Formalisation de la cohérence et calcul des séquences de coupe minimales pour les systèmes binaires dynamiques et réparables

Pierre-Yves Chaux

▶ **To cite this version:**

Pierre-Yves Chaux. Formalisation de la cohérence et calcul des séquences de coupe minimales pour les systèmes binaires dynamiques et réparables. Autre. École normale supérieure de Cachan - ENS Cachan, 2013. Français. NNT : 2013DENS0014 . tel-00910331

HAL Id: tel-00910331

https://theses.hal.science/tel-00910331

Submitted on 27 Nov 2013

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.

ENSC-2013-N°443

THESE DE DOCTORAT

DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

présentée par

Monsieur Pierre-Yves CHAUX

pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Domaine :

ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE

Titre :

Formalisation de la cohérence et calcul des séquences de coupe minimales pour les systèmes binaires dynamiques réparables.

Thèse présentée à Cachan, le 15 Avril 2013, devant le jury composé de :

Jean-François PETIN	Professeur des Universités, CRAN - Université de Lorraine	Président
Christophe BERENGUER	Professeur des Universités, GIPSA-lab - Grenoble INP	Rapporteur
Antoine RAUZY	Chargé de recherches CNRS - HDR, LIX - École Polytechnique	Rapporteur
Jean-Jacques LESAGE	Professeur des Universités, LURPA - ENS Cachan	Directeur de thèse
Jean-Marc ROUSSEL	Maître de Conférences, LURPA - ENS Cachan	Encadrant
Marc BOUISSOU	Chercheur Senior HDR MRI - EDF R&D	Encadrant
Gilles DELEUZE	Chercheur Senior MRI - EDF R&D	Encadrant

Laboratoire Universitaire de Recherche en Production Automatisée

ENS de Cachan/ EA 1385/Université de Paris Sud 11

61 avenue du Président Wilson - 94235 Cachan Cedex

Remerciements

C'est avec grand plaisir que j'exprime en avant-propos à ce mémoire mes remerciements à toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce mémoire. Les travaux de recherche présentés dans ce mémoire sont le résultat d'une collaboration CIFRE entre le groupe T52 du département MRI d'EDF R&D et le Laboratoire Universitaire de Recherche en Production Automatisée (LURPA) de l'ENS Cachan.

Je tiens tout d'abord à remercier Jean-Jacques Lesage, pour la confiance qu'il m'a accordée à travers toute la durée de cette thèse en acceptant de diriger ces travaux. Je le remercie pour sa patience et pour ses grandes qualités humaines et scientifiques qui ont permis une collaboration toujours agréable. Je remercie également Jean-Marc Roussel, d'abord pour m'avoir accueilli au LURPA et d'avoir accepté le rôle de co-encadrant, puis pour sa disponibilité et son support tout au long de mes travaux de Master puis de Thèse.

Je remercie Gilles Deleuze de m'avoir accueilli dans le groupe T52 et de m'avoir donné le gout à la résolution de problèmes répondant à des besoins industriels forts, tout cela dans une ambiance bonne enfant. Je remercie également Marc Bouissou pour l'apport scientifique qu'il a pu m'apporter au cours de mon séjour chez EDF R&D. D'une manière plus générale, je remercie tout le département MRI de l'accueil qui m'a été réservé.

Je remercie Jean-François Petin d'avoir accepté de participer à mon jury et d'en avoir tenu la présidence. Je remercie également Antoine Rauzy et Christophe Bérenguer d'avoir accepté le rôle de rapporteur pour ce mémoire. Je les remercie également pour les questions et discutions qu'ils ont apportées, autant sur les aspects formels de ces travaux que sur les perspectives pouvant en être tirées.

Les années passées au LURPA resteront gravées dans ma mémoire, grâce à toutes les personnes que j'ai pu y rencontrer. Pour cela, je les remercie pour la bonne humeur ré-

gnant dans le laboratoire, spécialement lors des (longues) discutions, parfois scientifiques, au coin café. Je tiens à remercier tout particulièrement les doctorants et amis dont certains m'ont accompagné pendant l'ensemble des sept années de ma scolarité ENS.

Je remercie mes associés dans l'achat et la restauration d'une maison dans le Morvan. Cette grande aventure humaine et Morvandaise m'aura permis de me changer les idées le temps de plusieurs Week-End, que ce soit à travers de la démolition ou de la construction.

Enfin, je remercie ma famille, qui aura eu la patience de me soutenir et de m'encourager tout au long de ces nombreuses années d'études, même dans les moments difficiles.

« Une civilisation sans la Science,
ce serait aussi absurde qu'un poisson sans bicyclette. »
P.Desproges (1939-88),
Humoriste Limousin.

Table des matières

Table des matières iii

Table des figures vii

Liste des tableaux ix

Introduction Générale. 1

1 Etat de l'art 5
1.1 Analyse qualitative de la fiabilité des systèmes binaires dynamiques et réparables. . . . 7
1.1.1 Systèmes binaires . . . 7
1.1.2 Systèmes statiques, systèmes dynamiques. . . . 8
1.1.3 Composants et systèmes réparables. . . . 8
1.1.4 Analyses qualitatives de la sûreté d'un système binaire dynamique réparable . . . 8
1.2 Approches structurelles. . . . 9
1.2.1 Arbres de défaillance statiques . . . 10
1.2.1.1 Fonction de structure . . . 11
1.2.1.2 Analyse qualitative . . . 12
1.2.1.3 Bilan . . . 13
1.2.2 Les arbres de défaillance dynamiques . . . 13
1.2.2.1 Modélisation par ADD . . . 13
1.2.2.2 Description de la défaillance des composants élémentaires 16
1.2.2.3 Représentation des scénarios dysfonctionnels . . . 17
1.2.2.4 Calcul des Séquences de Coupe Minimales . . . 20
1.2.2.5 Extension des ADD aux composants réparables . . . 22
1.2.3 Les Qualitative Fault Trees (QFT) . . . 23
1.2.3.1 Définition du modèle . . . 23
1.2.3.2 Expression des scénarios dysfonctionnels . . . 24
1.2.3.3 Analyse qualitative . . . 24
1.2.4 Les Boolean logic Driven Markov Processes (BDMP) . . . 25
1.2.4.1 Description des modes et des défaillances des composants élémentaires . . . 26
1.2.4.2 Représentation des scénarios dysfonctionnels. . . . 27
1.2.4.3 Définition et calcul des Séquences de Coupe Minimale . . . 28
1.2.5 AltaRica - Automates de mode - système de transitions gardées. . . . 29
1.3 Approches non structurelles. . . . 31

1.3.1 Chaînes de Markov. . . . 31
1.3.2 Réseaux Bayésiens. . . . 32
1.3.3 Réseaux de Petri. . . . 33
1.4 Bilan et contributions. . . . 34

2 Cohérence des systèmes dynamiques réparables - Séquences de Coupe Minimales 37
2.1 Problématique. . . . 39
2.2 Cohérence des systèmes statiques, Coupes minimales. . . . 41
2.2.1 Cohérence des systèmes statiques. . . . 42
2.2.2 Définition des Coupes Minimales pour les systèmes statiques. . . . 44
2.2.3 Relation avec les arbres de défaillance statiques. . . . 44
2.3 Cadre formel pour l'analyse des systèmes dynamiques réparables. . . . 45
2.3.1 Représentation par automate fini de la fonction de structure pour les systèmes binaires dynamiques réparables. . . . 45
2.3.2 Scénarios caractéristiques du système. . . . 47
2.3.3 Modèle générique d'un composant réparable. . . . 51
2.3.4 Relation entre défaillance des composants et scénarios dysfonctionnels. 52
2.4 Analyse des acceptions courantes de séquences de coupes minimales 54
2.4.1 Critère d'arrêts à la première défaillance 54
2.4.2 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance 55
2.4.3 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance ou de réparation 55
2.4.4 Analyse des séquences d'un exemple 56
2.5 Cohérence d'un système dynamique réparable. . . . 58
2.5.1 Semi-cohérence d'un système dynamique réparable. . . . 59
2.5.1.1 Ajout d'un événement dans une séquence défaillante. . . . 60
2.5.1.2 Ajout d'un événement dans une séquence non-défaillante. . 63
2.5.1.3 Ajout d'une séquence dans une séquence dysfonctionnelle . 66
2.5.1.4 Bilan 69
2.5.2 Condition de cohérence d'un système semi-cohérent. . . . 72
2.6 Proposition d'énoncé informel des règles de cohérence des systèmes dynamiques réparables. . . . 73
2.7 Noyau du langage défaillant et Séquences de Coupes Minimales. . . . 74
2.7.1 Utilisation des règles de cohérence. . . . 74
2.7.2 Noyau du langage défaillant. . . . 76
2.7.3 Séquences de coupe Minimales. . . . 79
2.8 Calcul du noyau du langage défaillant et de l'ensemble des séquences de coupes minimales. . . . 81
2.8.1 Définition de l'ensemble des séquences sans circuits de l'AF. . . . 82
2.8.2 Algorithme de calcul du noyau du langage défaillant. . . . 85
2.8.2.1 Principe de calcul 85
2.8.2.2 Génération des séquences de l'ensemble $\mathcal{L}_F^{NL}$ à partir de l'AF 86
2.8.2.3 Calcul du noyau $Kern(\mathcal{L}_F)$ 89
2.8.2.4 Complexité du calcul du noyau. . . . 91

2.8.2.5 Calcul direct des séquences de coupe minimales. 93
2.8.3 Un exemple illustratif 95
2.9 Bilan. 96

3 Séquences de Coupe Minimales d'un système de composants multi-modes, application aux BDMP. 101
3.1 Introduction. 103
3.2 Modélisation de la sûreté d'un système par BDMP. 103
3.2.1 Hypothèses restrictives sur le modèle BDMP. 105
3.2.2 Analyse sémantique des BMDP. 106
3.2.2.1 Description logique de la structure d'arbre de défaillance. . 108
3.2.2.2 Description logique de la structure de gâchette. 109
3.2.2.3 Description événementielle du comportement défaillant des feuilles d'un BDMP. 112
3.2.2.4 Représentation d'un BDMP. 113
3.3 Génération de l'automate fini "équivalent" à un BDMP. 115
3.3.1 Description d'un BDMP par un automate fini. 115
3.3.2 Mécanisme de génération de l'automate. 116
3.4 Calcul des séquences de coupe minimales d'un système à composants multimodes. 120
3.4.1 Composants multimodes, cas des BDMP. 121
3.4.2 Séquences d'un système à composants multimodes. 122
3.4.3 Prise en compte des modes de composants pour le calcul du noyau. 122
3.5 Résultats sur le système à trois pompes. 126
3.6 Bilan. 128

4 Présentation d'un cas d'étude : Aval du condenseur d'une tranche nucléaire et système d'alimentation secouru 131
4.1 Introduction. 133
4.2 Présentation du cas d'étude. 133
4.3 Modèle BDMP du système. 137
4.3.1 Modèle des turbopompes. 137
4.3.2 Modèle des pompes d'extraction. 137
4.3.3 Modèle de l'alimentation électrique. 138
4.3.4 Modèle complet 140
4.4 Calcul du noyau du langage défaillant et des séquences de coupe minimales. 140
4.4.1 Calcul de l'automate équivalent. 141
4.4.2 Calcul des séquences du noyau, séquences de coupe minimales. . . . 141
4.4.2.1 Calcul des séquences de coupe minimales par le calcul du noyau du langage défaillant. 142
4.4.2.2 Analyse des résultats. 143
4.4.2.3 Calcul direct des séquences de coupes minimales. 144
4.5 Bilan 145

Conclusion & Perspectives 147

Bibliographie **151**

Annexes **i**

Table des figures

1.1 Exemple d'arbre de défaillance . . . 10
1.2 Représentation des portes d'un arbre de défaillance . . . 11
1.3 Représentation d'une porte PAND . . . 14
1.4 Représentation d'une porte SPARE . . . 14
1.5 Partage de composant entre deux portes SPARE . . . 15
1.6 Représentation d'une porte FDEP . . . 16
1.7 Traduction formelle du comportement d'une porte Spare [MRL11]. . . . 17
1.8 Représentation des scénarios d'un ADD par un failure automaton. . . . 19
1.9 Traduction d'une porte PAND en Well formed Stochastic Net [BR04] . . . 20
1.10 Arbre de défaillance dynamique représentant la sûreté du système HCAS [BD05]. . . . 21
1.11 Ajout de la réparabilité d'un composant par Repair Box[BR04] . . . 22
1.12 Modèle RdP d'une porte WARM SPARE prenant en compte la réparabilité du composant principal [BR04] . . . 23
1.13 Portes dynamiques utilisées dans les Qualitative Fault Trees(QFT) [WP09] 23
1.14 Table de vérité temporelle permettant la définition du comportement des portes d'un QFT [WP09] . . . 24
1.15 Exemple de BFT [WP09] . . . 25
1.16 Exemple de BDMP . . . 26
1.17 Type de feuilles d'un BDMP . . . 27
1.18 Processus de Markov associés à une feuille SF possédant des taux de défaillance différents dans son mode dormant (a) et actif (b) [BB03]. . . . 28
1.19 Exemple d'automate de mode pour la description de comportement d'une vanne [Rau02] . . . 30
1.20 Architecture du projet ALTARICA [PR12] . . . 30
1.21 Exemple de réseau Bayésien. . . . 32
1.22 Exemple de réseau de Petri modélisant une faute à l'aide d'une transition t_F(a) et son graphe d'accessibilité (b) [LS87] . . . 33

2.1 Détermination des séquences de coupe minimales des systèmes dynamiques réparables . . . 39
2.2 Exemple d'AF représentant une fonction de structure dynamique. . . . 47
2.3 Relation d'inclusion entre séquences de coupes et séquences défaillantes . . . 50
2.4 Modèle générique d'un composant réparable d'un système dynamique . . . 51
2.5 Exemple d'AF représentant une fonction de structure dynamique. . . . 57
2.6 Fonction de structure d'un système dynamique à 3 composants réparables sous forme d'automate. . . . 61
2.7 Exemple d'automate fini représentant une fonction de structure dynamique. 62

2.8 Ajout d'un événemement de réparation dans des séquences défaillantes. . . 63
2.9 Ajout d'événement de réparation dans une séquence non-défaillante. 64
2.10 Ajout d'événements de défaillance dans une séquence non-défaillante. . . . 65
2.11 Ajout des séquences f_3r_3 à travers la séquence f_1f_2. 68
2.12 Séquences générées par ajout de $\delta = f_3r_3$ dans $\sigma = f_1f_2$. 69
2.13 Ajouts possibles des séquences r_1f_1 dans la séquence f_1f_2. 69
2.14 Exemple de séquence défaillante $uabv$ passant deux fois par le même état . 78
2.15 Exemple de séquence défaillante passant deux fois dans le même état . . . 83
2.16 Relations entre le noyau du langage défaillant, l'ensemble des séquences défaillantes sans boucles et le langage défaillant. 85
2.17 Principe de calcul du noyau 86
2.18 Inclusion de σ' dans σ permettant l'identification de $(\delta, \sigma'_1, \sigma'_2, \sigma'_3)$ 91
2.19 Utilisation des algorithmes 91
2.20 Principe de calcul direct des séquences de coupe minimales. 94
2.21 Automate à 10 états, support de l'exemple 95
2.22 Relations d'inclusion entre les différents langages et ensembles de séquences 98

3.1 Exemple support pour l'analyse d'un BDMP 103
3.2 Modèle BDMP du système à trois pompes 105
3.3 Situation d'un noeud $N_i \in N$ d'un BDMP 109
3.4 Envelope du comportement d'une feuille L_i. 112
3.5 Structure de la représentation d'un BDMP. 113
3.6 Phases de calcul de l'automate fini equivalent au BDMP de la figure 3.2. . 118
3.7 Exemple d'un composant c sous n modes 121
3.8 Modèle de composant c d'un BDMP 122
3.9 Ajout d'un événement de défaillance dans une séquence défaillante de composants multimodes 123
3.10 Modification du mode du composant 3 à travers une séquence f_1f_2 124
3.11 Modification de l'automate pour l'adaptation de l'algorithme 3 aux composants multimodes 126

4.1 Architecture simplifiée d'un Réacteur à Eau Présurisée (REP) 134
4.2 Système Aval du condenseur. 135
4.3 Diagramme de fiabilité complet du système aval du condenseur. 135
4.4 Diagramme de fiabilité simplifié du système aval du condenseur. 136
4.5 Système d'alimentation secouru permettant l'alimentation des pompes d'extraction. 136
4.6 Modèle BDMP des turbopompes 137
4.7 Modèle BDMP des pompes d'extraction 138
4.8 Modèle BDMP du système d'alimentation 139
4.9 Modèle BDMP de la sûreté du système aval au condenseur. 140

Liste des tableaux

1.1 Récapitulatif de l'état de l'art sur les modèles d'analyse de la fiabilité des systèmes 35

2.1 Résultats de l'analyse des séquences $f_2 f_3 r_2 f_1$ et $f_1 f_2 r_1 f_3$ par les acceptions courantes de la définition du critère de minimalité 58
2.2 Récapitulatif des conditions de semi-cohérence d'un système dynamique réparable.(NC : non concluant, * : Cas mixte entre la distribution ordonnée des événements d'une séquence et l'ajout d'un événement de défaillance ou de réparation) 70
2.3 Inclusions possibles de σ par σ' en utilisant $\sigma = abcabc$ et $\sigma' = ac$. 90
2.4 Nombre de séquences défaillantes du système à 10 états 96
2.5 Séquences du noyau du système à 10 états 97

3.1 Propriétés des noeuds $N_i \in N^+$ du BDMP de la figure 3.2 108
3.2 Propriétés des gâchettes $T_i \in T$ du BDMP de la figure 3.2 108
3.3 Résultats du calcul des séquences défaillantes du système à trois pompes . 126
3.4 Séquences du noyau du système à trois pompes 127

4.1 Nombre et longueur des séquences de $\mathcal{L}_F^{NL}$,$\mathcal{L}_F^R$,$Kern(\mathcal{L}_F)$ et SCM 142
4.2 Nombre et longueur de séquences dans $\mathcal{L}_F^{NL} \cap \mathcal{L}_{CS}$ et $\mathcal{L}_F^R \cap \mathcal{L}_{CS}$, entrées de l'algorithme 3 pour le calcul direct des séquences de coupe minimales. . . . 145

Introduction Générale.

Contexte Industriel

Cette thèse a été réalisée dans le cadre d'une convention CIFRE (Convention Industrielle de Formation par la REcherche) entre le Laboratoire Universitaire de Recherche en Production Automatisée (LURPA) de l'École Normale Supérieure de Cachan et le département Management des Risques Industriels (MRI) d'EDF R&D.

En tant qu'exploitant de moyen de production d'électricité, EDF doit s'assurer du risque que représentent ses installations, autant pour leurs utilisateurs que pour l'environnement. Si cet enjeu de sûreté est commun à l'ensemble du parc de production, le niveau de sûreté à assurer sur les installations de la filière nucléaire est plus sévère. Le cadre de l'analyse des risques des installations est dans ce cas défini dans un périmètre précis, fixé par l'Autorité de Sureté Nuclaire (ASN).

Un des rôles d'EDF R&D est de développer des méthodes et outils permettant une modélisation toujours plus pertinente de la sûreté des installations, notamment pour la détermination et la représentation des scénarios de défaillance critiques impactant la sûreté des systèmes dynamiques de composants réparables. C'est dans ce cadre qu'ont été proposés en 2003 les *Boolean logic Driven Markov Processes (BDMP)*. Ce modèle permet d'une part la représentation de la relation entre la sûreté du système et la fiabilité de ses composants et d'autre part la modélisation des mécanismes de redondance complexes intervenant dans le système. L'instrumentation de ce modèle, également développée au sein d'EDF sur la plateforme KB3 depuis une quinzaine d'années, permet l'automatisation des études de sûreté de fonctionnement.

Cette thèse a été lancée dans le cadre d'un projet ayant pour objectif d'améliorer les méthodes d'analyse de la sûreté des systèmes de contrôle-commande. La difficulté de quantification précise des taux de défaillance de tels systèmes ainsi que leur comporte-

ment complexe ne permettent en effet pas toujours l'obtention de résultats concluants par des études quantitatives. C'est pourquoi EDF tient à développer les technique d'analyse qualitative sur des systèmes dynamiques et réparables, afin d'extraire les scénarios de défaillance les plus significatifs, en particulier lorsqu'ils sont représentés à l'aide de BDMP.

Contexte scientifique

De nombreuses méthodes d'analyse des riques ont été proposées depuis une trentaine d'années. On peut citer par exemple : les arbres de défaillance (Fault Tree Analysis [VGRH81]), les études FMEA (Failure Mode and Effects Analysis [Rei79]), HAZOP (HAZard and OPerability study [MNPF95]) ou les SMHA (State Machine Hazard Analysis [LS87]). Cette multiplicité est justifiée par la diversité des besoins et des objectifs des études de la sûreté d'un système. Ces méthodes diffèrent donc par leur domaine d'application et par la nature des résultats qu'elles permettent d'obtenir. Alors que, par exemple, HAZOP est une méthode d'analyse qualitative des risques par utilisation de mots clefs initialement développée pour les systèmes de procédés chimiques, les arbres de défaillance ont été développés pour l'étude des causes et des des probabilité de défaillance. Notre étude est focalisée sur les arbres de défaillance, dont les objectifs coïncident avec ceux d'EDF R&D.

Les arbres de défaillance ont été initialement proposés par [Wat61] dans les laboratoires Bell Telephone, puis décrits comme représentation graphique de la fonction de structure définie par [BES61] dans [Fus76]. Leur adoption par la NASA à travers le fault tree handbook [VGRH81] a permis une large diffusion de ce modèle dans le monde industriel, et compte encore à l'heure actuelle parmi les méthodes plus largement utilisées. Les arbres de défaillance permettent, par une approche graphique, la représentation des combinaisons de défaillances de composants qui conduisent à la défaillance du système. Cette relation est décrite à l'aide de portes logiques, représentation des opérateurs classiques de l'algèbre de Boole.

Les arbres de défaillance sont utilisés dans deux types d'études. Les études quantitatives ont pour objectif de déterminer la probabilité de défaillance du système sur un

temps de mission donné, connaissant les taux de défaillance des composants constituant le système. Les études qualitatives ont pour objectif l'obtention de l'ensemble des combinaisons minimales de défaillance de composants permettant d'obtenir la défaillance du système. Ces combinaisons également nommées *coupes minimales*, permettent d'une part de décrire l'ensemble des défaillances du système de façon la plus concise et d'autre part de faciliter la réalisation des études quantitatives.

Les principales limitations des arbres de défaillance sont de ne pouvoir modéliser que la sûreté des systèmes statiques et non réparables[VGRH81], ceci par la description de combinaisons non ordonnées de défaillances de composants de base. Afin de lever ces limitations, de nombreux travaux ont cherché à étendre ce modèle pour permettre la représentation de l'influence de l'ordre d'apparition des défaillances de composants sur la défaillance du système. Dès [Fus76], la porte PAND est décrite comme la première porte dynamique réagissant différemment suivant l'ordre d'occurrence de ses entrées. Sont ensuite venues de nombreuses extensions parmi lesquelles les Arbres de défaillance dynamiques (Dynamic Faut Trees [DBB92]) et les Quantitative Fault Tree [WP06], destinés aux études quantitatives et qualitatives de systèmes dynamiques non réparables, ainsi que les Boolean logic Driven Markov Processes (BDMP)[BB03] principalement utilisés pour les analyses quantitatives des systèmes dynamiques réparables.

Ces différents modèles permettent de conduire des études de sûreté approfondies pour les systèmes dynamiques, réparables (BDMP) ou non (ADD,QFT). Les scénarios de défaillances et réparations des composants qui conduisent à la défaillance du système étudié sont alors 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*. Alors que toutes les analyses qualitatives sont basées sur les séquences de coupe, aucune définition formelle, et encore moins consensuelle, n'a pu en être trouvée dans la littérature. Des ensembles différents de séquences de coupe minimales ont même été calculés pour un même système ([MRL11] et [TD04]).

Objectifs des travaux et principaux résultats.

Cette thèse a pour objectif principal la proposition d'un cadre formel permettant la conduite des études qualitatives de la sûreté des systèmes dynamiques et réparables, in-

dépendament du modèle de sûreté initialement utilisé (ADD sans prise en compte des réparations, BDMP ...). La principale hypothèse que nous avons retenue pour ce faire est que dans tous les cas un automate fini est suffisant pour décrire l'ensemble des scénarios constitués à partir des événements de défaillance et de réparation exprimés par ces modèles de sûreté. Nous avons basé sur cet automate fini, son langage généré (langage dysfonctionnel) et son langage marqué (langage défaillant), la définition et la formalisation des règles permettant de caractériser la *cohérence* des systèmes dynamiques réparables. Ce paradigme de cohérence nous permet ensuite de définir des représentations *finies* et *minimales* des deux ensemble infinis que sont le langage dysfonctionnel et le langage défaillant : le *noyau du langage défaillant* est la représentation finie et minimale de tous les scénarios conduisant à la défaillance du système ; l'ensemble des *séquences de coup minimales* caractérise l'ensemble des scénarios s'arrêtant à sa première défaillance.

Une méthode de calcul de ces deux ensembles est alors proposée, d'abord pour les systèmes dont les composants ne comportent qu'un seul mode de fonctionnement, puis généralisée aux systèmes de composants multimodes.

Plan du mémoire

Cette thèse est organisée en quatre chapitres. Le chapitre 1 propose une analyse de l'état de l'art des modèles dynamiques basés sur les arbres de défaillance, afin d'une part de justifier l'hypothèse de représentation des scénarios de défaillance par automate fini et d'autre part d'analyser les différentes acceptions de *séquences de coupe* et *séquences de coupe minimales* proposées dans la littérature. Le chapitre 2 rappelle alors la définition de la cohérence des systèmes statiques avant de proposer une définition et une formalisation de la cohérence des systèmes dynamiques et réparables de composants ne possédant qu'un seul mode de fonctionnement. A partir de cette formalisation de la cohérence, le *noyau du langage défaillant* et *l'ensemble des séquences de coupe minimales* sont définis avant de proposer un algorithme général permettant leur calcul. Afin de permettre l'analyse qualitative des BDMP, le chapitre 3 étend les résultats obtenus dans le chapitre 2 aux systèmes de composants possédant différents modes de fonctionnement et propose une méthode permettant l'obtention d'un automate fini équivalent à un BDMP. Enfin le chapitre 4 développe une étude de cas, fournie par EDF R&D, qui permet d'illustrer

les résultats proposés dans les chapitres précédents, d'en montrer la pertinence et leur aptitude à conduire des études de sûreté de taille significative.

Chapitre 1
Etat de l'art

Sommaire

1.1 Analyse qualitative de la fiabilité des systèmes binaires dynamiques et réparables. 7
1.1.1 Systèmes binaires 7
1.1.2 Systèmes statiques, systèmes dynamiques. 8
1.1.3 Composants et systèmes réparables. 8
1.1.4 Analyses qualitatives de la sûreté d'un système binaire dynamique réparable 8
1.2 Approches structurelles. 9
1.2.1 Arbres de défaillance statiques 10
1.2.2 Les arbres de défaillance dynamiques 13
1.2.3 Les Qualitative Fault Trees (QFT) 23
1.2.4 Les Boolean logic Driven Markov Processes (BDMP) 25
1.2.5 AltaRica - Automates de mode - système de transitions gardées. 29
1.3 Approches non structurelles. 31
1.3.1 Chaînes de Markov. 31
1.3.2 Réseaux Bayésiens. 32
1.3.3 Réseaux de Petri. 33
1.4 Bilan et contributions. 34

1.1 Analyse qualitative de la fiabilité des systèmes binaires dynamiques et réparables.

Cette section a pour but d'introduire les concepts et définitions permettant de préciser et de délimiter le cadre de nos travaux, qui sont relatifs aux études qualitatives de la sûreté des systèmes binaires *dynamiques* et *réparables*. Chacun de ces concepts sera développé plus en détail dans le chapitre 2. Les capacités et limites des modèles de fiabilité des systèmes seront alors analysées en considérant :

- leur capacité à représenter un système binaire.
- leur capacité à représenter un système dynamique.
- leur capacité à modéliser un système et des composants réparables.
- leur capacité à calculer une représentation des scénarios de défaillances et de réparation modélisés.
- l'existence d'une définition des séquences de coupes minimales.

1.1.1 Systèmes binaires

Dans le contexte de la sûreté de fonctionnement, un système binaire est un système dont la défaillance est représentée par une valeur booléenne. Elle est donc soit vraie (système défaillant) soit fausse (système opérationnel). Cette définition provient des travaux de Z.W. Birnbaum [BES61], généralisant les études de Moore-Shannon sur les circuits de relais [MS56]. Le cadre mathématique associé à ces études considère un système comme "un assemblage de composants reliés par une structure". La défaillance du système est alors associée à la perte de la possibilité d'alimenter une sortie par une entrée à travers les composants du système reliés par la structure.

Cette définition est restrictive, car elle ne permet pas la modélisation de mécanismes plus complexes de défaillance, comme par exemple la modélisation de la dégradation de la performance du système. Afin de répondre à de tels besoins, les méthodes de fiabilité des systèmes multi-états (Multi-States Systems reliability) ont été proposées par [Mur75, BW78], mais ne seront pas retenues dans le cadre de cette étude.

1.1.2 Systèmes statiques, systèmes dynamiques.

Un système statique est un système dont la défaillance à chaque instant est uniquement déterminée par la défaillance de ses composants au même instant, comme c'est le cas dans les travaux de Z.W. Birnbaum [BES61]. Les raisons pour lesquelles un système ne peut être qualifié de statique au sens de cette définition sont diverses :

- La structure du système évolue au cours du temps. Un tel système est alors qualifié de système multiphasé [AAS86, ZST99, Dug91]
- La structure du système est fixe, mais c'est l'ordre d'occurrence des événements de défaillance de ces composants qui le conduit, ou non, à la défaillance. Un tel système est un système dynamique [DBB92].

Dans cette étude seuls les systèmes dynamiques sont considérés.

1.1.3 Composants et systèmes réparables.

Un système réparable peut être défini comme "un système comportant au moins un composant réparable". Dans cette étude, les réparations des composants sont considérées *"as good as new"*, autrement dit, il n'est pas possible de différencier un composant n'ayant connu aucune défaillance d'un composant ayant été réparé. La réparation du système (passage d'un système défaillant à un système opérationnel, pour les systèmes binaires) est alors provoquée par la réparation d'un ou plusieurs composants défaillants de ce système.

1.1.4 Analyses qualitatives de la sûreté d'un système binaire dynamique réparable

Initialement développées pour les systèmes statiques, les études qualitatives de la sûreté d'un système ont pour objectif principal de déterminer les *coupes minimales*. Une coupe est définie par [BES61] comme un ensemble de composants défaillants conduisant à la défaillance du système ; ces coupes sont un résultat uniquement déterminé par la structure du système. La notion de minimalité a été formalisée par [Rau01]. Cette notion de coupe minimale est alors rattachée à la notion de *prime implicants* d'une fonction booléenne [MJ56] provenant de la minimisation de la fonction de structure (sous sa forme booléenne) et son expression sous la forme canonique d'une somme de produits.

Une telle définition ne convient pas pour l'analyse des systèmes dynamiques

et réparables. En effet, les études qualitatives ne vont pas se focaliser sur l'analyse des coupes mais sur l'analyse des scénarios dysfonctionnels. Un scénario dysfonctionnel est la représentation informelle d'une histoire possible dans le système, exprimée par une séquence d'événements de défaillance et de réparation des composants, et pouvant conduire, ou non, à la défaillance du système. Le résultat attendu de ces études ne sera donc pas des coupes minimales, mais des scénarios, nommés *séquences de coupe minimales*, qui permettront de décrire la relation entre les défaillances et réparations de composants et la défaillance du système. Afin de permettre cette analyse, une représentation de ces scénarios est nécessaire. Les sections suivantes vont donc chercher à établir l'existence d'une représentation des scénarios décris par un modèle d'analyse de sûreté ainsi que l'existence d'une définition de l'ensemble des séquences de coupe minimales.

1.2 Approches structurelles.

Les approches structurelles consistent pour la plupart à représenter le système comme un assemblage de composants en interaction à travers une structure. La structure du système ne faisant pas l'objet d'une définition formelle, elle est considérée dans cette étude comme étant la part de l'architecture permettant la représentation de la relation entre la défaillance des composants du système est la défaillance du système [BES61]. La forme la plus simple de cette représentation est le diagramme de fiabilité, où chaque composant est représenté par un interrupteur [MS56]. Ces modèles sont majoritairement construits par expertise et la modification du système implique la modification du modèle de fiabilité.

Afin de simplifier la construction de tels modèles, une représentation graphique de la structure unissant les composants et la défaillance du système est utilisée. Dans cette étude, les modèles structurels à base d'arbre de défaillance sont considérés. Leur capacité de représentation des systèmes dynamiques de composants réparables est analysée. Afin d'étudier la capacité des différents modèles pour la conduite d'étude qualitative sur la fiabilité d'un système, l'existence d'une représentation formelle permettant l'obtention de résultats (tels que des coupes minimales ou des séquences de coupe minimales) est également analysée.

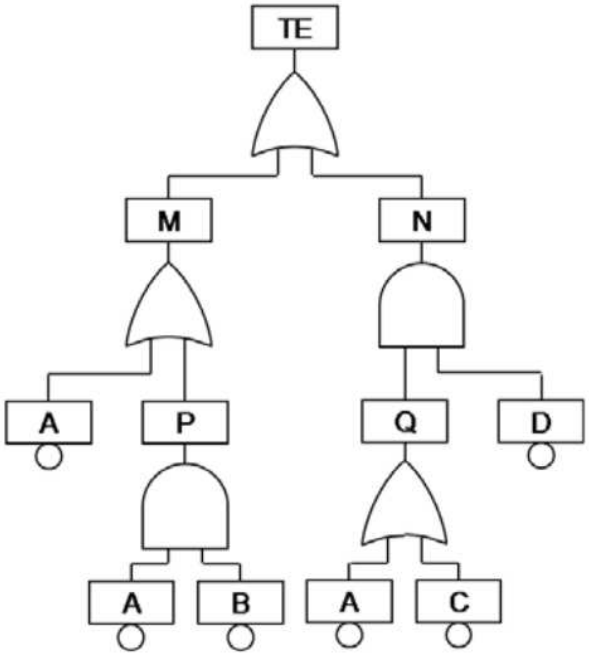


Figure 1.1 – Exemple d'arbre de défaillance

1.2.1 Arbres de défaillance statiques

Les arbres de défaillance (AD) [VGRH81] sont *un modèle de fiabilité pour les systèmes statiques* permettant l'analyse prévisionnelle des risques de systèmes industriels.

Ils se présentent sous une forme d'arbre qui, de façon graphique, lie par l'intermédiaire de portes logiques les défaillances de composants élémentaires à la défaillance du système. Comme présenté en figure 1.1, à la racine de l'arbre se trouve l'événement redouté (ou top event) représentant la défaillance du système, les feuilles de l'arbre représentant les défaillances des composants élémentaires (ou événements de base) du système.

La défaillance du système est supposée binaire ; le top event est alors représenté par une variable booléenne (vraie lorsque le système est défaillant). Les défaillances des composants du système sont également binaires et sont représentées chacune par une variable booléenne (vraie si le composant est défaillant). Les composants étant supposés indépendants, il n'existe aucune relation liant la défaillance d'un composant élémentaire aux défaillances des autres.

La construction d'un AD est réalisée grâce à des portes logiques statiques de trois

types :

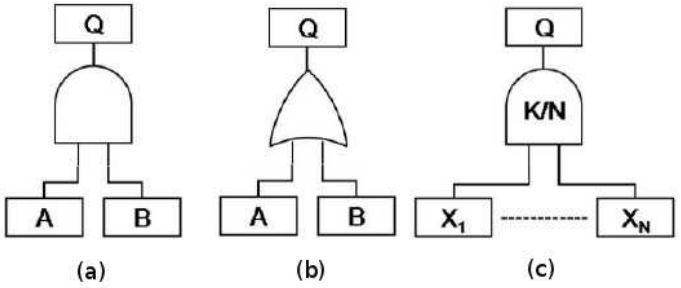


Figure 1.2 – Représentation des portes d'un arbre de défaillance

- La porte ET a sa sortie défaillante si les deux entrées sont défaillantes simultanément [VGRH81]. Sa primitive graphique est présentée en figure 1.2 (a).
- La porte OU a une sortie défaillante si au moins une des deux entrées est défaillante [VGRH81]. Sa primitive graphique est présentée en figure 1.2 (b).
- La porte K parmi N, également appelée porte votante, a une sortie défaillante si au moins un nombre donné K de ses entrées parmi les N existantes sont défaillantes. Elle est représentée en figure 1.2 (c).

1.2.1.1 Fonction de structure

Les portes décrites dans la section précédente sont associées aux opérateurs booléens communément utilisés par l'algèbre de Boole. La représentation de la relation liant les défaillances de composants élémentaires à la défaillance du système peut alors être réalisée à l'aide d'une fonction booléenne. Celle-ci est nommée *fonction de structure*. Son calcul part de l'événement redouté pour se diriger vers les événements de base. Ainsi sur l'exemple

de la figure 1.1 :

$$\begin{cases} TE = M + N \\ M = A + P \\ N = Q.D \\ P = A.B \\ Q = A + C \end{cases} \tag{1.1}$$

$$TE = (A + P) + Q.D \tag{1.2}$$

$$TE = (A + (A.B)) + (A + C).D \tag{1.3}$$

Cette fonction peut être simplifiée par utilisation du théorème d'absorption de l'algèbre Booléene en :

$$TE = A + C.D \tag{1.4}$$

1.2.1.2 Analyse qualitative

Les arbres de défaillance étant définis comme représentation d'une fonction de structure combinatoire, il est possible de réaliser des analyses qualitatives telles que décrites en section 1.1.4.

Ainsi sur l'exemple de la figure 1.1, la fonction de structure est donnée sous forme canonique par l'expression 1.4, les deux coupes minimales du système sont donc A et $C.D$. Cet ensemble de deux coupes minimales est suffisant pour la description qualitative de l'ensemble des défaillances du système.

L'information portée par les coupes minimales est double, elles décrivent d'une part toutes les possibilités de coupe permettant la défaillance du système et elles facilitent également le calcul de la probabilité de défaillance du système par l'utilisation de la formule d'inclusion-exclusion [Tri01].

Depuis son officialisation par [VGRH81] dans le Faut Tree Handbook et du fait de son succès, notamment parmi les industriels, des techniques ont été développées pour l'analyse des arbres de défaillance de très grande taille. La fonction de structure n'est alors plus représentée par une fonction booléenne mais par un arbre de décision binaire

(BDD) [RD97] ou par l'utilisation des zero-suppressed BDD décrite par [Min93],[Min01] dont l'utilisation est illustrée par [TD04].

1.2.1.3 Bilan

Les arbres de défaillance statiques présentent l'avantage de posséder une fonction de structure purement combinatoire. L'utilisation de cette fonction de structure a fait l'objet de nombreux travaux, autant dans sa représentation que dans le calcul des coupes minimales et leur utilisation. La limitation principale de ce modèle est son incapacité à représenter des mécanismes de défaillance dynamiques d'un système. Afin de pallier cette difficulté, des extensions aux arbres de défaillance ont été développées et seront pour les plus représentatives étudiées dans les sections suivantes.

1.2.2 Les arbres de défaillance dynamiques

Les arbres de défaillance dynamiques (ADD) (ou Dynamic Fault Trees (DFT)) sont un *modèle de fiabilité pour les systèmes binaires dynamiques de composants binaires non réparables*, initialement décrit par [DBB92, DSC00]. Basés sur les arbres de défaillance, ils permettent la représentation non plus d'un système statique mais dynamique. Cette capacité à représenter la relation entre l'ordre d'occurrence des défaillances de composants et la défaillance du système ou d'un sous-système provient de l'extension du modèle par l'ajout de portes dynamiques (PAND, SPARE, FDEP, SEQ) en plus des portes statiques (ET, OU, k/N). La valeur de sortie de ces nouvelles portes n'est plus seulement fonction de l'ensemble des composants défaillants en entrée de celles-ci, mais également de l'ordre dans lequel ces défaillances sont apparues.

1.2.2.1 Modélisation par ADD

Tels les arbres de défaillance statiques, les ADD sont basées sur une structure d'arbre décrivant la relation entre la défaillance des composants de base et la défaillance du système. Afin de prendre en compte l'ordre d'occurrence des défaillances, les portes PAND, SPARE, FDEP et SEQ ont été définies. Chacune de ces portes décrit une relation spécifique à l'exception de la porte SEQ dont il a été montré dans [BCS07] qu'elle pouvait être exprimée à l'aide de portes SPARE. Seules les portes PAND, SPARE et FDEP vont donc être décrites dans cette section.

1.2.2.1.1 La porte Priority-AND (PAND)

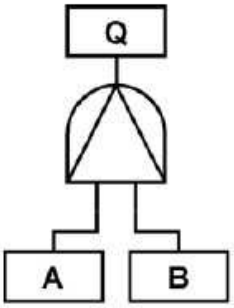


Figure 1.3 – Représentation d'une porte PAND

La porte PAND est l'extension d'une porte AND permettant la prise en compte de l'ordre d'occurrence des défaillances de ses entrées. Décrite initialement par [Fus76], elle est représentée en figure 1.3. Les défaillances des entrées de la porte PAND doivent occurrer par convention de gauche à droite sur leur représentation graphique. Ainsi sur la figure 1.3 pour que Q soit vrai, il faut que A et B soient vrais mais également que A soit vrai avant B.

Le rôle de cette porte est de permettre la description de la défaillance d'un sous-système (représenté par Q) conditionnée par l'ordre d'occurrence de la défaillance de ses composants (représentés par A et B).

1.2.2.1.2 La porte SPARE

La porte SPARE, décrite dans [DBB90] est représentée en figure 1.4. Cette porte possède une sortie et plusieurs entrées. Les entrées sont toutes des événements de base de l'arbre. De gauche à droite, la première entrée (A sur la figure 1.4) est l'entrée principale, elle décrit le composant principal du sous-système, initialement actif. Les autres entrées décrivent les composants de secours et sont initialement non-actives (dormantes). Les composants dormants peuvent connaître une défaillance avec un taux de défaillance nul, réduit ou égal au taux de défaillance en mode actif. Suivant le rapport de ces deux taux de défaillance, trois sous-types de portes SPARE sont définis :

– Les HOT SPARES : Le taux de défaillance en mode dormant est le même que celui

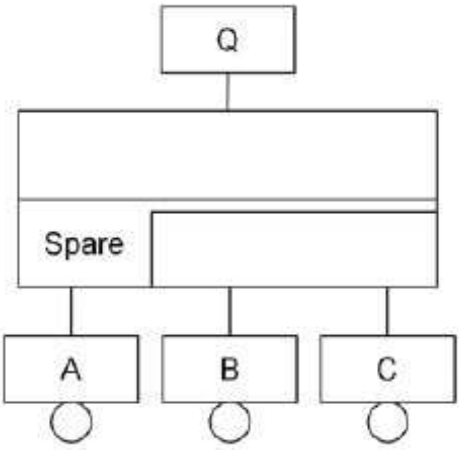


Figure 1.4 – Représentation d'une porte SPARE

en mode actif ;

- Les WARM SPARES : Le taux de défaillance en mode dormant est inférieur à celui en mode actif ;
- Les COLD SPARES : Le taux de défaillance en mode dormant est nul, le composant de secours ne peut donc pas défaillir en mode dormant.

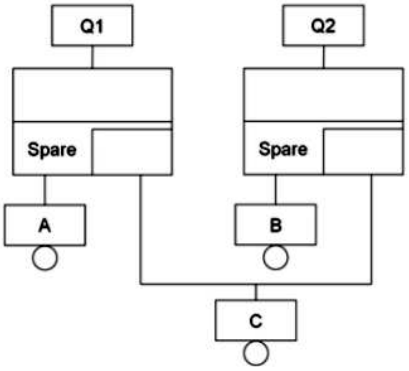


Figure 1.5 – Partage de composant entre deux portes SPARE

Le rôle de cette porte est de permettre la description des redondances (actives ou passives), mais uniquement entre composants. Les composants de secours peuvent être partagés entre plusieurs portes SPARE comme représenté en figure 1.5. Ce partage entraîne la définition d'un mécanisme de réservation du composant partagé par la première porte SPARE nécessitant son utilisation.

1.2.2.1.3 La porte FDEP

La porte FDEP permet de décrire la dépendance entre la défaillance de plusieurs composants. Cette porte n'a pas pour but de fournir une sortie de type défaillance mais de représenter cette dépendance, elle n'est donc pas associée à une sortie représentant la défaillance d'un sous-système. Elle est rattachée à la structure de l'arbre par un trait en pointillé. Cette porte est représentée en figure 1.6. Elle possède une entrée principale à gauche de celle-ci et des entrée secondaires. Les entrées secondaires sont toutes des défaillances de composants élémentaires. Ainsi sur la figure 1.6, la défaillance de l'événement principal T entraîne la défaillance instantanée et simultanée de toutes les entrées secondaires (A et B) [CSD00].

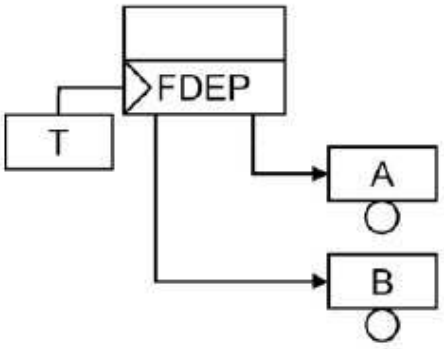


Figure 1.6 – Représentation d'une porte FDEP

Ce type de porte est principalement utilisé pour modéliser les causes communes de défaillance entre composants, comme par exemple le fait que deux composants électriques ont besoin d'une source d'énergie commune pour fonctionner. En cas de perte de cette source les composants alimentés sont donc considérés comme défaillants. L'intérêt de l'utilisation FDEP est uniquement graphique car il a été démontré que cette porte possède le même comportement qu'un ensemble de porte OU [MRL11], comme cela avait été suggéré dans [VSD+02].

1.2.2.2 Description de la défaillance des composants élémentaires

Tels les arbres de défaillance statiques, les arbres de défaillance dynamiques représentent la défaillance de chacun des composants élémentaires par une variable booléenne (vraie si le composant est défaillant, fausse sinon). Tous les composants élémentaires sont supposés non-réparables.

L'intégration de la porte SPARE dans les arbres de défaillance rend nécessaire la description du mode de fonctionnement de chacun des composants utilisés en secours par ces portes. La politique de réservation d'un composant par une porte SPARE doit également être décrite si ce composant est partagé entre plusieurs SPAREs (figure 1.5).

1.2.2.3 Représentation des scénarios dysfonctionnels

Un arbre de défaillance dynamique est une représentation informelle d'un ensemble de scénarios composés d'événements de défaillance de composants pouvant mener, ou non, à la défaillance du système. L'analyse de ces scénarios pour les études qualitatives va donc passer par une conversion des arbres de défaillances dynamiques (modèles informels) vers des modèles formels permettant une représentation explicite ou symbolique de ces scénarios. Deux types d'approches ont eu pour objectif de représenter ces scénarios : soit par le calcul d'une fonction de structure dynamique soit par l'utilisation d'un modèle à état.

1.2.2.3.1 Représentation des scénarios par une fonction de structure

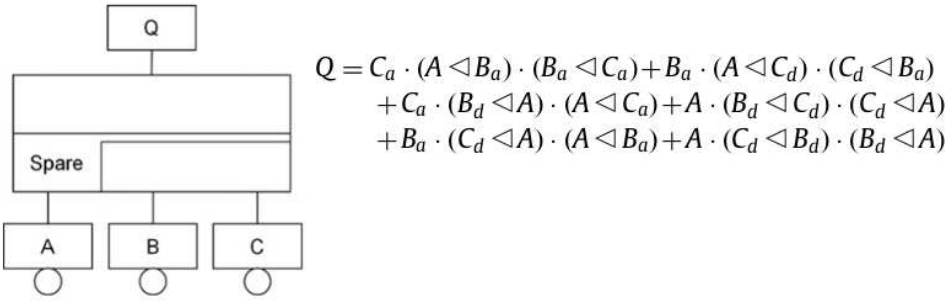


Figure 1.7 – Traduction formelle du comportement d'une porte Spare [MRL11].

La définition par [MRL11] de trois opérateurs temporels (Inclusive BEFORE ($\trianglelefteq$), non-inclusive BEFORE ($\lhd$) et SIMULTANEOUS($\triangle$)) permet la représentation de l'ensemble des scénarios de défaillance d'un arbre de défaillance dynamique sous forme d'une fonction de structure dynamique. L'utilisation de ces opérateurs nécessite que chacun des événements de base de l'arbre de défaillance dynamique soit associé à une date unique de défaillance. Afin de permettre l'écriture de cette fonction de structure, une extension de l'algèbre de boole est développée par association des opérateurs classiques de cette

algèbre avec les nouveaux opérateurs temporels. Ce formalisme propose une description formelle du comportement des portes dynamiques des ADD et des mécanismes de réservation des équipements de base par les portes SPARE et permet également la démonstration de nombreux théorèmes utiles pour le calcul de la fonction de structure (notamment le développement et la simplification des expressions utilisant ces opérateurs temporels).

Le comportement de chaque type de porte sur occurrence de défaillances est traduit dans une expression algébrique, un exemple de traduction d'une porte SPARE générique est présenté en figure 1.7. A partir des événements de base, et en remontant vers le top event, il est alors possible d'écrire la fonction de structure d'un ADD, relation entre l'événement redouté et les événements de base, tel qu'il est possible de le faire sur les arbres de défaillance statiques.

Cette méthode permet de plus la représentation de la fonction de structure sous une forme canonique (somme de mintermes), qui permet alors l'identification des séquences de coupe minimales sous forme de conditions que doivent respecter les scénarios pour permettre d'obtenir la défaillance du système.

1.2.2.3.2 Représentation des scénarios par un modèle à états.

Dès la définition des ADD, [DBB92, DSB93] présente une méthode de traduction automatique des ADD vers les chaînes de Markov. Cette méthode, illustrée dans [CSD00], passe d'abord par une représentation intermédiaire nommée *failure automaton*. Un *failure automaton* est un automate fini décrivant toutes les évolutions possibles dans l'arbre de défaillance dynamique sur occurrence des événements de base. Chacun des état code une combinaison de défaillance de de l'ensemble des composants et portes du système étudié. Un exemple de failure automaton est présenté en figure 1.8. Le *failure automaton* est ensuite converti en processus de Markov en associant à chacune de ses transitions un taux de défaillance de composant, permettant ainsi la réalisation des études quantitatives. Cette méthode est implémentée dans l'outil GALILEO [DSC00] en utilisant notamment la notion de module d'un ADD [GD97].

Une seconde possibilité est la description de l'ADD sous forme d'un réseau de Petri stochastique (SPN) [BR04]. Cette méthode est basée sur la paramétrisation de l'ADD

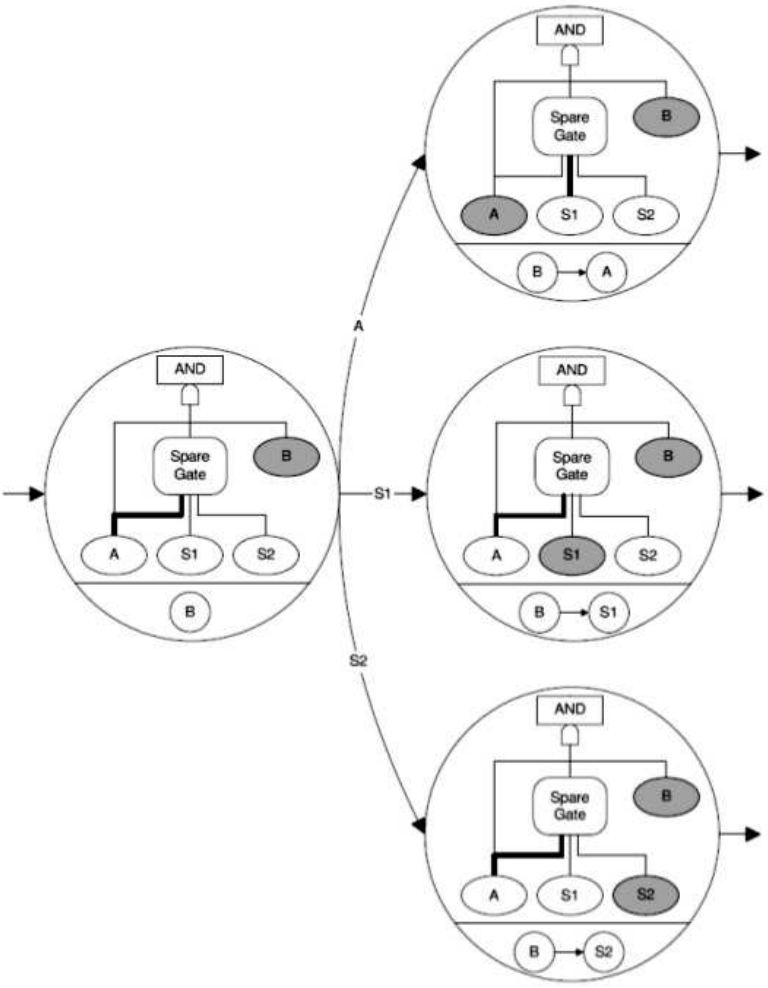


Figure 1.8 – Représentation des scénarios d'un ADD par un failure automaton.

afin d'en rendre la représentation plus compacte sous la forme d'un arbre de défaillance paramétrique (PFT). La seconde étape est la détection des modules du PFT [DR96] et leur classification en modules dynamiques ou statiques, suivant la présence ou non de portes dynamiques. Alors que les modules statiques sont analysés par les outils associés aux arbres de défaillance statiques, les modules dynamiques sont traduits en *Stochastic Well Formed Nets (SWN)*[CDFH93]. Cette traduction est réalisée en associant chacune des portes à un modèle SWN. Cette association est illustrée dans la figure 1.9 présentant le modèle SWN d'une porte PAND. L'objectif majeur de cette approche étant les études quantitatives, chacun des modules dynamiques sous forme de SWN est alors traduit vers un processus de Markov [DGM03] afin d'en calculer une probabilité de défaillance pour ensuite être remplacé par un événement de base dans le PFT. Les études qualitatives associées à ce modèle n'ont pas encore été développées, mais la représentation de l'ensemble des scénarios par un automate pourrait être réalisée par l'utilisation du graphe d'accessibilité associé aux

SWN [CDFH93]. Une étude similaire utilise cette fois une traduction de l'ADD complet vers un réseau de Petri généralisé stochastique [CR05], qui est traduit en processus de Markov [CDFH93] puis quantifié.

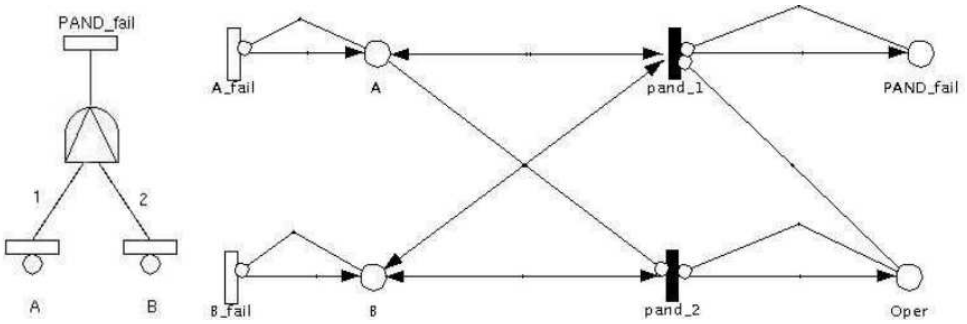


Figure 1.9 – Traduction d'une porte PAND en Well formed Stochastic Net [BR04]

1.2.2.4 Calcul des Séquences de Coupe Minimales

L'une des problématiques associées aux systèmes dynamiques dont l'étude de la sûreté est réalisée à l'aide d'arbre de défaillance dynamique est l'inexistence d'une définition universelle, ni même consensuelle, des séquences de coupes minimales.

Une première définition a été proposée par [TD04], celle-ci associe les séquences de coupe minimales d'un ADD aux coupes minimales d'un arbre de défaillance représentant la même structure que l'ADD analysé, mais sans les contraintes temporelles décrites par les portes dynamiques. Sur cet arbre de défaillance, les coupes minimales sont alors calculées par l'utilisation de Zero-suppressed BDD [Min93, Min01]. Après calcul de ces coupes minimales, les contraintes temporelles associées aux portes dynamiques de l'arbre de défaillance dynamique sont retranscrites sur les coupes minimales afin d'obtenir les séquences minimales. Cette méthode présente l'inconvénient de définir les séquences de coupe minimales comme étant uniquement un ordonnancement des événements présents dans une coupe minimale.

Une seconde définition a été donnée par [MRLB10, MRL11] à partir d'une représentation de la fonction de structure sous forme algébrique. Dans ces travaux, la fonction de structure est représentée algébriquement comme étant un ensemble de conditions que doit respecter une séquence pour conduire à la défaillance du système. La mise sous forme canonique de cette fonction de structure (somme de produits) permet d'identifier d'une

part les coupes minimales (termes combinatoires de la forme canonique) provenant de la partie purement statique de l'arbre et d'autre part les séquences de coupe minimales comme étant les termes dynamiques de la forme canonique.

Dans ces deux approches, les séquences de coupes sont obtenues sous leur forme implicite, ceci par un ensemble de conditions qu'elles doivent respecter, et non sous une forme explicite. De même, aucune réelle définition de ces séquences de coupe n'est proposée quant à leurs propriétés pour l'analyse du système, ce qui peut engendrer des écarts quand aux résultats obtenus. Ces écarts peuvent être illustrés sur l'exemple du HCAS (Hypothetical Cardiac Assist System [BD05]) dont l'arbre de défaillance dynamique et présenté en figure 1.10.

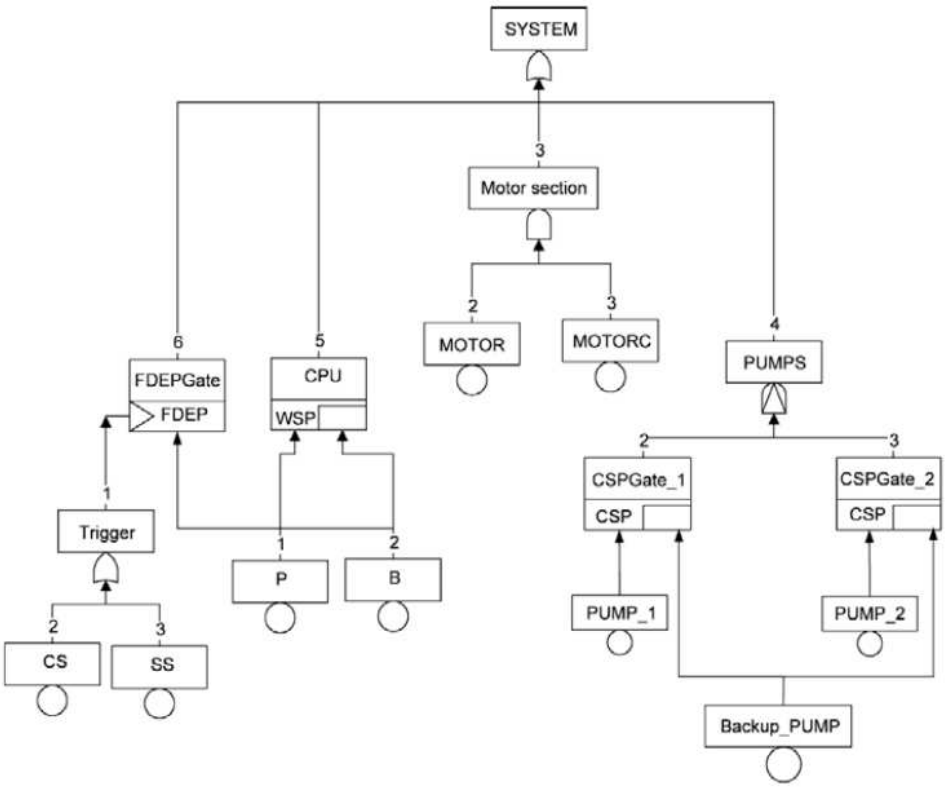


Figure 1.10 – Arbre de défaillance dynamique représentant la sûreté du système HCAS [BD05].

La méthode proposée par [TD04] est implémentée dans GALILEO et génère, après calcul, un ensemble de coupes minimales parmi lesquelles se trouve la coupe $Pump1$ $.Pump2$ $.BackupPump$. Par conséquent, toutes les séquences possibles par permutation

des événements de cette coupe minimale sont des séquences de coupe minimales. Or, d'après les résultats obtenus par [MRL11], seules les séquences $Pump1\ BackupPump_{active}\ Pump2$ et $Pump2\ Pump1\ BackupPump_{active}$ sont des séquences de coupe minimales. $Pump1.Pump2.BackupPump$ n'est donc pas une coupe minimale. Cette différence s'explique car certaines des séquences de coupe minimales obtenues par [TD04] à travers la coupe minimale $Pump1.Pump2.BackupPump$ ne sont pas possibles dans le modèle, mais ont été générées par la méthode utilisée. Les résultats obtenus sont conservatifs dans le sens où toutes les séquences de coupe minimales sont représentées, mais ils ne constituent pas une représentation minimale de ces séquences.

Ces approches font l'hypothèse de non-réparabilité des composants, elles ne pourront donc pas être la base de la définition des séquences de coupe minimales des systèmes dynamiques réparables.

1.2.2.5 Extension des ADD aux composants réparables

Afin d'étendre les capacités de modélisation des ADD aux composants réparables, [BR04] introduit le concept de repair boxes (figure 1.11). Associées aux événements élémentaires des ADD, elles permettent de modéliser un comportement de réparation du composant modélisé. Ce comportement et ses conséquences sur les portes de l'ADD sont pris en compte par modification des modèles des comportements des portes par réseau de Petri. Cette modification est présentée en figure 1.12 sur le modèle d'une porte WARM SPARE.

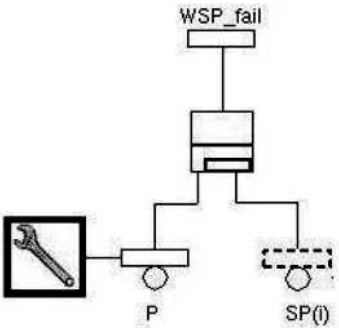


Figure 1.11 – Ajout de la réparabilité d'un composant par Repair Box[BR04]

La complexité du modèle croît avec le nombre de composants réparables modélisés.

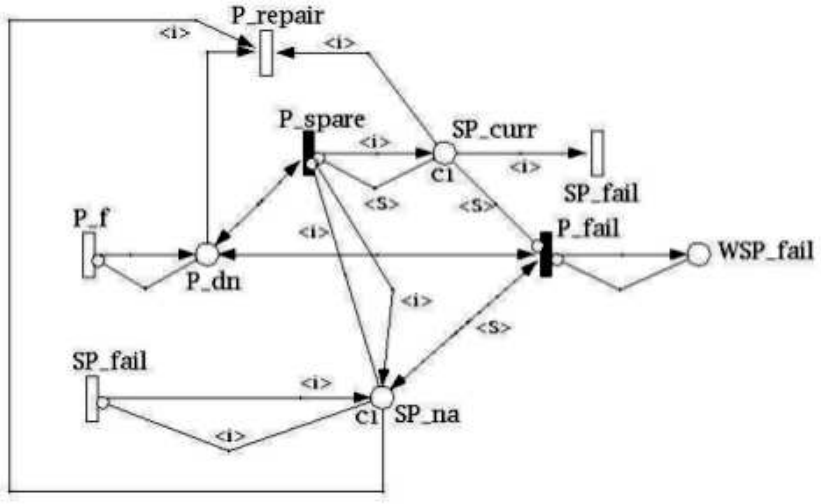


Figure 1.12 – Modèle RdP d'une porte WARM SPARE prenant en compte la réparabilité du composant principal [BR04]

Cette extension ne permet la prise en compte de la réparabilité que d'un nombre limité de composants et est bornée aux études utilisant la représentation réseau de Petri des arbres de défaillance dynamiques. Son objectif est donc borné à la représentation locale de la réparabilité, et ne constitue pas une méthode pour la modélisation d'un système réparable constitué uniquement de composants réparables.

1.2.3 Les Qualitative Fault Trees (QFT)

Une autre extension des arbres de défaillance statiques pour l'expression de la fiabilité d'un système dynamique et son analyse qualitative est proposée par les *qualitative fault trees(QFT)*. Cette extension nommée PANDORA [WP06] a pour but d'exprimer les conditions d'ordre entre les événements de base de l'arbre permettant d'obtenir la défaillance du système. Le concept de base utilisé pour cette extension est une redéfinition de la porte PAND permettant notamment de lever les ambiguïtés décrites dans [VSD+02].

1.2.3.1 Définition du modèle

Les QFT utilisent des événements de base non-réparables reliés par une structure d'arbre comprenant les portes statiques des arbres de défaillance ainsi que trois types de portes dynamiques (représentées en figure 1.13) : la Priority-AND, la Simultaneous-AND et la Priority-OR. Les deux premières (PAND, SAND) permettent de représenter toutes

Figure 1.13 – Portes dynamiques utilisées dans les Qualitative Fault Trees(QFT) [WP09]

les combinaisons de précédence entre deux événements et ainsi de déterminer quelle relation temporelle est nécessaire afin d'obtenir la défaillance. Ces relations, au nombre de trois, sont nommées *before, after* et *simultaneous*. La dernière porte (POR), proposée dans [WBP07], permet la représentation d'une porte OU exclusive (XOR) associée à une condition temporelle, telle que présentée dans [VGRH81]. Pour que la condition représentée par cette porte soit vraie, il faut que si le premier événement ait occurré, le second n'ait pas lieu ou ait lieu après le premier événement.

1.2.3.2 Expression des scénarios dysfonctionnels

X	Y	X+Y	X.Y	X<Y	X&Y	X|Y
0	0	0	0	0	0	0
0	1	1	0	0	0	0
1	0	1	0	0	0	1
1	1	1	1	0	1	0
1	2	1	2	2	0	1
2	1	1	2	0	0	0

Figure 1.14 – Table de vérité temporelle permettant la définition du comportement des portes d'un QFT [WP09]

Afin d'exprimer les séquences d'événements, chacun des événements de l'arbre est associé à son ordre d'occurrence. La date d'un événement X est notée $S(X)$, $S(X) = 1$ exprime que l'événement X est le premier à apparaître $S(X) = 2$ le second et $S(X) = 0$ signifie que l'événement n'apparaît pas dans la séquence considérée. Cette définition permet l'expression des relations décrites par les portes à travers la construction de tables de vérité temporelles (TTT). Ces tables permettent l'expression du comportement des portes et la démonstration de théorèmes sur les opérateurs. Un exemple de TTT est présenté en figure 1.14, cette table permet de définir le comportement des portes OU (+), ET (.), PAND (<), SAND (&) et POR (|).

1.2.3.3 Analyse qualitative

L'analyse qualitative d'un QFT est basée sur l'écriture d'une fonction de structure, relation entre l'événement redouté et les événements de base en utilisant les opérateurs de l'algèbre de Boole, les opérateurs associés aux portes dynamiques et sa mise sous forme canonique par utilisation de théorème. Cette forme canonique [WP07], nommée *base temporal form (BTF)*, est une somme de produits de doublets (figure 1.15). Chacun des doublets représentant une condition temporelle entre deux événements de base, chacun des termes de la somme représente une condition suffisante pour aboutir à la défaillance du système.

$$[X < Z].[Y > Z] + [X < Y].[YZ] + [X < Y].[Z < Y]$$

Figure 1.15 – Exemple de BFT [WP09]

1.2.4 Les Boolean logic Driven Markov Processes (BDMP)

Les Boolean logic Driven Markov Processes (BDMP) sont un *modèle de fiabilité pour les systèmes binaires dynamiques de composants binaires réparables*, basé sur les arbres de défaillance et initialement décrit par [BB03]. Tels les arbres de défaillance dynamiques, ils permettent la représentation de la sûreté d'un système dynamique et permettent également l'utilisation de composants réparables pouvant défaillir sous un mode dormant, actif ou à la sollicitation et la représentation de redondances où le système principal et le système de secours peuvent être un sous-système ou un composant. Ils ont été conçus dans le but de générer une chaîne de Markov par utilisation du langage de description FIGARO [BBBV91]. Ils ont donc été plus particulièrement conçus dans l'objectif de conduire des analyses quantitatives plutôt que des analyses qualitatives.

Un BDMP (dont un exemple est présenté en figure 1.16) est caractérisé par 4 éléments :

- Le Top event, au plus haut niveau du BDMP représente la défaillance du système,
- Une structure d'arbre de défaillance, utilisant des portes OU, AND, K/N et PAND,
- Des gâchettes (flèches en pointillés) représentant les liens de redondance,
- Les feuilles, au plus bas niveau représentent les comportements de défaillance et de changement de mode des composants du système.

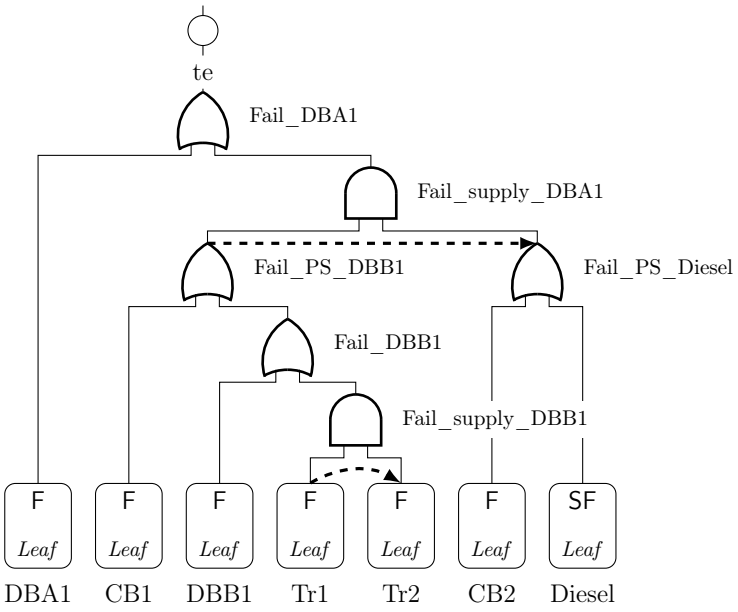


Figure 1.16 – Exemple de BDMP

Sur l'exemple de la figure 1.16, les composants du système sont au nombre de 7 sont reliés par 6 portes. Deux redondances sont présentes dans ce système. La première est celle reliant les composants Tr1 et Tr2 ; Tr1 est le composant principal car placé à l'origine de la gâchette, Tr2 est le composant de secours car placé sous la destination de la gâchette. La seconde redondance est celle entre le sous-système représenté par la sortie de la porte $Fail_PS_DBB1$ et celui représenté par la sortie de la porte $Fail_PS_Diesel$.

1.2.4.1 Description des modes et des défaillances des composants élémentaires

Chaque feuille du BDMP modélise la défaillance et le changement de mode d'un composant. Chaque composant est donc caractérisé par deux variables booléennes :

- Sa défaillance, vraie si le composant est défaillant, est déterminée en fonction de l'état du composant.
- Son mode, vrai si actif et faux si dormant, est déterminé à l'aide des gâchettes du BDMP. Le mode d'un composant représente son utilisation par le système. Il est donc *Actif* si le composant est utilisé par le système et *Dormant* sinon. La différenciation entre ces deux modes est nécessaire lorsqu'un composant est utilisé en secours. Un composant est en mode dormant jusqu'à ce que la défaillance du composant principal le fasse passer en mode actif.

Sur l'exemple de la figure 1.16, les composants toujours actifs (et donc principaux) sont DBA1, CB1 DBB1 et Tr1. Pour que Tr2 soit actif il faut que Tr1 soit défaillant. Pour que CB2 ou Diesel soient actifs, il faut que le sous-système représenté par la porte $Fail_PS_DBB1$ soit défaillant, c'est-à-dire que la sortie de cette porte soit vraie.

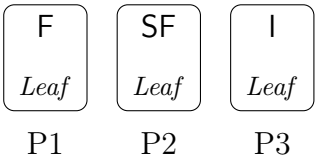


Figure 1.17 – Type de feuilles d'un BDMP

Les feuilles peuvent être de trois types, présentés en figure 1.17.

- Les feuilles de type F représentent un composant qui ne peut défaillir que lorsqu'il est en mode actif.
- Les feuilles de type SF représentent un composant qui peut défaillir lorsqu'il est en mode actif ou en mode dormant. Les taux de défaillance associés à chacun des modes peuvent être différents.
- Les feuilles de type I représentent la défaillance à la sollicitation d'un composant, cette défaillance ne peut intervenir que lors du passage du mode dormant au mode actif du composant.

1.2.4.2 Représentation des scénarios dysfonctionnels.

Aucune méthode n'a été développée sur les BDMP pour permettre la représentation formelle de tous les scénarios dysfonctionnels décrits par un modèle. Il est tout de même possible de connaître les scénarios dysfonctionnels élémentaires associés à chacun des composants du système, grâce à leur description par les processus de Markov associés, présentés en figure 1.18 [BB03]. Aucune méthode systématique ne permet alors, soit de reconstruire la chaîne de Markov équivalente au modèle BDMP considéré, soit d'obtenir une représentation de l'ensemble des scénarios dysfonctionnels décrits à l'aide d'un automate fini. Cependant, il est possible de calculer en extension un nombre fini de scénarios dysfonctionnels à partir de la représentation Figaro du modèle BDMP par utilisation de l'outil FIGSEQ [BBBV91]. Cet outil permet l'exploration de l'espace d'états à l'aide des règles d'évolution décrites par [BB03], également représentées à l'aide du langage

FIGARO. Cette méthode, orientée vers la quantification de la probabilité d'occurrence des scénarios, ne permet pas d'extraire l'ensemble de tous les scénarios possibles dans le système mais seulement un nombre fini de scénarios menant à la défaillance du système satisfaisont différent critères qui peuvent être :

- Le nombre de transitions traversées,
- Le nombre de défaillances ou de réparations présentes dans le scénario,
- La probabilité minimale du scénario,
- L'absence de passage dans un état déjà traversé par le scénario.

L'outil génère alors tous les scénarios répondant aux critères sélectionnés et s'arrêtant au premier état représentant la défaillance du système rencontré.

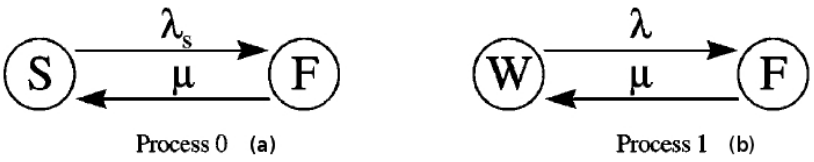


Figure 1.18 – Processus de Markov associés à une feuille SF possédant des taux de défaillance différents dans son mode dormant (a) et actif (b) [BB03].

1.2.4.3 Définition et calcul des Séquences de Coupe Minimale

Afin de conduire les études qualitatives sur les BDMP, une première définition des séquences de coupe minimales a été proposée par [Bou06]. Une séquence minimale est alors définie en s'appuyant sur un automate fini comme "*une séquence menant à la défaillance du système telle que si l'on retire un sous-ensemble non vide quelconque des événements de la séquence, on obtient une séquence refusée par l'automate.*", deux méthodes sont alors développées pour permettre le calcul des séquences de coupe minimale. La première consiste à extraire de l'automate les séquences sans boucle et de les comparer une à une. La seconde méthode est uniquement adaptée aux BDMP et permet le calcul sans passer par une représentation automate. La fonction de structure est alors utilisée afin de déterminer les coupes minimales de la structure, à partir de chacune de ces coupes, les séquences de coupe sont générées par ordonnancement des événements de défaillance permettant d'obtenir les composants défaillants décrits par la coupe.

Plusieurs remarques peuvent être faites sur ces méthodes :

- La première méthode est difficilement applicable car la représentation automate d'un BDMP n'ést pas définie formellement.
- La définition de la minimalité des séquences de coupe d'un BDMP s'appuie sur la minimalité des coupes d'un arbre de défaillance équivalent. Cette minimalité est alors uniquement définie sur le nombre de composants défaillant.
- Les séquences générées à partir d'un BDMP, ne s'appuyant pas sur un automate et étant un développement des coupes minimales par ajout des événements de défaillance associés au composants, elles ne peuvent comporter que des événements de réparation.
- L'ensemble des séquences minimales n'est pas défini vis-à-vis de son rôle pour le système.

Les BDMP présentent l'avantage d'être le seul formalisme (basé sur les arbres de défaillance) permettant de représenter de manière native et explicite la réparation des composants d'un système dynamique. C'est pourquoi, après avoir présenté dans le chapitre 2 une définition formelle et une technique de détermination des séquences de coupe minimale, indépendamment du modèle d'analyse de la sûreté utilisé, nous en proposerons dans le chapitre 3 une application au cas particulier des BDMP. Néanmoins, nous montrerons par une étude de cas développée dans le chapitre 4 que, même si la définition des séquences de coupes minimales proposée pour les BDMP [Bou06] constitue une base intéressante, elle en permet une bonne approximation mais ne permet pas de les déterminer exhaustivement.

1.2.5 AltaRica - Automates de mode - système de transitions gardées.

Le langage de description AltaRica, développé par [PR99], permet de décrire un système comme étant un assemblage de composants décrits par automate fini, reliés par une architecture hiérarchique. La sémantique associée à ce langage est basée initialement sur les automates de modes [Rau02], puis sur les systèmes de transitions gardées [Rau08].

Un exemple d'automate de mode modélisant le comportement et la défaillance d'une vanne est présenté en figure 1.19. Chaque composant est alors décrit par un ensemble fini d'états codant les valeurs de ses variables d'état. Les transitions traduisent alors les

variations des variables d'états lors de l'occurrence d'un événement. La transition peut être franchie si et seulement si la garde associée (expression combinatoire des variables d'état) est vérifiée.

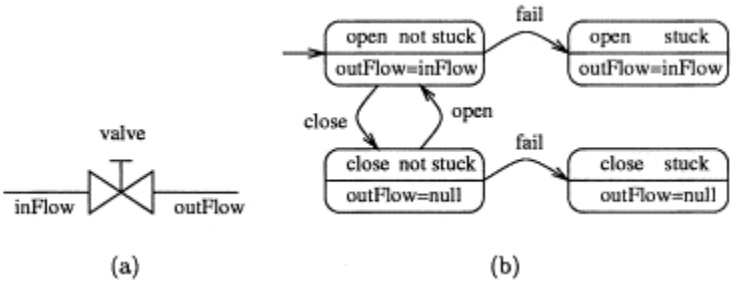


Figure 1.19 – Exemple d'automate de mode pour la description de comportement d'une vanne [Rau02]

Un avantage de ce formalisme est sa généricité, et donc la capacité de pouvoir générer de nombreux modèles ayants des objectifs différents à partir d'une description AltaRica d'un système. Il est donc possible d'obtenir une représentation sous forme d'espace d'état des scénarios possibles dans le système, que ce soit sous la forme de machine de Moore ou de Mealy ou de graphe de Markov lorsque les transitions modélisées sont stochastiques.

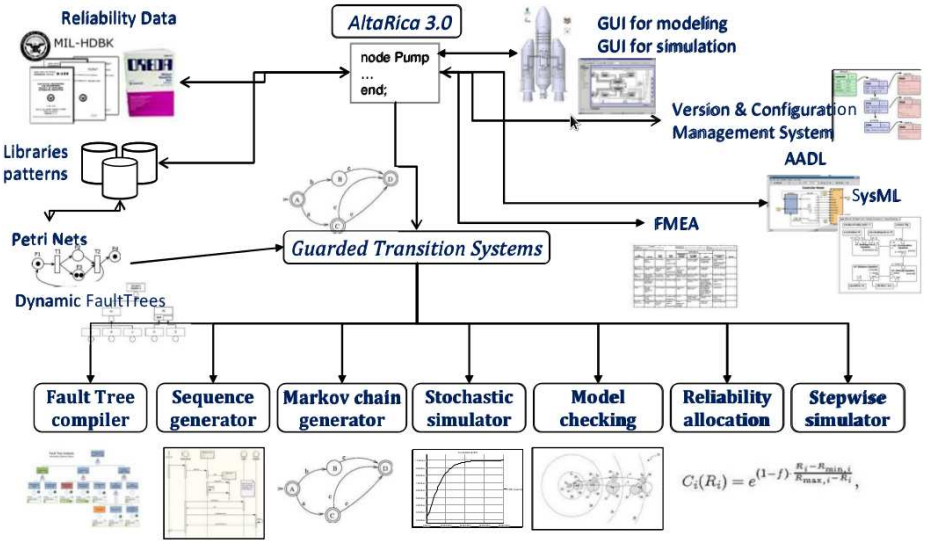


Figure 1.20 – Architecture du projet ALTARICA [PR12]

La relation entre les différents modèles est présenté en figure 1.20, Altarica est alors le langage pivot permettant la modélisation et la simulation des systèmes, associés à leurs

données fiabilistes. L'utilisation d'une représentation formelle par systèmes de transitions gardées permet alors d'effectuer différents calculs de fiabilité/disponibilité tels les arbres de défaillance ou les chaînes de Markov. Un tel modèle, par sa généricité, permet alors la représentation d'une part de modèles structuré type arbre de défaillance (statique ([Rau02]) ou dynamique ([PPRd10]) et d'autre part de tout modèle événementiel permettant la représentation de la défaillance du système tels les chaînes de Markov ou les réseau de Petri stochastique. De part sa popularité et par sa généricité, ce cadre formel pourrait être le support des travaux de cette thèse.

1.3 Approches non structurelles.

Sans prétendre proposer un panorama exhaustif des techniques d'analyse des risques, cette section s'étendra seulement sur la manière dont les scénarios d'événements de défaillance et de réparation peuvent être représentés à l'aide de modèles ne donnant pas une représentation explicite de la structure reliant les différents composants du système. Ces modèles ont pour objectif la représentation explicite ou implicite de l'ensemble des scénarios du système en passant éventuellement par une représentation de l'architecture du système.

L'utilisation de formalismes de haut niveau pour ces modèles permet la représentation des scénarios de défaillance pour des systèmes n'entrant pas dans le cadre de cette étude ; ils permettent ainsi pour certains de modéliser des systèmes dont la défaillance est non binaire ou des systèmes multiphasés. De plus, l'absence d'une représentation de la structure du système rend ces modèles moins adaptés pour les études qualitatives que pour les études quantitatives.

1.3.1 Chaînes de Markov.

Les chaînes de Markov ([How60]) sont un outil classique en sûreté de fonctionnement. Une chaîne de Markov est une modélisation d'un processus de Markov (cas particulier de processus stochastique), à l'aide d'un espace discret pouvant être représenté à l'aide d'un graphe orienté. Dans le cadre de la sûreté de fonctionnement cette représentation permet de représenter tous les scénarios d'événements stochastiques permettant de modéliser les

évolutions entre les états d'un système (représentés par le sommets du graphe).

L'inconvénient majeur de ce modèle est la difficulté de construction du graphe. Chaque sommet représentant un état du système, ce modèle est alors très sensible au phénomène d'explosion combinatoire lorsque la taille du système est importante. Il est alors quasiment impossible de construire manuellement la chaîne de Markov associée sans introduire d'erreurs. Néanmoins, les chaînes de Markov peuvent être générées de manière automatique à partir de différent modèles (par exemple à partir d'arbres de défaillance dynamiques). Ce sont alors plus particulièrement les analyses quantitatives qui sont visées.

1.3.2 Réseaux Bayésiens.

Les réseaux Bayésiens sont des graphes orientées acycliques permettant la représentation de variables aléatoires (associées aux noeuds du graphe) ainsi que les relations entre ces variables (à travers les arc du graphe) [Jen96]. Il permettent la représentation d'une loi de probabilité jointe sur l'ensemble des variables du système. Alors que les noeuds racine du réseau Bayésien sont associés à des tables de probabilités marginales, les relations causales sont décrites à l'aide de tables de probabilités conditionnelles. Un exemple de réseau Bayésien représentant trois variables est donné en figure 1.21. Les probabilité associées à chaque valeur possible pour la variable C sont déduites des probabilités associées aux valeurs des variables A et B

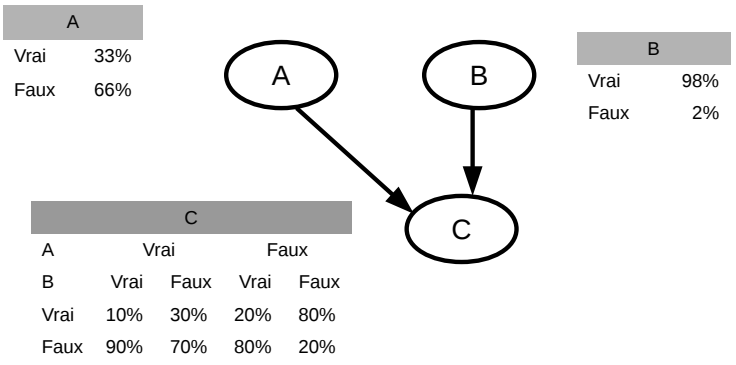


Figure 1.21 – Exemple de réseau Bayésien.

L'utilisation des réseaux Bayésiens dans le domaine de la sûreté de fonctionnement permet de modéliser un système uniquement de façon quantitative à travers ses variables [WMOSI12]. Cette vision quantitative permet également la modélisation de différents fac-

teurs (dont la maintenance, les facteurs humains et organisationnels) pouvant avoir une influence sur la sûreté du système.

Un des inconvénients des réseaux Bayésien est la complexité de leur construction, notamment à cause des tables de probabilités conditionnelles qui peuvent être de taille importante sur des réseaux complexes. Afin de palier cette difficulté, des méthodes de génération automatique ont été développées, notamment à base d'arbres de défaillance [BPMC01] ou d'arbres de défaillance dynamiques [BD05]. Puisqu'ils privilègent les études quantitatives et ne proposant pas de représentation événementielle des défaillances et réparations de composants, les réseaux Bayésien ne seront pas considérés dans ces travaux.

1.3.3 Réseaux de Petri.

Afin de pallier la difficulté de construction présente dans les chaînes de Markov, d'autres modèles ont été adaptés pour la représentation de processus stochastique. c'est par exemple le cas des réseaux de Petri (issus des travaux de [Pet66], formalisés dans [Pet81]), qui après avoir été utilisés dans la modélisation du comportement de systèmes physiques ou logiciels [AJSM84], ont été étendus pour prendre en compte les défaillances pouvant intervenir dans ces systèmes.

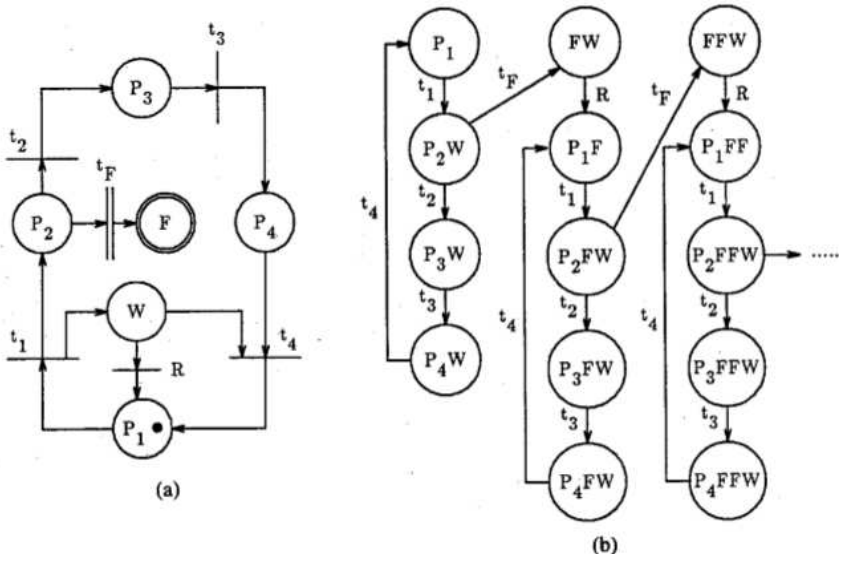


Figure 1.22 – Exemple de réseau de Petri modélisant une faute à l'aide d'une transition t_F(a) et son graphe d'accessibilité (b) [LS87]

Deux types d'approches ont alors été développées : la modélisation des défaillances par des pertes ou des ajouts de jetons dans les places [MF76], ou par ajouts de transitions permettant la modélisation des événements de défaillance [LS87].

A partir d'un réseau de Petri, il est alors aisé d'accéder aux scénarios possibles dans le système (dont certains peuvent le conduire dans un état défaillant) en générant son graphe d'accessibilité. Un exemple d'une telle approche est illustré sur la figure 1.22. Dans le cas général, un tel graphe va comporter en même temps des transitions associées au comportement légal du système ainsi que des transitions associées aux événements de défaillance et de réparation des composants. Lorsque l'ensemble de toutes les transitions du réseau de Petri sont stochastiques, le graphe d'accessibilité permet l'obtention de la chaîne de Markov associée comme cela est illustré sur les arbres de défaillance dynamiques ([CDFH93, DGM03, CR05], cf. section 1.2.2.3).

1.4 Bilan et contributions.

Ce chapitre nous a permis de passer en revue une sélection de méthodes permettant l'analyse qualitative des systèmes dynamiques de la fiabilité en s'attachant à l'évaluation :

- de leur capacité à représenter un système dynamique,
- de leur capacité à modéliser un système et des composants réparables,
- de leur capacité à calculer une représentation des scénarios modélisés,
- de l'existence d'une définition des séquence de coupes minimales.

Les résultats de cette étude bibliographique sont synthétisés dans le tableau 1.4.

Les travaux permettant une définition des séquences de coupe minimales sont naturellement ceux basés sur une représentation de la structure du système. Les modèles basés sur les arbres de défaillance sont parmi les plus populaires mais leur capacité à modéliser soit la dynamique du système soit la réparabilité du système et des composants est variable. Alors que la notion de coupe minimale est définie formellement sur les arbres de défaillance, la notion de séquence de coupe minimale l'est peu, ou imparfaitement, et dépend du modèle considéré. Les modèles proposant une définition de ces séquences sont les arbres de défaillance dynamiques, et les Qualitative fault trees, mais ces définitions restent limitées par la non réparabilité des composants. Les BDMP permettent la représentation

Tableau 1.1 – Récapitulatif de l'état de l'art sur les modèles d'analyse de la fiabilité des systèmes

Formalisme	S. non binaires	Sys. dynamiques	Sys. réparables	Représentation Scénarios	Séq. de coupe Minimales
Arbre de défaillance statique	Non	Non	/	/	/
Dynamic fault tree	Non	Oui	Possible par extension	Oui (Failure automaton)	Oui mais différentes définition
Qualitative fault tree	Non	Oui	Non	Non	Oui
Boolean logic Driven Markov Processes	Non	Oui	Oui	Oui	Non
Réseau de Petri	Oui	Oui	Oui	Oui	Non
Automates de modes - Altarica	Oui	Oui	Oui	Oui	Non

des systèmes dynamiques et réparables de manière native mais n'ont pas été associés à une méthode de représentation des scénarios décrits et la définition initiale des séquences de coupe minimales nécessite de faire abstraction de la réparabilité des composants.

L'objectif principal de nos travaux est de permettre l'obtention des représentations minimales de deux ensembles infinis de séquences : *le noyau du langage défaillant* et *l'ensemble des séquences de coupe minimales*. Ces deux ensembles seront définis comme les représentations minimales de respectivement l'ensemble des séquences conduisant à la défaillance du système et l'ensemble des séquences de coupe. L'hypothèse principale qui est faite dans cette étude est que, quelque soit le modèle de sûreté utilisé, les scénarios qu'il représente peuvent toujours être modélisés à l'aide d'un automate fini dont le langage généré et le langage marqué définissent ces scénarios. L'automate fini devient alors la représentation exhaustive de toutes les séquences possibles. Cette hypothèse est justifiée par le fait que la plupart des modèles basés sur la description de la structure (dont les AD, ADD, BDMP et les Automate de mode) ou non (dont les chaîne de Markov et les RdP) permettent de se ramener à cette représentation. Puisque la représentation par automate fini est indépendante du modèle de sûreté utilisé pour l'étude, cela va alors permettre d'obtenir une définition des séquences de coupe minimales, générique et indépendante du modèle.

Sous l'hypothèse que l'ensemble de tous les scénarios dysfonctionnels modélisés par un modèle de fiabilité dynamique peuvent être représentés par un automate fini, les contributions de cette étude sont :

1. La définition de la *cohérence* des systèmes dynamiques réparables.
2. La définition de *l'ensemble des séquences de coupes minimales*.
3. L'expression d'une méthode de *calcul systématique de ces séquences* à partir de l'automate fini.
4. La définition d'une méthode permettant l'obtention systématique d'une représentation de *l'ensemble des scénarios d'un BDMP par un automate fini équivalent*.
5. L'extension du calcul des séquences de coupe minimales aux *systèmes de composants multi-modes* pour l'application au cas des BDMP.

Chapitre

2

Cohérence des systèmes dynamiques réparables - Séquences de Coupe Minimales

Sommaire

2.1 Problématique. 39
2.2 Cohérence des systèmes statiques, Coupes minimales. 41
2.2.1 Cohérence des systèmes statiques. 42
2.2.2 Définition des Coupes Minimales pour les systèmes statiques. 44
2.2.3 Relation avec les arbres de défaillance statiques. 44
2.3 Cadre formel pour l'analyse des systèmes dynamiques réparables. 45
2.3.1 Représentation par automate fini de la fonction de structure pour les systèmes binaires dynamiques réparables. 45
2.3.2 Scénarios caractéristiques du système. 47
2.3.3 Modèle générique d'un composant réparable. 51
2.3.4 Relation entre défaillance des composants et scénarios dysfonctionnels. . 52
2.4 Analyse des acceptions courantes de séquences de coupes minimales 54
2.4.1 Critère d'arrêts à la première défaillance 54
2.4.2 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance 55
2.4.3 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance ou de réparation 55
2.4.4 Analyse des séquences d'un exemple 56
2.5 Cohérence d'un système dynamique réparable. 58
2.5.1 Semi-cohérence d'un système dynamique réparable. 59
2.5.2 Condition de cohérence d'un système semi-cohérent. 72
2.6 Proposition d'énoncé informel des règles de cohérence des systèmes dynamiques réparables. 73
2.7 Noyau du langage défaillant et Séquences de Coupes Minimales. 74
2.7.1 Utilisation des règles de cohérence. 74
2.7.2 Noyau du langage défaillant. 76
2.7.3 Séquences de coupe Minimales. 79
2.8 Calcul du noyau du langage défaillant et de l'ensemble des séquences de coupes minimales. 81

2.8.1 Définition de l'ensemble des séquences sans circuits de l'AF. 82
2.8.2 Algorithme de calcul du noyau du langage défaillant. 85
2.8.3 Un exemple illustratif 95
2.9 Bilan. 96

2.1 Problématique.

L'étude qualitative de la fiabilité des systèmes dynamiques à composants réparables passe par la détermination et l'étude des scénarios d'événements de défaillance et de réparation possibles dans le système. Contrairement aux études qualitatives sur les systèmes statiques pour lesquels le nombre de combinaisons de composants défaillants est fini, l'ensemble de tous les scénarios possibles dans les systèmes dynamiques réparables est de taille infinie. Cette infinité de scénarios est générée par la capacité de réparation des composants du système ; ceux-ci pouvant être réparés, ils peuvent donner lieu à une infinité de cycles d'événements de défaillance et de réparation.

L'analyse exhaustive des scénarios de défaillance et de réparation dans le système est donc impossible. Pour pallier cette difficulté, et restreindre l'étude qualitative à un ensemble fini de scénarios caractéristiques du système, cet ensemble fini de scénarios doit pouvoir garantir la conservation des propriétés de défaillance du système étudié. Pour ce faire, une notion de *cohérence* du modèle de fiabilité du système dynamique réparable est nécessaire, comme elle l'a été pour les systèmes statiques de manière à limiter leur étude qualitative à l'examen des coupes minimales.

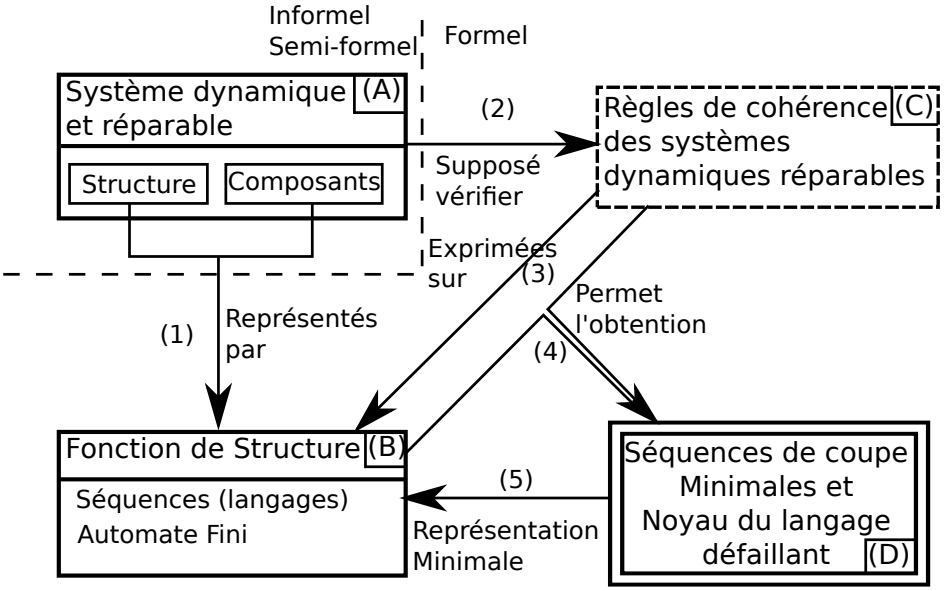


Figure 2.1 – Détermination des séquences de coupe minimales des systèmes dynamiques réparables

Le principe général de notre approche est présenté en figure 2.1. Dans le chapitre précédent, nous avons montré que la plupart des modèles de sûreté dynamiques décrivant des scénarios dysfonctionnels peuvent être représentés sous la forme d'automate fini. Nos ferons donc cette hypothèse de représentation par automate fini. Partant de cette hypothèse, l'ensemble des scénarios possibles dans un système (constitué d'une structure et d'un ensemble de composants (A)) dont certains peuvent atteindre la défaillance du système, est représenté par un ensemble de séquences décrit par un automate fini (B), cet automate sera considéré comme une représentation de la fonction de structure du système.

Afin de pouvoir définir une représentation minimale des ensembles de séquences possibles dans le système (D), l'utilisation de règles de cohérence adaptées aux systèmes dynamiques et réparables (C) est nécessaire. Les systèmes étudiés seront donc considérés comme cohérents (2). Ces règles de cohérence décrivent un ensemble de conditions sur l'automate fini et les séquences qu'il représente. Ces conditions ne peuvent être exprimées sur le système car elles nécessitent l'utilisation d'une représentation formelle de la structure, elles seront donc exprimées sur les séquences et l'automate fini (3). Ces règles de cohérence pourront alors être utilisées de pair avec la représentation automate de la structure afin de définir formellement les concepts de séquences de coupe minimales et de noyau du langage défaillant (4) et également de décrire leurs propriétés. Ce processus pourra alors valider ces deux ensembles comme représentations minimales de la structure du système représenté à l'aide d'un automate fini (5).

L'étude présentée dans ce chapitre va donc être décomposée de la manière suivante :

- La section 2.2 rappelle la définition de la cohérence des systèmes statiques et le cadre formel associé à cette définition. L'utilisation de cette définition pour l'identification des Coupes Minimales est ensuite abordée avant d'expliciter la relation avec les arbres de défaillance statiques.
- La section 2.3 propose alors un cadre formel basé sur la représentation des scénarios d'un système dynamique réparable en utilisant les automates finis. Après avoir justifié cette représentation, la relation entre ces scénarios et l'ensemble des composants défaillants à l'issue de ces scénarios ainsi que la définition des scénarios caractéristiques du système sont données.
- La section 2.5 définit alors la notion de cohérence des systèmes dynamiques dans le

cadre développé dans la section précédente.

– La section 2.6 propose un énoncé informel de cette cohérence, à l'image de l'énoncé informel classique de la cohérence des systèmes statiques.
– La section 2.7 définit la notion de noyau du langage défaillant, ensemble fini et minimal permettant la représentation de l'ensemble infini des scénarios menant à la défaillance du système, ainsi que les séquences de coupe minimales, sous-ensemble du noyau.
– La section 2.8 propose enfin un algorithme général permettant le calcul de ces deux ensembles.

2.2 Cohérence des systèmes statiques, Coupes minimales.

Tout système industriel est conçu pour réaliser au moins une fonction dans un environnement donné. La capacité du système à réaliser cette fonction dépend de l'état de fonctionnement des composants qui le constituent. Plus le nombre de ses composants défaillants est important, moins le système est susceptible de réaliser la fonction qui lui est associée. La relation entre le nombre de composants opérationnels et la fiabilité du système est associée à la notion de cohérence d'un système.

Cette notion de cohérence est principalement utilisée pour deux objectifs :

– Permettre à travers l'énoncé de règles simples une modélisation de la défaillance du système qui décrit une relation entre l'augmentation du nombre de composants défaillants et la diminution de la fiabilité du système.
– Identifier sur un système cohérent, les relations minimales entre les défaillances de composants et la défaillance du système.

Cette relation de cohérence a historiquement été définie sur les systèmes binaires multicomposants [BES61]. La définition d'un tel système est rappelée en section 2.2.1. Les règles de cohérence pour les systèmes statiques seront alors énoncées et discutées. La section 2.2.2 définira le concept de coupe minimale comme étant les relations minimales entre les défaillances de composants et la défaillance du système. La section 2.2.3 établit la relation entre ces systèmes et la représentation de leur défaillance par arbre de défaillance statique.

2.2.1 Cohérence des systèmes statiques.

Les travaux de Z.W. Birnbaum [BES61, BE65] ont été les premiers à introduire cette notion de cohérence sur les systèmes binaires. Un système binaire étant un système dont la défaillance est booléenne, il ne peut donc être que défaillant ou bien opérationnel. Partant de la description d'un système comme étant constitué d'un ensemble de composants reliés par une structure, il définit les concepts de semi-cohérence et de cohérence sur les systèmes binaires multicomposants.

Le modèle utilisé pour chacun des composants i du système est une représentation de leur état de fonctionnement par une variable Booléenne x_i : $x_i = 0$ représente un composant défaillant et $x_i = 1$ un composant opérationnel. L'état de tous les composants est décrit par un vecteur de dimension n (ou n est le nombre de composants) $\mathbf{x} = (x_1, x_2, ..., x_n)$.

La fiabilité du système dépend de la structure du système. Dans le cas des systèmes binaires, cette structure est représentée par la fonction de structure $\Phi(\mathbf{x}) = \{0, 1\}$. Elle ne dépend que du vecteur d'état des composants $\mathbf{x}$. Cette fonction est égale à 0 si le système est défaillant et à 1 si le système est opérationnel. Cette définition dépend uniquement de l'état des composants (et pas de l'ordre d'occurrence des défaillances), elle n'est donc adaptée qu'à l'étude des systèmes statiques de composants binaires. La fonction de structure $\Phi(\mathbf{x})$, a alors été représentée à l'aide d'une fonction Booléenne des états x_i des composants permettant une représentation formelle de la relation entre l'état du système et l'état de ses composants dans le cadre des systèmes binaires statiques.

Deux catégories de vecteurs $\mathbf{x}$, caractérisées par la valeur de la fonction de structure, sont d'intérêt pour l'étude de la fiabilité d'un système :

- *les chemins*, vecteurs $\mathbf{x}$ tels que le système est opérationnel ($\Phi(\mathbf{x}) = 1$).
- *les coupes*, vecteurs $\mathbf{x}$ tels que le système est défaillant ($\Phi(\mathbf{x}) = 0$).

Pour la plupart des systèmes, un accroissement du nombre de composants défaillants implique une diminution de la fiabilité. Soient deux vecteurs d'état de composants $\mathbf{x}$ et $\mathbf{y}$, tels que $\mathbf{x}$ comporte au moins un composant défaillant de plus que $\mathbf{y}$ et tous les composants défaillants en $\mathbf{y}$ le sont également en $\mathbf{x}$. La relation entre ces deux vecteurs d'état peut

être exprimée par ($\mathbf{y} > \mathbf{x}$). Si le système est opérationnel en $\mathbf{x}$ alors il ne peut être rendu défaillant en $\mathbf{y}$ par une augmentation du nombre de composants opérationnels. De même si le système est défaillant en $\mathbf{y}$ alors l'augmentation du nombre de composants défaillants en $\mathbf{x}$ implique que le système est défaillant. Ces règles de variation sont associées à la monotonie de la fonction de structure $\Phi(\mathbf{x})$. Un système respectant ces règles est dit *semi-cohérent* et est défini par la proposition :

$$\forall \mathbf{y} > \mathbf{x}, \Phi(\mathbf{y}) \geq \Phi(\mathbf{x}) \tag{2.1}$$

Si un système semi-cohérent est opérationnel lorsque tous les composants sont opérationnels ($\mathbf{x} = \mathbf{1} = [1, 1, ..., 1]$) et est défaillant lorsque tous les composants sont défaillants ($\mathbf{x} = \mathbf{0} = [0, 0, ..., 0]$) alors ce système est dit *cohérent*. Un système cohérent est défini par les valeurs de sa fonction de structure dans ces deux cas :

$$\Phi(\mathbf{0}) = 0, \Phi(\mathbf{1}) = 1$$

Afin de donner une définition informelle de la cohérence d'un système, un ensemble de *règles de cohérence* a été défini. Une énonciation courante de ces règles peut se trouver dans [LL03] :

- Règle 1 : Tous les composants du système sont pertinents.
- Règle 2 : La défaillance de tous les composants implique la défaillance du système.
- Règle 3 : Le bon fonctionnement de tous les composants implique le bon fonctionnement du système
- Règle 4 : Lorsque le système est défaillant, aucune défaillance additionnelle de composant ne peut rendre le système opérationnel.
- Règle 5 : Lorsque le système est opérationnel, aucune réparation de composant ne peut faire défaillir le système.

La règle 1 traduit la bonne définition de l'ensemble des composants pertinents pour représenter les défaillances du système. Si la fiabilité d'un composant du système n'influe pas sur la fiabilité du système, alors il n'est pas pertinent de le considérer dans une analyse des défaillances. Les règles 2 et 3 traduisent les conditions aux bornes de la fonction de structure, elles permettent de vérifier qu'un système semi-cohérent est cohérent. Les règles 4 et 5 traduisent les conditions de semi-cohérence à travers l'expression de la monotonie de la fonction de structure du système.

Il est important de rappeler que cette définition n'est valable que pour les systèmes statiques, même si l'énoncé informel des règles 4 et 5 peut faire penser à l'existence d'une relation temporelle entre la défaillance (ou le bon fonctionnement) du système et les ajouts de défaillance ou réparation des composants.

2.2.2 Définition des Coupes Minimales pour les systèmes statiques.

La définition de la fiabilité des systèmes binaires statiques et cohérents à l'aide d'une fonction de structure a permis l'émergence de travaux sur la définition et de calcul des *coupes minimales*.

Une *coupe minimale* est une coupe (vecteur d'état de composants $\mathbf{x}$ tel que le système est défaillant ($\Phi(\mathbf{x}) = 0$)) contenant un ensemble minimal de composants défaillants qui conduit à la défaillance du système. La propriété de semi-cohérence des systèmes cohérents permet alors de garantir la défaillance du système ($\Phi(\mathbf{y}) = 0$) lorsque le vecteur $\mathbf{y}$ considéré accroît l'ensemble des composants défaillants d'une coupe minimale $\mathbf{x}$.

La fonction de structure étant une représentation de la structure du système, une coupe minimale est considérée comme une propriété du système. La représentation de la fonction de structure par une fonction Booléenne a permis de rattacher la notion de coupe minimale à celle d'impliquant premier développé par E.J. McCluskey Jr dans le cadre de la minimisation de fonction Booléenne [MJ56]. Cette relation entre les impliquants premiers et les coupes minimales des systèmes statiques (cohérents ou non) est par ailleurs formalisée par [Rau01].

2.2.3 Relation avec les arbres de défaillance statiques.

La représentation par fonction booléenne de la fonction de structure a notamment permis la création d'une représentation graphique par arbre de défaillance statique [BC73, Fus76, VGRH81] dont l'objectif n'est plus la représentation de la fiabilité d'un système, mais de sa défaillance. La représentation de la défaillance du système passe alors par une seconde fonction de structure $\Phi'(\overline{\mathbf{x}})$, complément de la fonction de structure définie pour l'étude de fiabilité ($\Phi'(\overline{\mathbf{x}}) = \overline{\Phi(\mathbf{x})}$). Cette nouvelle fonction de structure est alors vraie

lorsque le système est défaillant, elle dépend de l'ensemble des défaillances de chacun des composants (représenté par le vecteur $\overline{\mathbf{x}}$).

2.3 Cadre formel pour l'analyse des systèmes dynamiques réparables.

La définition de la cohérence des systèmes statiques et le concept de coupe minimale présentés respectivement en sections 2.2.1 et 2.2.2 ne sont adaptées qu'aux systèmes statiques où la défaillance du système de dépend pas de l'ordre d'occurrence des défaillances des composants. La prise en compte de l'ordre d'occurrence des défaillances pour l'étude des systèmes dynamiques nécessite de se placer dans un cadre formel différent.

C'est pourquoi nos travaux proposent de se placer dans le cadre de la théorie des langages et des automates à états finis. La partie 2.3.1 justifiera ce choix et décrira la représentation associée à la fonction de structure. Les sections 2.3.3 et 2.3.4 s'intéresseront respectivement au modèle de composant utilisé dans ces travaux et à la relation entre les séquences de défaillances/réparation et l'ensemble des composants défaillants en fin de séquence. La section 2.3.2 décrira les scénarios caractéristiques du système permettant l'obtention ou non de la défaillance du système et donnera une première définition des séquences de coupe.

2.3.1 Représentation par automate fini de la fonction de structure pour les systèmes binaires dynamiques réparables.

Le lecteur trouvera en annexe les bases et notations relative à la théorie des langages utilisées dans ces travaux.

La prise en compte de la dynamique nécessite de s'intéresser à la fois aux ensembles de composants défaillants et à l'ordre dans lequel ces défaillances se produisent. Nous avons pour ce faire retenu la notion de scénarios de défaillance et de réparation (abrégés en "scénarios" dans la suite du document). Un scénario décrit un ensemble ordonné d'occurrences d'*événements de défaillances et de réparations* de chacun des composants. Si l'ensemble des événements de défaillance ou de réparation des composants est représenté

par un *alphabet* Σ alors un scénario est représenté par une séquence σ d'événements de Σ, $\sigma \in \Sigma^*$.

La représentation de la structure du système pour l'étude de sa défaillance va alors être associée à une *fonction de structure dynamique* qui sera notée $\Phi_d(\sigma)$. Celle-ci ne peut plus être exprimée comme une fonction du vecteur de l'état des composants $\mathbf{x}$, mais comme une fonction des séquences $\sigma \in \Sigma^*$. Les valeurs prises par cette fonction restent les mêmes : 0 si le système est défaillant et 1 si le système est opérationnel.

Nous avons choisi de représenter cette fonction de structure par l'utilisation des automates à états finis (AF) avec états marqués, définis à l'aide du 5-uplet suivant :

$$AF =< \Sigma, Q, q_O, Q_M, \delta > \tag{2.2}$$

Avec :

- Σ est l'alphabet des événements de défaillance et de réparation des composants.
- Q est l'ensemble des états du système.
- q_0 est l'état initial du système : dans la majorité des modèles de fiabilité tous les composants ainsi que le système sont opérationnels en situation initiale.
- Q_M est l'ensemble des états marqués. Ils représentent les états ou le système est défaillant ($\Phi_d(\sigma) = 0$) et sont représentés sur l'automate par un double cercle.
- δ est la fonction de transition ($\delta : (Q, \Sigma) \rightarrow Q$). Chacune des transitions n'est étiquetée que d'un seul événement de défaillance ou de réparation. Ainsi l'état q' atteint depuis l'état $q \in Q$ par occurrence de $\alpha \in \Sigma$ est noté $\delta(q, \alpha) = q'$. Par extension, l'état $p \in Q$ atteint depuis q par une séquence d'événement $\sigma \in \Sigma^*$ est noté $\delta(q, \sigma) = p$.

Les hypothèses associées à cet automate fini sont les suivantes :

- Pour qu'une étude des défaillances du système ait un sens, l'AF possède au minimum un état final représentant la défaillance du système.
- L'AF est déterministe et ne possède qu'un seul état initial q_0. Cette hypothèse permet de garantir qu'une même cause (représentée par une séquence d'événements de défaillance et de réparation) permet d'obtenir le même effet (l'état atteint par la séquence à partir de l'état initial est toujours le même).

– Les états ne sont pas équivalents deux à deux. Deux états q_1, q_2 sont dit équivalents si pour chacun des événements de Σ les états atteints à partir des deux états sont identiques ($\forall u \in \Sigma, \delta(q_1, u) = \delta(q_2, u)$). Cette hypothèse permet de garantir la minimalité de la représentation des scénarios et permet également d'associer les répétitions de séquence aux circuits de l'automate dans la section 2.8.1.

La valeur de la fonction de structure dynamique $\Phi_d(\sigma)$ est définie par le marquage de l'état atteint par la séquence σ à partir de l'état initial q_0. L'état atteint par la séquence σ à partir de l'état initial est désigné par $\delta(q_0, \sigma)$, cet état est unique car l'automate considéré est, par hypothèse, déterministe. Si l'état atteint est marqué, il représente une situation où le système est défaillant $\Phi_d(\sigma) = 0$, s'il n'est pas marqué alors il représente une situation ou le système est opérationnel $\Phi_d(\sigma) = 1$. La fonction de structure dynamique est alors définie par

$$\Phi_d(\sigma) = \begin{cases} 0 & \forall \sigma \in \Sigma^*, \delta(q_0, \sigma) \in Q_M \\ 1 & \forall \sigma \in \Sigma^*, \delta(q_0, \sigma) \notin Q_M \end{cases} \tag{2.3}$$

Un tel automate permet donc de représenter toutes les valeurs de la fonction de structure en fonction des scénarios de défaillance et de réparation des composants, ceci indépendamment du modèle utilisé en amont pour représenter la sûreté du système. Cette représentation est donc suffisante pour réaliser les études qualitatives des systèmes dynamiques de composants réparables.

2.3.2 Scénarios caractéristiques du système.

A partir de la représentation de la fonction de structure par automate fini, il est possible de définir trois types de scénarios.

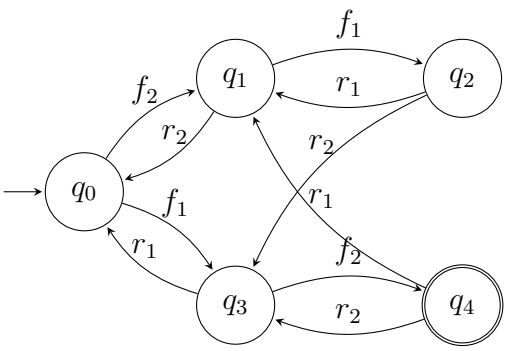


Figure 2.2 – Exemple d'AF représentant une fonction de structure dynamique.

Ces définitions seront illustrées par l'automate de la figure 2.2. Cet automate représente la fonction de structure dynamique d'un système à deux composants réparables 1 et 2. Chaque composant i ne possède qu'un événement de défaillance f_i et un événement de réparation r_i ($\Sigma = \{f_1, f_2, r_1, r_2\}$). Pour que le système soit défaillant, il faut que les deux composants soient défaillants et que la dernière défaillance du composant 1 ait eu lieu avant la dernière défaillance du composant 2.

Tous les scénarios pouvant ou non conduire à la panne du système et satisfaisant le mécanisme décrit ci-dessus sont représentés par les séquences possibles dans l'automate. Ces séquences seront appelées des *séquences dysfonctionnelles*. Ces séquences dysfonctionnelles sont constituées d'événements de défaillance et de réparation de composant, elles sont donc définies sur Σ^*. L'ensemble de toutes ces séquences constitue le *langage dysfonctionnel* $\mathcal{L}_D$ ($\mathcal{L}_D \subseteq \Sigma^*$) qui est le langage généré par l'automate fini. Le langage dysfonctionnel représente donc l'ensemble de tous les scénarios d'événements de défaillance et de réparation possibles dans le modèle. Sur l'exemple de la figure 2.2, les séquences $f_1f_2, f_1f_2r_2f_2$ et f_1 sont des exemples de séquences dysfonctionnelles, $f_1f_1r_1$ n'est pas possible dans le modèle, ce n'est donc pas une séquence dysfonctionnelle.

Chaque scénario qui conduit à la panne du système est représenté par une *séquence défaillante*, caractérisée par le fait que le dernier état atteint par ces séquences à partir de l'état initial est un état marqué. Les autres états traversés par cette séquence peuvent être marqués ou non. L'ensemble de toutes les séquences défaillantes constitue le *langage défaillant* $\mathcal{L}_F$ qui est le langage marqué par l'automate fini. Ce langage représente l'ensemble des scénarios conduisant à la défaillance du système. Sur l'exemple de la figure 2.2, les séquences $f_1f_2, f_1f_2r_2f_2$ et $f_2f_1r_2f_2$ sont des exemples de séquences défaillantes.

Chaque séquence défaillante est également une séquence dysfonctionnelle, ce qui peut s'écrire :

$$\forall \sigma \in \mathcal{L}_F, \sigma \in \mathcal{L}_D$$

ou :

$$\mathcal{L}_F \subseteq \mathcal{L}_D \tag{2.4}$$

Il est également possible de définir l'ensemble des séquences dysfonctionnelles qui ne sont pas des séquences défaillantes par la notation $\mathcal{L}_D \setminus \mathcal{L}_F$.

La connaissance des langages $(\mathcal{L}_D, \mathcal{L}_F)$ permet de définir l'ensemble des valeurs de la fonction de structure. Si une séquence est une séquence défaillante, alors le système est défaillant à l'issue de celle-ci. Si cette séquence est une séquence dysfonctionnelle mais pas une séquence défaillante alors le système est opérationnel à l'issue de celle-ci. Il est donc possible de définir la fonction de structure par :

$$\Phi_d(\sigma) = \begin{cases} 0 & \forall \sigma \in \mathcal{L}_F \\ 1 & \forall \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \end{cases} \tag{2.5}$$

Le langage dysfonctionnel $\mathcal{L}_D$ contient toutes les séquences possibles des événements de défaillance et de réparation du système ($\mathcal{L}_D \subset \Sigma^*, \Sigma = \Sigma_F \cup \Sigma_R, \epsilon \in \mathcal{L}_D$). Si nous faisons l'hypothèse que deux événements ne peuvent occurrer simultanément, toutes les séquences dysfonctionelles sont le résultat de la concaténation d'une séquence dysfonctionelles plus courte et d'un événement de défaillance ou de réparation. Le langage $\mathcal{L}_D$ est alors *préfixe clos* car il vérifie la propriété suivante, qui exprime que tous les préfixes de tous les mots de $\mathcal{L}_D$ sont également des mots de $\mathcal{L}_D$:

$$\mathcal{L}_D = Pref(\mathcal{L}_D) \Leftrightarrow \forall \sigma \in \mathcal{L}_D, \forall (u, e) \in (\Sigma^*)^2 \wedge \sigma = ue, u \in \mathcal{L}_D$$

Puisque $\mathcal{L}_D$ est préfixe clos, alors $\mathcal{L}_D$ est un langage régulier (puisque obtenus récursivement par concaténation des langages singletons). Il est donc possible d'utiliser le théorème de Kleene [Kle56], qui exprime qu'un langage régulier peut être représenté par au moins un automate qui le génère, pour affirmer qu'il existe donc au moins un automate permettant de représenter l'ensemble du langage dysfonctionnel.

Sous l'hypothèse que la défaillance du système est possible dans certaines situations modélisées par les états de l'automate représentant le langage dysfonctionnel, il est possible d'utiliser une nouvelle fois du théorème de Kleene. S'il existe un automate marquant un

langage alors ce langage est régulier. Le langage défaillant $\mathcal{L}_F$ représentant la défaillance du système est donc régulier.

L'étude de la défaillance du système s'intéresse particulièrement aux séquences de coupes. Ces séquences décrivent les scénarios qui, à partir de l'état initial mènent à la première défaillance du système, quelles que soient les évolutions possibles à l'issue de cette défaillance. Ces scénarios sont représentés par des séquences n'évoluant pas au-delà du premier état marqué rencontré. *Une séquence de coupe est donc une séquence défaillante dont seul le dernier état rencontré est un état marqué.* L'ensemble des séquences de coupe constitue le langage $\mathcal{L}_{CS}$. Toutes les séquences de coupe mènent à la défaillance du système, ce sont donc toutes des séquences du langage défaillant. Le langage des séquences de coupe $\mathcal{L}_{CS}$ est donc entièrement contenu dans le langage défaillant $\mathcal{L}_F$:

$$\mathcal{L}_{CS} \subseteq \mathcal{L}_F$$

La relation d'inclusion entre les séquences de coupes dans les séquences défaillantes est illustrée par la figure 2.3. Toute séquence défaillante passant par un état marqué autre que celui atteint en fin de séquence n'est pas une séquence de coupe.

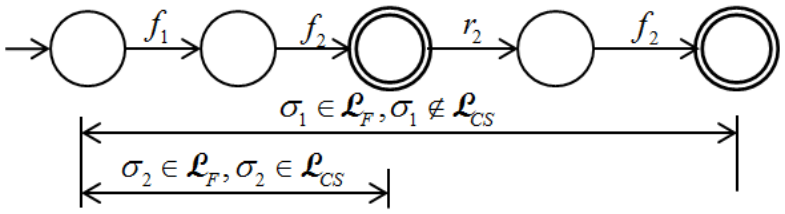


Figure 2.3 – Relation d'inclusion entre séquences de coupes et séquences défaillantes

La définition des séquences de coupe peut être formalisée à l'aide de la notion de préfixe d'une séquence. L'ensemble des préfixes d'une séquence σ_1, noté $Pref(\sigma_1)$ et est constitué de toutes les séquences de taille inférieure ou égale à σ_1 constituées des événements en début de cette séquence. Soit $\sigma_1 = f_1 f_2 r_2 f_2$, l'ensembles des préfixes de σ_1 est alors $Pref(\sigma_1) = \{\epsilon, f_1, f_1 f_2, f_1 f_2 r_2, f_1 f_2 r_2 f_2\}$, où ϵ est la séquence vide de symboles.

L'ensemble des séquence de coupes $\mathcal{L}_{CS}$ peut donc être définie formellement par :

$$\mathcal{L}_{CS} = \{\sigma \in \mathcal{L}_F | \forall \sigma_1 \in Pref(\sigma) \wedge (\sigma_1 \neq \sigma), \delta(q_0, \sigma_1) \notin Q_m\} \tag{2.6}$$

2.3.3 Modèle générique d'un composant réparable.

L'ensemble des composants du système étudié est noté C. A chacun des composants $c_i \in C$ sont associés deux alphabets d'événements : l'alphabet des événements de défaillance $\Sigma_{c_i,F}$ est l'alphabet des événements de réparation $\Sigma_{c_i,R}$. Chacun des événements de défaillance de $\Sigma_{c_i,F}$ génère le passage d'un état où le composant est opérationnel à un état où le composant est en panne. Chacun des événements de l'alphabet des réparations $\Sigma_{c_i,R}$ provoque le passage d'un état où le composant est en panne à un état où le composant est opérationnel. L'alphabet complet Σ_{c_i} de chacun des composants est alors l'union de ces deux alphabets ($\Sigma_{c_i} = \Sigma_{c_i,F} \cup \Sigma_{c_i,R}$).

Dans ce chapitre seuls les composants auxquels est associé un unique événement de défaillance et un unique événement de réparation sont considérés $Card(\Sigma_{c_i,R}) = Card(\Sigma_{c_i,F}) = 1$. Un tel composant possède un état unique dans lequel il est opérationnel et un état unique dans lequel il est défaillant. Afin d'assurer l'indépendance des composants du système, nous faisons l'hypothèse qu'un composant donné ne pourra subir d'évolution sur occurrence d'un événement propre à un autre composant. Pour cela tous les alphabets de composants sont considérés comme étant disjoints deux à deux :

$$\forall (c_i, c_j) \in C^2, c_i \neq c_j, \Sigma_{c_i} \cap \Sigma_{c_j} = \emptyset \tag{2.7}$$

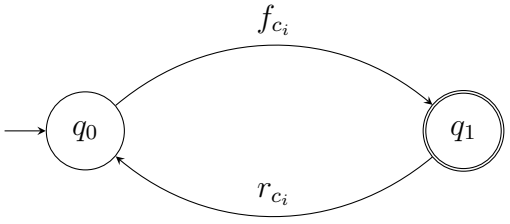


Figure 2.4 – Modèle générique d'un composant réparable d'un système dynamique

Le modèle générique des composants utilisé dans ce chapitre peut alors être représenté par l'automate de la figure 2.4. L'état q_0 est l'état initial où le composant est opérationnel, l'état q_1 l'état où le composant est défaillant. Ce composant possède un événement de

défaillance f_{c_i} et un événement de réparation r_{c_i}. La démarche de recherche des séquences de coupe minimales que nous allons développer permet cependant de considérer des modèles de défaillance/réparation de composant plus sophistiqués (défaillances et réparations sous différents modes de fonctionnement...). Un tel modèle sera proposé dans le chapitre 3.

2.3.4 Relation entre défaillance des composants et scénarios dysfonctionnels.

L'étude de la défaillance d'un système dynamique par l'étude des scénarios d'événements de défaillance et de réparation des composants possibles dans le système nécessite de créer un moyen de retrouver l'ensemble des composants défaillants en fin de séquence afin d'identifier l'ensemble des composants défaillants en fin de séquence. Cette donnée va notamment être utile pour l'utilisation des règles de cohérence.

Le passage d'une séquence dysfontionnelle aux niveaux de défaillance des composants peut se décomposer en deux étapes :

- La détermination de la séquence d'événements influants sur le composant.
- La détermination de l'état de défaillance (ou non) du composant lorsqu'il est soumis à cette séquence.

La réalisation de ces deux étapes dépend du modèle de comportement associé à chacun des composants.

La détermination de l'ensemble des événements d'une séquence donnée influants sur chacun des composants est effectuée en regardant le long de la séquence considérée à quel alphabet de composants chacun des événements de défaillance ou de réparation appartient. L'opérateur de projection d'une séquence sur un alphabet permet de réaliser cette opération pour chacun des composants. Il peut être défini par la proposition récursive suivante :

$$\forall \sigma = \sigma_1\sigma_2...\sigma_{|\sigma|} \in \Sigma^*, Proj(\sigma/\Sigma_{c_i}) : \begin{cases} \epsilon \text{ si } \sigma = \epsilon \\ Proj(\delta/\Sigma_{c_i}) \text{ si } \sigma = \delta\alpha \text{ et } \alpha \notin \Sigma_{c_i} \\ Proj(\delta/\Sigma_{c_i})\alpha \text{ si } \sigma = \delta\alpha \text{ et } \alpha \in \Sigma_{c_i} \end{cases} \tag{2.8}$$

La seconde étape est de relier la projection à l'état de défaillance du composant à l'issue de la séquence grâce au comportement d'un composant décrit en section 2.3.3. L'hypothèse d'indépendance des composants permet d'affirmer que l'état de défaillance atteint par le composant à l'issue de la projection est le même que celui atteint à l'issue de la séquence complète. Ceci car aucun événement hors de cette projection (donc n'appartenant pas à Σ_{c_i}) ne peut faire évoluer le composant. Si le résultat de la projection $Proj(\sigma/\Sigma_{c_i})$ de σ sur l'alphabet Σ_{c_i} du composant c_i est vide ($= \epsilon$) ou si le dernier événement de la projection est un événement de réparation ($\in \Sigma_{c_i,R}$) alors à l'issue de la séquence σ le composant est opérationnel. A l'inverse si le dernier événement de la projection est un événement de défaillance ($\in \Sigma_{c_i,F}$) alors le composant est défaillant à l'issue de la séquence σ. Ce dernier événement de la projection de σ sur Σ_{c_i} est alors noté $Proj(\sigma/\Sigma_{c_i})_{|Proj(\sigma/\Sigma_{c_i})|}$.

Ce passage d'une séquence dysfonctionnelle à l'état de défaillance des composants peut se traduire par une fonction FC (*failled componants*) qui permet de déterminer l'ensemble des composants défaillants à l'issue d'une séquence σ du langage dysfonctionnel :

$$\begin{aligned} FC :& \mathcal{L}_D \rightarrow C \\ \forall \sigma \in \mathcal{L}_D, FC(\sigma) =& \left\{ c_i \in C | Proj(\sigma/\Sigma_{c_i})_{|Proj(\sigma/\Sigma_{c_i})|} \in \Sigma_{c_i,F} \right\} \end{aligned} \tag{2.9}$$

Soit un système dont la défaillance est représentée par l'automate de la figure 2.2, ce système est constitué de deux composants ($C = \{1, 2\}$). Chacun des composants possède un seul événement de défaillance ($\Sigma_{1,F} = \{f_1\}, \Sigma_{2,F} = \{f_2\}$) et un seul événement de réparation ($\Sigma_{1,R} = r_1, \Sigma_{2,R} = r_2$). Soit la séquence $\sigma = f_2 r_2 f_1 f_2 r_1, \sigma \in \mathcal{L}_D$, les projections de cette séquence sur les alphabets des composants sont :

$$\begin{aligned} Proj(\sigma/\Sigma_1) &= f_1 r_1 \\ Proj(\sigma/\Sigma_2) &= f_2 r_2 f_2 \end{aligned}$$

Le dernier événement de la projection $Proj(\sigma/\Sigma_1)$ est un événement de réparation, le composant 1 est donc opérationnel à l'issue de la séquence σ. Le dernier événement de la projection $Proj(\sigma/\Sigma_2)$ est un événement de défaillance, le composant 2 est donc défaillant à l'issue de la séquence σ. L'ensemble des composants défaillants à l'issue de σ est donc :

$$FC(\sigma) = \{2\}, \sigma = f_2 r_2 f_1 f_2 r_1$$

2.4 Analyse des acceptions courantes de séquences de coupes minimales

L'objectif de cette section est d'analyser plusieurs critères courants utilisés pour la définition des séquences de coupes minimales. Les critères de minimalité étudiés sont les suivants :

- C1 : arrêts des séquences au premier état représentant la défaillance du système.
- C2 : non obtention d'une sous-séquences défaillante par retrait uniquement d'événements de défaillance dans une séquence.
- C3 : non obtention d'une sous-séquences défaillante par retrait d'événements de défaillance et de réparation dans une séquence.

Chacun de ces critère permet de restreindre le langage défaillant à des sous-langages permettant de réduire

L'étude de la recherche de l'ensemble des séquences de coupe minimales est réalisée en considérant le système de quatre composants dont la fonction de structure est représentée par l'automate de la figure 2.5 en considérant tour à tour les langages obtenus à partir du langage défaillant par utilisation des différents critères.

Dans la suite, la relation entre une sous-séquence σ' de σ est notée $\sigma \supset \sigma'$ ou $\sigma' \subset \sigma$.

2.4.1 Critère d'arrêts à la première défaillance

Le critère C1 limite l'explorations des séquences défaillante au premier état de défaillance rencontré. Il permet de sélectionner à partir de l'ensemble du langage défaillant $\mathcal{L}_F$ l'ensemble des séquence de coupes $\mathcal{L}_{CS}$ tel que défini dans la section **??**.

Le langage obtenus par utilisation du critère 1 est donc exactement $\mathcal{L}_{C1} = \mathcal{L}_{CS}$, $\mathcal{L}_{C1}$ est également un sous-langage du langage défaillant.

$$\mathcal{L}_F \supseteq \mathcal{L}_{C1} \tag{2.10}$$

2.4.2 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance

Le critère C2 reflète sur les système dynamiques le mécanisme permettant la construction des coupes à partir des coupes minimales. Alors que n'importe quelle coupe (par exemple $A.B.C$) du système peut être obtenue par ajout de défaillances de composants (A et C) à une coupe minimale (C), ce critère considère que n'importe quelle séquence défaillante peut être obtenue par ajout d'événements de défaillance dans une séquence de coupe minimale.

Le critère de sélection permettant de déterminer si une séquence défaillante est une séquence minimale peut alors être formalisé par une expression de la restriction du langage défaillant à un sous ensemble. Cet ensemble est alors décrit par la proposition 2.11.

$$\mathcal{L}_{C2} = \{\sigma \in \mathcal{L}_F | \nexists \sigma' \in \mathcal{L}_F, \sigma' \subset \sigma, \{\sigma - \sigma'\} \cap \Sigma_F^* = \emptyset\} \tag{2.11}$$

Cette proposition décrit que s'il est possible de trouver une sous-séquence défaillante σ' à la séquence considérée σ en ne retirant que des événements de défaillance (représenté par la proposition $\{\sigma - \sigma'\}$, alors σ n'est pas minimale car il est possible de l'obtenir par ajout d'événements de défaillance dans σ'.

Comme il s'agit d'une restriction du langage défaillant, $\mathcal{L}_{C2}$ est une sous langage de $\mathcal{L}_F$:

$$\mathcal{L}_F \supseteq \mathcal{L}_{C2} \tag{2.12}$$

Il est également possible d'associer le critère C2 au critère C1 afin d'obtenir une restriction des séquences de coupe permettant d'acceder à une première définition de l'ensemble des séquences de coupe minimale $SCM1$:

$$\mathcal{L}_F \supseteq \mathcal{L}_{CS} \supseteq SCM1 \text{ Avec } \mathcal{L}_{CS} = \mathcal{L}_{C1}, SCM1 = \mathcal{L}_{C1+C2} \tag{2.13}$$

2.4.3 Critère d'obtention de séquence défaillante par retrait d'événements de défaillance ou de réparation

Le critère C3 est une généralisation du critère C2 apparus lors du passage au systèmes réparables. Il exprime que n'importe quelle séquence du langage défaillant peut

être reconstruite à partir d'une séquence de coupe minimale par ajout soit d'événements de défaillance ou de réparation. Une séquence minimale σ est alors définie comme une séquence défaillante où il n'est pas possible d'obtenir une séquence défaillante σ par retrait d'événement de défaillance ou de réparation.

L'ensemble des séquences minimales suivant le critère C3 est donc un sous ensemble du langage défaillant pouvant être exprimé par :

$$\mathcal{L}_{C3} = \{\sigma \in \mathcal{L}_F | \nexists \sigma' \in \mathcal{L}_F, \sigma' \subset \sigma, \{\sigma - \sigma'\} \in \Sigma^*\}\mathcal{L}_F \supseteq \mathcal{L}_{C3} \tag{2.14}$$

Le critère C3 est plus sélectif que le critère C2 car un plus grand nombre de séquences sont considérée comme non minimales. Soit le cas ou la séquence $\sigma = f_1 f_2 r_1 f_3$ inclue la séquence défaillante $\sigma' = f_2 f_3$. Afin d'obtenir la séquence σ, il est necessaire j'ajouter une événement de défaillance (f_1) et de réparation (r_1) dans σ'. L'utilisation du critère C2 ne peut permettre la reconstruction de σ à partir de σ', σ est alors considéré comme minimale, ce qui n'est pas le cas si on utilise le critère C3. Le langage $\mathcal{L}_{C3}$ est donc un sous langage de $\mathcal{L}_{C2}$ car il décrit une condition plus restrictive. La relation entre les différent langage est donc la suivante :

$$\mathcal{L}_F \supseteq \mathcal{L}_{C2} \supseteq \mathcal{L}_{C3} \tag{2.15}$$

Il est également possible d'associer le critère C3 au critère C1 afin d'obtenir une restriction des séquences de coupe permettant d'acceder à une seconde définition de l'ensemble des séquences de coupe minimale $SCM2$. Le langage $\mathcal{L}_{C3}$ est un sous langage de $\mathcal{L}_{C2}$, la définition $SCM2$

$$\mathcal{L}_F \supseteq \mathcal{L}_{CS} \supseteq SCM1 \supseteq SCM2 \text{ Avec } \mathcal{L}_{CS} = \mathcal{L}_{C1}, SCM1 = \mathcal{L}_{C1+C2}, SCM2 = \mathcal{L}_{C1+C3} \tag{2.16}$$

2.4.4 Analyse des séquences d'un exemple

Considérons maintenant le système de trois composants dont la fonction de structure est représenté par l'automate de la figure 2.2.

Considérons les séquences $f_2 f_3 r_2 f_1$ et $f_1 f_2 r_1 f_3$. La séquence $f_2 f_3 r_2 f_1$ est attendue comme étant minimale car les événements f_2 et r_2 permettent d'accéder à la défaillance d'abord du composant 3 puis du composant 1. Par contre la séquence $f_1 f_2 r_1 f_3$ n'est pas attendue comme étant minimale car l'ajout des événements f_1 et r_1 à la séquence $f_2 f_3$

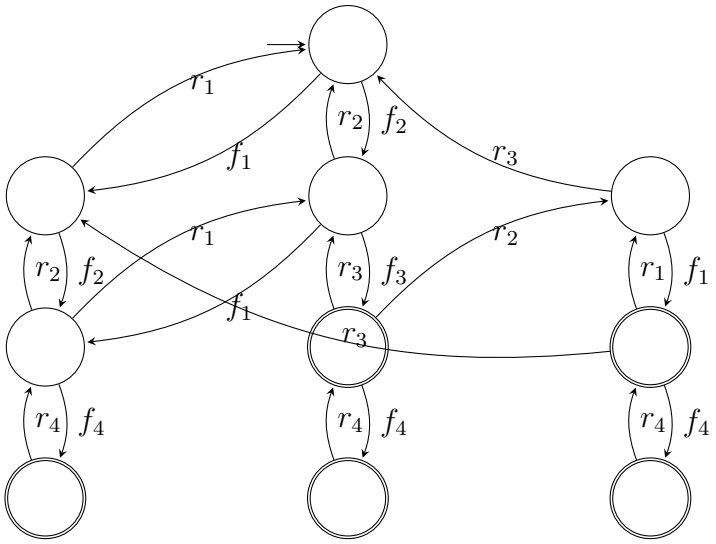


Figure 2.5 – Exemple d'AF représentant une fonction de structure dynamique.

n'apporte aucune information supplémentaire. Analysons maintenant ces deux séquences vis-à-vis des acceptions C2 et C3 de minimalité des séquences.

Soit la séquence défaillante $f_2 f_3 r_2 f_1$ permettant d'obtenir la défaillance du système avec les composants 1 et 3 défaillants en fin de séquence. Suivant le critère C2, la séquence $f_2 f_3 r_2 f_1$ est considérée comme minimale, car il n'est pas possible d'obtenir un sous-mot marqué par l'automate en ne retirant que des événements de défaillance. Suivant le critère C3, cette séquence n'est pas considérée comme minimale, car le sous mot $f_2 f_3$ est une séquence marquée par l'automate.

Soit la séquence défaillante $f_1 f_2 r_1 f_3$ permettant d'obtenir la défaillance du système avec les composants 2 et 3 défaillants en fin de séquence. Par le critère C2, cette séquence est considérée comme inimale car il n'est pas possible d'obtenir un sous mot marqué par l'automate en ne retirant que des événement de défaillance. Pourtant, si on retire les événements f_1 et r_1 de la séquence considéré il est possible d'obtenir une séquence défaillante $f_2 f_3$ marquée par l'automate, sous mot de la séquence $f_1 f_2 r_1 f_3$ et attendue comme étant minimale, ce qui est validé par le critère C2 de minimalité qui identifie la séquence$f_1 f_2 r_1 f_3$ comme non-minimale

Les résultats de l'analyse des séquence $f_2 f_3 r_2 f_1$ et $f_1 f_2 r_1 f_3$ par les critères courants de minimalité sont présenté en tableau 2.1. Il est possible de remarquer qu'aucune des

	$f_2 f_3 r_2 f_1$	$f_1 f_2 r_1 f_3$
Attendu	Minimale	Non-Minimale
Critère C1	Non-Minimale	Non-Minimale
Critère C2	Minimale	Minimale

Tableau 2.1 – Résultats de l'analyse des séquences $f_2 f_3 r_2 f_1$ et $f_1 f_2 r_1 f_3$ par les acceptions courantes de la définition du critère de minimalité

acception considérées ne permet d'atteindre le résultat attendu. Cette difficulté provient du fait que, pour certaine séquences, le critère de minimalité doit pouvoir retirer autant les événements de défaillance que de réparation. Ainsi l'acception A1 est beaucoup trop sélective car elle retire l'ensemble des événements de défaillance et de réparation afin de tester la minimalité alors que l'acception A2 n'est pas suffisamment sélective car elle ne teste que les sous-mots obtenus par retrait uniquement des événements de défaillance.

Une nouvelle définition est donc nécessaire. Cette définition doit permettre d'identifier dans quels cas un événement de réparation peut être retiré afin d'identifier un sous-mot de la séquence considéré, marqué par l'automate et permettant ainsi de conclure sur la non-minimalité de cette séquence.

2.5 Cohérence d'un système dynamique réparable.

Le passage du domaine des systèmes statiques aux systèmes dynamiques réparables nécessite une redéfinition de la notion de cohérence des systèmes.

La fonction de structure du système ne dépend plus seulement de l'état de défaillance des composants, mais des scénarios composés d'événements de défaillance et de réparation des composants (cf. 2.3.1). *La cohérence d'un système dynamique doit donc être définie comme étant une relation entre ces scénarios (représentés par des séquences d'événements), leur longueur, l'ensemble final des composants défaillants et l'état résultant du système.*

La section 2.5.1 se propose de définir la propriété de semi-cohérence pour les systèmes dynamiques réparables, la section 2.5.2 définira les conditions de cohérence des systèmes

semi-cohérents.

2.5.1 Semi-cohérence d'un système dynamique réparable.

L'objectif de cette section est de donner les critères selon lesquels un système dynamique réparable sera considéré comme étant semi-cohérent.

La semi-cohérence des systèmes statiques est une relation entre les variations de l'ensemble des composants défaillants et la défaillance résultante du système. Pour les systèmes dynamiques, la fonction de structure est définie sur les scénarios d'événements de défaillance et de réparation des composants. Ces scénarios sont représentés par des séquences dysfonctionnelles constituant le langage dysfonctionnel $\mathcal{L}_D$.

La semi-cohérence d'un système dynamique réparable doit donc permettre la définition de la relation entre l'accroissement d'une séquence dysfonctionnelle et la défaillance du système.

Soit une séquence σ, possible dans le système ($\sigma \in \mathcal{L}_D$), notre objectif est de décrire dans quelles conditions une séquence σ', de longueur supérieure à la longueur de la séquence σ, obtenue par ajout d'événements de l'alphabet Σ au cours de la séquence, permet d'obtenir soit la conservation de la défaillance du système, soit la conservation du bon fonctionnement du système. L'accroissement de la séquence σ peut entraîner différents effets sur l'ensemble des composants défaillants :

- l'accroissement de l'ensemble des composants défaillants,
- la diminution de l'ensemble des composants défaillants,
- la conservation de l'ensemble des composants défaillants.

La variation de l'ensemble des composants défaillants est obtenu dès l'ajout d'un unique événement de défaillance ou de réparation. Nous nous intéresserons donc dans un premier temps à l'ajout d'un unique événement de défaillance ou de réparation au cours d'une séquence défaillante ($\sigma \in \mathcal{L}_F$) ou non-défaillante ($\sigma \in \mathcal{L}_D \setminus \mathcal{L}_F$). L'ajout de tout événement de défaillance ou de réparation de l'alphabet Σ génère une variation de l'ensemble des composants défaillants. La conservation de cet ensemble ne peut donc pas

être étudiée qu'en cherchant sous quelles conditions la distribution d'une autre séquence dans la séquence σ permet d'obtenir, d'une part, la conservation de l'ensemble des composants défaillant et d'autre part la conservation de l'opérabilité du système ou de sa défaillance.

Afin d'illustrer les différents cas étudiés, un système à trois composants est considéré. L'automate représentant sa fonction de structure est présenté en figure 2.6. Pour que ce système soit défaillant, il faut que le composant 3 soit défaillant ou que la dernière défaillance du composant 1 ait eu lieu avant la dernière défaillance du composant 2 et que les composants 1 et 2 soient défaillants. Pour les besoins de l'étude, l'hypothèse de la cohérence de ce système est posée.

L'étude de l'ajout d'un événement supplémentaire modifiant l'ensemble des composants défaillants en fin de séquence doit se porter sur deux types de séquences, les séquences non défaillantes ($\sigma \in \mathcal{L}_D \setminus \mathcal{L}_F$) et les séquences défaillantes ($\sigma \in \mathcal{L}_F$).

2.5.1.1 Ajout d'un événement dans une séquence défaillante.

Soit une séquence défaillante $\sigma, \sigma \in \mathcal{L}_F$, à l'issue de celle-ci le système est donc défaillant. Afin d'augmenter la longueur de cette séquence, un événement u va être ajouté au cours de celle-ci. Cet événement peut être soit un événement de réparation ($\exists c_i \in C, u \in \Sigma_{c_i,R}$) soit un événement de défaillance ($\exists c_i \in C, u \in \Sigma_{c_i,F}$) relatif à un composant du système.

Soit u un événement de défaillance, il est possible d'insérer u dans la séquence σ (qui peut s'écrire $\sigma = \sigma_1\sigma_2$, σ_1 et σ_2 peuvent être vides de symboles) afin d'obtenir une séquence plus longue $\sigma' = \sigma_1 u \sigma_2$ modélisant un scénario possible ($\sigma' \in \mathcal{L}_D$). Si σ' est possible, c'est qu'à l'issue de σ_1 le composant associé à u était non défaillant et donc que dans σ_2 ce même composant n'est pas réparé. A l'issue de σ', le composant dont u est un événement de défaillance est alors défaillant. L'ensemble des composants défaillants $FC(\sigma')$ contient alors l'ensemble des composants défaillant de σ ainsi que le composant c_i correspondant à l'événement de défaillance ajouté. Un sur-ensemble de l'ensemble des composants défaillants de la séquence initiale est obtenu ($FC(\sigma') \supset FC(\sigma)$). *Sous l'hypothèse qu'un*

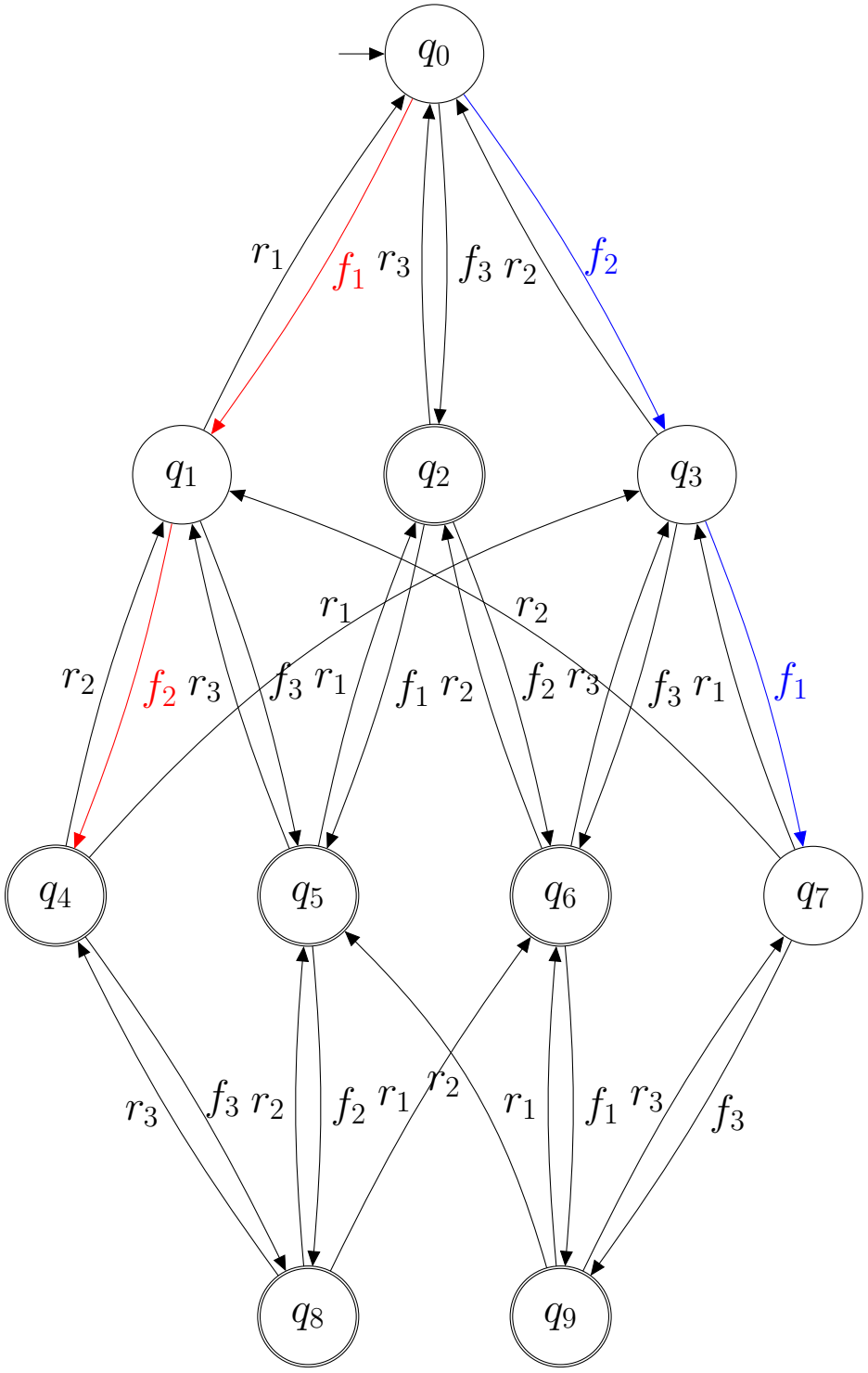


Figure 2.6 – Fonction de structure d'un système dynamique à 3 composants réparables sous forme d'automate.

événement de défaillance ne peut masquer la défaillance du système et de manière homogène à la définition de la cohérence des systèmes binaires statiques, le système doit être défaillant à l'issue de la séquence σ'. Ceci est alors exprimé indépendamment de la séquence défaillante σ et de l'événement de défaillance considérés par la proposition :

$$\forall \sigma \in \mathcal{L}_F, \forall (\sigma_1, \sigma_2) \in (\Sigma^*)^2 | \sigma = \sigma_1\sigma_2, \forall u \in \Sigma_F : \left.\begin{array}{l} \sigma' = \sigma_1 u \sigma_2 \\ \sigma' \in \mathcal{L}_D \\ FC(\sigma') \supset FC(\sigma) \end{array}\right\} \Longrightarrow \sigma' \in \mathcal{L}_F \tag{2.17}$$

L'illustration de cette propriété peut être faite sur la séquence $\sigma = f_1 f_2$ (tracée en rouge sur la figure 2.6) avec $u = f_3, u \in \Sigma_{3,F}$ un événement de défaillance. A l'issue de la séquence σ, seuls les composants 1 et 2 sont défaillants ($FC(\sigma) = \{1, 2\}$), le système est également défaillant. Les séquences qu'il est possible de construire à partir de σ en ajoutant f_3 sont $f_1 f_2 f_3, f_1 f_3 f_2$ et $f_3 f_1 f_2$. Chacun des scénarios décrits par ces séquences est possible dans le système, les séquences correspondantes appartiennent donc toutes au langage dysfonctionnel $\mathcal{L}_D$ et sont illustrées en figure 2.7.

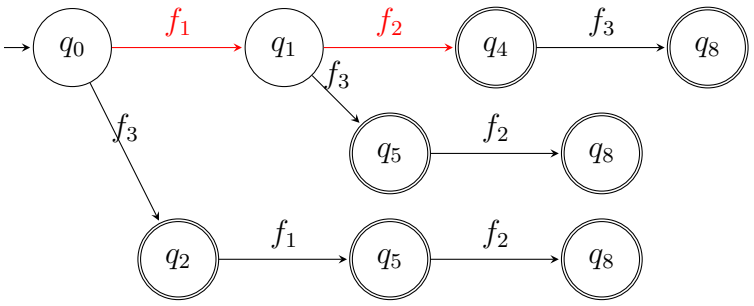


Figure 2.7 – Exemple d'automate fini représentant une fonction de structure dynamique.

L'ajout de l'événement f_3 permet de passer d'un ensemble de composants défaillants $FC(\sigma) = \{1, 2\}$ à un sur-ensemble $FC(\sigma') = \{1, 2, 3\}$. Conformément à la proposition 2.17 le système est défaillant à l'issue de toutes ces séquences, tous les états atteints par les séquences dont la longueur a été augmentée par ajout de f_3 sont des états marqués.

Considérons cette fois u, un événement de réparation ($\exists c_i \in C, u \in \Sigma_{c_i,R}$), l'ajout de cet événement permet également d'augmenter la taille de la séquence afin d'obtenir des séquences $\sigma' = \sigma_1 u \sigma_2$ mais diminue l'ensemble des composants défaillants ($FC(\sigma') \subset$

$FC(\sigma)$). Cette diminution ne permet pas de conclure sur la défaillance du système à l'issue des nouvelles séquences σ'. Cette incapacité à déterminer le niveau de défaillance du système à l'issue de ces séquences σ' est illustrée sur la figure 2.8.

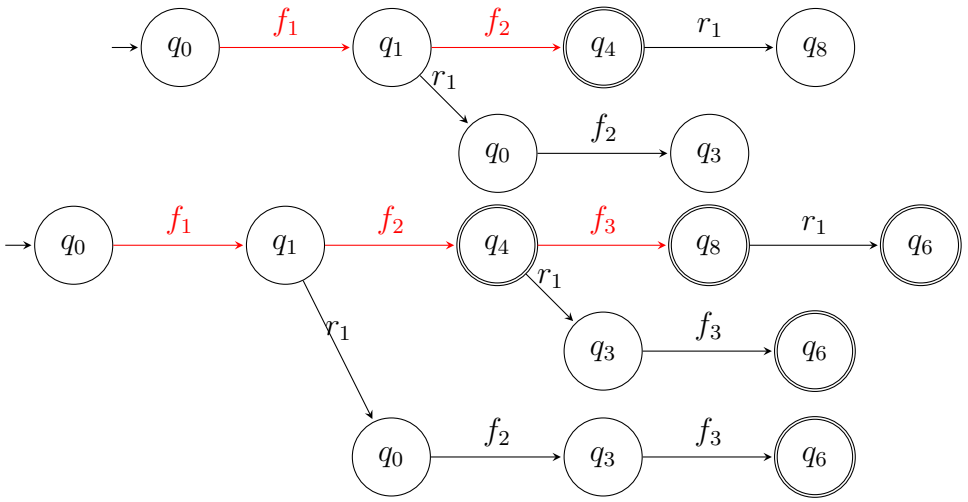


Figure 2.8 – Ajout d'un événemement de réparation dans des séquences défaillantes.

Cette figure représente les nouvelles séquences σ' qu'il est possible de construire en prenant $\sigma = f_1 f_2$ ou $\sigma = f_1 f_2 f_3$ avec $u = r_1$. Dans le cas ou $\sigma = f_1 f_2$, l'ajout de r_1 dans la séquence modifie le niveau de défaillance du système en fin de séquence, le système est opérationnel à l'issue de tous les σ'. La séquence $r_1 f_1 f_2$ n'est pas représentée car elle n'appartient pas au langage dysfonctionnel : elle n'est pas possible dans le modèle si à l'état initial le composant 1 n'est pas défaillant. Dans le second cas ($\sigma = f_1 f_2 f_3$), l'ajout de l'événement de réparation ne modifie pas l'état de défaillance du système en fin de séquence : le système conserve sa défaillance. L'effet de l'ajout d'un événement de réparation au cours d'une séquence initialement défaillante va donc dépendre de la séquence défaillante et de l'événement de réparation considéré, il est impossible, dans le cas général, de conclure indépendamment de ces deux éléments.

2.5.1.2 Ajout d'un événement dans une séquence non-défaillante.

Soit une séquence dysfonctionnelle non-défaillante $\sigma, \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F$, à l'issue de celle-ci le système est donc opérationnel. Afin d'augmenter la longueur de cette séquence, on se propose maintenant d'y ajouter un événement u. Cet événement peut être soit un événement de réparation d'un composant ($\exists c_i \in C, u \in \Sigma_{c_i,R}$), soit un événement de défaillance

$(\exists c_i \in C, u \in \Sigma_{c_i,F})$.

Soit u un événement de réparation, il est possible d'insérer u dans une séquence $\sigma = \sigma_1\sigma_2$ afin d'obtenir une séquence plus longue $\sigma' = \sigma_1 u \sigma_2$ modélisant un scénario possible ($\sigma' \in \mathcal{L}_D$). A l'issue de la séquence σ', le composant dont u est un événement de réparation est alors opérationnel. Le nouvel ensemble des composants défaillants $FC(\sigma')$ contient tous les composants défaillants de la séquence σ à l'exception de celui auquel appartient l'événement de réparation. Un sous-ensemble de l'ensemble des composants défaillants de la séquence initiale est alors obtenu ($FC(\sigma') \subset FC(\sigma)$). Afin de rester homogène à la définition de la semi-cohérence des systèmes statiques et sous l'hypotèse qu'*un événement de réparation ne peut contribuer à la défaillance du système*, l'ajout de cet événement de réparation diminuant l'ensemble des composants défaillants doit permettre au système de rester opérationnel en fin de séquence. Cette propriété est alors exprimée indépendamment de la séquence non-défaillante σ et de l'événement u de réparation considérés par la proposition :

$$\forall \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F, \forall (\sigma_1, \sigma_2) \in (\Sigma^*)^2 | \sigma = \sigma_1\sigma_2, \forall u \in \Sigma_R : \left.\begin{array}{l} \sigma' = \sigma_1 u \sigma_2 \\ \sigma' \in \mathcal{L}_D \\ FC(\sigma') \subset FC(\sigma) \end{array}\right\} \Longrightarrow \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \tag{2.18}$$

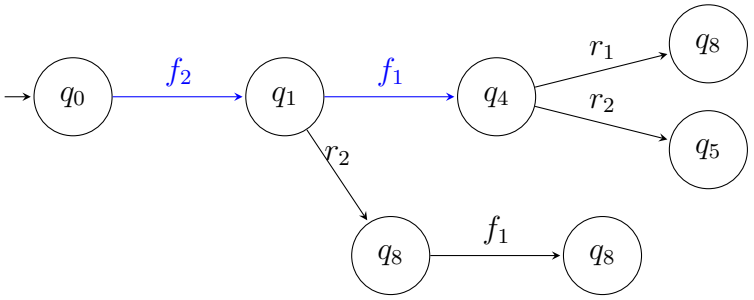


Figure 2.9 – Ajout d'événement de réparation dans une séquence non-défaillante.

Cette propriété est illustrée sur la séquence $\sigma = f_2 f_1$ (tracée en bleu sur la figure 2.6) avec $u \in \{r_1, r_2\}$ un événement de réparation. A l'issue de la séquence σ, seuls les composants 1 et 2 sont défaillants ($FC(\sigma) = \{1, 2\}$), le système est opérationnel. Les séquences du langage dysfonctionnel $\mathcal{L}_D$ qu'il est possible de construire à partir de σ en ajoutant r_2 sont $f_2 f_1 r_2$ et $f_2 r_2 f_1$. L'utilisation de r_1 permet de construire la séquence $f_2 f_1 r_1$ du

langage dysfonctionnel. Ces séquences sont illustrées dans la figure 2.9. L'ajout de l'événement r_1 a permis de passer d'un ensemble de composants défaillants $FC(\sigma) = \{1, 2\}$ à un sous-ensemble $FC(\sigma') = \{2\}$. Conformément à la proposition 2.18 l'état atteint par la séquence allongée par r_1 n'est pas un état marqué : le système est opérationnel à l'issue de σ'. L'ajout de l'événement r_2 à permis de passer d'un ensemble de composants défaillants $FC(\sigma) = \{1, 2\}$ à un sous-ensemble $FC(\sigma') = \{1\}$. Conformément à la proposition 2.18 les états atteints par les séquences allongées par r_2 ne sont pas des états marqués : le système est opérationnel à l'issue de σ'.

Soit u un événement de défaillance ($\exists c_i \in C, u \in \Sigma_{c_i,F}$), l'ajout de cet événement permet également d'augmenter la taille de la séquence afin d'obtenir des séquences $\sigma' = \sigma_1 u \sigma_2$, mais augmente la taille de l'ensemble des composants défaillants ($FC(\sigma') \supset FC(\sigma)$). Cette augmentation ne permet pas de conclure quand à la défaillance du système à l'issue des nouvelles séquences σ'. Cette incapacité à déterminer l'état de défaillance du système à l'issue des séquences allongées σ' est illustrée en figure 2.10.

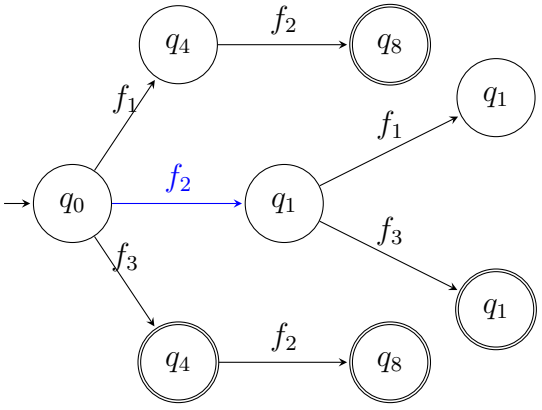


Figure 2.10 – Ajout d'événements de défaillance dans une séquence non-défaillante.

Dans cette figure la séquence non-défaillante considérée est la séquence $\sigma = f_2$, les événements de défaillance utilisés sont f_1 et f_3. Alors que l'ajout de f_3 dans σ conduit dans tous les cas à une défaillance du système l'ajout de f_1 ne permet l'obtention d'un système défaillant que lorsque cet ajout a lieu avant f_2. La position de cet ajout ayant une influence sur l'état de défaillance final du système, aucune conclusion générale n'est possible sur l'effet de l'ajout d'un événement de défaillance au sein d'une séquence non-défaillante indépendamment de la séquence et de l'événement de défaillance.

2.5.1.3 Ajout d'une séquence dans une séquence dysfonctionnelle

L'objectif de cette section est d'augmenter la taille d'une séquence dysfonctionnelle σ soit défaillante ($\sigma \in \mathcal{L}_F$) soit non-défaillante ($\sigma \in \mathcal{L}_D \setminus \mathcal{L}_F$) sans changer ni l'ensemble des composants défaillants ni le niveau de défaillance du système. Cette augmentation va passer par la distribution non plus d'un événement, mais des événements d'une séquence δ d'événements de l'alphabet ($\delta \in \Sigma^*$) dans σ pour construire des séquences dysfonctionnelles σ' en préservant les ordres d'occurrences des événements de ces séquences. Les deux séquences δ et σ sont donc incluses dans σ', ces inclusions sont dénotées respectivement $\sigma' \supseteq \delta$ et $\sigma' \supseteq \sigma$. Les séquences σ' doivent également représenter des scénarios possibles pour le système ($\sigma' \in \mathcal{L}_D$). Les séquences σ' sont alors telles ques :

$$\sigma \in \mathcal{L}_D, \delta \in \Sigma^*, \begin{cases} \sigma' \in \mathcal{L}_D \\ |\sigma'| = |\sigma| + |\delta| \\ \sigma' \supseteq \sigma \\ \sigma' \supseteq \delta \end{cases} \tag{2.19}$$

La construction des séquences σ' doit permettre la conservation de l'ensemble des composants défaillants $FC(\sigma)$ ainsi que de l'état de défaillance du système (opérationnel ou défaillant). Cette conservation est exprimée par la proposition suivante :

$$(\sigma, \sigma') \in (\mathcal{L}_D)^2, \begin{cases} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \implies \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \implies \sigma' \in \mathcal{L}_F \end{cases} \tag{2.20}$$

L'objectif est alors d'exprimer sous quelles conditions les séquences δ permettent la conservation exprimée par la proposition 2.5.1.3 ainsi que d'expliciter la construction des séquences σ' décrites par la proposition 2.19 à partir d'une séquence dysfonctionnelle σ. Cette étude va se dérouler en deux temps : l'identification de la sous-séquence de σ où l'ajout de δ peut être réalisé sans influence sur la séquence initiale, puis la distribution des événements de δ dans σ.

Soient une séquence dysfonctionnelle σ du langage dysfonctionnel ($\sigma \in \mathcal{L}_D$) et une séquence δ d'événements de l'alphabet Σ ($\delta \in \Sigma^*$). La séquence σ représente un scénario possible d'événements de défaillance et de réparation de composants réparables d'un système dynamique. L'ajout des événements de la séquence δ peut alors avoir un effet sur l'état de défaillance du système en fonction des positions dans lesquelles ils sont ajoutés

à travers la séquence σ. Il s'agit maintenant d'identifier dans quelle partie de σ il est possible d'ajouter les événements composant δ en ayant conservation de la défaillance. σ est alors décomposée en un préfixe σ_1, un suffixe σ_3 et une sous-séquence σ_2 ($\sigma = \sigma_1\sigma_2\sigma_3$). La sous-séquence σ_2 est la sous-séquence de σ où les événements de la séquence δ peuvent être insérés sans influence sur le niveau de défaillance du système ou l'ensemble des composants défaillants en fin de séquence σ, σ_2 est alors appelée zone de non-influence de δ sur σ. Cette non-influence peut être exprimée par : *une séquence est non-influente sur une autre si son insertion un nombre illimité de fois n'importe où dans la séquence n'a d'effet ni sur la défaillance du système ni sur l'ensemble des composants défaillants.* Pour exprimer cette propriété, la sous-séquence σ_2 est séparée en en deux parties σ_{21} et σ_{22} ($\sigma_2 = \sigma_{21}\sigma_{22}$) afin de pouvoir insérer un nombre illimité de fois δ. Cette insertion, si elle est possible, doit toujours conserver la défaillance et l'ensemble des composants défaillants quelque soit la position d'insertion dans σ_2. Cette propriété peut alors être mise sous la forme :

$$
\begin{aligned}
&\forall \sigma \in \mathcal{L}_D, \forall \delta \in \Sigma^*(\sigma_1, \sigma_2, \sigma_3) \in (\Sigma^*)^3 | \sigma = \sigma_1\sigma_2\sigma_3 \\
&\forall (\sigma_{21}, \sigma_{22}) \in (\Sigma^*)^2 | \sigma_2 = \sigma_{21}\sigma_{22}, \\
&\forall \sigma' \in \{\sigma_1\sigma_{21}\delta^*\sigma_{22}\sigma_3\}, \sigma' \in \mathcal{L}_D \implies \begin{cases} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \implies \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \implies \sigma' \in \mathcal{L}_F \end{cases}
\end{aligned} \tag{2.21}
$$

La première ligne de cette formalisation exprime une décomposition de σ où σ_2 est une zone de non-influence de δ sur σ. La seconde exprime la décomposition de cette zone de non-influence permettant d'insérer un nombre illimité de fois δ (exprimé par δ^*). La dernière ligne exprime que toutes les séquences construites doivent, si elles existent ($\in \mathcal{L}_D$), conserver l'état de défaillance du système et l'ensemble des composants défaillants en fin de séquence.

Une fois l'existence d'une zone de non-influence σ_2 de δ sur σ identifiée, toutes les séquences construites par distribution ordonnée des événements de δ dans σ_2 permet d'avoir conservation au terme de la proposition . Les séquences ainsi construites sont

alors exprimées par la proposition suivante :

$$\sigma = \sigma_1\sigma_2\sigma_3 \in \mathcal{L}_D, \delta \in \Sigma^*,$$
$$\forall \sigma' \in \mathcal{L}_D, \begin{cases} \sigma' = \sigma_1\sigma'_2\sigma_3 \\ |\sigma'_2| = |\sigma_2| + |\delta| \\ \sigma'_2 \supseteq \sigma_2 \\ \sigma'_2 \supseteq \delta \end{cases} \implies \begin{cases} FC(\sigma') = FC(\sigma) \\ \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \implies \sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F \\ \sigma \in \mathcal{L}_F \implies \sigma' \in \mathcal{L}_F \end{cases} \tag{2.22}$$

L'illustration de ces différentes propriétés ainsi que la construction des nouvelles séquences est réalisée en considérant une séquence défaillante $\sigma = f_1 f_2$ de la figure 2.6. L'objectif est d'ajouter les séquences $\delta = f_3 r_3$ et $\delta = r_1 f_1$, en passant par l'identification de la zone non-influente associée à chacun des δ dans σ.

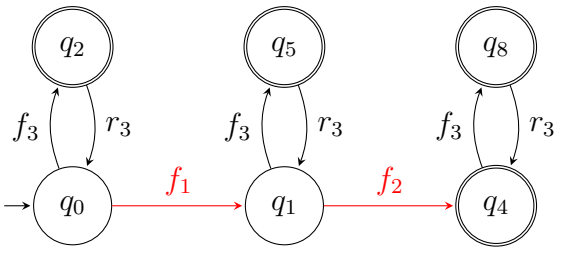


Figure 2.11 – Ajout des séquences $f_3 r_3$ à travers la séquence $f_1 f_2$.

Dans le cas où $\sigma = f_1 f_2$ et $\delta = f_3 r_3$, tous les ajouts possibles ($\in \mathcal{L}_D$) de la séquence δ dans la séquence σ permet d'arriver en fin de séquence dans l'état marqué q_4 (cf. figure 2.11). Comme l'ajout un nombre illimité de fois de la séquence δ dans σ permet la conservation de la défaillance du système au sens de la proposition 2.21, la zone non-influente de δ sur σ est la séquence σ au complet. La décomposition de $\sigma = \sigma_1\sigma_2\sigma_3$ est alors $\sigma_1 = \epsilon, \sigma_2 = \sigma, \sigma_3 = \epsilon$. Toutes les compositions de σ avec δ sur $\sigma_2 = \sigma$ sont toutes des séquences défaillantes conformément à la proposition 2.22. Ces séquences ainsi construites sont représentées en figure 2.12.

Dans le cas où $\sigma = f_1 f_2$ et $\delta = r_1 f_1$, l'ajout de la séquence δ dans la séquence σ n'est possible qu'après occurrence de f_1 (seul cas où $\sigma' \in \mathcal{L}_D$). Cet ajout permet d'arriver en fin de séquence dans l'état marqué q_4 si et seulement si cet ajout est effectué avant occurrence de f_2. Si δ est ajouté après occurrence de f_2, la séquence obtenue atteint l'état q_7 non marqué (cf. figure 2.13).

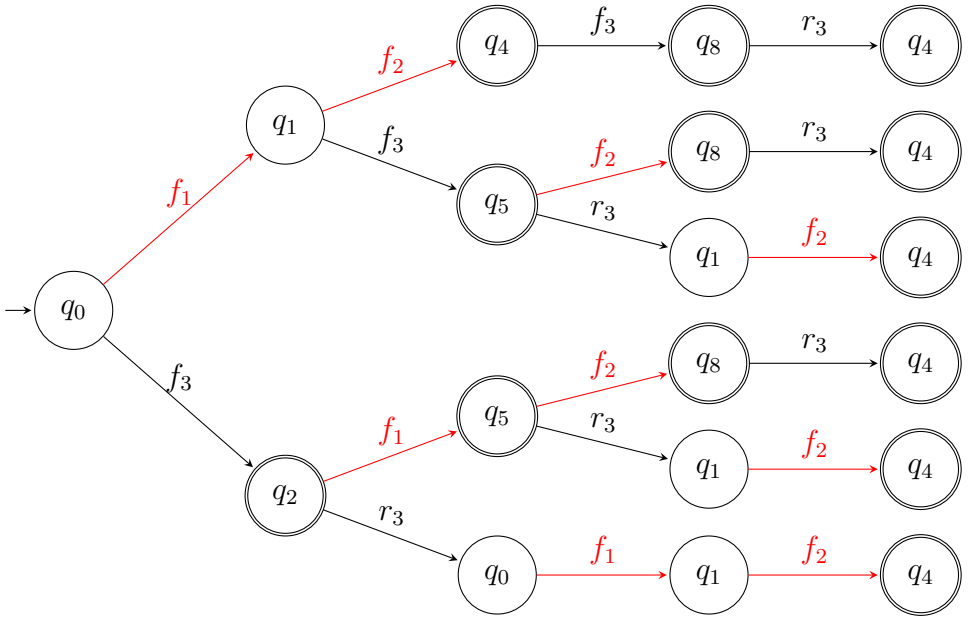


Figure 2.12 – Séquences générées par ajout de $\delta = f_3 r_3$ dans $\sigma = f_1 f_2$.

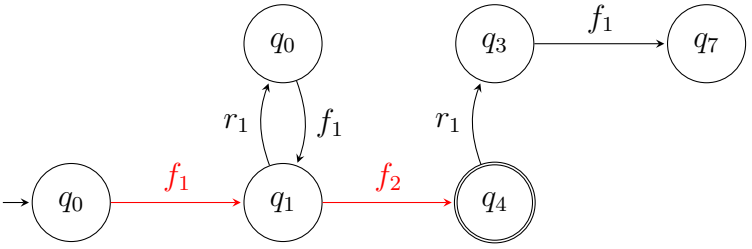


Figure 2.13 – Ajouts possibles des séquences $r_1 f_1$ dans la séquence $f_1 f_2$.

L'ajout d'un nombre illimité de fois permet la conservation de la défaillance du système au sens de la proposition 2.21 uniquement si cet ajout est effectué avant f_2. La zone de non-influence de δ sur σ est donc $\sigma_2 = f_1$. La décomposition de σ permettant la construction des nouvelles séquences est alors $\sigma = \sigma_1\sigma_2\sigma_3$ avec $\sigma_1 = \epsilon, \sigma_2 = f_1, \sigma_3 = f_2$.La seule séquence défaillante qu'il est possible de construire conformément à la proposition 2.22 est alors $\sigma' = f_1 r_1 f_1 f_2$

2.5.1.4 Bilan

La semi-cohérence d'un système dynamique réparable consiste à respecter une relation liant l'augmentation de la longueur des séquences, la variation de l'ensemble des composants défaillants et l'état de défaillance du système en fin de séquence. Les résultats des sections précédentes sont résumés dans le tableau 2.2

Si $\sigma \in \mathcal{L}_F$	Ajout d'un événement $\lvert\sigma'\rvert = \lvert\sigma\rvert + 1$	Distribution d'une séquence $\lvert\sigma'\rvert > \lvert\sigma\rvert + 1$
Augmentation de FC $FC(\sigma') \supset FC(\sigma)$	Sous vérification de la proposition 2.17 $\sigma' \in \mathcal{L}_F$	*
Diminution de FC $FC(\sigma') \subset FC(\sigma)$	NC	NC
Conservation de FC $FC(\sigma') = FC(\sigma)$	NC	Sous vérification des propositions 2.21 et 2.22 $\sigma' \in \mathcal{L}_F$

Si $\sigma \in \mathcal{L}_D \setminus \mathcal{L}_F$	Ajout d'un événement $\lvert\sigma'\rvert = \lvert\sigma\rvert + 1$	Distribution d'une séquence $\lvert\sigma'\rvert > \lvert\sigma\rvert + 1$
Augmentation de FC $FC(\sigma') \supset FC(\sigma)$	NC	NC
Diminution de FC $FC(\sigma') \subset FC(\sigma)$	Sous vérification de la proposition 2.18 $\sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F$	*
Conservation de FC $FC(\sigma') = FC(\sigma)$	NC	Sous vérification des propositions 2.21 et 2.22 $\sigma' \in \mathcal{L}_D \setminus \mathcal{L}_F$

Tableau 2.2 – Récapitulatif des conditions de semi-cohérence d'un système dynamique réparable.(NC : non concluant, * : Cas mixte entre la distribution ordonnée des événements d'une séquence et l'ajout d'un événement de défaillance ou de réparation)

Les trois conditions de semi-cohérence qui ont été formulées sont donc :

- Pour une séquence défaillante : l'augmentation de la longueur de la séquence par ajout, au cours de la séquence, d'un événement faisant croître l'ensemble des composants défaillants (événement de défaillance) doit conduire à l'obtention d'une séquence défaillante si celle-ci est une séquence dysfonctionnelle.
- Pour une séquence dysfonctionnelle mais non-défaillante : l'augmentation de la longueur de la séquence par ajout, au cours de la séquence, d'un événement faisant décroître l'ensemble des composants défaillants (événement de réparation) doit conduire à l'obtention d'une séquence non-défaillante si celle-ci est une séquence dysfonctionnelle.
- L'augmentation de la longueur d'une séquence dysfonctionnelle par distribution ordonnée des événements d'une seconde séquence telle que les séquences dysfonctionnelles obtenues ne font pas varier l'ensemble des composants défaillants permet, sous conditions de vérifier les conditions 2.21 et 2.22, de conserver l'état de défaillance du système en fin de séquence.

Il est également possible de conclure dans les cas où l'augmentation et la diminution de l'ensemble des composants défaillants sont causés par la distribution d'une séquence δ dans une séquence dysfonctionnelle σ. De tel cas sont représentés par (*) dans le tableau 2.2. L'étude d'une telle variation peut alors se décomposer en plusieurs étapes :

- l'étude de la distribution d'une partie de la fonction δ qui conserve l'ensemble des composants défaillants,
- les études successives des ajouts des événements de δ faisant soit augmenter l'ensemble des composants défaillants $FC(\sigma)$ si σ est défaillante, soit diminuer si σ est non-défaillante.

Ce processus pourrait par exemple être mis en oeuvre pour étudier la distribution des événements d'une séquence $\delta = r_1 f_1 f_3$ sur la séquence défaillante $\sigma = f_1 f_2$. Dans ce cas l'étude de la distribution des événements de la séquence $\delta' = r_1 f_1$ serait réalisée avant d'étudier l'ajout de l'événements f_3 sur les séquences obtenues par l'étude précédente.

2.5.2 Condition de cohérence d'un système semi-cohérent.

La condition de cohérence d'un système semi-cohérent est décrite par une relation liant les états du système où soit les composants du système sont tous défaillants soit tous opérationnels et l'état de défaillance du système.

Cette condition de cohérence peut alors être exprimée sur la fonction de structure dynamique $\Phi_D(\sigma)$ complètement décrite par le couple $(\mathcal{L}_D, \mathcal{L}_F)$ du langage dysfonctionnel et du langage défaillant. Pour les systèmes statiques rappelons que cette condition est décomposée en deux propositions :

- Lorsque tous les composants sont défaillants, le système doit être défaillant.
- Lorsque tous les composants sont opérationnels, le système doit être opérationnel.

La transcription de la première relation pour les systèmes dynamiques est réalisée sur les séquences σ, possibles dans le système ($\sigma \in \mathcal{L}_D$), telles que tous les composants sont défaillants en fin de séquence ($FC(\sigma) = C$). Une telle séquence doit mener à la défaillance du système ($\sigma \in \mathcal{L}_F$). Cette première partie de la condition de cohérence des systèmes dynamiques semi-cohérents peut alors être exprimée sous la forme de la proposition suivante ;

$$\forall \sigma \in \mathcal{L}_d, FC(\sigma) = C \implies \sigma \in \mathcal{L}_F \tag{2.23}$$

L'expression de seconde relation est réalisée sur des séquences σ, possibles dans le système ($\sigma \in \mathcal{L}_D$), telles qu'en fin de séquence aucun composant n'est défaillant ($FC(\sigma) = \emptyset$). Une telle séquence doit permettre au système d'être opérationnel (donc non-défaillant $\sigma \notin \mathcal{L}_F$). La seconde partie de la propriété de cohérence d'un système semi-cohérent peut alors être mise sous la forme de la proposition suivante :

$$\forall \sigma \in \mathcal{L}_D, FC(\sigma) = \emptyset \implies \sigma \in \mathcal{L}_D \setminus \mathcal{L}_F \tag{2.24}$$

Un système semi-cohérent (qui vérifie les propositions 2.17,2.18,2.21 et 2.22) est donc cohérent s'il vérifie également les propositions 2.23 et 2.24.

2.6 Proposition d'énoncé informel des règles de cohérence des systèmes dynamiques réparables.

Afin de simplifier la compréhension des règles de cohérence pour les systèmes dynamiques réparables, un énoncé informel en est proposé dans cette section. Cet énoncé doit permettre d'identifier aisément chacune des propriétés qui permettent de valider la semi-cohérence et la cohérence. Il est composé des 5 règles suivantes, dont l'énoncé est inspiré de [LL03] :

- *Règle 1 : Tous les composants du système sont pertinents.*
- *Règle 2 : Une séquence menant à terme à la défaillance de tous les composants doit également mener à un système défaillant.*
- *Règle 3 : Une séquence menant à terme à l'opérabilité de tout les composants doit également mener à un système opérationnel*
- *Règle 4 : Dans les cas des séquences défaillantes :*
  - *Toute séquence dysfonctionnelle obtenue par l'ajout d'un événement de défaillance dans une séquence défaillante est une séquence défaillante.*
  - *Toute séquence dysfonctionnelle obtenue par distribution des événements d'une séquence n'ayant pas d'influence sur la séquence défaillante considérée est une séquence défaillante.*
- *Règle 5 : Dans le cas des séquences dysfonctionnelles non-défaillantes :*
  - *Toute séquence dysfonctionnelle obtenue par l'ajout d'un événement de réparation dans une séquence dysfonctionnelle non défaillante est une séquence non-défaillante*
  - *Toute séquence dysfonctionnelle obtenue par distribution des événements d'une séquence n'ayant pas d'influence sur la séquence dysfonctionnelle non-défaillante considérée est une séquence non-défaillante.*

La règle 1 associe la pertinence de l'étude de défaillance au choix des composants retenus par l'expert. Elle n'est pas explicitée dans le cadre de nos travaux et sera considérée vérifiée si le modèle d'analyse des risques est le résultat d'une construction experte. Les règles 2 et 3 décrivent les conditions pour lesquelles un système semi-cohérent est cohérent, elles se rapportent respectivement aux propositions 2.23 et 2.24 établies en section

2.5.2. La règle 4 est la description de l'effet d'une augmentation de la longueur de séquence pour les séquences défaillantes. Elle décrit deux cas : celui associé à l'augmentation de l'ensemble des composants défaillants et celui associé à la conservation de l'ensemble des composants défaillants. Elle exprime donc la proposition 2.17 ainsi que les propositions 2.21 et 2.22 appliquées uniquement sur les séquences défaillantes. La règle 5 concerne l'augmentation de la longueur des séquences dysfonctionnelles non-défaillantes avec soit diminution de l'ensemble des composants défaillants soit conservation de cet ensemble. Elle exprime la proposition 2.18 ainsi que les propositions 2.21 et 2.22 appliquées aux séquences dysfonctionnelles non-défaillantes.

L'objectif de nos travaux n'est pas la validation de ces propriétés. Elles seront donc considérées comme étant axiomatiques et respectées pour les modèles de systèmes dynamiques cohérents réparables étudiés. Ces règles ont pour nos travaux comme principal objectif de permettre des définitions formelles des notions de *noyau du langage défaillant* et de l'ensemble des *Séquences de Coupes Minimales* (SCM) qui vont maintenant être définies.

2.7 Noyau du langage défaillant et Séquences de Coupes Minimales.

Cette section a pour objectif de donner une définition formelle pour le noyau du langage défaillant ainsi que pour l'ensemble des séquences de coupe minimales. Cette définition va s'appuyer sur les propriétés de cohérence pour les systèmes dynamiques réparables explicitées précédemment.

La section 2.7.1 explicite l'utilisation des règles de cohérence pour la génération de séquences plus longues à partir d'une séquence dysfonctionnelle donnée. La section 2.7.2 va alors définir le noyau du langage défaillant à partir de ces règles, avant de donner une définition des séquences de coupe minimales en section 2.7.3 ainsi que leurs propriétés.

2.7.1 Utilisation des règles de cohérence.

Les règles 4 et 5 de cohérence des systèmes dynamiques réparables, sont les règles de semi-cohérence qui permettent, à partir d'une séquence donnée, d'obtenir des séquences

rallongées préservant le niveau de défaillance du système. Chacune de ces deux règles peut être utilisée pour définir une fonction admettant en entrée un ensemble de séquences et générant un ensemble de séquences plus longues.

La règle 4 permet de définir une fonction CR_4 admettant en entrée un ensemble S de séquences défaillantes ($S \subseteq \mathcal{L}_F$) et générant un second ensemble S' de séquences défaillantes ($S' \subseteq \mathcal{L}_F$). Cette génération est effectuée soit en ajoutant un événement de défaillance (conformément à la proposition 2.17) soit en distribuant sous certaines conditions les événements d'une autre séquence permettant la préservation de l'ensemble des composants défaillants (conformément à la propositions 2.21 et 2.22). L'ensemble S' des séquences générées doit au moins contenir les séquences défaillantes de l'ensemble S et peut également contenir des séquences défaillantes de longueur supérieures, S' est donc un sur-ensemble de l'ensemble S ($S' \supseteq S$). Cette fonction CR_4 est alors définie par la proposition suivante :

$$CR_4 : S \xrightarrow{2.17,2.21,2.22} S', (S, S') \in (\mathcal{L}_F)^2, S' \supseteq S \tag{2.25}$$

L'ensemble le plus grand qu'il est possible d'utiliser en entrée de la fonction CR_4 est le langage défaillant $\mathcal{L}_F$. Comme $\mathcal{L}_F$ est l'ensemble de toutes les séquences défaillantes, la fonction CR_4 ne peut générer que des séquences qui sont déjà élément de $\mathcal{L}_F$.

$$CR_4 : \mathcal{L}_F \xrightarrow{2.17,2.21,2.22} \mathcal{L}_F \tag{2.26}$$

La règle 5 permet de définir une fonction CR_5. Cette fonction a pour entrée un ensemble S de séquences dysfonctionnelles non défaillantes ($S \subseteq \mathcal{L}_D \setminus \mathcal{L}_F$) et pour sortie un ensemble S' de séquences dysfonctionnelles non défaillantes ($S' \subseteq \mathcal{L}_D \setminus \mathcal{L}_F$). Son rôle est d'augmenter la longueur des séquences dans S en ayant conservation de l'opérabilité du système en fin de séquence. L'augmentation peut se faire de deux manières, soit par ajout d'un événement de réparation (proposition 2.18) soit par distribution sous conditions des événements d'une seconde séquence (propositions 2.21 et 2.22). L'ensemble S' des séquences générées doit au moins contenir les séquences de l'ensemble S et peut également contenir des séquences allongées à partir des séquences de S. S' est alors un surensemble de S ($S' \supseteq S$). La fonction CR_5 peut donc être définie par :

$$CR_5 : S \xrightarrow{2.18,2.21,2.22} S', (S, S') \in (\mathcal{L}_D \setminus \mathcal{L}_F)^2, S' \supseteq S \tag{2.27}$$

L'ensemble maximal utilisable en entrée de la fonction CR_5 est l'ensemble de toutes les séquences dysfonctionnelles non défaillantes $\mathcal{L}_D \setminus \mathcal{L}_F$. Dans ce cas la sortie contient alors au moins toutes les séquences utilisée en entrée et doit être contenue dans l'ensemble des séquences dysfontionnelles non-défaillantes, on obtient alors exactement $\mathcal{L}_D \setminus \mathcal{L}_F$.

$$CR_5 : \mathcal{L}_D \setminus \mathcal{L}_F \xrightarrow{2.18,2.21,2.22} \mathcal{L}_D \setminus \mathcal{L}_F \tag{2.28}$$

L'objectif des études qualitatives de défaillance est d'obtenir l'ensemble minimal des séquences les plus courtes qui caractérise toutes les défaillances possibles du système. **Les règle 4 et 5 de cohérence des systèmes dynamiques réparables modélisent le problème inverse en proposant un moyen de rallonger des séquences tout en conservant la défaillance ou l'opérabilité du système.** Les sections suivantes vont expliciter l'utilisation de la règle 4 pour la définition du noyaux du langage défaillant et de l'ensemble des séquences de coupes minimales.

2.7.2 Noyau du langage défaillant.

L'enchaînement des événements de défaillance et de réparation des composants génère un langage défaillant de taille infinie. Il est donc impossible d'en donner une description en extension. Par ailleurs, une description en compréhension, par une expression régulière par exemple, qui caractérise le langage marqué par l'automate fini n'est pas aisément exploitable par l'expert fiabliliste. Une représentation finie, en extension minimale du langage défaillant $\mathcal{L}_F$ est donc nécessaire pour palier cette difficulté de modélisation.

Nous appelons cette représentation minimale *noyau (ou kernel) du langage défaillant* noté $Kern(\mathcal{L}_F)$. $Kern(\mathcal{L}_F)$ est l'ensemble de cardinal minimal des séquences défaillantes permettant de générer l'ensemble du langage défaillant $\mathcal{L}_F$ par application de CR_4. Le noyau peut alors être défini par la proposition suivante :

$$\begin{cases} Kern(\mathcal{L}_F) \xrightarrow{CR_4} \mathcal{L}_F \\ \nexists S \subset Kern(\mathcal{L}_F) \xrightarrow{CR_4} \mathcal{L}_F \end{cases} \tag{2.29}$$

La première partie de la proposition 2.29 définit le kernel comme étant un ensemble de séquences permettant de générer l'ensemble du langage défaillant à l'aide de CR_4, la seconde définit la minimalité du noyau par l'impossibilité d'avoir un ensemble de séquences plus petit permettant de générer l'ensemble du langage à l'aide de CR_4. Chacune des

séquences présente dans le noyau doit être une séquence défaillante afin de pouvoir être utilisée en entrée de CR_4, de plus par utilisation de CR_4, le noyau $Kern(\mathcal{L}_F)$ est un sous-ensemble de séquences défaillantes.

$$Kern(\mathcal{L}_F) \subseteq \mathcal{L}_F \tag{2.30}$$

A partir de l'exemple de la figure 2.6, l'appartenance de différentes séquences au noyau du langage défaillant peut être analysée :

- La séquence f_3 est une séquence défaillante ne pouvant être générée par une séquence de longueur plus courte. C'est donc une séquence du noyau (c'est toujours le cas pour les séquences de longueur 1).
- La séquence $f_1 f_3 f_2$ est une séquence de défaillance qui peut être générée par l'utilisation de la règle 4 de cohérence, soit par ajout de l'événement f_3 dans $f_1 f_2$, soit par ajouts successifs de f_1 et f_2 dans la séquence initialement défaillante f_3. $f_1 f_3 f_2$ peut donc être générée à l'aide de séquences défaillantes plus courtes, ce n'est pas une séquence du noyau.

Un certain nombre de propriétés des séquences du noyau peuvent être données :

Propriété 1. *Toute séquence du noyau ne peut être générée que par elle même :*

$$\forall \sigma \in Kern(\mathcal{L}_F), \nexists \sigma' \neq \sigma, \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR_4} S, \sigma \in S$$

Démonstration. Soit une séquence σ du noyau ($\sigma \in Kern(\mathcal{L}_F)$) pouvant être générée par une séquence de longueur inférieure σ' ($\sigma' \neq \sigma$) :

$$\exists \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR_4} S, \sigma \in S$$

Il est alors possible de retirer σ du noyau $Kern(\mathcal{L}_F)$ tout en conservant la génération de l'ensemble du langage défaillant :

$$Kern(\mathcal{L}_F) \setminus \{\sigma\} \xrightarrow{CR_4} \mathcal{L}_F$$

Le noyau $Kern(\mathcal{L}_F)$ ne respecte alors plus sa propriété de minimalité provenant de sa définition en 2.29. □

Propriété 2. *Le noyau du langage défaillant $Kern(\mathcal{L}_F)$ est unique.*

Démonstration. Soient $Kern_1(\mathcal{L}_F)$ et $Kern_2(\mathcal{L}_F)$ deux noyaux tels que définis par la proposition 2.29. Ces deux noyaux sont suposés disjoints $Kern_1(\mathcal{L}_F) \cap Kern_2(\mathcal{L}_F) = \emptyset$. Soit une séquence σ n'appartenant qu'à un seul de ces noyaux ($\sigma \in Kern_1(\mathcal{L}_F), \sigma \notin Kern_2(\mathcal{L}_F)$). Si $\sigma \notin Kern_2(\mathcal{L}_F)$ alors il existe $\sigma' \in \mathcal{L}_F$ générant σ par $CR4$ ($\exists \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR_4} S, \sigma \in S$). Si $\sigma \in Kern_1(\mathcal{L}_F)$ alors il n'existe pas de séquence $\sigma' \in \mathcal{L}_F$ générant σ par $CR4$ ($\nexists \sigma' \in \mathcal{L}_F, \{\sigma'\} \xrightarrow{CR_4} S, \sigma \in S$). Ces deux propositions sont en contradiction, ce cas n'est donc pas possible, les deux noyaux ne peuvent être disjoints ($Kern_1(\mathcal{L}_F) \cup Kern_2(\mathcal{L}_F) \neq \emptyset$).

Soit une séquence σ n'appartenant qu'à un seul de ces noyaux (par exemple $\sigma \in Kern_1(\mathcal{L}_F), \sigma \notin Kern_2(\mathcal{L}_F)$). Si $\sigma \notin Kern_2(\mathcal{L}_F)$ alors il existe $\sigma' \in \mathcal{L}_F$ générant σ par $CR4$ ($\{\sigma'\} \xrightarrow{CR_4} S, \sigma \in S$). Si cette séquence σ' existe, alors la séquence σ ne peut faire partie du noyau $Kern_1(\mathcal{L}_F)$ car elle peut être générée par une séquence plus courte (en contradiction avec la propriété 1). Une séquence ne peut donc être que dans un seul des deux noyaux, on a alors $Kern_1(\mathcal{L}_F) = Kern_2(\mathcal{L}_F)$.

□

Propriété 3. *Une séquence σ du noyau $Kern(\mathcal{L}_F)$ ne peut passer deux fois par le même état de l'automate représentant la fonction de structure.*

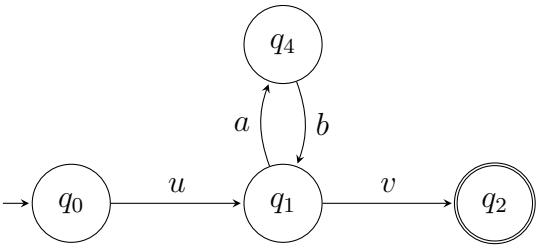


Figure 2.14 – Exemple de séquence défaillante $uabv$ passant deux fois par le même état

Démonstration. Soit la séquence $\sigma = uabv$ parmi l'infinité définie par l'expression régulière $\{u(ab)^*v\}$ du noyau du langage défaillant ($\sigma \in Kern(\mathcal{L}_F)$) appartenant au langage marqué par l'automate fini de la figure 2.14. Cette séquence passe deux fois par l'état q_1. D'après la propriété 1 cette séquence faisant partie du noyau, elle ne peut être générée à l'aide de CR_4 que par elle même. Or la séquence σ peut être générée par distribution des événements de la séquence ab dans $\sigma' = uv$ par l'utilisation de la règle 4, et en particulier grâce aux propositions 2.21 et 2.22. L'ajout de ab dans σ' est possible si la décomposition de σ' est telle que $\sigma' = \sigma'_1\sigma'_2\sigma'_3$ avec $\sigma'_1 = u, \sigma'_2 = \epsilon$ et $\sigma'_3 = v$. σ peut alors être générée par une séquence plus courte, elle n'est donc pas un élément du noyau $Kern(\mathcal{L}_F)$. □

Propriété 4. *Le nombre de séquences dans le noyau $Kern(\mathcal{L}_F)$ est fini.*

Démonstration. D'après la propriété 3, une séquence σ du noyau ne passe pas deux fois par un même état de l'automate. L'automate fini possède par définition un nombre fini d'états. La longueur maximale d'une séquence σ du noyau est donc $|\sigma| \leq (Card(Q) - 1)$. Le nombre de séquences de longueur donnée qu'il est possible de construire à partir d'un alphabet fini d'événements est fini. Le nombre de séquences dans le noyau $Kern(\mathcal{L}_F)$ est donc fini. □

2.7.3 Séquences de coupe Minimales.

L'ensemble des séquence de coupe minimale est l'ensemble des séquences du noyau du langage défaillant qui permet de générer l'ensemble des séquences de coupes $\mathcal{L}_{CS}$. Une séquence de coupe à été définie en section 2.3.2 comme étant une séquence défaillante dont le seul état marqué traversé (représentant la défaillance du système) est celui atteint en fin de séquence.

L'ensemble des séquences de coupe minimales (SCM) est donc le sous-ensemble de cardinal minimal de $\mathcal{L}_{CS}$ qui permet de générer un ensemble de séquences défaillantes contenant toutes les séquences de coupe, $\mathcal{L}_{CS}$ peut donc être décrit par la proposition suivante :

$$\begin{cases} SCM \subseteq \mathcal{L}_{CS}, SCM \xrightarrow{CR_4} S, S \supseteq \mathcal{L}_{CS} \text{ avec } S \subseteq \mathcal{L}_F \\ \nexists S' \subset SCM(\mathcal{L}_F) \xrightarrow{CR_4} S, S \supseteq \mathcal{L}_{CS} \text{ avec } S \subseteq \mathcal{L}_F \end{cases} \tag{2.31}$$

La première partie de la proposition 2.31 définit l'ensemble des séquences de coupe minimales comme étant un ensemble de séquences permettant de générer un sur-ensemble des séquences de coupe. La seconde définit la minimalité de l'ensemble des séquences de coupe minimales par l'impossibilité d'avoir un ensemble plus petit permettant de générer au moins l'ensemble des séquences de coupe à l'aide de CR_4.

Propriété 5. *Une séquence de coupe minimale est une séquence du noyau du langage défaillant $Kern(\mathcal{L}_F)$.*

$$\forall \sigma \in SCM, \sigma \in Kern(\mathcal{L}_F) \tag{2.32}$$

Démonstration. Soit σ une séquence de coupe minimale n'appartenant pas au noyau du langage défaillant ($\sigma \notin Kern(\mathcal{L}_F)$. σ est par définition une séquence défaillante ($\sigma \in \mathcal{L}_{CS} \subseteq \mathcal{L}_F$). Si σ n'est pas élément du noyau ($\sigma \in SCM$), alors il existe une séquence plus courte σ' permettant de générer σ à l'aide de CR_4. Il est donc possible de générer au moins l'ensemble des séquences de coupe (dont σ) en utilisant une réduction de l'ensemble des séquences de coupe minimale à $SCM \setminus \{\sigma\}$. Dans ce cas SCM ne respecte pas la condition de minimalité exprimée par la proposition 2.31. □

La propriété 5 est vérifiée pour toutes les séquences de coupe minimales, l'ensemble SCM des séquences de coupe minimales est donc un sous-ensemble du noyau du langage défaillant :

$$SCM \subseteq Kern(\mathcal{L}_F) \tag{2.33}$$

Ceci implique que les propriétés 1, 3 et 4 sont également vérifiées pour l'ensemble des séquences de coupe minimales. La propriété 4 permet notamment de valider que l'ensemble des séquences de coupe minimales est fini.

L'ensemble des séquences de coupe minimales est également par définition un sous-ensemble des séquences de coupes $\mathcal{L}_{CS}$ ($SCM \subseteq \mathcal{L}_C S$). l'ensemble SCM est alors défini comme étant l'intersection entre le noyau du langage défaillant et l'ensemble des séquences de coupe :

$$SCM = \mathcal{L}_{CS} \cap Kern(\mathcal{L}_F) \tag{2.34}$$

A partir de l'exemple de la figure 2.6, l'appartenance de différentes séquences défaillantes à l'ensemble des séquences de coupe minimales peut être analysée :

- $\sigma = f_3 f_1$ est une séquence défaillante, mais n'est pas une séquence de coupe car elle traverse deux états marqués.
- $\sigma = f_1 f_2$ est une séquence de coupe car seul le dernier état atteint est marqué. Cette séquence de coupe ne peut pas être générée à l'aide d'une séquence de coupe de longueur inférieure, c'est donc une séquence de coupe minimale.
- $\sigma = f_1 f_3$ est une séquence de coupe car seul le dernier état atteint est marqué. Cette séquence de coupe peut être générée par l'ajout de l'événement de défaillance f_1 dans la séquence de coupe f_3, conformément à la règle 4 de cohérence des sys-

tèmes dynamiques ; ce n'est donc pas une séquence de coupe minimale.

Les définitions du noyau du langage défaillant ainsi que de l'ensemble des séquences de coupe minimales étant établies, le calcul de ces deux ensembles va être explicité dans la section suivante.

2.8 Calcul du noyau du langage défaillant et de l'ensemble des séquences de coupes minimales.

Dans la section 2.3.2, on a défini le concept d'ensemble de séquences de coupe minimales pour les systèmes dynamiques réparables et cohérents. L'objectif de cette partie est de déterminer un algorithme général permettant le calcul des séquences de coupe minimales et du noyau du langage défaillant quelque soit l'automate fini considéré (conforme aux hypothèses développées en section 2.3.1).

L'application $CR4$ permet d'obtenir un sur-ensemble d'un ensemble de séquences défaillantes, tout en restant dans le langage défaillant, par l'utilisation des règles de cohérence. Si on considère un ensemble S de séquences défaillantes, alors chaque séquence de $CR_4(S)$, est soit une séquence de S soit le résultat d'un allongement d'une séquence de S par ajout d'un événement de défaillance ou par distribution des événements d'une séquence non-influente. Chaque élément σ de $CR_4(S)$ peut être le résultat d'une ou plusieurs de ces opérations, ainsi sur l'exemple de la figure 2.6, la séquence $f_1 f_2 f_3 r_1$ peut être obtenue soit par ajout de f_2 dans la séquence défaillante $f_1 f_3 r_1$ soit par distribution ordonnée des événements de $f_1 r_1$ dans la séquence défaillante $f_2 f_3$. Certaines séquences σ de $CR_4(S)$ n'ont pas un antécédent unique dans S, CR_4 est alors non injective, et donc, non bijective.

L'application CR_4, non bijective, n'admet pas d'application inverse CR_4^{-1} pouvant permettre un calcul direct du noyau à partir du langage défaillant.

On se propose donc de calculer, à partir de la fonction de structure (par la connaissance du couple $(\mathcal{L}_F, \mathcal{L}_D)$ représenté par un automate fini, le noyau du langage défaillant par identification et retrait des séquences du langage défaillant ne pouvant appartenir au noyau. Une séquence défaillante est considérée comme ne faisant pas partie du noyau

lorsqu'il est possible d'identifier un séquence plus courte permettant de la déterminer à l'aide de $CR4$, cette séquence ne respecte alors pas la propriété 1 des séquences du noyau.

Cependant, le nombre de séquences dans le langage défaillant étant infini (notamment à cause des cycles de défaillance et de réparation des composants), ce calcul n'est pas réalisable. Il est donc nécessaire de déterminer un ensemble fini de séquences du langage défaillant permettant ce calcul tout en ayant conservation de la propriété provenant de la définition 2.29 du noyau $Kern(\mathcal{L}_F)$, qui garantit la possibilité de calculer l'ensemble du langage défaillant à l'aide de $CR4$. Cet ensemble fini est *l'ensemble des séquences ne passant pas deux fois par un même état*. Cet ensemble est définit en section 2.8.1, ainsi que quelques propriétés garantissant la conservation de la capacité à recalculer $\mathcal{L}_F$. La section 2.8.2 présentera alors l'algorithme permettant le calcul du noyau $Kern(\mathcal{L}_F)$ et de l'ensemble SCM des séquences de coupe minimales.

2.8.1 Définition de l'ensemble des séquences sans circuits de l'AF.

Conformément à la propriété 3 sur les séquences appartenant au noyau $Kern(\mathcal{L}_F)$, aucune de ces séquences ne passe deux fois dans un même état. Cette propriété va être dérivée sur les séquences du langage défaillant $\mathcal{L}_F$ permettant ainsi, à partir d'un ensemble infini de séquences, de déterminer un ensemble fini permettant le calcul du noyau.

A partir du langage défaillant $\mathcal{L}_F$, il est possible de réaliser une séparation entre deux sous-ensembles :

- $\mathcal{L}_F^{NL}$ représentant un ensemble fini de séquences ne passant pas deux fois par le même état.
- $\mathcal{L}_F^{L}$ représentant un ensemble infini de séquences passant au moins deux fois dans un des états traversés par cette séquence.

Chacune des séquences défaillantes est dans l'une seulement de ces catégories, le langage défaillant est donc l'union de ces deux sous-langages disjoint.

$$\mathcal{L}_F = \mathcal{L}_F^{NL} \cup \mathcal{L}_F^{L} \tag{2.35}$$

$$\mathcal{L}_F^{NL} \cap \mathcal{L}_F^{L} = \emptyset \tag{2.36}$$

Par utilisation de la fonction de transition δ de l'automate sur les préfixes d'une sé-

quence à partir de l'état initial q_0, il est possible de définir formellement $\mathcal{L}_F^{NL}$ par la proposition :

$$\mathcal{L}_F^{NL} = \{\sigma \in \mathcal{L}_F | \forall (\sigma_1, \sigma_2) \in (Pref(\sigma))^2 \wedge \sigma_1 \neq \sigma_2, \delta(q_0, \sigma_1) \neq \delta(q_0, \sigma_2)\} \tag{2.37}$$

$$\mathcal{L}_F^L = \mathcal{L}_F \setminus \mathcal{L}_F^{NL} \tag{2.38}$$

Propriété 6. *Chacune des séquences de $\mathcal{L}_F^L$ peut être calculée à l'aide d'une séquence de $\mathcal{L}_F^{NL}$ à l'aide des propositions 2.21, 2.22 de $CR4$*

$$\forall \sigma \in \mathcal{L}_F^L, \exists \sigma' \in \mathcal{L}_F^{NL}, \sigma' \xrightarrow{2.21,2.22} S, \sigma \in S \tag{2.39}$$

Démonstration. Soit une séquence défaillante σ passant deux fois par un même état q, $\sigma \in \mathcal{L}_F^L$. Soit la décomposition de σ en $\sigma = \sigma_1 \alpha \sigma_3$ telle que, à l'issue de σ_1, le système se trouve dans le même état qu'à l'issue de $\sigma_1\alpha$ ($\delta(q_0, \sigma_1) = \delta(q_0, \sigma_1\alpha)$). Soit la séquence défaillante $\sigma' = \sigma_1\sigma_3$, σ' ne passe alors plus qu'une seule fois par l'état q donc $\sigma' \in \mathcal{L}_F^{NL}$. σ' peut générer σ à l'aide de CR_4 par ajout de la séquence α après σ_1 (conformément aux propositions 2.21 et 2.22).

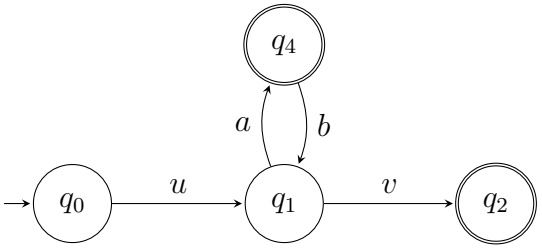


Figure 2.15 – Exemple de séquence défaillante passant deux fois dans le même état

Ce principe peut être illustré en considérant $\sigma = uabv$ décomposée en $\sigma_1 = u$, $\sigma_3 = v$ et $\alpha = ab$. Cette séquence est présentée en figure 2.15, extraite de l'exemple de la figure 2.6. La séquence σ est alors générée par un ajout unique de $\alpha = ab$ dans $\sigma' = uv$ après $\sigma_1 = u$.

□

Propriété 7. *Le sous-langage fini des séquences ne passant pas deux fois par le même état, $\mathcal{L}_F^{NL}$, est suffisant pour calculer l'ensemble du langage défaillant $\mathcal{L}_F$ à l'aide de $CR4$.*

$$\mathcal{L}_F^{NL} \xrightarrow{CR4} \mathcal{L}_F \tag{2.40}$$

Démonstration. D'après la propriété 6, toutes les séquences de $\mathcal{L}_F^L$ peuvent être calculée à l'aide d'une séquence de $\mathcal{L}_F^{NL}$, on a donc :

$$\mathcal{L}_F^{NL} \xrightarrow{CR4} S, S \supseteq \mathcal{L}_F^L$$

De même par définition, tous les éléments de S, en entrée de CR_4, se trouvent dans l'image $CR_4(S)$, on a donc également :

$$\mathcal{L}_F^{NL} \xrightarrow{CR4} S, S \supseteq \mathcal{L}_F^{NL}$$

Or d'après la définition du noyau, l'ensemble maximal qu'il est possible d'obtenir en sortie de CR_4 est $\mathcal{L}_F$. De plus les deux ensembles $\mathcal{L}_F^L$ et $\mathcal{L}_F^{NL}$ doivent être dans $CR_4(\mathcal{L}_F^{NL})$; la définition 2.36 de ces deux langages ($\mathcal{L}_F^L \cup \mathcal{L}_F^{NL} = \mathcal{L}_F$) permet alors de vérifier que $CR_4(\mathcal{L}_F^{NL}) = \mathcal{L}_F$.

□

Propriété 8. *Le noyau du langage défaillant $Kern(\mathcal{L}_F)$ est entièrement contenu dans $\mathcal{L}_F^{NL}$*

$$Kern(\mathcal{L}_F) \subseteq \mathcal{L}_F^{NL} \tag{2.41}$$

Démonstration. D'après la définition du noyau (2.30), le noyau est un sous-ensemble de $\mathcal{L}_F$. De plus, d'après la propriété 3 les séquences du noyau ne passent pas deux fois par le même état. Donc quelque soit la séquence du noyau considérée, par définition, cette séquence appartient à $\mathcal{L}_F^{NL}$. □

Après définition de $\mathcal{L}_F^{NL}$ et de ses propriétés, les relations entre les différents ensembles sont présentées en figure 2.16.

Notre objectif est le calcul du noyau du langage défaillant $Kern(\mathcal{L}_F)$. Ce noyau permet la génération du langage défaillant $\mathcal{L}_F$ par utilisation de $CR4$. La fonction inverse de CR_4 n'existant pas, il faut alors identifier dans $\mathcal{L}_F$ les séquences du noyau. Cette recherche est impossible car le langage défaillant est de cardinal infini. Pour pallier cette difficulté et permettre le calcul, un langage de cardinal fini est défini comme étant l'ensemble des séquences défaillantes ne passant pas deux fois par un même état. Cet ensemble est suffisant pour permettre la reconstruction du langage défaillant par CR_4, de plus cet ensemble contient toutes les séquences du noyau. Le calcul va maintenant consister à rechercher dans l'ensemble fini $\mathcal{L}_F^{NL}$ un ensemble fini de séquence constituant le noyau $Kern(\mathcal{L}_F)$.

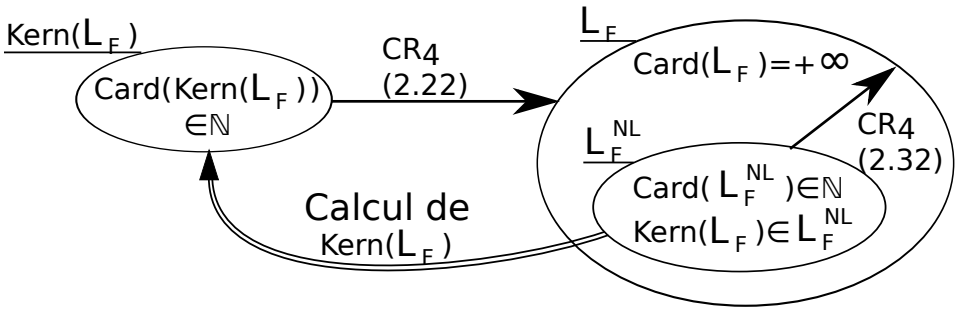


Figure 2.16 – Relations entre le noyau du langage défaillant, l'ensemble des séquences défaillantes sans boucles et le langage défaillant.

2.8.2 Algorithme de calcul du noyau du langage défaillant.

La restriction de l'ensemble infini des séquences défaillantes $\mathcal{L}_F$ à un ensemble fini de séquences $\mathcal{L}_F^{NL}$ permet maintenant un calcul des séquences du noyau $Kern(\mathcal{L}_F)$ par un traitement fini de séquences. L'algorithme de calcul du noyau est décomposé en deux phases successives :

- Une première phase d'exploration de l'automate servant à générer l'ensemble fini de séquences ne passant pas deux fois par le même état $\mathcal{L}_F^{NL}$, et un ensemble fini $\mathcal{L}_F^R$ de séquences défaillantes plus courtes que la plus longue séquence de $\mathcal{L}_F^{NL}$.
- Une seconde phase d'identification des séquences du noyau $Kern(\mathcal{L}_F)$ dans l'ensemble $\mathcal{L}_F^{NL}$ calculé par l'étape précédente. Cette identification est effectuée en retirant de l'ensemble $\mathcal{L}_F^{NL}$ les séquences pouvant être recalculées à l'aide d'une séquence plus courte par l'utilisation de $CR4$.

2.8.2.1 Principe de calcul

A partir de l'ensemble fini des séquences défaillantes sans boucle $\mathcal{L}_F^{NL}$, il est nécessaire d'identifier dans ces séquences quelles sont celles qui sont dans le noyau $Kern(\mathcal{L}_F)$. Conformément à la propriété 1, une séquence du noyau ne peut être générée que par elle-même à travers $CR4$. Considérant une séquence σ de $\mathcal{L}_F^{NL}$ et s'il existe une séquence de $\mathcal{L}_F$ de taille inférieure permettant de générer σ par $CR4$, alors σ ne peut être dans le noyau.

La séquence σ peut être générée par deux mécanismes, soit par l'ajout d'un événement de défaillance dans une séquence plus courte d'un événement, soit par la distribution ordonnée des événements d'une séquence dans une séquence plus courte. Ce principe est

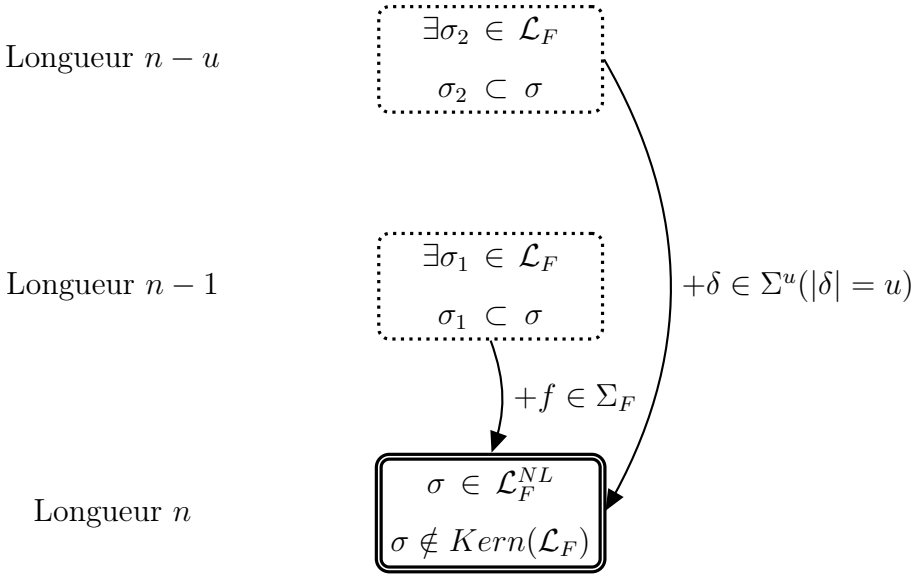


Figure 2.17 – Principe de calcul du noyau

illustré en figure 2.17 où σ peut être générée par ajout d'un événement de défaillance dans σ_1 ou par distribution des événements de δ dans σ_2.

Deux ensembles finis de séquences sont donc nécessaires pour effectuer le calcul du noyau, d'une part $\mathcal{L}_F^{NL}$, l'ensemble fini de séquences défaillantes et d'autre part l'ensemble fini de séquences défaillantes de longueur inférieure à la séquence de $\mathcal{L}_F^{NL}$ de longueur maximale. Ce second ensemble, restriction du langage défaillant, est noté $\mathcal{L}_F^R$ et est définit par la proposition suivante :

$$\mathcal{L}_F^R = \{\sigma \in \mathcal{L}_F \mid |\sigma| \leq Max(|\sigma| - 1, \sigma \in \mathcal{L}_F^{NL})\} \tag{2.42}$$

2.8.2.2 Génération des séquences de l'ensemble $\mathcal{L}_F^{NL}$ à partir de l'AF

L'objectif de cette première partie du calcul est de générer un nombre fini de séquences de $\mathcal{L}_F$ ne passant pas deux fois par le même état ($\mathcal{L}_F^{NL}$) et un nombre fini de séquences plus courtes du langage défaillant ($\mathcal{L}_F^R$). Ces séquences sont toutes des séquences possibles dans l'automate ($\mathcal{L}_D$). Deux algorithmes permettant le calcul de ces séquences par exploration de l'automate sont présentés. L'algorithme 1 permet le calcul de l'ensemble $\mathcal{L}_F^{NL}$, et l'algorithme 2 le calcul de $\mathcal{L}_F^R$.

La seule entrée nécessaire pour le calcul de $\mathcal{L}_F^{NL}$ est l'automate, considéré conforme aux hypothèses décrites en 2.3.1. Le principe utilisé pour générer toutes les séquences

Algorithme 1 Calcul de $\mathcal{L}_F^{NL}$ à partir de l'automate A

Require: $A =< \Sigma, Q, q_i, Q_m, \delta >$ Automate conforme aux propriétés décrites en 2.3.1

1: *# Initialisation :*
2: $S \leftarrow \epsilon$ *# Ensemble temporaire de séquences utilisés par la génération*
3: $KS(\epsilon) = q_O$ *# Séquences des états traversés par chaque séquence d'événements*
4: $\mathcal{L}_F^{NL} = \emptyset$
5: $\mathcal{L}_F^{R} = \emptyset$
6: *# Calcul de* $\mathcal{L}_F^{NL}$
7: **for all** $\sigma \in S$ **do**
8: $q_i = KS(\sigma)_{|KS(\sigma)|}$ *# Sélection du dernier état atteint par* σ
9: **for all** $\delta_x \in \delta, \delta_x(q_i, u) \in Q, \delta_x(q_i, u) \notin KS(\delta)$ **do**
10: *# Sélection de toutes les transitions ne revenant pas dans un état déjà rencontré par* σ *: Nouvelle séquence* σu
11: $S \leftarrow \sigma u$
12: $KS(\sigma u) = KS(\sigma)\delta_x(q_i, u)$
13: **if** $\delta_x(q_i, u) \in Q_M$ **then**
14: *# Si le dernier état est marqué : Ajout de* σu *dans* $\mathcal{L}_F^{NL}$
15: $\mathcal{L}_F^{NL} \leftarrow \sigma u$
16: **end if**
17: **end for**
18: **end for**
19: **return** $\mathcal{L}_F^{NL}$

sans boucle de l'automate est de conserver, associés à chacune des séquences, les états déjà traversés. Cette information est portée par les séquences d'états $KS(\sigma)$ associées à chacune des séquences σ. Cette séquence d'états porte deux informations : les états traversés par la séquence ainsi que l'ordre dans lequel ils ont été traversés. Après initialisation de l'automate sur son état initial (l 1-4) et sur chaque séquence σ précédemment identifiée sur le calcul (après l'initialisation , uniquement ϵ), les transitions pouvant faire évoluer l'automate à partir du dernier état atteint par σ (q_i) et ne permettant pas d'atteindre un état déjà connu à travers σ sont identifiées (l 5-7). A partir de ces transitions, les séquences construites par concaténation de l'événement associé à la transition avec σ sont placées dans l'ensemble S des séquences identifiées (l 9-10). Si le dernier état atteint par

Algorithme 2 Calcul de $\mathcal{L}_F^R$ à partir de l'automate A connaissant $\mathcal{L}_F^{NL}$

Require: $A =< \Sigma, Q, q_i, Q_m, \delta >$ Automate conforme aux propriétés décrites en 2.3.1, $\mathcal{L}_F^{NL}$

1: *# Initialisation :*
2: $S \leftarrow \epsilon$ *# Ensemble temporaire de séquences utilisés par la génération*
3: $KS(\epsilon) = q_O$ *# Séquences des états traversés par chaque séquence*
4: $\mathcal{L}_F^R = \emptyset$
5: *# Calcul de* $\mathcal{L}_F^R$
6: $x = Max(|\sigma| - 1, \sigma \in \mathcal{L}_F^{NL})$ *# Longueur maximale des séquences défaillantes à extraire de l'automate*
7: **for all** $\sigma \in S$ **do**
8: *# Restriction sur la longueur*
9: **if** $|\sigma| < x - 1$ **then**
10: $q_i = KS(\sigma)_{|KS(\sigma)|}$ *# Sélection du dernier état atteint par* σ
11: **for all** $\delta_x \in \delta, \delta_x(q_i, u) \in Q$ **do**
12: $S \leftarrow \sigma u$
13: $KS(\sigma u) = KS(\sigma)\delta_x(q_i, u)$
14: **if** $\delta_x(q_i, u) \in Q_M$ **then**
15: *# Si le dernier état est marqué : Ajout de* σu *dans* $\mathcal{L}_F R$
16: $\mathcal{L}_F^R \leftarrow \sigma u$
17: **end if**
18: **end for**
19: **end if**
20: **end for**
21: **return** $\mathcal{L}_F^R$

cette nouvelle séquence est marqué, alors la séquence est placée dans l'ensemble $\mathcal{L}_F^{NL}$ (l 13). En fin de génération , l'ensemble $\mathcal{L}_F^{NL}$ est retourné par l'algorithme.

Le calcul de $\mathcal{L}_F^R$ est basé sur le même principe que le calcul de $\mathcal{L}_F^{NL}$, mais nécessite la connaissance de $\mathcal{L}_F^{NL}$ afin de déterminer la longueur maximale des séquences défaillantes à extraire. Une fois cette longueur connue (Alg. 2, l 6), le calcul est borné en longueur (Alg. 2, l 9) mais n'est pas restreint par la condition de non passage deux fois dans un

même état.

2.8.2.3 Calcul du noyau $Kern(\mathcal{L}_F)$

Algorithme 3 Calcul de $Kern(\mathcal{L}_F)$ à partir des séquences de $\mathcal{L}_F^{NL}$ et de $\mathcal{L}_F^R$

Require: $\mathcal{L}_F^{NL}$: Ensemble des séquences défaillantes sans boucle, calculées par l'algorithme 1, $FC(\sigma)$: Fonction permettant le calcul de l'ensemble des composants défaillants à l'issu de σ (2.3.4)

1: *# Initialisation :*
2: $Kern(\mathcal{L}_F) = \mathcal{L}_F^{NL}$ *# Initialisation du noyau*
3: **for all** $\sigma \in \mathcal{L}_F^{NL}$ **do**
4: *# Test de génération de σ par ajout d'un événement de défaillance dans une séquence plus courte suivant la proposition 2.18*
5: **for all** $\sigma' \in \mathcal{L}_F^R \wedge |\sigma'| = |\sigma| - 1$ **do**
6: **if** $\sigma' \in \sigma \wedge FC(\sigma') \subset FC(\sigma) \wedge Card(FC(\sigma')) = Card(FC(\sigma)) - 1$ **then**
7: $Kern(\mathcal{L}_F)$=$Kern(\mathcal{L}_F) \setminus \{\sigma\}$
8: **end if**
9: **end for**
10: *# Test de génération de σ par distribution d'une séquence dans une séquence plus courte suivant les propositions 2.21 et 2.22*
11: **for all** $\sigma' \in \mathcal{L}_F^R$ **do**
12: **if** $\sigma' \in \sigma \wedge FC(\sigma') = FC(\sigma)$ **then**
13: *# Pour chaque décomposition possible :*
14: **for all** $(\delta, \sigma'_1, \sigma'_2, \sigma'_3) \in DEC(\sigma, \sigma')$ **do**
15: **if** $\forall(\sigma'_{21}, \sigma'_{22}), \sigma'_{21}\sigma'_{22} = \sigma'_2, \exists\delta(q_0, \sigma'_1\sigma'_{21}\delta\sigma'_{22}\sigma'_3) \in Q \wedge \delta(q_0, \sigma'_1\sigma'_{21}) = \delta(q_0, \sigma'_1\sigma'_{21}\delta)$ **then**
16: $Kern(\mathcal{L}_F)$=$Kern(\mathcal{L}_F) \setminus \{\sigma\}$
17: **end if**
18: **end for**
19: **end if**
20: **end for**
21: **end for**
22: **return** $Kern(\mathcal{L}_F)$

L'algorithme 3 permet pour chacune des séquences σ de $\mathcal{L}_F^{NL}$ (calculées par l'algorithme 1) d'identifier les séquences de $\mathcal{L}_F^R$ (calculées par l'algorithme 2) pouvant potentiellement générer σ par $CR4$ en validant cette génération dans le cas où σ n'est pas une séquence du noyau.

L'ajout d'un événement de défaillance dans une séquence défaillante σ' conformément à la proposition 2.17 peut générer σ. La recherche d'une telle séquence est effectuée entre les lignes 4 et 9. S'il existe une séquence σ' dans $\mathcal{L}_F^R$ de longueur $|\sigma| - 1$ et possédant un composant défaillant de moins que σ, alors σ peut être générée par σ'. σ ne fait alors pas partie du noyau $Kern(\mathcal{L}_F)$ (l 7).

La distribution d'une séquence δ dans une séquence défaillante σ' permettant d'obtenir des séquences conservant le même ensemble de composants défaillants que σ', dont σ, conformément aux propositions 2.21 et 2.22 (l 10-19). Il faut donc identifier pour chaque séquence σ les séquences σ' plus courtes, inclues dans σ et possédant le même ensemble de composants défaillants. Pour chaque couple (σ, σ') il est nécessaire d'identifier la ou les inclusions possibles de σ' en $\sigma' = \sigma'_1\sigma'_2\sigma'_3$, ainsi que la séquence δ dont les événements sont distribués dans σ'. Cette inclusion est décrite par la figure 2.18 et un exemple d'inclusion multiple pour le couple $(abcabc, ac)$ est donné en tableau 2.3. Si parmi ces inclusions au moins une valide la proposition 2.21 alors σ est générée par la proposition 2.22, elle ne peut donc pas être dans le noyau (l 15). Afin que la proposition 2.21 soit validée par une décomposition de σ' en $(\sigma'_1, \sigma'_2, \sigma'_3)$, il faut que chaque séquence $\sigma'_1\sigma'_{21}\delta\sigma'_{22}\sigma'_3$ avec $\sigma'_{21}\sigma'_{22} = \sigma'_2$ soit possible dans l'automate $(\exists Succ(\sigma'_1\sigma'_{21}\delta\sigma'_{22}\sigma'_3) \in Q)$ et passe deux fois dans le même état après $\sigma'_1\sigma'_{21}$ et $\sigma'_1\sigma'_{21}\delta$, c'est-à-dire que leurs successeurs sont les mêmes $(Succ(\sigma'_1\sigma'_{21}) = Succ(\sigma'_1\sigma'_{21}\delta))$ (l 14).

Inclusions possibles	δ	σ'_1	σ'_2	σ'_3
abc**c**ab**c**	$babc$	a	c	ϵ
abca**b**c	$bcab$	a	ϵ	c
abc**a**b**c**	$abcb$	ϵ	a	c

Tableau 2.3 – Inclusions possibles de σ par σ' en utilisant $\sigma = abcabc$ et $\sigma' = ac$.

L'obtention des séquences de coupe minimales SCM peut être réalisée de deux ma-

$$\begin{aligned}\sigma =& s_0 \dots s_a s_b s_c \dots s_u s_v s_w \dots s_x s_y s_z \dots s_{|\sigma|} \\ \sigma' =& \sigma'_0 \dots \sigma'_{|\sigma'|} \\ =& \overbrace{s_0 s_1}^{\sigma'_1} \underbrace{s_b s_v s_y}_{\sigma' 2} \overbrace{s_{|\sigma|-2} s_{|\sigma|-1} s_{|\sigma|}}^{\sigma'_3} \\ \delta =& s_2 \dots s_a s_c \dots s_u s_w \dots s_x s_z \dots s_{|\sigma|-3}\end{aligned}$$

Figure 2.18 – Inclusion de σ' dans σ permettant l'identification de $(\delta, \sigma'_1, \sigma'_2, \sigma'_3)$

nières différentes : soit par restriction de $Kern(\mathcal{L}_F)$, obtenu par les algorithmes précédents, aux seules séquences de coupe, soit par utilisation des algorithmes 1 et 3 avec l'ensemble des séquences de coupe $\mathcal{L}_{CS}$ en entrée.

2.8.2.4 Complexité du calcul du noyau.

L'objectif de cette section est de donner une borne supérieure asymptotique de la complexité du calcul du noyau du langage défaillants. Les entrées et sorties des algorithmes 1 à 3 sont représentés en figure 2.19.

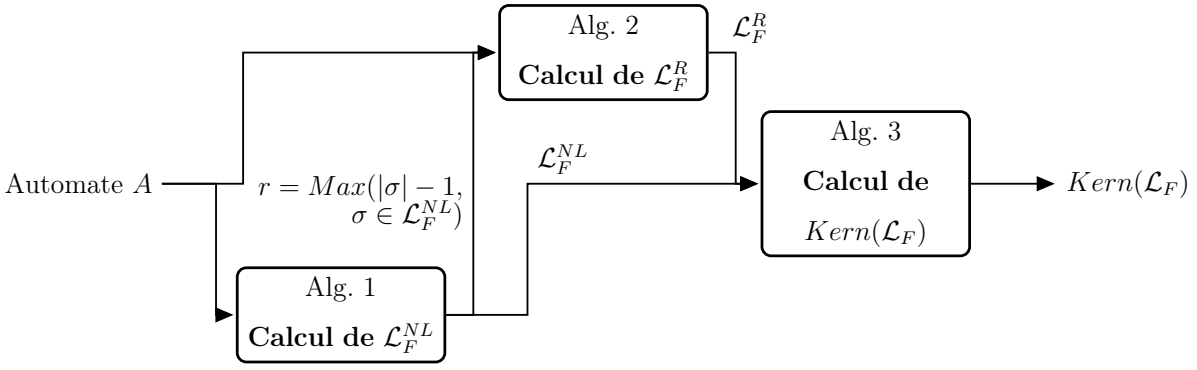


Figure 2.19 – Utilisation des algorithmes

L'hypothèse supplémentaire posée sur l'automate pour le calcul de complexité est : *l'automate est considéré comme fortement connexe : il existe une transition entre chaque couple d'états q_i et q_j de l'automate, le nombre d'états de l'automate est noté $q = Card(Q)$.* Cette hypothèse permet alors de quantifier la complexité en fonction du nombre d'états de l'automate plutôt qu'en nombre de transitions.

L'algorithme 1 ($A1$) a pour fonction la recherche de tous les chemins partant de l'état

initial, finissant dans un état marqué et ne passant pas deux fois dans un même état. Le problème de recherche d'un de ces chemins est d'une complexité de classe **P**. Cet algorithme ne recherche pas une solution au problème, mais l'ensemble de toutes les solutions à un problème polynômial. La classe de complexité spécifique pour le dénombrement de solution de problèmes polynômiaux est notée #**P** (sharp-P), cette classe décrit notre problème qui peut s'exprimer comme étant : *Combien y a-t-il de séquences dans l'automate ne passant pas deux fois dans un même état et finissant dans un état marqué.* Dans le pire des cas, tous les états sont marqués, le nombre de solutions potentielles est alors donné par une somme des arrangements d'états dont le premier élément est l'état initial q_0, et dont la longueur maximale est le nombre d'états.

$$A1(q) = 1 + \sum_{k=0}^{k=q-1} A_{q-1}^{k} = 1 + \sum_{k=0}^{k=q-1} \frac{q-1!}{(q-1-k)!} \tag{2.43}$$

Le terme le plus grand de cette somme est $(q-1)!$, une borne supérieure asymptotique de la complexité de l'algorithme 1 pouvant être donnée est $q!$ soit encore :

$$\mathcal{O}(A1(q)) = \mathcal{O}(q!) \tag{2.44}$$

L'algorithme 2 ($A2$) a pour fonction la recherche de tous les chemins partant de l'état initial, finissant dans un état marqué et de longueur inférieure ou égale à $r = Max(|\sigma| - 1, \sigma \in \mathcal{L}_F^{NL})$. Le problème de recherche d'un de ces chemin est d'une complexité de classe **P**. Comme cet algorithme a pour fonction un dénombrement des solutions à ce problème, la classe de complexité de l'algorithme 2 est également #**P**. Dans le pire des cas, tous les états sont marqués, le nombre de solutions potentielles est alors donné par la somme des combinaisons avec répétitions dont le premier élément est l'état initial q_0 et dont la longueur maximale est $r = Max(|\sigma| - 1, \sigma \in \mathcal{L}_F^{NL})$.

$$A2(q,r) = 1 + \sum_{k=0}^{k=r} \Gamma_{q-1}^{k} = 1 + \sum_{k=0}^{k=r} \frac{(q+k-1)!}{k!(q-1)!} \tag{2.45}$$

Une bonne supérieure asymptotique de la complexité de l'algorithme 2 pouvant être donnée est :

$$\mathcal{O}(A2(q,r)) = \mathcal{O}(q^r) \tag{2.46}$$

L'algorithme 3 ($A3$) compare chaque séquence du langage défaillant sans boucle $\mathcal{L}_F^{NL}$ avec chaque séquence plus courte provenant du langage défaillant restreint $\mathcal{L}_F^R$. En considérant que chacune des séquences de $\mathcal{L}_F^{NL}$ est comparée à chacune des séquences de $\mathcal{L}_F^R$, qu'il n'existe qu'une seule inclusion entre chaque séquences comparées, il est possible de poser comme borne asymptotique de complexité :

$$\mathcal{O}(A3(\mathcal{L}_F^{NL}, \mathcal{L}_F^R)) = \mathcal{O}(Card(\mathcal{L}_F^{NL}) * Card(\mathcal{L}_F^R))$$

L'algorithme 3 ($A3$) est donc de complexité linéaire vis a vis des nombres de séquences dans chacun des deux langages.

La complexité des algorithmes 1 et 2 ne provient pas de la difficulté à explorer les chemins de l'automate, mais de la taille des solutions à déterminer (tous les chemins, de manière exhaustive). La complexité de l'algorithme 3 est alors liée à la taille des langages obtenus par les algorithmes 1 et 2. Cette complexité est calculée au pire des cas à partir d'une hypothèse simplificatrice mais pessimiste. Dans la réalité, les cas traités révèlent une complexité de calcul moindre, qui nous semble acceptable par rapport aux techniques existantes.

2.8.2.5 Calcul direct des séquences de coupe minimales.

Il est également possible de réaliser le calcul direct des séquences de coupe minimales sans passer par un calcul préalable des séquences du noyau. Un inconvénient de ce calcul est que l'ensemble des séquences obtenues permet uniquement le calcul d'au moins l'ensemble des séquences de coupe (par définition cf. 2.31) et ne permet plus la validation de la conservation de la capacité à recalculer exactement l'ensemble des séquences défaillantes par utilisation du langage dysfonctionnel représenté par l'automate et des règles de cohérence. Cette incapacité de reconstruction provient du fait que la conservation de toutes les informations (donc des séquences) permettant la reconstruction complète n'est plus garantie : il peut exister des séquences du noyau ne faisant pas partie de l'ensemble des séquences de coupes.

La seule recherche des séquences de coupe minimales permet une diminution de la complexité du calcul, car ces séquences se trouvent toutes dans l'ensemble des séquences de coupe ne passant pas deux fois par un même état (combinaison de la propriété 6 ap-

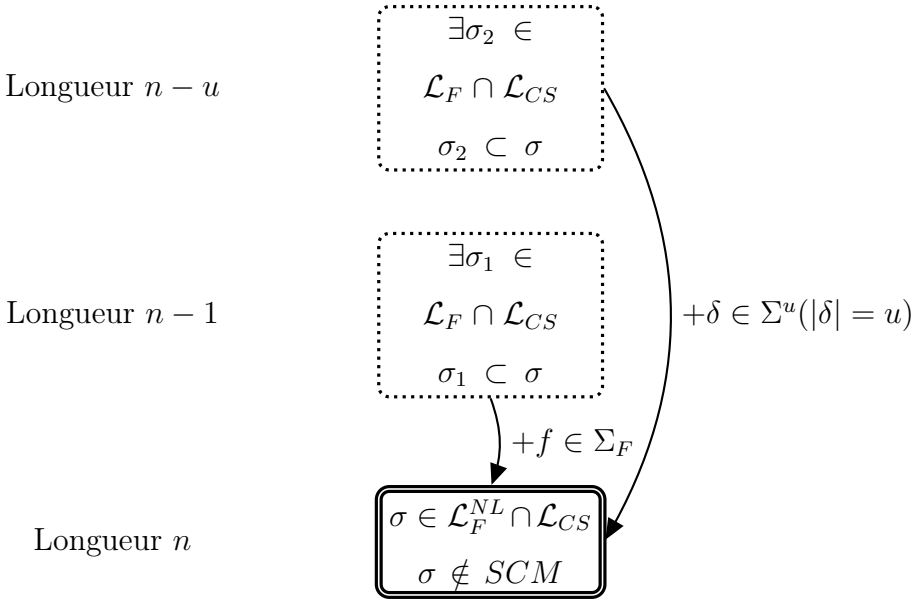


Figure 2.20 – Principe de calcul direct des séquences de coupe minimales.

pliquée aux définitions 2.6 et 2.31). La situation est alors représentée par la figure 2.20. Une séquence de coupe minimale ne peut pas être générée à l'aide d'une séquence de taille inférieure, ni par ajout d'un événement de défaillance, ni par distribution des événements d'une autre séquence sans modification de l'ensemble des états défaillants.

Le calcul direct des séquences de coupe minimales est, comme pour le calcul du noyau, réalisé par l'algorithme 3 par modification des entrées de l'algorithme. Les nouvelles entrées sont respectivement les intersections des ensembles $\mathcal{L}_F^{NL}$ et $\mathcal{L}_F^R$ avec l'ensemble des séquences de coupe $\mathcal{L}_{CS}$. La première intersection définit l'ensemble dans lequel les séquences de coupe minimales sont recherchées et le second est l'ensemble des séquences pouvant potentiellement générer les séquences du premier ensemble à l'aide de $CR4$.

Dans le cadre d'une étude de fiabilité qualitative, où l'expert cherche à déterminer quelles sont les causes pouvant permettre, à partir de l'état initial, d'obtenir la première défaillance du système, le calcul direct des séquences de coupe minimales est à privilégier. Ce calcul, permettant de diminuer largement la complexité du problème (cf. 4.4), ne permet plus de garantir, dans certain cas, la conservation de la description de l'ensemble de toutes les défaillances du système. Si l'objectif de l'étude qualitative de défaillance est de décrire au mieux la défaillance du système tout en validant l'exhaustivité de cette représentation, le calcul du noyau est à privilégier.

2.8.3 Un exemple illustratif

La figure 2.21 rappelle l'automate fini donné en figure 2.6 qui a servi d'exemple illustratif tout au long de ce chapitre (3 composants réparables).

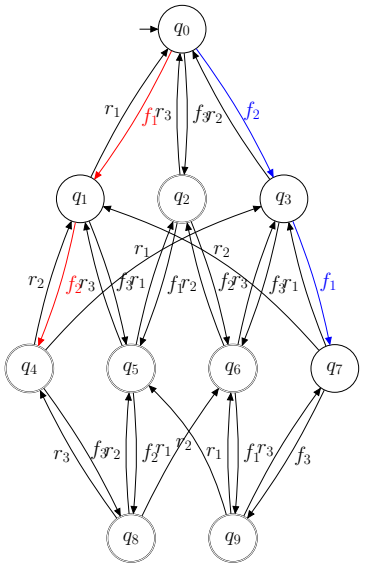


Figure 2.21 – Automate à 10 états, support de l'exemple

L'algorithme 1 appliqué à l'exemple de la figure 2.21 permet l'obtention de 161 séquences défaillantes ne passant pas deux fois dans un même état ($\mathcal{L}_F^{NL}$). Ces séquences sont de longueur comprise entre 1 et 9. Le calcul du langage défaillant est alors restreint aux séquences pouvant générer les séquences obtenues dans $\mathcal{L}_F^{NL}$. Les séquences recherchées sont alors de taille strictement inférieure à la plus grande des séquences de $\mathcal{L}_F^{NL}$. Le langage défaillant restreint $\mathcal{L}_F^R$ ainsi obtenu ne comporte alors que des séquences dont la longueur est au maximum $Max(|\sigma|, \sigma \in \mathcal{L}_F^{NL}) - 1$ et qui vaut ici $(9-1=8)$. L'algorithme 2 permet alors de calculer un ensemble de 5838 séquences dans le langage $\mathcal{L}_F^R$. Un synthèse des résultats obtenus sur l'automate à 10 états présenté en figure 2.21 par ces deux algorithmes est présentée dans le tableau 2.4.

Toutes les séquences de $\mathcal{L}_F^{NL}$ sont alors analysée par l'algorithme 3. Elles sont comparées aux séquences de $\mathcal{L}_F^R$ afin de déterminer si ces séquences font partie du noyau $Kern(\mathcal{L}_F)$. L'appartenance des séquences du noyau aux séquences de coupe sera ensuite analysée afin d'extraire du noyau $Kern(\mathcal{L}_F)$ l'ensemble SCM des séquences de coupes minimales. Les résultats de ces analyses sont présentés dans le tableau 2.5.

Tableau 2.4 – Nombre de séquences défaillantes du système à 10 états

Longueur	Nb séq. $\mathcal{L}_F^{NL}$	Nb séq. $\mathcal{L}_F^{R}$
1	1	1
2	5	5
3	8	13
4	16	50
5	22	121
6	34	455
7	40	1093
8	23	4100
9	12	/
>9	Aucune	
Total	161	5838

Il est possible de remarquer sur cet exemple que toutes les séquences du noyau sont des séquences de coupe minimales ($Kern(\mathcal{L}_F) = SCM$) et qu'aucune de ces séquences ne comporte d'événements de réparation. L'identité entre le noyau et l'ensemble des séquences de coupe minimales peut être expliquée par le fait que le comportement du système vis à vis des défaillances de composants n'est pas modifié après la défaillance de celui-ci. Par conséquent seule l'étude de la première défaillance à travers le calcul des séquences de coupes minimale est utile pour l'analyse qualitative du système.

2.9 Bilan.

Dans ce chapitre, une définition formelle des séquences de coupe minimales pour les système binaires, dynamiques et réparables a été proposée, moyennant les hypothèses suivantes :

- Les composants du système sont en nombre fini et indépendants du point de vue de leurs événements de défaillance et de réparation.
- Chacun des composants comporte un unique événement de défaillance et un unique événement de réparation.

Tableau 2.5 – Séquences du noyau du système à 10 états

Longueur	Séq. $\in \mathcal{L}_F^{NL}$	Séq. $\in Kern(\mathcal{L}_F)$	Séq. $\in SCM$
1	f_3	f_3	f_3
2	f_1f_2	f_1f_2	f_1f_2
	f_1f_3		
	f_3f_1		
	f_3f_2		
	f_2f_3		
3	$f_1f_3r_1$	Aucune	Aucune
	$f_1f_3f_2$		
	$f_3f_1f_2$		
	$f_3f_2f_1$		
	$f_2f_3r_2$		
	...		
>3	...	Aucune	Aucune
Total	161	2	2

– La défaillance du système peut être décrite par un automate à états fini déterministe ne possédant pas d'états équivalent dont l'alphabet d'entrée est composé de tous les événements de défaillance et de réparation provenant des composants.
– Un événement de défaillance ne peut masquer la défaillance du système.
– Un événement de réparation ne peut contribuer à la défaillance du système.

Les différents langages que nous avons été amené à définir et à manipuler sont présentés en figure 2.22. Dans cette figure, les ensembles de taille infinie sont soulignés.

Les langages $\mathcal{L}_D$ et $\mathcal{L}_F$ sont nécessaires et suffisants à la définition de la fonction de structure dynamique, ils permettent alors de définir l'état de défaillance du système en fonction des événements de défaillance et de réparation de ses composants. Ces deux langages sont :

– $\mathcal{L}_D$, le langage dysfonctionnel, dont les séquences décrivent tous les scénarios de défaillance et de réparation possibles dans le système. Dans le cas des systèmes

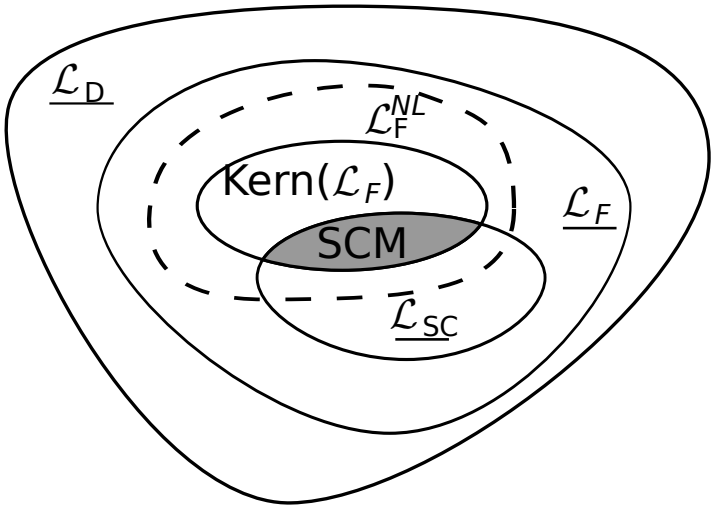


Figure 2.22 – Relations d'inclusion entre les différents langages et ensembles de séquences

réparables, ce langage est de taille infinie.

- $\mathcal{L}_F$, le langage défaillant, qui décrit parmi les séquences du langage défaillant celles qui terminent dans un état ou le système est défaillant. Dans le cas des systèmes réparables, ce langage est de taille infinie.

Dans le langage défaillant, le langage $\mathcal{L}_{CS}$ est le sous-ensemble des séquences de coupe. Pour les systèmes dynamiques et réparables, ce langage est de taille infinie. Les séquences de coupe sont le plus souvant de grand intérêt pour le modélisateur, car elle représentent les scénarios qui permettent d'arriver à la première défaillance du système depuis l'état initial.

Une définition de la cohérence des systèmes binaires dynamiques et réparables a ensuite été donnée. Cette définition a permis la création d'une représentation finie, suffisante pour caractériser des langages de taille infinie $\mathcal{L}_F$ et $\mathcal{L}_{CS}$. Les ensembles finis de séquences suivants ont alors été définis :

- $Kern(\mathcal{L}_F)$ est l'ensemble fini des séquences du langage défaillant permettant la représentation la plus compacte de tous les scénarios du langage défaillant d'un système cohérent. Il permet à l'aide de la règle 4 des systèmes cohérents de reconstruire l'ensemble du langage défaillant.
- SCM, l'ensemble des séquences de coupe minimales, est un sous ensemble de $Kern(\mathcal{L}_F)$ qui permet la reconstruction d'au minimum l'ensemble de toute les séquences de coupes $\mathcal{L}_{CS}$.

Les propriétés associées à chacun de ces deux ensembles finis de séquences à permis la création d'un algorithme pour le calcul de ces deux ensembles, après restriction au langage fini $\mathcal{L}_F^{NL}$ des séquences ne passant pas deux fois par un même état. La définition et l'utilisation de ce nouvel ensemble est nécessaire afin de pouvoir mettre en pratique les algorithmes développés pour le calcul du noyau et de l'ensemble des séquence de coupes minimales. Un premier exemple basique a ensuite été traité.

Jusqu'à présent, nous n'avons retenu qu'un modèle basique de défaillance des composants, ne comportant qu'un seul événement de défaillance et un seul événement de réparation. De manière à montrer la généricité des concepts, formalismes et algorithmes proposés dans ce chapitre, et afin de permettre la modélisation de comportements plus sophistiqués, nous allons maintenant considérer que l'analyse des risques à été conduite à l'aide des Boolean logic Driven Markov Processes (BDMP). Les BDMP sont un modèle de défaillance dynamique qui intègrent de manière native la réparabilité et qui proposent plusieurs modèles assez sophistiqués de défaillance pour les composants.

Chapitre
3
Séquences de Coupe Minimales d'un système de composants multimodes, application aux BDMP.

Sommaire

3.1 Introduction. 103
3.2 Modélisation de la sûreté d'un système par BDMP. 103
3.2.1 Hypothèses restrictives sur le modèle BDMP. 105
3.2.2 Analyse sémantique des BMDP. 106
3.3 Génération de l'automate fini "équivalent" à un BDMP. . . . 115
3.3.1 Description d'un BDMP par un automate fini. 115
3.3.2 Mécanisme de génération de l'automate. 116
3.4 Calcul des séquences de coupe minimales d'un système à composants multimodes. 120
3.4.1 Composants multimodes, cas des BDMP. 121
3.4.2 Séquences d'un système à composants multimodes. 122
3.4.3 Prise en compte des modes de composants pour le calcul du noyau. . . . 122
3.5 Résultats sur le système à trois pompes. 126
3.6 Bilan. 128

3.1 Introduction.

L'étude qualitative des défaillances d'un système dynamique réparable par Boolean logic Driven Markov Processes (BDMP) est intéressante car les BDMP ont été conçus pour modéliser les systèmes réparables de manière naturelle et native. Une extension de nos travaux permettant la prise en compte de différents modes de fonctionnement pour les composants doit cependant être présentée. Ce chapitre est composé de deux sections. La première section (3.2) propose une analyse sémantique des BDMP qui permet la proposition d'un algorithme de génération d'un "automate fini équivalent" en section 3.3. Cet automate permet une représentation explicite des scénarios dysfonctionnels décrits par un BDMP. La section 3.4 propose ensuite une évolution du cadre formel et des algorithmes de calcul présentés dans le chapitre précédent, pour la prise en compte des composants multimodes.

3.2 Modélisation de la sûreté d'un système par BDMP.

Dans l'objectif d'illustrer d'une part les travaux de formalisation des Boolean logic Driven Markov Processes (BDMP) et d'autre part l'extraction des Séquences de Coupes Minimales sur ces modèles, un exemple, volontairement simple, a été retenu. Il est présenté en figure 3.1.

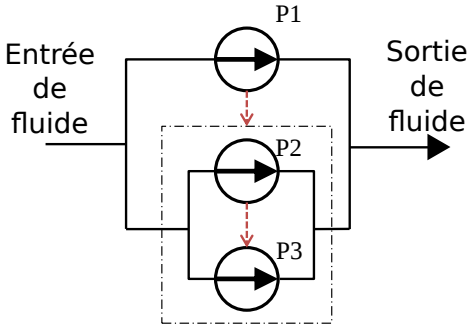


Figure 3.1 – Exemple support pour l'analyse d'un BDMP

Le système considéré est une station de pompage constituée de trois pompes. La fonction principale ce système est de fournir en fluide la sortie à partir de l'entrée grâce à l'utilisation de différentes pompes. Le système est composé de deux parties : la pompe

principale $P1$ et le groupe de pompe $P2 + P3$ en secours. Deux niveaux de redondance sont possibles. Le groupe de pompe $P2 + P3$ est en redondance passive de la pompe $P1$ et la pompe $P3$ en redondance passive de la pompe $P2$.

Le fonctionnement du système est le suivant :

- En situation normale, seule la pompe $P1$ est en fonctionnement, les autres pompes sont à l'arrêt.
- Lorsque la pompe $P1$ est défaillante, le groupe de secours $P2+P3$ est sollicité. Dans ce groupe de pompes seule $P2$ est mise en marche afin de réaliser la distribution de fluide.
- Lorsque les pompes $P1$ et $P2$ sont défaillantes, la pompe $P3$ est démarrée.
- Lorsque les trois pompes sont défaillantes, le système est défaillant.
- Lorsque les pompe $P1$ et $P2$ sont défaillantes, dès que la pompe P2 est réparée, le sous-système redondant revient en situation normale où $P2$ est la seule pompe en fonctionnement.
- Dès que la pompe $P1$ est réparée, le système revient en situation normale où $P1$ est la seule pompe en fonctionnement.

Nous ferons les hypothèses suivantes :

- Les pompes P1 et P2 ne peuvent défaillir que lorsqu'elles sont en fonctionnement.
- La pompe P3 peut défaillir en mode actif (en fonctionnement) et en mode dormant (à l'arrêt).
- Toutes les pompes peuvent être réparées, quel que soit le mode dans lequel elles se trouvent.

Le BDMP (dont la syntaxe a été présentée dans le chapitre 1) représentant l'ensemble des défaillances possibles de ce système est présenté en figure 3.2. Composé des 3 *feuilles*, représentant les trois pompes, ce BDMP représente l'ensemble de tous les scénarios de défaillance et de réparation possibles dans le système ainsi que leurs effets sur la défaillance du système. La défaillance du système complet est représentée par le *top event te*, le système est défaillant si la pompe $P1$ et si le sous-système de secours, représenté par la porte Def_{Sec}, sont défaillants. Un premier niveau de redondance est présent entre ces deux éléments, il est représenté par la *gâchette* ayant pour origine la feuille modélisant la pompe $P1$ et pour destination la porte représentant la défaillance du système de secours.

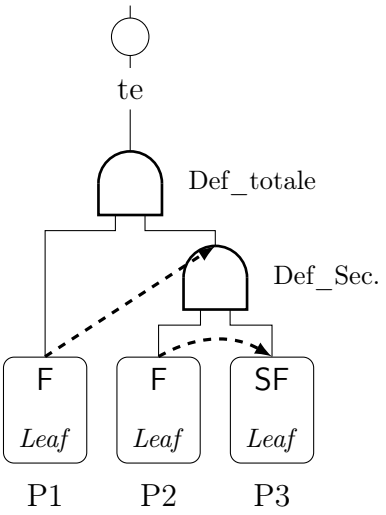


Figure 3.2 – Modèle BDMP du système à trois pompes

Ce sous-système de secours, représenté par la porte Def_{Sec} est défaillant si les pompes $P2$ et $P3$ sont défaillantes. Une deuxième redondance est présente dans ce sous-système entre la pompe $P2$ et la pompe $P3$, elle est représentée par une gâchette entre les deux feuilles modélisant ces deux composants.

3.2.1 Hypothèses restrictives sur le modèle BDMP.

Deux hypothèses restrictives sont faites, consistant à limiter notre étude à une sous-classe de BDMP sans porte PAND et sans composants pouvant défaillir à la sollicitation.

H1 : La structure d'arbre de défaillance du BDMP est statique. Cette hypothèse interdit l'utilisation des portes de type PAND dans la structure d'arbre de défaillance. Cette structure est alors statique, cela qui permet de reporter l'ensemble de la dynamique du modèle sur les feuilles du BDMP. Cette hypothèse permet de garantir que le modèle construit par le modélisateur est cohérent. L'utilisation de porte PAND peut en effet conduire à une modélisation où le système considéré est non défaillant alors que tous ses composants le sont ; un exemple d'un tel modèle utilisant une porte PAND dans un arbre de défaillance dynamique est présent dans [BD05].

H2 : Il est fait abstraction de la défaillance à la sollicitation. Cette hypothèse restreint les BDMP à une sous-classe n'utilisant que les feuilles F et SF. Les défaillances à la sollicitation étant provoquées par le changement de mode du composant associé (lui-même provoqué par la défaillance d'autres composants). L'utilisation des feuilles de type

I ne permet plus de garantir l'indépendance entre composants en termes d'événements, car un événement de défaillance à la sollicitation est, s'il occurre, simultané à un événement de défaillance d'un second composant qui provoque la mise en marche du composant considéré. Afin de vérifier l'hypothèse d'indépendance événementielle entre composants, nécessaire pour l'utilisation des résultats du chapitre 2, la feuille I n'est pas utilisée. Cette hypothèse a également pour conséquence que chacun des composants du système est modélisé par une feuille unique, car la modélisation d'un composant pouvant défaillir à la sollicitation et dans son mode actif ou dormant, nécessite l'utilisation d'au moins deux feuilles (une pour la défaillance à la sollicitation et une pour la défaillance en mode actif/dormant).

L'adoption de ces deux hypothèses limite donc l'étude aux BDMP ne comportant que des feuilles de type F et SF et n'utilisant que des portes statiques AND et OR, les portes k/N ne sont pas considérées dans cette étude car elles peuvent être représentées à l'aide d'un assemblage de portes AND et OR.

3.2.2 Analyse sémantique des BMDP.

L'objectif de cette partie est de préciser la sémantique des BDMP de manière à en proposer une traduction systématique sous forme d'un automate fini équivalent, dans l'objectif de réaliser des études qualitatives. Un BDMP est composé de quatre types d'éléments (Top Event, Porte, Feuilles et gâchettes), il peut donc se décrire à l'aide du 4-uplet suivant :

$$< L, G, T, te >$$

L'ensemble $L = \{L_i\}$ est l'ensemble des feuilles du BDMP (représentant les composants du système). Celles-ci peuvent être de deux natures :

- les feuilles de type F, décrivant le comportement d'un composant dont la défaillance ne peut occurrer que lorsqu'il est dans un mode actif ; elles sont placées dans l'ensemble L_F
- les feuilles de type SF dont la défaillance peut aussi bien occurrer dans leur mode actif que dans leur mode dormant ; elles sont placées dans l'ensemble L_{SF}.

L'ensemble L peut donc être décrit par en $L = L_F \bigcup L_{SF}$.

L'ensemble $G = \{G_i\}$ est l'ensemble des portes statiques du BDMP. Puisque ces portes peuvent être de deux types (OR ou AND), elles sont respectivement placées dans les ensembles G_{OR} et G_{AND} ($G = G_{OR} \bigcup G_{AND}$).

L'ensemble des noeuds d'un BDMP $N = \{N_i\}$ peut maintenant être défini comme l'union des portes et des feuilles, $N = G \bigcup L$. Une extension de cette définition peut être réalisée en ajoutant le Top Event te à l'ensemble des noeuds N afin d'obtenir l'ensemble $N^+ = N \bigcup \{te\}$. L'état de chacun des noeuds du BDMP est décrit par un couple de variables booléennes (F_{N_i}, M_{N_i}), F_{N_i} étant l'état de défaillance du noeud (*Vrai* si le noeud est défaillant) et M_{N_i} son mode (*Vrai* si le noeud est dans un mode actif et *Faux* si le noeud est en mode dormant). La définition de l'ensemble des noeuds permet notamment de décrire les entrées d'une porte G_i comme un ensemble $Inputs(G_i)$ d'au minimum deux noeuds, pouvant être indifféremment des portes ou des feuilles ($\forall G_i \in G, Inputs(G_i) = \{N_k | N_k \in N\}, Card(Inputs(G_i)) \geq 2$).

L'ensemble $T = \{T_i\}$ est l'ensemble des gâchettes du BDMP. Chaque gâchette T_i possède un noeud origine nommé $Orig(T_i) = N_j, N_j \in N$ et un noeud destination nommé $Dest(T_i) = N_k, N_k \in N$. Une gâchette T_i est définie comme un couple de deux noeuds différents $(Orig(T_i), Dest(T_i)) \in N^2, Orig(T_i) \neq Dest(T_i))$.

Le Top Event te est la représentation de la défaillance du système modélisé par le BDMP. Ce top event ne possède qu'une seule entrée : son ensemble d'entrées $Inputs(te)$ est donc définit par $Inputs(te) = \{N_k | N_k \in N\}, Card(Inputs(te)) = 1$.

Conformément à la sémantique qui vient d'être formulée, l'ensemble des données à extraire du BDMP de la figure 3.2 est présenté dans les tableaux 3.1 et 3.2. Le tableau 3.1 décrit chacun des noeuds $N_i \in N^+$ en donnant son ensemble de référence (L_F, L_{SF}, G_{AND} ou G_{OR}) ainsi que l'ensemble de ses entrées $Inputs(N_i)$. Le tableau 3.2 présente les données relatives aux gâchettes $T_i \in T$ précisant leur origine ($Orig(T_i)$) et leur destination ($Dest(T_i)$).

L'utilisation de ces données va maintenant permettre de relier la structure du BDMP

N_i	Catégorie	Entrées ($Inputs(N_i)$)
te	/	Def_totale
Def_totale	G_{AND}	P1, Def_Sec.
Def_Sec.	G_{AND}	P2, P3
P1	L_F	/
P2	L_F	/
P3	L_{SF}	/

Tableau 3.1 – Propriétés des noeuds $N_i \in N^+$ du BDMP de la figure 3.2

T_i	Origine et destination $((Orig(T_i), Dest(T_i)))$
T_1	(P1,Def_Sec.)
T_2	(P2, P3)

Tableau 3.2 – Propriétés des gâchettes $T_i \in T$ du BDMP de la figure 3.2

au calcul de l'état de chacun de ses noeuds représentés par le couple de variables booléennes (F_{N_i}, M_{N_i}), qui représentent respectivement l'état de défaillance et le mode de fonctionnement du noeud (actif ou dormant). La partie 3.2.2.1 va décrire le calcul de la défaillance de chacun des noeuds et la partie 3.2.2.2 le calcul de son mode. La partie 3.2.2.3 décrit le comportement des feuilles du BDMP par automate fini.

3.2.2.1 Description logique de la structure d'arbre de défaillance.

La structure d'arbre de défaillance d'un BDMP (construit en utilisant les feuilles comme les événements de base de l'arbre) relie la défaillance du système (F_{te}) aux défaillances élémentaires (F_{L_i}) provenant des feuilles L_i de L ($L_i \in L$). Cette relation est définie en utilisant des portes AND et OR. Seuls les états de défaillance des portes peuvent être calculés à l'aide de la structure car les états de défaillance des feuilles résultent des défaillances et réparations subies par les composants modélisés.

Chacune des portes $G_i \in G$ possède donc un état de défaillance qui est calculé suivant les propositions suivantes :

– Pour les portes OR, la valeur de leur état de défaillance est définie par :

$$F_{G_i} = \bigvee_{N_j \in Inputs(G_i)} F_{N_j}$$

– Pour les portes AND, la valeur de leur état de défaillance est définie par :

$$F_{G_i} = \bigwedge_{N_j \in Inputs(G_i)} F_{N_j}$$

La structure acyclique de l'arbre de défaillance permet de déterminer de façon univoque la relation entre la défaillance des composants et la défaillance du système. Cette fonction est déterminée en partant de la racine de l'arbre (top event) en propageant le calcul vers le niveau le plus bas (feuilles de l'arbre). Le résultat de ce calcul est la fonction de structure statique qui lie la défaillance du top event (F_{te}) aux défaillances des feuilles $\{F_{L_i} | L_i \in L\}$. Pour notre exemple de la figure 3.2 ce calcul donne :

$$\begin{aligned} F_{te} &= F_{Def_Totale} \\ &= F_{P1} \wedge F_{Def_{Sec.}} \\ &= F_{P1} \wedge (F_{P2} \wedge F_{P3}) \end{aligned}$$

3.2.2.2 Description logique de la structure de gâchette.

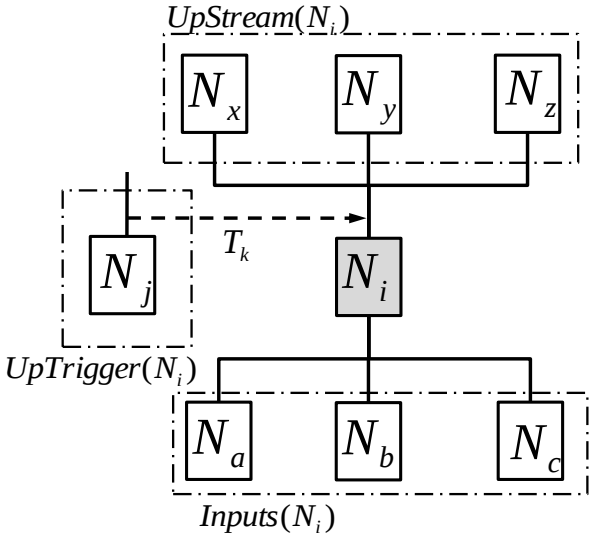


Figure 3.3 – Situation d'un noeud $N_i \in N$ d'un BDMP

Le calcul du mode de chacun des noeuds du BDMP a pour objectif de déterminer la relation entre le mode de chacune des feuilles M_{L_i} en fonction des états de défaillance F_{L_j}.

Ce calcul passe par l'utilisation des gâchettes. Une gâchette $T_i \in T$ est la description d'une redondance entre les sous-systèmes ou composants. Lorsque le sous-système à l'origine de la gâchette est défaillant (N_j fig. 3.3), il autorise le passage en mode actif du sous-système redondant, placé sous la destination de la gâchette (N_i fig. 3.3). Si celui-ci est utilisé par un sous-système actif (N_x ou N_y ou N_z fig. 3.3), il devient alors actif. La description du calcul de mode est faite pour un noeud N_i de N, dont la situation est présentée en figure 3.3.

Afin de préciser la situation du noeud N_i, deux fonctions supplémentaires sont nécessaires :

- La fonction $UpStream(N_i)$ définit l'ensemble des noeuds qui utilisent N_i comme entrée dans la structure d'arbre de défaillance du BDMP. Ces noeuds possèdent au minimum une entrée, ce sont donc des portes ou le top event. Cette fonction est définie par :

$$UpStream(N_i) = \{N_j \in G \cup \{te\} | N_i \in Inputs(N_j)\}$$

  Sur la figure 3.3, $UpStream(N_i) = \{N_x, N_y, N_z\}$.
- La fonction $UpTrigger(N_i)$ définie le noeud origine de la gâchette ayant comme destination N_i, si elle existe. Ce noeud origine (unique, car il n'y a au maximum qu'une gâchette ayant pour destination N_i) peut être soit une porte soit une feuille. Cette fonction est donc définie par :

$$UpTrigger(N_i) = \{Orig(T_j) \in N | Dest(T_j) = N_i\}, Card(UpTrigger(N_i)) \leq 1$$

  Sur la figure 3.3, $UpTrigger(N_i) = \{N_j\}$.

Le calcul du mode de chacun des noeuds N_i de N va alors dépendre de sa situation :

- Cas 1 : Le noeud n'est ni une entrée de porte ni une destination de gâchette (mais il peut en être l'origine) ($UpStream(N_i) = \emptyset \wedge UpTrigger(N_i) = \emptyset$).

$$M_{N_i} = 0$$

- Cas 2 : Le noeud est l'entrée d'au moins une porte, mais n'est la destination d'aucune gâchette ($UpStream(N_i) \neq \emptyset \wedge UpTrigger(N_i) = \emptyset$).

$$M_{N_i} = \vee_{G_j \in UpStream(N_i)} M_{G_j}$$

– Cas 3 : Le noeud est l'entrée d'au moins une porte et la destination d'une gâchette ($UpStream(N_i) \neq \emptyset \wedge UpTrigger(N_i) \neq \emptyset$).

$$M_{N_i} = \left(\vee_{G_j \in UpStream(N_i)} M_{G_j} \right) \wedge F_{N_k}, N_k \in UpTrigger(N_i)$$

– Cas 4 : Le noeud n'est l'entrée d'aucune porte, mais est la destination d'une gâchette ($UpStream(N_i) = \emptyset \wedge UpTrigger(N_i) \neq \emptyset$).

$$M_{N_i} = F_{N_k}, N_k \in UpTrigger(N_i)$$

Le mode du top event est quant à lui toujours $Vrai$ car il représente le mode du système considéré, qui est supposé toujours actif (pour les besoins de l'étude), On a alors :

$$M_{te} = 1$$

Afin de déterminer le mode M_{L_i} de chacune des feuilles, l'ensemble des modes des noeuds M_{N_i} du BDMP doit être calculé en partant de la racine (top event) et propagé vers le niveau le plus bas (les feuilles). La forme acyclique de la structure d'arbre de défaillance du BDMP garantit la calculabilité de ces modes. Le résultat est un ensemble d'équations combinatoires : $\forall L_i \in L, M_{L_i} = f(F_{L_j}, L_j \in L)$ qui relient le mode de chacune des feuilles aux états de défaillances de celles-ci. L'application de ce calcul pour le BDMP de la figure 3.2 donne :

$$\begin{aligned} M_{P3} &= M_{Def_Sec.} \wedge F_{P2} \\ &= (M_{Def_totale} \wedge F_{P1}) \wedge F_{P2} \\ &= (M_{te} \wedge F_{P1}) \wedge F_{P2} \\ &= (Vrai \wedge F_{P1}) \wedge F_{P2} \\ &= F_{P1} \wedge F_{P2} \end{aligned}$$

Soit pour l'ensemble des feuilles du BDMP de la figure 3.2 :

$$\begin{aligned} M_{P1} &= Vrai \\ M_{P2} &= F_{P1} \\ M_{P3} &= F_{P1} \wedge F_{P2} \end{aligned}$$

3.2.2.3 Description événementielle du comportement défaillant des feuilles d'un BDMP.

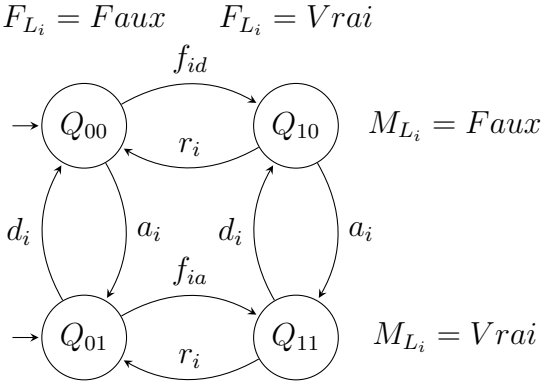


Figure 3.4 – Envelope du comportement d'une feuille L_i.

Le comportement des feuilles $L_i \in L$ du BDMP peut être décrit par un automate fini. Cet automate évolue sur occurrence de deux types d'événements : les événements de changement de mode (contenus dans l'alphabet Σ_M) et les événements de défaillance et de réparation (contenus respectivement dans les alphabet Σ_F et Σ_R). Les événements de changement de mode traduisent le changement de comportement de la feuille lors de la variation de son mode de fonctionnement. Les événements de défaillance et de réparation sont les événements générés par la feuille et représentent la faute et la réparation du composant modélisé. L'automate fini dont l'alphabet complet est $\Sigma_M \cup \Sigma_F \cup \Sigma_R$, et représentant l'enveloppe des comportements des feuilles F et SF, est représenté en figure 3.4. Cet automate représente le comportement d'une feuille SF où la défaillance en mode dormant est possible. Le modèle de la feuille F où cette défaillance n'est pas possible est obtenu en retirant la transition $\delta(Q_{00}, f_{id}, Q_{10})$ (allant de l'état Q_{00} à l'état Q_{10} sur occurrence de f_{id}).

Cet automate est constitué de quatre états qui permettent de représenter toutes les combinaisons possibles liant l'état de défaillance (défaillant, non défaillant) et le mode de fonctionnement (mode actif, mode dormant) du composant. Le choix de l'état initial entre les états Q_{00} et Q_{01} est effectué en fonction du mode initial de la feuille, déterminé lors de l'initialisation du modèle. L'état Q_{00} est l'état initial pour une feuille en mode dormant à l'initialisation et l'état Q_{01} est initial lorsque la feuille est en mode actif à l'initialisation.

Les transitions entre les états de cet automate sont étiquetées par les événements de

défaillance et de réparation appartenant à l'alphabet $\Sigma_F \cup \Sigma_R$ ou par les événements de changement de mode (placés dans Σ_M) :

- f_{id} est l'événement de défaillance en mode dormant du composant modélisé.
- f_{ia} est l'événement de défaillance en mode actif du composant modélisé.
- r_i est l'événement de réparation qui peut occurrer dans chacun des deux modes.
- a_i est l'événement de passage du mode dormant au mode actif.
- d_i est l'événement de passage du mode actif au mode dormant.

Alors que les évolutions dues aux défaillances et aux réparations du composant modélisé sont spontanées, les événements de changement de mode sont générés par le changement de valeur de la variable M_{L_i} représentant le mode de la feuille (actif ou dormant). Ils permettent de refléter les variations du mode, calculé par la fonction combinatoire $M_{L_i} = f(F_{L_i}, L_i \in L)$ selon la méthode explicitée en 3.2.2.2, sur le modèle automate fini.

3.2.2.4 Représentation d'un BDMP.

Finalement, à l'issue de l'analyse de la sémantique des BDMP, une nouvelle représentation est possible. Cette représentation, illustrée en figure 3.5, peut être décomposée en trois parties.

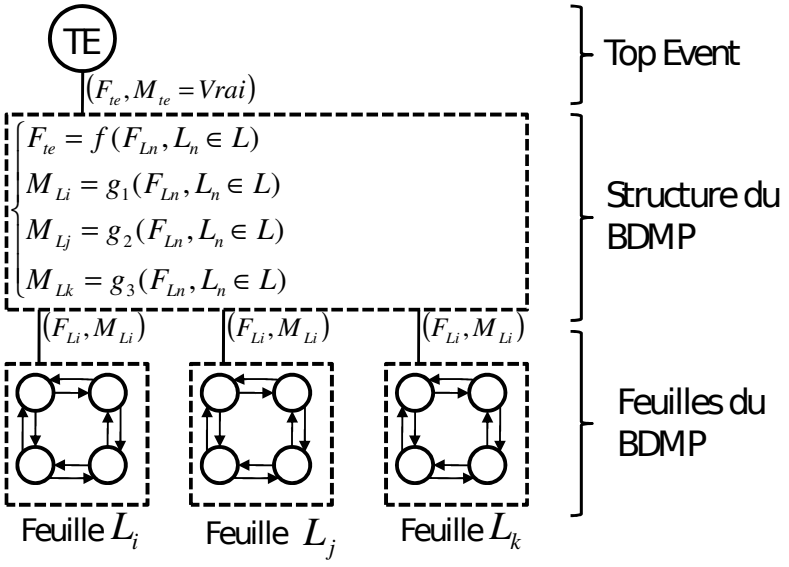


Figure 3.5 – Structure de la représentation d'un BDMP.

Le Top event est la représentation de l'état de défaillance du système, il est associé au couple de variables $(F_{te}, M_{te} = 1)$. La variable F_{te} est calculée via la structure en utilisant les états de défaillance des feuilles. La variable M_{te}, toujours vraie, va permettre l'activation des différents sous-systèmes, permettant le calcul de chacun des modes des feuilles.

La structure du BDMP (arbre de défaillance et gâchettes) est représentée à l'aide d'un système d'équations combinatoires. Ce système est composé de $n + 1$ équations avec n le nombre de feuilles ($n = Card(L)$). Les équations qui le constituent sont les suivantes :

- Une équation $F_{te} = f(F_{L_i}), L_i \in L$ qui permet le calcul de la défaillance du Top Event en fonction des états de défaillance des feuilles.
- n équations $M_{L_i} = g_i(F_{L_i}, M_te), L_i \in L$ qui permettent le calcul du mode de chacune des feuilles en fonction des états de défaillance des feuilles.

Les feuilles du BDMP sont représentées chacune par un automate fini. Ces automates permettent d'actualiser les états de défaillance F_{L_i} en fonction des événements de défaillance ou de réparation intervenant sur chacun des composants. Elles évoluent également en fonction des variations du mode M_{L_i} associé à chacune d'elles. La structure du BDMP étant purement combinatoire, elles permettent la description de la dynamique des BDMP. L'information du type de feuille (F ou SF) est portée par la forme de l'automate dont l'état initial est déterminé en fonction de la valeur de leur mode à l'initialisation du BDMP.

Le calcul des évolutions du BDMP est régi par la dynamique suivante :

1. Initialisation du BDMP : En situation initiale toutes les feuilles du BDMP sont dans un des états non défaillants : les variables de défaillance F_{L_i} des feuilles sont toutes à 0. Les modes initiaux de chacune des feuilles sont calculés à l'aide de la structure, permettant de définir l'état initial de chacun des automates de feuille.
2. L'évolution du BDMP est actualisée sur occurence d'un événement de défaillance ou de réparation provenant d'un des automates de feuilles.
3. Les valeurs des états de défaillance F_{L_i} sont alors mises à jour en fonction de l'état actif dans chacun des automates de feuille.
4. Les modes M_{L_i} de chacune des feuilles, ainsi que la valeur de défaillance du top

event F_{te}, sont mis à jour.

5. Les événements de changement de mode sont générés afin que chacun des automates de feuille soit dans un état reflétant le mode M_{L_i} courant pour celle-ci.

6. Retour à l'étape 2.

A partir de cette représentation, il semble possible de traduire les BDMP dans un cadre algébrique proposant une sémantique plus riche tels les automates de mode [Rau02], permettant alors de nombreuse possibilité d'exploitation de ce modèle. Ces travaux utilisent les BDMP uniquement dans l'objectif d'obtenir une représentation automate fini de la fonction de structure d'un système dynamique réparable de composants multimode, la section suivante propose donc un moyen d'obtenir cette représentation sans introduire un nouveau cadre algébrique spécifique dont la richesse n'est pas, içi, nécessaire.

3.3 Génération de l'automate fini "équivalent" à un BDMP.

3.3.1 Description d'un BDMP par un automate fini.

Après avoir défini une représentation formelle des BDMP, il est nécessaire de générer l'automate fini représentant le langage dysfonctionnel et le langage défaillant, implicitement décrits par un BDMP. Ces deux langages sont en fait le langage généré ($\mathcal{L}_D$) et le langage marqué ($\mathcal{L}_F$) par cet automate. De ce fait, et par abus de langage, nous parlerons d'automate fini équivalent à un BDMP, cette équivalence n'étant pas formellement démontrable.

Cet automate est défini par le 5-uplet (2.3.1) : $< \Sigma, Q, q_0, Q_M, \delta >$ avec dans chacun de ces ensembles :

- L'alphabet Σ qui est composé de tous les événements de défaillance (f_{id},f_{ia}) et de réparation (r_i) provenant des feuilles du BDMP. Il est donc défini par $\Sigma = \bigcup_{L_i \in L} \Sigma_F^{L_i} \cup \Sigma_R^{L_i}$.
- Chaque état $q \in Q$ de l'automate représente un état du BDMP. Un état du BDMP est uniquement défini par l'état de défaillance F_{L_i} de chacune de ses feuilles. Un état de l'automate est alors associé à un vecteur $\mathbf{F}_{q_i}$, représentation d'un état du BDMP, contenant la valeur des variables de défaillance des feuilles dans l'état du

BDMP considéré. Le nombre d'états de l'automate sera donc égal à $2^{Card(L)}$, nombre de combinaisons possibles pour les valeur du vecteur $\mathbf{F}_{q_i}$. Les états du BDMP sont stockés dans l'ensemble $S\mathbf{F}$.

- L'état initial de l'automate est défini par l'état initial du BDMP. Par hypothèse, dans l'état initial, aucune des feuilles du BDMP n'est dans un état de défaillance. L'état initial q_0 de l'automate est donc associé à l'état du BDMP $\mathbf{F}_{q_0}$ dont toutes les composantes sont nulles.
- L'ensemble des états marqués de l'automate représente les états du BDMP où le système modélisé est défaillant. Ils correspondent donc aux états où le Top event est défaillant ($F_{te} = 1$).
- La fonction de transition δ représente les évolutions possibles entre les états $q \in Q$ de l'automate (donc du BDMP) sur occurence d'un événement de l'alphabet Σ.

L'alphabet Σ, ainsi que l'état initial q_0 sont facilement déductibles du BDMP. Le mécanisme de génération va donc calculer l'ensemble des états Q, des états marqués Q_M et la fonction de transition δ.

3.3.2 Mécanisme de génération de l'automate.

La génération de l'automate consiste à parcourir l'espace des états du BDMP pour générer l'ensemble des états Q, des états marqués Q_M et la fonction de transition de l'automate. Cette exploration part de l'état initial du BDMP et explore les possibilités d'évolutions dans le BDMP pour générer les éléments de l'automate. Ce mécanisme est réalisé par l'algorithme 4.

Les entrées de l'algorithme 4 sont :

- La liste des feuilles L du BDMP.
- L'alphabet Σ des événements de défaillance et de réparation associés à chacune des feuilles du BDMP.
- Chacun des automates de feuilles (décrits dans la partie 3.2.2.3), Caractérisé par deux fonctions :
  - Une fonction $\Delta_{L_i} : (F_{L_i}, M_{L_i}, u \in \Sigma) \rightarrow F_{L_i}$ permettant le calcul de l'effet d'un événement sur la défaillance de la feuille.

Algorithme 4 Génération des états et transitions de l'AF équivalent à un BDMP

Require: $L = \{L_i\}$: Ensemble des feuilles du BDMP, $\Sigma = \Sigma_f \bigcup \Sigma_r$: Alphabet des événements de défaillance et réparation, $\{(\Delta_{L_i}, \Theta_{L_i})\}$: représentation fonctionnelle des automates de feuilles $(F_{te}, \{M_{Li}\})$ Système d'équations combinatoires représentant la structure du BDMP.

1: # *G*énération de l'état initial.
2: $\mathbf{F}_{q0} \leftarrow [F_{L_k} = False | L_k \in L]$
3: $S\mathbf{F} \leftarrow \mathbf{F}_{q0}$
4: $Q \leftarrow q_0$
5: **for** each q_i in Q **do**
6: # *E*tape 1 : Calcul des Modes M_{L_k}.
7: $M_{L_k} = M_{L_k}(\mathbf{F}_{qi})$
8: # *E*tape 2 : Calcul des événements possibles (placés dans SE) dans l'état courant q_i.
9: $SE = \cup_{L_k \in L} \Theta_{L_k}(F_{L_k}, M_{L_k})$
10: # *E*tape 3 : Génération des successeurs q_n pour chaque événement u possible
11: **for** each $u \in SE$ **do**
12: $\mathbf{F}_{qn} = [\Delta_{L_k}(F_{L_k}, M_{L_k}, u) | L_k \in L]$
13: **if** $\mathbf{F}_{qn} \notin S\mathbf{F}$ **then**
14: $S\mathbf{F} \leftarrow \mathbf{F}_{qn}$
15: $Q \leftarrow q_n$
16: # *T*est du marquage de l'état q_n généré en fonction du Top Event.
17: **if** $F_{te}(\mathbf{F}_{q_n}) = True$ **then**
18: $Q_M \leftarrow q_n$
19: **end if**
20: **end if**
21: $\delta \leftarrow < q_i, u, q_n >$
22: **end for**
23: **end for**
24: **return** $< \Sigma, Q, q_0, Q_m, \delta >$

- Une fonction $\Theta_{L_i} : (F_{L_i}, M_{L_i}) \rightarrow \Sigma^{L_i}$ qui détermine les événements possibles dans l'automate à partir de l'état défini par le couple (F_{L_i}, M_{L_i}).

– Les équations combinatoires associées à la structure du BDMP permettant le calcul des modes $M_{L_i}, L_i \in L$ des feuilles du BDMP et de la défaillance du système (représenté par le niveau de défaillance du top event F_{te}).

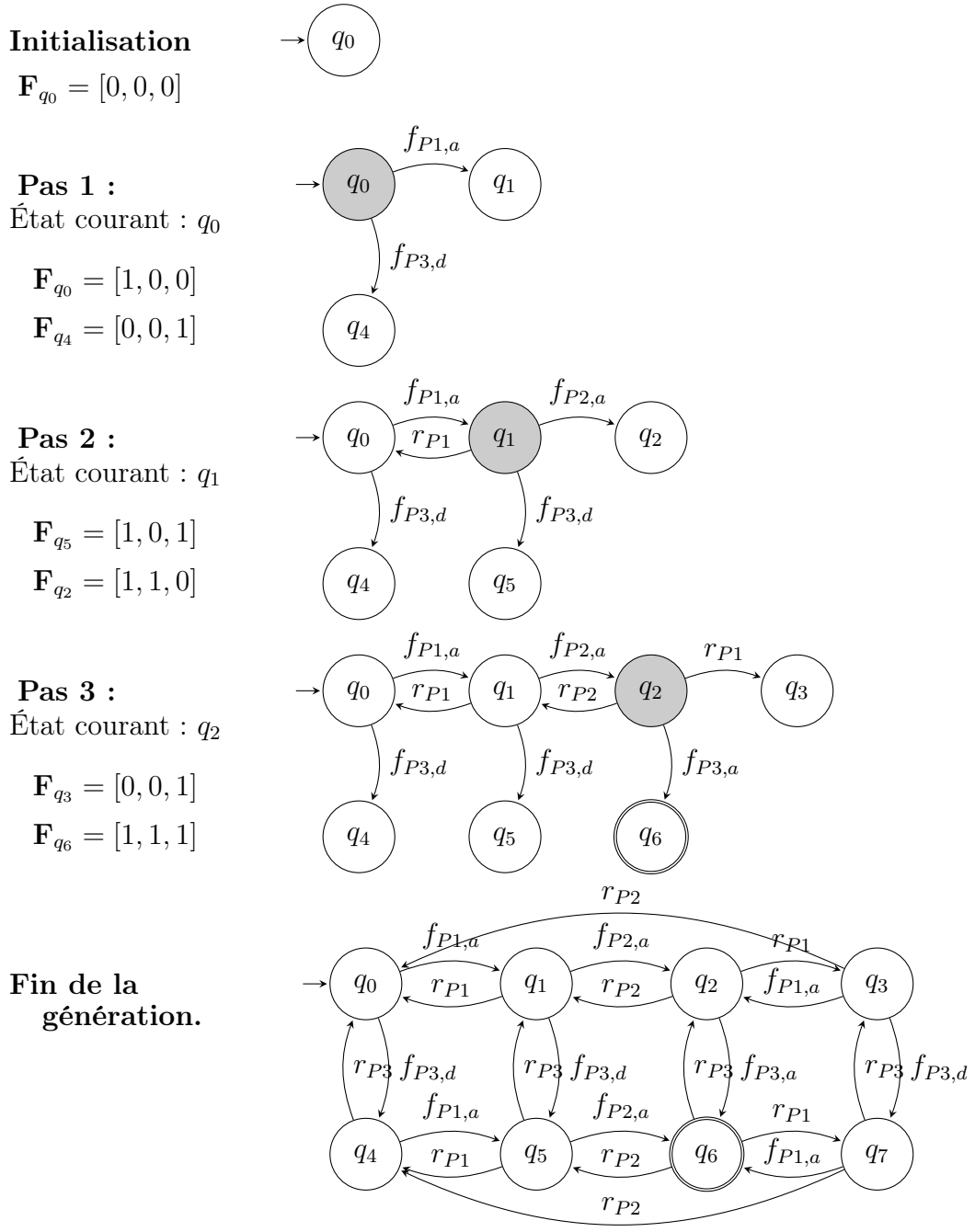


Figure 3.6 – Phases de calcul de l'automate fini equivalent au BDMP de la figure 3.2.

La première étape de la génération de l'automate (Alg4,1-4) consiste à créer l'état

initial q_0, départ de cette génération. Cette création consiste à placer un état q_0 dans Q et lui associer l'état du BDMP $\mathbf{F}_{q0}$. Toutes les composantes de l'état du BDMP $\mathbf{F}_{q0}$ sont nulles, cet état est placé dans l'ensemble des états du BDMP $S\mathbf{F}$.

A partir de l'état courant q_i ($q_i = q_0$ pour le premier pas), les modes M_{L_k} des feuilles sont mis à jour (Alg4,7). Chaque couple (F_{L_k}, M_{L_k}) est alors connu, l'état actif de chacune des feuilles peut donc être déterminé. La connaissance des états courants des feuilles permet de déterminer quelles sont les évolutions possibles à partir du modèle automate de feuilles présenté en partie 3.2.2.3. Les événements correspondants, calculés à l'aide de chacune des fonctions $\Theta_{L_i} : (F_{L_i}, M_{L_i})$ sont placés dans l'ensemble SE (Alg4,20).

Pour chacun des événements possibles (Alg4,11) de SE, l'état suivant du BDMP (représenté par le vecteur $\mathbf{F}_{q_n}$) est calculé. Deux cas sont possibles :

- L'état du BDMP $\mathbf{F}_{q_n}$ a déjà été rencontré auparavant ($\mathbf{F}_{q_n} \in S\mathbf{F}$). Das ce cas seule la transition allant de l'état courant (q_0 pour le premier pas) vers l'état q_n sur occurrence de l'événement u est ajoutée à la fonction de transition δ (Alg4,20).
- L'état du BDMP $\mathbf{F}_{q_n}$ n'a pas été rencontré auparavant ($\mathbf{F}_{q_n} \notin S\mathbf{F}$(Alg4,13)). Dans ce cas :
  - L'état du BDMP $\mathbf{F}_{q_n}$ est ajouté dans l'ensemble des états $S\mathbf{F}$ (Alg4,14).
  - L'état de l'automate q_n est ajouté dans l'ensemble des états Q de l'automate (Alg4,15).
  - Si dans ce nouvel état le système est défaillant ($F_{te} = Vrai$), alors l'état q_n est également placé dans l'ensemble des états marqués Q_M (Alg4,16-18).
  - La transition allant de l'état courant (q_0 pour le premier pas) vers l'état de q_n sur occurrence de u est ajoutée à la fonction de transition δ (Alg4,20).

Lorsque l'algorithme a terminé la génération des états successeurs de l'état q_i, un autre état non traité est sélectionné dans Q et la génération continue. L'algorithme se termine lorsque tous les états de Q précédemment calculés ont été traités. La convergence de cet algorithme est assurée par le nombre maximal d'états générés, borné par le nombre de combinaisons possibles pour le vecteur $\mathbf{F}$ ($2^{Card(L)}$ avec L ensemble des feuilles du BDMP).

Le résultat de l'application de l'algorithme 4 sur le BDMP de la figure 3.2 est représenté en figure 3.6. A partir de l'état initial q_0, les états successeurs q_1 et q_4 sont générés.

Pour le pas suivant, l'état q_1 est choisi arbitrairement comme état courant pour poursuivre la génération. L'algorithme s'arrête lorsque tous les états générés (içi 8) ont été utilisés. Le seul état marqué rencontré est l'état q_6, qui correspond à l'état où tous les composants sont défaillants.

A l'issue de cette génération, seuls les éléments correspondants au 5-uplet de l'automate $< \Sigma, Q, q_0, Q_M, \delta >$ sont conservés. Cet automate décrit l'ensemble de toutes les séquences d'événements de défaillance et de réparation possibles dans le BDMP ($\mathcal{L}_D$), et parmi celles-ci, celles qui mènent à la défaillance du système($\mathcal{L}_F$).

3.4 Calcul des séquences de coupe minimales d'un système à composants multimodes.

Le chapitre précédent a proposé un cadre formel permettant la définition et l'obtention des séquences de coupe minimales d'un système dynamique de composants réparables. Ces composants sont supposés ne posséder au plus qu'un événement de défaillance et un événement de réparation et avoir un seul mode de fonctionnement.

Cette hypothèse n'est pas satisfaisante pour permettre le calcul des séquences de coupe minimales sur des modèles représentant des composants ayant au moins deux modes de fonctionnement (actif et dormant par exemple). Parmi les modèles structurels permettant cette représentation, les *Dynamic Fault Trees* et les *Boolean logic Driven Markov Processes* (objet de ce chapitre) retiennent l'hypothèse de composants pouvant être en mode actif ou en mode dormant.

La section suivante (3.4.1) décrit le modèle automate fini sous-jacent à la description d'un composant générique multimode ainsi que sa restriction au modèle de composant retenu par les BDMP. L'effet de cette représentation sur les scénarios caractéristiques du système sera alors décrit en section 3.4.2. La section 3.4.3 proposera une modification des algorithmes de calcul du noyau (donc des séquences de coupe minimales) permettant la prise en compte de différents modes.

3.4.1 Composants multimodes, cas des BDMP.

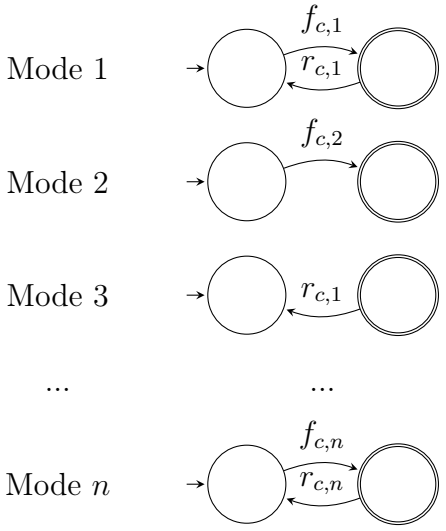


Figure 3.7 – Exemple d'un composant c sous n modes

Dans le cadre de ces travaux, les composants sont considérés multimodes lorsqu'ils possèdent au moins deux modes différents. Le modèle associé a un composant c possédant n modes est présenté en figure 3.7.

Un mode de composant est défini par au plus un événement de défaillance et au plus un événement de réparation permettant l'évolution entre un état où le composant est opérationnel et un état où le composant est défaillant. Ainsi sur la figure 3.7, le mode 1 du composant est constitué d'un événement de défaillance ($f_{c,1}$) et d'un événement de réparation ($r_{c,1}$). Le second mode n'est constitué que d'un unique événement de défaillance ($f_{c,2}$). Il est également possible qu'un événement soit partagé entre plusieurs modes, tel le mode 3 de la figure 3.7 qui ne possède qu'un événement de réparation ($r_{c,1}$), identique à l'événement de réparation du mode 1. Les changements de mode sont supposés être conservatifs : si un composant est défaillant alors le passage d'un mode m à un mode n ne peut ni faire défaillir le composant ni le réparer. Seuls les événements associés au composant peuvent faire varier le niveau de défaillance du composant.

Dans le cadre des BDMP, deux types de composants multimodes sont définis à l'aide des deux types de feuilles (SF et F). Ces composants sont définis suivant les deux modes possibles, actif ou dormant. Conformément à la description des feuilles proposée en section 3.2.2.3, le modèle associé à ces composants est représenté en figure 3.8.

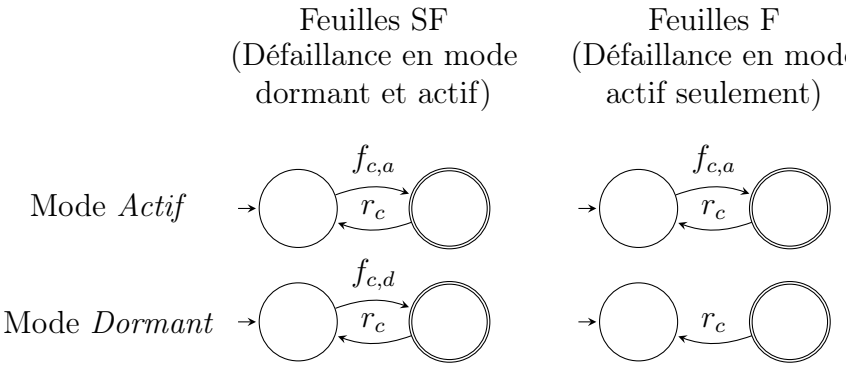


Figure 3.8 – Modèle de composant c d'un BDMP

3.4.2 Séquences d'un système à composants multimodes.

L'automate généré à partir d'un système à composants multimodes possède un alphabet constitué de l'ensemble des événements de défaillance et de réparation caractéristiques des modes des composants. Un tel automate est présenté en figure 3.6. Les séquences dysfonctionnelles (respectivement défaillantes) générées (respectivement marquées) par un tel automate sont donc constituées de ces événements. Sur l'automate de la figure 3.6, $f_{P1,a}f_{P2,a}r_{P1}$ est une séquence dysfonctionnelle et $f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$ une séquence défaillante.

De telles séquences permettent d'exprimer, non seulement un scénario décrivant les défaillances et réparations successives de composant, mais également de connaître les modes dans lesquels les composants subissant ces événements puisque le codage des événement permet d'indiquer le mode correspondant $(f_{c,\mathbf{n}}; r_{c,\mathbf{n}})$. *Il ne suffit alors pas de masquer les informations de modes présentes sur les événements des séquences avant de calculer le noyau du langage défaillant ou l'ensemble des séquences de coupe minimales.*

3.4.3 Prise en compte des modes de composants pour le calcul du noyau.

Le principe de calcul des séquences du noyau $Kern(\mathcal{L}_F)$ et de l'ensemble des séquences de coupe minimales reste identique à celui utilisé pour les systèmes de composants ne possédant qu'un mode de fonctionnement. A partir des séquences ne passant pas deux fois dans le même état $(\mathcal{L}_F^{NL})$ seules sont conservées celles qui ne peuvent être générées à l'aide d'une séquence plus courte de $\mathcal{L}_F^R$. La génération peut être effectuée à l'aide de l'application CR_4 associée à la quatrième règle de cohérence des systèmes dynamiques

réparables. L'application $CR4$, développée dans la section 2.7, définit deux procédures de génération : soit par ajout d'un événement de défaillance soit par distribution ordonnée des événements d'une séquence dans une séquence plus courte sans modification de l'ensemble des composants défaillants.

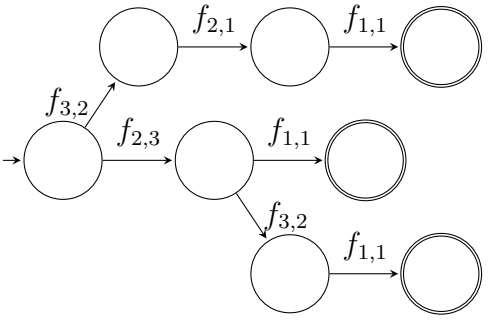


Figure 3.9 – Ajout d'un événement de défaillance dans une séquence défaillante de composants multimodes

L'ajout d'événement de défaillance dans une séquence afin d'augmenter d'une part la longueur de la séquence défaillante considérée et d'autre part l'ensemble des composants défaillants n'est pas modifié par le passage au domaine des composants multimodes. Ainsi sur la figure 3.9, portion d'automate représentant les séquences dysfonctionnelles et défaillantes d'un système à trois composants multimodes, l'ajout de l'événement de défaillance du composant 3 sous mode 2 ($f_{3,2}$) dans la séquence défaillante $f_{2,3}f_{1,1}$ permet toujours de conserver la défaillance du système. Cependant, l'ajout de $f_{3,2}$ avant $f_{2,3}$ change le mode du composant 2 de son mode 3 à son mode 1. Les séquences générées par ajout de $f_{3,2}$ sont alors :

- $f_{2,3}f_{3,2}f_{1,1}$
- $f_{3,2}f_{2,1}f_{1,1}$

La séquence $f_{2,3}f_{3,2}f_{1,1}$ est détectée comme ne faisant pas partie du noyau, car l'ajout de $f_{3,2}$ dans $f_{2,3}f_{1,1}$ après $f_{2,3}$ ne modifie ni la défaillance du système ni les modes des composants correspondant aux événements de la séquence (la séquence $f_{2,3}f_{1,1}$ est incluse dans la séquence $f_{2,3}f_{3,2}f_{1,1}$). A l'opposé, la séquence $f_{3,2}f_{2,1}f_{1,1}$ ne peut être éliminée du noyau car la variation de mode induite sur le composant 2 par l'ajout de l'événement $f_{3,2}$ en première position de séquence ne permet plus de détecter l'inclusion de $f_{2,3}f_{1,1}$ dans $f_{3,2}f_{2,1}f_{1,1}$. L'algorithme de calcul des séquences du noyau n'a donc pas besoin d'être

modifié pour les systèmes de composants multimodes dans sa partie détectant les ajouts d'événement de défaillance. Le noyau va alors conserver les séquences permettant de décrire non seulement les défaillances et les réparations des composants mais également les modes dans lesquels ils interviennent.

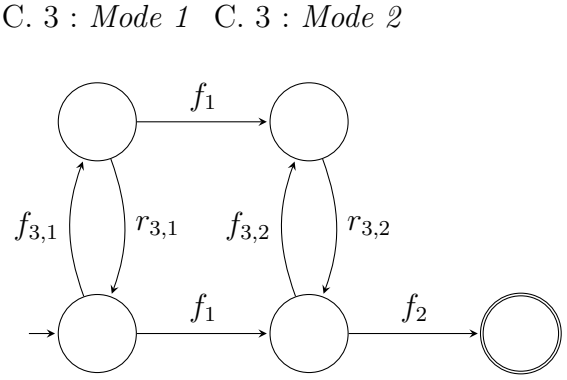


Figure 3.10 – Modification du mode du composant 3 à travers une séquence $f_1 f_2$

La distribution ordonnée des événements d'une séquence dans une autre afin d'obtenir une séquence plus longue en garantissant la conservation de la défaillance du système est illustrée par l'utilisation des séquences $f_{3,1}r_{3,1}$, $f_{3,2}r_{3,2}$ et $f_1 f_2$. L'utilisation de ces séquences sera appuyée par l'automate de la figure 3.10 représentant une partie des scénarios d'un système où seul le composant 3 est multimode. La variation de mode du composant 3 est générée par la défaillance du composant 1, la défaillance du composant 3 ne peut faire varier le mode des composants 1 et 2. Parmi les séquences défaillantes qu'il est possible d'extraire de cette portion d'automate, les suivantes vont supporter la démonstration :

- $f_1 f_2$
- $f_{3,1}r_{3,1}f_1 f_2$
- $f_1 f_{3,2}r_{3,2}f_2$
- $f_{3,1}f_1 r_{3,2}f_2$

Parmi ces séquences, seule $f_1 f_2$ doit être conservée comme candidate à l'appartenance au noyau par l'algorithme de calcul, les autres doivent être exclues du noyau. En effet, les séquences $f_{3,1}r_{3,1}f_1 f_2$ et $f_1 f_{3,2}r_{3,2}f_2$ sont éliminées par l'algorithme 3, car elles traversent deux fois un même état, cependant $f_{3,1}f_1 r_{3,2}f_2$ n'est pas éliminée du noyau par l'algorithme. Afin de supprimer $f_{3,1}$ et $r_{3,2}$ de la séquence $f_{3,1}f_1 r_{3,2}f_2$ et ne conserver que $f_1 f_2$. La condition qui permet l'élimination de $f_{3,1}$ et $r_{3,2}$ est que la séquence $f_{3,1}r_{3,2}$ est non influente sur $f_1 f_2$ conformément à la proposition 2.21. Cette propriété de non influence

passe par la vérification que les séquences $f_{3,1}r_{3,2}f_1f_2$ et $f_1f_{3,1}r_{3,2}f_2$ existent et préservent la défaillance du système et l'ensemble des composants défaillants. La variation de mode du composant 3 compromet la vérification de l'existence de ces séquences.

La solution proposée pour permettre de nouveau l'élimination de la séquence $f_{3,1}f_1r_{3,2}f_2$ des séquences du noyau est de masquer l'information de mode des événements distribués dans la séquence défaillante f_1f_2. Cette suppression de l'information est basée sur la création pour chacun des composants c multimodes d'un événement de défaillance f_c et d'un événement de réparation r_c. Ces deux événements modélisent respectivent la défaillance ou la réparation dans un des modes du composant c :

$$f_c = (f_{c,1}, f_{c,2}, ..., f_{c,n})$$
$$r_c = (r_{c,1}, r_{c,2}, ..., r_{c,n})$$

Il est alors nécessaire d'adapter la partie de l'algorithme 3 concernant la détection de la distribution des événements d'une séquence dans une autre (l11-l20) selon la procédure suivante. Cette procédure est effectuée après le calcul des ensembles finis de séquences $\mathcal{L}_F^{NL}$ et $\mathcal{L}_F^R$ par l'algorithme 1.

1. Pour chacune des transitions dans l'automate portant une information de mode : création d'une transition équivalente sans information de mode
2. Après identification du 4-uplet $(\delta, \sigma'_1, \sigma'_2, \sigma'_3)$, proposition de génération de σ (Alg. 3 l 14) : remplacement de tous les événements de δ portant une information de mode par son équivalent (f_c, r_c)
3. Le test de l'appartenance au noyau de la séquence σ considérée par l'algorithme est poursuivi selon la même procédure que celle appliquée sur les systèmes de composants monomodes.

Ainsi sur l'exemple de la figure 3.10 l'automate est modifié suivant la première étape de la procédure, le résultat est présenté en figure 3.11. Dans ce nouvel automate toutes les transitions étiquetées $f_{3,1}$, $r_{3,1}$, $f_{3,2}$ ou $r_{3,2}$ ont permis la création de transitions équivalentes étiquetées f_3 ou r_3. Lorsque l'algorithme 3 testera l'appartenance au noyau de $f_{3,1}f_1r_{3,2}f_2$ pour la distribution des événements de $\delta = f_{3,1}r_{3,2}$ dans f_1f_2, δ sera transformée en $\delta' = f_3r_3$ suivant la seconde étape de la procédure. L'algorithme testera alors l'existence des séquences $f_3r_3f_1f_2$ et $f_1f_3r_3f_2$ dans l'automate ainsi que la conservation de la défaillance

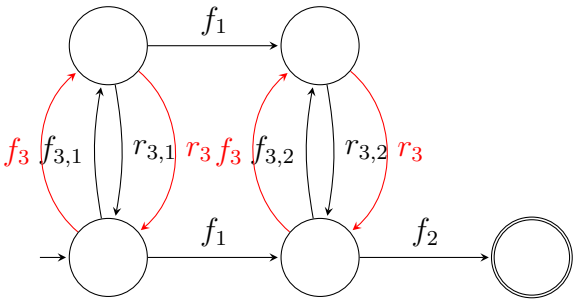


Figure 3.11 – Modification de l'automate pour l'adaptation de l'algorithme 3 aux composants multimodes

du système et de l'ensemble des composants défaillants. La modification de l'automate permet cette vérification et donc le retrait de la séquence $f_{3,1}f_1r_{3,2}f_2$ du noyau $Kern(\mathcal{L}_L)$.

3.5 Résultats sur le système à trois pompes.

Le système à trois pompes présenté en figure 3.1, dont la défaillance est modélisée par le BDMP de la figure 3.2, permet de générer l'automate équivalent présenté en figure 3.6 grâce l'algorithme 4. Cet automate est composé de 8 états dont 1 marqué et de 22 transitions.

Tableau 3.3 – Résultats du calcul des séquences défaillantes du système à trois pompes

Longueur	Nb séq. $\mathcal{L}_F^{NL}$	Nb séq. $\mathcal{L}_F^{R}$
1	0	0
2	0	0
3	3	3
4	0	0
5	2	25
6	0	0
7	2	/
>7	0	/
Total	7	28

L'algorithme 1 permet de trouver 7 séquences défaillantes ne passant pas deux fois

dans un même état ($Card(\mathcal{L}_F^{NL}) = 7$). Ces séquences sont de longueur comprise entre 3 et 7. Les séquences pouvant générer les séquences de $\mathcal{L}_F^{NL}$ sont les séquences du langage défaillant de longueur strictement inférieure et placées dans le langage défaillant restreint $\mathcal{L}_F^R$. Leur longueur est déterminée par $Max(|\sigma|, \sigma \in \mathcal{L}_F^{NL} - 1$ qui vaut donc ici 6 ($7-1=6$). L'algorithme 2 permet alors de calculer un ensemble de 28 séquences. Un synthèse des résultats obtenus par ces deux algorithmes est présentée dans le tableau 3.3.

Toutes les 7 séquences de $\mathcal{L}_F^{NL}$ sont alors analysées par l'algorithme 3 et comparées aux séquences de $\mathcal{L}_F^R$ afin de déterminer si ces séquences font partie du noyau $Kern(\mathcal{L}_F)$. L'appartenance des séquences du noyau aux séquences de coupes sera ensuite analysée afin d'extraire du noyau $Kern(\mathcal{L}_F)$ l'ensemble SCM des séquences de coupe minimales. Les résultats de ces analyses sont présentés dans le tableau 3.4. Dans ce cas d'étude, toutes les séquences du noyau sont des séquences de coupes minimales, le temps total de calcul sur cet exemple n'est pas significatif car inférieur à la seconde.

Tableau 3.4 – Séquences du noyau du système à trois pompes

Longueur	Séq. $\in \mathcal{L}_F^{NL}$	Séq. $\in Kern(\mathcal{L}_F)$	Séq. $\in SCM$
3	$f_{P3,d}f_{P1,a}f_{P2,a}$	$f_{P3,d}f_{P1,a}f_{P2,a}$	$f_{P3,d}f_{P1,a}f_{P2,a}$
	$f_{P1,a}f_{P3,d}f_{P2,a}$	$f_{P1,a}f_{P3,d}f_{P2,a}$	$f_{P1,a}f_{P3,d}f_{P2,a}$
	$f_{P1,a}f_{P2,a}f_{P3,a}$	$f_{P1,a}f_{P2,a}f_{P3,a}$	$f_{P1,a}f_{P2,a}f_{P3,a}$
5	$f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$	$f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$	$f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$
	$f_{P3,d}f_{P1,a}r_{P3}f_{P2,a}f_{P3,a}$		
7	$f_{P3,d}f_{P1,a}r_{P3}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$		
	$f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}r_{P2}f_{P1,a}f_{P2,a}$		
Total	7	4	4

La séquence $f_{P3,d}f_{P1,a}r_{P3}f_{P2,a}f_{P3,a}$ n'a pas été retenue, car elle peut être générée par distribution des événements de la séquence $f_{P3,d}r_{P3}$ dans la séquence $f_{P1,a}f_{P2,a}f_{P3,a}$. De même les séquences $f_{P3,d}f_{P1,a}r_{P3}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$ et $f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}r_{P2}f_{P1,a}f_{P2,a}$ sont supprimées du noyau, car elles résultent respectivement de la distribution des événements des séquences $f_{P3,d}r_{P3}$ dans $f_{P1,a}f_{P2,a}r_{P1}f_{P3,d}f_{P1,a}$ et des événements de la séquence

$f_{P1,a}f_{P2,a}r_{P1}r_{P2}$ dans la séquence $f_{P3,d}f_{P1,a}f_{P2,a}$.

Il est également possible de remarquer que la séquence de coupe minimale $f_{P1,a}$ $f_{P2,a}$ r_{P1} $f_{P3,d}$ $f_{P1,a}$ possède un événement de réparation (r_{P1}). Cet événement est conservé, car la défaillance de la pompe P1 est nécessaire pour faire basculer la pompe P2 dans son mode actif pour qu'elle puisse défaillir, permettant ainsi de décrire une séquence de coupe minimale où, dans l'ordre, P2 défaille en mode actif puis P3 en mode dormant puis P1 en mode actif.

3.6 Bilan.

A partir de la définition formelle de la cohérence des systèmes dynamiques réparables ayant permis la définition des notions de séquences de coupe minimales et du noyau du langage défaillant, ce chapitre a permis l'extension du cadre formel pour permettre l'analyse qualitative de systèmes à composants multimodes.

Le cas spécifique des BDMP a pu être traité moyennant le développement de procédures spécifiques traduisant leur sémantique et des deux hypothèses suivantes :

- La structure d'arbre de défaillance du BDMP est statique.
- Il est fait abstraction de la défaillance à la sollicitation.

L'extension du cadre formel a nécessité l'obtention de deux nouveaux résultats. Un premier résultat a été obtenu sous la forme d'un processus de génération systématique d'un automate équivalent au BDMP représentant l'ensemble des scénarios possibles et pouvant conduire à la défaillance du système, ceci pour permettre l'utilisation de BDMP dans le cadre développé précédemment. La construction de l'algorithme de génération est passé par une analyse sémantique préalable des BDMP dans l'objectif de leur utilisation pour les analyses qualitatives. L'automate obtenu respecte alors les hypothèses décrites dans le chapitre 2 et peut donc être utilisé par les algorithmes de calcul des séquences de coupes minimales. Le second résultat est l'évolution des algorithmes de calcul pour permettre la prise en compte des défaillances et réparations de composants dans différents modes de fonctionnement. Cette modification étant nécessaire pour le calcul des

séquences de coupe d'un BDMP. Ces deux résultats sont illustrés sur un exemple simple dont la défaillance est modélisée par un BDMP.

A l'issue de chapitre, le calcul du noyau du langage défaillant et des séquences de coupe minimales est maintenant possible pour les systèmes dynamique réparables dont la défaillance est modélisée à l'aide d'un BDMP. Le traitement d'un exemple simple a permis de mettre en évidence que les séquences de coupe minimales d'un tel système peuvent être composées aussi bien d'événements de défaillance que de réparation. Ces séquences décrivent des scénarios composés d'événements de défaillance et de réparation décrivant également le modes dans lesquels les composants les génèrent.

Chapitre

4

Présentation d'un cas d'étude : Aval du condenseur d'une tranche nucléaire et système d'alimentation secouru

Sommaire

4.1 Introduction. 133
4.2 Présentation du cas d'étude. 133
4.3 Modèle BDMP du système. 137
4.3.1 Modèle des turbopompes. 137
4.3.2 Modèle des pompes d'extraction. 137
4.3.3 Modèle de l'alimentation électrique. 138
4.3.4 Modèle complet 140
4.4 Calcul du noyau du langage défaillant et des séquences de coupe minimales. 140
4.4.1 Calcul de l'automate équivalent. 141
4.4.2 Calcul des séquences du noyau, séquences de coupe minimales. 141
4.5 Bilan 145

4.1 Introduction.

Les exemples utilisés dans les chapitres précédents ne pouvant, par leur simplicité, être considérés comme représentatifs des systèmes étudiés dans le domaine industriel, ce chapitre développe le calcul des séquences de coupe minimales sur un exemple de taille significative. Afin de permettre ce calcul, les algorithmes décrits dans les chapitres précédents ont été implémentés dans le cadre de la construction d'un démonstrateur admettant en entrée soit un automate fini soit un BDMP.

Après une description succincte du cas d'étude dans la section 4.2, la modélisation de la défaillance de ce système par l'utilisation des BDMP est décrite en section 4.3. La section 4.4 décrit alors le processus de calcul et analyse certaines séquences de coupes minimales.

4.2 Présentation du cas d'étude.

Le cas d'étude considéré dans ce chapitre a pour objectif d'illustrer la méthode proposée dans le chapitre 3 sur un système représentatif de ceux rencontrés par EDF. Les données et modèles proposés dans ce chapitre sont donc représentatifs, mais pour des raisons de confidentialité, ne correspondent pas à la réalité.

Le système retenu pour cette étude est une partie du circuit secondaire d'un Réacteur à Eau Présurisée (REP). Il fait notamment partie du système étudié dans le cadre du projet ANR APPRODYN [ABB+12] pour l'évaluation des méthodes d'analyse quantitative pour la fiabilité dynamique des systèmes. Le cadre de l'étude est représenté en figure 4.1, il concerne le système aval du condenseur, permettant l'alimentation en eau des générateurs de vapeur, système permettant l'échange d'énergie entre le circuit primaire et secondaire, ainsi qu'une partie de son alimentation électrique.

Le système aval du condenseur est présenté plus en détail en figure 4.2 et est associé au diagramme de fiabilité de la figure 4.3. Quatre fonctions principales sont identifiables dans ce système :

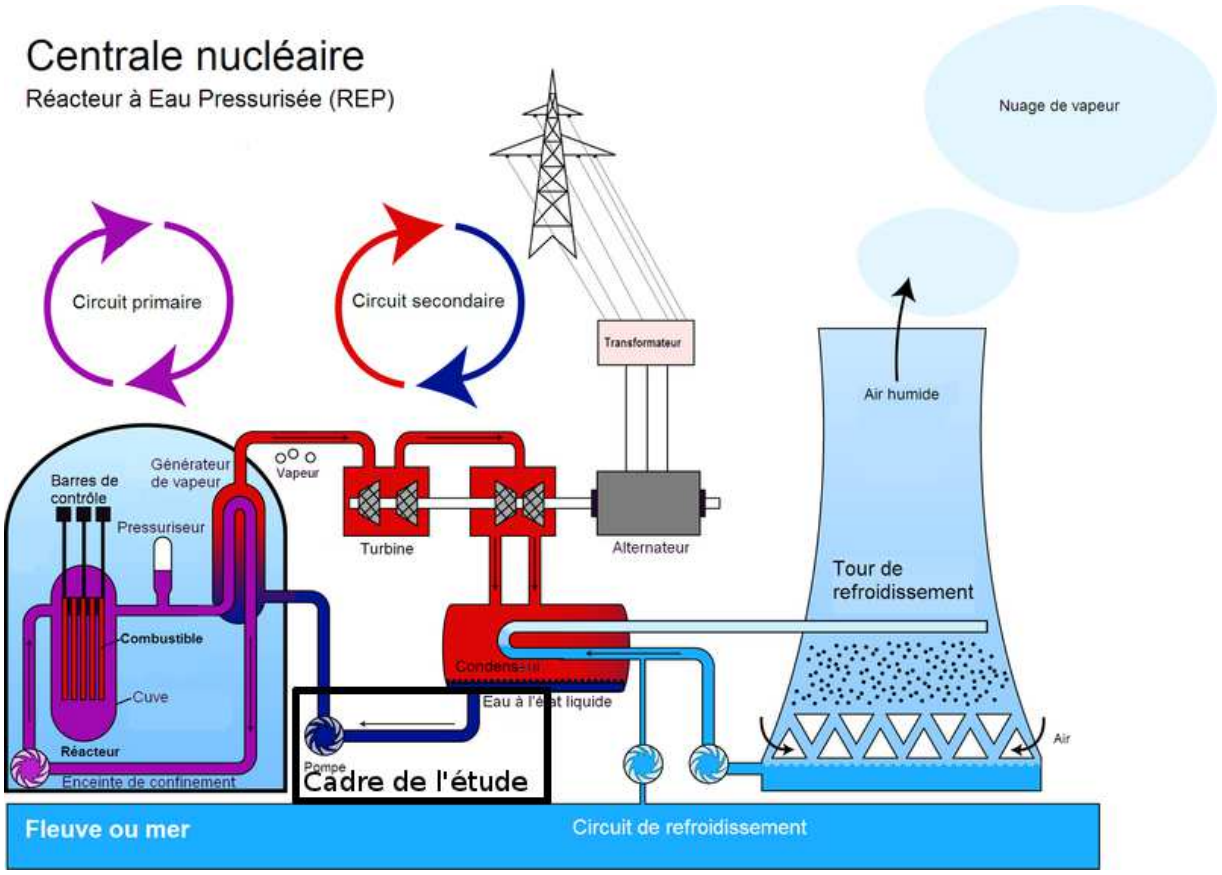


Figure 4.1 – Architecture simplifiée d'un Réacteur à Eau Présurisée (REP)

1. Extraire l'eau du condenseur et permettre d'assurer un débit d'eau de refroidissement.
2. Mettre sous pression l'eau de refroidissement.
3. Réchauffer l'eau de refroidissement.
4. Distribuer l'eau de refroidissement vers les générateurs de vapeur.

Seule l'étude de la défaillance des fonctions 1 et 2 sera réalisée, par ailleurs le barillet VVP (circuit Vapeur Principal) est ignoré dans cette étude. Au final seuls les groupes de pompes d'extraction (CEX) et de turbopompe (TPA) sont étudiés, le diagramme de fiabilité associé est présenté en figure 4.4

La fonction d'extraction de l'eau du condenseur est réalisée par les trois pompes d'extraction (CEX), fonctionnant en redondance 2/3. En situation normale, deux des pompes sont en fonctionnement, la troisième en attente. En cas de défaillance d'une des pompes en fonctionnement, la pompe redondante est démarrée. Lorsque deux des pompes de ce

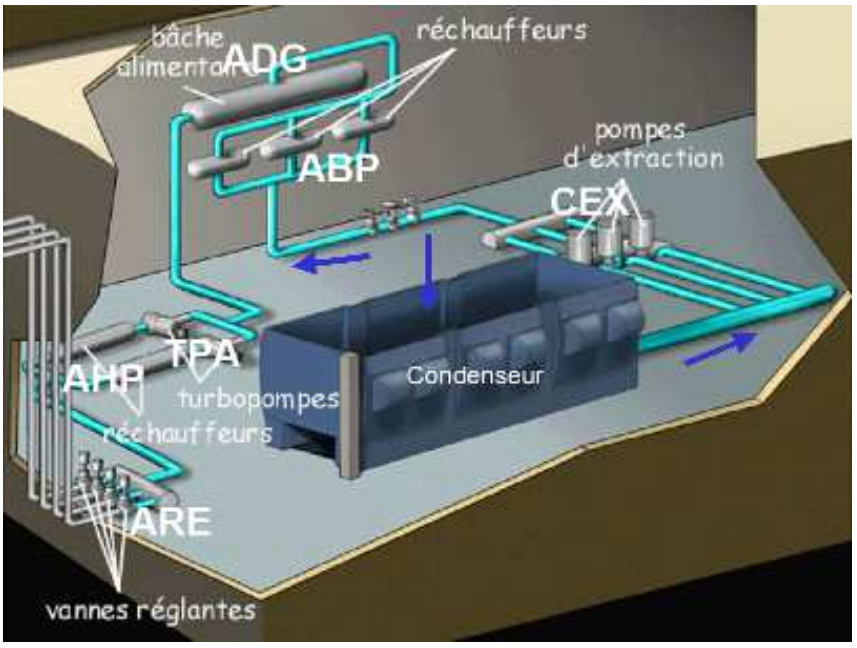


Figure 4.2 – Système Aval du condenseur.

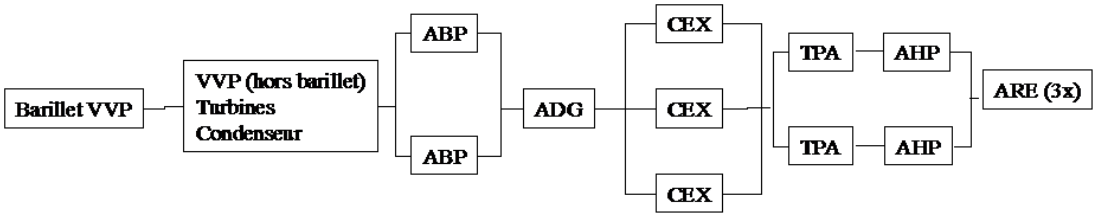


Figure 4.3 – Diagramme de fiabilité complet du système aval du condenseur.

groupe sont défaillantes, le système n'est plus capable d'alimenter les générateurs de vapeur en eau de refroidissement, il est donc défaillant.

La mise sous pression est réalisée par le groupe de deux turbopompes (TPA), en redondance passive. En situation normale, une seule des deux turbopompes est active, l'une d'entre elles est en charge maximale alors que la seconde est en mode dormant. Si la turbopompe principale subit une défaillance, la turbopompe redondante est alors mise en marche afin d'assurer la mise en pression de l'eau de refroidissement. Lorsque les deux Turbopompes sont défaillantes, le système est considéré comme défaillant : la fonction 2 n'est plus assurée.

Chacune des pompes d'extraction est alimentée électriquement par le système d'alimentation secouru présenté en figure 4.5. Ce système d'alimentation est une version sim-

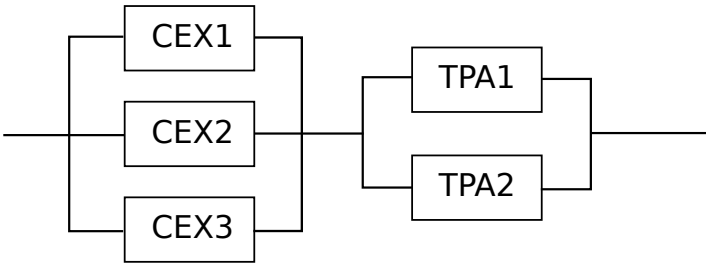


Figure 4.4 – Diagramme de fiabilité simplifié du système aval du condenseur.

plifiée de celui étudié dans [Bou08]. Il permet l'alimentation de différents équipements, soit à l'aide de transformateurs électriques permettant le passage d'une haute tension à une basse tension, soit à l'aide d'un générateur diesel de secours.

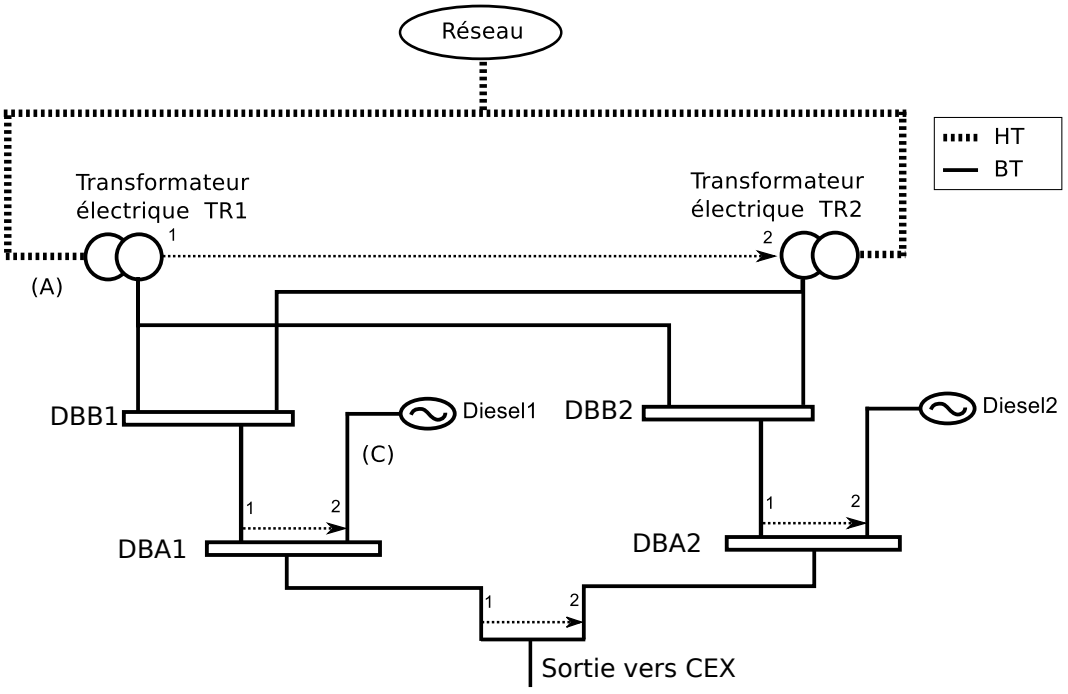


Figure 4.5 – Système d'alimentation secouru permettant l'alimentation des pompes d'extraction.

Le système est composé de deux lignes en redondance passive. La première ligne, active en situation normale est constituée des tableaux électriques DBB1 et DBA1 et du diesel 1. La seconde, à l'arrêt en situation normale, est composée des tableaux DBA2 et DBB2 et du diesel 2. L'alimentation de ces deux lignes est faite par les deux transformateurs en redondance passive, le transformateur 1 étant le principal, le transformateur 2 celui de secours. Les systèmes redondants sont activés lors de la défaillance des systèmes princi-

paux et désactivés lors de la réparation des systèmes principaux pour revenir en situation normale.

4.3 Modèle BDMP du système.

Dans cette section, nous proposons un modèle BDMP représentant la fiabilité du système explicité dans la section précédente. Ce modèle est conforme aux restrictions présentées en section 3.2.1 (pas de porte PAND ni défaillance à la sollicitation) afin de permettre la génération d'un automate équivalent portant tous les scénarios possibles dans le système.

4.3.1 Modèle des turbopompes.

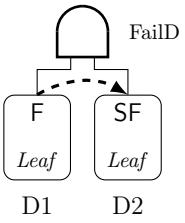


Figure 4.6 – Modèle BDMP des turbopompes

Les deux turbopompes $TPA1$ et $TPA2$ sont modélisées respectivement par les feuilles $D1$ et $D2$. La défaillance possible de la pompe $TPA2$ en mode dormant est prise en compte en lui associant la feuille $D2$, de type SF.

Ces deux pompes sont en parallèles, elles sont donc reliées par une porte ET. La redondance possible entre $TPA1$ et $TPA2$ est modélisée par une gâchette entre ces deux feuilles. Le modèle BDMP associé à ce sous-système est présenté en figure 4.6. La défaillance de ce sous-système est associée à la sortie de la porte $FailD$.

4.3.2 Modèle des pompes d'extraction.

La sûreté du sous-système composé des 3 pompes d'extraction est modélisée par le BDMP présenté en figure 4.7. Les trois pompes d'extraction $CEX1$ à $CEX3$ sont modé-

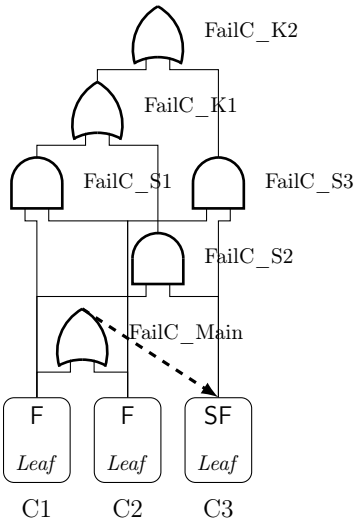


Figure 4.7 – Modèle BDMP des pompes d'extraction

lisées respectivement par les feuilles $C1$ à $C3$. Afin de prendre en compte la défaillance en mode dormant de la pompe $CEX3$, la feuille $C3$ est donc la seule à être de type SF, car seule cette pompe peut se trouver en mode dormant.

Le sous-système composé de ces pompes est défaillant si deux des pompes sont défaillantes parmi les trois. La définition sémantique proposée en section 3.2.2 ne permettant pas l'illustration des portes k/N, la relation $2/3$ est modélisée à l'aide de trois portes ET (FailC_S1, FailC_S3 et FailC_S3) et deux portes OU (FailC_K1 et FailC_K2).

La pompe $C3$ est en secours d'une des deux pompes $C1$ et $C2$, elle est démarrée si l'une ou l'autre de ces pompes est défaillante. Afin de modéliser cette relation, la porte FailC_Main est ajoutée afin de modéliser la défaillance d'au moins une des pompes $CEX1$ et $CEX2$. Cette porte n'est pas utilisée par l'arbre de défaillance (sa sortie n'est pas rattachée au top event à travers l'arbre de défaillance) car elle sert uniquement à générer l'activation de la pompe $CEX3$. Le lien de redondance est donc modélisé par la gâchette partant de cette porte et ayant comme destination la feuille $C3$. La défaillance de ce sous-système est représentée par la sortie de la porte $FailC_K2$.

4.3.3 Modèle de l'alimentation électrique.

Les défaillances du sous-système d'alimentation sont modélisées par le BDMP de la figure 4.8. Les deux transformateurs sont modélisés par les feuilles $Tr1$ et $Tr2$, le réseau

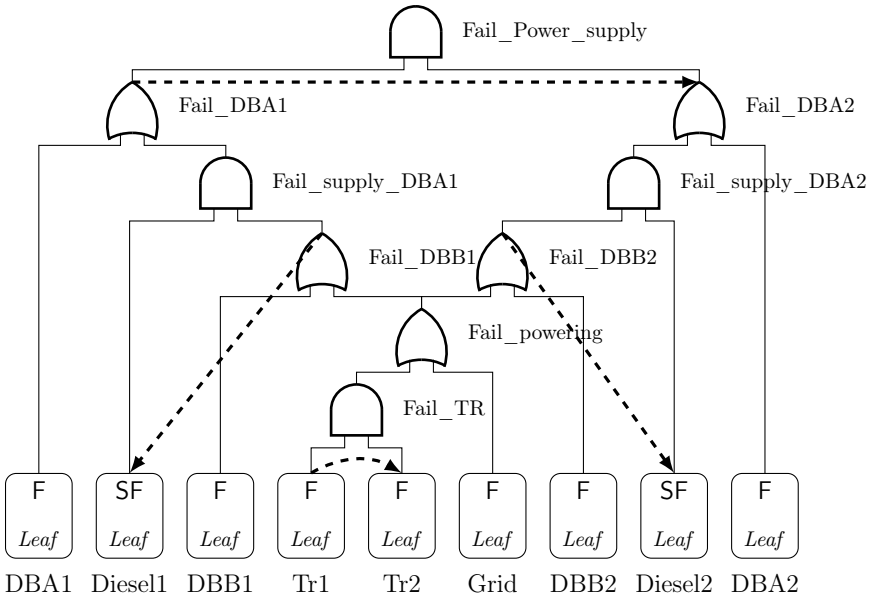


Figure 4.8 – Modèle BDMP du système d'alimentation

électrique par la feuille *Grid*. Les deux transformateurs étant en redondance passive, une gâchette est présente entre les deux feuilles les modélisant. L'alimentation n'est plus possible si les deux transformateurs sont défaillants ou si le réseau ne peut fournir l'énergie. Cette relation est modélisée par la sortie de la porte *OU* $Fail_Powering$.

L'alimentation des pompes d'extraction est réalisée à l'aide d'une des deux lignes d'alimentation en redondance passive. Chacune de ces deux lignes peut fournir l'énergie soit provenant du réseau à travers l'un des deux transformateurs, soit à l'aide de son générateur diesel. Elles sont modélisées respectivement à l'aide des feuilles $DBA\{1,2\}$, $Diesel\{1,2\}$ et $DBB\{1,2\}$. Chacun des diesels étant dans son mode dormant en situation normale et dans l'objectif de prendre en compte les défaillances pouvant intervenir sur ces matériels dans ce mode, les diesels sont alors modélisés à l'aide de feuilles SF. Ces diesels sont en redondance passive de l'alimentation par les tableaux électriques $DBB\{1,2\}$, ces redondances sont modélisées à l'aide de gâchettes ayant pour origine les portes $Fail_DBB\{1,2\}$ et pour destination les feuilles modélisant les diesels. Les deux lignes d'alimentation sont en redondance passive avec la ligne 1 en principale, la modélisation de cette redondance est effectuée à l'aide de la gâchette partant de la porte $Fail_DBA1$ à la porte $Fail_DBA2$. La défaillance de ce sous-système est représentée par la sortie de la porte $Fail_Power_supply$.

4.3.4 Modèle complet

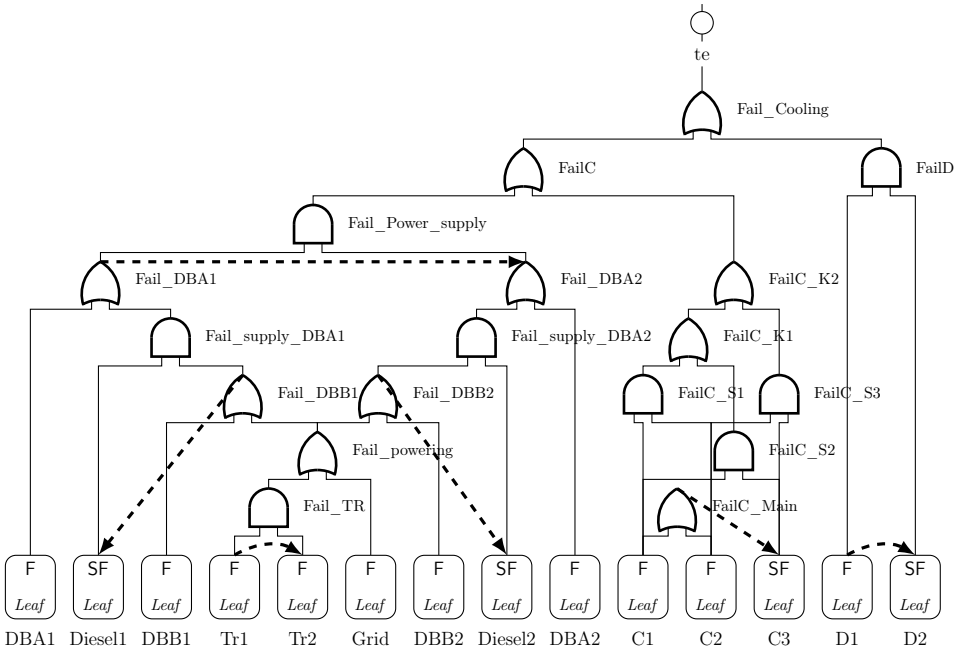


Figure 4.9 – Modèle BDMP de la sûreté du système aval au condenseur.

Le modèle de défaillance par BDMP du système complet est présenté en figure 4.9. Ce modèle résulte de l'assemblage des modèles BDMP des sous-systèmes présentés dans les sections précédentes. La défaillance du système est représentée par le top event *te*. Le système est défaillant si le sous-système des turbopompes est défaillant ou si le sous-système des pompes d'extraction est défaillant (porte $Fail_Cooling$). Le sous-système des pompes d'extraction est défaillant si deux des pompes sont défaillantes ou s'il n'est pas possible de les alimenter électriquement (porte $FailC$).

La défaillance du système complet est alors modélisée par un BDMP de 14 feuilles, reliées par 18 portes. La modélisation des mécanismes de redondance a nécessité l'utilisation de 6 gâchettes.

4.4 Calcul du noyau du langage défaillant et des séquences de coupe minimales.

Afin de permettre le calcul des séquences du noyau du langage défaillant $Kern(\mathcal{L}_F)$ et celles de l'ensemble des séquences de coupe minimales SCM, l'ensemble des algorithmes permettant la génération de l'automate (4), l'extraction des séquences (1, 2) et le calcul

du noyau (3) ont été implémentés dans une maquette développée en Python. Le calcul du système d'équations combinatoires permettant la description de la fonction de structure de l'arbre de défaillance statique et des modes des feuilles du BDMP est également réalisé sous cet environement.

Les algorithmes de calcul ayant été développés et décrits dans les chapitres précédents, cette section présentera uniquement les résultats de l'analyse sur ce cas d'étude en s'attardant plus spécifiquement sur certaines des séquences retournées. L'implémentation expérimentale n'ayant pas été développée dans un objectif d'optimisation du temps de calcul, les temps de calcul associés à chaque étape ne sont donnés qu'à titre indicatif. La machine utilisée est un PC de bureau doté d'un CPU 4 coeurs Intel(R) Xeon(R) à 2.80GHz et équipé de 8Go de mémoire.

4.4.1 Calcul de l'automate équivalent.

L'automate équivalent au BDMP de la figure 4.9 est composé de 16 384 états (dont 13 300 sont marqués et représentent des états où le système est défaillant), et de 229 376 transitions. Le nombre d'états correspond exactement à celui attendu pour un automate généré à partir d'un BDMP de 14 feuilles ($2^{14} = 16384$ cf. 3.3.1). Le temps de génération de cet automate a été d'environ 50 secondes. Cet automate porte l'ensemble des séquences dysfonctionnelles (possibles dans le modèle) et des séquences défaillantes (séquence conduisant à la défaillance du système en fin de séquences).

4.4.2 Calcul des séquences du noyau, séquences de coupe minimales.

La complexité du calcul des séquence du noyau $Kern(\mathcal{L}_f)$ et des séquences de coupe minimales SCM est fortement dépendante de la longueur maximale des séquences généré, cette longueur étant fortement liée au nombre d'états dans l'automate. Dans la pratique, les ingénieurs fiabilistes qui analysent des systèmes critiques (énergie, transport ...) s'intéressent le plus souvent aux séquences de longueur maximale 5 à 6. La longueur 6 est donc retenue comme cible dans notre étude de cas

4.4.2.1 Calcul des séquences de coupe minimales par le calcul du noyau du langage défaillant.

Le calcul des séquences du noyau $Kern(\mathcal{L}_F)$ est réalisé à l'aide des algorithmes 1, 2 et 3 présentés dans le chapitre 2. La première étape est de calculer l'ensemble $\mathcal{L}_F^{NL}$ des séquences ne passant pas deux fois dans un même état de l'AF. Ce calcul est limité aux séquences de longueur inférieure ou égale à 6, les résultats sont présentés dans le tableau 4.1. La longueur maximale de ces séquences étant de 6, le calcul du langage défaillant restreint $\mathcal{L}_F^R$ est borné au séquences de longueur $Max(|\sigma|, \sigma \in \mathcal{L}_F^{NL}) - 1 = 6 - 1 = 5$. Ce langage contient toutes les séquences défaillantes pouvant permettre la génération des séquences de $\mathcal{L}_F^{NL}$ à l'aide de l'application $CR4$ associée à la quatrième règle de cohérence. Le nombre des séquences trouvées et leur longueur sont présents dans le tableau 4.1.

Tableau 4.1 – Nombre et longueur des séquences de $\mathcal{L}_F^{NL}$,$\mathcal{L}_F^R$,$Kern(\mathcal{L}_F)$ et SCM

Longueur	Nb. séq. $\mathcal{L}_F^{NL}$	Nb. séq. $\mathcal{L}_F^R$	$Kern(\mathcal{L}_F)$	SCM
1	0	0	0	0
2	9	9	9	9
3	255	255	19	19
4	4 614	4 897	39	39
5	68 745	79 594	17	17
6	930 294	/	47	47
Total	1 003 917	84 755	131	131

Une fois ces deux ensembles calculés, la génération de chacune des séquences de $\mathcal{L}_F^{NL}$ par une séquence de longueur strictement inférieure de $\mathcal{L}_F^R$ est évaluée. Ainsi pour évaluer si les séquences de $\mathcal{L}_F^{NL}$ de longueur 3 sont des séquences du noyau $Kern(\mathcal{L}_F)$, elles doivent être comparées au séquences de $\mathcal{L}_F^R$ de longueur inférieure ou égale à 2. Au plus, $255 * 9 = 2295$ tests de génération sont donc nécessaires. Le calcul des séquences du noyau à l'ordre 6 représente alors, au plus, $8,5E10$ tests de génération.

A l'issue de ce calcul, seules 131 séquences sont retournées par le code de calcul comme étant des séquences du noyau $Kern(\mathcal{L}_F)$. Le temps de calcul du noyau a été d'un peu plus de 10 heures. Le tableau 4.1 récapitule les résultats de ce calcul, l'ensemble de toutes les séquences du noyau calculées est présenté en annexe. Il est possible de remarquer sur

cet exemple, que toutes les séquences du noyau sont également des séquences de coupe minimales, comme pour l'exemple du chapitre précédent.

4.4.2.2 Analyse des résultats.

Les séquences de coupe minimales de longueur 2 décrivent les scénarios les plus courts permettant d'obtenir la défaillance du système. Parmi celles-ci, la séquence $f_{DBA1,a}f_{DBA2,a}$ est la seule qui concerne le système d'alimentation, les autres décrivent des scénarios d'événements de défaillance provenant des systèmes de pompes. Il est également intéressant de remarquer que les défaillances des pompes interviennent uniquement à cette longueur, les séquences de coupe minimales de longueur supérieure ne décrivent que des scénarios de défaillance et de réparation du système d'alimentation secouru.

A partir de la longueur 4, les séquences de coupe minimales peuvent contenir des événements de réparation, c'est par exemple le cas de $f_{DBA1,a}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{DBA1,a}$. Dans cette séquence, la première défaillance de $DBA1$ conduit à la défaillance $f_{DBB2,a}$, le composant $DBA1$ est alors réparé pour permettre la défaillance du générateur Diesel 2 en mode dormant, puis il défaille à nouveau. Cette séquence décrit un scénario complexe permettant d'obtenir un système défaillant dont les composants $DBA1$,$DBB2$ et $Diesel2$ sont défaillants, la défaillance de $DBB2$ intervenant dans son mode actif, la défaillance de $Diesel2$ dans son mode dormant et la défaillance de $DBB1$ dans son mode actif. Cette séquence est la seule des séquences de coupe minimales à décrire un tel scénario.

A partir de la longueur 6, il est possible d'identifier des séquences de coupe minimales incluant d'autres séquences de coupe minimales de tailles inférieures. C'est le cas, par exemple, pour les séquences

- $f_{Diesel1,d}f_{Grid,a}f_{DBB2,a}r_{Grid}f_{Diesel2,d}f_{DBA1,a}$
- $f_{Grid,a}f_{Diesel1,a}f_{DBB2,a}r_{Grid}f_{Diesel2,d}f_{DBA1,a}$

incluant chacune la séquence $f_{Grid,a}f_{Diesel2,d}f_{DBA1,a}$. Cette inclusion est légitime, car conformément au cadre défini dans cette étude, il n'est pas possible de générer ces deux séquences de longueur 6 à partir de la séquence de longueur 3 par l'utilisation des règles de cohérence des systèmes dynamiques et réparables. Les règles de cohérence ne permettent en effet pas d'ajouter l'événement de réparation r_{Grid} à la séquence $f_{Grid,a}f_{Diesel2,d}f_{DBA1,a}$ et de conclure sur la défaillance du système à l'issue des séquences ainsi obtenues. Cet

événement de réparation ne peut non plus être ajouté par la distribution des événements d'une séquence non influente sur $f_{Grid,a} f_{Diesel2,d} f_{DBA1,a}$.

Une séquence de coupe minimale, dans le cas des BDMP, est une séquence d'événements de défaillance *et de réparation* provenant des composants et ne peut pas être définie comme une séquence n'incluant aucune autre séquence de coupe minimale. L'approximation utilisant comme définition pour les séquences de coupe minimales la non-inclusion d'une séquence de taille inférieure, utilisée notamment dans [Bou06], ne permet alors plus de garantir la conservation de la capacité de reconstruction d'au moins toutes les séquences de coupe par utilisation des règles de cohérence. Néanmoins l'utilisation de cette approximation, cependant non conservative, permet de diminuer largement le temps de calcul. De plus le nombre de séquences de coupe minimales non générées par cette approximation reste limité : sur cet exemple seules 2 séquences de coupe minimales sont perdues sur l'ensemble des 131 présentes jusqu'a l'ordre 6.

4.4.2.3 Calcul direct des séquences de coupes minimales.

Si l'objectif de l'étude de défaillance se limite à l'analyse de la première défaillance à travers le calcul des séquences de coupes minimales, il est possible de limiter le nombre de séquences générées pour diminuer les temps de calcul. Le principe du calcul est décrit en section 2.8.2.5 et consiste à placer en entrée de l'algorithme 3 les intersections des ensembles de séquences $\mathcal{L}_F^{NL}$ et $\mathcal{L}_F^R$ avec l'ensemble des séquences de coupe $\mathcal{L}_{CS}$. Le nombre des séquences calculées ainsi que leur longueur sont présentés dans le tableau 4.4.2.3. Ainsi pour déterminer les séquences de coupe minimales de longueur 6, $215451 * 31003 \simeq 6,78E9$ tests de génération sont, au maximum, nécessaires. Sur ce cas d'étude, la restriction du mécanisme de calcul aux seules séquences de coupe minimales permet une diminution de l'ordre de 91% du coût de calcul à la longueur 6, pour l'obtention du même ensemble de séquences. Le temps de calcul n'est alors plus que de l'ordre de 20 minutes. Il est également nécessaire de rappeler que, même si dans cet exemple le noyau du langage défaillant est identique à l'ensemble des séquences de coupe minimales, rien ne garantit l'égalité de ces deux ensembles dans le cas général.

Tableau 4.2 – Nombre et longueur de séquences dans $\mathcal{L}_F^{NL} \cap \mathcal{L}_{CS}$ et $\mathcal{L}_F^{R} \cap \mathcal{L}_{CS}$, entrées de l'algorithme 3 pour le calcul direct des séquences de coupe minimales.

Longueur	Nb. séq. $\mathcal{L}_F^{NL} \cap \mathcal{L}_{CS}$	Nb. séq. $\mathcal{L}_F^{R} \cap \mathcal{L}_{CS}$	SCM
1	0	0	0
2	9	9	9
3	172	172	19
4	2 226	2 402	39
5	23 458	28 420	17
6	215 451	/	47
Total	241 316	31 003	131

4.5 Bilan

Les calculs des séquences du noyau du langage défaillant $Kern(\mathcal{L}_F)$ et des séquences de coupes minimales SCM ont été limités au calcul des séquences de longueur inférieure ou égale à 6. Ces deux calculs permettent l'obtention du même ensemble de séquences avec des temps de calcul respectifs de 10 heures et 20 minutes. Le calcul direct des séquences de coupes minimales permet un calcul plus rapide, mais ne permet pas de garantir la conservation de la capacité à régénérer l'ensemble des séquences défaillantes de longueur inférieure ou égale à 6 à l'aide des règles de cohérence.

Le traitement de cet exemple a également permis de remarquer que l'utilisation d'une définition des séquences de coupe minimales basée sur un critère de non-inclusion d'une séquence de coupe minimale de taille inférieure permet de réaliser une approximation de l'ensemble des séquences de coupe minimales non conservative, elle garantit pas l'exhaustivité du calcul de l'ensemble des séquences de coupes minimales d'un BDMP.

Conclusions & perspectives.

Dans ce mémoire nous avons présenté un cadre formel, basé sur la théorie des langages et les automates à états finis, permettant la définition de la cohérence des systèmes dynamiques réparables, du noyau du langage défaillant et de l'ensemble de séquences de coupe minimales ainsi que leur calcul.

L'hypothèse principale retenue dans cette étude est que les scénarios d'événements de défaillance et de réparation décrits par un modèle représentant la défaillance du système peuvent toujours être représentés à l'aide d'un automate fini.

Moyennant cette hypothèse, les résultats suivants ont été obtenus :

- Définition de la cohérence des systèmes dynamiques et réparables à partir des langages générés et marqués par l'automate fini.
- Définition du noyaux du langage défaillant et des séquences de coupe minimales comme représentations minimales respectivement du langage défaillant et de l'ensemble des séquences de coupe.
- Définition d'algorithmes permettant de calculer ces deux ensembles par restriction du langage défaillant, de taille infinie, à un ensemble fini de séquences défaillantes ne passant pas deux fois par un même état.
- Extensions pour la prise en compte de modes de fonctionnement multiples des composants pour les calcul du noyau et de l'ensemble des séquences de coupes minimales.
- Réalisation d'une analyse sémantique d'une sous-classe de BDMP dans l'objectif de réaliser des analyses qualitatives sur ce modèle.
- Construction d'une méthode systématique permettant l'obtention d'un automate fini représentant l'ensemble des toutes les séquences décrites par un BDMP.

L'ensemble des définitions et méthodes proposées a été illustré par différents exemples, définis soit par un automate fini soit par un BDMP. La difficulté du passage à l'échelle à également été illustrée sur un exemple de taille significative.

Les perspectives ouvertes par ces travaux sont nombreuses.

En premier lieu, une des préoccupations de l'ingénieur fiabiliste est de savoir si le modèle construit représente un système dynamique cohérent. Or dans nos travaux les systèmes sont considérés, par hypothèse, comme étant cohérents car il n'existe pour l'heure aucune méthode permettant la démonstration de cette cohérence de manière univoque. Le développement de méthodes de vérification de modèles de sûreté, basées sur la description formelle de la cohérence proposée dans cette thèse, est donc nécessaire. Il serait également utile d'analyser de quelle manière les systèmes dits incohérents ne respectent pas les critères exposés dans le chapitre 2. Cette analyse permettrait d'une part de classifier les systèmes en fonction du type d'incohérence rencontré (en fonction du critère non respecté) et également d'étudier quel est l'effet de cette incohérence sur la définition du noyau du langage défaillant et sur l'ensemble des séquences de coupes minimales ainsi que sur la manière de les calculer.

La méthode proposée pour le calcul du noyau du langage défaillant et de l'ensemble des séquences de coupe minimales peut également faire l'objet de plusieurs développements complémentaires. Nos travaux ne proposent en effet pas de critère formel permettant de choisir entre le calcul du noyau et celui de l'ensemble des séquences de coupe minimales. L'existence d'un tel critère permettrait de garantir l'exhaustivité de la représentation de l'ensemble du langage défaillant à travers le calcul des séquences de coupe minimale (de complexité moindre pour certains systèmes). Un tel critère pourrait par exemple être exprimé en terme d'accessibilité et de coaccessibilité de certains états non marqués, accessibles uniquement après passage dans un état marqué. Le temps de calcul pour l'analyse des séquences de grande longueur pourrait également être réduit par une optimisation des algorithmes d'extractions des séquences et de calcul du noyau. Cela pouvant passer par une modification de la structure de données utilisée dans notre maquette expérimentale.

D'une manière plus générale, le cadre formel proposé peut sans doutes être étendu pour la prise en compte de mécanismes de défaillance plus complexes. Ces nouvelles extensions pourraient par exemple prendre en compte les phases dans lesquelles se trouve le système, permettant alors de traiter les systèmes multiphasés dynamiques réparables. La prise en compte des variations de phase pourrait, par exemple, nécessiter la création d'un nouvel alphabet d'événements de variation de phase, dont les occurrences ne peuvent faire varier l'état de défaillance des composants.

Afin de lever une des restrictions posée sur les BDMP dans nos travaux, il serait par ailleurs intéressant d'intégrer la possibilité de défaillance à la sollicitation des composants (défaillance lors du changement de mode). Cette extension semble assez complexe, car l'événement de défaillance à la sollicitation n'est pas indépendant de l'événement de défaillance du composant (ou du sous-système générant la variation de mode). L'hypothèse d'indépendance des événements de défaillance et de réparation des composants ne serait alors plus respectée.

Enfin, il nous paraît également nécessaire de repenser la meilleure manière de représenter les résultats des analyses qualitatives, car les systèmes complexes présentent un grand nombre de séquences de coupe minimales. Pour permettre une amélioration de cette représentation, deux types d'approches nous semblent possibles. La première consiste, par un traitement en amont du calcul du noyau, à déterminer des modules indépendants dans le modèle de défaillance du système comme il est possible de le faire pour les arbres de défaillance [BE65] ou les arbres de défaillance dynamiques [GD97]. La seconde possibilité est un post-traitement permettant de regrouper les séquences du noyau en fonction des composants impliqués. Cale permettrait éventuellement de retrouver les modules indépendants du système.

Bibliographie

[AAS86] M. Alam and U.M. Al-Saggaf. Quantitative reliability evaluation of repairable phased-mission systems using markov approach. *IEEE Transactions on Reliability*, 35(5) :498–503, 1986.

[ABB+12] Jean-François Aubry, Génia Babykina, Anne Barros, Nicolae Brinzei, Gilles Deleuze, Benoîte de Saporta, François Dufour, Yves Langeron, and Huilong Zhang. Rapport final du projet APPRODYN : APPROches de la fiabilité DYNamique pour modéliser des systèmes critiques. Technical report, March 2012.

[AJSM84] P. Azema, G. Juanole, E. Sanchis, and M. Montbernard. Specification and verification of distributed systems using prolog interpreted petri nets. In *Proceedings of the 7th international conference on Software engineering*, pages 510–518. IEEE Press, 1984.

[BB03] M. Bouissou and J.L. Bon. A new formalism that combines advantages of fault-trees and Markov models : Boolean logic Driven Markov Processes. *Reliability Engineering and System Safety*, 82(2) :149–163, 2003.

[BBBV91] M. Bouissou, H. Bouhadana, M. Bannelier, and N. Villatte. Knowledge modelling and reliability processing : Presentation of the FIGARO language and associated tools. In *IFAC/IFIP/EWICS/SRE Symposium,Safecomp'91, Trondheim (Norvège)*, pages 69–75, 1991.

[BC73] R.E. Barlow and P. Chatterjee. Introduction to fault tree analysis. Technical report, DTIC Document, 1973.

[BCS07] H. Boudali, P. Crouzen, and M. Stoelinga. A compositional semantics for Dynamic Fault Trees in terms of Interactive Markov Chains. *Lecture Notes in Computer Science*, 4762 :441, 2007.

[BD05] H. Boudali and J.B. Dugan. A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering and System Safety*, 87(3) :337–349, 2005.

[BE65] Z.W. Birnbaum and JD Esary. Modules of coherent binary systems. *Journal of the Society for Industrial and Applied Mathematics*, 13(2) :444–462, 1965.

[BES61] Z.W. Birnbaum, J.D. Esary, and S.C. Saunders. Multi-component systems and structures and their reliability. *Technometrics*, 3(1) :55–77, 1961.

[Bou06] M. Bouissou. Détermination efficace de scenarii minimaux de défaillance pour des systèmes séquentiels. In *15ème colloque de fiabilité et maintenabilité, Lille (France)*, 2006.

[Bou08] M. Bouissou. *Gestion de la complexité dans les études quantitatives de sûreté de fonctionnement de systèmes*. Tec & Doc Lavoisier, 2008.

[BPMC01] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. Improving the analysis of dependable systems by mapping fault trees into bayesian networks. *Reliability Engineering & System Safety*, 71(3) :249–260, 2001.

[BR04] A. Bobbio and DC Raiteri. Parametric fault trees with dynamic gates and repair boxes. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, pages 459–465, 2004.

[BW78] R.E. Barlow and A.S. Wu. Coherent systems with multi-state components. *Mathematics of Operations Research*, 3 :275–281, 1978.

[CDFH93] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed colored nets and symmetric modeling applications. *IEEE Transactions on Computers*, 42(11) :1343–1360, 1993.

[CR05] D. Codetta-Raiteri. The conversion of dynamic fault trees to stochastic Petri nets, as a case of graph transformation. *Electronic Notes in Theoretical Computer Science*, 127(2) :45–60, 2005.

[CSD00] D. Coppit, K.J. Sullivan, and J.B. Dugan. Formal semantics of models for computational engineering : a casestudy on dynamic fault trees. In *11th International Symposium on Software Reliability Engineering, 2000. ISSRE 2000. San Jose (USA)*, pages 270–282, 2000.

[DBB90] J.B. Dugan, S.J. Bavuso, and M.A. Boyd. Fault trees and sequence dependencies. In *Reliability and Maintainability Symposium, 1990. Proceedings., Annual, Los Angeles (USA)*, pages 286–293. IEEE, 1990.

[DBB92] JB Dugan, SJ Bavuso, and MA Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3) :363–377, 1992.

[DGM03] C. Delamare, Y. Gardan, and P. Moreaux. Efficient implementation for performance evaluation of synchronous decomposition of high level stochastic Petri nets. In *Proc. ICALP2003 Workshop on Stochastic Petri Nets and Related Formalisms, Eindhoven (The Netherlands)*, pages 164–183, 2003.

[DR96] Y. Dutuit and A. Rauzy. A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability*, 45(3) :422–425, 1996.

[DSB93] J. Dugan Salvatore and J. Bechta. Fault trees and Markov models for reliability analysis of fault-tolerant digital systems. *Reliability Engineering & System Safety*, 39(3) :291–307, 1993.

[DSC00] J.B. Dugan, K.J. Sullivan, and D. Coppit. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *IEEE Transactions on Reliability*, 49(1) :49–59, 2000.

[Dug91] J.B. Dugan. Automated analysis of phased-mission reliability. *IEEE Transactions on Reliability*, 40(1) :45–52, 1991.

[Fus76] JB Fussell. Fault tree analysis : concepts and techniques. *Generic Techniques in System Reliability Assessment*, 1976.

[GD97] R. Gulati and J.B. Dugan. A modular approach for analyzing static and dynamic fault trees. In *Reliability and Maintainability Symposium. 1997 Proceedings, Annual, Philadelphia (USA)*, pages 57–63. IEEE, 1997.

[How60] R.A. Howard. *Dynamic programming and Markov process*. The M.I.T. Press, 1960.

[Jen96] F.V. Jensen. *An introduction to Bayesian networks*, volume 74. UCL press London, 1996.

[Kle56] SC Kleene. Representation of events in nearve nets and finite automata. *Automata Studies*, 34 :3–42, 1956.

[LL03] A. Lisnianski and G. Levitin. *Multi-state system reliability : assessment, optimization and applications*, volume 6. World Scientific Pub Co Inc, 2003.

[LS87] N.G. Leveson and J.L. Stolzy. Safety analysis using petri nets. *IEEE Transactions on Software Engineering*, 3 :386–397, 1987.

[MF76] P. Merlin and D. Farber. Recoverability of communication protocols–implications of a theoretical study. *IEEE Transactions on Communications*, 24(9) :1036–1043, 1976.

[Min93] S. Minato. Zero-suppressed bdds for set manipulation in combinatorial problems. In *30th Conference on Design Automation,Dallas (USA)*, pages 272–277. IEEE, 1993.

[Min01] S. Minato. Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer (STTT)*, 3(2) :156–170, 2001.

[MJ56] E.J. McCluskey Jr. Minimization of boolean functions. *Bell System Technical Journal*, 35 :1417–1444, 1956.

[MNPF95] J.A. McDermid, M. Nicholson, D.J. Pumfrey, and P. Fenelon. Experience with the application of hazop to computer-based systems. In *Computer Assurance, 1995. COMPASS'95. Systems Integrity, Software Safety and Process Security. Proceedings of the Tenth Annual Conference on*, pages 37–48. IEEE, 1995.

[MRL11] G. Merle, J.-M. Roussel, and J.-J. Lesage. Algebraic Determination of the Structure Function of Dynamic Fault Trees. *Reliability Engineering and System Safety*, 96(2) :267–277, 2011.

[MRLB10] G. Merle, J.M. Roussel, J.J. Lesage, and A. Bobbio. Probabilistic Algebraic Analysis of Fault Trees With Priority Dynamic Gates and Repeated Events. *IEEE Transactions on Reliability*, 59(1) :250–261, 2010.

[MS56] E.F. Moore and C.E. Shannon. Reliable circuits using less reliable relays. *Journal of the Franklin Institute*, 262(3) :191–208, 1956.

[Mur75] JD Murchland. Fundamental concepts and relations for reliability analysis of multi-state systems. *Reliability and Fault Tree Analysis, RE Barlow and JB Fussell (Eds), SIAM*, 1975.

[Pet66] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Technische Universität Darmstadt, 1966.

[Pet81] J.L. Peterson. Petri net theory and the modeling of systems. *PRENTICE-HALL, INC., ENGLEWOOD CLIFFS, NJ 07632, 1981, 290*, 1981.

[PPRd10] B. Perrot, T. Prosvirnova, A. Rauzy, and J.-P. Sahut d'Izarn. Arbres de défaillance dynamiques : une bibliothèque pour la nouvelle génération d'altarica. In E. Fadier, editor, *Actes du congrès LambdaMu'17 (actes électroniques)*, La Rochelle (France), October 2010. IMdR.

[PR99] G. Point and A. Rauzy. Altarica : constraint automata as a description language. *Journal européen des systèmes automatisés*, 33(8-9) :1033–1052, 1999.

[PR12] T. Prosvirnova and A. Rauzy. Système de transitions gardées : formalisme pivot de modélisation pour la sûreté de fonctionnement. In J.F. Barbet, editor, *Actes du Congrès Lambda-Mu 18*, Octobre 2012.

[Rau01] A. Rauzy. Mathematical foundations of minimal cutsets. *IEEE Transactions on Reliability*, 50(4) :389–396, 2001.

[Rau02] A. Rauzy. Mode automata and their compilation into fault trees. *Reliability Engineering and System Safety*, 78(1) :1–12, 2002.

[Rau08] A. Rauzy. Guarded transition systems : a new states/events formalism for reliability studies. *Journal of Risk and Reliability*, 222(4) :495–505, 2008.

[RD97] A. Rauzy and Y. Dutuit. Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within Aralia. *Reliability Engineering and System Safety*, 58(2) :127–144, 1997.

[Rei79] D.J. Reifer. Software failure modes and effects analysis. *Reliability, IEEE Transactions on*, 28(3) :247–249, 1979.

[TD04] Z. Tang and JB Dugan. Minimal cut set/Sequence generation for dynamic fault trees. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS Los Angeles (USA)*, pages 207–213, 2004.

[Tri01] K.S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., 2001.

[VGRH81] WE Vesely, FF Goldberg, NH Roberts, and DF Haasl. *Fault Tree Handbook*, volume NUREG-0492. Nuclear Regulatory Commission, 1981.

[VSD+02] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick III, and J. Railsback. Fault tree handbook with Aerospace Applications version 1.1. *NASA Office of Safety and Mission Assurance, NASA HQ*, 2002.

[Wat61] HA Watson. Launch control safety study. *Bell labs*, 1961.

[WBP07] M. Walker, L. Bottaci, and Y. Papadopoulos. Compositional temporal fault tree analysis. *Computer safety, reliability, and security*, pages 106–119, 2007.

[WMOSI12] P. Weber, G. Medina-Oliva, C. Simon, and B. Iung. Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25(4) :671–682, 2012.

[WP06] M. Walker and Y. Papadopoulos. Pandora : The time of priority-and gates. In *Information Control Problems in Manufacturing*, volume 12, pages 237–242, 2006.

[WP07] M. Walker and Y. Papadopoulos. Pandora 2 : The time of priority-or gates. In *Dependable Control of Discrete Systems*, volume 1, pages 25–30, 2007.

[WP09] M. Walker and Y. Papadopoulos. Qualitative temporal analysis : Towards a full implementation of the fault tree handbook. *Control Engineering Practice*, 17(10) :1115–1125, 2009.

[ZST99] X. Zang, N. Sun, and K.S. Trivedi. A bdd-based algorithm for reliability analysis of phased-mission systems. *IEEE Transactions on Reliability*, 48(1) :50–60, 1999.

Annexes

Langages et automates finis : bases et notations utilisées dans le mémoire.

Un *alphabet* Σ est un ensemble fini de *lettres ou symboles* : $\Sigma = \{a, b\}$ est un alphabet dont a et b sont des symboles.

Un *mot* (ou *séquence*) est une concaténation d'un nombre fini de lettre : a, b, aa, bb, ba, ab sont des mots sur l'alphabet Σ. La longueur d'un mot u est notée $|u|$ et correspond au nombre de lettres composant le mot : $u = aabba, |u| = 5$.ϵ est le mot vide de symbole.

La *concaténation* est un opérateur associatif, non commutatif entre symboles ou mots dont ϵ est l'élément neutre. Soient u,v et w trois mots alors : $u(vw) = (uv)w$ et $uv \neq vu$.

Un ensemble de mots constitue un *Langage* : $\mathcal{L} = \{\epsilon, ab, aabb\}$ est un langage sur Σ.

Le langage comportant tous les mots de longueur n qu'il est possible de construire à l'aide des lettres de Σ est noté Σ^n : $\Sigma^0 = \{\epsilon\}, \Sigma^1 = \{a, b\}$.

Le langage comportant tous les mots qu'il est possible de construire à l'aide des lettres de Σ est noté Σ^*. Cette notation utilise la fermeture itérative (ou étoile) de Kleene (*). Ce langage est le résultat de la concaténation à l'infini des Σ^n : $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3$.... La séquence ϵ est élément de Σ^*.

L'ensemble des séquences appartenant à un langage $\mathcal{L}_1$ et n'appartenant pas à un langage $\mathcal{L}_2$ est noté $\mathcal{L}_1 \setminus \mathcal{L}_2$.

Un mot u est dit *préfixe* d'un mot v ($u \in Pref(v)$) s'il existe un mot $w \in \Sigma^*$ tel que $uw = v$.

Un *langage régulier (ou rationnel)* sur un alphabet Σ est un langage obtenu à partir des langages singletons sur Σ et d'un nombre fini d'unions, de concaténations et de fermetures itératives.

Le *Théorème de Kleene* établit une équivalence entre les langages réguliers et les langages représentables par une expression régulière ou par automate fini. Autrement dit, *il existe au moins un automate fini qui représente un langage régulier donné.*

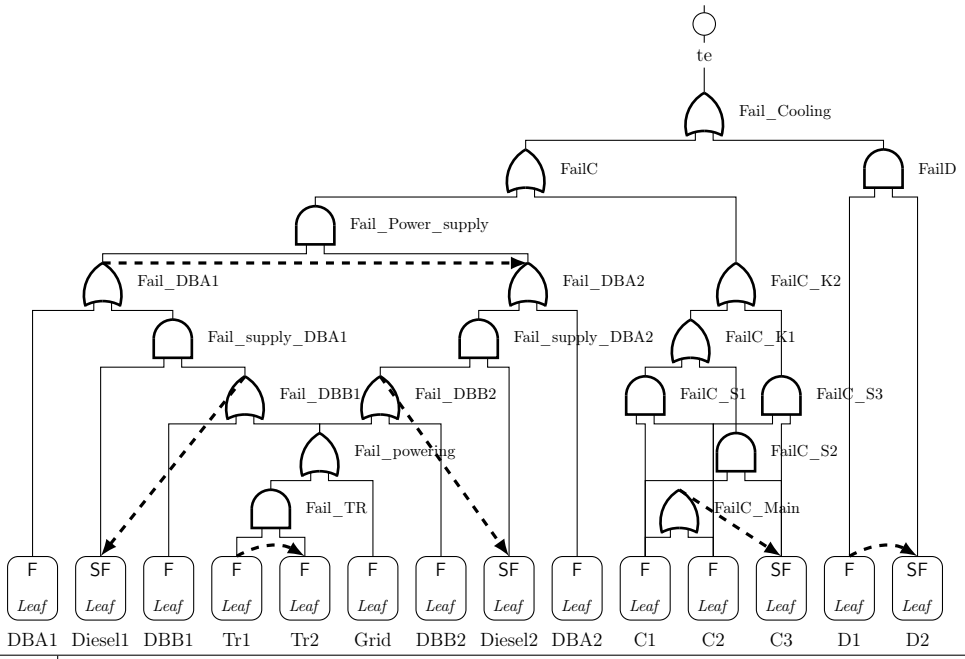
Le langage généré par un automate fini est l'ensemble des mots reconnus par cet automate.

Le langage marqué par un automate fini est l'ensemble des mots reconnus par cet automate permettant d'atteindre un état marqué à partir d'un état initial.

Un automate fini peut être défini par le 5-uplet suivant : $< \Sigma, Q, Q_0, Q_M, \delta >$ où :

- Σ est l'alphabet des symboles d'entrée.
- Q est l'ensemble des états de l'automate.
- Q_0 est l'ensembles des états initiaux de l'automate.
- Q_M est l'ensembles des état marqués de l'automate.
- δ est la fonction de transition.

Séquences de coupe minimales de l'exemple du chapitre 4 de longeur inférieure ou égale à 6.



Longueur	Séquence
2	$f_{DBA1,a}f_{DBA2,a}$
	$f_{C3,d}f_{C2,a}$
	$f_{C3,d}f_{C1,a}$
	$f_{C2,a}f_{C3,a}$
	$f_{C2,a}f_{C1,a}$
	$f_{C1,a}f_{C3,a}$
	$f_{C1,a}f_{C2,a}$
	$f_{D2,d}f_{D1,a}$
	$f_{D1,a}f_{D2,a}$
3	$f_{Diesel1,d}f_{Diesel2,d}f_{Grid,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{DBA2,a}$
	$f_{Diesel1,d}f_{Grid,a}f_{Diesel2,a}$
	$f_{Diesel1,d}f_{Grid,a}f_{DBA2,a}$
	$f_{Diesel2,d}f_{Diesel1,d}f_{Grid,a}$
	$f_{Diesel2,d}f_{DBA1,a}f_{DBB2,a}$
	$f_{Diesel2,d}f_{DBA1,a}f_{Grid,a}$
	$f_{Diesel2,d}f_{Grid,a}f_{Diesel1,a}$

	$f_{Diesel2,d}f_{Grid,a}f_{DBA1,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBA2,a}$
	$f_{DBA1,a}f_{Diesel2,d}f_{DBB2,a}$
	$f_{DBA1,a}f_{Diesel2,d}f_{Grid,a}$
	$f_{DBA1,a}f_{DBB2,a}f_{Diesel2,a}$
	$f_{DBA1,a}f_{Grid,a}f_{Diesel2,a}$
	$f_{Grid,a}f_{Diesel1,a}f_{Diesel2,a}$
	$f_{Grid,a}f_{Diesel1,a}f_{DBA2,a}$
	$f_{Grid,a}f_{Diesel2,d}f_{Diesel1,a}$
	$f_{Grid,a}f_{Diesel2,d}f_{DBA1,a}$
	$f_{Grid,a}f_{DBA1,a}f_{Diesel2,a}$
4	$f_{Diesel1,d}f_{Diesel2,d}f_{DBB1,a}f_{DBB2,a}$
	$f_{Diesel1,d}f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{Diesel2,d}f_{DBB2,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}f_{Diesel2,a}$
	$f_{Diesel1,d}f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Diesel1,d}f_{Tr1,a}f_{Tr2,a}f_{Diesel2,a}$
	$f_{Diesel1,d}f_{Tr1,a}f_{Tr2,a}f_{DBA2,a}$
	$f_{Diesel2,d}f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}$
	$f_{Diesel2,d}f_{Diesel1,d}f_{Tr1,a}f_{Tr2,a}$
	$f_{Diesel2,d}f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}$
	$f_{Diesel2,d}f_{DBB1,a}f_{Diesel1,a}f_{Grid,a}$
	$f_{Diesel2,d}f_{Tr1,a}f_{Diesel1,d}f_{Tr2,a}$
	$f_{Diesel2,d}f_{Tr1,a}f_{DBA1,a}f_{Tr2,a}$
	$f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}$
	$f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}f_{DBA1,a}$
	$f_{Diesel2,d}f_{DBA1,a}f_{Tr1,a}f_{Tr2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Diesel2,d}f_{DBB2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Diesel2,d}f_{Grid,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}f_{Diesel2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Grid,a}f_{Diesel2,a}$
	$f_{DBB1,a}f_{Diesel2,d}f_{Diesel1,a}f_{DBB2,a}$
	$f_{DBB1,a}f_{Diesel2,d}f_{Diesel1,a}f_{Grid,a}$

	$f_{Tr1,a}f_{Diesel1,d}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{Diesel1,d}f_{Tr2,a}f_{Diesel2,a}$
	$f_{Tr1,a}f_{Diesel1,d}f_{Tr2,a}f_{DBA2,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{Diesel1,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{DBA1,a}f_{Tr2,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}f_{Diesel1,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}f_{DBA1,a}$
	$f_{Tr1,a}f_{DBA1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{DBA1,a}f_{Tr2,a}f_{Diesel2,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}f_{Diesel2,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}f_{DBA2,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel2,d}f_{Diesel1,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel2,d}f_{DBA1,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{DBA1,a}f_{Diesel2,a}$
	$f_{DBA1,a}f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}$
	$f_{DBA1,a}f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{DBA1,a}f_{Tr1,a}f_{Tr2,a}f_{Diesel2,a}$
5	$f_{Diesel2,d}f_{DBB1,a}f_{Diesel1,a}f_{Tr1,a}f_{Tr2,a}$
	$f_{Diesel2,d}f_{DBB1,a}f_{Tr1,a}f_{Diesel1,a}f_{Tr2,a}$
	$f_{Diesel2,d}f_{Tr1,a}f_{DBB1,a}f_{Diesel1,a}f_{Tr2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{Tr1,a}f_{Tr2,a}f_{Diesel2,a}$
	$f_{DBB1,a}f_{Diesel2,d}f_{Diesel1,a}f_{Tr1,a}f_{Tr2,a}$
	$f_{DBB1,a}f_{Diesel2,d}f_{Tr1,a}f_{Diesel1,a}f_{Tr2,a}$
	$f_{DBB1,a}f_{Tr1,a}f_{Diesel1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{DBB1,a}f_{Tr1,a}f_{Diesel1,a}f_{Tr2,a}f_{Diesel2,a}$
	$f_{DBB1,a}f_{Tr1,a}f_{Diesel2,d}f_{Diesel1,a}f_{Tr2,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{DBB1,a}f_{Diesel1,a}f_{Tr2,a}$
	$f_{Tr1,a}f_{DBB1,a}f_{Diesel1,a}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{DBB1,a}f_{Diesel1,a}f_{Tr2,a}f_{Diesel2,a}$
	$f_{Tr1,a}f_{DBB1,a}f_{Diesel2,d}f_{Diesel1,a}f_{Tr2,a}$
	$\mathbf{f_{DBA1,a}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{DBA1,a}}$ **(événement de réparation)**

	$\mathbf{f_{Grid,a}f_{Diesel1,a}r_{Grid}f_{Diesel2,d}f_{Grid,a}}$ **(événement de réparation)**
6	$f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}r_{diesel1}f_{Diesel2,d}f_{Diesel1,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}r_{diesel1}f_{Diesel2,d}f_{DBA1,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}r_{DBB1}f_{Diesel2,d}f_{DBB1,a}$
	$f_{Diesel1,d}f_{DBB1,a}f_{DBB2,a}r_{DBB1}f_{Diesel2,d}f_{DBA1,a}$
	$f_{Diesel1,d}f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Diesel1,d}f_{DBA1,a}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{DBB1,a}$
	$f_{Diesel1,d}f_{Grid,a}f_{DBB2,a}r_{Grid}f_{Diesel2,d}f_{DBB1,a}$
	$\mathbf{f_{Diesel1,d}f_{Grid,a}f_{DBB2,a}r_{Grid}f_{Diesel2,d}f_{DBA1,a}}$
	(La SCM $f_{Grid,a}f_{Diesel2,d}f_{DBA1,a}$ **est incluse.)**
	$f_{Diesel2,d}f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel1,d}f_{Tr1,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}r_{diesel1}f_{Diesel2,d}f_{Diesel1,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}r_{diesel1}f_{Diesel2,d}f_{DBA1,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}r_{DBB1}f_{Diesel2,d}f_{DBB1,a}$
	$f_{DBB1,a}f_{Diesel1,a}f_{DBB2,a}r_{DBB1}f_{Diesel2,d}f_{DBA1,a}$
	$f_{DBB1,a}f_{DBA1,a}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{Diesel1,a}$
	$f_{Tr1,a}f_{Diesel1,d}f_{Tr2,a}r_{Tr1}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Diesel2,d}f_{Tr2,a}r_{Tr1}f_{Diesel1,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{DBA1,a}f_{Tr2,a}r_{Tr1}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Grid,a}f_{Diesel1,a}r_{Grid}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}r_{Tr1}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}r_{Tr1}f_{Diesel2,d}f_{Grid,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}r_{Tr2}f_{Diesel2,d}f_{Grid,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel1,a}r_{Tr2}f_{Diesel2,d}f_{Tr2,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{Diesel2,d}r_{Tr1}f_{Diesel1,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel1,d}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel1,d}f_{Tr1,a}f_{Diesel2,a}$
	$f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel1,d}f_{Tr1,a}f_{DBA2,a}$
	$f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{Diesel2,d}f_{Diesel1,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Tr2,a}r_{Tr1}f_{DBA1,a}f_{Diesel2,d}f_{Tr1,a}$
	$f_{Tr1,a}f_{Tr2,a}f_{DBA1,a}r_{Tr1}f_{Diesel2,d}f_{Tr1,a}$
	$f_{DBA1,a}f_{Diesel1,d}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{DBB1,a}$
	$f_{DBA1,a}f_{DBB1,a}f_{DBB2,a}r_{DBA1}f_{Diesel2,d}f_{Diesel1,a}$

	$f_{DBA1,a} f_{DBB2,a} f_{Diesel1,d} r_{DBA1} f_{Diesel2,d} f_{DBB1,a}$
	$f_{DBA1,a} f_{DBB2,a} f_{DBB1,a} r_{DBA1} f_{Diesel2,d} f_{Diesel1,a}$
	$f_{DBA1,a} f_{DBB2,a} r_{DBA1} f_{Diesel1,d} f_{Diesel2,d} f_{DBB1,a}$
	$f_{DBA1,a} f_{DBB2,a} r_{DBA1} f_{Diesel2,d} f_{Diesel1,d} f_{DBB1,a}$
	$f_{DBA1,a} f_{DBB2,a} r_{DBA1} f_{Diesel2,d} f_{DBB1,a} f_{Diesel1,a}$
	$f_{DBA1,a} f_{DBB2,a} r_{DBA1} f_{DBB1,a} f_{Diesel2,d} f_{Diesel1,a}$
	$f_{DBA1,a} f_{Tr1,a} f_{Tr2,a} r_{Tr1} f_{Diesel2,d} f_{Tr1,a}$
	$f_{Grid,a} f_{Diesel1,a} f_{DBB1,a} r_{Grid} f_{Diesel2,d} f_{DBB2,a}$
	$f_{Grid,a} f_{Diesel1,a} f_{DBB2,a} r_{Grid} f_{Diesel2,d} f_{DBB1,a}$
	$\mathbf{f_{Grid,a} f_{Diesel1,a} f_{DBB2,a} r_{Grid} f_{Diesel2,d} f_{DBA1,a}}$, **(La SCM** $f_{Grid,a} f_{Diesel2,d} f_{DBA1,a}$ **est incluse.)**
	$f_{Grid,a} f_{Diesel1,a} f_{Tr1,a} r_{Grid} f_{Diesel2,d} f_{Tr2,a}$
	$f_{Grid,a} f_{Diesel1,a} r_{Grid} f_{Diesel2,d} f_{DBB1,a} f_{DBB2,a}$
	$f_{Grid,a} f_{Diesel1,a} r_{Grid} f_{Diesel2,d} f_{Tr1,a} f_{Tr2,a}$
	$f_{Grid,a} f_{Diesel1,a} r_{Grid} f_{DBB1,a} f_{Diesel2,d} f_{DBB2,a}$
	$f_{Grid,a} f_{Diesel1,a} r_{Grid} f_{Tr1,a} f_{Diesel2,d} f_{Tr2,a}$
	$f_{Grid,a} f_{Tr1,a} f_{Diesel1,a} r_{Grid} f_{Diesel2,d} f_{Tr2,a}$

Résumé - L'analyse prévisionnelle des risques d'un système complexe repose aujourd'hui sur une modélisation de la dynamique du système vis-à-vis des défaillances et réparations de ses composants. L'analyse qualitative d'un tel système consiste à rechercher et à analyser les scénarios conduisant à la panne. En raison de leur nombre très important, il est courant de ne s'intéresser qu'aux scénarios les plus caractéristiques : les Séquences de Coupe Minimales (SCM). L'absence de formalisation de ces SCM a favorisé l'émergence de définitions spécifiques à certains outils de modélisation et de définitions informelles. Les travaux présentés dans cette thèse proposent : *i)* un cadre et une définition formelle des séquences de coupe minimales, tout deux indépendants de l'outil de modélisation de fiabilité utilisé, *ii)* une méthode permettant leur calcul, méthode basée sur des propriétés déduites de leur définition, *iii)* l'extension des premières définitions aux composants multi-modes. Ce cadre permet le calcul des SCM pour des installations décrites avec les Boolean logic Driven Markov Processes (BDMP). Sous l'hypothèse que l'ensemble des scénarios représentés implicitement via le modèle de sûreté établi peut être modélisé à l'aide d'un automate fini, ces travaux définissent la notion de cohérence des systèmes dynamiques et réparables comme le moyen d'obtenir une représentation minimale de l'ensemble des scénarios menant à la défaillance du système.

Mots Clef - Sûreté de fonctionnement, Etudes qualitatives, Systèmes dynamiques et réparables, Séquences de coupe minimales, BDMP.

Abstract - Preventive risk assessment of a complex system relies on a dynamic models which describe the link between the system failure and the scenarios of failure and repair events from its components. The qualitative analysis of a binary dynamic and repairable system is aiming at compute and analyse the scenarios that lead to the system failure. Since such systems describe a large set of scenarios, only the most representative ones, called Minimal Cut Sequences (MCS), are of interest for the safety engineer. The lack of a formal definition for the MCS promoted the emergence of multiple definitions either specific to a given model (and thus not generic) or informal. This work proposes *i)* a formal framework and definition for the MCS, independent of the reliability model used, *ii)* a method to compute them using property extracted from their formal definition, *iii)* an extension of the formal framework for multi-states components in order to perform the qualitative analyses of Boolean logic Driven Markov Processes (BDMP) models. Under the hypothesis that the scenarios implicitly described by any reliability model can always be represented by a finite automaton, this work defines the coherency for dynamic and repairable systems as the way to give a minimal representation of all scenarios that are leading to the system failure.

Keywords - System safety, Qualitative studies, Repairable systems, Qualitative analyses, Minimal Cut Sequences, BDMP.