t_i & \text{si } f(\gamma, i) \\ 0 & \text{sinon} \end{cases} \quad (3.30)$$

Exemple 3.14. Le système de la Figure 3.16 contient trois portes SPARE partageant deux composants de rechange. La Figure 3.17 est le graphe de dépendance acyclique de la Figure 3.16.

En utilisant l'équation (3.27), la fonction de répartition des composants est exprimée par récurrence par les équations (3.31)-(3.35), à partir des expressions des composants c_i et des variables de temps t_i .

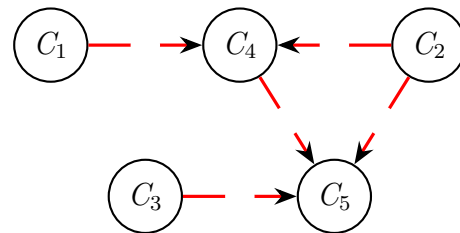
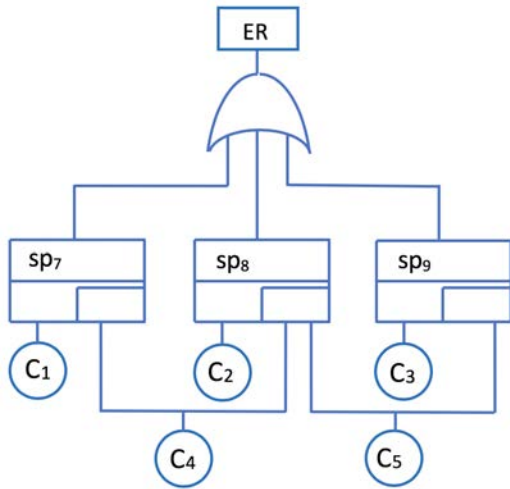


FIGURE 3.16 – Arbres de défaillances.

FIGURE 3.17 – Graphe de dépendance.

$$f(sp_7, 1) = True; \quad f(sp_8, 2) = True; \quad f(sp_9, 3) = True \quad (3.31)$$

$$\begin{aligned} f(sp_7, 4) &= c_1 \wedge (\neg f(sp_7, 1) \vee (f(sp_8, 2) \wedge t_1 < t_2)) \\ &= c_1 \wedge t_1 < t_2 \end{aligned} \quad (3.32)$$

$$\begin{aligned} f(sp_8, 4) &= c_2 \wedge (\neg f(sp_8, 2) \vee (f(sp_7, 1) \wedge t_2 < t_1)) \\ &= c_2 \wedge t_2 < t_1 \end{aligned} \quad (3.33)$$

$$\begin{aligned} f(sp_8, 5) &= c_2 \wedge (c_4 \vee f(sp_7, 4)) \\ &\quad \wedge f(sp_9, 3) \wedge t_2 < t_3 \\ &\quad \wedge \neg f(sp_8, 4) \vee (f(sp_9, 3) \wedge t_4 < t_3) \\ &= c_2 \wedge f(sp_7, 4) \wedge t_2 < t_3 \wedge (\neg f(sp_8, 4) \vee t_4 < t_3) \end{aligned} \quad (3.34)$$

$$\begin{aligned} f(sp_9, 5) &= c_3 \wedge (\neg f(sp_9, 3) \vee ((f(sp_8, 2) \wedge t_3 < t_2) \vee (f(sp_8, 4) \wedge t_3 < t_4))) \\ &= c_3 \wedge (t_3 < t_2 \vee (f(sp_8, 4) \wedge t_3 < t_4)) \end{aligned} \quad (3.35)$$

L'activation des portes sp_7 , sp_8 et sp_9 est modélisée par les équations (3.36)-(3.38) et l'instant d'activation est calculé par les équations (3.39)-(3.41).

$$c_{sp_7} = c_1 \wedge (c_4 \vee f(sp_8, 4)) \quad (3.36)$$

$$c_{sp_8} = c_2 \wedge (c_4 \vee f(sp_7, 4)) \wedge (c_5 \vee f(sp_9, 5)) \quad (3.37)$$

$$c_{sp_9} = c_3 \wedge (c_5 \vee f(sp_8, 5)) \quad (3.38)$$

$$t_{sp_7} = \begin{cases} \max(t_1, t_4) = t_4 & \text{si } f(sp_7, 4) \\ t_1 & \text{sinon} \end{cases} \quad (3.39)$$

$$t_{sp_8} = \begin{cases} \max(t_2, t_4, t_5) = \max(t_4, t_5) & \text{si } f(sp_8, 4) \wedge f(sp_8, 5) \\ \max(t_2, t_4) = t_4 & \text{si } \neg f(sp_8, 5) \\ \max(t_2, t_5) = t_5 & \text{si } \neg f(sp_8, 4) \\ t_2 & \text{sinon} \end{cases} \quad (3.40)$$

$$t_{sp_9} = \begin{cases} \max(t_3, t_5) = t_5 & \text{si } f(sp_9, 5) \\ t_3 & \text{sinon} \end{cases} \quad (3.41)$$

Jusqu'à présent, nous avons considéré les composants en redondance chaude, qui défont identiquement avant et après sollicitation. En redondance froide, ils ne peuvent pas défaillir avant d'être sollicités (leur état avant la sollicitation correspond à un arrêt).

Redondance froide

Dans le cas d'une redondance *froide*, nous ajoutons une contrainte à l'expression de la porte, afin de représenter le fait qu'un composant de rechange ne peut pas défaillir avant sollicitation.

Pour le système de la Figure 3.16, cette contrainte modélise le fait que C_4 ne peut pas défaillir avant sollicitation :

- soit C_4 a été sollicité pour remplacer C_1 : $f(sp_7, 4) \Leftrightarrow (c_1 \wedge c_4 \wedge t_1 < t_4)$,
- soit C_4 a été sollicité pour remplacer C_2 : $f(sp_8, 4) \Leftrightarrow (c_2 \wedge c_4 \wedge t_2 < t_4)$,
- soit C_4 n'a pas encore été sollicité, ce qui est représenté par : $\neg c_4$.

Par conséquent, l'expression (3.42) est ajoutée à la fonction de structure du système.

$$\begin{aligned} & (c_1 \wedge c_4 \wedge t_1 < t_4) \vee (c_2 \wedge c_4 \wedge t_2 < t_4) \vee \neg c_3 \\ \Leftrightarrow & c_3 \Rightarrow (c_1 \wedge c_4 \wedge t_1 < t_4) \vee (c_2 \wedge c_4 \wedge t_2 < t_4) \end{aligned} \quad (3.42)$$

Pour chaque composant de rechange en redondance froide, cette contrainte s'écrit sous la forme d'une disjonction en considérant le cas où il n'aurait pas été sollicité, ainsi que chaque composant qu'il peut avoir remplacé.

3.3.3 Interactions entre les portes dynamiques

Nous proposons de considérer les interactions entre la porte FDEP et les portes PAND et SPARE. Nous avons fait l'hypothèse que deux défaillances ne peuvent se produire simultanément.

D'une part, les portes FDEP mènent à des défaillances simultanées de composants. D'autre part, les portes PAND et SPARE nécessitent l'ordre d'occurrence des défaillances. Pour une porte PAND, cet ordre permet de déterminer si la porte est activée ; dans le cas de portes SPARE, cet ordre permet d'attribuer un composant de rechange à la première porte qui le sollicite.

Selon Ruijters et Stoelinga (2015), lorsque des composants utilisés par des portes SPARE ayant plusieurs entrées défaillent de manière simultanée, ces portes peuvent réclamer le même composant de rechange. Dans le cas du système de la Figure 3.19, si C_3 est fonctionnel et si C_1 et C_2 défaillent simultanément à cause de la porte FDEP, alors une porte parmi sp_1 ou sp_2 solliciterait C_3 et l'autre défaillirait. Si l'ordre de sollicitation des portes SPARE n'est pas déterministe, les outils d'analyse proposent une solution à ce non-déterminisme mais ces solutions sont différentes d'un outil à l'autre, provoquant des différences entre les analyses d'un même système.

Puisque nous souhaitons fournir une modélisation déterministe, un choix non déterministe (comme un tir aléatoire proposé par certains outils) ne nous convient pas lorsqu'il s'agit de déterminer l'attribution d'un composant de rechange, sollicité simultanément

par deux portes. Par conséquent, nous prenons comme hypothèses que :

- Les différentes entrées d'une porte FDEP ne doivent pas être situées dans plusieurs branches (i.e. entrées constituées d'une porte) d'une porte PAND (Figure 3.18) ;
- Les différentes entrées d'une porte FDEP ne doivent pas être des entrées de portes SPARE partageant un composant de rechange (ou plusieurs) (Figure 3.19).

Si un modèle ne respectait pas ces hypothèses, alors nous proposerions de modéliser ce processus différemment pour que cette modélisation concorde avec nos hypothèses.

De cette façon, nous évitons deux situations : deux portes SPARE sollicitant le même composant et deux entrées d'une porte PAND défaillant simultanément.

Les exemples des Figures 3.18 et 3.19 ne sont pas considérés par notre approche.

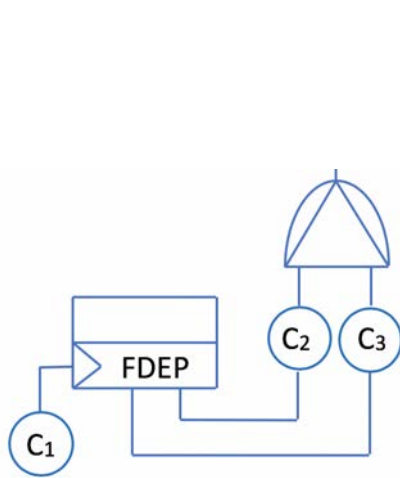


FIGURE 3.18 – Portes FDEP et PAND.

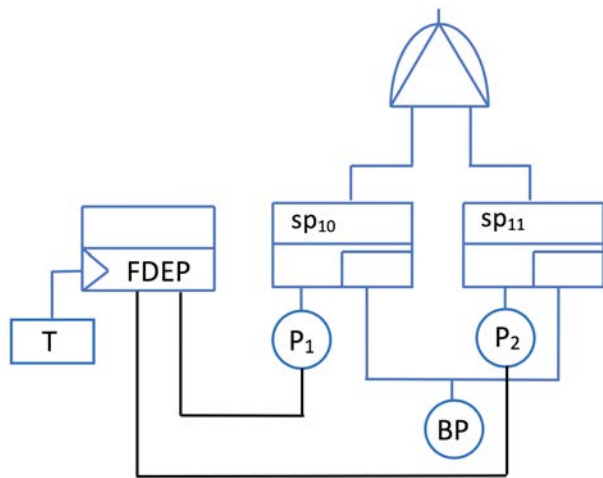


FIGURE 3.19 – Portes FDEP et SPARE.

3.3.4 Obtention d'une fonction de structure booléenne

À partir de l'expression de c_γ et t_γ pour chaque porte, nous obtenons la fonction de structure, qui est exprimée grâce aux variables booléennes c_i , modélisant l'état des composants, et aux variables de temps t_i , représentant l'instant de défaillance de ces composants.

Afin d'utiliser les techniques de satisfiabilité pour analyser la fonction de structure, elle doit être exprimée sous une forme booléenne. Par conséquent, nous transformons les équations conditionnelles et les opérateurs minimum et maximum en expressions booléennes. La variable t_i n'étant pas booléenne, nous proposons par la suite une variable booléenne $t_{i,j}$ décrivant l'ordre d'occurrence entre deux défaillances.

3.3.5 Gestion des équations conditionnelles

Nous nous focalisons tout d'abord sur les équations conditionnelles et proposons de les transformer en les considérant comme des disjonctions de cas.

Exemple 3.15. Considérons le système de la Figure 3.20. Pour déterminer s'il fonctionne, il est nécessaire de déterminer l'ordre d'occurrence des défaillances des portes sp_3 et sp_4 , dont les composants de rechange sont en redondance chaude. La porte sp_4 a été activée après sp_3 si l'équation (3.43) est vraie.

$$c_{sp_3} \wedge c_{sp_4} \wedge t_{sp_3} < t_{sp_4} \quad (3.43)$$

Les instants d'activation des portes sp_3 et sp_4 sont rappelés :

$$t_{sp_3} = \begin{cases} t_1 & \text{si } t_2 < t_1 \\ \max(t_1, t_3) & \text{sinon} \end{cases} \quad t_{sp_4} = \begin{cases} \max(t_2, t_3, t_4) & \text{si } t_2 < t_1 \\ \max(t_2, t_4) & \text{sinon} \end{cases}$$

Pour exprimer $t_{sp_3} < t_{sp_4}$, nous transformons les équations conditionnelles en une disjonction, selon l'ordre d'occurrence des défaillances de C_1 et C_2 .

Nous remplaçons ensuite t_{sp_3} et t_{sp_4} et obtenons l'équation (3.44).

$$\begin{aligned} t_{sp_3} < t_{sp_4} &= t_2 < t_1 \wedge t_{sp_3} < t_{sp_4} \\ &\vee t_2 > t_1 \wedge t_{sp_3} < t_{sp_4} \\ &= t_2 < t_1 \wedge t_1 < \max(t_2, t_3, t_4) \\ &\vee t_2 > t_1 \wedge \max(t_1, t_3) < \max(t_2, t_4) \end{aligned} \quad (3.44)$$

Les équations conditionnelles ont été ôtées. Néanmoins, cette expression contient des variables non booléennes et les opérateurs "max" et "<".

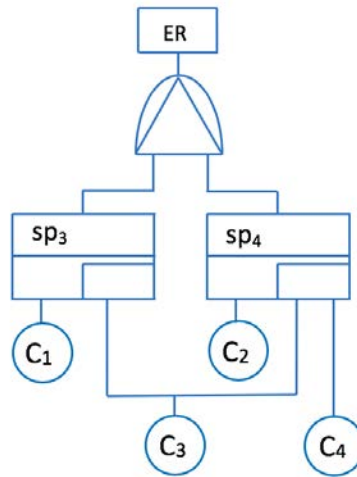


FIGURE 3.20 – Portes PAND et SPARE avec un composant partagé et un non partagé.

3.3.6 Suppression des opérateurs minimum et maximum

Les opérateurs minimum et maximum apparaissent au sein d'inégalités entres des variables de temps. Afin de s'affranchir des opérateurs *minimum* et *maximum*, nous pro-

posons quatre règles de réécriture (3.45)-(3.48), selon la position de l'opérateur.

$$\min(x, y) < z \Leftrightarrow x < z \vee y < z \quad (3.45)$$

$$\max(x, y) < z \Leftrightarrow x < z \wedge y < z \quad (3.46)$$

$$z < \min(x, y) \Leftrightarrow z < x \wedge z < y \quad (3.47)$$

$$z < \max(x, y) \Leftrightarrow z < x \vee z < y \quad (3.48)$$

Exemple 3.16. En utilisant l'équation (3.48), nous modifions l'équation (3.44) et obtenons l'équation (3.49). Nous obtenons l'équation (3.50) grâce à l'équation (3.46).

$$\begin{aligned} t_{sp_3} < t_{sp_4} = & \quad t_2 < t_1 \wedge (t_1 < t_2 \vee t_1 < t_3 \vee t_1 < t_4) \\ & \vee \quad t_2 > t_1 \wedge (\max(t_1, t_3) < t_2 \vee \max(t_1, t_3) < t_4) \end{aligned} \quad (3.49)$$

$$\begin{aligned} t_{sp_3} < t_{sp_4} = & \quad t_2 < t_1 \wedge (t_1 < t_2 \vee t_1 < t_3 \vee t_1 < t_4) \\ & \vee \quad \cancel{t_2 > t_1} \wedge (t_1 < t_2 \wedge t_3 < t_2) \\ & \vee \quad t_2 > t_1 \wedge (t_1 < t_4 \wedge t_3 < t_4) \end{aligned} \quad (3.50)$$

Nous factorisons ensuite l'expression de $t_{sp_3} < t_{sp_4}$ en exprimant toutes les inégalités avec l'opérateur "<" (et non ">"). En supprimant les clauses absurdes ou redondantes, nous obtenons l'équation (3.51).

$$\begin{aligned} t_{sp_3} < t_{sp_4} = & \quad t_2 < t_1 \wedge (\cancel{t_1 < t_2} \vee t_1 < t_3 \vee t_1 < t_4) \\ & \vee \quad t_1 < t_2 \wedge (t_1 < t_4 \vee t_3 < t_4) \end{aligned} \quad (3.51)$$

L'équation (3.52) donne une expression factorisée de $t_{sp_3} < t_{sp_4}$ en fonction de t_1, t_2, t_3, t_4 .

$$\begin{aligned} t_{sp_3} < t_{sp_4} = & \quad t_2 < t_1 \wedge (t_1 < t_3 \vee t_1 < t_4) \\ & \vee \quad t_1 < t_2 \wedge (t_1 < t_4 \vee t_3 < t_4) \end{aligned} \quad (3.52)$$

3.3.7 Suppression des variables non booléennes

Après la transformations des définitions conditionnelles en disjonctions, la constante $+\infty$ peut apparaître dans les formules et doit être supprimée : $\forall i, t_i < +\infty \Rightarrow \top$.

La variable t_i n'étant pas booléenne, nous proposons une variable booléenne $t_{i,j}$ décrivant l'ordre d'occurrence entre deux défaillances, qui représente l'inégalité $t_i < t_j$.

Exemple 3.17. En utilisant les variables booléennes $t_{i,j}$, l'équation (3.52) devient (3.53).

$$\begin{aligned} t_{sp_3, sp_4} = & \quad t_{2,1} \wedge (t_{1,3} \vee t_{1,4}) \\ & \vee \quad t_{1,2} \wedge (t_{1,4} \vee t_{3,4}) \end{aligned} \quad (3.53)$$

Puisque les composants ne défont pas simultanément, nous obtenons deux variables

redondantes $t_{i,j}$ et $t_{j,i}$ pour chaque couple de composants (i, j) , car $t_{i,j} = \neg t_{j,i}$.

Afin de minimiser le nombre de variables, nous tirons profit de la numérotation des composants et nous choisissons d'utiliser uniquement les variables $t_{i,j}$ où $i < j$. Par conséquent, nous représentons les inégalités $t_i < t_j$ et $t_j < t_i$ par $t_{i,j}$ et $\neg t_{i,j}$, respectivement.

L'équation (3.54) donne $t_{sp3} < t_{sp4}$ en fonction des variables $t_{1,2}, t_{1,3}, t_{1,4}, t_{2,3}, t_{3,4}$.

$$\begin{aligned} t_{sp3} < t_{sp4} = & \quad \neg t_{1,2} \wedge (t_{1,3} \vee t_{1,4}) \\ & \vee \quad t_{1,2} \wedge (t_{1,4} \vee t_{3,4}) \end{aligned} \quad (3.54)$$

La variable de temps $t_{i,j}$ est soumise à la règle de transitivité suivante :

$$\forall(i, j, k), (t_{i,j} \wedge t_{j,k} \Rightarrow t_{i,k}) \wedge (\neg t_{i,j} \wedge t_{i,k} \Rightarrow t_{j,k}).$$

À partir de la fonction de structure exprimée sous une forme booléenne, nous pouvons faire appel aux techniques de satisfiabilité. Cependant, puisque la fonction de structure a été obtenue grâce à l'arbre de défaillances, elle exprime la défaillance du système. Un appel au solveur SAT fournirait un ensemble de défaillances parmi les composants garantissant la défaillance du système. Pour déterminer des configurations assurant le fonctionnement du système, nous proposons de dualiser la fonction de structure.

3.3.8 Dualisation de la fonction de structure

La fonction de structure d'un arbre de défaillances dynamique associe à une configuration, un couple (c, t) où

- c est l'expression booléenne exprimant la défaillance du système, et
- t est l'instant de défaillance du système.

En fournissant l'expression c à un solveur SAT, il produirait une coupe. Puisque nous cherchons les liens et que le lien est le dual de la coupe, nous considérons le dual de la fonction de structure, correspondant à son complément

$$\bar{f} = \neg f.$$

Les littéraux⁴ de la fonction de structure, construits à partir de c_i ou $t_{i,j}$ sont donc remplacés par leurs compléments

$$x_i = \neg c_i \text{ et } t_{j,i} = \neg t_{i,j}$$

où C_i fonctionne lorsque $x_i = 1$. Par la suite, nous utilisons uniquement les variables x_i et $t_{i,j}$ (avec $i < j$), et leurs compléments.

4. Les littéraux sont les variables booléennes et leurs négations.

Exemple 3.18. Nous illustrons la dualisation de la fonction de structure, en considérant la porte sp_3 de la Figure 3.20 dont l'expression booléenne est donnée par l'équation (3.55)

$$c_{sp_3} = c_1 \wedge (c_3 \vee (c_2 \wedge \neg t_{1,2})) \quad (3.55)$$

L'expression booléenne dualisée de la porte sp_3 est donnée par l'équation (3.56).

$$\begin{aligned} x_{sp_3} &= \neg(c_1 \wedge (c_3 \vee (c_2 \wedge \neg t_{1,2}))) \\ &= \neg c_1 \vee (\neg c_3 \wedge \neg(c_2 \wedge \neg t_{1,2})) \\ &= \neg c_1 \vee (\neg c_3 \wedge (\neg c_2 \vee t_{1,2})) \\ &= x_1 \vee (x_3 \wedge (x_2 \vee t_{1,2})) \end{aligned} \quad (3.56)$$

Nous obtenons également $x_{sp_4} = x_2 \vee (x_3 \wedge (x_1 \vee \neg t_{1,2})) \vee x_4$.

Exemple 3.19. Écrivons l'expressions de chaque porte de l'arbre de défaillances de l'exemple 3.1.

$$c_{OU} = c_{ET} \vee c_{PAND} \vee c_{sp_1} \vee c_{sp_2} \quad (3.57)$$

$$c_{ET} = (c_2 \vee c_1) \wedge (c_3 \vee c_1) \quad (3.58)$$

$$c_{PAND} = c_4 \wedge c_5 \wedge t_{4,5} \quad (3.59)$$

$$c_{sp_1} = c_6 \wedge (c_8 \vee (c_7 \wedge \neg t_{6,7})) \quad (3.60)$$

$$c_{sp_2} = c_7 \wedge c_9 \wedge (c_8 \vee (c_7 \wedge t_{6,7})) \quad (3.61)$$

La fonction de structure de ce système est donnée par l'équation (3.62).

$$\begin{aligned} f &= c_1 \vee (c_2 \wedge c_3) \\ &\vee c_4 \wedge c_5 \wedge t_{4,5} \\ &\vee c_6 \wedge (c_8 \vee (c_7 \wedge \neg t_{6,7})) \\ &\vee c_7 \wedge c_9 \wedge (c_8 \vee (c_7 \wedge t_{6,7})) \end{aligned} \quad (3.62)$$

La fonction de structure dualisée de ce système est donnée par l'équation (3.63).

$$\begin{aligned} \neg f &= x_1 \wedge (x_2 \vee x_3) \\ &\wedge x_4 \vee x_5 \vee \neg t_{4,5} \\ &\wedge x_6 \vee (x_8 \wedge (x_7 \vee t_{6,7})) \\ &\wedge x_7 \vee x_9 \vee (x_8 \wedge (x_7 \vee \neg t_{6,7})) \end{aligned} \quad (3.63)$$

La fonction de structure dualisée est exprimée sous forme booléenne. Puisqu'un lien ne décrit que les états des composants du système, nous proposons de définir une extension des liens aux systèmes dynamiques, afin d'inclure l'ordre d'occurrence des défaillances ⁵.

5. Seules les variables de temps de la fonction de structure sont nécessaires pour l'analyse du système.

3.4 Calcul des séquences de lien minimales

3.4.1 Définition des séquences de lien

La fonction de structure dualisée d'un système dynamique contient :

- des variables booléennes x_i décrivant l'état des composants, et
- des variables booléennes $t_{i,j}$ décrivant l'ordre d'occurrence des défaillances.

Il est nécessaire d'adapter la notion de lien aux systèmes dynamiques, afin de prendre en compte l'ordre d'occurrence des défaillances. Nous définissons les séquences de lien minimales (MTSSs) comme l'extension des liens minimaux aux systèmes dynamiques.

Définition 3.20. Une *configuration séquentielle* est un ensemble de variables x_i et de littéraux construits à partir de variables $t_{i,j}$ (de la fonction de structure) tel que

$$\text{si } x_i \in TSS \text{ et } x_j \notin TSS, \text{ alors } t_{j,i} \in TSS \text{ ou } \neg t_{i,j} \in TSS.$$

Définition 3.21. Une *séquence de lien* (TSS) est une configuration séquentielle telle que, si les composants C_i (représentés par les variables $x_i \in TSS$) fonctionnent et si les ordres d'occurrence des défaillances $t_{i,j}$ (ou $t_{j,i}$) sont respectés, alors le système fonctionne. \square

Remarque 3.22. Le dual d'une séquence de lien est une séquence de coupe.

Une séquence de lien s'interprète comme un ensemble de composants dont le fonctionnement garantit que le système fonctionne, à condition que les défaillances aient eu lieu dans l'ordre donné lorsqu'un ordre est précisé.

Exemple 3.23. La fonction de structure dualisée de la Figure 3.20 est donnée par les équations (3.64)-(3.67).

$$\neg f = x_{sp_3} \vee x_{sp_4} \vee \neg t_{sp_3, sp_4} \quad (3.64)$$

$$x_{sp_3} = x_1 \vee (x_3 \wedge (x_2 \vee t_{1,2})) \quad (3.65)$$

$$x_{sp_4} = x_2 \vee (x_3 \wedge (x_1 \vee \neg t_{1,2})) \vee x_4 \quad (3.66)$$

$$t_{sp_3, sp_4} = (\neg t_{1,2} \wedge (t_{1,3} \vee t_{1,4})) \vee (t_{1,2} \wedge (t_{1,4} \vee t_{3,4})) \quad (3.67)$$

La séquence $\{x_3, x_4, t_{1,2}\}$ est une séquence de lien, car C_4 fonctionne.

Considérons $\{t_{1,2}, t_{3,4}\}$, où tous les composants ont défailli, C_1 avant C_2 et C_3 avant C_4 . L'ordre entre les défaillances de C_2 et C_3 n'est pas précisé. Puisque C_2 a défailli après C_1 , la porte sp_3 a sollicité C_3 et C_3 a défailli avant C_4 , ce qui implique que sp_3 a été activée avant sp_4 . Le système a défailli et $\{t_{1,2}, t_{3,4}\}$ est une séquence de coupe. \square

Nous définissons à présent la complétion et la minimalité.

Définition 3.24. Soit une configuration séquentielle T , elle est dite *complète* si elle contient la contrainte $t_{\alpha,\gamma}$ à chaque fois qu'elle contient les deux contraintes $t_{\alpha,\beta}$ et $t_{\beta,\gamma}$, quel que soit β , que les contraintes impliquent des littéraux positifs ou négatifs.⁶

Définition 3.25. Nous définissons la *complétion* T^* d'une configuration séquentielle T comme la plus petite (par rapport à l'inclusion ensembliste) TSS complète qui contient T . L'ensemble T^* peut être calculé en ajoutant des contraintes implicites jusqu'à ce qu'un point fixe soit atteint.

Exemple 3.26. Dans l'exemple 3.23, $\{x_3, x_4, t_{1,2}, t_{1,3}, t_{1,4}, t_{2,3}, t_{3,4}\}$ est une TSS complète, car les variables $t_{i,j}$ manquantes comme $t_{2,4}$ ne sont pas dans la fonction de structure.

Nous proposons de définir un ordre partiel sur les TSSs.

Définition 3.27. Deux TSSs T_1 et T_2 sont ordonnées et $T_1 \preceq T_2$ si et seulement si :

$$T_1^* \subseteq T_2^*. \quad (3.68)$$

Si les deux TSSs ne possèdent pas les mêmes variables de temps, nous prenons uniquement en compte les variables de temps pertinentes pour les deux TSSs afin de déterminer la complétion de chaque TSS. \square

Remarque 3.28. La relation \preceq entre les TSSs est partielle et deux TSSs peuvent être incomparables comme $\{x_3, t_{1,2}\}$ et $\{x_1\}$, ou $\{x_1, \neg t_{1,2}\}$ et $\{x_2, t_{1,2}\}$.

Exemple 3.29. Dans l'exemple 3.23, comparons les TSSs $\{x_1, x_2, x_3\}$ et $\{x_2, t_{1,2}\}$. La variable de temps $t_{1,2}$ n'est pas pertinente pour la TSS $\{x_1, x_2, x_3\}$, pour laquelle les composants C_1 et C_2 fonctionnent. Par conséquent, cela revient à comparer $\{x_1, x_2, x_3\}$ et $\{x_2\}$. Les TSSs $\{x_1, x_2, x_3\}$ et $\{x_2, t_{1,2}\}$ sont donc comparables :

$$\{x_2\} \preceq \{x_1, x_2, x_3\} \Rightarrow \{x_2, t_{1,2}\} \preceq \{x_1, x_2, x_3\}. \quad (3.69)$$

Remarque 3.30. Soit une séquence de lien. Si toute défaillance supplémentaire conduit à une séquence de coupe, alors cette séquence de lien est minimale.

3.4.2 Calcul des séquences de lien minimales

À partir de l'expression booléenne de la fonction de structure dualisée d'un système dynamique, nous faisons appel aux techniques de satisfiabilité. Notre objectif est d'utiliser le modèle⁷ généré par le solveur SAT pour en déduire une séquence de lien. Parmi ces

6. Seules les variables de temps de la fonction de structure peuvent être ajoutées par complétion.

7. Le modèle généré par un solveur SAT est une assignation de variables.

séquences de lien, nous identifions les séquences de lien minimales, symbolisant les configurations du système permettant d'évaluer la fiabilité du système grâce à une extension des travaux de Brînzei et Aubry (2018) aux systèmes dynamiques.

Nous proposons un algorithme qui utilise un solveur SAT et calcule les séquences de lien minimales à partir de la fonction de structure dualisée. Le principe est le même que celui de l'algorithme introduit précédemment dans le Chapitre 2 pour le format CNF. Nous l'adaptions aux systèmes dynamiques afin qu'il prenne en compte l'aspect temporel. Le principe de l'algorithme est présenté sur la Figure 3.21.

La fonction de structure dualisée (sous forme booléenne) est envoyée au solveur SAT qui génère un modèle. Afin que le solveur SAT produise un nouveau modèle, la formule envoyée au solveur SAT est modifiée en intégrant la négation du premier modèle et le solveur SAT est appelé à nouveau. Ce processus est répété tant que le solveur SAT renvoie un modèle. Lorsque le solveur SAT ne trouve plus de nouveaux modèles et qu'il renvoie UNSAT, nous déterminons les séquences de lien minimales en utilisant l'algorithme du chapitre 2, conçu pour les fonctions de structure au format DNF.

Lorsque le solveur SAT est appelé pour la première fois, si la fonction de structure est satisfiable, il renvoie un premier modèle m_1 . Le modèle obtenu peut contenir des variables booléennes x_i assignées à faux, écrites $\neg x_i$. Comme nous considérons des systèmes cohérents, la défaillance d'un composant ne contribue pas au fonctionnement du système. Cela implique qu'un composant défaillant ne permet pas le fonctionnement du système. Nous pouvons supprimer ce littéral du modèle sans le corrompre.

Par conséquent il est toujours possible de supprimer les littéraux $\neg x$ dans les modèles calculés par le solveur SAT, où x représente le fonctionnement d'un composant.

Quant aux variables de temps et leurs compléments présents dans les modèles m_i et m'_i , il se peut que le solveur SAT génère des instanciations superflues.

Remarque 3.31. Si le modèle contient le littéral x_i et le littéral $t_{i,j}$ (ou $\neg t_{j,i}$), alors le littéral $t_{i,j}$ (ou $\neg t_{j,i}$) peut être supprimé. En effet, puisque le modèle contient le littéral x_i , alors le composant C_i fonctionne et la variable $t_{i,j}$ est superflue.

Par conséquent, afin d'obtenir les modèles m'_i , nous supprimons des modèles m_i :

- les littéraux x_i assignées à faux, et
- les littéraux $t_{i,j}$ si $i < j$ (ou $\neg t_{j,i}$ si $j < i$) lorsque le modèle contient la variable x_i .

Lorsque plus aucun littéral ne peut être supprimé de m_i , le modèle réduit obtenu est noté m'_i et il représente une séquence de lien. Il ne contient que des variables de temps dont au moins le premier composant (du couple (i,j) de la variable $t_{i,j}$) a effectivement défailli, cf. exemple 3.33.

Remarque 3.32. Nous noterons $\neg m'_1$ la clause obtenue en appliquant la négation à l'intérieur du cube⁸ m'_1 .

8. Un cube est conjonction de littéraux, tandis qu'une clause est une disjonction de littéraux.

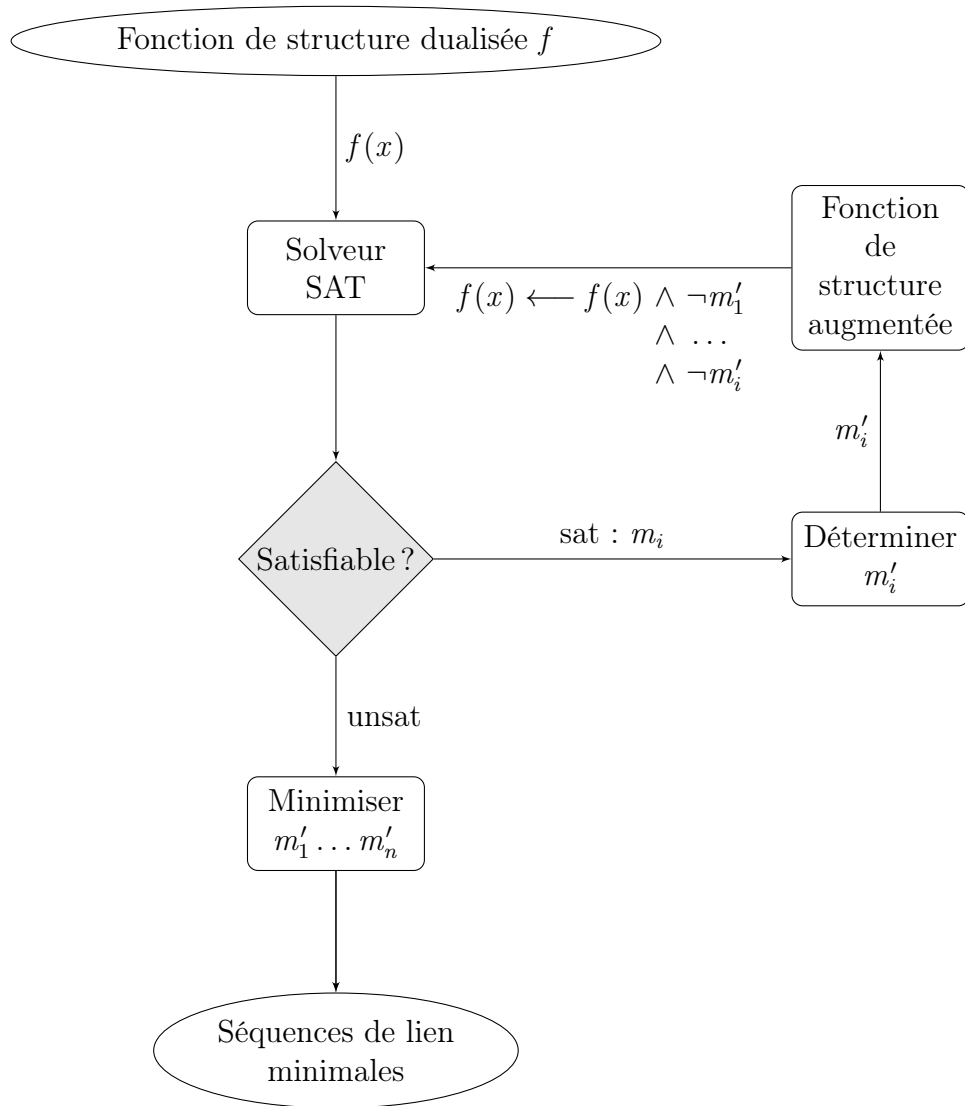


FIGURE 3.21 – Principe de l’algorithme d’obtention des MTSSs.

Exemple 3.33. Afin d'illustrer l'étape de réduction du modèle, nous supposons que le solveur génère le modèle suivant, à un certain point du calcul :

$$\{x_1, \neg x_2, \neg x_3, \neg t_{1,2}, t_{1,3}, \neg t_{2,3}\}.$$

Nous pouvons supprimer $\neg x_2$ et $\neg x_3$ puisque ce sont des variables x_i assignées à faux. Nous pouvons également supprimer $t_{1,3}$ puisque le composant C_1 fonctionne (x_1). Le résultat de l'étape de réduction est donc $\{x_1, \neg t_{1,2}, \neg t_{2,3}\}$ et nous ajoutons la clause suivante :

$$\neg(x_1 \wedge \neg t_{1,2} \wedge \neg t_{2,3}) = \neg x_1 \vee t_{1,2} \vee t_{2,3}$$

à la formule envoyée au solveur SAT pour l'appel suivant. □

Afin de calculer les TSSs suivantes, la clause $\neg m'_1$ est ajoutée à l'entrée envoyée au solveur SAT, afin qu'il détermine un modèle différent du précédent. Nous répétons l'opération jusqu'à ce que le solveur SAT déclare que l'entrée n'est pas satisfiable.

D'après le théorème 2.39, à ce point, nous avons calculé un ensemble \mathcal{T} de TSSs qui couvre l'ensemble des TSSs. Ainsi, pour chaque TSS T , \mathcal{T} contient une TSS $T' \preceq T$. Cependant, il contient des TSSs qui ne sont pas minimales. Il faut donc éliminer de \mathcal{T} toutes les TSSs T pour lesquelles \mathcal{T} contient un $T' \preceq T$. Cela correspond à l'étape *Minimiser* de la Figure 3.21 et reprend l'algorithme de minimisation du chapitre 2.

À partir des séquences de lien minimales obtenues, nous déterminons à présent le polynôme de fiabilité du système dans le but d'effectuer une étude quantitative.

3.5 Calcul de la fiabilité

L'analyse quantitative de la fiabilité d'un système repose sur le calcul du polynôme de fiabilité, que nous avons déterminé dans le Chapitre 2 en appliquant l'approche de Brânzei et Aubry (2015, 2018) basée sur le diagramme de Hasse.

Le diagramme de Hasse permet de tirer profit de l'ordre partiel régissant l'ensemble des configurations. Pour modéliser les systèmes dynamiques par des diagrammes de Hasse, nous proposons une extension des diagrammes de Hasse aux systèmes dynamiques, afin que les noeuds prennent en considération l'ordre d'occurrence des défaillances. Seules les variables de temps $t_{i,j}$ de la fonction de structure (et leur complément) sont considérés pour le diagramme de Hasse.

Dans cette section, nous utilisons les séquences de lien minimales obtenues grâce aux techniques de satisfiabilité, que nous hiérarchisons dans le diagramme de Hasse, afin d'obtenir le diagramme de Hasse restreint aux états de fonctionnement du système.

3.5.1 Diagramme de Hasse d'un système dynamique

Les noeuds du diagramme de Hasse d'un système statique contiennent autant de variables booléennes que le système contient de composants. Les composants fonctionnels sont représentés par des 1 et les composants défaillants par des 0.

- L'adaptation du diagramme de Hasse aux systèmes dynamiques nécessite de connaître :
- le nombre de composants du système, et
 - les ordres de défaillances $t_{i,j}$ de la fonction de structure.⁹

Définition 3.34. Le *diagramme de Hasse* d'un système dynamique non réparable est un graphe orienté constitué d'éléments ordonnés qui sont les *noeuds*, représentant les séquences de lien et les séquences de coupe, et d'*arcs* symbolisant la relation entre deux éléments, par un segment les reliant, en respectant l'ordre d'occurrence des défaillances.

Afin de les différencier, les séquences de coupe sont représentées par des noeuds non encadrés et les séquences de lien par des noeuds encadrés. Les séquences de lien minimales sont représentées par des noeuds encadrés et grisés.

Puisque les systèmes sont supposés non réparables, les arcs sont orientés vers le bas et représentent une défaillance. Un chemin est la représentation d'un scénario de défaillances.

Exemple 3.35. La Figure 3.23 représente le diagramme de Hasse de la porte PAND à deux entrées (Figure 3.22). Ses séquences de lien minimales sont : $\{x_2, t_{1,2}\}$, $\{\neg t_{1,2}\}$, respectivement représentées par les noeuds $\langle 01 \rangle \wedge t_{1,2}$ et $\langle 00 \rangle \wedge \neg t_{1,2}$.

L'*ordre* d'un noeud est le nombre de "1" qui composent le mot booléen.

Exemple 3.36. Le noeud $\langle 11 \rangle$ est d'ordre 2 et le noeud $\langle 01 \rangle \wedge t_{1,2}$ est d'ordre 1.

Remarque 3.37. Un noeud contient la variable $t_{i,j}$ (si $i < j$), ou $\neg t_{j,i}$ (si $j < i$), dès lors que C_i défaille, même si C_j n'a pas encore défailli, puisque l'ordre des défaillances entre C_j et C_i est fixé dès l'occurrence de la première défaillance du couple (i, j) .

Par exemple, le noeud $01 \wedge t_{1,2}$ de la Figure 3.23 contient la variable $t_{1,2}$, car le composant C_1 a défailli et que la défaillance de C_2 ne peut se produire qu'a posteriori.



FIGURE 3.22 – Porte PAND.

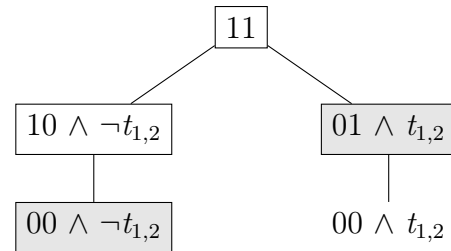


FIGURE 3.23 – Diagramme de Hasse.

9. Deux systèmes ayant un même nombre de composants et des ordres de défaillances différents ont des diagrammes de Hasse différents, comme démontré par les figures 3.24 et 3.25.

Remarque 3.38. Le diagramme de Hasse d'un système dynamique peut contenir plusieurs noeuds dont les mêmes composants sont défaillants et dont l'ordre d'occurrence des défaillances diffère, comme les noeuds $\langle 00 \rangle \wedge t_{1,2}$ et $\langle 00 \rangle \wedge \neg t_{1,2}$ de la Figure 3.23.

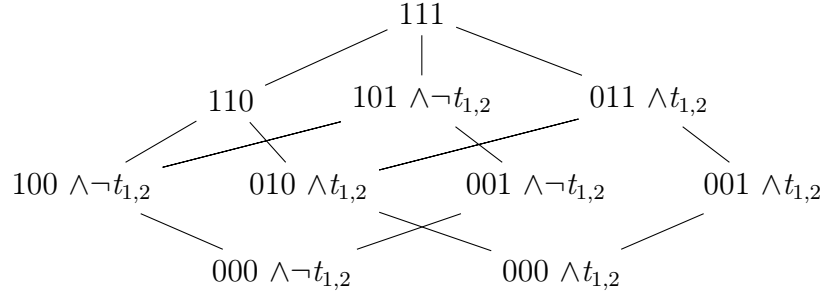


FIGURE 3.24 – Diagramme de Hasse d'un système à trois composants et une variable temporelle considérée $t_{1,2}$.

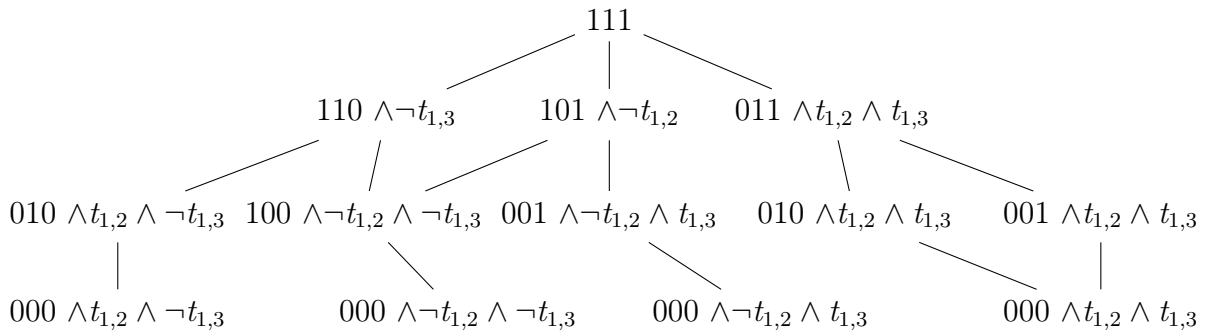


FIGURE 3.25 – Diagramme de Hasse d'un système à trois composants et deux variables de temps considérées $t_{1,2}$ et $t_{1,3}$.

Probabilité et redondance froide

Le calcul de probabilité dans le cas d'une porte SPARE dépend du type de redondance, chaude ou froide. Nous distinguons ceux qui ne défaillent pas avant sollicitation (froide) et ceux qui défaillent avec le même taux de défaillance avant et après sollicitation (chaude).

Pour un arbre de défaillances, transformer une redondance chaude en une redondance froide provoque la suppression de certains noeuds du diagramme de Hasse, car ce ne sont plus des états possibles du système.

Exemple 3.39. La Figure 3.27 est le diagramme de Hasse du système de la Figure 3.26. Les portes sp_1 et sp_2 étant des cold SPARE, le composant C_3 ne peut défaillir avant d'être sollicité. Par conséquent, le noeud $\langle 110 \rangle$ n'appartient pas au diagramme de Hasse.

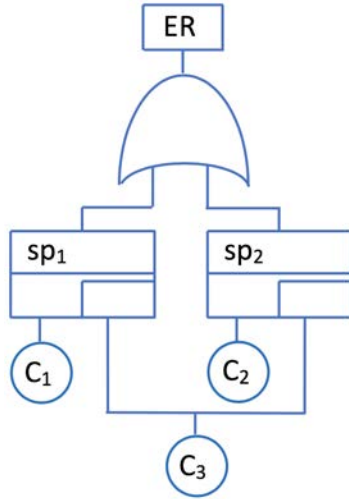


FIGURE 3.26 – DFT.

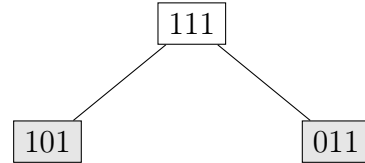


FIGURE 3.27 – Diagramme de Hasse.

Diagramme de Hasse et porte FDEP

La présence d'une porte FDEP impacte l'arbre de défaillances, puisque l'activation de la gâchette provoque l'activation instantanée des entrées de la porte FDEP. Dans le diagramme de Hasse, l'activation de la gâchette ne correspond donc pas à un arc mais à un chemin. Des états de transition (représentés en pointillé) permettent de représenter les défaillances. Nous supposons que le temps de séjour dans les états de transition est nul. L'état de transition est une aide à la construction. Il n'appartient pas au diagramme.

Exemple 3.40. L'arbre de défaillances de la Figure 3.28 contient une porte FDEP, dont la gâchette (C_1) conduit à la défaillance du composant C_2 lorsqu'elle est activée. Le diagramme de Hasse de ce système est représenté par la Figure 3.29.

En considérant l'état $\langle 1111 \rangle$, la défaillance de C_1 provoque la défaillance instantanée de C_2 et le noeud $\langle 0111 \rangle$ n'a qu'un seul arc (vers le bas), qui le relie au noeud $\langle 0011 \rangle$ où C_2 a défailli.

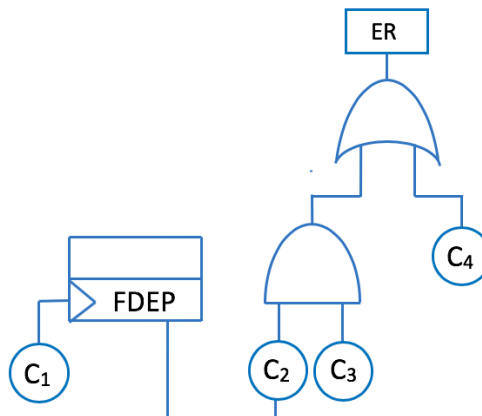


FIGURE 3.28 – Arbre de défaillances dynamique.

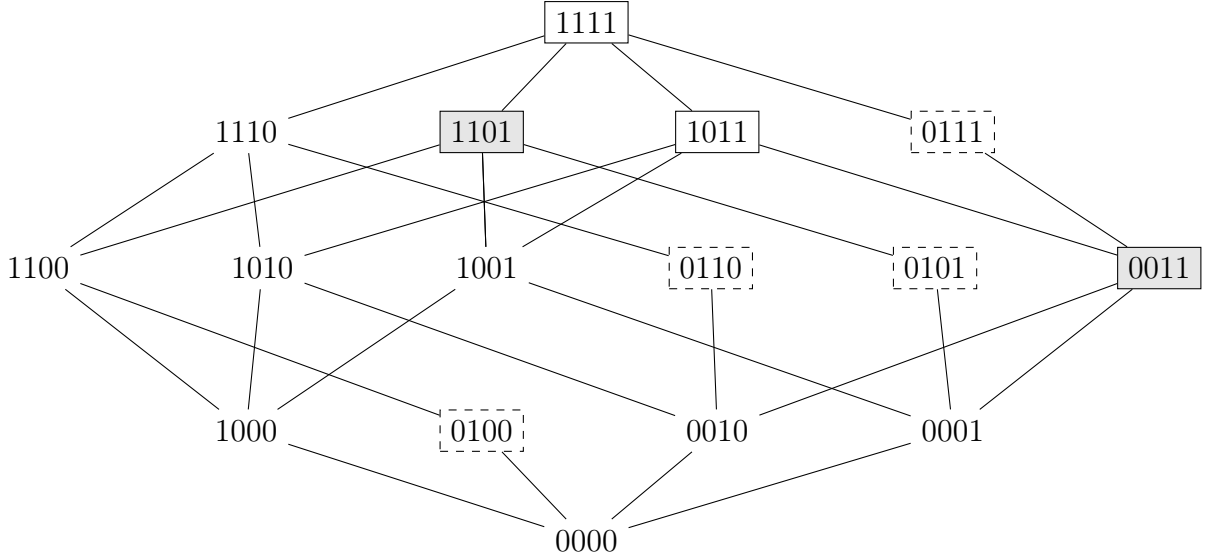


FIGURE 3.29 – Diagramme de Hasse.

Construction du diagramme de Hasse

Le diagramme de Hasse est construit à partir des séquences de lien minimales, calculées à l'aide des techniques de satisfiabilité. Seul le diagramme de Hasse restreint aux états de fonctionnement du système est nécessaire à l'obtention du polynôme de fiabilité.

Puisque les variables de temps précisent l'ordre d'occurrence des défaillances, elles doivent être respectées par les noeuds. La construction du diagramme de Hasse débute par les séquences de lien minimales, placées en fonction de leur ordre. À partir de chaque noeud sont créés les noeuds d'ordre supérieur (de 1) et directement reliés par un arc au noeud considéré, en respectant les contraintes temporelles dictées par les variables de temps¹⁰.

Exemple 3.41. La Figure 3.30 illustre la première étape de construction du diagramme de Hasse d'un système dont la fonction de structure est $f = (x_2 \vee \neg t_{1,2}) \wedge (x_3 \vee \neg t_{1,3})$ et dont les séquences de lien minimales sont :

$$x_3 \wedge \neg t_{1,2} \wedge t_{1,3}, \quad \neg t_{1,2} \wedge \neg t_{1,3}, \quad x_2 \wedge t_{1,2} \wedge \neg t_{1,3}, \quad x_2 \wedge x_3 \wedge t_{1,2} \wedge t_{1,3}. \quad (3.70)$$

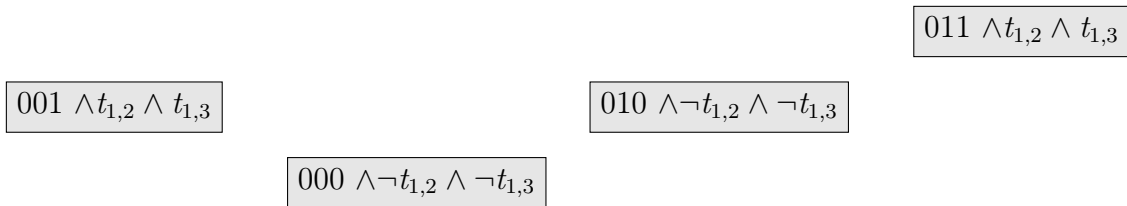


FIGURE 3.30 – Construction du diagramme de Hasse de l'exemple 3.41.

10. Il existe au plus autant de noeuds que celui-ci contient de composants défaillants.

Exemple 3.42. Considérons la séquence de lien $\langle 001 \rangle \wedge \neg t_{1,2} \wedge t_{1,3}$. Puisque les composants C_1 et C_2 sont défaillants et C_2 a défailli avant C_1 (d'après $\neg t_{1,2}$), la dernière défaillance a avoir eu lieu est C_1 . Le seul noeud d'ordre 2 relié à $\langle 001 \rangle \wedge \neg t_{1,2} \wedge t_{1,3}$ est : $\langle 101 \rangle \wedge \neg t_{1,2}$.

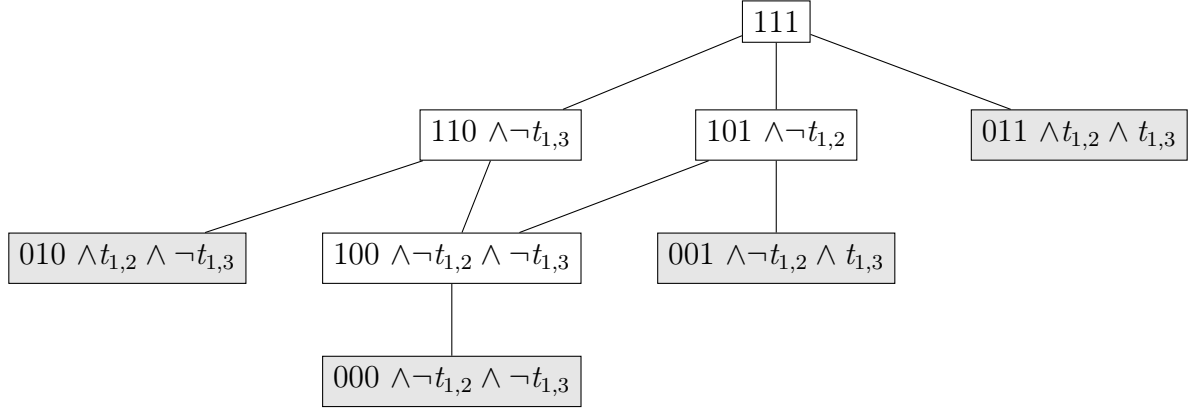


FIGURE 3.31 – Diagramme de Hasse d'un système à trois composants et deux variables de temps considérées $t_{1,2}$ et $t_{1,3}$.

Dans le cas des systèmes dynamiques, le diagramme de Hasse et le calcul de la fiabilité dépendent non seulement des états des composants, mais aussi de l'ordre d'occurrence des défaillances. Le calcul de fiabilité doit être adapté pour en tenir compte.

3.5.2 Attribution d'un poids à chaque noeud

Afin de déterminer le polynôme de fiabilité à partir du diagramme de Hasse, nous utilisons une pondération attribuée à chaque noeud. L'algorithme d'attribution des poids est identique à celui de la section 2.6.2.

Nous attribuons un poids à chaque noeud du diagramme de Hasse restreint et, lorsque tous les poids initiaux sont attribués, nous augmentons ou réduisons ces poids afin que chacun soit égal 1, afin d'obtenir le polynôme de fiabilité.

Exemple 3.43. La Figure 3.35 représente le diagramme de Hasse restreint d'un système, dont l'arbre de défaillances est donné par la Figure 3.32, où les composants sont ordonnés comme suit : $\langle P_1, P_2, BP \rangle$.

Remarque 3.44. Concernant les noeuds encadrés en pointillé où la gâchette de la porte FDEP vient de défailir, ils représentent un état de transition et sont supprimés (ou négligés) lors de l'attribution des poids. Nous ne leur attribuons donc pas de poids.

Exemple 3.45. La Figure 3.34 représente le diagramme de Hasse du système de la Figure 3.33. Les noeuds $\langle 110 \rangle$ et $\langle 100 \rangle$ sont des états de transition du système et il ne leur est pas attribué de poids.

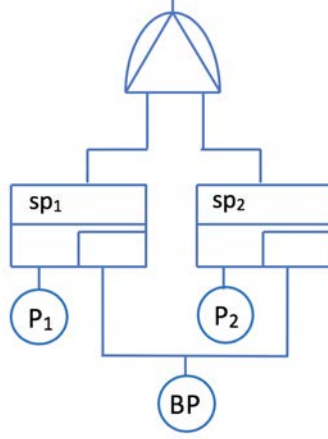


FIGURE 3.32 – Porte PAND et portes SPARE partageant un composant.

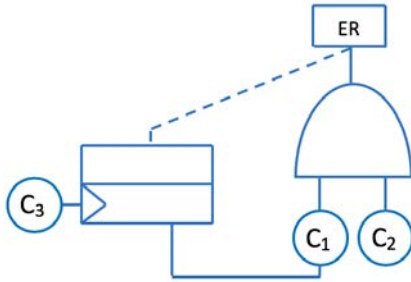


FIGURE 3.33 – Arbre de défaillances.

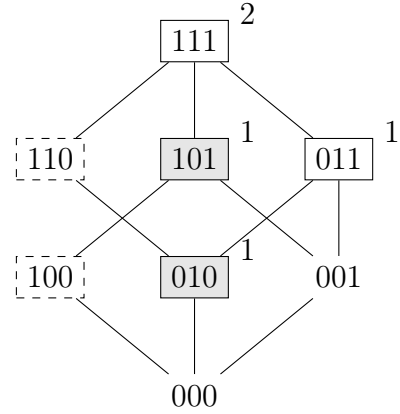


FIGURE 3.34 – Diagramme de Hasse.

3.5.3 Obtention du polynôme de fiabilité

L'attribution initiale des poids aux noeuds du diagramme de Hasse est effectuée. L'algorithme d'obtention du polynôme est identique à celui de la section 2.6.2.

Pour chaque noeud, si son poids vaut 1, il est supprimé du diagramme ; sinon, le poids du noeud (et celui des noeuds qu'il couvre) est augmenté (resp. diminué) de 1 si sa valeur est inférieure (resp. supérieure) à 1¹¹ et le monôme du noeud est ajouté (resp. soustrait) au polynôme de fiabilité autant de fois que le poids a été augmenté ou diminué. Cette phase s'achève lorsque tous les poids sont égaux à 1, ce qui signifie que nous avons obtenu le polynôme de fiabilité R .

Exemple 3.46. Le système des Figures 3.32 et 3.35, les noeuds représentant les séquences de lien minimales. L'ordre des composants est le suivant : $\langle P_1, P_2, BP \rangle$.

11. Durant cette phase, un poids peut être nul ou négatif.

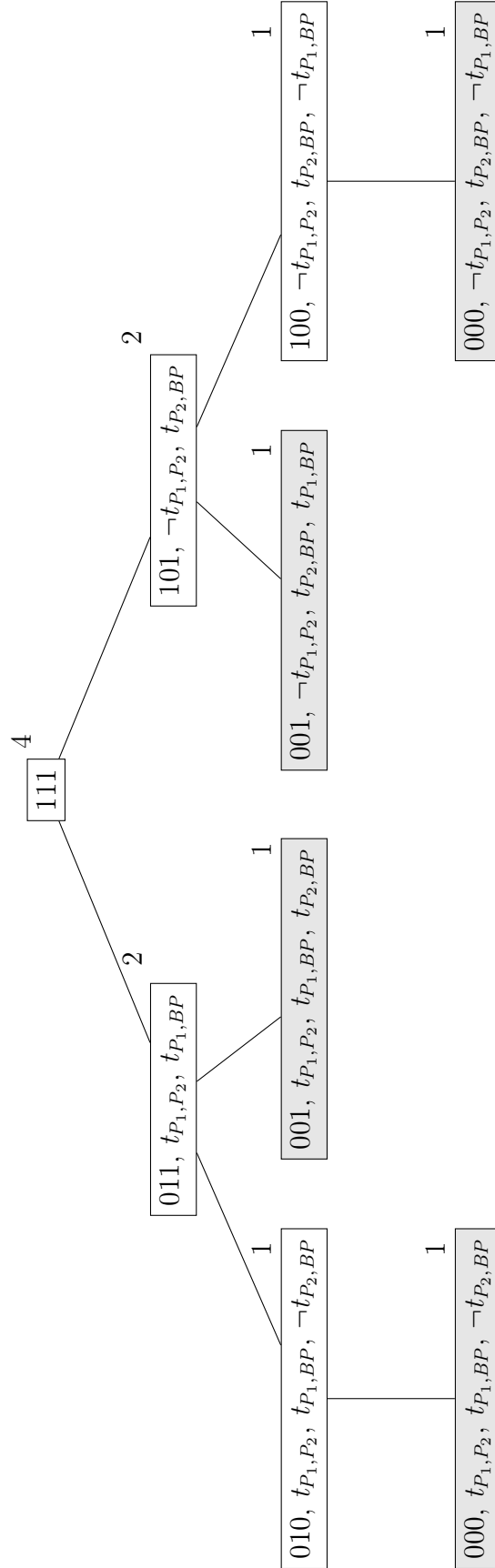


FIGURE 3.35 – Diagramme de Hasse du système de la Figure 3.32.

Nous obtenons l'équation (3.71) en ajoutant au polynôme de fiabilité R les monômes associés, et le diagramme devient la Figure 3.36 en supprimant ces noeuds.

$$R = P(000, t_{P_1, P_2}, t_{P_1, BP}, \neg t_{P_2, BP}) + P(000, \neg t_{P_1, P_2}, t_{P_2, BP}, \neg t_{P_1, BP}) \quad (3.71)$$

$$+ P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP}) + P(001, \neg t_{P_1, P_2}, t_{P_2, BP}, t_{P_1, BP})$$

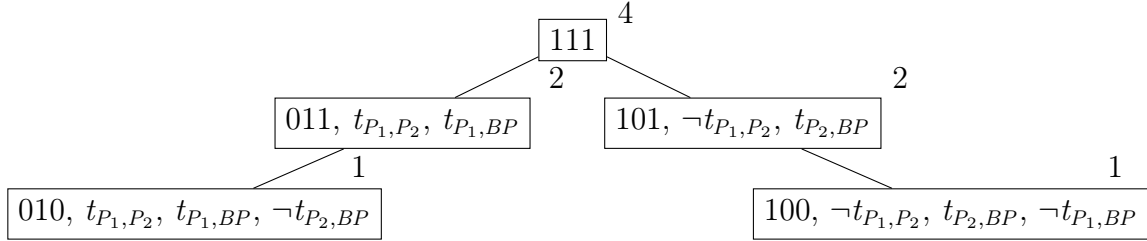


FIGURE 3.36 – Diagramme de Hasse après la première suppression.

Puisque les noeuds "010, $t_{P_1, BP}$ " et "100, $t_{P_2, BP}$ " ont un poids de 1, alors ils sont supprimés du diagramme de Hasse de la Figure 3.36. Nous obtenons la Figure 3.37.

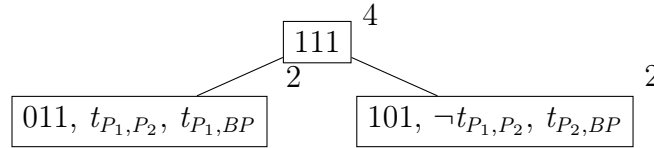


FIGURE 3.37 – Diagramme de Hasse après la deuxième suppression.

Les noeuds "011, $t_{P_1, P_2}, t_{P_1, BP}$ " et "101, $\neg t_{P_1, P_2}, t_{P_2, BP}$ " ont un poids de 2, leurs monômes sont chacun ôtés une fois de l'expression de R et nous obtenons l'équation (3.72) :

$$R = P(000, t_{P_1, P_2}, t_{P_1, BP}, \neg t_{P_2, BP}) + P(000, \neg t_{P_1, P_2}, t_{P_2, BP}, \neg t_{P_1, BP}) \quad (3.72)$$

$$+ P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP}) + P(001, \neg t_{P_1, P_2}, t_{P_2, BP}, t_{P_1, BP})$$

$$- R_{P_2} \cdot R_{BP} - R_{P_1} \cdot R_{BP}$$

Les poids de ces noeuds sont diminué de 1 et celui du noeud "111" est diminué de 2, une fois due à "011, $t_{P_1, P_2}, t_{P_1, BP}$ " et une fois due à "101, $\neg t_{P_1, P_2}, t_{P_2, BP}$ ". Ces noeuds ayant un poids de 1, ils sont supprimés. Le poids du noeud "111" est diminué de 1 afin d'être égal à 1 et le monôme associé est ôté une fois du polynôme de fiabilité. Le polynôme de fiabilité du système est l'équation 3.73.

$$R = P(000, t_{P_1, P_2}, t_{P_1, BP}, \neg t_{P_2, BP}) + P(000, \neg t_{P_1, P_2}, t_{P_2, BP}, \neg t_{P_1, BP}) \quad (3.73)$$

$$+ P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP}) + P(001, \neg t_{P_1, P_2}, t_{P_2, BP}, t_{P_1, BP})$$

$$- R_{P_2} \cdot R_{BP} - R_{P_1} \cdot R_{BP} - R_{P_1} \cdot R_{P_2} \cdot R_{BP}$$

Exemple 3.47. L'équation (3.74) est le polynôme de fiabilité du système de la Figure 3.26.

$$R_S = R_1 \cdot R_3 + R_2 \cdot R_3 - R_1 \cdot R_2 \cdot R_3 \quad (3.74)$$

Nous cherchons à évaluer la probabilité de chaque monôme du polynôme de fiabilité.

3.5.4 Calcul de probabilité d'un monôme

La fiabilité d'un composant C_i est noté R_i .

Si un monôme ne contient pas de variable de temps, sa probabilité s'exprime uniquement en fonction des variables R_i des composants C_i ; sinon, sa probabilité dépend des variables R_i et de l'ordre d'occurrence des défaillances.

3.5.4.1 Calcul de probabilité d'un monôme sans variable de temps

Le calcul de la probabilité d'un monôme sans variable temporelle est similaire à celui d'un monôme dans le diagramme de Hasse d'un système statique, dans la section 2.6.

Pour un monôme G , où les composants $C_i \in \mathbb{C}_1$ fonctionnent et les composants $C_i \in \mathbb{C}_2$ défaillent (\mathbb{C}_1 et \mathbb{C}_2 sont des partitions de \mathbb{C}^{12}), la probabilité de G est :

$$P(G) = \prod_{C_i \in \mathbb{C}_1} R_i.$$

De plus, rappelons que la probabilité du monôme associé à un nœud inclut la probabilité des monômes des nœuds qu'il couvre dans le diagramme de Hasse.

3.5.4.2 Calcul de probabilité d'un monôme avec une variable de temps

Lorsque le monôme contient une variable de temps $t_{i,j}$, il est nécessaire de prendre en compte l'ordre d'occurrence des défaillances des composants C_i et C_j .

D'après Brânzei et Aubry (2018)¹³, la probabilité du monôme associé à un nœud inclut la probabilité des monômes des nœuds qu'il couvre dans le diagramme de Hasse.

Nous proposons d'interpréter la variable $t_{i,j}$ dans un cube de la façon suivante :

- soit C_j a défailli après C_i ;
- soit C_j fonctionne.

Exemple 3.48. Ainsi, la probabilité $P(001, t_{1,2})$ signifie que :

- C_3 fonctionne, et
- soit C_2 a défailli après C_1 ; soit C_2 fonctionne.

12. \mathbb{C} est l'ensemble de tous les composants du système.

13. Les cubes y sont appelés mintermes et les clauses, maxtermes.

L'équation (3.75) donne la probabilité $P(001, t_{1,2})$.

$$\begin{aligned} P(001, t_{1,2}) &= P(C_3 \text{ fonctionne}) \cdot (P(C_2 \text{ fonctionne}) + P(C_2 \text{ a défailli après } C_1)) \\ &= R_3 \times (R_2 + P(\neg x_1, \neg x_2, t_{1,2})). \end{aligned} \quad (3.75)$$

De même, l'équation (3.76) donne la probabilité $P(001, \neg t_{1,2})$.

$$\begin{aligned} P(001, \neg t_{1,2}) &= P(C_3 \text{ fonctionne}) \cdot (P(C_1 \text{ fonctionne}) + P(C_1 \text{ a défailli après } C_2)) \\ &= R_3 \times (R_1 + P(\neg x_1, \neg x_2, \neg t_{1,2})). \end{aligned} \quad (3.76)$$

3.5.4.3 Calcul de probabilité d'un monôme avec plusieurs variables de temps

Lorsque le monôme contient plusieurs variables de temps, nous procédons de la même façon avec toutes les variables de temps, afin d'exprimer la probabilité du monôme en fonction des variables de temps et de la fiabilité des composants R_i .

Considérons le noeud $n_C = \langle 0010001 \rangle$ de la Figure 3.38 et notons que $i, j, k, l \notin \{2, 3\}$.

$$\begin{array}{cc} 3 & 7 \\ \uparrow & \uparrow \\ n_C = & 0010001, t_{i,j}, \neg t_{k,l} \end{array}$$

FIGURE 3.38 – MTSS avec contrainte de temps.

L'équation (3.77) permet de déterminer la probabilité du cube n_C .

$$\begin{aligned} P(n_C) &= (P(C_j \text{ fonctionne}) + P(C_j \text{ a défailli après } C_i)) \\ &\quad \cdot (P(C_k \text{ fonctionne}) + P(C_k \text{ a défailli après } C_l)) \\ &\quad \cdot P(C_7 \text{ fonctionne}) \\ &\quad \cdot P(C_3 \text{ fonctionne}). \end{aligned} \quad (3.77)$$

Ainsi, l'équation (3.78) donne la probabilité du monôme associé au noeud n_C .

$$P(n_C) = (R_j + P(\neg x_i, \neg x_j, t_{i,j})) \cdot (R_k + P(\neg x_k, \neg x_l, t_{l,k})) \cdot R_7 \cdot R_3. \quad (3.78)$$

Nous proposons de généraliser à un état N où les composants $C_i \in \mathbb{C}_1$ fonctionnent, les composants $C_i \in \mathbb{C}_2$ défaillent et les variables de temps t_{j_α, k_α} , où il y a ω variables de temps : $t_{j_1, k_1}, \dots, t_{j_\omega, k_\omega}$; $\mathbb{C} = \mathbb{C}_1 \cup \mathbb{C}_2$; $\mathbb{C}_1 \cap \mathbb{C}_2 = \emptyset$.

L'équation (3.79) est l'expression de la probabilité du monôme associé au noeud N .

$$P(N) = \prod_{C_i \in \mathbb{C}} R_i \times \prod_{\delta \in \{1, \dots, \omega\}, (j_\delta, k_\delta) \in (j, k)^\omega} (R_j + P(\neg x_{j_\delta}, \neg x_{k_\delta}, t_{j_\delta, k_\delta})) \quad (3.79)$$

3.5.5 Calcul de probabilité des variables de temps

La probabilité $P(000, t_{1,2}, t_{2,3})$ se calcule grâce à l'équation (3.80), où f_i désigne la densité de probabilité et F_i , la fonction de répartition.

$$P(t_{1,2}, t_{2,3}) = \int_0^t f_3(u) \cdot \left(\int_0^u f_2(v) \cdot F_1(v) \cdot dv \right) \cdot du \quad (3.80)$$

Dans l'équation (3.80), bien que plusieurs variables de temps fassent référence à un même composant C_2 , l'ordre d'occurrence des défaillances est déterminé :

C_1 a défailli avant C_2 , qui a défailli avant C_3 .

Si l'ordre d'occurrence des défaillances n'est pas unique, nous proposons de réaliser une disjonction, comme suit, où $i < j < k$: $P(t_{i,j}, t_{i,k}) = P(t_{i,j}, t_{j,k}) + P(t_{i,k}, \neg t_{j,k})$.

Exemple 3.49. Le polynôme de fiabilité de l'exemple 3.46 est le suivant :

$$\begin{aligned} R = & P(000, t_{P_1, P_2}, t_{P_1, BP}, \neg t_{P_2, BP}) + P(000, \neg t_{P_1, P_2}, t_{P_2, BP}, \neg t_{P_1, BP}) \\ & + P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP}) + P(001, \neg t_{P_1, P_2}, t_{P_2, BP}, t_{P_1, BP}) \\ & - R_{P_2} \cdot R_{BP} - R_{P_1} \cdot R_{BP} - R_{P_1} \cdot R_{P_2} \cdot R_{BP} \end{aligned}$$

Les composants ont des distributions exponentielles de taux de défaillance $\lambda = 5 \cdot 10^{-6} h^{-1}$.

$$f(t) = \lambda \cdot e^{-\lambda \cdot t} \quad F(t) = 1 - e^{-\lambda \cdot t}$$

L'équation (3.81) permet de calculer $P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP})$.

$$\begin{aligned} P &= P(001, t_{P_1, P_2}, t_{P_1, BP}, t_{P_2, BP}) \quad (3.81) \\ &= R_{BP} \cdot (R_{P_2} + P(\neg x_{P_1}, \neg x_{P_2}, \neg t_{P_1, P_2})) \\ &= R_{BP} \cdot (R_{P_2} + \int_0^t f_{P_2}(u) \cdot F_{P_1}(u) \cdot du) \\ &= R_{BP} \cdot (R_{P_2} + \int_0^t \lambda \cdot e^{-\lambda \cdot u} \cdot (1 - e^{-\lambda \cdot u}) \cdot du) \\ &= R_{BP} \cdot (R_{P_2} + \lambda \cdot \int_0^t e^{-\lambda \cdot u} \cdot du - \lambda \cdot \int_0^t e^{-2 \cdot \lambda \cdot u} \cdot du) \\ &= R_{BP} \cdot (R_{P_2} + \lambda \cdot \left(-\frac{1}{\lambda} \cdot e^{-\lambda \cdot t} + \frac{1}{\lambda} \right) - \lambda \cdot \left(-\frac{1}{2 \cdot \lambda} \cdot e^{-2 \cdot \lambda \cdot t} + \frac{1}{2 \cdot \lambda} \right)) \\ &= R_{BP} \cdot (R_{P_2} + (-e^{-\lambda \cdot t} + 1) - (-\frac{1}{2} \cdot e^{-2 \cdot \lambda \cdot t} + \frac{1}{2})) \end{aligned}$$

Dans la section suivante, nous illustrons notre approche par un exemple.

3.6 Application

L'approche présentée dans ce chapitre a pour objectif d'automatiser l'obtention des MTSSs pour des systèmes industriels, dont une génération manuelle serait longue et source d'erreurs. Nous proposons d'illustrer notre approche en l'appliquant à un Système d'Assistance Cardiaque (CAS : Cardiac Assist System) issu de Merle et al. (2014, 2016); Boudali et Dugan (2005), lui-même inspiré de Vemuri et al. (1999). Le système CAS a été conçu pour traiter les défaillances électriques et mécaniques du cœur.

Ce système se divise en 4 sous-systèmes : la gâchette, l'unité CPU, l'unité des moteurs et l'unité des pompes. La Figure 3.39 présente l'arbre de défaillances de ce système.

L'unité CPU est modélisée par une Warm SPARE (wsp) avec un CPU principal (P) et un CPU de rechange (B). La barre transversale du commutateur (CS) et le système de supervision (SS) représentent la gâchette. La défaillance de l'un ou l'autre cause la défaillance des deux CPU. Pour l'unité des moteurs, un composant au minimum parmi le moteur (M) et le câble (MC) doit fonctionner¹⁴. L'unité des pompes est constituée de deux Cold SPARE csp_1 et csp_2 . Chacune possède une pompe principale (P_1 , P_2) et elles partagent un composant redondant (BP). Pour que l'unité des pompes défaille, les 3 pompes (P_1 , P_2 et BP) doivent défaillir et la porte sp_1 doit défaillir avant la porte sp_2 .

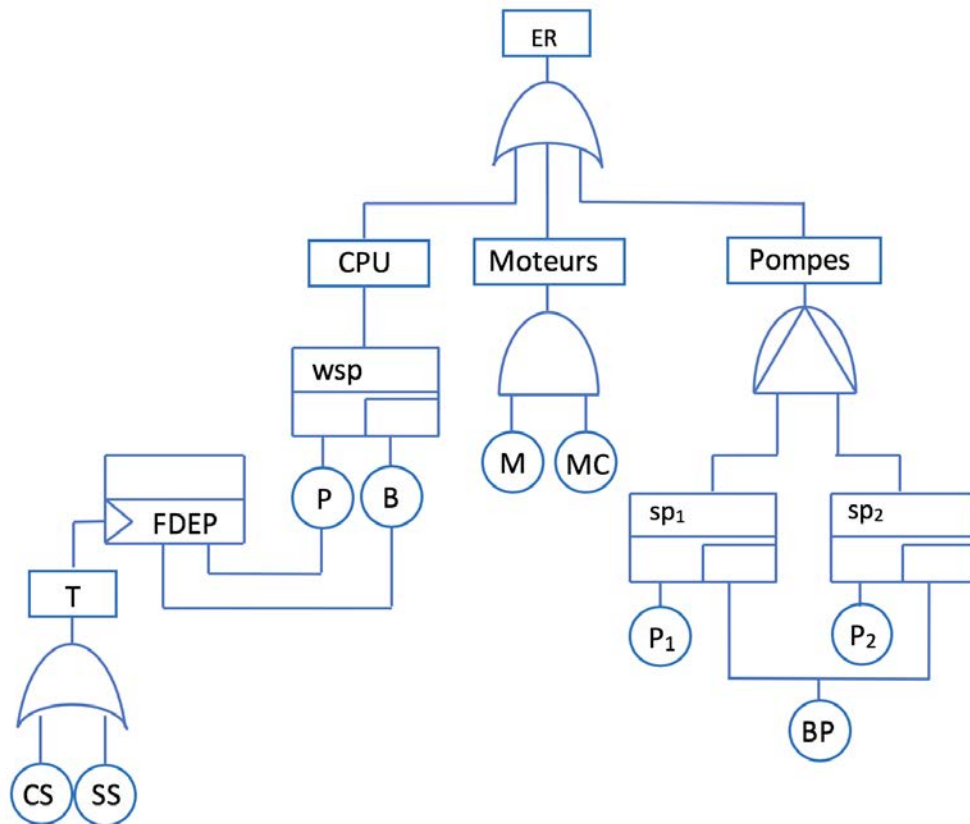


FIGURE 3.39 – Arbre de défaillances dynamique du système CAS.

14. Cela semble surprenant, mais le système est décrit ainsi dans la littérature.

3.6.1 Fonction de structure

Les expressions c_{CAS} et t_{CAS} sont données en fonction des portes en entrée de la porte OU, par les équations (3.82)-(3.83).

$$c_{CAS} = c_{CPU} \vee c_{Motors} \vee c_{Pumps} \quad (3.82)$$

$$t_{CAS} = \min(t_{CPU}, t_{Motors}, t_{Pumps}) \quad (3.83)$$

CPU L'unité CPU est modélisée par une porte Warm SPARE, cela signifie que le composant de rechange B peut défaillir avant d'avoir été sollicité par la porte. À partir des équations (3.19) et (3.20), nous déduisons les équations (3.84) et (3.85).

$$c_{CPU} = c_P \wedge c_B \quad (3.84)$$

$$t_{CPU} = \min(t_T, \max(t_P, t_B)) \quad (3.85)$$

Gâchette Nous modélisons la gâchette de la porte FDEP par les équations (3.86)-(3.88), obtenues grâce à l'équation (3.13).

$$c_T = c_{CS} \wedge c_{SS} \quad (3.86)$$

$$t_T = \min(t_{CS}, t_{SS}) \quad (3.87)$$

$$c_T \Rightarrow c_P \wedge c_B \quad (3.88)$$

Moteurs L'unité des moteurs est modélisée par une porte ET, dont les équations sont (3.6) et (3.7) qui est représentée par les équations (3.89) et (3.90).

$$c_{Motors} = c_M \wedge c_{MC} \quad (3.89)$$

$$t_{Motors} = \max(t_M, t_{MC}) \quad (3.90)$$

Pompes Une porte PAND modélise l'unité des pompes. En ayant recours aux équations (3.15) et (3.92), nous obtenons les équations (3.91) et (3.92).

$$c_{Pompes} = c_{sp_1} \wedge c_{sp_2} \wedge t_{sp_1, sp_2} \quad (3.91)$$

$$t_{Pompes} = \begin{cases} t_{sp_2} & \text{si } c_{sp_1} \wedge c_{sp_2} \wedge t_{sp_1} < t_{sp_2} \\ +\infty & \text{sinon} \end{cases} \quad (3.92)$$

En se référant aux équations (3.21) et (3.22), nous écrivons les équations (3.93)-(3.96) décrivant les portes SPARE sp_1 et sp_2 .

$$c_{sp_1} = c_{P_1} \wedge (c_{BP} \vee t_{P_2, P_1}) \quad (3.93)$$

$$c_{sp_2} = c_{P_2} \wedge (c_{BP} \vee t_{P_1, P_2}) \quad (3.94)$$

$$t_{csp_1} = \begin{cases} \max(t_{P_1}, t_{BP}) & \text{si } t_{P_1} < t_{P_2} \\ t_{P_1} & \text{sinon} \end{cases} \quad (3.95)$$

$$t_{csp_2} = \begin{cases} \max(t_{P_2}, t_{BP}) & \text{si } t_{P_2} < t_{P_1} \\ t_{P_2} & \text{sinon} \end{cases} \quad (3.96)$$

Afin d'exprimer t_{sp_1, sp_2} , nous éliminons d'abord des équations (3.95) et (3.96) les équations conditionnelles (sous-section 3.3.5), puis nous transformons les minima et maxima (sous-section 3.3.6). Nous obtenons l'équation (3.97).

$$\begin{aligned} t_{sp_1, sp_2} &= t_{P_1, P_2} \wedge \max(t_{P_1}, t_{BP}) < t_{P_2} \\ &\vee t_{P_2, P_1} \wedge t_{P_1} < \max(t_{P_2}, t_{BP}) \\ &= t_{P_1, P_2} \wedge t_{BP, P_2} \\ &\vee t_{P_2, P_1} \wedge t_{P_1, BP} \end{aligned} \quad (3.97)$$

Le composant BP étant en redondance froide, une contrainte de temps doit être ajoutée pour prendre en compte le fait que, s'il a défailli, alors cette défaillance est survenue après sa sollicitation par sp_1 ou sp_2 . Ainsi, soit BP n'a pas défailli, soit il a défailli après (au moins) l'un des composants principaux, ce qui est modélisé par l'équation (3.98).

$$\neg c_{BP} \vee \min(t_{P_1}, t_{P_2}) < t_{BP}. \quad (3.98)$$

En utilisant l'équation (3.45) pour transformer l'opérateur minimum en une disjonction, nous obtenons la contrainte (3.99) que nous ajoutons à la fonction de structure.

$$t_{P_1, BP} \vee t_{P_2, BP} \vee \neg c_{BP} \quad (3.99)$$

Puisque nous utilisons des variables de temps, il est crucial de vérifier que les règles de transitivité sont respectées afin de prévenir toute contradiction. La transitivité entre les variables de temps est traduite par les implications des équations (3.100)-(3.102).

$$t_{P_2, P_1} \wedge t_{P_1, BP} \Rightarrow t_{P_2, BP} \quad (3.100)$$

$$\neg t_{P_2, P_1} \wedge t_{P_2, BP} \Rightarrow t_{P_1, BP} \quad (3.101)$$

$$t_{P_2, BP} \wedge \neg t_{P_1, BP} \Rightarrow t_{P_2, P_1} \quad (3.102)$$

3.6.2 Analyse de la fonction de structure

L'expression booléenne de la fonction de structure obtenue est la suivante :

$$\begin{aligned}
c_{CAS} &= c_{CPU} \vee c_{Motors} \vee c_{Pumps} \\
c_{CPU} &= c_P \wedge c_B \\
c_T &= c_{CS} \wedge c_{SS} \\
c_T &\Rightarrow c_P \wedge c_B \\
c_{Motors} &= c_M \wedge c_{MC} \\
c_{Pompes} &= c_{sp_1} \wedge c_{sp_2} \wedge t_{sp_1, sp_2} \\
c_{sp_1} &= c_{P_1} \wedge (c_{BP} \vee t_{P_2, P_1}) \\
c_{sp_2} &= c_{P_2} \wedge (c_{BP} \vee t_{P_1, P_2}) \\
t_{sp_1, sp_2} &= (t_{P_1, P_2} \wedge t_{BP, P_2}) \vee (t_{P_2, P_1} \wedge t_{P_1, BP}) \\
c_{BP} &\Rightarrow t_{P_1, BP} \vee t_{P_2, BP} \\
t_{P_2, P_1} \wedge t_{P_1, BP} &\Rightarrow t_{P_2, BP} \\
\neg t_{P_2, P_1} \wedge t_{P_2, BP} &\Rightarrow t_{P_1, BP} \\
t_{P_2, BP} \wedge \neg t_{P_1, BP} &\Rightarrow t_{P_2, P_1}
\end{aligned}$$

À partir de la fonction de structure du système, nous utilisons l'algorithme de la section 3.4. Cet algorithme génère les séquences du lien minimales, qui sont toutes d'ordre 6, en 0,7 secondes. Les séquences de lien minimales sont les suivantes :

$$\begin{aligned}
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_2, BP}, t_{BP, P_1}\} \\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_2, BP}, t_{BP, P_1}\} \\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_1, P_2}, t_{P_2, BP}\} \\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_1, P_2}, t_{P_2, BP}\} \\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_1, BP}, x_{P_2}\} \\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_1, BP}, x_{P_2}\} \\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_2, P_1}, x_{BP}\} \\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_2, P_1}, x_{BP}\} \\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_2, BP}, t_{BP, P_1}\} \\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_2, BP}, t_{BP, P_1}\} \\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_1, P_2}, t_{P_2, BP}\} \\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_1, P_2}, t_{P_2, BP}\} \\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_1, BP}, x_{P_2}\} \\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_1, BP}, x_{P_2}\} \\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_2, P_1}, x_{BP}\} \\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_2, P_1}, x_{BP}\}
\end{aligned}$$

Nous remarquons que les composants CS et SS apparaissent dans chaque MTSS, ce qui implique que ces composants ont un impact crucial sur l'occurrence de l'évènement redouté. La défaillance de l'un ou de l'autre conduit immédiatement à la défaillance du système. Par conséquent, il est primordial de s'assurer qu'il est très peu probable que les composants CS et SS défaillent. Nous remarquons également que les composants M et MC apparaissent dans la moitié des MTSSs. Nous pouvons donc en déduire qu'ils ont une importance non négligeable dans la réalisation de l'évènement redouté.

Les travaux de Merle et al. (2016) donnent huit séquences de coupe minimales (MCSSs) :

- 2 MCSSs d'ordre 1 :
 - $\{CS\}$ et
 - $\{SS\}$;
- 4 MCSSs d'ordre 2 :
 - $\{Bd, P\}$ ¹⁵,
 - $\{P, Ba\}$,
 - $\{MOTOR, MOTORC\}$ et
 - $\{MOTORC, MOTOR\}$;
- 2 MCSSs d'ordre 3 :
 - $\{P2, P1, BP\}$ et
 - $\{P1, BP, P2\}$.

3.6.3 Obtention du polynôme de fiabilité

Afin de calculer la fiabilité du système, nous devons déterminer le polynôme de fiabilité à partir du diagramme de Hasse. Les séquences de liens minimales déterminées précédemment nous permettent d'obtenir le diagramme de Hasse.

Les taux de défaillances proposés par Boudali et Dugan (2005) pour chaque composant pour des distributions exponentielles sont répertoriées dans le tableau 3.1.

Composant	Taux de défaillance (10^{-4}) [h^{-1}]
CS	1
SS	2
P, B	4
P_1, P_2, BP	5
M	5
MC	1

Tableau 3.1 – Taux de défaillance des composants.

15. La notation a ou d , en indice d'un composant, précise si la défaillance du composant a eu lieu lorsqu'il était en attente (dormant) ou en service (active).

La réalisation du diagramme de Hasse du système nécessiterait une implémentation de l'approche proposée par Brînzei et Aubry (2018), adaptée aux systèmes dynamiques.

Nous proposons de scinder ce système en trois sous-systèmes : l'unité CPU, l'unité moteurs et l'unité pompes pour l'étude quantitative. Cette séparation en trois sous-systèmes est possible, car l'événement redouté est obtenu comme un « OU logique » entre les événements représentant les défaillances de ces trois sous-systèmes. Les sous-systèmes sont indépendants entre eux car ils sont constitués des composants propres, non-partagés entre eux (Figure 3.39). Cette indépendance permettra par conséquent de calculer la fiabilité de chaque sous-système indépendamment des autres sous-systèmes.

3.6.3.1 Analyse de l'unité CPU

Considérons l'unité CPU du système CAS (Figure 3.40), dont la porte SPARE est en redondance "tiède" (warm spare).

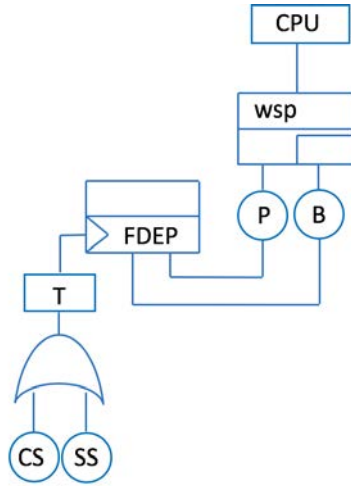


FIGURE 3.40 – Unité CPU du système CAS.

Les séquences de lien minimales de l'unité CPU sont les suivantes :

$$\{\{x_P, x_{CS}, x_{SS}\}, \{x_B, x_{CS}, x_{SS}\}\} \quad (3.103)$$

D'après le tableau 3.1, les composants ont des distributions exponentielles de taux de défaillance $\lambda_{CS} = 1 \cdot 10^{-4}h^{-1}$, $\lambda_{SS} = 2 \cdot 10^{-4}h^{-1}$ et $\lambda_B = \lambda_P = 4 \cdot 10^{-4}h^{-1}$.

$$R_{CS}(t) = e^{-\lambda_{CS} \cdot t} = e^{-1 \cdot 10^{-4} \cdot t} \quad (3.104)$$

$$R_{SS}(t) = e^{-\lambda_{SS} \cdot t} = e^{-2 \cdot 10^{-4} \cdot t} \quad (3.105)$$

$$R_P(t) = e^{-\lambda_P \cdot t} = e^{-4 \cdot 10^{-4} \cdot t} \quad (3.106)$$

Puisque la porte SPARE est en redondance "tiède", le composant B n'a pas le même

taux de défaillance lorsqu'il est en attente (inactif) et après avoir été sollicité (actif). Il a un coefficient de dormance $\beta = 0,5$.

Le taux de défaillance du composant B vaut $\lambda_B = 0,0004$ lorsqu'il est actif et $\beta \cdot \lambda_B = 0,0002$ lorsqu'il est inactif.

L'équation (3.107) permet d'obtenir la valeur de la fiabilité de l'unité CPU, en calculant la probabilité que :

- les composants CS et SS fonctionnent et que
- les composants P et B fonctionnent ; ou
- le composant P fonctionne et le composant B défaille ; ou
- le composant P défaille et le composant B fonctionne.

$$R_{CPU} = e^{-\lambda_{CS} \cdot t} \cdot e^{-\lambda_{SS} \cdot t} \cdot \left(e^{-\lambda_P \cdot t} \cdot e^{-\beta \cdot \lambda_B \cdot t} + e^{-\lambda_P \cdot t} \cdot (1 - (e^{-\beta \cdot \lambda_B \cdot t})) + \lambda_P \cdot e^{-\lambda_P \cdot t} \cdot \frac{1 - (e^{-(\lambda_P + \beta \cdot \lambda_B - \lambda_B) \cdot t})}{\lambda_P + \beta \cdot \lambda_B - \lambda_B} \right) \quad (3.107)$$

La fiabilité de l'unité CPU est de 67,66% après 1000 heures et la Figure 3.41 illustre l'évolution de la fiabilité de l'unité des processeurs durant 1000 heures.

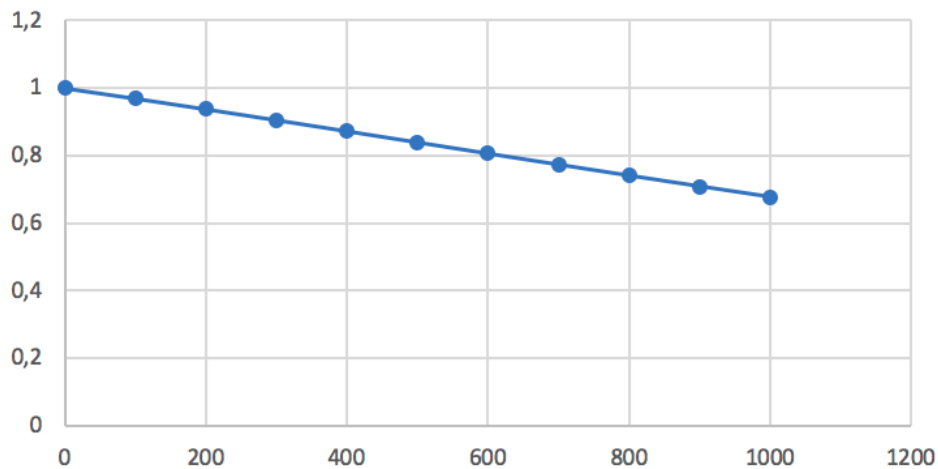


FIGURE 3.41 – Courbe de fiabilité de l'unité des processeurs du système CAS.

3.6.3.2 Analyse de l'unité des moteurs

Considérons l'unité des moteurs du système CAS (Figure 3.42).

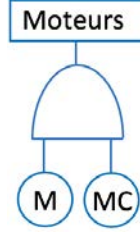


FIGURE 3.42 – Unité moteurs du système CAS.

L'équation (3.108) donne la fonction de structure de l'unité moteurs et l'équation (3.109) donne la fonction de structure dualisée de l'unité moteurs.

$$f_{moteurs} = c_M \wedge c_{MC} \quad (3.108)$$

$$\neg f_{moteurs} = x_M \vee x_{MC} \quad (3.109)$$

La Figure 3.43 donne le diagramme de Hasse de l'unité moteurs, où les composants sont ordonnés dans l'ordre $\langle M, MC \rangle$.

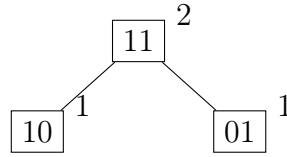


FIGURE 3.43 – Diagramme de Hasse de l'unité moteurs.

L'équation (3.110) est le polynôme de fiabilité de l'unité moteurs.

$$R_{moteurs} = R_M + R_{MC} - R_M \cdot R_{MC} \quad (3.110)$$

D'après le tableau 3.1, les composants M et MC ont une distribution exponentielle de taux de défaillance $\lambda_M = 5 \cdot 10^{-4}h^{-1}$ et respectivement $\lambda_{MC} = 1 \cdot 10^{-4}h^{-1}$.

$$R_M(t) = e^{-\lambda_M \cdot t} = e^{-5 \cdot 10^{-4} \cdot t} \quad (3.111)$$

$$R_{MC}(t) = e^{-\lambda_{MC} \cdot t} = e^{-1 \cdot 10^{-4} \cdot t} \quad (3.112)$$

À partir de l'équation (3.110), et des équations (3.111) et (3.112), l'équation (3.113) calcule la fiabilité de l'unité moteurs après un temps de mission $t = 1000$ heures.

$$R_{moteurs}(t) = e^{-5 \cdot 10^{-4} \cdot t} + e^{-1 \cdot 10^{-4} \cdot t} - (e^{-5 \cdot 10^{-4} \cdot t} \cdot e^{-1 \cdot 10^{-4} \cdot t}) \quad (3.113)$$

La fiabilité de l'unité moteurs est de 96,25% après 1000 heures et la Figure 3.44 illustre l'évolution de la fiabilité de l'unité des moteurs durant 1000 heures.

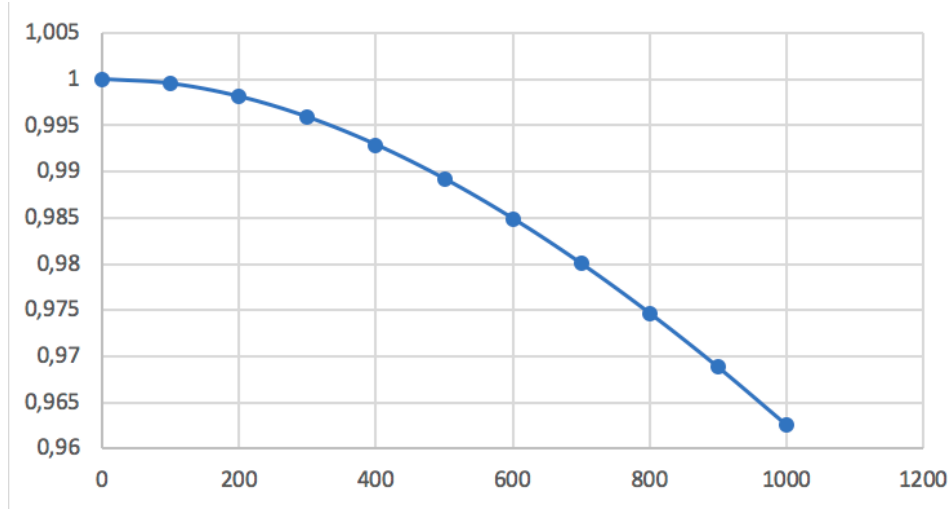


FIGURE 3.44 – Courbe de fiabilité de l'unité des moteurs du système CAS.

3.6.3.3 Analyse de l'unité des pompes

Considérons l'unité des pompes du système CAS (Figure 3.45). Les portes sp_1 et sp_2 sont des portes SPARE, dont le composant BP est en redondance froide.

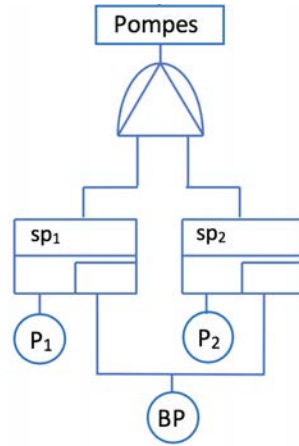


FIGURE 3.45 – Unité des pompes du système CAS.

Les équations (3.114)-(3.117) décrivent le comportement des portes de la Figure 3.45.

$$c_{Pompes} = c_{sp_1} \wedge c_{sp_2} \wedge t_{sp_1, sp_2} \quad (3.114)$$

$$c_{sp_1} = c_{P_1} \wedge (c_{BP} \vee t_{P_2, P_1}) \quad (3.115)$$

$$c_{sp_2} = c_{P_2} \wedge (c_{BP} \vee t_{P_1, P_2}) \quad (3.116)$$

$$t_{sp_1, sp_2} = (t_{P_1, P_2} \wedge \neg t_{P_2, BP}) \vee (\neg t_{P_1, P_2} \wedge t_{P_1, BP}) \quad (3.117)$$

La redondance froide du composant de rechange BP , qui ne peut défaillir qu'après avoir été sollicité, est décrite par : $c_{BP} \Rightarrow t_{P_1,BP} \vee t_{P_2,BP}$.

Les équations (3.118)-(3.120) expriment les règles de transitivité à respecter.

$$t_{P_2,P_1} \wedge t_{P_1,BP} \Rightarrow t_{P_2,BP} \quad (3.118)$$

$$\neg t_{P_2,P_1} \wedge t_{P_2,BP} \Rightarrow t_{P_1,BP} \quad (3.119)$$

$$t_{P_2,BP} \wedge \neg t_{P_1,BP} \Rightarrow t_{P_2,P_1} \quad (3.120)$$

L'équation (3.121) donne la fonction de structure de l'unité pompes.

$$\begin{aligned} f_{pompes} &= c_{sp_1} \wedge c_{sp_2} \wedge t_{sp_1,sp_2} \\ &= c_{P_1} \wedge (c_{BP} \vee t_{P_2,P_1}) \wedge c_{P_2} \wedge (c_{BP} \vee t_{P_1,P_2}) \\ &\quad \wedge ((t_{P_1,P_2} \wedge \neg t_{P_2,BP}) \vee (\neg t_{P_1,P_2} \wedge t_{P_1,BP})) \end{aligned} \quad (3.121)$$

L'équation (3.122) donne la fonction de structure dualisée de l'unité pompes.

$$\begin{aligned} \neg f_{pompes} &= x_{P_1} \vee (x_{BP} \wedge t_{P_2,P_1}) \vee x_{P_2} \vee (x_{BP} \wedge t_{P_1,P_2}) \\ &\quad \vee ((t_{P_1,P_2} \vee \neg t_{P_2,BP}) \wedge (\neg t_{P_1,P_2} \vee t_{P_1,BP})) \end{aligned} \quad (3.122)$$

Les séquences de lien minimales de l'unité des pompes sont les suivantes :

$$\{BP, t_{P_1,BP}, t_{BP,P_2}\}, \{BP, t_{P_2,P_1}, t_{P_1,BP}\}, \{t_{P_1,P_2}, t_{P_2,BP}\}, \{t_{P_2,BP}, t_{BP,P_1}\}.$$

Grâce aux séquences de lien minimales, nous construisons le diagramme de Hasse et obtenons la Figure 3.46, où les composants sont ordonnés dans l'ordre $\langle P_1, P_2, BP \rangle$.

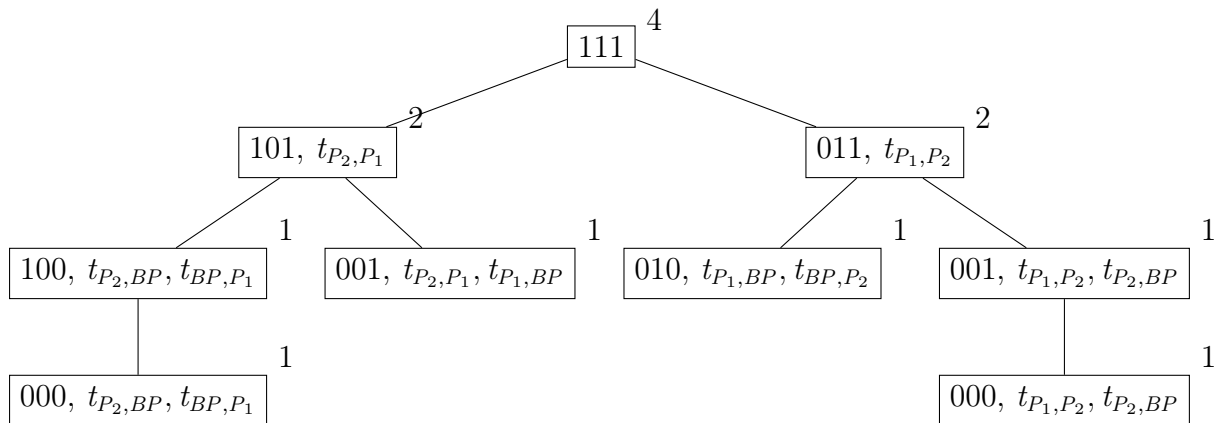


FIGURE 3.46 – Diagramme de Hasse de l'unité constituée des pompes du système CAS.

D'après le tableau 3.1, les composants P_1 , P_2 et BP ont des distributions exponentielles de taux de défaillance $\lambda = 5 \cdot 10^{-4} h^{-1}$.

$$f_{P_1}(t) = f_{P_2}(t) = \lambda \cdot e^{-\lambda \cdot t} \quad (3.123)$$

$$F_{P_1}(t) = F_{P_2}(t) = 1 - e^{-\lambda \cdot t} \quad (3.124)$$

$$R_{P_1}(t) = R_{P_2}(t) = e^{-\lambda \cdot t} \quad (3.125)$$

Le composant BP ne peut défaillir qu'une fois qu'il a été sollicité (actif).

Le temps de fonctionnement du composant BP débute donc lorsque le composant qu'il doit remplacer défaille (instant de temps auquel nous ferons référence par $t = u$) ; et se termine lorsque le composant BP défaille.

Par conséquent, nous obtenons les distributions suivantes :

$$f_{BP}(t)_{\text{sachant que BP a été activé à l'instant } u} = \lambda \cdot e^{-\lambda \cdot (t-u)} \quad (3.126)$$

$$F_{BP}(t)_{\text{sachant que BP a été activé à l'instant } u} = 1 - e^{-\lambda \cdot (t-u)} \quad (3.127)$$

$$R_{BP}(t)_{\text{sachant que BP a été activé à l'instant } u} = e^{-\lambda \cdot (t-u)} \quad (3.128)$$

L'équation (3.131) permet de déterminer la valeur de la fiabilité de l'unité des pompes.

$$R_{\text{pompes}} = R_{P_1}(t) \cdot R_{P_2}(t) \quad (3.129)$$

$$\begin{aligned} & + 2 \cdot R_{P_1}(t) \cdot \int_0^t f_{P_2}(u) \cdot du \\ & + 2 \cdot \int_0^t f_{P_1}(u) \int_u^t f_{P_2}(v) \cdot dv \cdot R_{P_3}(t-u) du \\ & + 2 \cdot \int_0^t f_{P_1}(u) \cdot \int_u^t f_{P_2}(v) \cdot \int_v^t f_{BP}(w-u) \cdot dw \cdot dv \cdot du \end{aligned}$$

$$R_{\text{pompes}} = e^{-\lambda \cdot t} \cdot e^{-\lambda \cdot t} \quad (3.130)$$

$$\begin{aligned} & + 2 \cdot e^{-\lambda \cdot t} \cdot \int_0^t f_{P_2}(u) \cdot du \\ & + 2 \cdot \int_0^t \lambda \cdot e^{-\lambda \cdot u} \cdot \int_u^t \lambda \cdot e^{-\lambda \cdot v} \cdot dv \cdot e^{-\lambda \cdot (t-u)} \cdot du \\ & + 2 \cdot \int_0^t \lambda \cdot e^{-\lambda \cdot u} \cdot \int_u^t \lambda \cdot e^{-\lambda \cdot v} \cdot \int_v^t \lambda \cdot e^{-\lambda \cdot (w-u)} \cdot dw \cdot dv \cdot du \end{aligned}$$

$$R_{\text{pompes}} = e^{-2 \cdot \lambda \cdot t} \quad (3.131)$$

$$\begin{aligned} & + 2 \cdot (e^{-\lambda \cdot t} - e^{-2 \cdot \lambda \cdot t}) \\ & + 2 \cdot (e^{-\lambda \cdot t} - e^{-2 \cdot \lambda \cdot t} - (\lambda \cdot t \cdot e^{-\lambda \cdot t})) \\ & + 2 \cdot \left(\frac{1}{4} - e^{-\lambda \cdot t} + e^{-2 \cdot \lambda \cdot t} \cdot \left(\frac{3}{4} + \frac{\lambda \cdot t}{2} \right) \right) \end{aligned}$$

L'équation (3.129) calcule, dans l'ordre, la probabilité que l'unité des pompes se trouve dans l'état correspondant :

- au noeud $\{111\}$: $R_{P_1}(t) \cdot R_{P_2}(t)$ (le composant R_{P_3} n'est pas encore sollicité) ;
- aux noeuds $\{101, t_{P_2, P_1}\}$ et $\{100, t_{P_2, BP}, t_{BP, P_1}\}$, et $\{011, t_{P_1, P_2}\}$ et $\{010, t_{P_1, BP}, t_{BP, P_2}\}$
 $(R_1 \cap \overline{R_2} \cap R_3 \cup R_1 \cap \overline{R_2} \cap \overline{R_3} = R_1 \cap \overline{R_2}$, et $\overline{R_1} \cap R_2 \cap R_3 \cup \overline{R_1} \cap R_2 \cap \overline{R_3} = \overline{R_1} \cap R_2$) ;
- aux noeuds $\{001, t_{P_2, P_1}, t_{P_1, BP}\}$ et $\{001, t_{P_1, P_2}, t_{P_2, BP}\}$;
- aux noeuds $\{000, t_{P_2, BP}, t_{BP, P_1}\}$ et $\{000, t_{P_1, BP}, t_{BP, P_2}\}$.

La fiabilité de l'unité des pompes après 1000 heures de fonctionnement est de 97,73% et la Figure 3.47 illustre l'évolution de la fiabilité de l'unité des pompes durant 1000 heures.

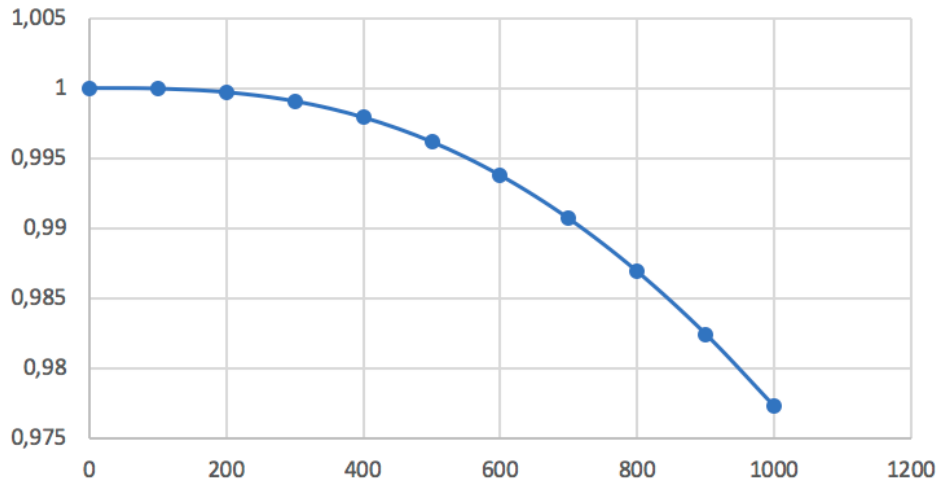


FIGURE 3.47 – Courbe de fiabilité de l'unité des pompes du système CAS.

Enfin, nous déterminons la probabilité que le système fonctionne, soit sa fiabilité R_{CAS} . La défiabilité F_{CAS} est la probabilité que l'unité CPU défaille, ou que l'unité CPU fonctionne et que l'unité des moteurs défaille, ou que les unités CPU et moteurs fonctionnent et que l'unité des pompes défaille.

$$F_{CAS} = (1 - R_{CPU}) + R_{CPU} \cdot (1 - R_{moteurs}) + R_{CPU} \cdot R_{moteurs} \cdot (1 - R_{pompes}) \quad (3.132)$$

$$R_{CAS} = 1 - ((1 - R_{CPU}) + R_{CPU} \cdot (1 - R_{moteurs}) + R_{CPU} \cdot R_{moteurs} \cdot (1 - R_{pompes})) \quad (3.133)$$

La fiabilité du système d'assistance cardiaque après 1000 heures de fonctionnement

est de 63,65% et la Figure 3.47 illustre l'évolution de la fiabilité du système.

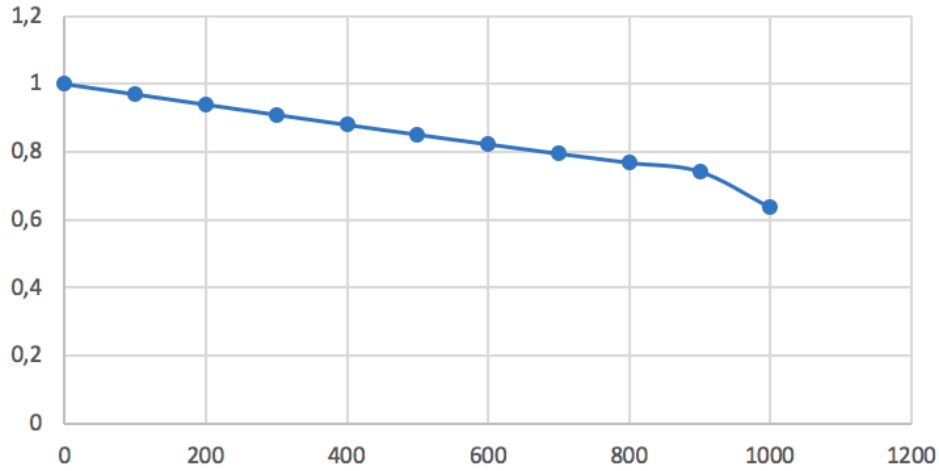


FIGURE 3.48 – Courbe de fiabilité du système CAS.

Le résultat obtenu par notre approche, en utilisant les séquences de lien minimales calculées grâce aux techniques de satisfiabilité, est le même que celui obtenu par Merle et al. (2014).

3.7 Conclusion

Nous avons tout d'abord proposé une adaptation de la fonction de structure aux systèmes dynamiques et une fonction de répartition des composants de rechange des portes SPARE (afin de déterminer pour quelles portes sont utilisés, ou ont été utilisés, les composants). Nous avons ensuite proposé des règles de réécriture de la fonction de structure afin qu'elle soit exprimée sous une forme booléenne, en suggérant l'utilisation d'une variable $t_{i,j}$ afin de considérer l'ordre d'occurrence des défaillances de C_i et C_j comme un booléen. Afin d'effectuer des études qualitatives et quantitatives de systèmes dynamiques, nous avons défini les séquences de lien minimales, nous avons étendu la méthode d'obtention des liens minimaux grâce aux techniques de satisfiabilité du Chapitre 2 aux systèmes dynamiques et nous avons proposé une extension des travaux de Brânzei et Aubry (2018) aux systèmes dynamiques, permettant de déterminer les séquences de lien minimales. Nous avons enfin appliqué notre approche à un cas d'étude de la littérature, un système d'assistance cardiaque.

Notre approche détermine les coupes minimales des systèmes et sous-systèmes industriels en un temps raisonnable dès lors que la somme du nombre de composants et de portes ne dépasse pas une centaine de variables. Ce temps de réponse pourrait être amélioré en filtrant les liens non minimaux durant la boucle d'interaction avec le solveur SAT.

Cela réduirait la taille de l'entrée fournie au solveur SAT et après plusieurs dizaines de milliers de modèles générés, le temps de calcul des liens minimaux serait réduit.

Des optimisations algorithmiques rendrait notre approche plus compétitive. Il serait intéressant d'optimiser la boucle qui récupère le modèle produit par le solveur de satisfiabilité et qui modifie l'entrée envoyée au solveur. Cette entrée contient la définition des variables, la fonction de structure et autant de lignes que le nombre d'appels faits au solveur. En effet, après de très nombreux appels au solveur, la lecture de la formule (envoyée au solveur SAT) ralentit l'obtention d'un nouveau modèle. Nous proposons d'ajouter une étape périodique, tous les 1000 ou 5000 modèles générés par exemple. Cette étape aurait pour objectif de supprimer les modèles couverts par d'autres modèles de la formule, ce qui permettrait d'éliminer les lignes inutiles et d'accélérer le calcul, essentiellement après pour les formules (envoyées au solveur SAT) où 10000 séquences de lien ou plus doivent être générés.

L'utilisation d'un solveur incrémental permettrait d'ajouter des clauses pendant le calcul. Cela permettrait de ne pas reprendre le calcul dès le début à chaque fois qu'un nouveau modèle est généré, ce qui entraînerait un gain de temps conséquent.

Par ailleurs, il serait intéressant d'implémenter l'approche de génération du polynôme de fiabilité des systèmes dynamiques à partir des séquences de liens minimales, afin de permettre de considérer des systèmes de grande taille.

Conclusion

Dans ce mémoire nous avons proposé une approche employant les techniques de satisfiabilité afin d'évaluer la fiabilité de systèmes industriels et de pouvoir déterminer l'ensemble de liens minimaux de ces systèmes.

Dans le chapitre 1, nous avons présenté la Sûreté de Fonctionnement. Nous avons dressé un état de l'art des différentes modélisations permettant de décrire le comportement d'un système, essentiellement dans le cadre de systèmes binaires non réparables. Nous avons ensuite présenté des techniques et outils du domaine des méthodes formelles. Nous avons développé les techniques d'évaluation probabiliste des systèmes industriels. Enfin, nous avons justifié le choix d'une modélisation par arbres de défaillances et l'intérêt de la satisfiabilité pour étudier la fonction de structure.

Dans le chapitre 2, nous avons proposé une approche pour les systèmes modélisés par arbres de défaillances statiques, dont l'objectif est d'identifier les liens minimaux du système à partir de la fonction de structure. Nous avons proposé trois algorithmes, qui diffèrent selon la forme de la fonction de structure. Nous proposons une dualisation de la fonction de structure afin de générer les liens minimaux du système et de construire le diagramme de Hasse restreint aux états de fonctionnement du système. Le travail développé dans le chapitre 2 a fait l'objet d'une publication à la conférence *Mathematical Models on Reliability* (Duroeulx et al. (2017)).

Dans le chapitre 3, nous avons proposé une approche pour les systèmes modélisés par arbres de défaillances dynamiques. Cette approche est entièrement basée sur l'utilisation des techniques de satisfiabilité et d'un solveur SAT, ce qui nécessite d'écrire la fonction de structure sous la forme d'une expression booléenne. Nous avons proposé une méthodologie permettant d'obtenir l'expression booléenne de la fonction de structure pour les systèmes dynamiques, en considérant les portes statiques ET, OU, KooN et les portes dynamiques FDEP, PAND et SPARE. Nous avons particulièrement détaillé le cas des portes SPARE, dont les entrées peuvent être sollicitées par plusieurs portes SPARE. Par la suite, nous avons introduit la notion de séquence de lien minimale pour les systèmes dynamiques. Enfin, nous avons étendu le diagramme de Hasse aux systèmes dynamiques et présenté

l'obtention du polynôme de la fiabilité dans le cas de systèmes dynamiques. L'approche qualitative développée dans ce chapitre, permettant d'obtenir la fonction de structure et d'identifier les séquences de lien minimales, a conduit à une publication à la conférence *European Safety and Reliability Conference* (Duroeulx et al. (2019)).

Ces travaux présentent une utilisation possible des techniques de satisfiabilité pour les études de sûreté de fonctionnement. En fournissant la fonction de structure au solveur SAT, celui-ci génère une configuration du système garantissant son fonctionnement et il suffit ensuite de le resolliciter jusqu'à pouvoir identifier l'ensemble des liens minimaux. Des algorithmes de calcul ont été proposés et un prototype logiciel a été implémenté. En outre, leur adaptation aux systèmes dynamiques permet de considérer la possibilité de les adapter par la suite à d'autres extensions des arbres de défaillances.

Plusieurs perspectives de recherche peuvent être formulées pour des travaux futurs, d'une part en perfectionnant nos algorithmes, d'autre part, en élargissant le champ des systèmes étudiés.

Dans le chapitre 2, un prototype logiciel a été proposé dans le but de calculer les liens minimaux à partir de la fonction de structure (CNF) en employant un solveur SAT. Une optimisation de l'algorithme permettrait d'améliorer son efficacité et de traiter des systèmes de plus grande taille (en la somme des composants et des portes). À mesure que de nouveaux modèles sont générés, des clauses représentant des noeuds plus bas (dans le diagramme de Hasse) que ceux des clauses précédentes sont ajoutées à la fonction envoyée au solveur SAT. La suppression des clauses devenues superflues de la fonction envoyée au solveur SAT permettrait un traitement plus rapide de l'entrée et une génération plus rapide des modèles, lorsque plusieurs dizaines de milliers de modèles ont déjà été obtenus.

Un prototype logiciel a été créé par le passé au Cran par Jérémie Fleurant (2014), afin d'automatiser la génération du diagramme de Hasse et du polynôme de fiabilité à partir de l'ensemble des liens minimaux pour des systèmes statiques. Il serait intéressant d'étendre ces travaux à des systèmes dynamiques. Cela permettrait l'obtention automatique du polynôme de fiabilité à partir des séquences de lien minimales.

Une hypothèse très fréquente dans les études de fiabilité est de supposer que tous les composants sont fonctionnels à l'instant initial. Néanmoins, il peut être intéressant de considérer qu'un composant puisse être défaillant à la mise en service du système. Si la défaillance d'un composant survenait avant l'instant initial, nous pourrions construire un diagramme de Hasse sans le composant en question (tant que le système est cohérent).

Dans l'application du chapitre 3, le système considéré est représenté par un arbre de défaillances dynamique dont la porte la plus haute est une porte OU, dont chaque entrée est constituée de composants différents. Cela nous a permis de scinder l'arbre de défaillances en trois parties et de les analyser séparément, afin d'éviter la construction du diagramme de Hasse complet. Il serait intéressant de développer une méthode permettant de scinder l'arbre de défaillances en plusieurs sous-arbres, même lorsque la porte la plus haute n'est pas une porte OU. Cela permettrait de faire le diagramme de Hasse de chacun, d'écrire le polynôme de fiabilité de chacun et enfin de déterminer le polynôme de fiabilité du système complet, à partir de ceux des sous-arbres. Cela permettrait de réduire la complexité des équations et d'augmenter la rapidité des calculs réalisés par le solveur SAT, sans nécessiter d'approximation.

Merle et al. (2014) propose une approche permettant de calculer les séquences de coupes minimales d'un système dynamique à partir de sa fonction de structure. Dans des travaux ultérieurs, Merle et al. (2016) propose de coupler cette approche à des simulations de Monte Carlo afin d'obtenir les séquences de coupes minimales. Nous pourrions employer cette approche pour identifier les séquences de coupes, puis étendre notre troisième algorithme du chapitre 2 aux systèmes dynamiques, afin de déterminer les séquences de liens minimales à partir des séquences de coupes minimales et d'obtenir le polynôme de fiabilité de ces systèmes.

La modélisation des portes SPARE passe sous silence le fait que des entrées soient en attente tant qu'elles n'ont pas été sollicitées, bien que cela puisse être déduit des défaillances des autres entrées (puisque l'ordre de sollicitation des entrées est connu). Une représentation plus détaillée des entrées des portes SPARE ferait la distinction entre trois états possibles : en attente, en fonctionnement et défaillant. Il semble alors naturel de représenter les entrées des portes SPARE par des variables entières.

De surcroît, la modélisation par variables entières permettrait également de prendre en considération les modes de fonctionnement dégradés, des composants pouvant fonctionner partiellement. Il serait intéressant d'étendre notre approche basée sur les techniques de satisfiabilité afin qu'elle permette de considérer les systèmes multi-états, pour lesquels les composants et le système pourront être représentés par des variables entières.

La satisfiabilité modulo théories (SMT) permet d'une part de considérer d'autres théories que la logique booléenne et d'autre part, de s'intéresser à des formules où interviennent plusieurs théories combinées (souvent sans quantificateurs). L'utilisation d'un solveur SMT permettrait de considérer des variables non booléennes pour modéliser des entités de rechange et des modes de fonctionnement dégradés.

Néanmoins, cette modélisation des systèmes multi-états n'utilise qu'un intervalle fini et fixe d'entiers et l'utilisation des solveurs SMT devient particulièrement intéressante

lorsque les opérateurs de ces théories sont utilisés, comme l'addition sur les entiers.

Une autre possibilité serait de modéliser les entrées des portes SPARE consisterait à les représenter par plusieurs booléens, grâce à la correspondance entre le système décimal et le système binaire.

Cette approche nécessite une étape de réécriture de la formule, mais peut être envoyée à un solveur SAT. Au contraire, l'utilisation de variables entières permet d'éviter l'étape de réécriture mais nécessite un solveur SMT et l'utilisation de plusieurs théories (booléenne et arithmétique linéaire sur les entiers), ce qui a pour conséquence un temps de calcul plus long. De plus, l'utilisation d'un solveur SMT devient intéressante lorsque les opérations arithmétiques disponibles dans l'outil de résolution SMT interviennent dans la modélisation des systèmes.

Pour conclure, nous pourrions envisager d'utiliser la représentation proposée par Zaitseva et Levashenko (2013), à l'aide d'un diagramme de décision à valeurs multiples (multi-valued decision diagram ou MDD) pour les systèmes multi-états. Nous pourrions proposer une extension du diagramme de Hasse aux systèmes multi-états. À partir de celui-ci, nous chercherions à étendre l'approche de Brânzei et Aubry (2018) aux systèmes multi-états.

Annexe

Conversion d'un fichier cp à smt2

Un fichier cp contient la fonction de structure d'un système dans un format textuel, sous une forme compacte et explicite, obtenu à partir d'un arbre de défaillances. Il se compose d'une première partie contenant la déclaration des variables et d'une seconde partie répertoriant les règles associées à l'arbre de défaillances. Ces règles décrivent chaque porte de l'arbre de défaillances, en fonction de ses entrées.

Un fichier smt2 contient la fonction de structure d'un système dans un format textuel moins compacte et moins explicite que le format cp, mais intelligible par un solveur SMT.

Ces travaux ont été réalisé par Romain Masson (2019), dans le cadre d'un master, afin d'implémenter une conversion automatique du format cp vers le format smt2. Les fichiers cp contiennent la fonction de structure de systèmes modélisés par des arbres de défaillances statiques (contenant les portes ET, OU et KooN).

Un *lexer* et un *parser* sont créés afin de convertir les fichiers cp en smt2. Le lexer a pour but de décomposer le contenu du fichier cp en tuples (type,nom), appelés *tokens*, afin d'écrire efficacement au format smt2. Le parser utilise ces tokens, les intègre dans une liste et écrit la fonction de structure dans un fichier smt2 selon les informations contenues dans les tokens. La conversion des tokens est présentée par la Figure 3.49.


<pre>("(", "") ("redoute", "r1") ("==", "") ("(", "") ("gate", "g1") ("&&", "") ("gate", "g2") (")", "") (")", "") ("; ", "")</pre>		<pre>{assert (or (not r1) (and g1 g2))} {assert (or (not (and g1 g2)) r1)}</pre>
---	---	--

FIGURE 3.49 – Conversion des tokens vers smt2s

Dans le fichier cp, les portes ET et OU sont respectivement modélisés par des conjonctions et des disjonctions. Quant aux portes KooN, elles sont décrites par la valeur de K et les N entrées de la porte. Afin de procéder à la conversion vers le format smt2, chaque combinaison de K entrées activées parmi les N entrées de la porte est considérée.

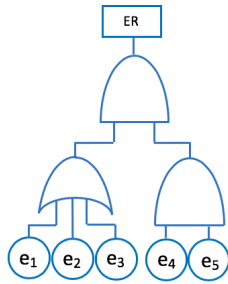


FIGURE 3.50 – Arbre de défaillance d'un système *Test*.

```

1  variable
2    bool r1,g1,g2,e1,e2,e3,e4,e5;
3
4  rule
5    (r1==(g1 && g2));
6    (g1==(e1 || e2 || e3));
7    (g2==(e4 || e5));

```

FIGURE 3.51 – Fichier cp du système *Test*.

```

1 ;Tokens convertis en format smt2:
2 ;;
3 (set-logic QF_UFLIA)
4 (declare-fun r1 () Bool)
5 (declare-fun g1 () Bool)
6 (declare-fun g2 () Bool)
7 (declare-fun e1 () Bool)
8 (declare-fun e2 () Bool)
9 (declare-fun e3 () Bool)
10 (declare-fun e4 () Bool)
11 (declare-fun e5 () Bool)
12 ;;
13 (assert (or (not r1) (and g1 g2)))
14 (assert (or (not (and g1 g2)) r1))
15 ;;
16 (assert (or (not g1) e1 e2 e3))
17 (assert (or (not (or e1 e2 e3)) g1))
18 ;;
19 (assert (or (not g2) e4 e5))
20 (assert (or (not (or e4 e5)) g2))
21 ;;
22 (assert (not r1))
23 ;;
24 (check-sat)
25 (get-model)

```

FIGURE 3.52 – Fichier smt2 du système *Test*.

```

romain@romain-VirtualBox:~/Desktop/PFE/SMT$ python3 mincutsSMT
mt2 MinCutsResult.txt
Nombre de coupes 'relatives' trouvées : 1
Nombre de coupes 'relatives' trouvées : 2
Nombre de coupes 'relatives' trouvées : 3
Suppression des variables supp : 1 / 3 => 33.33 %
Suppression des variables supp : 2 / 3 => 66.67 %
Suppression des variables supp : 3 / 3 => 100.0 %
Nombre d'iteration dans calcul MinCuts : 2 / 3 => 66.67 %
Nombre d'iteration dans calcul MinCuts : 3 / 3 => 100.0 %
-> 3 minimal cut sets found
-> Time needed for the computation: 0.059201717376708984
romain@romain-VirtualBox:~/Desktop/PFE/SMT$

```

FIGURE 3.53 – Calcul des coupes minimales du système *Test*.

```

Minimal cut sets are :
{'e4', 'e5', 'e1'}
{'e4', 'e5', 'e2'}
{'e4', 'e5', 'e3'}

```

FIGURE 3.54 – Coupes minimales du système *Test*.

Exemple 3.50. À partir de la fonction de structure au format cp de la Figure 3.51, qui contient des portes ET et OU, le lexer et le parser permettent de générer le fichier smt2 de la Figure 3.52.

La Figure 3.53 présente le calcul des coupes minimales du système *Test*, effectué en 0,06 secondes. La Figure 3.54 présente les coupes minimales obtenues, qui sont les suivantes :

$$\{\{e_1, e_4, e_5\}, \{e_2, e_4, e_5\}, \{e_3, e_4, e_5\}\}.$$

Bibliographie

- Akers, S. B. (1978). Binary decision diagrams. IEEE Transactions on computers (6), 509–516.
- Aubry, J.-F., N. Brinzei, et M.-H. Mazouni (2016). Systems Dependability Assessment : Benefits of Petri Net Models. John Wiley & Sons.
- Audemard, G. et L. Simon (2009). Glucose : a solver that predicts learnt clauses quality. SAT Competition, 7–8.
- Avizienis, A., J.-C. Laprie, et B. Randell (2004). Dependability and its threats : a taxonomy. In Building the Information Society, pp. 91–120. Springer.
- Ballarini, P., H. Djafri, M. Dufflot, S. Haddad, et N. Pekergin (2011). Cosmos : a statistical model checker for the hybrid automata stochastic logic. In 2011 Eighth International Conference on Quantitative Evaluation of SysTems, pp. 143–144. IEEE.
- Bell, R. (2006). Introduction to iec 61508. In Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55, pp. 3–12. Australian Computer Society, Inc.
- Bennetts, R. (1975). On the analysis of fault trees. IEEE Transactions on reliability 24(3), 175–185.
- Biere, A. (2011). Lingeling and friends at the sat competition 2011. FMV Report Series Technical Report 11(1).
- Biere, A., M. Heule, H. van Maaren, et T. Walsh (2009a). Conflict-driven clause learning sat solvers. Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, 131–153.
- Biere, A., M. Heule, H. van Maaren, et T. Walsh (Eds.) (2009b). Handbook of satisfiability, Volume 185. IOS Press.
- Birnbaum, Z. W., J. D. Esary, et S. C. Saunders (1961). Multi-component systems and structures and their reliability. Technometrics 3(1), 55–77.

- Birolini, A. (1999). Quality and reliability of technical systems. IEEE Transactions on Reliability 48(2), 205–206.
- Boudali, H. et J. B. Dugan (2005). A discrete-time bayesian network reliability modeling and analysis framework. Reliability Engineering & System Safety 87(3), 337–349.
- Bouissou, M. et J.-L. Bon (2003). A new formalism that combines advantages of fault-trees and markov models : Boolean logic driven markov processes. Reliability Engineering & System Safety 82(2), 149–163.
- Boy de la Tour, T. (1992). An optimality result for clause form translation. Journal of Symbolic Computation 14(4), 283–301.
- Boyd, M. A. (1992). Dynamic fault tree models : techniques for analysis of advanced fault tolerant computer systems.
- Brameret, P.-A. (2015). Assessment of reliability indicators from automatically generated partial Markov chains. Ph. D. thesis.
- Brameret, P.-A., A. Rauzy, et J.-M. Roussel (2015). Automated generation of partial markov chain from high level descriptions. Reliability Engineering & System Safety 139, 179–187.
- Brînzei, N. et J.-F. Aubry (2015). An approach of reliability assessment of systems based on graphs models. European Safety and Reliability Conference ESREL CRC Press/Balkema, Taylor & Francis Group, 1485–1493.
- Brînzei, N. et J.-F. Aubry (2018). Graphs models and algorithms for reliability assessment of coherent and non-coherent systems. Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability 232(2), 201–215.
- Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. Computers, IEEE Transactions on 100(8), 677–691.
- Čepin, M. (2011). Reliability block diagram. In Assessment of Power System Reliability, pp. 119–123. Springer.
- Chaux, P.-Y. (2013). Formalisation de la cohérence et calcul des séquences de coupe minimales pour les systèmes binaires dynamiques et réparables. Ph. D. thesis.
- Chiola, G., M. A. Marsan, G. Balbo, et G. Conte (1993). Generalized stochastic petri nets : A definition at the net level and its implications. IEEE Transactions on software engineering 19(2), 89–107.

- Clarhaut, J. (2009). Prise en compte des séquences de défaillances pour la conception de systèmes d'automatisation. Application au feroutage. Ph. D. thesis.
- Clarke, E. M. (1997). Model checking. In International Conference on Foundations of Software Technology and Theoretical Computer Science, pp. 54–56. Springer.
- Clarke, E. M. et J. M. Wing (1996). Formal methods : State of the art and future directions. ACM Computing Surveys (CSUR) 28(4), 626–643.
- Cocozza-Thivent, C. (2018). Processus de renouvellement markovien processus de markov déterministes par morceaux.
- Cook, S. A. (1971). The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing, pp. 151–158. ACM.
- Costa, U. S., A. M. Moreira, et D. Déharbe (2000). A cache-based parallel genetic algorithm for the bdd variable ordering problem. In Proc. of SBAC-PAD, pp. 99–104.
- Coudert, O. et J. C. Madre (1993). Fault tree analysis : 10/sup 20/prime implicants and beyond. In Reliability and Maintainability Symposium, 1993. Proceedings., Annual, pp. 240–245. IEEE.
- David, A., K. G. Larsen, A. Legay, M. Mikučionis, et D. B. Poulsen (2015). Uppaal smc tutorial. International Journal on Software Tools for Technology Transfer 17(4), 397–415.
- Davis, M., G. Logemann, et D. W. Loveland (1961). A machine program for theorem-proving. New York University, Institute of Mathematical Sciences.
- Dugan, J. B., S. J. Bavuso, et M. A. Boyd (1990). Fault trees and sequence dependencies. In Reliability and Maintainability Symposium, 1990. Proceedings., Annual, pp. 286–293. IEEE.
- Dugan, J. B., S. J. Bavuso, et M. A. Boyd (1992). Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on reliability 41(3), 363–377.
- Duroeulx, M., N. Brinzei, M. Duflot, et S. Merz (2017). Satisfiability techniques for computing minimal tie sets in reliability assessment. Mathematical Models on Reliability (MMR).
- Duroeulx, M., N. Brinzei, M. Duflot, et S. Merz (2019). Integrating satisfiability solving in the assessment of system reliability modeled by dynamic fault trees. European Safety and Reliability Conference, 2425–2432.

- Dutuit, Y. et A. Rauzy (2005). Approximate estimation of system reliability via fault trees. Reliability engineering & system safety 87(2), 163–172.
- Eén et Sörensson (2003). An extensible SAT-solver. 6th Intl. Conf. Theory and Applications of Satisfiability Testing 2919, 502–518.
- Emerson, E. A. et E. M. Clarke (1980). Characterizing correctness properties of parallel programs using fixpoints. In International Colloquium on Automata, Languages, and Programming, pp. 169–181. Springer.
- Fleurant, J. (2014). Évaluation de la fiabilité à partir de la fonction de structure et du diagramme de hasse. Technical report, Université de Lorraine.
- Fussell, J., E. Aber, et R. Rahl (1976). On the quantitative analysis of priority-and failure logic. IEEE Transactions on Reliability 25(5), 324–326.
- Garey, M. R., D. S. Johnson, et L. Stockmeyer (1974). Some simplified np-complete problems. In Proceedings of the sixth annual ACM symposium on Theory of computing, pp. 47–63. ACM.
- Gascard, E. et Z. Simeu-Abazi (2018). Quantitative analysis of dynamic fault trees by means of monte carlo simulations : Event-driven simulation approach. Reliability Engineering & System Safety 180, 487–504.
- Gnedenko, B., I. A. Ushakov, et I. Ushakov (1995). Probabilistic reliability engineering. John Wiley & Sons.
- Han, J., E. Taylor, J. Gao, et J. Fortes (2005). Faults, error bounds and reliability of nanoelectronic circuits. In 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP’05), pp. 247–253. IEEE.
- Heule, M. J., O. Kullmann, et V. W. Marek (2016). Solving and verifying the boolean pythagorean triples problem via cube-and-conquer. In International Conference on Theory and Applications of Satisfiability Testing, pp. 228–245. Springer.
- Ibáñez-Llano, C., A. Rauzy, E. Meléndez, et F. Nieto (2010). A reduction approach to improve the quantification of linked fault trees through binary decision diagrams. Reliability Engineering & System Safety 95(12), 1314–1323.
- IEC (2000). Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sûreté.
- IEC (812).

- IEEE (1990). Ieee standard glossary of software engineering terminology (ieee std 610.12-1990). los alamos. CA : IEEE Computer Society 169.
- Ionescu, D.-R. (2016). Évaluation quantitative de séquences d'événements en sûreté de fonctionnement à l'aide de la théorie des langages probabilistes. Ph. D. thesis.
- Karp, R. M. (1972). Reducibility among combinatorial problems. In Complexity of computer computations, pp. 85–103. Springer.
- Kaufmann, A., D. Grouchko, et R. Cruon (1977). Mathematical Models for the Study of the Reliability of Systems (Mathematics in Science and Engineering ed.), Volume 124. Elsevier.
- Kriaa, S. (2016). Joint safety and security modeling for risk assessment in cyber physical systems. Ph. D. thesis.
- Kvassay, M., E. Zaitseva, V. Levashenko, et J. Kostolny (2014). Minimal cut vectors and logical differential calculus. In 2014 IEEE 44th International Symposium on Multiple-Valued Logic, pp. 167–172. IEEE.
- Kvassay, M., E. Zaitseva, V. Levashenko, et J. Kostolny (2016). Binary decision diagrams in reliability analysis of standard system structures. In 2016 International Conference on Information and Digital Technologies (IDT), pp. 164–172. IEEE.
- Kwiatkowska, M., G. Norman, et D. Parker (2009). Prism : probabilistic model checking for performance and reliability analysis. ACM SIGMETRICS Performance Evaluation Review 36(4), 40–45.
- Le Berre, D. et A. Parrain (2010). The sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation 7(2-3), 59–64.
- Lebeaupin, B. (2017). Vers un langage de haut niveau pour une ingénierie des exigences agile dans le domaine des systèmes embarqués avioniques. Ph. D. thesis, Paris Saclay.
- Lee, C.-Y. (1959). Representation of switching circuits by binary-decision programs. The Bell System Technical Journal 38(4), 985–999.
- Legay, A., B. Delahaye, et S. Bensalem (2010). Statistical model checking : An overview. In International conference on runtime verification, pp. 122–135. Springer.
- Leveson, N. G. et J. L. Stolzy (1987). Safety analysis using petri nets. IEEE Transactions on software engineering (3), 386–397.
- Limnios, N. et al. (2005). Stochastic systems in merging phase space. World Scientific.

- Mahmood, Y. A., A. Ahmadi, A. K. Verma, A. Srividya, et U. Kumar (2013). Fuzzy fault tree analysis : a review of concept and application. International Journal of System Assurance Engineering and Management 4(1), 19–32.
- Masson, R. (2019). Évaluation de la fiabilité de systèmes critiques par arbre de défaillances dynamique et liens minimaux. Technical report, Université de Lorraine.
- Matousek, J. et J. Nešetřil (1998). Invitation to discrete mathematics oxford university press.
- McDermid, J. A., M. Nicholson, D. J. Pumfrey, et P. Fenelon (1995). Experience with the application of hazop to computer-based systems. In COMPASS'95 Proceedings of the Tenth Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security, pp. 37–48. IEEE.
- Merle, G. (2010). Algebraic modelling of Dynamic Fault Trees, contribution to qualitative and quantitative analysis. Ph. D. thesis, École normale supérieure de Cachan-ENS Cachan.
- Merle, G. et J.-M. Roussel (2007). Algebraic modelling of fault trees with priority and gates. In 1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07), pp. pp–175.
- Merle, G., J.-M. Roussel, et J.-J. Lesage (2010). Improving the efficiency of dynamic fault tree analysis by considering gate fdep as static. In European Safety and Reliability Conference (ESREL 2010), pp. pp–845. Taylor & Francis.
- Merle, G., J.-M. Roussel, et J.-J. Lesage (2011). Algebraic determination of the structure function of dynamic fault trees. Reliability Engineering & System Safety 96(2), 267–277.
- Merle, G., J.-M. Roussel, et J.-J. Lesage (2014). Quantitative analysis of dynamic fault trees based on the structure function. Quality and Reliability Engineering International 30(1), 143–156.
- Merle, G., J.-M. Roussel, J.-J. Lesage, V. Perchet, et N. Vayatis (2016). Quantitative analysis of dynamic fault trees based on the coupling of structure functions and monte carlo simulation. Quality and Reliability Engineering International 32(1), 7–18.
- Minato, S.-i. (1993). Zero-suppressed bdds for set manipulation in combinatorial problems. In Proceedings of the 30th international Design Automation Conference, pp. 272–277. ACM.
- Minato, S.-i. (2001). Zero-suppressed bdds and their applications. International Journal on Software Tools for Technology Transfer 3(2), 156–170.

- Mortureux, Y. (2001). La sûreté de fonctionnement : méthodes pour maîtriser les risques.
- Murata, T. (1989). Petri nets : Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580.
- Narushima, H. (1974). Principle of inclusion-exclusion on semilattices. Journal of Combinatorial Theory, Series A 17(2), 196–203.
- Narushima, H. (1982). Principle of inclusion-exclusion on partially ordered sets. Discrete Mathematics 42(2-3), 243–250.
- Nonnengart, A., G. Rock, et C. Weidenbach (1998). On generating small clause normal forms. In International Conference on Automated Deduction, pp. 397–411. Springer.
- Petri, C. A. (1962). Kommunikation mit automaten.
- Piètre-Cambacédès, L. (2010). Des relations entre sûreté et sécurité. Ph. D. thesis.
- Piètre-Cambacédès, L., Y. Deflesselle, et M. Bouissou (2011). Security modeling with bdmp : from theory to implementation. In 2011 Conference on Network and Information Systems Security, pp. 1–8. IEEE.
- Pirou, P.-Y. (2015). Contribution à l’analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables. Ph. D. thesis.
- Pirou, P.-Y., J.-M. Faure, et J.-J. Lesage (2019). Finding the minimal cut sequences of dynamic, repairable, and reconfigurable systems from generalized boolean logic driven markov process models. Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability, 1748006X19827128.
- Plaisted, D. A. et S. Greenbaum (1986). A structure-preserving clause form translation. Journal of Symbolic Computation 2(3), 293–304.
- Rauzy, A. et Y. Dutuit (1997). Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within aralia. Reliability Engineering & System Safety 58(2), 127–144.
- Rauzy, A. B. (2011). Sequence algebra, sequence decision diagrams and dynamic fault trees. Reliability Engineering & System Safety 96(7), 785–792.
- Reibman, A. et K. Trivedi (1988). Numerical transient analysis of markov models. Computers & Operations Research 15(1), 19–36.
- Reifer, D. J. (1979). Software failure modes and effects analysis. IEEE Transactions on reliability 28(3), 247–249.

- Ringler, J. (1988). Sur un petit problème pratique d'estimation en fiabilité. Revue de statistique appliquée 36(4), 25–32.
- Rogova, E. et G. Lodewijks (2015). Braking system redundancy requirements for moving walks. Reliability Engineering and System Safety 133, 203–211.
- Ruijters, E., C. E. Budde, M. C. Nakhaee, M. Stoelinga, D. Bucur, D. Hiemstra, et S. Schivo (2019a). Ffort : A benchmark suite for fault tree analysis.
- Ruijters, E., C. E. Budde, M. C. Nakhaee, M. Stoelinga, D. Bucur, D. Hiemstra, et S. Schivo (2019b). FFORT website (University of Twente, Formal Methods and Tools group. ed.). <https://dftbenchmarks.utwente.nl>.
- Ruijters, E. et M. Stoelinga (2015). Fault tree analysis : a survey of the state-of-the-art in modeling, analysis and tools. Computer science review 15, 29–62.
- Satodev (2018). GRIF - GRaphical Interface for reliability Forecasting software. <http://grif-workshop.com/>.
- Sheridan, D. (2004). The optimality of a fast cnf conversion and its use with sat. SAT 2.
- Silva, J. P. M. et K. A. Sakallah (2003). Grasp—a new search algorithm for satisfiability. In The Best of ICCAD, pp. 73–89. Springer.
- Stamatelatos, M., W. Vesely, J. Dugan, J. Fragola, J. Minarick, et J. Railsback (2002). Fault tree handbook with aerospace applications.
- Stone, G. et R. Van Heeswijk (1977). Parameter estimation for the weibull distribution. IEEE Transactions on Electrical Insulation (4), 253–261.
- Tanaka, H., L. Fan, F. Lai, et K. Toguchi (1983). Fault-tree analysis by fuzzy probability. IEEE Transactions on reliability 32(5), 453–457.
- Tang, Z. et J. B. Dugan (2004). Minimal cut set/sequence generation for dynamic fault trees. In Reliability and Maintainability, 2004 Annual Symposium-RAMS, pp. 207–213. IEEE.
- Tapia, M., T. Guima, et A. Katbab (1991). Calculus for a multivalued-logic algebraic system. Applied Mathematics and Computation 42(3), 255–285.
- Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In Automation of reasoning, pp. 466–483. Springer.
- Vemuri, K. K., J. B. Dugan, et K. J. Sullivan (1999). Automatic synthesis of fault trees for computer-based systems. IEEE Transactions on Reliability 48(4), 394–402.

- Villemeur, A. (1988). Sureté de fonctionnement des systèmes industriels : fiabilité-facteurs humains, informatisation.
- Watson, H. et al. (1961). Launch control safety study. Bell labs.
- Weibull, W. et al. (1951). A statistical distribution function of wide applicability. Journal of applied mechanics 18(3), 293–297.
- Younes, H. L., M. Kwiatkowska, G. Norman, et D. Parker (2006). Numerical vs. statistical probabilistic model checking. International Journal on Software Tools for Technology Transfer 8(3), 216–228.
- Zaitseva, E. et V. Levashenko (2013). Multiple-valued logic mathematical approaches for multi-state system reliability analysis. Journal of Applied Logic 11(3), 350–362.
- Zio, E. (2013). System reliability and risk analysis. In The Monte Carlo Simulation Method for System Reliability and Risk Analysis, pp. 7–17. Springer.

Résumé

Cette thèse s'intéresse à la conception des systèmes critiques, dont le fonctionnement est impacté par des défaillances, qui pourraient être dangereuses pour les biens et les personnes qui l'entourent. Lors de sa conception, il est essentiel de réaliser une analyse de sûreté de fonctionnement pour déterminer les potentielles défaillances, leur criticité et leur probabilité d'occurrence. Cette analyse permet de statuer sur la confiance qu'il est justifié d'accorder au système et de renforcer le système si nécessaire. L'objectif de cette thèse est de faire intervenir les techniques de satisfiabilité pour préparer le calcul de la fiabilité du système : sa probabilité d'assurer sa mission pour un temps donné. Dans une première partie, nous nous intéressons aux systèmes statiques, ceux dont l'état (marche, arrêt) ne dépend que de l'état de ses composants. Nous modélisons le système par un arbre de défaillances, qui est un outil de modélisation très répandu dans la communauté de la sûreté de fonctionnement. La fonction de structure est une formule décrivant les combinaisons de défaillances qui sont tolérées ou non par le système, qui peut être déterminée à partir de l'arbre de défaillances du système. Nous faisons appel aux techniques de satisfiabilité pour identifier les liens minimaux, sous-ensembles des composants dont le fonctionnement garantit le fonctionnement du système. Nous modélisons également le système par un diagramme de Hasse, qui représente l'état du système en fonction de l'état de ses composants. L'évaluation probabiliste du niveau de confiance accordé au système est basée sur le polynôme de fiabilité, obtenu à partir du diagramme de Hasse. Dans une seconde partie, nous considérons les systèmes dits dynamiques, pour lesquels l'ordre d'occurrence des défaillances impacte le fonctionnement du système. C'est par exemple le cas des générateurs électriques, dont la défaillance prive de courant les autres composants et les empêche d'assurer leur fonction. Afin d'adapter aux systèmes dynamiques l'approche développée dans la première partie, nous définissons les séquences de lien minimales, extension des liens minimaux aux systèmes dynamiques, que nous déterminons grâce aux techniques de satisfiabilité. Nous proposons également une adaptation du diagramme de Hasse aux systèmes dynamiques afin de déterminer leur fiabilité.

Mots clés : fiabilité, sûreté de fonctionnement, satisfiabilité, arbres de défaillances, évaluation probabiliste, systèmes critiques.

Abstract

This thesis focuses on designing critical systems, those functioning is impacted by failures that could be dangerous for goods and people. During its design, it is crucial to convey a dependability analysis in order to determine the potential failures, their criticity and their probability of occurrence. The aim of this thesis is to involve satisfiability techniques in the computation of the reliability of the system : its probability to ensure its mission for a given time. In the first part, we consider static systems, those for which the functioning only depends on the functioning of their components functioning. We model the system by a fault tree, which is a widespread modeling tool in the reliability community, from which we can obtain the structure function. The structure function is a formula describing the combinations of components failure which are tolerated or not by the system. We also model the system by a Hasse diagram, which represents the states of the system depending on the states of the components. The probabilistic assessment of the trust placed to the system is based on the reliability function determined from the Hasse diagram. In the second part, we consider dynamic systems, for which the order between failures has an impact on the system. For example, it is the case for electric generators, which deprive the other components of electricity when they fail. In order to adapt to dynamic systems, the approach developed of the first part, we define minimal tie set sequences as the extension of minimal tie sets for dynamic systems, and we compute them by using satisfiability techniques. We also propose an adaptation of Hasse diagrams for dynamic systems to determine the reliability function.

Keywords : reliability, dependability, satisfiability, fault trees, probabilistic assessment, critical systems.