Contribution à l'analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables

Pierre-Yves Piriou

NNT : 2015SACLN012

THESE DE DOCTORAT
DE L'UNIVERSITE PARIS-SACLAY,
préparée à l'ENS Cachan

ÉCOLE DOCTORALE N° 580
Sciences et Technologies de l'Information et de la Communication

Spécialité *Electronique, Electrotechnique, Automatique*

Par

Monsieur Pierre-Yves Piriou

Contribution à l'analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables

Thèse présentée et soutenue à *Cachan*, le *27 novembre 2015*.

Composition du Jury :

M[me] Kanoun Karama	Directeur de Recherche, LAAS–CNRS	Présidente
M. Niel Eric	Professeur, INSA de Lyon – Ampère	Rapporteur
M. Pétin Jean-François	Professeur, ENSEM – CRAN	Rapporteur
M. Bouissou Marc	Chercheur Senior HDR, EDF R&D	Examinateur
M. Faure Jean-Marc	Professeur, SUPMECA Paris – LURPA	Directeur de thèse
M. Lesage Jean-Jacques	Professeur, ENS Cachan – LURPA	Co-directeur de thèse
M[me] Devic Catherine	Manager, EDF R&D	Invitée

Remerciements

Je tiens tout d'abord à remercier sincèrement les membres de mon jury pour avoir accepté d'examiner le résultat de mes trois années de recherche au LURPA. Un merci tout particulier à M. Eric Niel et M. Jean-François Pétin qui ont eu la lourde tâche de se plonger dans les 200 pages de ce mémoire et qui m'ont gratifié d'un rapport très complet et pertinent sur celui-ci.

Je remercie également M. Jean-Marc Faure et M. Jean-Jacques Lesage pour la qualité de leur encadrement, pour l'intérêt qu'ils ont témoigné à mes idées, et pour tous les échanges scientifiques qui ont animé nos réunions de travail ces trois dernières années, me permettant de clarifier mes idées.

Je n'oublie évidemment pas tous les membres du LURPA (anciens et nouveaux, permanents et doctorants, Nol@g-iens et CIVIL-iens...) pour la bonne ambiance qui y règne. Un grand merci également à Serge, Claudine, Marc, Patrice et Philippe pour leur aide inestimable.

C'est également grâce aux différents échanges que j'ai pu avoir avec mes collègues du projet CONNEXION que les idées se sont petit à petit développées. J'aimerais donc les remercier pour cela. Je remercie également le projet CONNEXION et ses managers pour le financement de ma thèse et pour la confiance qu'ils m'ont accordée.

Ma famille et mes amis comptent énormément pour moi, et je les remercie pour leur soutien, leurs encouragements et de manière générale tout ce qu'ils ont toujours fait pour moi. J'ai également une pensée pour Cécile Hubert, ma bien aimée et aimante grand-mère, qui hélas ne m'aura jamais connu docteur.

Enfin, je remercie amoureusement Chloë, la femme de mes jours et de mes nuits, pour son aide précieuse dans la relecture de mon mémoire, pour sa musique essentielle à mon équilibre et pour tous les instants qu'elle partage avec moi et nos petites boules de poils sur pattes : Sirius et nos regrettés Igor et Ludwig.

Liste d'écoute

La Musique est une composante essentielle à mon équilibre. La vie aurait pour moi peu de saveur si je ne pouvais l'écouter, la pratiquer et la partager. Voici une liste d'œuvres triées sur le volet parmi celles qui ont accompagné la rédaction de ce mémoire.

- *Sonates du Rosaire* (C90-C105) de H.I.F. Bieber, 1678
- *Les trilles du diable* de G. Tartini, *c.*1713
- *Partita n°2 pour violon seul* (BWV1004) de J.S. Bach, 1717-1723
- *Les Indes galantes* de J.P. Rameau, 1735-1736
- *Passion selon Saint Matthieu* (BWV244) de J.S. Bach, 1736
- *Stabat Mater* de G.B. Pergolesi, 1736
- *Les Eléments* de J.F. Rebel, 1737
- *Fandango* (R146) de Padre A. Soler 1729-1783
- *La Belle Hélène* de J. Offenbach, 1864
- *Concerto pour violon en ré mineur* (op. 47) de J. Sibelius, 1905
- *Un boeuf sur le toit* (op. 58) de D. Milhaud, 1919
- *Les Planètes* (op.32) de G. Holst, 1914-1917
- *Prélude, Arioso et Fughette sur le nom de BACH* (H81) de A. Honegger, 1932
- *Symphonie n°9* (op. 70) de D. Chostakovitch, 1945
- *Las cuatro estaciones porteñas* de A. Piazzola, 1965-1970
- *Danzón n°2* de A. Márquez, 1994

Table des matières

Table des matières v

Table des figures ix

Liste des tableaux xv

Introduction Générale 1

1 Positionnement scientifique 5

1.1 Introduction . . . 7

1.2 Cadre général . . . 7

1.2.1 Analyse de Sûreté de Fonctionnement basée sur les modèles d'un système critique . . . 7

1.2.2 Classe de systèmes considérée . . . 10

1.2.3 Système reconfigurable . . . 13

1.3 Modélisation pour la sûreté de fonctionnement . . . 14

1.3.1 Modélisation structurelle . . . 15

1.3.2 Modélisation comportementale . . . 17

1.3.3 Modélisation multi-vues . . . 19

1.4 Méthodes d'analyse . . . 23

1.4.1 Analyse qualitative . . . 24

1.4.2 Analyse quantitative . . . 25

1.5 Bilan et contributions . . . 28

2 Modélisation par BDMP Généralisé 31

2.1 Introduction . . . 33

2.2 Capacités et limites de la modélisation BDMP . . . 33

2.2.1 Rappel de la définition des BDMP . . . 34

2.2.2 Exemples de modélisation BDMP . . . 37

2.2.3 Modélisation des mécanismes de commutation en BDMP 41
2.2.4 Bilan sur la modélisation BDMP 47
2.3 Définition des BDMP Généralisés 47
2.3.1 Définition de la syntaxe 48
2.3.2 Définition de la sémantique 54
2.3.3 Algorithme de simulation à événements discrets d'un modèle GBDMP 61
2.4 Construction de modèles GBDMP 67
2.4.1 Méthodologie générale 67
2.4.2 Exemples de modélisation GBDMP 73
2.5 Bilan 86

3 Analyse d'un modèle GBDMP 89
3.1 Introduction 91
3.2 Analyse qualitative 91
3.2.1 Définition des Séquences de Coupe Minimales d'un système dynamique, réparable et reconfigurable 92
3.2.2 Calcul de l'ensemble des SCM à partir d'un modèle GBDMP . . . 99
3.2.3 Etude comparative 107
3.3 Analyse quantitative 113
3.3.1 Méthode de construction automatique de chaîne de Markov limitée 114
3.3.2 Traduction d'un modèle GBDMP en un modèle AltaRica 116
3.3.3 Etude comparative 123
3.4 Bilan 128

4 Application : partie d'une centrale nucléaire à eau pressurisée 131
4.1 Introduction 133
4.2 SAGE : un outil pour l'édition et la manipulation des modèles 133
4.3 Description du processus de conception 136
4.4 Description du cas 139
4.4.1 Le procédé physique 140
4.4.2 Le contrôle-commande 143
4.4.3 Analyse préliminaire des risques 146
4.4.4 Maintenance des échangeurs 147
4.5 Modélisation GBDMP du système 148

4.5.1 Construction des PMC 148
4.5.2 Modélisation de l'architecture opérationnelle de CC 150
4.5.3 Construction de la structure du GBDMP pour la partie procédé . 151
4.5.4 Construction des machines de Moore 152
4.5.5 Modèle complet 157
4.6 Analyse 160
4.6.1 Identification de scénarios critiques 160
4.6.2 Disponibilité des fonctions 163
4.7 Bilan 167

Conclusions & Perspectives **169**

Notations **175**

Bibliographie **179**

A Théorie des graphes : définitions et notations **191**

B Niveaux de profondeur des sommets d'un modèle GBDMP **195**

C Théorie des langages : définitions et notations **197**

D Exempe de traduction de modèle GBDMP vers un modèle AltaRica **199**

Table des figures

1 Représentation la plus simplifiée d'un système bouclé. 1

1.1 Exemple très simple de système dynamique 10
1.2 Exemple de d'ADD modélisant la SdF d'un système d'assistance cardiaque ([Boudali and Dugan, 2005]) 16
1.3 Exemple de chaîne de Markov pour deux composants 18
1.4 RdPSG modélisant une cellule flexible faillible [Ajmone Marsan et al., 1994] 19
1.5 Traduction d'une porte PAND en un RdPS [Codetta-Raiteri, 2005] . . . 20
1.6 Exemple de modèle AltaRica pour un système simple d'alimentation d'eau (extrait de [Prosvirnova et al., 2013]) 21
1.7 Vues *bloc* (à gauche) et *arbre de défaillance* (à droite) d'un modèle HiP-HOPS d'une redondance passive simple 22
1.8 Indisponibilité d'un système en fonction du temps, évaluée par simulation de Monte Carlo (extrait de [Durga Rao et al., 2009]) 27

2.1 Exemple de modèle BDMP. a) vue structurelle ; b) vue comportementale 35
2.2 Trois exemples de PMP caractéristiques (de gauche à droite : F, SF et SF_I [1]) 37
2.3 Modélisation d'une redondance passive 38
2.4 Report de charge entre deux pompes 39
2.5 Modélisation d'un report de charge 39
2.6 Modélisation d'un système multi-phases 40
2.7 Tentative de prise en compte d'un composant responsable de la commutation entre deux composants 42
2.8 Tentative de modélisation BDMP de composants multi-états 44
2.9 Deux vannes réalisant un mélange en deux phases 45
2.10 Modélisation BDMP d'un système multi-phases 46

2.11 Exemple didactique de modèle GBDMP ; a) vue *arbre de défaillance enrichi* ; b) vue *PMC* ; c) vue *machine de Moore* 49
2.12 Principe de l'utilisation d'une machine de Moore pour spécifier une logique de commande 52
2.13 Machine de Moore modélisant une gâchette de BDMP 53
2.14 Exemple d'évolution d'un modèle GBDMP entre deux états stables . . . 61
2.15 Graphe des dépendances entre les variables de l'exemple didactique . . . 62
2.16 Contre exemple de modèle GBDMP ne respectant pas la règle 7. 63
2.17 Evolution simple d'un modèle GBDMP, déterminée par l'algorithme 1 . . 66
2.18 Méta-modèle de données issues d'une analyse préliminaire du fonctionnement et des risques de dysfonctionnement d'un système (extrait de [Piriou et al., 2014a]). 70
2.19 Extrait de modèle GBDMP illustrant la propagation des facteurs permettant de déterminer le mode d'opération des PMC 71
2.20 Deux groupes de composants redondants, selon deux stratégies différentes 74
2.21 Arbre de défaillance modélisant la structure du système décrite par la Figure 2.20 75
2.22 PMC de type SF 76
2.23 PMC de type Co modélisant le comportement des composants $D1$ et $D2$ 76
2.24 Machines de Moore spécifiant deux stratégies de reconfiguration (en haut [2] : $M1 = str(S1)$; en bas : $M2 = str(S2)$). 78
2.25 Vue *arbre de défaillance enrichi* du modèle GBDMP pour le système faisant intervenir deux stratégies de redondance différentes appliquées par des composants faillibles 79
2.26 Arbre de défaillance modélisant la structure du système pour le cas du report de charge 82
2.27 PMC Po modélisant le comportement des pompes P_1 et P_2 82
2.28 Vue *arbre de défaillance enrichi* du modèle GBDMP des pompes multi-états 83
2.29 Arbre de défaillance modélisant la structure du système pour le cas des vannes bloquantes 84
2.30 PMC Va modélisant le comportement des vannes V_1 et V_2 85
2.31 PMC Mi générant les événements de changement de phase 85

2.32 Machine de Moore spécifiant la stratégie de reconfiguration $M3$, pour piloter le changement de phase. 85
2.33 Vue *arbre de défaillance enrichi* du modèle GBDMP des vannes bloquantes 86

3.1 Modèle de deux composants en redondance passive 93
3.2 Modèle de deux composants en redondance passive et d'un composant responsable de la commutation 94
3.3 Automate à état modélisant le comportement dysfonctionnel d'un système bi-phasé à deux composants 98
3.4 Exemple simple de modèle GBDMP 100
3.5 Construction partielle de l'automate équivalent au GBDMP proposé. . . 104
3.6 Système hydraulique, représenté avec son alimentation électrique 108
3.7 Modèle BDMP $\mathcal{M}_1$ du système hydraulique et de son alimentation électrique ([Chaux et al., 2012]) 109
3.8 Modèle GBDMP $\mathcal{M}_2$ du système hydraulique et de son alimentation électrique 110
3.9 Machines de Moore spécifiant des stratégies de reconfiguration "au plus tard", au succès non garanti, pour une redondance 2 parmi 3 : M5 (en haut), et 1 parmi 2 : M6 (en bas). 110
3.10 Principe de la construction de Chaîne de Markov limitée [Brameret, 2015] 114
3.11 Agrégation des états non explorés dans un état puits [Brameret et al., 2015] 116
3.12 Première version du modèle : $\mathcal{M}_0$ 124
3.13 Deuxième version du modèle : $\mathcal{M}_1$ 125
3.14 Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}_0$ et $\mathcal{M}_1$ 126
3.15 Machine de Moore caractérisant la troisième version du modèle $\mathcal{M}_2$ . . . 126
3.16 Machine de Moore caractérisant la quatrième version du modèle $\mathcal{M}_3$ . . . 127
3.17 Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}_1$, $\mathcal{M}_2$ et $\mathcal{M}_3$ 127
3.18 Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}'_1$, $\mathcal{M}'_2$ et $\mathcal{M}'_3$ 128

4.1 Copie d'écran du simulateur graphique de l'outil SAGE 134
4.2 Copie d'écran de la vue *PMC*, via le simulateur graphique de l'outil SAGE 135

4.3 Schéma du processus de conception d'une architecture opérationnelle de CC . . . 137
4.4 Les six systèmes élémentaires d'une partie de centrale nucléaire à eau pressurisé et leurs interconnexions . . . 138
4.5 Schéma mécanique du système élémentaire RRA . . . 141
4.6 Schéma mécanique du système élémentaire PTR . . . 142
4.7 Planche SCADE spécifiant l'UA1 (à droite, contenu d'une boîte de type SC2) [3]. . . . 144
4.8 Planche SCADE spécifiant l'UA2 [4]. . . . 144
4.9 Planches SCADE spécifiant l'UA4 (en haut) et l'UA5 (en bas) [5]. . . . 145
4.10 Schéma de l'architecture opérationnelle de CC . . . 146
4.11 PMC modélisant le comportement dysfonctionnel d'une pompe du système RRA (à gauche : SF) et du système PTR (à droite : Po) . . . 148
4.12 PMC modélisant les vannes manuelles (à gauche : VM) et les vannes automatisées (à droite : VA) . . . 149
4.13 PMC de type ET modélisant le comportement dysfonctionnel d'un échangeur thermiques . . . 149
4.14 PMC de type UC modélisant un composant de contrôle . . . 150
4.15 PMC de type Co modélisant la conduite de la centrale . . . 150
4.16 Vue *arbre de défaillance enrichi* du modèle GBDMP pour l'architecture opérationnelle de CC . . . 150
4.17 Structure du modèle GBDMP pour la partie procédé du système . . . 151
4.18 Commutateur S1 (contenant la machine de Moore M_{req1}) contrôlant la réquisition de la fonction RCP . . . 153
4.19 Commutateur S2 (contenant la machine de Moore M_{red1}) modélisant la redondance entre les pompes du système RRA . . . 153
4.20 Commutateur S3 (contenant la machine de Moore M_{sauv}) modélisant la stratégie de sauvegarde du RRA . . . 154
4.21 Commutateur S4 (contenant la machine de Moore M_{req2}) contrôlant la réquisition de la fonction RPR . . . 154
4.22 Commutateur S5a (contenant la machine de Moore M_{red2}) modélisant la redondance entre les pompes du système PTR . . . 155

4.23 Commutateur S6 (contenant la machine de Moore M_{mtn1}) modélisant la stratégie de maintenance des échangeurs de RRA 156
4.24 Commutateur S7 (contenant la machine de Moore M_{mtn2}) modélisant la stratégie de maintenance des échangeurs du PTR 156
4.25 Commutateur S8a (contenant la machine de Moore M_{flux1}) modélisant l'effet des vannes manuelles 157
4.26 Commutateur S9 (contenant la machine de Moore M_{flux2}) modélisant l'effet des vannes automatisées 157
4.27 Vue *PMC* du modèle GBDMP pour les systèmes PTR et RRA 158
4.28 Vue *Machine de Moore* du modèle GBDMP pour les systèmes PTR et RRA 158
4.29 Vue *arbre de défaillance enrichi* du modèle GBDMP pour les systèmes PTR et RRA 159
4.30 Estimation de l'indisponibilité de la fonction RPC 164
4.31 Estimation de l'indisponibilité de la fonction RPR 164
4.32 Estimation de l'indisponibilité de la fonction RCP 165

D.1 Modèle GBDMP à traduire 199
D.2 Traduction du modèle en langage AltaRica 201

Liste des tableaux

2.1 Correspondances entre les états de la vanne et ceux du PMP associé aux feuilles la modélisant 46
2.2 Principaux avantages et inconvénients du formalisme BDMP 47
2.3 Exemples d'interprétation de phrases type d'un corpus de texte décrivant une stratégie de reconfiguration. 73
2.4 Comportement décrit par le modèle pour un scénario impliquant les composants $C1a$, $C1b$ et $C1c$ 80
2.5 Comportement décrit par le modèle pour un scénario impliquant les composants $C2a$, $C2b$ et $D2$ 81
2.6 Comportement décrit par le modèle pour un scénario impliquant les composants P_0, P_1 et P_2 84
2.7 Apports du formalisme GBDMP par rapport au formalisme BDMP . . . 87

3.1 Extrait des résultats de l'analyse qualitative pour le système hydraulique 113

4.1 Ensemble des SCM (de longueur inférieure ou égale à 5) conduisant à la perte de la fonction RPC 161
4.2 Récapitulatif des paramètres et résultats du calcul de disponibilité 166

Introduction Générale

Cette thèse a été financée par le projet CONNEXION (COntrôle-Commande Nucléaire Numérique pour l'EXport et la rénovatION), qui a pour objectif de consolider et de développer l'avance technologique de la filière nucléaire française, en ce qui concerne les systèmes de Contrôle-Commande des centrales nucléaires, tant au niveau de la conception que de l'exploitation. Ce projet répond à la deuxième édition de l'appel à projets BGLE (Briques Génériques du Logiciel Embarqué), ayant pour finalité concrète la fourniture d'outils logiciels innovants. Les contributions du projet reposent sur une collaboration entre des fournisseurs de logiciels embarqués (Atos Worldgrid, Rolls-Royce Civil Nuclear, CORYS TESS, Esterel Technologies, All4Tec), des intégrateurs (EDF, AREVA, ALSTOM) et des partenaires académiques (CEA, INRIA, CNRS/CRAN, ENS Cachan, LIG, Telecom ParisTech). Dans le cadre de ce projet, l'auteur de la présente thèse est employé par le LURPA (Laboratoire Universitaire de Recherche en Production Automatisée) de l'ENS Cachan, afin de mener un travail de recherche pour contribuer à la validation de la Sûreté de Fonctionnement (SdF) d'un système bouclé (cf. Figure 1). Cet objectif s'inscrit dans une démarche d'amélioration continue de la sûreté de fonctionnement des centrales nucléaires.

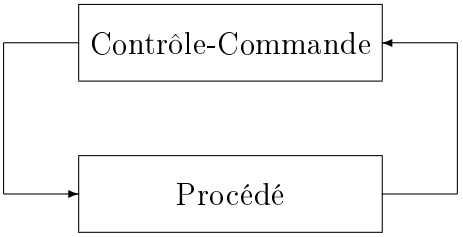

Figure 1 – Représentation la plus simplifiée d'un système bouclé.

Les contributions scientifiques sur la modélisation et l'analyse de la SdF des systèmes critiques sont nombreuses. Toutefois, les travaux existants se limitent classiquement à la prise en compte du *procédé* (ou *partie opérative*), le *Contrôle-Commande* (CC) étant ou bien supposé exempt de défaut (implicitement ou explicitement), ou bien considéré

indépendamment dans des études transverses. Il est important à ce stade, de distinguer le partitionnement présenté sur la Figure 1 d'un autre partitionnement usuel consistant à diviser un système complexe entre sa partie *matérielle* et sa partie *logicielle*. Il existe des techniques visant à garantir formellement la conformité du logiciel à sa spécification (model checking, theorem proving...). Ces techniques peuvent être utilisées afin d'éliminer les erreurs humaines, qui constituent l'unique cause de la défaillance du logiciel. En revanche, certaines défaillances de la partie matérielle sont de nature fondamentalement différente, et ne peuvent pas toujours être imputées à la responsabilité humaine. Ces mécanismes de défaillance sont classiquement assimilés à des processus stochastiques. Or, une *architecture opérationnelle* de CC est, par définition, le résultat de l'allocation d'une *architecture fonctionnelle* sur une *architecture matérielle*. Aussi, les développements présentés dans cette thèse sont basés sur le constat que le comportement dysfonctionnel de la partie matérielle du CC a un impact sensible sur la SdF du système bouclé [Piriou et al., 2014b].

Par ailleurs, les formalismes et méthodes de modélisation de la SdF existants offrent généralement une appréhension basique et implicite des stratégies de reconfiguration du procédé (redondances passives, changements de phase de mission...). La prise en compte dans les analyses de SdF de la partie du CC ayant pour rôle d'appliquer ces stratégies implique une représentation explicite et précise de celles-ci. Une première approche dans ce sens est proposée dans [Piriou et al., 2015]. Ainsi, les objectifs de cette thèse pour l'analyse basée sur les modèles de la SdF se situent à deux niveaux :

- la prise en compte des défaillances (et réparations) matérielles du CC ;
- la prise en compte explicite et précise des stratégies de reconfiguration du procédé.

Ce mémoire de thèse est organisé en quatre parties visant à formaliser la problématique scientifique, à développer les contributions revendiquées en terme de modélisation et d'analyse et à les illustrer sur un cas d'étude représentatif des problématiques industriels.

La première partie précise la problématique scientifique faisant l'objet de la thèse, et fournit une synthèse de l'état de l'art sur la modélisation et l'analyse de la SdF des systèmes critiques. Cette synthèse introduit notamment le formalisme BDMP (Boolean logic Driven Markov Processes), qui a été développé chez EDF R&D pour la modélisation

des systèmes dynamiques et réparables. Nous verrons qu'il constitue une base pertinente sur laquelle appuyer nos travaux.

La seconde partie discute dans un premier temps les capacités des BDMP à prendre en compte les deux aspects énoncés ci-dessus. Dans un second temps, les principes caractérisant les BDMP sont étendus pour considérer les systèmes dynamiques, réparables et reconfigurables. Le formalisme résultant de cette extension est appelé BDMP Généralisé (GBDMP), et constitue la contribution majeure de cette thèse. Enfin, une réflexion méthodologique est proposée afin de guider la construction des modèles GBDMP.

La troisième partie décrit deux méthodes d'analyse d'un modèle GBDMP. La première est une méthode d'analyse qualitative et consiste à recherche de l'ensemble des Séquences de Coupe Minimales (SCM) du système. Il s'agit du plus petit ensemble nécessaire et suffisant pour représenter tous les scénarios d'événements conduisant à la première défaillance du système. La seconde méthode d'analyse est quantitative et permet de calculer des indicateurs probabilistes de SdF grâce à une génération limitée de chaîne de Markov. Cette méthode est directement issue des travaux développée dans [Brameret et al., 2015]. Nous nous limiterons donc à montrer comment elle peut être utilisée à partir d'un modèle GBDMP.

La quatrième partie présente un prototype d'outil logiciel à l'usage de l'ingénieur expert en SdF, jouant pour cette thèse un rôle de preuve de concept. L'analyse basée sur un modèle GBDMP d'un système bouclé (utilisant l'outil) est ensuite illustrée sur un cas d'étude (partie simplifiée d'une centrale nucléaire), construit en collaboration avec des partenaires industriels d'EDF R&D.

Enfin, des conclusions seront tirées de ce travail, et des perspectives pour le poursuivre seront proposées à la fin du mémoire.

Chapitre 1
Positionnement scientifique

Sommaire

1.1 Introduction 7

1.2 Cadre général 7

1.2.1 Analyse de Sûreté de Fonctionnement basée sur les modèles d'un système critique 7

1.2.1.1 Sûreté de Fonctionnement 7

1.2.1.2 Analyse de Sûreté de Fonctionnement 8

1.2.1.3 Analyse basée sur les modèles 9

1.2.2 Classe de systèmes considérée 10

1.2.2.1 Système dynamique 10

1.2.2.2 Système réparable 11

1.2.2.3 Composants multi-états 11

1.2.2.4 Système multi-phases 12

1.2.2.5 Système à fiabilité statique 12

1.2.3 Système reconfigurable 13

1.3 Modélisation pour la sûreté de fonctionnement 14

1.3.1 Modélisation structurelle 15

1.3.1.1 Diagrammes de fiabilité 15

1.3.1.2 Arbres de défaillance 16

1.3.2 Modélisation comportementale 17

1.3.2.1 Chaînes de Markov 17

1.3.2.2 Réseaux de Petri 18
1.3.3 Modélisation multi-vues 19
1.3.3.1 AltaRica 20
1.3.3.2 HiP-HOPS 21
1.3.3.3 BDMP 22
1.3.3.4 Autres approches 23
1.4 Méthodes d'analyse 23
1.4.1 Analyse qualitative 24
1.4.1.1 Approches algébriques 24
1.4.1.2 Approche par la théorie des langages 25
1.4.2 Analyse quantitative 25
1.4.2.1 Calcul exact sur chaîne de Markov 26
1.4.2.2 Estimation par simulation de Monte Carlo 26
1.4.2.3 Approximations astucieuses 27
1.5 Bilan et contributions 28

1.1 Introduction

Le travail de recherche reporté dans le présent mémoire s'intéresse à l'analyse de Sûreté de Fonctionnement (SdF) basée sur les modèles (*model based safety analysis* dans la littérature anglophone) des systèmes dynamiques réparables et reconfigurables. Dans ce chapitre, nous introduisons dans un premier temps les concepts élémentaires relatifs au socle scientifique et à l'objet d'étude de cette thèse. En particulier, nous précisons ce que l'on désigne par *analyse de SdF basée sur les modèles*, et par *systèmes dynamiques réparables et reconfigurables* (section 1.2). Puis, une synthèse bibliographique dans le domaine de la modélisation et de l'analyse de la SdF des systèmes critiques est présentée dans les sections 1.3 et 1.4. Enfin, les contributions revendiquées dans cette thèse seront synthétisées dans la section 1.5.

1.2 Cadre général

1.2.1 Analyse de Sûreté de Fonctionnement basée sur les modèles d'un système critique

1.2.1.1 Sûreté de Fonctionnement

La Sûreté de Fonctionnement désigne un ensemble de concepts, méthodes et outils permettant de qualifier et de quantifier le degré de confiance qu'il est légitime d'accorder à la capacité d'un système à réaliser correctement son service en temps voulu. Les activités de SdF visent d'une part à déterminer des indicateurs qualitatifs tels que des scénarios d'événements particulièrement critiques, ou des preuves de conformité du comportement nominal du système. Par ailleurs, elles ont également pour objectif d'estimer des attributs quantifiés du système étudié comme la *fiabilité* (*reliability* en anglais, probabilité que le système soit opérationnel depuis le début de sa mission), la *disponibilité* (*availability* en anglais, probabilité que le système soit opérationnel à un instant donné), ou encore le temps moyen avant une défaillance (*Mean Time To Failure* en anglais)... Les concepts de base et la terminologie associés au concept de SdF (pour la langue française) peuvent être consultés dans les deux ouvrages de référence [Laprie et al., 1995] et [Villemeur, 1988].

Un moyen répandu pour améliorer la SdF d'un système est la *redondance*. Le principe est de "sur-dimensionner" le système de manière à lui offrir une flexibilité lui permettant

de tolérer certaines défaillances que l'on sait fréquentes et/ou dangereuses. On distingue deux types de redondance :

- *redondance matérielle* : plusieurs composants peuvent réaliser la même fonction, et donc le même service ;
- *redondance fonctionnelle* : plusieurs solutions techniques peuvent assurer le même service.

Ces deux types de redondance peuvent être mises en place de deux manières :

- *redondance active* : toutes les solutions opèrent en permanence en parallèle ;
- *redondance passive* : le système se reconfigure dynamiquement pour adopter telle ou telle solution selon son état dysfonctionnel.

1.2.1.2 Analyse de Sûreté de Fonctionnement

Afin d'estimer la SdF d'un système, on procède à des analyses visant à caractériser son comportement dysfonctionnel. Ce processus peut se décomposer en deux phases, chaque phase ayant deux facettes complémentaires (qualitative et quantitative) :

- l'analyse préliminaire des risques ;
- l'analyse du comportement dysfonctionnel.

L'objectif de la première phase est de formaliser la connaissance de base sur les risques de dysfonctionnement des composants du système. La méthode la plus couramment retenue pour produire les résultats qualitatifs attendus dans cette phase est l'AMDEC (Analyse des Modes de Défaillances, de leurs Effets et de leur Criticité). Elle consiste à inventorier de manière systématique les différentes modalités de défaillance de chaque composant ainsi que leurs impacts sur le composant et son environnement (cf. [Ridoux, 1999] pour une description plus détaillée). En parallèle de cette analyse qualitative, il convient de quantifier les probabilités d'occurrence des événements de panne (et éventuellement de réparation) des composants le conduisant dans (ou le sortant de) ses différents modes de défaillance. Ces données quantitatives sont le plus souvent estimées grâce à des retours d'expériences (REX), mais peuvent, dans certains cas, être calculées théoriquement si on connaît les lois physiques de dégradation du composant

(par exemple [Meister and Persoz, 2000] pour les cuves de réacteur nucléaire, [Fukuda, 1991] pour des semi-conducteurs...).

La deuxième phase consiste à exploiter la connaissance produite par la première, en vue de comprendre le comportement dysfonctionnel du système étudié. D'un point de vue qualitatif, il s'agit d'identifier les scénarios d'événements (de panne, de réparation, ou plus généralement : événements significatifs pour la SdF) conduisant à la défaillance du système. D'un point de vue quantitatif, on cherche à estimer des indicateurs probabilistes de la SdF du système (*fiabilité*, *disponibilité*, *temps moyen avant la première défaillance*...). Les contributions revendiquées dans cette thèse concernent cette phase du processus d'analyse de la SdF d'un système critique. Une sélection de techniques usuelles d'analyse permettant d'atteindre les objectifs de cette phase sera présentée dans la section 1.4.

1.2.1.3 Analyse basée sur les modèles

Un modèle de système est une abstraction de celui-ci visant à représenter un ou plusieurs aspects de sa structure et/ou de son comportement, sous une forme idéalisée et approximative ([IEE, 1990]). La modélisation peut être vue comme une mise en forme de connaissances pour faciliter leur exploitation à des fins bien précises.

L'ingénierie basée sur les modèles est l'utilisation formalisée de modèles comme supports des activités liées aux processus de conception des systèmes, et ce pendant toutes les phases de vie du système [INC, 2007]. Ainsi, les approches méthodologiques associées proposent un cadre de conception permettant de capitaliser plusieurs aspects du système de manière *intégrée*. Les bénéfices attendus sont multiples et s'expriment en termes de maîtrise des coûts de développement, d'amélioration de la qualité des études, de la communication entre les acteurs du projet, etc...

Afin de faciliter l'intégration des activités de SdF dans les processus de conception, il est important que celles-ci s'inscrivent dans cette démarche d'ingénierie. Ainsi, les auteurs de [Joshi et al., 2006] définissent l'analyse de SdF basée sur les modèles comme une approche dans laquelle les ingénieurs système et les ingénieurs de SdF partagent des modèles construits selon un processus de développement commun. Ce travail a été conduit conformément à cette approche.

1.2.2 Classe de systèmes considérée

Le cadre de l'étude étant défini, la classe de systèmes étudiée dans ce travail doit maintenant être précisée : nous nous intéressons aux systèmes dynamiques, réparables, multi-états et multi-phases, à fiabilité statique.

1.2.2.1 Système dynamique

L'état dysfonctionnel d'un système est intimement lié à celui de ses composants. Pour certains systèmes, la connaissance de l'état dysfonctionnel de chaque composant est suffisante pour en déduire celui du système (on trouve cette acception par exemple dans [Birnbaum et al., 1961]). Ces systèmes sont qualifiés de *statiques* (au sens de la SdF).

En revanche pour les autres systèmes (*dynamiques*), cette information ne suffit pas. Il est nécessaire de connaître également l'ordre d'occurrence des événements conduisant le système dans l'état considéré (cette acception est introduite dans [Dugan et al., 1992]). En particulier, la distinction entre redondance active et passive n'a de sens que pour un système dynamique. En effet, pour un système statique, on ne pourra pas prendre en considération la modification du comportement dysfonctionnel d'un composant de secours en fonction de l'état dysfonctionnel du composant principal.

Un exemple de système dynamique très simple est représenté sur la Figure 1.1. L'interrupteur S n'est utilisé que pour remplacer $C1$ par $C2$ en cas de défaillance du premier. Pour ce système, le fait de savoir que S et $C1$ sont défaillants et que $C2$ ne l'est pas, sans connaître la séquence d'événement conduisant à cet état, ne permet pas de conclure que le système est défaillant ou non. En effet, si la défaillance de $C1$ survient après celle de S, la commutation sur $C2$ ne peut pas être réalisée et le service n'est plus assuré. En revanche, si la défaillance de S survient après celle de $C1$, le service peut être commuté normalement, et être correctement assuré par $C2$, tant qu'il n'est lui-même pas défaillant.

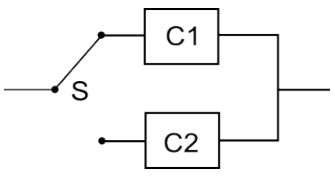

Figure 1.1 – Exemple très simple de système dynamique

Un autre exemple de phénomène ne pouvant être considéré que pour les systèmes dynamiques, est la défaillance à la sollicitation d'un composant [Meshkat et al., 2002]. Ces défaillances ne surviennent que suite à la demande d'activation du composant ; elles sont donc provoquées ou révélées par l'occurrence d'un autre événement.

1.2.2.2 Système réparable

Un système est dit réparable si au moins un de ses composants est réparable [Crow, 1975]. La réparation d'un composant est l'événement dual de sa défaillance. Ainsi, l'ensemble des scénarios décrivant le comportement dysfonctionnel d'un système réparable est infini.

La prise en compte de composants réparables implique souvent des difficultés dans la description du comportement du système. Par exemple, dans le cas proposé sur la Figure 1.1, le principe du *remplacement* de $C1$ par $C2$ est *a priori* simple et sans ambiguïté. Mais dans le cas où $C1$ est réparable, la stratégie de *rétablissement* à adopter est moins évidente : doit on rétablir le service sur $C1$ dès qu'il est réparé, ou attendre que $C2$ ne soit lui-même plus capable d'assurer le service ? Le problème s'intensifie encore si on considère également réparable l'interrupteur S qui réalise des commutations.

1.2.2.3 Composants multi-états

Dans de nombreux cas, l'état dysfonctionnel d'un composant peut être traduit par une simple variable binaire (le composant est défaillant ou ne l'est pas). Mais pour certains systèmes, cette information binaire est insuffisante. Les effets sur la SdF du système des différents états dysfonctionnels des composants sont alors considérés qualitativement et/ou quantitativement [Lisnianski and Levitin, 2003]. La prise en compte de composants multi-états permet notamment de modéliser la *dégradation* discrète d'un système pour nuancer la notion de défaillance. En outre, elle confère une granularité plus fine pour décrire le comportement dysfonctionnel du système.

Par la suite, pour caractériser l'état d'un composant, nous parlerons de *mode d'opération* : ce que le composant est sensé faire, et de *mode de défaillance* : quel est le type de défaillance du composant [1]. Aussi, les taux de défaillance et de réparation d'un composant varient selon ses modes d'opération et de défaillance. Par ailleurs, on peut

1. Pour des raisons d'homogénéité, l'inactivité et la non défaillance, peuvent être considérés respectivement comme un mode d'opération et un mode de défaillance.

étendre la notion de *défaillance à la sollicitation*, à celle, plus générale, de *défaillance à la commutation* (entre deux modes d'opération).

Par exemple, la *rupture* et la *fuite* d'une pompe sont deux modes de défaillance dont l'impact sur la réalisation du pompage est différent (dans le cas de la rupture, le pompage n'est plus du tout réalisé, alors que dans celui de la fuite, il est seulement dégradé). Un autre exemple parlant est celui de la vanne TOR (*Tout Ou Rien*) sujette à un blocage physique : bien que défaillante, elle réalise le service attendu ou non, selon qu'elle est bloquée en position ouverte ou fermée. Par ailleurs, dans le cas mentionné ci-dessus (cf. Figure 1.1), on a fait l'hypothèse que l'état dysfonctionnel de l'interrupteur S était binaire, et que l'effet de sa défaillance se traduisait par le figeage de sa position. Si on envisage d'autres modes de défaillance de cet interrupteur (par exemple un faux contact, une commutation intempestive...), le comportement dysfonctionnel du système s'en trouverait sensiblement modifié.

1.2.2.4 Système multi-phases

La mention "multi-phases" accolée à un système, indique qu'il doit réaliser une mission qui aura été décomposée en plusieurs phases. Le passage d'une phase à l'autre induit une modification de la structure du système et/ou de son comportement dysfonctionnel et/ou de ses critères de succès ([Burdick et al., 1977] et [Meshkat et al., 2003]). Un exemple représentatif de système multi-phase est l'avion, dont les composants seront utilisés différemment, durant les phases de décollage, de pilotage manuel, de pilotage automatique et d'atterrissage (les trains d'atterrissage ne sont par exemple utilisés que dans la première et la dernière phase de la mission).

Il n'est pas rare que les composants d'un système multi-phases soient multi-états. En effet, un changement de phase de mission se traduit souvent par un changement de mode d'opération de certains composants.

1.2.2.5 Système à fiabilité statique

Le caractère statique ou dynamique de la fiabilité d'un système est déterminé par le fait de considérer les données élémentaires de fiabilité des composants respectivement comme des constantes ou comme des variables du temps (cf. [Marseguerra et al., 1998]). Pour un système à fiabilité dynamique, on cherchera notamment à modéliser le

phénomène de vieillissement continu des composants (taux de défaillance qui croit avec le temps), ainsi que les interactions entre l'évolution continue de variables du procédé et l'évolution discrète du comportement dysfonctionnel des composants (modélisation hybride).

Cette thèse se limite à l'étude des systèmes à fiabilité statique.

1.2.3 Système reconfigurable

Afin de prendre du recul par rapport aux notions de système multi-phases et de composants multi-états, nous introduisons à présent une notion plus générale, mais peu usitée dans le domaine de l'analyse de SdF : la *reconfiguration*. Cette notion est souvent associée à des thématiques de contrôle des Systèmes à Événements Discrets (SED). Ainsi, dans [Faraut et al., 2010], les auteurs décrivent une technique de synthèse de lois de contrôle pour un système reconfigurable. Par ailleurs, dans [Paoli et al., 2011], une méthode pour concevoir un contrôle tolérant aux fautes d'un système reconfigurable est proposée. Une grande partie des travaux sur le sujet concernent les systèmes manufacturiers ([Alcaraz-Mejia et al., 2006], [Nke and Lunze, 2011], [Schmidt, 2015]...), mais la littérature offre également certaines applications pour d'autres secteurs d'activité ([Gonzalez Berlanga et al., 2008] pour les systèmes aéronautiques, [Blodget et al., 2003] pour les systèmes embarqués...).

Dans le cadre de cette thèse, nous appelons système reconfigurable tout système qui dispose de mécanismes de commutation permettant de modifier dynamiquement sa structure et/ou le comportement de ses composants (changement de mode d'opération par exemple). Cette flexibilité du système peut alors être exploitée pour l'élaboration de stratégies de reconfiguration qui seront motivées par des objectifs très variés. En effet, une reconfiguration du système peut être souhaitée, entre autre, pour des raisons :

- fonctionnelles (suite à un changement de phase par exemple) ;
- de production (pour s'adapter aux aléas du carnet de commande) ;
- de maintenance (afin de réaliser des tests périodiques par exemple) ;
- de sécurité (pour conduire le système dans un état sauf) ;
- de tolérance aux fautes (la redondance passive est un exemple de reconfiguration).

Les mécanismes de commutation ont un impact certain sur la sûreté de fonctionnement du système, sous plusieurs aspects. Par exemple :

- les redondances passives sont évidemment mises en place pour améliorer la sûreté de fonctionnement ;
- le changement de mode d'opération d'un composant implique souvent une modification de son comportement dysfonctionnel (par exemple, les taux de défaillance d'un composant peuvent augmenter si on le "surcharge" pour s'adapter aux aléas du carnet de commande) ;
- un changement de phase peut modifier les critères de succès ;
- les commutations augmentent les risques de défaillance à la sollicitation.

De plus, le succès d'une stratégie de reconfiguration n'est pas garanti. En effet, la réalisation d'un mécanisme de commutation peut échouer à cause de la défaillance de l'entité qui le pilote (unité de contrôle, interrupteur, opérateur...), auquel cas l'objectif escompté n'est pas atteint. Il est donc légitime de considérer qu'un composant (ou un sous-système) est non opérationnel non seulement lorsqu'il est défaillant, mais également lorsque le mécanisme de commutation qui était sensé le commuter sur le bon mode a échoué.

Afin de qualifier et de quantifier les impacts d'une stratégie de reconfiguration d'un système (et de son possible échec), il est nécessaire de prendre en compte les mécanismes de commutation et leurs effets dans les modèles de SdF, et ce de manière suffisamment réaliste et précise. Lorsque l'objet d'étude est un système bouclé, ces reconfigurations sont gérées de manière automatique par la partie contrôle-commande et peuvent donc être modélisées formellement comme des lois de commande. Aussi, l'analyse basée sur les modèles d'un système bouclé, en vue de valider une architecture de commande, rend indispensable la prise en compte des mécanismes de commutation dans les modèles de SdF.

1.3 Modélisation pour la sûreté de fonctionnement

Dans cette section, nous présentons brièvement les principaux formalismes de modélisation de la SdF. Ceux ci peuvent traditionnellement être classés dans deux catégories : les formalismes orientés *structure* et ceux orientés *comportement* ([Andrews and Moss,

1993]). Cependant, les approches récentes d'analyse basée sur les modèles poussent à l'utilisation de cadres de modélisation intégrant plusieurs formalismes.

1.3.1 Modélisation structurelle

Ce type de formalisme retranscrit avant tout la structure du système. De ce fait, il est parfaitement indiqué pour la modélisation des systèmes statiques, mais peut parfois être utilisé pour les systèmes dynamiques, après introduction de nouvelles primitives. Dans ce cas, le comportement dysfonctionnel du système est déterminé à travers une interprétation générique du comportement de ces primitives.

1.3.1.1 Diagrammes de fiabilité

Les *diagrammes de fiabilité* sont les modèles de SdF les plus basiques et intuitifs. Il s'agit de représenter le système comme un assemblage de blocs se comportant comme des interrupteurs montés en série ou en parallèle pour relier une entrée à une sortie. Un événement de défaillance correspond alors à l'ouverture d'un interrupteur, et le service est considéré perdu s'il n'existe aucun chemin entre l'entrée et la sortie. Cette idée de modélisation a initialement émergé pour améliorer la fiabilité des circuits composés de relais électromécaniques (cf. [Moore and Shannon, 1956]). Ces modèles sont performants pour représenter la SdF des systèmes statiques. Ils permettent en particulier de visualiser très rapidement les *coupes* du système, c'est à dire les combinaisons de défaillance de composants suffisantes pour provoquer la défaillance du système. En revanche, ils sont inadaptés pour prendre en compte les aspects dynamiques, dans leur version originale.

La notion de diagramme de fiabilité dynamique a été proposée dans [Distefano and Liudong, 2006]. Le principe est d'associer aux blocs une machine états-transitions, pour introduire des comportements plus élaborés que celui d'un interrupteur. De plus, des relations de dépendance sont ajoutées entre deux événements de deux blocs. Ces relations permettent d'ordonner ou de corréler l'occurrence de ces événements. Ainsi, certains aspects dynamiques comme la redondance passive, ou la défaillance à la commutation peuvent être modélisés. Malheureusement les utilisations de ce formalisme dans la littérature sont rares, ce qui rend difficile l'appréciation de ses réelles capacités. Toutefois, on peut penser que le caractère binaire des relations de dépendance introduites par le formalisme risque d'être limitant pour considérer des stratégies de reconfiguration

complexes, impliquant plus de deux unités.

1.3.1.2 Arbres de défaillance

Les *arbres de défaillance* et leurs extensions sont les modèles de SdF les plus utilisés dans la littérature. [Ruijters and Stoelinga, 2014] fournit une large revue de l'état de l'art sur les arbres de défaillance. Le principe est de relier l'occurrence d'un événement redouté à celui d'un ensemble d'événements de défaillance élémentaires par l'intermédiaire de portes logiques. Pour les arbres de défaillance classiques ([Vesely et al., 1981]), la logique est purement combinatoire (portes Booléennes ET, OU et K/N), ce qui ne permet pas de représenter d'éventuelles dépendances entre les événements élémentaires. C'est pourquoi de nombreuses extensions ont été proposées dans la littérature, se matérialisant par l'ajout de portes logiques séquentielles afin d'intégrer des aspects dynamiques. On citera par exemple les Arbres de Défaillance Dynamiques (ADD) [Dugan et al., 1992] (un exemple est représenté sur la Figure 1.2), les Arbres de Défaillance Paramétrés (ADP) [Bobbio and Raiteri, 2004], et les Arbres de Défaillance Qualitatifs (ADQ) [Walker and Papadopoulos, 2009].

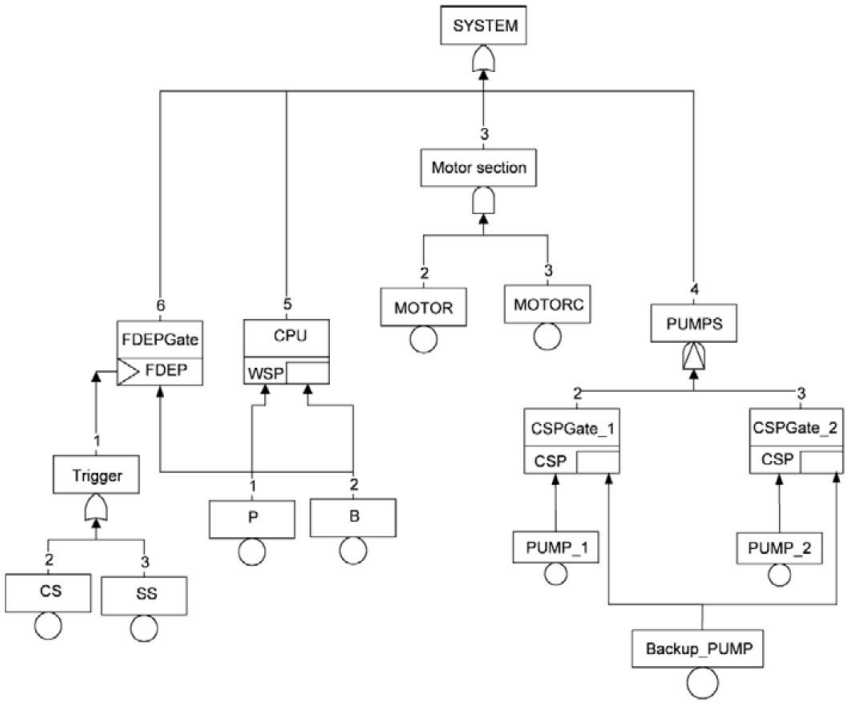

Figure 1.2 – Exemple de d'ADD modélisant la SdF d'un système d'assistance cardiaque ([Boudali and Dugan, 2005])

Les trois portes les plus couramment utilisées dans les ADD sont les portes PAND

(Priority AND), SPARE (littéralement : "de rechange") et FDEP (Functional DEPendency). La porte PAND a été introduite dans [Fussell, 1976] pour ajouter une contrainte d'ordre d'occurrence aux entrées d'une porte ET. [Dugan et al., 1990] définit la porte SPARE (qui peut se décliner en 3 variantes : *hot*, *warm* et *cold*), pour modéliser une redondance passive dont les composants redondants peuvent défaillir à l'arrêt (*hot/warm*) ou non (*cold*). Enfin, la corrélation entre plusieurs événements peut être prise en compte grâce à la porte FDEP (également introduite dans [Dugan et al., 1990]).

La grande force des formalismes basés sur les arbres de défaillance est leur simplicité. La méthodologie de construction des modèles est déductive et s'apparente à une démarche de décomposition fonctionnelle d'un système. Aussi, leur usage est très répandu en contexte industriel. L'inconvénient majeur de ces formalismes est qu'ils restent imprécis en terme de modélisation des aspects dynamiques. En effet, quel que soit le nombre de portes dynamiques ajouté dans les primitives du formalisme, les comportements modélisables restent limités par les comportements génériques décrits par ces portes.

1.3.2 Modélisation comportementale

La deuxième catégorie de formalismes utilisés en SdF est dédiée à la modélisation du comportement dysfonctionnel d'un système. Ces modèles représentent les scénarios de défaillance de manière plus précise et spécifique que les modèles structurels. En outre, et contrairement à ceux-ci, les formalismes comportementaux ne sont pas initialement conçus pour supporter les analyses de SdF mais ont démontré leur pertinence et leur efficacité dans de nombreuses applications.

1.3.2.1 Chaînes de Markov

L'usage de *chaînes de Markov* [Markov, 1971] pour modéliser le comportement dysfonctionnel des systèmes est très répandu dans la littérature et est préconisé par plusieurs standards industriels (par exemple dans le domaine aéronautique [SAE, 1996]). Pour ce type de modélisation, un comportement dysfonctionnel est vu comme un processus stochastique vérifiant la propriété de Markov [2]. Cette hypothèse est valide pour de nombreux cas de défaillances matérielles non systématiques, et se traduit, entre autre, par la non

2. Un processus stochastique vérifie la propriété de Markov si et seulement si son évolution future ne dépend que de son état présent, et non de ses états passés. On parle d'un processus sans mémoire.

prise en compte des phénomènes de vieillissement. L'hypothèse semble plus difficile à justifier pour les réparations, mais elle est néanmoins admise par soucis d'homogénéité des modèles, et de capacité à conduire des calculs numériques.

Malgré cette hypothèse, la chaîne de Markov reste un outil permettant une modélisation comportementale d'une précision incomparable par rapport aux formalismes de la première catégorie. De plus, elle est définie dans un cadre mathématique propice au développement de techniques d'analyses efficaces (cf. [Stewart, 1994]). Malheureusement, la représentation de chaque état étant explicite, ce type de modélisation est sujet au problème classique d'explosion combinatoire. La construction manuelle de tels modèles pour des systèmes de taille importante est impossible. Par conséquent, les chaînes de Markov sont soit construites manuellement pour décrire finement des comportements locaux très spécifiques, soit générées automatiquement à partir d'un autre modèle, pour faciliter l'analyse quantitative. Une chaîne de Markov modélisant le comportement dysfonctionnel de deux composants A et B, ne pouvant être réparés que dans l'ordre de leurs défaillances, est représentée sur la Figure 1.3.

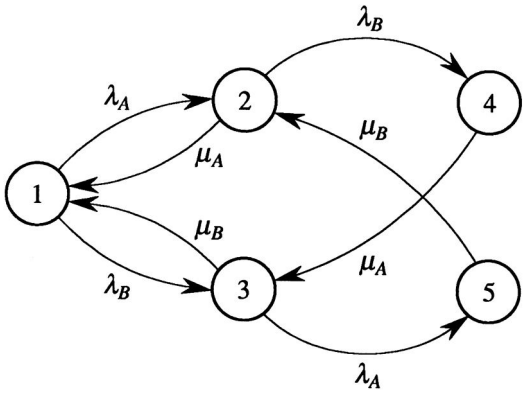

Figure 1.3 – Exemple de chaîne de Markov pour deux composants

1.3.2.2 Réseaux de Petri

Pour limiter le problème d'explosion combinatoire rencontré avec les chaînes de Markov, de nombreux travaux exploitent les capacités d'abstraction des *Réseaux de Petri* (RdP) [Murata, 1989]. Ces graphes permettent de représenter des mécanismes de transition sans avoir à décrire explicitement les états du modèle. Un *Réseau de Petri Stochastique* (RdPS) est un RdP pour lequel l'occurrence des transitions est déterminé par

des lois stochastiques. [Molloy, 1982] montre qu'un RdPS *borné* est équivalent à une chaîne de Markov (qui peut alors être générée en construisant le graphe des marquages accessibles du réseau).

Finalement, [Ajmone Marsan et al., 1994] définit les *Réseaux de Petri Stochastiques Généralisés* (RdPSG) pour modéliser conjointement les comportements attendus et dysfonctionnels d'un système (cf. Figure 1.4). Cette généralisation permet de prendre en compte des mécanismes de commutation, et donc de modéliser la SdF des systèmes reconfigurables (tels que décrit dans la sous-section 1.2.3). Toutefois, les modèles sont fastidieux à construire, et *in fine* pratiquement illisibles, dès lors que les interactions entre les composants deviennent complexes.

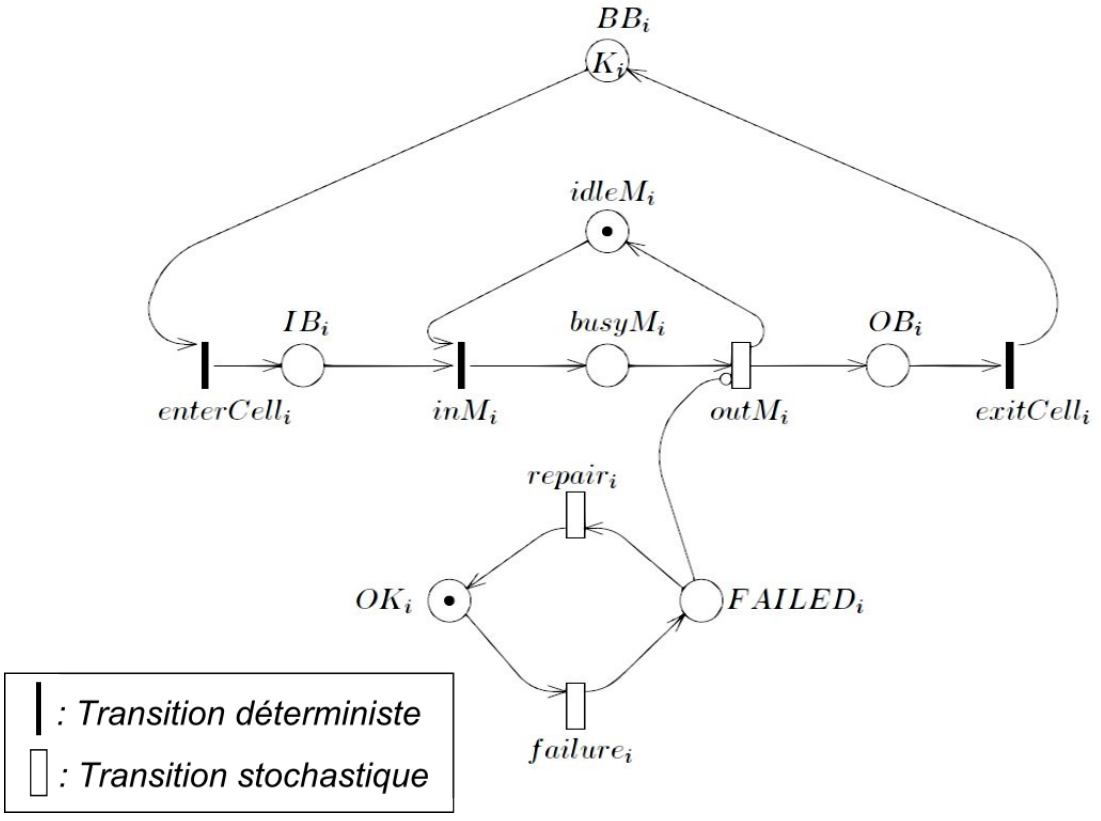

Figure 1.4 – RdPSG modélisant une cellule flexible faillible [Ajmone Marsan et al., 1994]

1.3.3 Modélisation multi-vues

Les deux catégories de modèles exposées ci-dessus sont complémentaires. Aussi, les approches modernes tendent à exploiter les avantages des deux catégories, soit de manière séquentielle : par exemple, transformation systématique d'un ADD en chaîne de

Markov [Dugan et al., 1993] ou en RdPS [Codetta-Raiteri, 2005] (cf. Figure 1.5), puis éventuellement, enrichissement du modèle par expertise ; soit de manière intégrée dans un cadre de modélisation haut niveau. Cette dernière approche de modélisation multi-vues est conforme à la démarche d'analyse basée sur les modèles (cf. sous-section 1.2.1.3). Dans la suite, une sélection de quelques travaux existants sur le sujet est présentée.

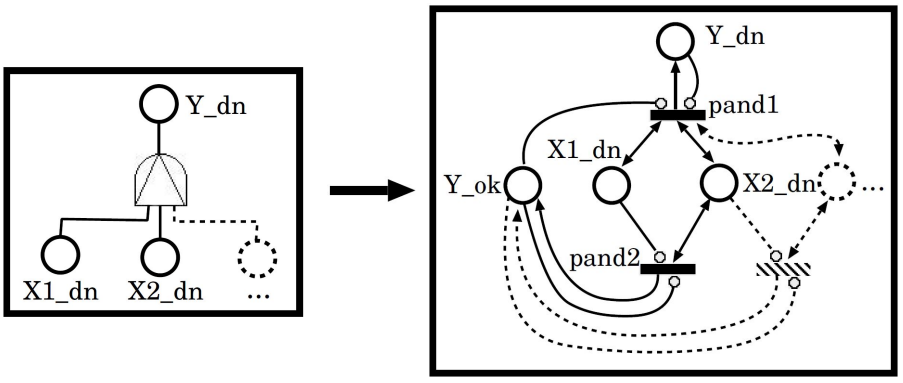

Figure 1.5 – Traduction d'une porte PAND en un RdPS [Codetta-Raiteri, 2005]

1.3.3.1 AltaRica

AltaRica est un langage de modélisation de haut-niveau dédié à l'analyse de SdF basée sur les modèles [Batteux et al., 2013]. La dernière version de ce langage est issue de la combinaison des formalismes S2ML (System Structure Modeling Language) et GTS (Guarded Transition System) [Rauzy, 2008]. Ces deux formalismes permettent de traduire respectivement la structure et le comportement (au sens large) d'un système. Plusieurs outils d'analyse ont été développés pour ces modèles (générateur de séquences critiques, simulation stochastique...). La polyvalence de ce langage lui a permis d'acquérir une popularité remarquable dans les milieux industriels (citons les implémentations industrielles Cecilia OCAS pour Dassault Aviation, Simfia pour EADS Apsys et Safety Designer pour Dassault Systèmes). Les performances du langage ont été éprouvées sur de nombreux cas d'application industriels ([Boiteau et al., 2006], [Adeline et al., 2010], ...).

Toutefois, les qualités de ce langage en terme de pouvoir d'expression ont une contrepartie : un modèle AltaRica n'a pas de représentation graphique intrinsèque. Aussi, afin de faciliter l'exploitation des modèles, les contributeurs du langage AltaRica ont développé l'outil GraphXica, qui permet de spécifier la forme et l'animation d'une repré-

sentation graphique associée à un modèle AltaRica construit sous certaines contraintes (cf. [Prosvirnova et al., 2013]). Cet outil est idéal pour aider à l'interprétation et à la communication des résultats d'analyses, mais n'est pas conçu pour aider l'expert à construire le modèle, étape qui doit être réalisée en lignes de code (cf. Figure 1.6). Dans le cadre de cette thèse, nous souhaitons proposer un formalisme de modélisation graphiquement représentable par nature, pour faciliter sa construction par l'expert de SdF. Nous chercherons néanmoins à interfacer nos résultats avec AltaRica, afin de bénéficier des outils développés pour celui-ci, et de toucher une plus large communauté scientifique et industrielle.

```
domain PumpState {WORKING, FAILED}
class Pump
  PumpState state (init = WORKING);
  Real input (reset = 0.0), output (reset = 0.0);
  event failure (delay = exponential(0.0005));
  event repair (delay = exponential(0.02));
  transition
    failure: state==WORKING -> state := FAILED;
    repair: state==FAILED -> state := WORKING;
  assertion
    output := if (state==WORKING) then input
              else 0.0;
end
```

```
class Tank
  Real input (reset = 0.0);
end
block WaterSupplySystem
  Pump pump;
  Tank tank;
  Real input (reset = 1.0);
  observer Boolean tank.isEmpty =
                          (tank.input == 0.0);
  assertion
    pump.input := input;
    tank.input := pump.output;
end
```

Figure 1.6 – Exemple de modèle AltaRica pour un système simple d'alimentation d'eau (extrait de [Prosvirnova et al., 2013])

1.3.3.2 HiP-HOPS

A première vue (cf. Figure 1.7), un modèle HiP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies) peut faire penser à un diagramme de fiabilité hiérarchique (ensemble de blocs inter-connectés et encapsulés) [Papadopoulos and McDermid, 1999]. La différence fondamentale est que les liens entre les blocs sont porteurs de flux (débit de liquide, courant électrique, paquet d'information...), ce qui permet notamment de propager précisément des effets spécifiques de certains modes de défaillance, sur des variables caractéristiques du système. Ainsi, cette approche est bien adaptée pour réaliser des analyses de performance du système dépassant le cadre de la SdF mais néanmoins basées sur une analyse préliminaire des risques. En outre, ce type de modélisation est propice à un interfaçage avec des logiciels tels que Matlab/SIMULINK [Papadopoulos and Maruhn, 2001]. Une utilisation très intéressante de cette approche pour optimiser la conception d'architectures tolérantes aux fautes, a été proposée dans [Papadopoulos et al., 2011].

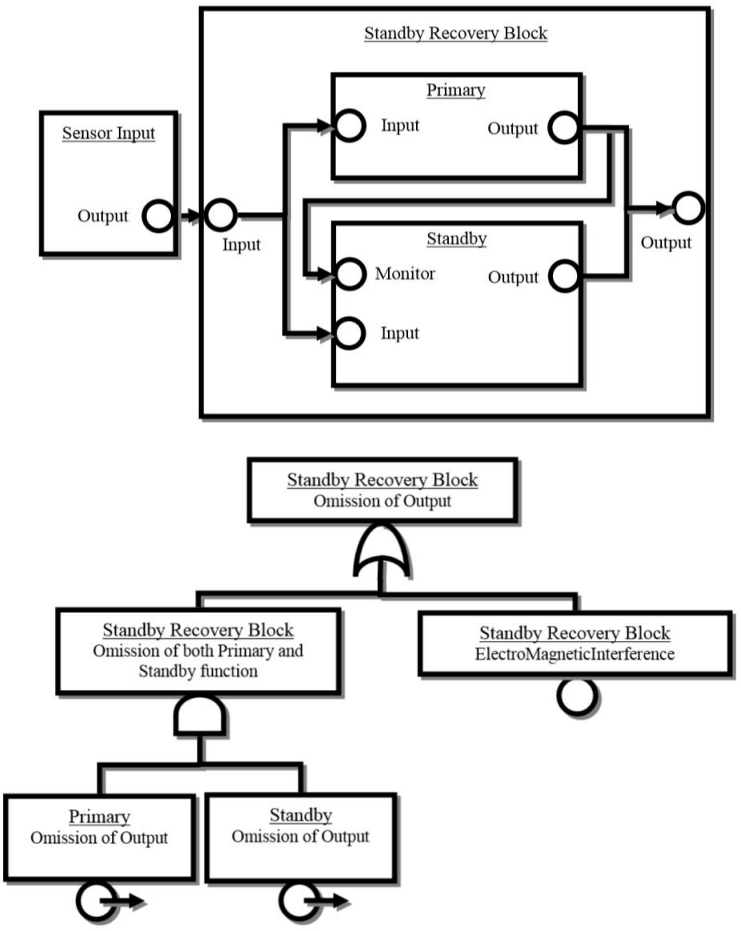

Figure 1.7 – Vues *bloc* (à gauche) et *arbre de défaillance* (à droite) d'un modèle HiP-HOPS d'une redondance passive simple

Cependant, l'approche HiP-HOPS ne permet pas la prise en compte des systèmes réparables.

1.3.3.3 BDMP

Comme son nom l'indique, le formalisme BDMP (Boolean logic Driven Markov Processes) combine les avantages des théories Booléennes et Markoviennes [Bouissou and Bon, 2003]. Il s'agit d'un arbre de défaillance simple, dont les feuilles sont associées à des processus de Markov, et non à de simples variables Booléennes. Ces processus permettent de traduire un comportement dynamique localisé précis, et notamment de considérer des réparations, des défaillances à la sollicitation... De plus, le formalisme introduit la primitive de *gâchette* dans les arbres de défaillance, offrant de ce fait la possibilité de décrire des mécanismes de commutation basiques.

Le formalisme BDMP a été proposé dans [Bouissou and Bon, 2003] afin de répondre

a un besoin de modélisation pour les systèmes de production d'énergie et a également été expérimenté pour des applications de sécurité informatique ([Pietre-Cambacedes and Bouissou, 2010],[Pietre-Cambacedes et al., 2011],[Kriaa et al., 2012]). Dans [Chaux et al., 2011], le comportement décrit par un modèle BDMP est interprété à base d'automate à états fini. Le formalisme BDMP a été implémenté dans l'outil KB3 [Renault et al., 1999], à l'aide du langage de spécification haut niveau FIGARO [Bouissou et al., 1991]. Ainsi, il hérite des outils développés pour FIGARO : FIGSEQ (outil d'exploration de séquence), YAMS (outil de simulation de Monte Carlo)...

Une description plus détaillée de ce formalisme sera fournie dans le second chapitre.

1.3.3.4 Autres approches

Nous citons également, mais sans plus nous y attarder, les approches :

- SAML (Safety Analysis Modeling Language) [Gudemann and Ortmeier, 2010] : un langage basé sur des automates à états stochastiques et une loi de composition parallèle.
- AADL Error Model (Architecture Analysis & Design Language) [Feiler and Rugina, 2007] : une extension pour la SdF du langage de description d'architecture de systèmes embarqués AADL.
- DFM (Dynamic Flowgraph Methodology) [Garrett et al., 1995] : une approche originale permettant de représenter sous forme de graphe les interactions fonctionnelles et dysfonctionnelles entre les variables d'évolution d'un système.
- Sophia [Cancila et al., 2009] : un profil SysML [OMG, 2005] pour les analyses de SdF.
- MeDISIS (Méthode D'Intégration des analyses de Sûreté de fonctionnement au processus d'ingénierie Système) [David et al., 2010] : un autre méta-modèle SysML pour les analyses de SdF.

1.4 Méthodes d'analyse

Les différentes techniques d'analyses de SdF fournissent des résultats qualitatifs ou quantitatifs. Les premiers permettent d'identifier les "faiblesses" du système en vue

d'orienter les choix de modification à appliquer lors d'un éventuel retour en conception, tandis que les seconds permettent d'estimer les probabilités de succès (ou d'échec) du système. Dans cette section, nous présentons une sélection de techniques d'analyse de SdF existantes.

1.4.1 Analyse qualitative

Les analyses qualitatives (pour un système dynamique) consistent à extraire du modèle les Séquences de Coupes Minimales (SCM), c'est à dire, les séquences d'événements les plus courtes conduisant à l'échec du système [Tang and Dugan, 2004].

1.4.1.1 Approches algébriques

Dans [Merle et al., 2011], les auteurs étendent l'algèbre de Boole en introduisant l'opérateur temporel $\lhd$, pour traduire une contrainte de séquentialité entre deux événements, afin de permettre une formulation algébrique de la fonction de structure d'un ADD. Cette formulation peut ensuite être raffinée sous une forme canonique minimale permettant de déduire l'ensemble des SCM. Par exemple, l'expression canonique de la fonction de structure de l'ADD représenté sur la Figure 1.2 est reportée ci-dessous [3] :

$$\begin{aligned} TE \;=\; & CS \;+\; SS \;+\; MOTOR \cdot MOTORC \;+\; P \cdot (B_d \lhd P) \\ & +\; B_a \cdot (P \lhd B_a) \;+\; BP_a \cdot (P2 \lhd P1) \cdot (P1 \lhd BP_a) \\ & +\; P2 \cdot (P1 \lhd BP_a) \cdot (BP_a \lhd P2) \end{aligned}$$

On déduit de cette expression l'ensemble des séquences de coupes minimales :

$$\begin{aligned} SCM = \{ & CS,\ SS,\ MOTOR \to MOTORC,\ MOTORC \to MOTOR, \\ & B_d \to P,\ P \to B_a,\ P2 \to P1 \to BP_a,\ P1 \to BP_a \to P2\} \end{aligned}$$

Cette approche est mathématiquement très élégante, mais malheureusement pas très opérationnelle. En effet, la taille de ces expressions algébriques augmente significativement avec celle du système, et le calcul symbolique permettant d'extraire les SCM devient alors difficile.

C'est pourquoi [Rauzy, 2011] propose une formulation algébrique alternative à celle-

3. Ici, $+$ et $\cdot$ correspondent respectivement aux opérateurs 'OU' et 'ET' de l'algèbre de Boole. De plus, $A \lhd B$ est vrai si et seulement si A est déjà vrai lorsque B devient vrai.

ci, en définissant une *algèbre des séquences*, et en lui associant une nouvelle structure de données : Sequence Decision Diagram (SDD). Cette structure de données est dérivée des Binary Decision Diagrams (BDD), définis dans [Bryant, 1986], pour encoder sous une forme synthétique et opérationnelle une fonction Booléenne. Ainsi, l'ensemble des SCM peut se représenter sous une forme graphique compacte, et les opérations entre les ensembles de séquences peuvent être implémentées de façon efficace.

Ces deux techniques sont valables uniquement pour les séquences dans lesquelles chaque événement n'apparait qu'une seule fois. Aussi, elles ne conviennent pas pour prendre en compte les systèmes réparables, pour lesquels des événements de défaillance et de réparation duaux peuvent survenir plusieurs fois alternativement.

1.4.1.2 Approche par la théorie des langages

Une autre voie est explorée dans [Chaux et al., 2013]. Cette nouvelle approche est basée sur une définition des critères de cohérence pour les systèmes dynamiques réparables binaires. Les critères sont formalisés dans le cadre de la théorie des langages, et sont utilisés pour calculer l'ensemble des SCM. Cet ensemble est alors défini comme le plus petit ensemble de séquences nécessaire et suffisant pour reconstruire tous les scénarios de défaillance du système. Ainsi, tout scénario de défaillance peut être construit à partir d'une séquence de coupe minimale, dans laquelle on injecte des événements de panne ou de réparation en respectant les règles de cohérence.

Cette technique sera détaillée et étendue dans le chapitre 3, en vue d'une application pour les systèmes dynamiques réparables reconfigurables.

1.4.2 Analyse quantitative

Les analyses quantitatives consistent le plus souvent à évaluer la *fiabilité* et/ou la *disponibilité* du système (ou parfois certaines valeurs temporelles ou fréquentielles, fortement liées à celles-ci). La *fiabilité* du système est la probabilité qu'il maintienne son service, sans interruption, sur une période donnée, tandis que sa *disponibilité* correspond au pourcentage de la période durant laquelle le service est assuré.

1.4.2.1 Calcul exact sur chaîne de Markov

Si on est capable de générer une chaîne de Markov représentant le comportement dysfonctionnel du système, et que cette chaîne est de taille "raisonnable", on peut envisager le calcul exact des attributs de SdF en résolvant directement la chaîne de Markov. En effet, la disponibilité à un instant donné, correspond à la probabilité que la chaîne de Markov soit dans un de ses états "non défaillant". La fiabilité se calcule de la même façon, en supprimant toutes les transitions partant d'un état "défaillant". La littérature sur les chaînes de Markov fournit plusieurs techniques pour résoudre une chaîne de Markov, ce qui revient à résoudre un système d'équations différentielles (par exponentiation de la matrice de transition, par uniformisation, par la méthode de Runge-Kutta, par la méthode d'Adams-Bashforth... ; ces méthodes sont décrites dans l'ouvrage [Stewart, 1994]).

Malheureusement, comme on l'a évoqué dans la sous-section 1.3.2.1, le calcul exact de ces attributs n'est possible que pour les petits systèmes. L'explosion combinatoire de l'espace d'état de la chaîne de Markov modélisant un système de taille importante, rend ces calculs impossibles avec les technologies informatiques actuelles.

1.4.2.2 Estimation par simulation de Monte Carlo

Une technique très répandue pour évaluer les attributs de SdF est la simulation de Monte Carlo [Durga Rao et al., 2009]. Il s'agit de simuler aléatoirement un grand nombre de scénarios sur le modèle, et de calculer, à plusieurs instants, le pourcentage de scénarios dans lesquels le système est défaillant. La courbe qui relie ces points est un estimateur du profil de la disponibilité du système (cf. Figure 1.8). Pour celui de la fiabilité, le principe est le même, à ceci près que l'on ne considère pas les évolutions du système postérieures à sa première défaillance.

Le principal avantage de cette technique est qu'elle est applicable quelles que soient la nature et la complexité du modèle considéré. Il suffit de connaître les lois d'évolution du modèle. Elle est donc idéale pour être appliquée sur un modèle dont les états sont décrits par intention, et non de manière explicite. Cependant, elle donne des résultats approchés et, pour que l'erreur soit négligeable par rapport aux incertitudes sur les données d'entrées, il est souvent nécessaire de jouer un très grand nombre de scénarios, ce qui peut prendre un temps considérable.

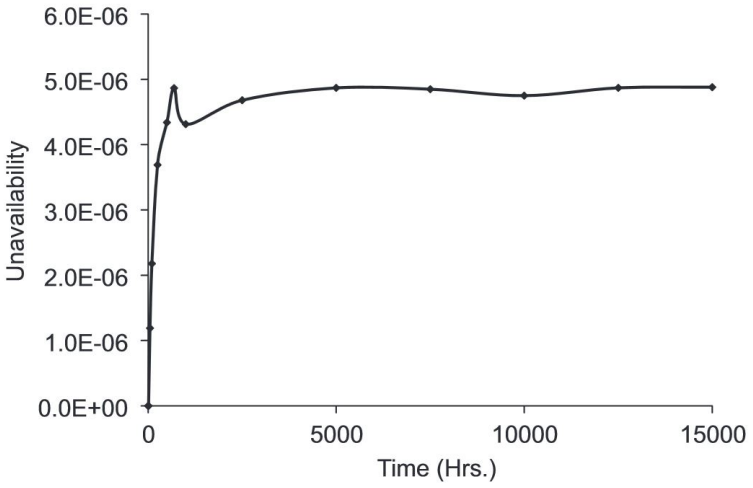

Figure 1.8 – Indisponibilité d'un système en fonction du temps, évaluée par simulation de Monte Carlo (extrait de [Durga Rao et al., 2009])

1.4.2.3 Approximations astucieuses

De nombreux travaux cherchent à approximer les attributs de SdF en les encadrant par des bornes, moyennant certaines hypothèses simplificatrices.

Par exemple, "l'école d'EDF" postule une alternance stricte entre défaillance du système et réparation totale (tous les composants sont réparés). Partant de cette hypothèse, les contributeurs proposent une méthode pour calculer une borne inférieure de la disponibilité du système ([Bon and Collet, 1994],[Bouissou and Lefebvre, 2002]).

D'autres auteurs cherchent à évaluer la probabilité d'occurrence des SCM (résultat de l'analyse qualitative), la somme de ces probabilités étant également une borne inférieure de la disponibilité du système ([Liu et al., 2007], [Ionescu et al., 2014]...).

Enfin, certains travaux consistent à diminuer la taille de la chaîne de Markov à résoudre. Le plus souvent, il s'agit de construire la chaîne de Markov complète, puis de la réduire soit par agrégation d'états (comme dans [Muntz et al., 1989]), soit par élimination de branches associées à des probabilités trop faibles pour influer sensiblement sur le résultat (comme dans [Fourneau et al., 2007]). Cependant, la construction de la chaîne de Markov complète est elle-même une opération coûteuse. C'est pourquoi [Brameret et al., 2015] propose une technique de construction limitée de chaîne de Markov, à partir d'une description haut-niveau du système, inspirée de l'algorithme de Dijkstra ([Dijkstra, 1959]). Cette technique a été implémentée pour fonctionner avec un modèle AltaRica en entrée. Elle sera décrite dans le chapitre 3 et appliquée pour l'analyse quantitative d'un système dynamique réparable reconfigurable.

1.5 Bilan et contributions

On rappelle que l'objectif de cette thèse est de proposer une méthode d'analyse de la SdF basée sur les modèles pour les systèmes dynamiques réparables et reconfigurables. La pertinence des résultats d'une telle analyse dépend en grande partie de la capacité du formalisme de modélisation considéré à représenter avec précision l'impact des différentes stratégies de reconfiguration (et de leurs échecs) sur la SdF du système. Ce chapitre a livré un aperçu de l'état de l'art sur l'analyse de SdF basée sur les modèles des systèmes critiques. Les commentaires apportés soulignent que les formalismes de modélisation existants sont :

- soit trop restrictifs pour prendre en compte de manière suffisamment précise les aspects liés au caractère reconfigurable des systèmes considérés ;
- soit inadaptés aux méthodes de travail des experts de SdF, responsables de la construction des modèles.

Toutefois, l'un d'entre eux, le formalisme BDMP, constitue une base favorable à nos travaux. En effet, sa capacité à modéliser pertinemment les systèmes dynamiques réparables n'est plus à démontrer. De plus, la primitive de gâchette permet d'introduire, bien qu'imparfaitement, la notion de reconfiguration, comme il sera montré au chapitre suivant. Par ailleurs, il est très adapté aux habitudes méthodologiques des experts de SdF - qui sont responsables de la construction de ces modèles -, car il se représente graphiquement comme un arbre de défaillance et repose sur des chaînes de Markov de petite taille.

Dans le chapitre suivant, nous développerons les avantages des BDMP et discuterons de leur possible exploitation pour la modélisation et l'analyse de la SdF des systèmes reconfigurables. Puis, nous proposerons une généralisation de ce formalisme pour les systèmes dynamiques réparables et reconfigurables : les BDMP Généralisés (GBDMP). Enfin, nous mènerons une réflexion méthodologique sur la construction des modèles.

Le troisième chapitre sera consacré à l'adaptation de deux techniques d'analyse (une qualitative et une quantitative) au formalisme de GBDMP. Dans un premier temps, les travaux décrits dans [Chaux et al., 2013] seront étendus, et un nouvel algorithme sera proposé pour générer efficacement les SCM d'un système à partir de son modèle GBDMP (quelle que soit la taille du modèle). Puis, une traduction d'un modèle GBDMP

en un modèle AltaRica sera proposée, afin de pouvoir utiliser les outils d'analyse développés pour ce langage. En particulier, nous expérimenterons la technique de génération limitée de chaîne de Markov, décrite dans [Brameret et al., 2015]. L'application de ces techniques sur des exemples simples, nous conduira à observer l'impact des stratégies de reconfiguration du système (et de leurs possibles échecs) sur sa SdF.

Enfin, cette approche sera appliquée dans le dernier chapitre, sur un cas d'étude représentatif des problématiques industrielles qui motivent cette thèse. Le cas est extrait d'une centrale nucléaire numérique simplifiée, conçue pour illustrer les travaux de recherche entrepris dans le cadre du projet CONNEXION. Elle fait intervenir plusieurs systèmes élémentaires en interaction, contrôlés par une architecture opérationnelle de contrôle-commande qu'il convient de valider sous l'angle de la SdF. Nous étudierons ainsi le comportement dysfonctionnel de deux systèmes élémentaires contrôlés pour réaliser plusieurs fonctions de refroidissement. Cette étude sera réalisée à l'aide d'un prototype d'outil logiciel pour la modélisation et l'analyse de SdF, développé dans le cadre de cette thèse, dont on décrira les fonctionnalités.

Chapitre
2
Modélisation par BDMP Généralisé

Sommaire

2.1 Introduction 33

2.2 Capacités et limites de la modélisation BDMP 33

2.2.1 Rappel de la définition des BDMP 34

2.2.2 Exemples de modélisation BDMP 37

2.2.2.1 Modélisation d'une redondance passive 37

2.2.2.2 Report de charge 38

2.2.2.3 Systèmes multi-phases 40

2.2.3 Modélisation des mécanismes de commutation en BDMP . . . 41

2.2.3.1 Discussion sur la modélisation des redondances passives . . . 41

2.2.3.2 Modélisation des composants multi-états 43

2.2.3.3 Modélisation des systèmes multi-phases avec modification des critères de succès des composants 44

2.2.4 Bilan sur la modélisation BDMP 47

2.3 Définition des BDMP Généralisés 47

2.3.1 Définition de la syntaxe 48

2.3.1.1 Processus de Markov Commutés 50

2.3.1.2 Machines de Moore 51

2.3.1.3 Contraintes de construction d'un modèle GBDMP 53

2.3.2 Définition de la sémantique 54

2.3.2.1 Interprétation des primitives 55

2.3.2.2 Principe d'évolution d'un modèle 59
2.3.3 Algorithme de simulation à événements discrets d'un modèle GBDMP 61
2.4 Construction de modèles GBDMP 67
2.4.1 Méthodologie générale 67
2.4.1.1 Construction de l'arbre de défaillance 67
2.4.1.2 Description des comportements élémentaires par des PMC . 68
2.4.1.3 Spécification des logiques de commutation par des machines de Moore 71
2.4.2 Exemples de modélisation GBDMP 73
2.4.2.1 Différentes stratégies de redondance appliquées par des composants de commutation faillibles 74
2.4.2.2 Le report de charge 81
2.4.2.3 Les vannes bloquantes 83
2.5 Bilan 86

2.1 Introduction

Afin d'analyser formellement la sûreté de fonctionnement d'un système, il est nécessaire de se doter d'un formalisme de modélisation, pour produire les modèles sur lesquels l'analyse sera basée. Ces modèles sont destinés à représenter de manière non ambigüe les scénarios d'événements pouvant affecter le fonctionnement du système (ou le restaurer si le système est réparable). Le chapitre précédent a permis de souligner la pertinence du formalisme BDMP pour la modélisation de la sûreté de fonctionnement des systèmes réparables dynamiques.

Le premier objectif de ce chapitre est de développer ce constat, en expliquant les principes originaux et prometteurs qui caractérisent ce formalisme. Puis, celui-ci sera poussé aux limites de son pouvoir d'expression, de manière à montrer que certains scénarios d'événements à prendre en compte pour la classe de système considérée dans cette thèse (cf. 1.2.2), ne sont pas pris en compte de manière suffisamment réaliste. En particulier, nous verrons que la modélisation des mécanismes de commutation proposée dans les BDMP est insuffisante pour considérer des stratégies complexes de reconfiguration du système. Plusieurs exemples simples seront introduits pour illustrer l'argumentation.

A partir de ces observations, les principes fondamentaux qui caractérisent le formalisme BDMP seront généralisés pour étendre ses capacités de modélisation à la classe de système considérée dans cette thèse. Le nouveau formalisme ainsi constitué sera dénommé BDMP Généralisé (Generalized BDMP en anglais : GBDMP). La seconde section est vouée à expliquer les idées qui en ont permis l'élaboration, à définir la syntaxe et la sémantique des modèles, et à proposer un algorithme pour simuler leur comportement.

Enfin, pour faciliter la prise en main du formalisme, une réflexion méthodologique sur la construction des modèles est proposée dans la dernière section de ce chapitre. Cette méthode sera appliquée sur trois exemples simples mais représentatifs de la classe de systèmes considérée.

2.2 Capacités et limites de la modélisation BDMP

Le formalisme BDMP combine les avantages des arbres de défaillance et des chaînes de Markov. Il permet ainsi une prise en compte d'aspects dynamiques complexes tout en conservant une représentation graphique intuitive.

2.2.1 Rappel de la définition des BDMP

[Bouissou and Bon, 2003] définit formellement un modèle BDMP par un 4-tuple $< \mathcal{F}, r, T, (P_i) >$ tel que [1] :

- $\mathcal{F}$ est un *arbre de défaillance multi-enraciné*, i.e. un 3-tuple $< N, E, \kappa >$ où :
  - $N = G \cup L$ tel que $G \cap L = \varnothing$ est un ensemble de *nœuds*, partitionné entre les *portes* (G) et les *feuilles* (L) ;
  - $E \subseteq G \times N$ est un ensemble d'arcs orientés, tel que $< N, E >$ est un graphe orienté acyclique ;
  - $\kappa \in G \longrightarrow \mathbb{N}^*$ est une fonction permettant de déterminer le type des portes. Soit g une porte ayant n *fils* : si $\kappa(g) = n$, alors g est une porte ET, si $\kappa(g) = 1$, alors g est une porte OU, et plus généralement, si $\kappa(g) = k$, alors g est une porte "k parmi n" ;
- $r \in G$ est la *racine principale* de $\mathcal{F}$;
- $T \subseteq (N \backslash \{r\}) \times (N \backslash \{r\})$ est un ensemble de *gâchettes*, tel que :
  $\forall (orig, dest) \in T : orig \neq dest$;
  $\forall ((orig_1, dest_1), (orig_2, dest_2)) \in T^2 : orig_1 \neq orig_2 \Rightarrow dest_1 \neq dest_2$;
- (P_i) est une famille de *Processus de Markov Pilotés* (PMP) associés aux feuilles. Un PMP est un 5-tuple $< \mathcal{Z}_0, \mathcal{Z}_1, \mathcal{X}_F, f_{0\to1}, f_{1\to0} >$ où :
  - $\mathcal{Z}_0$ et $\mathcal{Z}_1$ sont deux chaînes de Markov continues homogènes (dans la suite, $\mathcal{X}_0$ et $\mathcal{X}_1$ désignent leur espace d'état respectif) ;
  - $\mathcal{X}_F \subseteq \mathcal{X}_0 \cup \mathcal{X}_1$ est le sous-ensemble des états défaillants ;
  - $f_{0\to1} \in \mathcal{X}_0 \times \mathcal{X}_1 \longrightarrow [0, 1]$ est une *fonction de transfert probabiliste* entre les états de $\mathcal{Z}_0$ et ceux de $\mathcal{Z}_1$;
  - $f_{1\to0} \in \mathcal{X}_1 \times \mathcal{X}_0 \longrightarrow [0, 1]$ est une *fonction de transfert probabiliste* entre les états de $\mathcal{Z}_1$ et ceux de $\mathcal{Z}_0$.

1. On trouve dans la littérature plusieurs extensions du formalisme qui proposent des variantes de ces primitives. Par exemple, dans [Bouissou, 2007], l'auteur introduit des portes PAND et certains processus de Markov sont remplacés par des Réseaux de Petri. On trouve également la notion de gâchette inversée dans [Bouissou, 2005], pour modéliser des modes de défaillance exclusifs. Afin de focaliser la présentation sur ce qui fait l'essence du formalisme BDMP, nous nous limitons ici à sa définition initiale.

La Figure 2.1 montre la représentation graphique d'un exemple didactique de BDMP. La partie gauche représente la vue structurelle ; on y reconnaît un arbre de défaillance, enrichi par des gâchettes :

- $L = \{L1, L2, L3, L4\}$ et $G = \{G1, G2, G3, G4\}$;
- $G1$ et $G2$ sont des portes OU ($\kappa(G1) = \kappa(G2) = 1$), et $G3$ et $G4$ sont des portes ET ($\kappa(G3) = \kappa(G4) = 2$) ;
- la racine principale r est la porte $G4$;
- les gâchettes $(G1, L1)$ et $(G3, G2)$ sont représentées par des flèches en pointillés allant de l'origine vers la destination.

Le comportement des feuilles $L1$, $L2$, $L3$ et $L4$ est décrit par un PMP de type "SF" (pour Standby Failure), représenté sur la partie droite de la figure :

- les deux chaînes de Markov sont désignées par "Active mode" et "Inactive mode" ;
- leurs états défaillants sont F_1 et F_2 ;
- Les flèches en pointillés représentent la fonction de transfert ($f_{0\to1}(S, W) = 1$, $f_{0\to1}(S, F_2) = 0, f_{1\to0}(W, S) = 1...$).

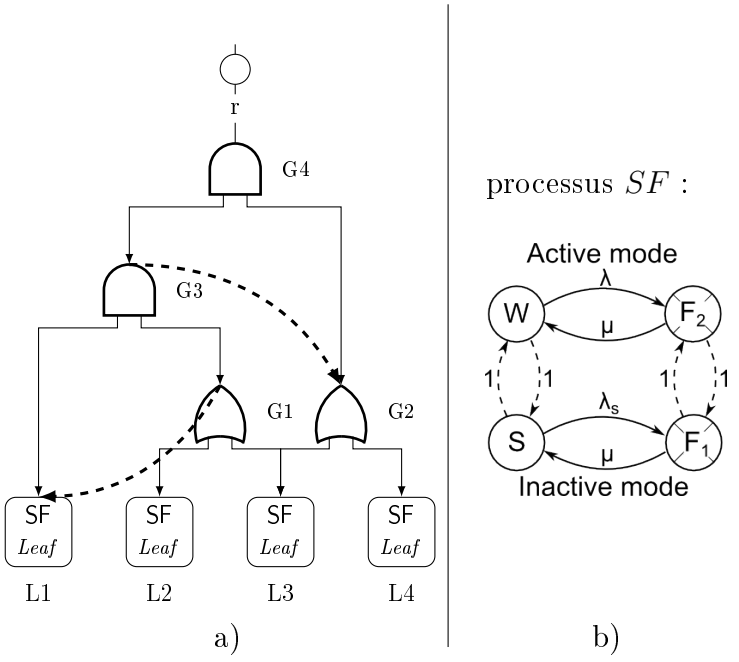

Figure 2.1 – Exemple de modèle BDMP. a) vue structurelle ; b) vue comportementale

Ainsi, un modèle BDMP est un arbre de défaillance dont les feuilles sont associées à des processus de Markov pilotés par des gâchettes. L'état de chaque nœud n est caractérisé par deux statuts [2] : un statut de *défaillance* F_n et un statut d'*activité* M_n. La défaillance d'une porte est déduite du statut de *défaillance* de ses *fils* (i.e. ses nœuds successeurs), selon le type de la porte. Ainsi, pour l'exemple proposé :

$$\begin{aligned} F_{G4} &= F_{G3} \wedge F_{G2} \\ &= (F_{L1} \wedge F_{G1}) \wedge (F_{L3} \vee F_{L4}) \\ &= (F_{L1} \wedge (F_{L2} \vee F_{L3})) \wedge (F_{L3} \vee F_{L4}) \end{aligned}$$

La défaillance des feuilles est déterminée par le PMP qui lui est rattaché. Un PMP est défini par deux chaînes de Markov, deux fonctions de transfert des probabilités (d'une chaîne à l'autre), et d'un partitionnement de l'espace d'état pour distinguer les états défaillants des états non défaillants. Les deux chaînes de Markov permettent de modéliser le comportement dysfonctionnel d'une feuille lorsqu'elle est respectivement active et inactive.

Le statut d'*activité* de certains nœuds est contrôlé par les gâchettes. Une gâchette est un arc orienté entre deux nœuds appelés l'*origine* et la *destination*. Lorsque l'*origine* est défaillante (resp. non défaillante), la *destination* est requise (resp. non requise). Un nœud est activé si et seulement si, d'une part, il est requis (dans le cas où il est la destination d'une gâchette) et d'autre part, au moins un de ses *pères* (i.e. ses nœuds antécédents) est actif (par défaut, une racine est toujours active). Ainsi, pour l'exemple proposé :

$$\begin{aligned} M_{L3} &= M_{G1} \vee M_{G2} \\ &= M_{G3} \vee (F_{G3} \wedge M_{G4}) \\ &= M_{G4} \vee F_{G3} \\ &= True \end{aligned}$$

Quand une feuille est activée (resp. désactivée), la distribution de probabilité sur les états de sa chaîne de Markov inactive (resp. active) est transférée sur les états de sa chaîne active (resp. inactive). Ce transfert peut être probabiliste. Les gâchettes permettent donc de modéliser les interactions entre ces processus stochastiques.

2. Dans [Bouissou and Bon, 2003], il est fait mention d'un troisième statut : le statut de *pertinence*. Il permet de réduire la complexité des algorithmes de calcul appliqués aux modèles. Il ne sera pas considéré dans ce travail, car il n'a pas d'incidence directe sur la modélisation.

2.2.2 Exemples de modélisation BDMP

Exemples de Processus de Markov Pilotés :

La Figure 2.2 représente trois PMP classiquement utilisés pour modéliser des composants génériques qui peuvent être activés et désactivés. Ces processus sont construits sur 4 états : S (Standby), F_1 (Défaillant et inactif), W (Working) et F_2 (Défaillant et actif). Les flèches dessinées en trait plein sont des transitions standards de chaîne de Markov continue (les étiquettes sont des taux de probabilité). Les flèches dessinées en pointillés représentent les fonctions de transfert (les étiquettes sont des valeurs de probabilité indépendantes du temps). Les états défaillants sont indiqués par une croix.

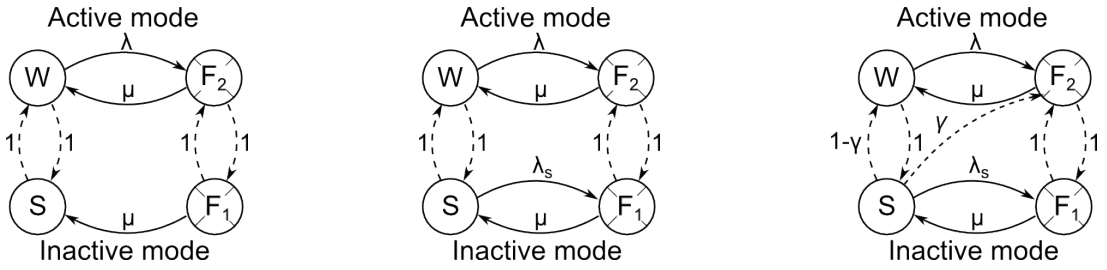

Figure 2.2 – Trois exemples de PMP caractéristiques (de gauche à droite : F, SF et SF_I [4])

Dans le premier cas (cf. processus de gauche), le composant peut être réparé alors qu'il est actif ou non (avec un taux de probabilité μ), en revanche il est susceptible de défaillir uniquement lorsqu'il est actif (avec un taux de probabilité λ). Dans le second cas (cf. processus du centre), le composant peut aussi défaillir lorsqu'il est inactif (avec un taux de probabilité λ_S). Dans le dernier cas (cf. processus de droite), le composant est susceptible de défaillir à la sollicitation (avec une probabilité γ), c'est à dire que lorsqu'il est activé, il y a une incertitude sur son état d'arrivée : W ou F_2.

Exemples de modèles BDMP :

Dans la suite, nous retranscrivons trois exemples proposés dans [Bouissou, 2005] afin d'illustrer les capacités de modélisation des BDMP.

2.2.2.1 Modélisation d'une redondance passive

La Figure 2.3 illustre l'utilisation la plus classique de la gâchette : la modélisation d'une redondance passive entre deux composants. Notons que la redondance entre sous-

4. F pour (simple) *Failure*, SF pour *Standby Failure*, et SF_I pour *Standby Failure with Immediate failure*.

systèmes se modélise de la même manière, en remplaçant les feuilles A et B par les modèles BDMP représentant la défaillance de ces sous-systèmes. Un exemple plus développé de modélisation de redondances passives peut être consulté dans [Carer et al., 2002].

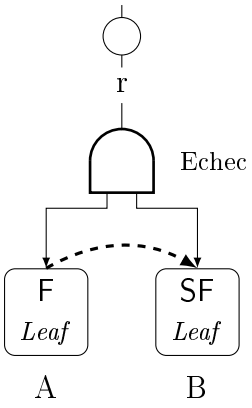

Figure 2.3 – Modélisation d'une redondance passive

La feuille B est activée (resp. désactivée) dès que la feuille A défaille (resp. est réparée). La feuille B sera associée à un PMP de type F ou SF selon que l'on souhaite retranscrire le comportement des portes "cold spare" ou "warm/hot spare" des arbres de défaillance dynamiques (cf. [Dugan et al., 1992]). Si par ailleurs, on souhaite modéliser que la feuille B peut défaillir à la sollicitation, on optera pour un PMP de type SF_I. Notons que la feuille A reste toujours active, et peut donc être associée à un PMP de type F.

2.2.2.2 Report de charge

Le report de charge est une stratégie de reconfiguration permettant d'améliorer la sûreté de fonctionnement d'un système. Il peut être mis en place lorsque deux composants (ou sous-systèmes) concourent à la réalisation d'une fonction, et que chacun peut assumer la charge de l'autre (souvent au prix d'un taux de défaillance plus élevé). Il s'agit de reporter la charge d'un composant sur l'autre. Si ce report doit s'effectuer en cas de défaillance d'un des deux composants, alors il peut être vu comme une stratégie de redondance fonctionnelle.

La Figure 2.4 représente un exemple de report de charge entre deux pompes dont le comportement fonctionnel et dysfonctionnel peut être décliné en trois modes (ce qui en fait des composants multi-états) :

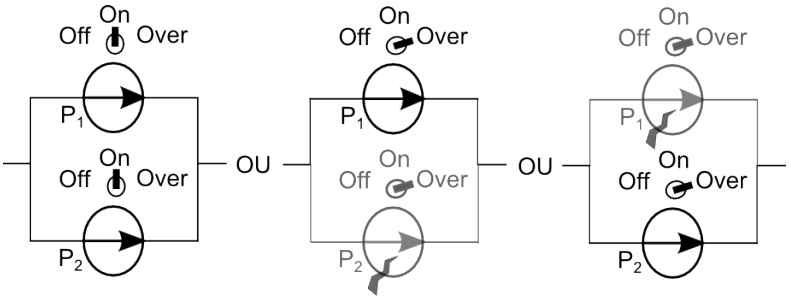

Figure 2.4 – Report de charge entre deux pompes

- *Off* : la pompe est inactive, elle réalise 0% de la fonction cible. Elle ne peut pas défaillir mais peut être réparée (taux de réparation μ).
- *On* : la pompe est active en régime nominal, elle réalise 50% de la fonction cible. Elle peut défaillir (taux de défaillance λ_1) et être réparée (taux de réparation μ).
- *Over* : la pompe est active en régime surchargé, elle réalise 100% de la fonction cible. Elle peut défaillir (taux de défaillance $\lambda_2 > \lambda_1$) et peut être réparée (taux de réparation μ).

La Figure 2.4 illustre que la fonction cible peut être réalisée de trois manières : les deux pompes sont actives en régime nominal et ne sont pas défaillantes ou au moins l'une des deux est active en régime surchargé et n'est pas défaillante.

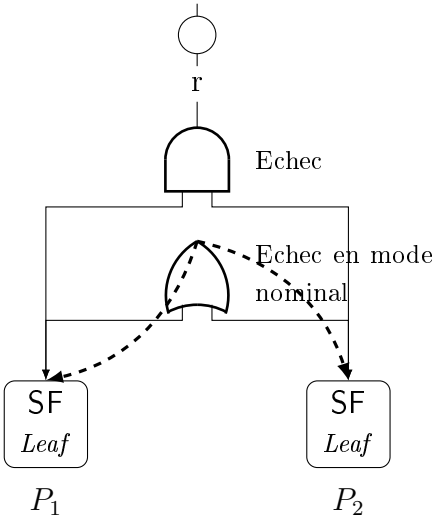

Figure 2.5 – Modélisation d'un report de charge

Ce cas peut être modélisé en BDMP, au prix d'un léger effort d'abstraction (cf. Figure 2.5) : le mode *Off* n'est pas considéré, le mode *On* prend le rôle du mode *inactif*, tandis

que le mode *Over* prend celui du mode *actif*. Dans ces conditions, une pompe peut être modélisée par une feuille de type SF, pour laquelle on pose : $\lambda_S = \lambda_1$ et $\lambda = \lambda_2$. Un autre exemple de modélisation de systèmes multi-états est traité dans [Bouissou, 2009].

2.2.2.3 Systèmes multi-phases

Le système modélisé par le BDMP représenté en Figure 2.6 réalise une mission divisée en deux phases. Durant la première phase, un seul composant est requis, tandis que dans la seconde, deux le sont. Ainsi, la structure du système change d'une phase à l'autre : d'un point de vue *diagramme de fiabilité*, les composants A et B fonctionnent *en parallèle* dans la première phase, et *en série* dans la seconde. Le composant C est en redondance passive de A et B.

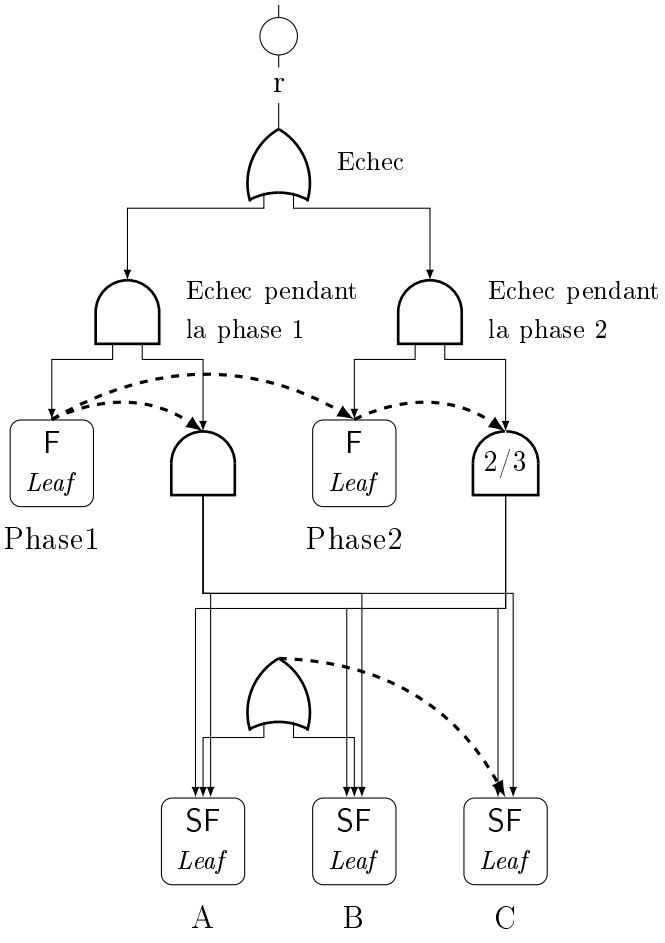

Figure 2.6 – Modélisation d'un système multi-phases

Ici, on fait l'hypothèse que le délai avant le déclenchement d'une phase est aléatoire et peut se modéliser par une feuille de BDMP associée à un PMP de type F. Dans ce cas, la "défaillance" d'une telle feuille doit alors se comprendre comme le déclenchement

de la phase correspondante. La gâchette entre les feuilles *Phase*1 et *Phase*2 permet de garantir la relation d'antécédence entre les déclenchements des deux phases. Un système multi-phases de taille plus conséquente est modélisé dans [Bouissou and Dutuit, 2004].

Ces trois exemples illustrent bien l'intérêt des BDMP : il est le seul formalisme basé sur les arbres de défaillance capable de modéliser des systèmes dynamiques réparables et permet même la modélisation d'un mécanisme de commutation basique (grâce à la primitive de gâchette).

2.2.3 Modélisation des mécanismes de commutation en BDMP

Nous discutons à présent du potentiel du formalisme BDMP pour supporter l'analyse de SdF basée sur les modèles des systèmes reconfigurables. La sous-section 2.2.2 fournit trois exemples de reconfiguration. Nous allons donc jeter un regard critique sur les modélisations BDMP proposées, afin de déterminer dans quelles limites le formalisme permet la prise en compte des mécanismes de commutation impliqués.

2.2.3.1 Discussion sur la modélisation des redondances passives

La redondance passive modélisée par une gâchette (cf. Figure 2.3) est régie par une stratégie faisant intervenir deux commutations :

- le *remplacement* : dès que le composant principal défaille, activer le composant de secours ;
- le *rétablissement* : dès que le composant principal est réparé, désactiver le composant de secours.

Ce modèle fait deux hypothèses implicites discutables : une seule stratégie de redondance passive est possible et son succès est garanti. Dès lors, il est impossible de considérer une autre stratégie de redondance (qui consisterait par ex. à ne procéder au rétablissement qu'en cas de défaillance du composant de secours). De plus, il n'existe aucun moyen de prendre en compte correctement la possible perte de la redondance. En effet, en supposant que l'exécution des mécanismes (remplacement et rétablissement) soit assurée par un troisième composant C, sa défaillance devrait empêcher le déclenchement de ces mécanismes.

Dans [Piriou et al., 2014b], il est montré que les différentes tentatives de modélisation de ce phénomène dans un BDMP sont infructueuses. Par exemple, considérons le modèle

représenté en Figure 2.7, qui semble constituer une bonne tentative.

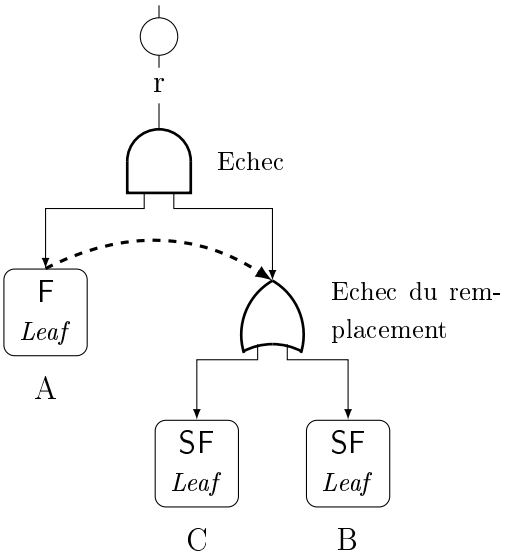

Figure 2.7 – Tentative de prise en compte d'un composant responsable de la commutation entre deux composants

Premièrement, l'échec du rétablissement n'est pas pris en compte. En particulier, ce modèle considère à tort que la séquence $\mathbf{f_A} \rightarrow \mathbf{f_C} \rightarrow \mathbf{r_A} \rightarrow \mathbf{f_B}$ [5] ne conduit pas à l'échec de la fonction cible. En effet, à la fin de cette séquence, le composant A est réparé et, comme il n'a été désactivé à aucun moment, il peut assurer le service. Par ailleurs, suite à la réparation de A, le bon rétablissement du service sur A ne devrait pas être possible, car C est indisponible pour déclencher ce mécanisme.

Deuxièmement, ce modèle traduit imparfaitement l'échec du remplacement. En effet, suite à la séquence $\mathbf{f_C} \rightarrow \mathbf{f_A}$, le modèle exprime bien que la fonction cible est perdue car A est défaillant et son remplacement a échoué. Pourtant, du point de vue du modèle, B a bien été activé. Son comportement dysfonctionnel est donc celui de son mode actif, qui est différent de celui de son mode inactif (cf. Figure 2.2). On prévoit ainsi que l'évaluation de la disponibilité du système, basée sur ce modèle sera pessimiste (dans le cas général où $\lambda > \lambda_S$).

De plus, C est ici considéré à l'instar d'un "actionneur monostable", dans le sens où il est nécessaire pendant toute la durée du remplacement (pas seulement au moment de la commutation). Ainsi, le modèle exprime que la séquence $\mathbf{f_A} \rightarrow \mathbf{f_C}$ conduit à la perte de la fonction cible, ce qui n'est pas vrai dans tous les cas, selon la nature de C.

5. f_X et r_X signifient respectivement défaillance et réparation du composant X

On a donc montré que la modélisation d'une redondance passive à l'aide d'une gâchette de BDMP, pouvait induire des imprécisions dans les résultats attendus des analyses qualitatives et quantitatives.

2.2.3.2 Modélisation des composants multi-états

Le report de charge (défini en sous-section 2.2.2.2) est un autre exemple de mécanisme de commutation. Dans ce cas, les composants ont plusieurs modes de fonctionnement : il ne s'agit pas simplement de les activer ou les désactiver, mais de les solliciter dans un mode particulier. Comme on a pu le voir, pour modéliser le report de charge à l'aide d'un BDMP, on est obligé d'ignorer le mode *Off*. Pour cause, une feuille de BDMP ne peut évoluer que dans deux modes. Dans le cas exposé dans la sous-section 2.2.2.2, cette abstraction est recevable car les pompes ne sont censées être désactivées à aucun moment.

Introduisons à présent une troisième pompe P_0 plus performante, capable de réaliser toute seule la fonction cible dans son régime nominal (cette pompe n'a pas de régime surchargé). On désire modéliser la stratégie de redondance suivante (on ne décrit ici que le remplacement, et non le rétablissement) :

- Initialement, P_0 réalise seule la fonction (P_1 et P_2 sont inactives) ;
- en cas de défaillance de P_0, activer P_1 et P_2 dans leur régime nominal ;
- en cas de défaillance de P_1 ou P_2, reporter la charge sur le composant indemne.

Afin de modéliser ce cas en BDMP, il est nécessaire de partitionner le comportement des pompes P_1 et P_2 entre trois feuilles de BDMP de type F. Les deux premières feuilles représentent les composants dans les modes *Off* et *On*, la troisième représente le mode *Over* de tantôt P_1, tantôt P_2 (selon les scénarios). Cette proposition est représentée par la Figure 2.8.

La difficulté ici est de prendre en compte l'exclusivité des différents modes d'opération d'un composant. Selon ce modèle, après la séquence $\mathbf{f_{P_0}} \rightarrow \mathbf{f_{P_1}}$, la pompe P_2 évolue à la fois en mode *On* et *Over*. Dès lors, le composant P_2 est susceptible de défaillir en mode *On* tout en continuant à être opérationnel en mode *Over*, ce qui constitue une incohérence. Cette difficulté renvoie, entre autre, à une limite du pouvoir d'expression de la gâchette identifiée au chapitre précédent. En effet, une solution ici consisterait à

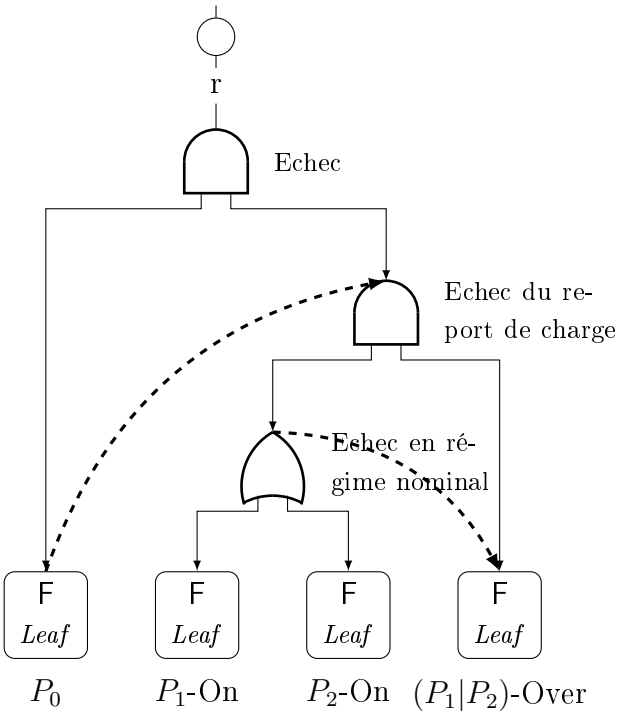

Figure 2.8 – Tentative de modélisation BDMP de composants multi-états

modifier le comportement de la gâchette de telle sorte que lors du report de charge, non seulement la destination est activée, mais aussi l'origine est désactivée. Or, comme on l'a mentionné plus haut, la stratégie de commutation appliquée par une gâchette est unique et non paramétrable.

L'autre inconvénient souligné dans l'exemple précédent est également observable dans celui-ci : on ne peut pas modéliser de façon réaliste l'échec possible des commutations liées au report de charge.

En outre, avec cette modélisation, on perd un atout précieux des BDMP : pouvoir modéliser le comportement d'un composant par une seule et même feuille. Or, cet atout est ce qui permet la clarté et la compacité des modèles.

2.2.3.3 Modélisation des systèmes multi-phases avec modification des critères de succès des composants

Dans l'exemple de système multi-phases proposé en sous-section 2.2.2.3, la structure du système change entre les deux phases. Le critère de succès du système est donc également modifié (en phase 1, un seul composant est requis ; en phase 2, les deux le sont). En revanche, le critère de succès des composants n'est pas modifié. Or, il existe

des cas de systèmes multi-phases, où un composant est considéré opérationnel dans une phase et non opérationnel dans une autre, alors que son état dysfonctionnel n'a pas changé.

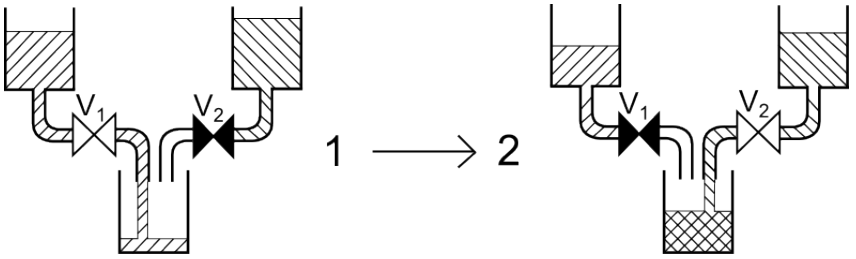

Figure 2.9 – Deux vannes réalisant un mélange en deux phases

Pour illustrer ce point, nous considérons à présent un procédé simple illustré par la Figure 2.9. Deux vannes TOR sont utilisées pour mélanger deux réactifs chimiques. La réaction étant sensible, il est indispensable que l'opération soit exécutée en deux phases :

- dans la première phase, V_1 doit être ouverte et V_2 fermée, afin de remplir la cuve avec le premier réactif ;
- dans la seconde phase, V_2 doit être ouverte et V_1 fermée, afin de remplir la cuve avec le second réactif ;

On considère que les vannes V_1 et V_2 peuvent se bloquer alors qu'elle sont ouvertes ou fermées. Une vanne bloquée sera alors considérée opérationnelle dans une phase et non opérationnelle dans l'autre. Par exemple, si V_1 est bloquée ouverte, elle réalise correctement le service dans la première phase, mais pas dans la seconde.

Dans le modèle BDMP de ce système (reporté en Figure 2.10), on est obligé de distinguer chaque composant selon la phase de mission dans laquelle il est observé. Ainsi par exemple, le comportement de la première vanne est réparti entre les feuilles V_1-ouverte et V_1-fermée selon qu'il est observé dans la phase 1 ou dans la phase 2. Les correspondances entre les états du PMP (de type SF_I, cf. Figure 2.2) associé à ces feuilles et les états de la vanne en question sont reportées dans le tableau 2.1.

Le modèle représente donc bien la fermeture de cette vanne lorsque la phase 2 commence, ainsi que le fait qu'elle puisse rester bloquée ouverte. En revanche, la stratégie de reconfiguration associée par nature à la gâchette implique que la feuille V_1-ouverte reste active dans la phase 2. Ainsi, le modèle peut évoluer dans des états contradictoires où

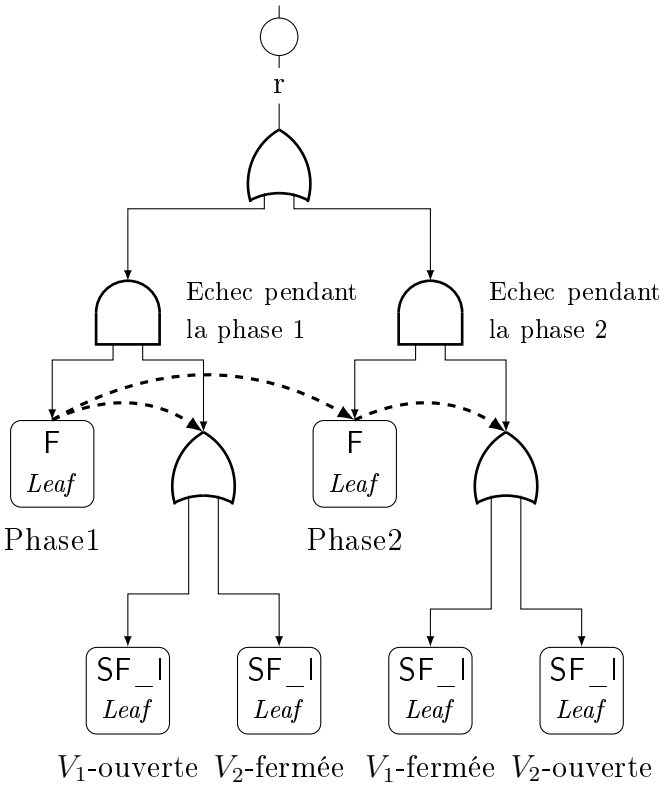

Figure 2.10 – Modélisation BDMP d'un système multi-phases

états d'un PMP de type SF_I	correspondances pour V_1-ouverte	correspondances pour V_1-fermée
S	fermée	ouverte
W	ouverte	fermée
F_1	bloquée fermée - devant être fermée	bloquée ouverte - devant être ouverte
F_2	bloquée fermée - devant être ouverte	bloquée ouverte - devant être fermée

Tableau 2.1 – Correspondances entre les états de la vanne et ceux du PMP associé aux feuilles la modélisant

la vanne est, par exemple, à la fois fermée (selon la feuille V_1-fermée) et ouverte (selon la feuille V_1-ouverte).

Pour les mêmes raisons que dans les cas précédents (contraintes associées à la modélisation des mécanismes de commutation), la modélisation BDMP de ce système est insatisfaisante.

2.2.4 Bilan sur la modélisation BDMP

Ce chapitre a permis de rappeler la définition du formalisme BDMP et de montrer par l'exemple ses capacités et limites pour modéliser la SdF des systèmes critiques. En particulier, le fait que ce formalisme soit basé sur des arbres de défaillance permet une modélisation guidée par la structure, très adaptée aux compétences des ingénieurs experts en SdF. En outre, l'ajout de processus de Markov pour décrire de manière spécifique les comportements élémentaires du système constitue un atout précieux des BDMP par rapport aux autres formalismes basés sur les arbres de défaillance.

Néanmoins, nous avons exhibé certaines approximations induites par le formalisme. La primitive de gâchette est notamment interprétée en faisant deux hypothèses implicites qui contraignent la modélisation des mécanismes de commutation : une seule stratégie est possible et son succès est garanti. On a montré que ces contraintes constituaient un obstacle à une modélisation réaliste et précise des stratégies de reconfiguration d'un système critique.

Par ailleurs, les comportements dysfonctionnels modélisables à l'aide des processus de Markov définis dans ce formalisme correspondent à des composants n'ayant que deux modes d'opération. Il a alors été montré que la modélisation BDMP de composants ayant plus de deux modes est souvent problématique.

Le tableau 2.2 récapitule cette analyse critique du formalisme.

Caractéristiques	**Avantages**	**Inconvénients**
structure d'arbre de défaillance	modélisation intuitive	
modélisation spécifique du comportement d'une feuille via un PMP	modélisation d'un composant par une seule feuille...	...s'il n'a pas plus de 2 modes d'opération
primitive de gâchette	permet de prendre en compte des mécanismes de commutation (et plus généralement introduit la notion de contrôle)	une seule stratégie de reconfiguration peut être transcrite, et son succès est supposé garanti

Tableau 2.2 – Principaux avantages et inconvénients du formalisme BDMP

2.3 Définition des BDMP Généralisés

Dans cette section, le formalisme BDMP est étendu afin d'enrichir son pouvoir d'expression. Cet enrichissement doit permettre une modélisation réaliste et précise des

mécanismes de commutation et des effets de leur échec sur la sûreté de fonctionnement du système. Le nouveau formalisme préserve le principe fondamental de processus Markoviens pilotés par une logique Booléenne, mais l'étend afin de pouvoir considérer n'importe quelle stratégie de reconfiguration, faisant intervenir des composants ayant un nombre quelconque de modes d'opération. Pour cette raison, nous parlerons de *BDMP Généralisés* (GBDMP). Ce formalisme s'inspire et renforce les travaux développés dans [Piriou et al., 2015].

2.3.1 Définition de la syntaxe

Cette sous-section décrit les définitions syntaxiques permettant de construire et de comprendre un modèle GBDMP.

Définition 2.1. *Un Generalized Boolean logic Driven Markov Processes est un 6-tuple* $< V, E, \kappa, \upsilon, str, pmc >$ *tel que :*

- $V = N \cup S = G \cup L \cup S$ *est un ensemble de sommets partitionné entre les nœuds* N *(i.e. les portes* G *et les feuilles* L*) et les commutateurs* S *;*
- $E = E_F \cup E_S$ *est un ensemble d'arcs orientés, connectant les portes à leurs fils :* $E_F \subseteq G \times N$ *; et les commutateurs à leurs nœuds d'entrée et de sortie :* $E_S \subseteq (N \times S) \cup (S \times N)$ *;*
- $\kappa : G \to \mathbb{N}^*$ *est une fonction permettant de déterminer le type logique des portes (comme pour les BDMP) ;*
- $\upsilon : E \to \mathbb{N}$ *est une fonction associant un paramètre à chaque arc ;*
- $str : S \to \mathbb{M}$ *est une fonction associant une machine de Moore - i.e. une logique de contrôle - à chaque commutateur*[6] *;*
- $pmc : L \to \mathbb{P}$ *est une fonction associant un Processus de Markov Commuté (PMC) - i.e. un comportement élémentaire - à chaque feuille*[7]*.*

La représentation graphique d'un exemple didactique de modèle GBDMP est reportée sur la Figure 2.11. Elle est divisée en trois parties : une vue structurelle qui se réfère à deux vues comportementales. La vue structurelle occupe la partie gauche. Les

6. $\mathbb{M}$ désigne l'ensemble des machines de Moore.
7. $\mathbb{P}$ désigne l'ensemble des PMC.

feuilles sont représentées par des rectangles arrondis ($L = \{C1, C2, C3\}$). Les portes sont dessinées comme des portes logiques ($G = \{G1, G2, G3\}$, avec $G1$ une porte ET : $\kappa(G1) = 2$) ; $G2$ et $G3$ des portes OU : $\kappa(G2) = \kappa(G3) = 1$). Les commutateurs sont dessinés en rectangles pointillés ($S = \{S1\}$). Les arcs orientés de E_F sont dessinés en traits pleins, tandis que les arcs de E_S sont dessinés en pointillé. La fonction υ est représentée à travers les étiquettes associées à ces arcs. Finalement, $str(S1) = M0$ et $pmc(C1) = pmc(C2) = pmc(C3) = SF$. Les modèles $M0$ et SF sont représentés sur la partie droite de la Figure 2.11. Enfin, aucune racine de l'arbre n'est désignée comme étant la racine principale, lors de la construction du modèle. Le choix sera fait lors de l'analyse. De cette manière, il est possible de baser plusieurs analyses (caractérisées par une fonction cible, une mission...) sur un seul et même modèle d'un système complexe réalisant plusieurs fonctions, selon différentes configurations.

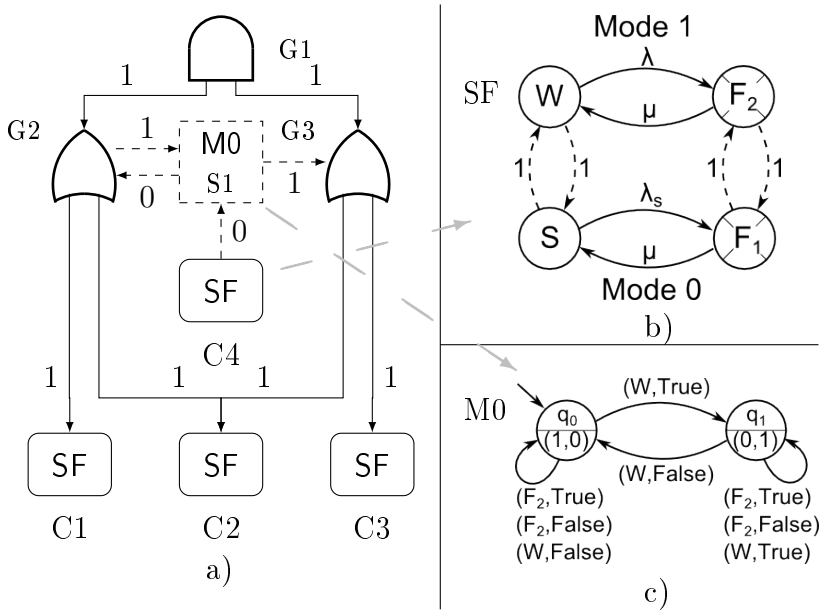

Figure 2.11 – Exemple didactique de modèle GBDMP ; a) vue *arbre de défaillance enrichi* ; b) vue *PMC* ; c) vue *machine de Moore*

Ainsi, un modèle GBDMP est défini comme un **arbre de défaillance**, construit sur des comportements élémentaires décrits par des **PMC**, contrôlés par des commutateurs, dont les logiques de contrôle sont décrites par des **machines de Moore**. Les différents aspects de la complexité du système sont répartis entre ces trois facettes du modèle. La définition des PMC et des machines de Moore, ainsi que les raisons de leur emploi dans le formalisme seront détaillées par la suite.

2.3.1.1 Processus de Markov Commutés

Comme pour les BDMP, un modèle GBDMP est basé sur des comportements élémentaires décrits par des processus de Markov.

Un modèle de type PMP (cf. sous-section 2.2.1) permet de modéliser des composants n'ayant que deux modes d'opération : *actif* et *inactif*. Nous proposons de généraliser le concept de PMP, en introduisant les Processus de Markov Commutés (PMC). Comme on peut le lire formellement dans la définition 2.2, un k-PMC est défini par une famille de k chaînes de Markov, une famille de $k(k-1)$ fonctions de transfert probabilistes, et un partitionnement de l'espace d'état pour distinguer les états défaillants des autres. Si une feuille est associée à un k-PMC, on dira qu'elle est de dimension k.

Définition 2.2. *Un Processus de Markov Commuté à k modes (k-PMC) P est un 3-tuple* $< (\mathcal{Z}_i^P)_{0\leq i<k}, \mathcal{X}_F^P, (f_{i\to j}^P)_{(i,j)\in[\![0,k-1]\!]^2} >$ *tel que :*

- $(\mathcal{Z}_i^P)_{0\leq i<k}$ *est une famille de chaînes de Markov homogènes i.e.* $\forall i \in [\![0, k-1]\!]$, $\mathcal{Z}_i^P$ *est un 3-tuple* $< \mathcal{X}_i^P, A_i^P, p0_i^P >$ *tel que :*
  - $\mathcal{X}_i^P$ *est un ensemble fini d'états ;*
  - $A_i^P : (\mathcal{X}_i^P)^2 \to \mathbb{R}^+$ *est la matrice des taux de transition ;*
  - $p0_i^P : \mathcal{X}_i^P \to [0,1]$ *est la distribution initiale de probabilité, définie telle que* $\sum_{x\in\mathcal{X}_i^P} p0_i^P(x) = 1$ *;*
- $\mathcal{X}_F^P \subseteq \mathcal{X}^P$ *(avec* $\mathcal{X}^P = \bigcup_{i=0}^{k-1} \mathcal{X}_i^P$*) est le sous-ensemble des états défaillants ;*
- $(f_{i\to j}^P)_{(i,j)\in[\![0,k-1]\!]^2}$ *est une famille de fonctions de transfert probabilistes :*
  - $\forall (i,j) \in [\![0,k-1]\!]^2$, *avec* $i \neq j$ *la fonction* $f_{i\to j}^P : \mathcal{X}_i^P \times \mathcal{X}_j^P \longrightarrow [0,1]$ *est définie tel que* $\forall x \in \mathcal{X}_i^P, \sum_{y\in\mathcal{X}_j^P} f_{i\to j}^P(x,y) = 1.$

On peut remarquer qu'un PMP est homogène à un 2-PMC. Aussi, un PMC se représente graphiquement de la même manière qu'un PMP (comme on peut le voir sur la partie b) de la Figure 2.11).

L'introduction de cette nouvelle primitive induit une modification sur la construction de l'arbre de défaillance. En effet, à présent qu'une feuille peut être commutée sur plus de deux modes, son statut d'*activité* ne peut plus être traduit par une simple

variable Booléenne (comme dans les BDMP). Les modes d'opération d'un composant étant dénombrables, ce statut sera représenté par une variable entière. Il faudra de surcroit définir une nouvelle manière de le calculer, en fonction de la structure et de l'évolution du modèle. Aussi, comme on le verra dans la sous-section 2.3.2, les *paramètres* associés aux arcs connectant les portes à leurs fils interviennent dans le calcul de ces statuts.

2.3.1.2 Machines de Moore

L'objectif de la primitive *commutateur* est de généraliser le principe de la gâchette, de telle sorte que n'importe quelle stratégie de reconfiguration puisse être traduite, avec ses possibles échecs.

Comme le suggère la définition 2.3, les machines de Moore (cf. [Moore, 1956]) sont des automates à entrées/sorties, telle que les symboles d'entrée et de sortie sont associés respectivement aux transitions et aux états.

Définition 2.3. *Une machine de Moore M est un 6-tuple* $< Q^M, q_0^M, \Sigma_I^M, \Sigma_O^M, trans^M, out^M >$ *tel que :*

- Q^M *est un ensemble fini d'états ;*
- $q_0^M \in Q^M$ *est l'état initial*[8] *;*
- Σ_I^M *est l'alphabet d'entrée ;*
- Σ_O^M *est l'alphabet de sortie ;*
- $trans^M : Q^M \times \Sigma_I^M \to Q^M$ *est la fonction de transition (cette fonction permet de changer d'état suite à la réception d'un signal d'entrée) ;*
- $out^M : Q^M \to \Sigma_O^M$ *est la fonction de sortie (cette fonction permet d'émettre un signal de sortie, associé à un état).*

Une machine de Moore se représente graphiquement comme un automate, dont les transitions sont étiquetées par les symboles d'entrée, et les états sont étiquetés par les symboles de sortie. Par exemple pour la machine de Moore représentée dans la partie

8. Pour cette définition, on impose que l'état initial soit unique, de sorte que la machine de Moore soit déterministe.

c) de la Figure 2.11, la transition entre les états q_0 et q_1 est étiquetée par le symbole d'entrée $(W, True) \in \Sigma_I$, et l'état q_0 est étiqueté par le symbole de sortie $(1, 0) \in \Sigma_O$.

Comme l'illustre la figure 2.12, une machine de Moore est particulièrement adaptée pour spécifier une *logique de contrôle*. Le principe d'une telle logique est d'observer l'évolution d'une ou plusieurs variables d'entrée, et d'en déduire des consignes d'actualisation d'une ou plusieurs variables de sortie. Dans ce cas, l'alphabet d'entrée Σ_I (resp. de sortie Σ_O) de la machine de Moore est l'ensemble de tous les vecteurs de valeurs d'entrée (resp. de sortie) possibles.

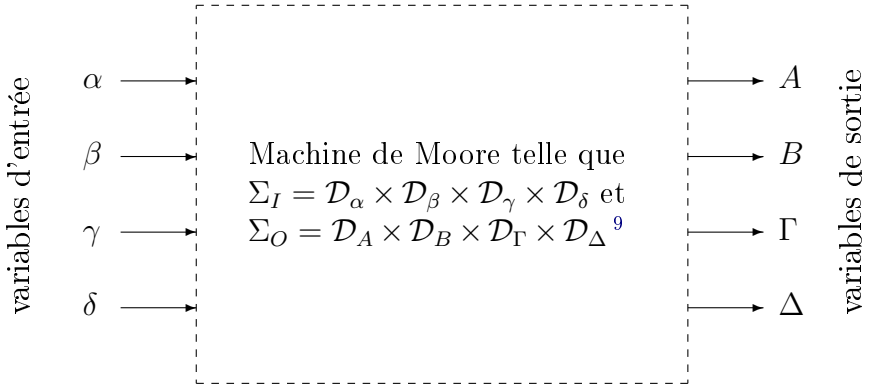

Figure 2.12 – Principe de l'utilisation d'une machine de Moore pour spécifier une logique de commande

Il a par ailleurs été suggéré dans la sous-section 1.2.3, qu'une stratégie de reconfiguration pouvait se voir comme une logique de contrôle. En effet, il s'agit de contrôler dans quel mode d'opération chaque composant doit être activé (ou désactivé), en fonction d'un certain contexte. Par exemple, dans le cas d'une stratégie de redondance passive, ce contexte est principalement caractérisé par les états dysfonctionnels d'un ensemble d'entités (composants ou sous-systèmes). Ainsi, une machine de Moore peut être utilisée pour spécifier une stratégie de reconfiguration.

A titre d'exemple, la stratégie traduite par une gâchette de BDMP peut être modélisée par la machine de Moore représentée sur la Figure 2.13. Une gâchette observe effectivement une variable d'entrée : le statut de *défaillance* du nœud *origine* (F_{orig}), et contrôle une variable de sortie : le statut d'*activité* du nœud *destination* (M_{dest}). Ce contrôle est régi par les règles suivantes :

- lorsque l'entrée est vraie, activer la sortie ;

9. où Σ_I et Σ_O sont les alphabets respectivement d'entrée et de sortie ; $\mathcal{D}_v$ est le domaine de définition de la variable v ; L'opérateur $\times$ est le produit cartésien d'ensemble.

- lorsque l'entrée est fausse, désactiver la sortie ;

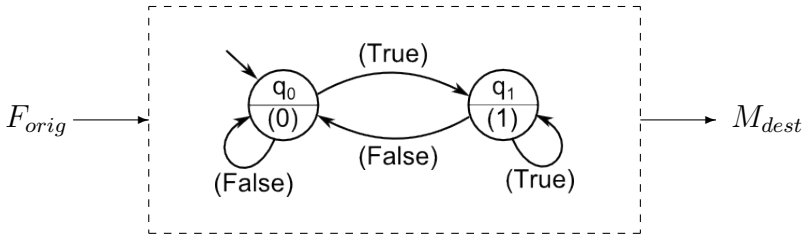

Figure 2.13 – Machine de Moore modélisant une gâchette de BDMP

Dans le cadre des GBDMP, de même que le comportement dysfonctionnel d'une *feuille* peut être spécifié à l'aide d'un PMC, la stratégie de reconfiguration d'un *commutateur* peut être spécifiée à l'aide d'une machine de Moore. Les paramètres associés aux arcs connectant un commutateur à ses nœuds d'entrée et de sortie permettent de déterminer l'ordre des variables dans les vecteurs d'entrée et de sortie de la machine de Moore. Ainsi, pour l'exemple didactique proposé sur la Figure 2.11, le commutateur $S1$ a deux nœuds d'entrée $(C4, G2)$ et deux nœuds de sortie $(G2, G3)$. La stratégie de reconfiguration qu'il traduit est spécifiée par la machine de Moore $M0$. Les symboles d'entrée et de sortie de cette machine sont des vecteurs de deux valeurs. Par exemple, le vecteur $(W, True)$ est constitué de W : une valeur prise par une variable relative à la feuille $C4$ (le paramètre 0 de l'arc connectant $C4$ à $S1$ correspond au rang de W dans le vecteur) ; et de $True$: une valeur prise par une variable relative à la porte $G2$ (le paramètre 1 de l'arc connectant $G2$ à $S1$ correspond au rang de $True$ dans le vecteur).

2.3.1.3 Contraintes de construction d'un modèle GBDMP

Afin d'être correctement interprété, un modèle GBDMP doit respecter sept règles. On dira que le modèle est *bien formé* s'il respecte ces règles. Dans la suite, elles seront énoncées et formalisées, en utilisant des notations usuelles de la théorie des graphes (ces notations sont rappelées dans l'annexe A). Nous définissons alors trois graphes construits à l'aide des éléments d'un modèle GBDMP. $\mathcal{G}_F =< N, E_F >$ désigne le graphe classiquement appelé l'*arbre de défaillance* ; $\mathcal{G}_S =< V, E_S >$ désigne le graphe permettant de déterminer les nœuds d'entrée et de sortie des commutateurs ; et $\mathcal{G} = \mathcal{G}_F \cup \overline{\mathcal{G}_S}$ [10] désigne le graphe global qui est utilisé pour formaliser la dernière règle.

10. $\overline{\mathcal{G}_S}$ est le graphe $\mathcal{G}_S$ dont la direction des arcs a été inversée

Les trois premières règles peuvent être décrites dès à présent. En revanche, à ce stade de l'exposé, il manque encore quelques éléments pour bien comprendre les quatre dernières ; c'est pourquoi celles-ci seront exposées plus loin dans le document.

La règle 1 traduit simplement le fait qu'une porte logique de type *k parmi n* n'a de sens que si $k \leq n$.

Règle 1. *Le type d'une porte doit être compatible avec le nombre de ses fils :*

$$\forall g \in G, \kappa(g) \leq d^+_{\mathcal{G}_F}(g)$$

La règle 2 garantit que l'activité d'un nœud n'est pas contrôlée par plusieurs stratégies de reconfiguration contradictoires. Dans le cas contraire, le modèle risquerait d'évoluer dans un état où une feuille doit être commutée dans deux modes d'opération différents en même temps, ce qui est impossible.

Règle 2. *Un nœud ne peut pas être en sortie de plusieurs commutateurs :*

$$\forall n \in N, d^-_{\mathcal{G}_S}(n) \leq 1$$

On a vu dans la sous-section 2.3.1.2 que les paramètres des arcs connectant les commutateurs à ses nœuds d'entrée et de sortie devaient permettre de déterminer un ordre entre des variables relatives à ces nœuds. Aussi, la règle 3 permet de s'assurer que si un commutateur est connecté à a nœuds d'entrée et b nœuds de sortie, les arcs matérialisant ces connections seront respectivement numérotés (via leur paramètre) de 0 à $a-1$ et de 0 à $b-1$.

Règle 3. *Les entrées et sorties d'un commutateur doivent être correctement numérotées :*

$$\forall s \in S : \{\upsilon((n,s))|(n,s) \in E_S\} = [\![0, d^-_{\mathcal{G}_S}(s) - 1]\!]$$
$$\textit{et } \{\upsilon((n,s))|(s,n) \in E_S\} = [\![0, d^+_{\mathcal{G}_S}(s) - 1]\!]$$

Cette règle permet de définir une nouvelle notation : dans la suite, $In(s,k)$ désigne le i-ème nœud d'entrée de s, i.e. le nœud $n \in N$ tel que : $(n,s) \in E_S \wedge \upsilon((n,s)) = k$. Sont également introduites les notations $I^-(s) = [\![0, d^-_{\mathcal{G}_S}(s) - 1]\!]$ et $I^+(s) = [\![0, d^+_{\mathcal{G}_S}(s) - 1]\!]$.

2.3.2 Définition de la sémantique

Dans cette sous-section, les primitives des GBDMP sont formellement interprétées, dans le but de déterminer le comportement transcrit par un modèle.

2.3.2.1 Interprétation des primitives

Pour chaque feuille $l \in L$, on notera $X_l \in \mathcal{X}^{pmc(l)}$ la variable d'état du PMC associé. Pour chaque commutateur $s \in S$, on notera $U_s \in Q^{str(s)}$ la variable d'état de la machine de Moore associée. Ces deux ensembles de variables suffisent à décrire complètement l'espace d'état du modèle GBDMP. Mais pour calculer l'évolution de ce modèle, on définit trois autres variables pour chaque nœud $n \in N$, dont les valeurs peuvent être déduites de celles des variables d'état :

- F_n : une variable Booléenne qui détermine le statut de *défaillance* d'un nœud ($F_n = True \Leftrightarrow n$ est défaillant) ;
- R_n : une variable binaire [11] qui détermine le statut de *réquisition* d'un nœud ($R_n = 1 \Leftrightarrow n$ est requis) ;
- M_n : une variable entière positive qui détermine le statut d'*activité* d'un nœud ($M_n = k \Leftrightarrow n$ est dans le mode d'opération numéro k) ;

Dans la suite, nous expliquons comment les variables du type F_n, R_n, M_n et U_s peuvent être déduites des variables du type X_l. Dans un souci d'illustration, chaque formule sera appliquée sur l'exemple didactique de la Figure 2.11.

a) Détermination de la valeur des statuts de défaillance

Comme avec les BDMP, le statut de *défaillance* d'une feuille $l \in L$ est déterminé par le PMC qui lui est associé :

$$X_l \in \mathcal{X}_F^{pmc(l)} \Longrightarrow F_l = True \tag{2.1}$$

par ex. : $X_{C1} \in \{F_1, F_2\} \Longrightarrow F_{C1} = True$.

En revanche, le calcul du statut de *défaillance* d'une porte est modifié par rapport aux BDMP. Avec les BDMP, un nœud non défaillant est considéré opérationnel, même s'il est inactif. Comme indiqué dans la sous-section 1.2.3, un composant (ou sous-système) ne réalise sa fonction que s'il est non défaillant *et requis*. En d'autres termes, si un nœud non défaillant n'est pas requis par le commutateur qui le contrôle, il ne doit pas être considéré comme étant opérationnel. Ainsi la valeur du statut de *défaillance* d'une porte est calculée en considérant le nombre de ses fils défaillants ou non requis. Ainsi,

11. Pour des raisons de cohérence des calculs, on distinguera les valeurs Booléennes $(False, True)$, des valeurs binaires $(0, 1)$. Contrairement aux premières, les secondes sont des entiers et peuvent donc être multipliées par un entier.

une porte $g \in G$ est défaillante si le nombre de ses fils non opérationnels est plus grand que $\kappa(g)$:

$$Card(\{n \in \Gamma^{+}_{\mathcal{G}_F}(g) | F_n \vee \neg R_n\}) \geq \kappa(g) \Longrightarrow F_g = True \tag{2.2}$$

par ex. : $Card(\{n \in \{G2, G3\} | F_n \vee \neg R_n\}) \geq 2 \Longrightarrow F_{G1} = True$.

b) Actualisation des variables d'état des commutateurs

Un commutateur $s \in S$ sert à contrôler le statut de *réquisition* de ses nœuds de sortie en fonction de l'état dysfonctionnel de ses nœuds d'entrée. Le statut de *défaillance* F_n d'un nœud d'entrée n est souvent suffisant pour caractériser son état dysfonctionnel. Mais certaines stratégies de commutation requièrent plus de détails sur l'état dysfonctionnel d'une feuille l (par ex. une stratégie peut varier selon le mode de défaillance d'une feuille). La variable d'entrée à considérer n'est alors plus F_l mais X_l. Aussi, la fonction $trans^{str(s)}$ permet d'actualiser la variable U_s, une fois que les valeurs de ces variables sont connues.

c) Détermination de la valeur des statuts de réquisition

En outre, si $n \in N$ est un nœud de sortie d'un commutateur $s \in S$ (unique grâce à la règle 2), son statut de *réquisition* est déterminé par la machine de Moore associée à s. Sinon, il est toujours requis :

$$\begin{cases} si\ \exists s \in S | (s,n) \in E_S & R_n = (out^{str(s)}(U_s))_{v((s,n))} \\ sinon & R_n = 1 \end{cases} \tag{2.3}$$

par ex. : $R_{G3} = o_1$ avec $(o_0, o_1) = out^{M1}(U_{S1})$

Ce nouvel éclairage permet d'exposer la règle 4 qui doit être respectée pour que chaque commutateur soit compatible avec la machine de Moore qui lui est associée. En effet, comme un commutateur contrôle des statuts de *réquisition*, et que ceux-ci sont représentés par des valeurs binaires, les symboles de sortie de la machine de Moore ne peuvent être que des vecteurs de valeurs binaires. De même, comme les variables d'entrée d'un commutateur sont ou bien des statuts de *défaillance* de feuille ou de porte, ou bien des variables d'état de feuille, les symboles d'entrée de la machine de Moore ne peuvent être que des vecteurs composés ou bien de valeurs Booléennes, ou bien d'état de PMC. On peut remarquer que l'exemple didactique (cf. Figure 2.11) respecte la règle 4. En effet, les symboles de sortie sont construits à l'aide des deux valeurs possibles pour les statuts de *réquisition* de $G2$ et $G3$ - à savoir 0 et 1 -. De même, les symboles d'entrée sont construits à l'aide d'un des états du PMC associé à $C4$ - à savoir, W, S, F_1 et F_2 -,

et d'une des deux valeurs possibles pour le statut de *défaillance* de $G2$ - à savoir $False$ et $True$ -.

Règle 4. *Les machines de Moore associées aux commutateurs doivent être compatibles avec leurs entrées/sorties :*

$\forall s \in S$:

$- \ \forall i \in \Sigma_I^{str(s)} : i = (i_k)_{k \in I^-(s)}|$

$$\forall k \in I^-(s), \begin{cases} si\ In(s,k) \in L : & i_k \in \{False, True\} \vee i_k \in \mathcal{X}^{pmc(In(s,k))} \\ sinon : & i_k \in \{False, True\} \end{cases}$$

$- \ \forall o \in \Sigma_O^{str(s)} : o = (o_k)_{k \in I^+(s)} | \forall k \in I^+(s), o_k \in \{0, 1\}$

d) Détermination de la valeur des statuts d'activité

Enfin, comme on l'a indiqué dans la sous-section 2.3.1.1 la nature et le calcul du statut d'*activité* d'un nœud sont modifiés par rapport aux BDMP. Sa valeur est déterminée par un calcul faisant intervenir le statut de *réquisition* du nœud concerné, le statut d'*activité* de ses pères et les paramètres des arcs le connectant à ses pères (cf. equation 2.4). Ce calcul formalise qu'un nœud est dans le mode i (i.e. son statut d'*activité* vaut i), si les trois conditions suivantes sont vérifiées :

- le nœud considéré est requis (i.e. son statut de *réquisition* est à 1) ;
- une branche de l'arbre le sollicite dans le mode i (i.e. i est le produit de la valeur du statut d'activité d'un de ses pères et du paramètre de l'arc le connectant à ce père [12]) ;
- aucune branche du système ne le sollicite dans un mode j plus élevé ($j > i$).

Cette interprétation induit un ordre de priorité implicite entre les modes d'opération d'un composant, qu'il conviendra de prendre en compte lors de la construction des modèles. Par convention, pour les composants pouvant être désactivés, le mode 0 représente le mode inactif.

12. Ainsi, il faut voir ce paramètre comme un moyen de caractériser la branche de l'arbre issue de cet arc, pour qu'elle modélise une configuration particulière.

Le statut d'*activité* d'un nœud $n \in N$ est donc calculé à l'aide de la formule :

$$\begin{cases} si\ \Gamma^-_{\mathcal{G}_F}(n) \neq \varnothing & M_n = R_n.\ \max\limits_{g \in \Gamma^-_{\mathcal{G}_F}(n)} (M_g.\upsilon((g,n))) \\ \\ sinon & M_n = R_n \end{cases} \tag{2.4}$$

par ex. : $M_{C2} = R_{C2}.\max(M_{G2}, M_{G3})$

Le statut d'*activité* d'une feuille l correspond au mode d'opération dans lequel elle doit être commutée. Ainsi, le PMC associé à l doit toujours évoluer de telle sorte que la chaîne de Markov active soit la chaîne $\mathcal{Z}^{pmc(l)}_{M_l}$. Pour que ce comportement soit possible, le modèle doit être construit de telle sorte qu'une feuille ne puisse pas être requise dans un mode qui ne correspond à aucune chaîne de Markov. Par exemple, si une feuille l est de dimension 3 (i.e. $pmc(l)$ est composé de 3 chaînes de Markov), son statut d'*activité* M_l devra toujours être compris entre 0 et 2. Toutefois, le cas est particulier pour les feuilles de dimension 1. Celles ci modélisent des composants qui n'ont qu'un seul mode d'opération, *a priori* d'activité. Or, comme le mode 0 représente, par convention, un mode d'inactivité (par opposition aux modes d'activité : 1,2,3...), l'unique chaîne de Markov de $pmc(l)$ sera associée au mode 1 (et non 0). Ces contraintes de construction sont formalisées par la règle 5 ci-dessous.

Règle 5. *Le PMC associé à un composant doit être compatible avec* $\mathcal{G}_F$ *:*

$$\forall l \in L, \begin{cases} l\ est\ de\ dimension\ 1 & \Longrightarrow\ \Upsilon(l) = 1 \\ l\ est\ de\ dimension\ k{>}1 & \Longrightarrow\ \Upsilon(l) < k \end{cases}$$

avec Υ *une fonction qui calcule la borne maximale du statut d'activité d'un nœud dans un modèle GBDMP :*

$$\Upsilon : \begin{array}{ccc} N & \to & \mathbb{N} \\ n & \mapsto & \begin{cases} if\ d^-_{\mathcal{G}_F}(n) = 0 & 1 \\ else & \max\limits_{g \in \Gamma^-_{\mathcal{G}_F}(n)} (\Upsilon(g).\upsilon((g,n))) \end{cases} \end{array}$$

On peut remarquer que l'exemple didactique de GBDMP proposé (cf. Figure 2.11) respecte la règle 5. En effet, pour ce modèle on peut calculer : $\Upsilon(C1) = \Upsilon(C2) = \Upsilon(C3) = \Upsilon(C4) = 1$; or, toutes les feuilles sont de dimension 2.

2.3.2.2 Principe d'évolution d'un modèle

Un modèle de GBDMP évolue sur occurrence de deux types d'événements :

- *événement provoqué* : commutation entre deux modes d'une feuille l. Un tel événement survient systématiquement quand l n'est pas dans le mode dans lequel elle devrait être (ce qui se traduit formellement par : $X_l \notin \mathcal{X}_{M_l}^{pmc(l)}$). Quand plusieurs événements de ce type sont en concurrence pour une même feuille, l'un d'entre eux est choisi aléatoirement à l'aide de la fonction $f_{i\to j}^{pmc(l)}$ (i étant le mode actuel, et $j = M_l$ le mode sollicité). Il s'agit non seulement des événements de commutation normale de composants, mais aussi des événements de défaillance à la commutation. L'occurrence d'un tel événement correspond au franchissement d'une transition en pointillé d'un PMC (cf. Figure 2.11).

- *événement spontané* : un tel événement ne peut survenir que si aucun événement provoqué ne peut plus - et donc ne doit plus - survenir (ce qui se traduit formellement par : $\forall l \in L, X_l \in \mathcal{X}_{M_l}^{pmc(l)}$). Il s'agit notamment des événements de défaillance (hors défaillance à la commutation) ou de réparation d'un composant, mais aussi d'événements externes de type changement de phase, intervention d'un opérateur... L'occurrence d'un tel événement correspond au franchissement d'une transition en trait plein d'un PMC (cf. Figure 2.11).

Les états d'un modèle GBDMP sont divisés en deux catégories [13] : les états *stables* et les états *instables*. Un état stable est un état depuis lequel le modèle ne peut évoluer que sur occurrence d'un événement *spontané*. Pour les autres états, seuls des événements *provoqués* peuvent survenir. En d'autres termes, un état est stable si la valeur des statuts de chaque feuille est en accord avec l'état du PMC qui lui est associé. Ainsi, la condition de stabilité d'un état se traduit formellement par la formule 2.5 (la première ligne vérifie que le PMC est dans le bon mode, et la seconde vérifie que le statut de *défaillance* a bien été actualisé).

$$\forall l \in L, \begin{cases} X_l \in \mathcal{X}_{M_l}^{pmc(l)} \\ \left(F_l \wedge X_l \in \mathcal{X}_F^{pmc(l)}\right) \vee \left(\neg F_l \wedge X_l \notin \mathcal{X}_F^{pmc(l)}\right) \end{cases} \tag{2.5}$$

13. On rappelle que les états du modèle sont déterminés par les variables d'état des feuilles et des commutateurs.

Pour initialiser un modèle GBDMP, il faut effectuer les actions suivantes dans l'ordre :

1. Initialiser la variable d'état de chaque commutateur s. Comme les machines de Moore sont déterministes, cette première étape est déterministe : $U_s = q_0^{str(s)}$.

2. Puis les variables R_n et M_n peuvent être initialisées pour chaque nœud n (en utilisant les équations 2.3 et 2.4).

3. La variable d'état de chaque feuille l peut alors être initialisée : la valeur de X_l est choisie aléatoirement dans $\mathcal{X}_{M_l}^{pmc(l)}$ à l'aide de la distribution de probabilité initiale $p0_{M_l}^{pmc(l)}$. Cette étape n'est donc pas déterministe dans le cas général.

4. Enfin, les statuts de défaillance des nœuds peuvent être initialisés à leur tour (en utilisant les équations 2.1 et 2.2).

Partant d'un état stable, l'occurrence d'un événement spontané a pour effet direct de modifier l'état X_l d'une feuille l. S'ensuit l'enchainement d'actions suivant avant de retrouver un état stable :

1. Toutes les autres variables doivent être actualisées, en utilisant les formules 2.1, 2.2, 2.3, 2.4 et les fonctions de transition des machines de Moore.

2. Occurrence des événements provoqués pour commuter les feuilles dans le bon mode. Les autres variables ne doivent pas être réactualisées tant que toutes les feuilles ne sont pas dans le bon mode, i.e. tant que la première ligne de la condition de stabilité n'est pas vérifiée (cf. formule 2.5).

3. Éventuellement, l'étape précédente a eu pour effet de modifier le statut de *défaillance* d'une feuille (cas de la défaillance à la commutation). Dans ce cas, la seconde ligne de la condition de stabilité n'est pas vérifiée (cf. formule 2.5) et il faut revenir à la première étape.

La Figure 2.14 illustre l'application de cette dynamique pour l'exemple didactique (cf. 2.11). Partant de l'état initial (dans ce cas déterministe), la feuille $C1$ défaille (franchissement de la transition entre les états W et F_2 de $pmc(C2)$), ce qui a pour conséquence de faire évoluer la machine de Moore associée à $S1$ dans l'état q_1. Cette évolution implique que $C1$ doit être commutée dans le mode 0, et $C3$ dans le mode 1 : ce qui explique les occurrences des 2 événements provoqués qui s'ensuivent. Dans ce cas, l'étape 3 ne

procède pas à un renvoi à l'étape 1, car il n'y a pas eu de défaillance à la sollicitation. Sur cette figure, les cadres en traits pleins correspondent à des états stables du système, tandis que les cadres en pointillés correspondent à des états instables.

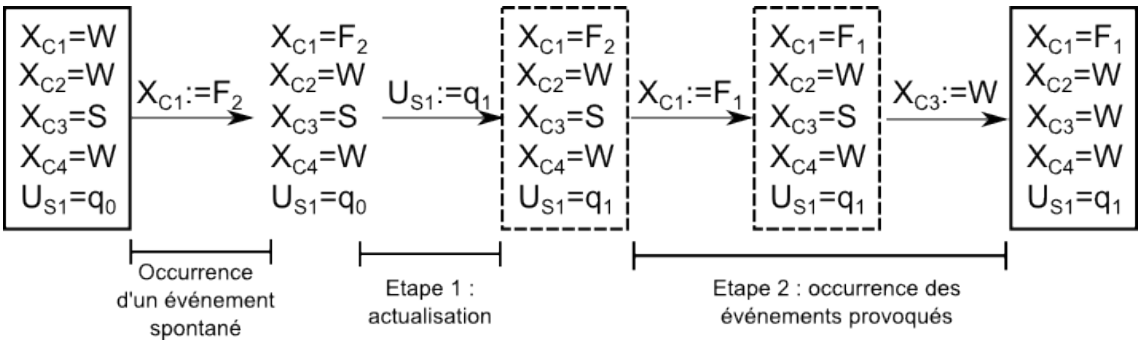

Figure 2.14 – Exemple d'évolution d'un modèle GBDMP entre deux états stables

Cette dynamique induit une nouvelle contrainte de construction des modèles. En effet, pour garantir que le modèle ne boucle pas indéfiniment suite à la sortie d'un état stable, une solution est d'interdire les événements de réparation à la commutation (ce qu'exprime la règle 6). Dans ce cas, l'étape 2 ci-dessus consiste en un passage par un nombre d'états instables inférieur ou égal au nombre de feuilles du modèle (au pire : toutes les feuilles doivent être commutées) ; et l'étape 3 ne renvoie à l'étape 1 qu'en cas de défaillance à la commutation d'une feuille. Le nombre de ces renvois est donc également limité par le nombre de feuilles car plus aucune défaillance à la commutation ne peut se produire dès lors que tous les composants sont défaillants. Le respect de la règle 6 garantit donc la séparation de chaque couple d'états stables successifs dans l'espace d'état du modèle, par au plus $Card(L)^2$ états instables.

Règle 6. *Une feuille ne peut pas être réparée à la commutation :*
Pour chaque feuille $l \in L$ de dimension k,

$$\forall (i,j) \in [\![0, k-1]\!]^2 | i \neq j, \forall (x,y) \in \mathcal{X}_i^{pmc(l)} \times \mathcal{X}_j^{pmc(l)} :$$
$$x \in \mathcal{X}_F^{pmc(l)} \wedge y \notin \mathcal{X}_F^{pmc(l)} \Longrightarrow f_{i \to j}^{pmc(l)}(x,y) = 0$$

2.3.3 Algorithme de simulation à événements discrets d'un modèle GBDMP

Avant de proposer un algorithme pour calculer l'évolution d'un modèle GBDMP, il est nécessaire de procéder à une analyse des dépendances entre les variables. En effet, comme on peut le voir avec le graphe dessiné sur la Figure 2.15, les dépendances entre

les variables de l'exemple didactique sont nombreuses et complexes (on peut notamment identifier plusieurs boucles). Il convient donc d'être rigoureux dans l'ordre d'actualisation des variables. Les formules 2.2, 2.3 et 2.4 indiquent que ces dépendances sont liées à la structure des graphes $\mathcal{G}_F$ et $\mathcal{G}_S$. En effet, on comprend à la lecture de la Figure 2.15 que les statuts de *défaillance* et de *réquisition* doivent être actualisés dans le sens des feuilles vers les racines, tandis que les statuts d'*activité* doivent être actualisés dans le sens des racines vers les feuilles. De plus, les variables d'état des commutateurs et les statuts de *réquisition* doivent être actualisés dans le sens des entrées vers les sorties des commutateurs.

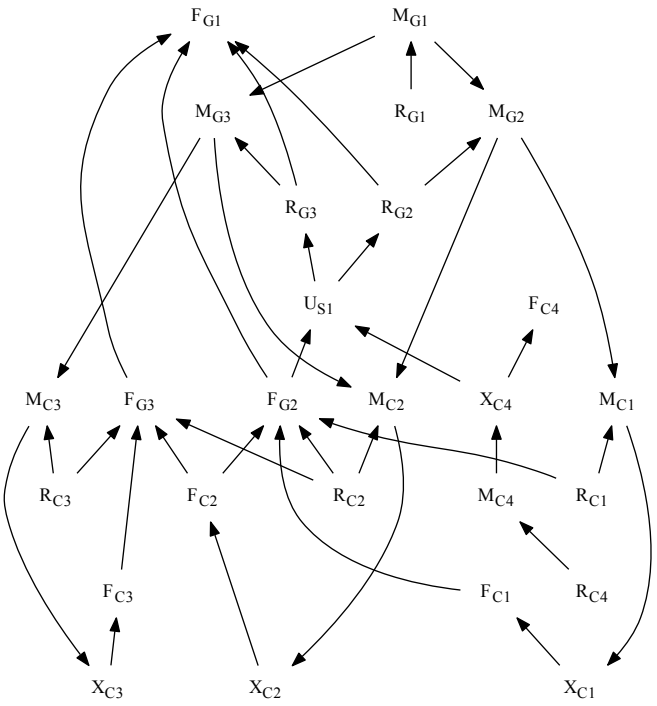

Figure 2.15 – Graphe des dépendances entre les variables de l'exemple didactique

Cette analyse des dépendances nous pousse à définir une dernière contrainte de construction du modèle, impactant les arcs entre les différents sommets du modèle (nœuds et commutateurs). Si la règle 7 n'est pas respectée, l'actualisation des variables ne sera pas possible.

Règle 7. *Il n'existe pas de circuit dans $\mathcal{G}$ contenant un arc de $\mathcal{G}_F$:*

$$\forall (x, y) \in N^2, x \succ_{\mathcal{G}_F} y \Longrightarrow \neg \left(x \rightleftarrows_{\mathcal{G}} y\right)$$

Pour illustrer ce point, nous prenons un contre-exemple très simple de modèle ne respectant pas cette règle (cf. Figure 2.16). Pour cet exemple, dans le graphe $\mathcal{G}$ (où les arcs en pointillés sont inversés [14]), il y a un circuit entre les sommets g, l et s, contenant un arc de $\mathcal{G}_F$ (l'arc entre g et l). Si l'on s'en réfère à la formule 2.2 et au principe d'un commutateur, ce circuit induit une boucle dans les dépendances entre les variables : $F_g \leftarrow R_l \leftarrow U_s \leftarrow F_g$. Cette boucle empêchera d'actualiser les variables, et donc de déterminer l'évolution du modèle.

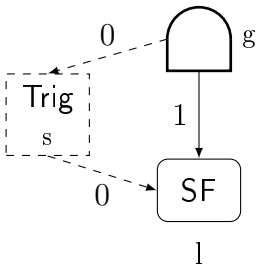

Figure 2.16 – Contre exemple de modèle GBDMP ne respectant pas la règle 7.

Afin d'organiser l'actualisation des variables, une fonction $Level : V \longrightarrow \mathbb{N}$ vérifiant la condition 2.6 est construite. Cette fonction permet de classer les sommets du modèle en niveaux de profondeur. Une telle classification n'est possible que si le modèle est bien formé, et en particulier, s'il respecte la règle 7. L'existence d'une fonction *Level* pour un modèle bien formé est prouvée dans l'annexe B.

$$\forall (v_1, v_2) \in V^2 \begin{cases} (v_1, v_2) \in E_F & \Longrightarrow \quad Level(v_2) > Level(v_1) \\ (v_1, v_2) \in E_S & \Longrightarrow \quad Level(v_1) \geq Level(v_2) \end{cases} \tag{2.6}$$

Ainsi, pour l'exemple didactique (cf. Figure 2.11), on peut vérifier que la fonction *Level* donnant les valeurs suivantes : $Level(G1) = 0$, $Level(G2) = Level(G3) = Level(C4) = Level(S1) = 1$ et $Level(C1) = Level(C2) = Level(C3) = 2$, vérifie bien la condition 2.6.

La fonction *Level* permet de déterminer un ordre d'actualisation des variables (on rappelle que cette opération se base sur une distribution de valeurs aux variables d'état des feuilles) :

- D'après la formule 2.1, $\forall l \in L, F_l$ ne dépend que de X_l. Il n'y a donc aucune contrainte d'antécédence entre les variables à actualiser.

14. On rappelle que $\mathcal{G} = \mathcal{G}_F \cup \overline{\mathcal{G}_S}$

- D'après la formule 2.2, $\forall(g,n) \in E_F, F_g$ dépend de F_n et R_n. Or, $(g,n) \in E_F \Rightarrow Level(n) > Level(g)$. On peut donc garantir le respect de ces contraintes en imposant le fait que le statut de *défaillance* d'un nœud n'est actualisé qu'après les statuts de *défaillance* et de *réquisition* des nœuds de niveau strictement supérieur.

- D'après le principe des commutateurs décrit dans la sous-section 2.3.2.1, $\forall(n,s) \in E_S, U_s$ dépend de F_n ou X_n. Or, $(n,s) \in E_S \Rightarrow Level(n) \geq Level(s)$. On peut donc garantir le respect de ces contraintes en imposant le fait que la variable d'état d'un commutateur n'est actualisée qu'après les statuts de *défaillance* des nœuds de niveau supérieur ou égal.

- D'après la formule 2.3, $\forall(s,n) \in E_S, R_n$ dépend de U_s. Or, $(s,n) \in E_S \Rightarrow Level(s) \geq Level(n)$. On peut donc garantir le respect de ces contraintes en imposant le fait que le statut de *réquisition* d'un nœud n'est actualisé qu'après les variables d'état des commutateurs de niveau supérieur ou égal.

- D'après la formule 2.4, $\forall n \in N | d^-_{\mathcal{G}_F}(n) = 0, M_n$ ne dépend que de R_n. On peut donc garantir le respect de ces contraintes en imposant le fait que les statuts d'*activité* ne sont actualisés qu'après les statuts de *réquisition*.

- D'après la formule 2.4, $\forall(g,n) \in E_F, M_n$ dépend de M_g et R_n. Or, $(g,n) \in E_F \Rightarrow Level(g) < Level(n)$. On peut donc garantir le respect de ces contraintes en imposant le fait que le statut d'*activité* d'un nœud n'est actualisé qu'après les statuts d'*activité* des nœuds de niveau strictement inférieur.

Cet ordre d'actualisation des variables est respecté dans l'algorithme 1 (cf. lignes 6-13 and 22-31). L'algorithme calcule une évolution possible d'un modèle GBDMP étant donnée une séquence d'événements spontanés. Dans le cas général, cette évolution n'est pas déterministe. En effet, les distributions de probabilités initiales et les fonctions de transfert des PMC induisent un comportement du modèle *a priori* aléatoire.

Afin d'illustrer dans les grandes lignes l'utilisation de cet algorithme[15], nous allons à présent calculer l'évolution du modèle de l'exemple didactique (cf. 2.11), pour une séquence de deux événements : défaillance de la feuille $C1$ (transition entre les états W et F_2), puis réparation de la feuille $C1$ (transition entre les états F_1 et S). Sur la Figure 2.17, seuls les états stables sont représentés.

15. L'illustration dans le détail occuperait trop de place.

Algorithme 1 Simulation à événements discrets d'un modèle GBDMP

Inputs:
- $< V, E, \kappa, \upsilon, str, pmc >$ un modèle GBDMP bien formé (cf. Définitions 2.1, 2.2, 2.3 et règles 1, 2, 3, 4, 5, 6 et 7).
- $\sigma = [e_1, ..., e_k]$ une séquence d'événements *spontanés*.

Outputs: Une évolution possible du modèle GBDMP.

1: *// Initialisation :*
2: $lev_{max} := \max_{v \in V}(Level(v))$
3: $\forall n \in N : F_n := False$
4: $\forall s \in S : U_s := q_0^{str(s)}$
5: $lev := lev_{max}$
6: **while** $lev \geq 0$ **do**
7: $\quad \forall n \in N | Level(n) = l$: initialiser R_n
8: $\quad lev := lev - 1$
9: **end while**
10: **while** $lev \leq lev_{max}$ **do**
11: $\quad \forall n \in N | Level(n) = l$: initialiser M_n
12: $\quad lev := lev + 1$
13: **end while**
14: $\forall l \in L$: initialiser X_l à l'aide de $p0_{M_l}^{pmc(l)}$
15: *// Boucle principale :*
16: $i := 0$
17: **while** $i \leq k$ **do**
18: $\quad$ **if** $i \neq 0$ **then**
19: $\quad\quad$ occurrence de e_i *// occurrence d'un événement spontané (modification de la valeur X_l pour la feuille l concernée)*
20: $\quad$ **end if**
21: $\quad isStable := False$
22: $\quad$ **while** $isStable = False$ **do**
23: $\quad\quad$ **while** $lev \geq 0$ **do**
24: $\quad\quad\quad \forall n \in N | Level(n) = l$: actualiser F_n
25: $\quad\quad\quad \forall s \in S | Level(s) = l$: actualiser U_s
26: $\quad\quad\quad \forall n \in N | Level(n) = l$: actualiser R_n
27: $\quad\quad\quad lev := lev - 1$
28: $\quad\quad$ **end while**
29: $\quad\quad$ **while** $lev \leq lev_{max}$ **do**
30: $\quad\quad\quad \forall n \in N | Level(n) = l$: actualiser M_n
31: $\quad\quad\quad lev := lev + 1$
32: $\quad\quad$ **end while**
33: $\quad\quad \forall l \in L | X_l \notin \mathcal{X}_{M_l}^{pmc(l)}$: actualiser X_l *// occurrence des événements provoqués*
34: $\quad\quad$ **if** $\forall l \in L | \left(F_l \wedge X_l \in \mathcal{X}_F^{pmc(l)}\right) \vee \left(\neg F_l \wedge X_l \notin \mathcal{X}_F^{pmc(l)}\right)$ **then**
35: $\quad\quad\quad isStable := True$
36: $\quad\quad$ **end if**
37: $\quad$ **end while**
38: $\quad i := i + 1$
39: **end while**

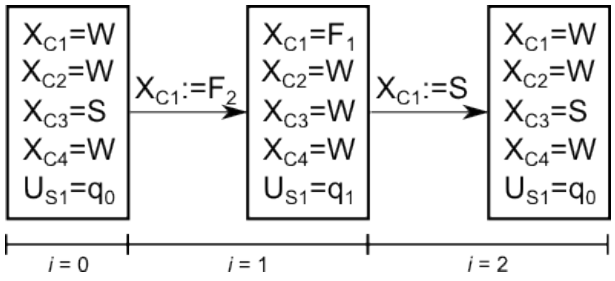

Figure 2.17 – Evolution simple d'un modèle GBDMP, déterminée par l'algorithme 1

Pour ce cas particulier, l'évolution est déterministe :

- *Initialisation*
  - les lignes 1 à 14 permettent de calculer l'état initial (cf. le premier cadre sur la Figure 2.14).
- $i := 0$
  - un premier passage des lignes 22 à 37 permettent de vérifier que le premier état est stable.
- $i := 1$
  - la ligne 19 applique l'effet de l'occurrence du premier événement : $X_{C1} := F_2$.
  - les lignes 23 à 32 permettent d'actualiser les statuts de *défaillance*, de *réquisition* et d'*activité* des nœuds, ainsi que la variable d'état de $S1$.
  - la ligne 33 applique les effets de l'occurrence des événements provoqués : $X_{C1} := F_1$ et $X_{C3} := W$.
  - le modèle est stable, donc on retourne à la ligne 17.
- $i := 2$
  - la ligne 19 applique les effets de l'occurrence du second événement : $X_{C1} := S$.
  - les lignes 23 à 32 permettent d'actualiser les statuts de *défaillance*, de *réquisition* et d'*activité* des nœuds, ainsi que la variable d'état de $S1$.
  - la ligne 33 applique les effets de l'occurrence des événements provoqués : $X_{C1} := W$ et $X_{C3} := S$.
  - le modèle est stable, donc l'algorithme termine.

2.4 Construction de modèles GBDMP

Cette section propose une réflexion méthodologique pour aider à la construction d'un modèle GBDMP, en partant d'une description souvent informelle du système considéré. Cette méthode est éprouvée sur trois exemples reprenant ou étendant les cas étudiés dans la section 2.2.

2.4.1 Méthodologie générale

La méthode proposée se déroule en trois étapes. Dans un premier temps, l'arbre de défaillance du système est construit en se basant sur sa décomposition fonctionnelle. Puis, les comportements élémentaires constituant la dynamique du modèle sont extraits d'une analyse préliminaire du fonctionnement et du dysfonctionnement des composants. Enfin, les logiques de contrôle sont spécifiées, en s'appuyant sur les stratégies de reconfiguration et leurs conditions d'application.

2.4.1.1 Construction de l'arbre de défaillance

Le formalisme GBDMP est conçu pour conserver les qualités de description structurelle des arbres de défaillance. La construction d'un modèle de GBDMP commence donc par la démarche déductive classique d'élaboration d'un arbre de défaillance ([Vesely et al., 1981]). Un système est défaillant lorsqu'il est inapte à réaliser au moins une de ses fonctions alors que celle-ci est requise. Les fonctions principales du système que l'on souhaite considérer sont donc les racines de l'arbre et la décomposition de celles-ci en sous-fonctions permet de construire l'arborescence[16]. Enfin, les composants jouant un rôle dans la réalisation de chaque sous-fonction seront modélisés par les feuilles de l'arbre. A cette étape, il est important de respecter la règle de construction 1 (une porte g doit avoir au moins $\kappa(g)$ fils).

Ainsi, il est possible de réaliser cette étape à partir d'une description fonctionnelle et matérielle du système. Si cette description est fournie par un modèle formel (de type Matlab/Simulink, UML/SysML, AADL...), l'automatisation de cette étape est envisageable. Ce sujet est largement traité dans la littérature ([Papadopoulos and Maruhn, 2001], [Xiang et al., 2011], [Joshi et al., 2007]...).

16. Cette décomposition doit prendre en compte les redondances fonctionnelles, qui se traduiront par des portes ET.

Dans le cas des systèmes reconfigurables, les fonctions concernent une partie du système dans une certaine configuration. Aussi, chaque branche de l'arbre de défaillance modélise l'état dysfonctionnel d'une partie du système dans une configuration particulière. La sous-section suivante explique comment paramétrer les branches de l'arbre, en étiquetant les arcs, de manière à caractériser la configuration qu'elle représente. De plus, elle montre comment décrire les comportements élémentaires associés aux feuilles de l'arbre.

2.4.1.2 Description des comportements élémentaires par des PMC

Afin de construire les PMC, il est nécessaire de procéder à une analyse du fonctionnement et du dysfonctionnement des composants suffisamment fine pour atteindre les objectifs de modélisation attendus. Ainsi, par exemple, si l'objectif est de confronter les performances, en terme de disponibilité, de plusieurs stratégies de redondance entre plusieurs composants multi-états, il sera nécessaire de connaître précisément le comportement dysfonctionnel des composants dans chacun de leur mode d'opération ainsi que lors de la commutation entre eux. Ce type de connaissances pourra être fourni par une analyse préliminaire des risques (AMDEC, REX...), et des documents techniques sur les modes d'opération des composants.

En outre, c'est à cette étape que l'on doit identifier quels sont les événements *spontanés* et *provoqués* à prendre en compte.

- Les événements *spontanés* sont les perturbations subies par le système qui ont un effet (direct ou indirect) sur son état dysfonctionnel. Les défaillances (hors défaillance à la commutation) et réparations de composants sont des exemples d'événements spontanés ayant un impact direct sur l'état dysfonctionnel du système. Les "chocs" provenant de l'environnement du système peuvent également entrer dans cette catégorie (incendie, explosion, inondation...). Un changement de phase, ou une action d'un opérateur sont des exemples d'événements spontanés n'ayant pas nécessairement d'effet direct sur l'état dysfonctionnel du système. Afin de garantir l'homogénéité des chaînes de Markov, dans le cadre de la modélisation GBDMP, les fréquences d'occurrence de ces événements sont associées à des lois de probabilité exponentielles.

- Les événements *provoqués* sont les actions entreprises par le système en réponse aux

événements *spontanés*, en vue d'obtenir un effet *a priori* bénéfique sur le fonctionnement du système (mais pas systématiquement). Étant donné que le système ne peut agir que par l'intermédiaire de ses composants, un tel événement se matérialise toujours par le changement de mode d'opération d'un ou plusieurs composants. Par exemple, l'action consistant à activer un composant de secours pour remplacer un composant défaillant est un événement *provoqué* ayant un effet bénéfique sur le fonctionnement du système. L'action de désactiver le composant défaillant pour faciliter sa réparation en est un autre. Une défaillance à la sollicitation (ou plus généralement : défaillance à la commutation) est une défaillance corrélée au changement de mode d'un composant. Ce type d'événement est donc homogène à un événement *provoqué*, bien que son effet, *a priori* bénéfique (commutation d'un composant), se révèle *a posteriori* néfaste (défaillance d'un composant).

Une fois que ces données sont réunies (modes d'opération, modes de défaillance, taux de probabilité, modalités de commutation...), il ne reste plus qu'à construire les PMC. Un PMC doit être construit pour chaque entité (composant ou processus externe au système) génératrice d'événements *spontanés* et que le système peut éventuellement contrôler par l'intermédiaire d'événements *provoqués*. Pour une raison de lisibilité et de cohérence des modèles, et puisque le formalisme le permet, il est conseillé de ne pas disperser le comportement d'un composant (brique élémentaire du système) dans plusieurs PMC (brique élémentaire du modèle).

Lorsque le nombre k de modes d'opération d'un composant devient grand, il peut paraître fastidieux d'avoir à définir les $k(k-1)$ fonctions de transfert entre les k chaînes de Markov (prises 2 à 2). Il semble cependant raisonnable de postuler que, dans la grande majorité des cas, les différentes facettes du comportement dysfonctionnel d'un composant, au travers de ses modes d'opérations, sont basées sur la même "structure". C'est à dire que le nombre d'états des chaînes de Markov est souvent le même, et que le schéma de transfert entre ces états varie peu d'une chaîne à l'autre. Ce sont surtout les valeurs des probabilités et des taux de probabilité qui changent, ce qui diminue grandement la difficulté de définir ces fonctions de transfert.

En outre, la construction des PMC peut être automatisée, si les données requises sont organisées selon un format préétabli pertinent. Ainsi, la publication [Piriou et al.,

2014a] montre qu'il est possible de construire un processus de Markov [17] de manière systématique à partir d'un jeu de données organisé conformément au méta-modèle reporté sur la Figure 2.18.

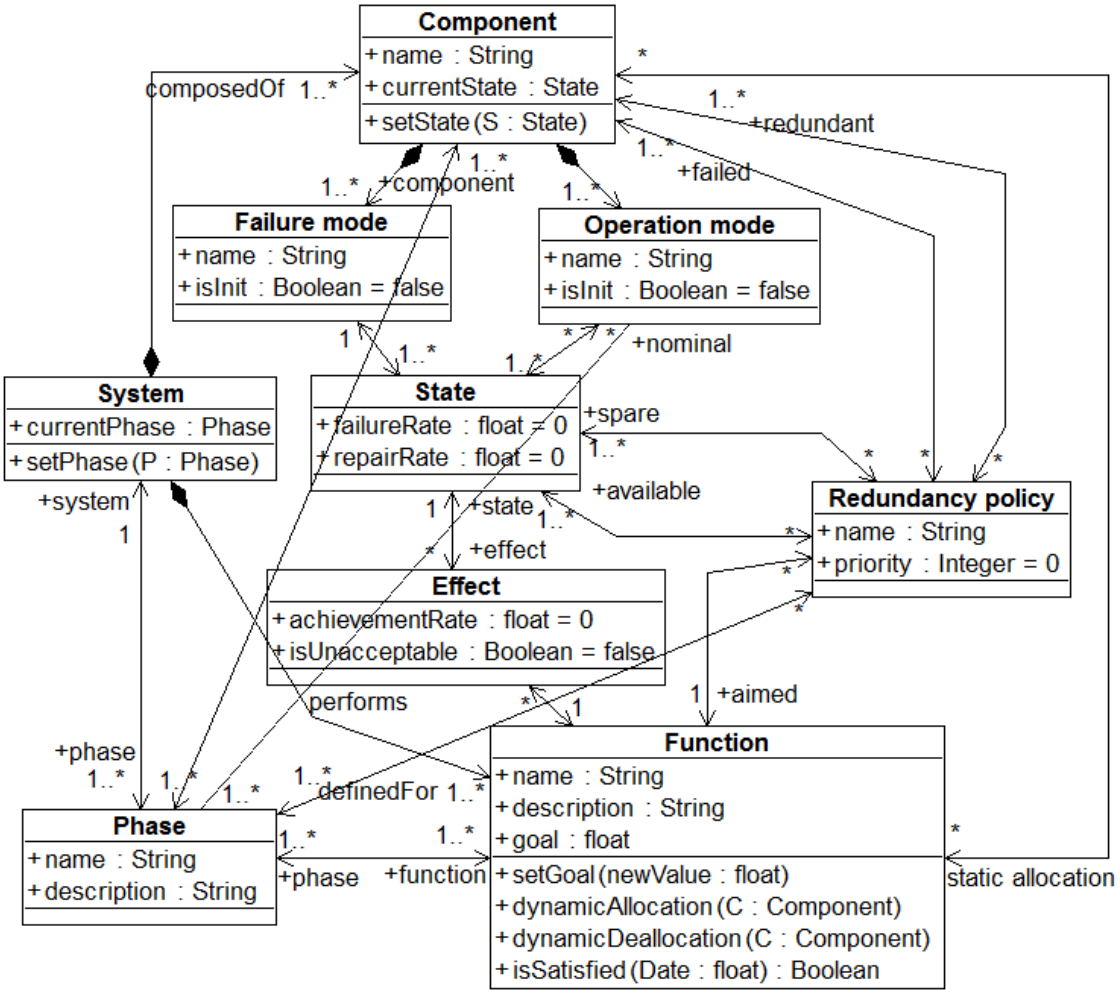

Figure 2.18 – Méta-modèle de données issues d'une analyse préliminaire du fonctionnement et des risques de dysfonctionnement d'un système (extrait de [Piriou et al., 2014a]).

Enfin, les PMC sont associés à des feuilles de l'arbre de défaillance. On a vu dans la sous-section précédente qu'une branche de l'arbre représente une partie du système dans une configuration particulière. Une configuration détermine dans quel mode d'opération doit être chaque composant de cette partie du système. Cette information est traduite dans l'arbre par les statuts d'*activité* des nœuds de la branche en question, selon l'interprétation décrite ci-après. Lorsqu'un nœud n'est pas requis (statut de *réquisition* égal à 0), il est inactif (statut d'*activité* égal à 0). Dans le cas contraire, il faut préciser dans

17. Le modèle à base de chaînes de Markov proposé dans cette publication est un 3-PMC, bien qu'il ne soit pas explicitement défini comme tel.

quel mode celui-ci est actif, en multipliant le statut de *réquisition* par un facteur entier. L'étiquette d'un arc est un moyen d'influer sur ces facteurs, et ce, pour tous les nœuds de la branche issue de l'arc. Ainsi, si un arc a pour étiquette la valeur 2, tous les statuts d'*activité* des nœuds de la branche issue de cet arc (et donc en particulier, les feuilles) seront multipliés par 2. C'est donc en définissant ces valeurs que l'on spécifie dans quel mode d'opération un PMC doit être commuté, lorsqu'il est sollicité par une branche particulière. Il apparait donc clairement que le non respect de la règle de construction 5 rendrait le modèle impossible à interpréter. En effet, il risquerait d'évoluer dans un état où un PMC associé à l'une de ses feuilles est sensé être commuté dans un mode qui n'a pas été défini.

Ainsi, pour l'exemple proposé sur la Figure 2.19, les nœuds $N1$, $N2$ et $N3$ représentent trois configurations différentes dans lesquels les nœuds $N4$, $N5$ et $N6$ sont requis dans les modes 1, 2 et 3. Afin de déterminer quelle configuration doit être active en fonction de l'évolution du modèle, les nœuds $N1$, $N2$ et $N3$ devront être les nœuds de sortie d'un commutateur décrivant la stratégie de reconfiguration (via sa machine de Moore associée).

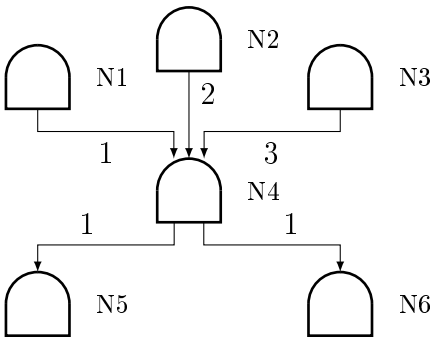

Figure 2.19 – Extrait de modèle GBDMP illustrant la propagation des facteurs permettant de déterminer le mode d'opération des PMC

La sous-section suivante explique comment construire les machines de Moore pour contrôler les statuts de *réquisition* des nœuds.

2.4.1.3 Spécification des logiques de commutation par des machines de Moore

Les commutateurs sont utilisés pour retranscrire dans le modèle des mécanismes de commutation. Ils réagissent à une évolution de leurs variables d'entrée (statuts de *dé-*

faillance ou variables d'état de PMC), et provoquent une évolution de leurs variables de sortie (statuts de *réquisition*). Comme on l'a vu dans la section précédente, l'évolution des variables d'entrée est une conséquence de l'occurrence d'un événement *spontané*, tandis que l'évolution des variables de sortie est une cause de l'occurrence d'un événement *provoqué*. La définition des machines de Moore associées aux commutateurs permet de spécifier comment les variables de sortie doivent évoluer en fonction de l'évolution des variables d'entrée.

La construction de ces machines consiste donc à formaliser des stratégies de reconfiguration, souvent décrites de manière informelle. Comme le mentionne la sous-section 1.2.3, le succès d'une telle stratégie n'est pas garanti. Aussi, dans le cas où les conditions d'application sont connues, elles doivent être prises en compte dans la formalisation de la stratégie. Ainsi par exemple, si un composant de contrôle est responsable de l'application d'une stratégie, et que l'effet de sa défaillance sur le comportement induit par la stratégie est connu, cet effet doit être pris en compte pour construire la machine de Moore.

Cette étape est délicate est requiert de procéder avec grande rigueur. Dans un premier temps, il convient d'identifier les variables d'entrée et de sortie qui interviennent dans la description de la stratégie (et de ses conditions d'application). Le tableau 2.3 associe des extraits de phrases types d'une telle description, avec la conclusion qui doit en être tirée en terme de variables à considérer.

Ce travail permet de construire les arcs reliant les commutateurs à leurs nœuds d'entrée et de sortie, en veillant à respecter les règles de construction 2, 3 et 7. Le respect de la règle 2 garantit qu'aucun statut de *réquisition* n'est contrôlé par deux stratégies contradictoires. Les alphabets d'entrée et de sortie des machines de Moore peuvent également être définis à ce stade, en respectant la règle de construction 4.

Chaque élément de l'alphabet d'entrée (resp. de sortie) correspond à une assignation de valeur aux variables d'entrée (resp. de sortie). Grâce à la fonction de sortie de la machine de Moore, l'espace d'état est ainsi exactement décrit par des assignations de valeurs aux variables de sortie. La taille de cet espace est donc bornée par le nombre des assignations possibles. La description de la stratégie de reconfiguration permet de déterminer quelle doit être l'assignation de valeur initiale aux variables de sortie, et donc quel est l'état initial de la machine de Moore. Enfin, il ne reste plus qu'à construire la fonction de transition, en respectant scrupuleusement la description de la stratégie.

extraits de phrase type	variables d'entrée à considérer
Si le sous-système A est défaillant (dans la configuration Z)...	La porte représentant A (dans la configuration Z) est un nœud d'entrée du commutateur. La variable d'entrée à considérer est son statut de *défaillance*.
Si le composant $C1$ est défaillant...	La feuille représentant $C1$ est un nœud d'entrée du commutateur. La variable d'entrée à considérer est son statut de *défaillance*.
Si le composant $C1$ est dans le mode de défaillance $M1$...	La feuille représentant $C2$ est un nœud d'entrée du commutateur. La variable d'entrée à considérer est l'état du PMC associé.
Si l'événement $E1$ survient...	La feuille génératrice de l'événement $E1$ est un nœud d'entrée du commutateur. La variable d'entrée à considérer est l'état du PMC associé.
extraits de phrase type	**variables de sortie à considérer**
...solliciter le sous-système B (dans la configuration Y).	La porte représentant B (dans la configuration Y) est un nœud de sortie du commutateur. La variable de sortie à considérer est son statut de réquisition.
...solliciter le composant $C2$.	La feuille représentant $C3$ est un nœud de sortie du commutateur. La variable de sortie à considérer est son statut de réquisition.
extrait de phrase type	**variables d'entrée et de sortie à considérer**
Si l'un des composants $C1$ ou $C2$ est défaillant, désactiver le composant défaillant.	Dans ce cas, l'information fournie par le statut de *défaillance* d'une porte OU entre $C1$ et $C2$ est insuffisante. Pour appliquer la seconde partie de la règle, il est nécessaire de connaître les statuts de *défaillance* des deux feuilles représentant ces composants, qui seront donc à la fois nœuds d'entrée et de sortie du commutateur.

Tableau 2.3 – Exemples d'interprétation de phrases type d'un corpus de texte décrivant une stratégie de reconfiguration.

Des exemples d'une telle formalisation sont proposés dans la sous-section suivante.

2.4.2 Exemples de modélisation GBDMP

Dans la suite, trois petits exemples de systèmes caractéristiques sont modélisés selon le formalisme GBDMP. Le premier exemple introduit deux redondances passives régies par deux stratégies de reconfiguration différentes. De plus, le succès de ces stratégies n'est pas garanti car les composants censés assurer leur application sont faillibles. Les deux autres exemples reprennent les cas traités dans la section 2.2 pour illustrer l'application de la modélisation de composants multi-états et de systèmes multi-phases.

2.4.2.1 Différentes stratégies de redondance appliquées par des composants de commutation faillibles

a) Description du cas

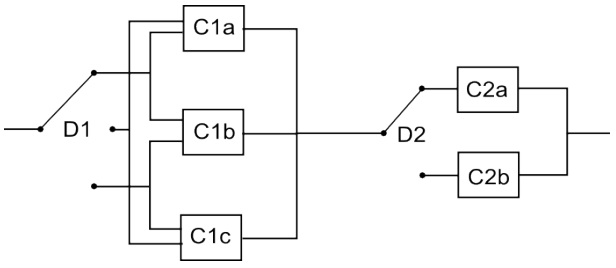

Figure 2.20 – Deux groupes de composants redondants, selon deux stratégies différentes

L'exemple schématisé par la Figure 2.20 introduit deux stratégies de redondance différentes, dont le succès n'est pas garanti. La nature des composants redondants n'est pas importante ici. La seule chose à savoir est qu'ils peuvent être activés et désactivés, et qu'ils sont susceptibles de défaillir et d'être réparés. Deux redondances passives sont appliquées par les composants de commutation $D1$ et $D2$, en se référant à deux stratégies différentes :

- stratégie appliquée par $D1$: initialement, $C1a$ et $C1b$ sont activés arbitrairement. Puis, si un des composants actifs est défaillant alors que les deux autres ne le sont pas, désactiver le composant défaillant, et activer le composant inactif.
- stratégie appliquée par $D2$: quel que soit l'état dysfonctionnel de $C2b$, si $C2a$ n'est pas défaillant, $C2a$ et $C2b$ doivent être respectivement actif et inactif et vice versa : si $C2a$ est défaillant, $C2a$ et $C2b$ doivent être respectivement inactif et actif.

Dans la première stratégie, on dit que la commutation est du type "au plus tard" car elle ne s'effectue que lorsqu'elle est absolument nécessaire, c'est à dire, lorsque le service n'est pas réalisé par la solution actuelle et qu'il le sera suite à la commutation sur une solution redondante. Comme on le justifiera dans le chapitre suivant, ce type de stratégie sera privilégié pour minimiser les commutations, et ainsi limiter les risques d'éventuelles défaillances à la sollicitation. Dans la seconde stratégie, on dit que la commutation est du type "au plus tôt" car elle s'effectue dès que la solution principale défaille / est réparée, sans prendre en compte l'aptitude de la solution de secours à réaliser le service ou non. Ce type de stratégie sera privilégié lorsque l'utilisation de la solution principale est

préférable à celle de la solution de secours (par exemple pour des raisons de performance ou de fiabilité). Il est intéressant de remarquer que la seconde stratégie ressemble à celle qui est implicitement traduite par une gâchette de BDMP. La différence réside dans le fait qu'elle contrôle non seulement l'activité de la solution de secours mais aussi celle de la solution principale, de telle sorte que les deux solutions ne soient jamais actives simultanément.

Par ailleurs, les composants responsables de l'application de ces stratégies ($D1$ et $D2$) peuvent défaillir selon deux modes de défaillance :

- *frozen* (figé) : plus aucune commutation n'est possible (taux de défaillance et de réparation : λ_f et μ_f).

- *bad contact* (faux contact) : aucun composant n'est sollicité (taux de défaillance et de réparation : λ_{bc} et μ_{bc}).

b) Construction de l'arbre de défaillance

La fonction décrite en Figure 2.20 est réalisée si au moins deux composants parmi $\{C1a, C1b, C1c\}$ et un composant parmi $\{C2a, C2b\}$ sont opérationnels. L'échec de cette fonction est modélisé (d'un point de vue statique) par l'arbre de défaillance représenté sur la Figure 2.21. Les composants $D1$ et $D2$ ne jouent pas un rôle direct dans la réalisation de la fonction cible. Ils ne sont utilisés que pour appliquer les redondances. Les feuilles représentant ces composants seront donc reliées aux autres nœuds, via les commutateurs qui seront introduits dans la suite pour modéliser ces redondances.

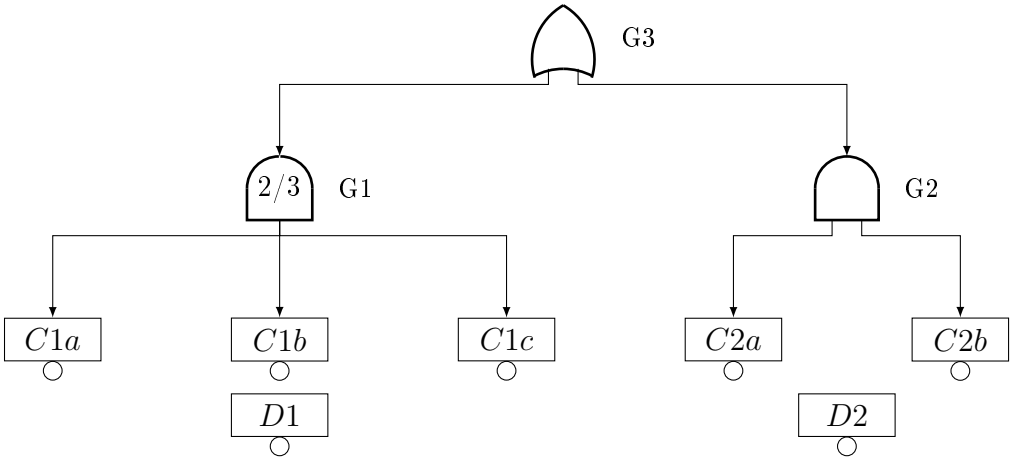

Figure 2.21 – Arbre de défaillance modélisant la structure du système décrite par la Figure 2.20

c) Modélisation des comportements élémentaires

Il convient à présent d'associer aux feuilles de cet arbre des PMC pour modéliser leur comportement. La description des composants $C1a$, $C1b$, $C1c$, $C2a$ et $C2b$ indique qu'ils peuvent être actif ou inactif et défaillant ou non défaillant. Comme leur comportement n'est pas plus détaillé, le 2-PMC équivalent au PMP de type SF convient pour le modéliser (cf. Figure 2.22). Étant donné que les composants ne peuvent être activés que dans un mode d'opération, tous les arcs de l'arbre de défaillance sont étiquetés par la valeur 1 [18]. Notons que cet étiquetage respecte la règle de construction 5.

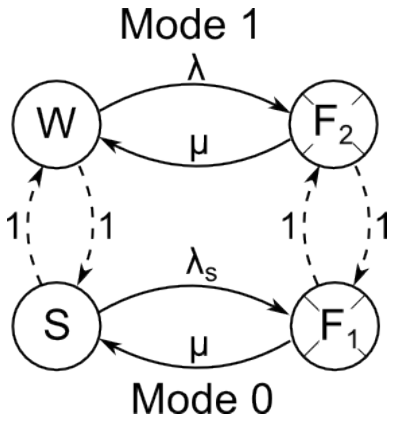

Figure 2.22 – PMC de type SF

En revanche, le comportement des composants $D1$ et $D2$ est décrit plus précisément. En effet, on sait qu'ils peuvent défaillir selon deux modes de défaillance, et peuvent être réparés. Cependant, ces composants ne semblent avoir qu'un unique mode d'opération. Leur comportement sera donc modélisé par le 1-PMC représenté sur la Figure 2.23. Initialement, ces composants sont non défaillants (i.e. la probabilité d'être dans l'état 0 initialement est 1). Dans la suite, ce PMC sera appelé Co (car il modélise un composant de **Co**ntrôle).

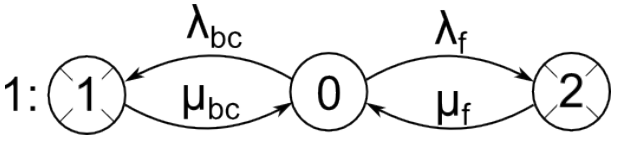

Figure 2.23 – PMC de type Co modélisant le comportement des composants $D1$ et $D2$

18. Par défaut, afin d'alléger la représentation graphique des modèles, la valeur associée à un arc dont l'étiquette n'est pas représentée est 1

d) Formalisation des stratégies de reconfiguration

La dernière étape consiste à ajouter deux commutateurs $S1$ et $S2$ au modèle pour représenter les mécanismes de commutation induits par les deux stratégies de redondance. La description de ces stratégies permet de déterminer une partie des variables d'entrée et de sortie :

- la description de la première stratégie indique que l'on cherche à contrôler l'activation des composants $C1a$, $C1b$ et $C1c$, en fonction de l'état dysfonctionnel de ces trois mêmes composants. Ainsi, les feuilles représentant ces composants sont à la fois nœud d'entrée et nœud de sortie du commutateur $S1$. Les variables d'entrée et sortie à considérer sont respectivement les statuts de *défaillance* (dans ce cas, cette information suffit) et les statuts de *réquisition* de ces nœuds.

- la description de la seconde stratégie indique que l'on cherche à contrôler l'activation des composants $C2a$ et $C2b$, en fonction de l'état dysfonctionnel de $C2a$ uniquement. Ainsi, la feuille représentant $C2a$ est à la fois nœud d'entrée et nœud de sortie du commutateur $S2$, tandis que la feuille représentant $C2b$ est uniquement nœud de sortie. Les variables d'entrée et de sortie à considérer sont respectivement le statut de *défaillance* de $C2a$ et les statuts de *réquisition* de $C2a$ et $C2b$.

En outre, les informations dont on dispose relatives aux effets des défaillances des composants $D1$ et $D2$ sur les mécanismes de commutation, conduisent à ajouter une entrée supplémentaire à ces commutateurs. En effet, la description des modes de défaillance de ces composants indique que leur état dysfonctionnel a une incidence sur les évolutions du commutateur. Ainsi, la feuille représentant $D1$ (resp. $D2$) doit être un nœud d'entrée de $S1$ (resp. $S2$). La variable d'entrée à considérer est l'état du PMC associé à cette feuille (dans ce cas, il est nécessaire de savoir dans quel mode de défaillance le composant a défailli). Cet exemple illustre un cas où le commutateur exploite une information précise contenue dans les processus de Markov.

Cette interprétation des descriptions des stratégies permet de relier les nœuds de l'arbre de défaillance aux commutateurs (en entrée et en sortie), et de définir les alphabets d'entrée et de sortie des machines de Moore associées. Finalement, il ne reste plus qu'à construire leurs fonctions de transition et de sortie pour retranscrire les stratégies décrites. Les deux machines de Moore issues de cette étape de modélisation sont représentées sur la Figure 2.24, et la vue *arbre de défaillance enrichi* du modèle GBDMP final

est reporté sur la Figure 2.25. Sur la Figure 2.24, le caractère "_" peut être remplacé par n'importe quelle valeur. De plus, on rappelle que les valeurs 0,1 et 2 prises par le premier terme des vecteurs d'entrée font référence aux états du PMC de type Co (c'est à dire aux modes de défaillance *OK*, *bad contact* et *frozen* d'un composant de commutation).

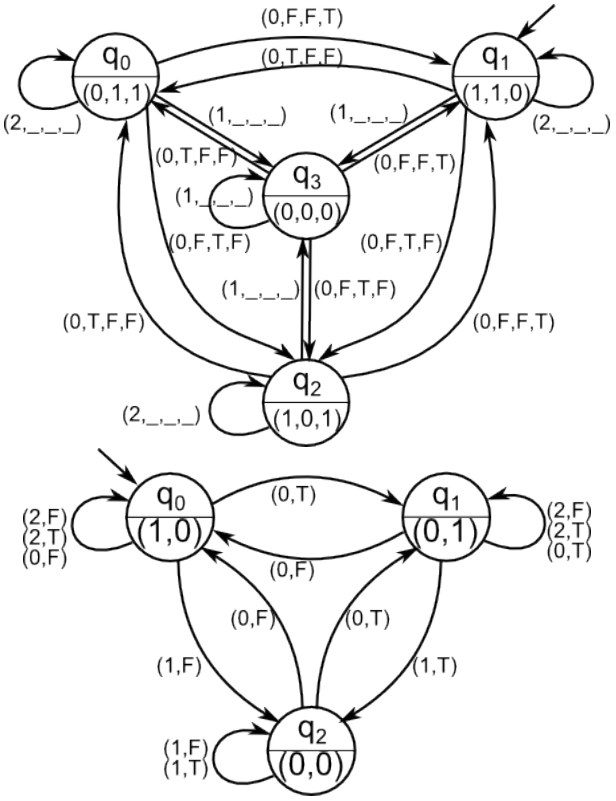

Figure 2.24 – Machines de Moore spécifiant deux stratégies de reconfiguration (en haut [20] : $M1 = str(S1)$; en bas : $M2 = str(S2)$).

On peut vérifier que l'ordre des nœuds d'entrée et de sortie des commutateurs (indiqué par les étiquettes des arcs dessinés en pointillé) coïncide avec la définition des alphabets d'entrée et de sortie des machines de Moore, afin de respecter les règles de construction 3 et 4. En effet, la Figure 2.24 permet de déduire que :

$$\begin{cases} \Sigma_I^{M1} = \{0,1,2\} \times \{False, True\} \times \{False, True\} \times \{False, True\} \\ \Sigma_O^{M1} = \{0,1\} \times \{0,1\} \times \{0,1\} \end{cases}$$

Or, on sait grâce à la définition du PMC associé à $D1$ (cf. Figure 2.23) et à la nature des statuts de *défaillance* et de *réquisition* que :

$$\begin{cases} \mathcal{D}_{X_{D1}} \times \mathcal{D}_{F_{C1a}} \times \mathcal{D}_{F_{C1b}} \times \mathcal{D}_{F_{C1c}} = \{0,1,2\} \times \{False, True\}^3 \\ \mathcal{D}_{R_{C1a}} \times \mathcal{D}_{R_{C1b}} \times \mathcal{D}_{R_{C1c}} = \{0,1\} \times \{0,1\} \times \{0,1\} \end{cases}$$

20. Pour alléger la figure, la machine de Moore $M1$ n'est pas représentée complètement : les self-loops doivent être étiquetées de telle sorte que la fonction de transition soit complète.

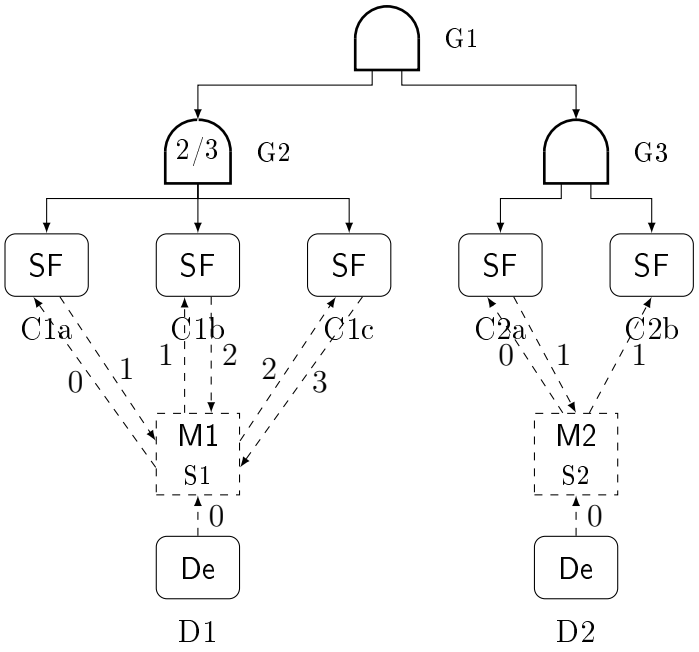

Figure 2.25 – Vue *arbre de défaillance enrichi* du modèle GBDMP pour le système faisant intervenir deux stratégies de redondance différentes appliquées par des composants faillibles

Compte tenu de ses entrées/sorties, le commutateur $S1$ est donc bien compatible avec la machine $M1$ (cf. Figure 2.25). La compatibilité entre le commutateur $S2$ et la machine $M2$ peut être observée de manière analogue.

e) Discussion

Pour tester ce modèle, le comportement attendu du système est maintenant confronté au comportement modélisé, à travers deux scénarios caractéristiques. Le premier scénario illustre la particularité de la stratégie de redondance "au plus tard" (qu'il est impossible de considérer avec le formalisme BDMP, comme on l'a souligné dans la sous-section 2.2.3.1) : $\mathbf{f_{C1c}} \rightarrow \mathbf{f_{C1a}} \rightarrow \mathbf{r_{C1c}} \rightarrow \mathbf{r_{C1a}}$. On rappelle que cette stratégie est motivée par une volonté de ne commuter les composants que lorsque leurs états dysfonctionnels le justifient réellement pour la réalisation de la fonction. Le comportement du système attendu, compte tenu de la description du cas est le suivant :

- Initialement, les composants $C1a$ et $C1b$ sont actifs, le composant $C1c$ est inactif ;
- suite à la défaillance (en mode passif) de $C1c$, aucune commutation n'est provoquée ;
- suite à la défaillance (en mode actif) de $C1a$, aucune commutation n'est provoquée. En effet, avec deux composants défaillants, la fonction est perdue quoi qu'il arrive,

donc une commutation n'est pas justifiée ;

- suite à la réparation de $C1c$, ce composant est activé et $C1a$ est désactivé. Cette commutation permet de réaliser à nouveau la fonction, elle est donc bien justifiée ;
- suite à la réparation de $C1a$, aucune commutation n'est provoquée. En effet, une nouvelle commutation serait injustifiée car la fonction est déjà correctement réalisée.

Les cellules du tableau 2.4 [21] sont remplies selon les résultats fournis par l'algorithme 1, appliqué au modèle proposé, pour cette séquence. Il confirme la conformité du comportement modélisé par rapport au comportement attendu pour ce scénario.

séquence	$0 \xrightarrow{f_{C1c}} 1 \xrightarrow{f_{C1a}} 2 \xrightarrow{r_{C1c}} 3 \xrightarrow{r_{C1a}} 4$				
X_{C1a}	W	W	F_2	F_1	S
X_{C1b}	W	W	W	W	W
X_{C1c}	S	F_1	F_1	W	W
F_{G1}	$False$	$False$	$True$	$False$	$False$

Tableau 2.4 – Comportement décrit par le modèle pour un scénario impliquant les composants $C1a$, $C1b$ et $C1c$

Le second scénario montre que l'application d'une stratégie de reconfiguration n'est pas toujours garantie, contrairement à ce qui est admis par le formalisme BDMP (comme on l'a souligné dans la sous-section 2.2.3.1) : $\mathbf{f_{C2a}} \rightarrow \mathbf{f_{D2}^{frozen}} \rightarrow \mathbf{r_{C2a}} \rightarrow \mathbf{f_{C2b}}$. Le comportement du système attendu, compte tenu de la description du cas est le suivant :

- Initialement, le composant $C2a$ est actif et le composant $C2b$ est inactif ;
- suite à la défaillance de $C2a$, ce composant est désactivé et $C2b$ est activé. Cette commutation a pu être appliquée car le composant $D2$ est opérationnel ;
- suite à la défaillance de $D2$ dans le mode $frozen$, aucune incidence directe n'est observée sur la réalisation de la fonction ;
- suite à la réparation de $C2a$, la commutation, prévue par la stratégie de redondance "au plus tôt", pour rétablir le service sur $C2a$ est impossible car le composant $D2$, en charge de cette commutation, est figé ;

21. On rappelle que les états S, W, F_1 et F_2 d'un PMP de type SF signifient respectivement "*Standby*", "*Working*", "*inactive Failure*" et "*active Failure*".

- suite à la défaillance de $C2b$, la fonction est perdue. Bien que le composant $C2a$ ne soit pas défaillant, le composant qui est sensé l'activer ($D2$) n'est pas en mesure de le faire.

Une fois encore, le tableau 2.5 confirme la conformité du comportement modélisé par rapport au comportement attendu pour ce scénario.

séquence	0	$\xrightarrow{f_{C2a}}$ 1	$\xrightarrow{f_{D2}^{frozen}}$ 2	$\xrightarrow{r_{C2a}}$ 3	$\xrightarrow{f_{C2b}}$ 4
X_{C2a}	W	F_1	F_1	S	S
X_{C2b}	S	W	W	W	F_2
X_{D2}	0	0	2	2	2
F_{G1}	$False$	$False$	$False$	$False$	$True$

Tableau 2.5 – Comportement décrit par le modèle pour un scénario impliquant les composants $C2a$, $C2b$ et $D2$

2.4.2.2 Le report de charge

Nous reprenons ici le cas du report de charge, dans la version exposée dans la sous-section 2.2.3.2. Le pompage peut être réalisée de trois manières :

- par P_0 ;
- par P_1 et P_2 en régime nominal ;
- par P_1 ou P_2 en régime surchargé.

Ce découpage fonctionnel permet de construire l'arbre de défaillance représenté sur la figure 2.26.

Comme P_0 n'a que deux modes d'opération (*off* et *on*), son comportement peut être modélisé simplement par le 2-PMC F. En revanche, pour les deux autres pompes, un troisième mode d'opération est à considérer ($Over$). Nous introduisons donc le 3-PMC Po (pour **Po**mpe), pour modéliser le comportement de ces pompes (cf. Figure 2.27). Ce PMC retranscrit bien le comportement de ce type de pompe, qui est décrit dans la sous-section 2.2.2.2.

Les feuilles P_1 et P_2 sont connectées à deux portes. La porte OU (resp. ET) se réfère aux deux pompes dans leur régime nominal (resp. surchargé), qui correspond au mode 1 (resp. 2). Ainsi, les arcs entre cette porte et les feuilles doivent être étiquetés par la valeur 1 (resp. 2). De cette manière, lorsque seule la porte OU est requise, elles sont

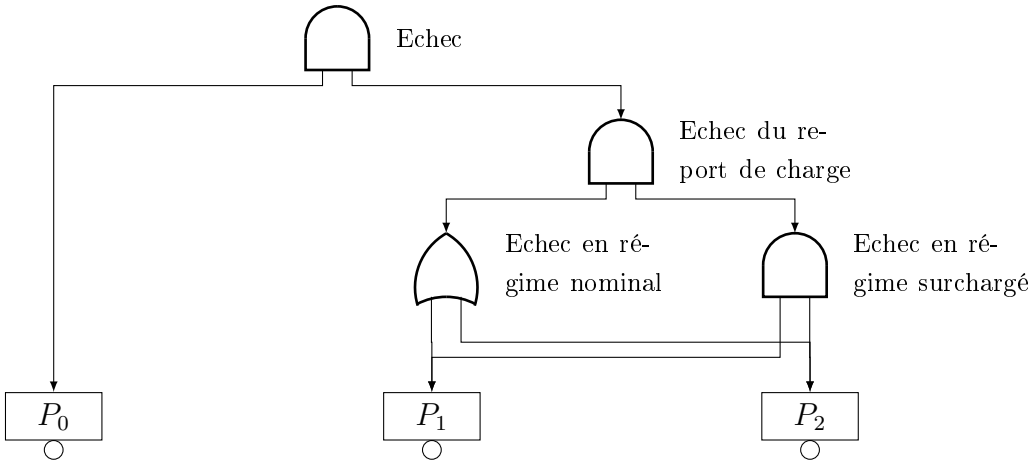

Figure 2.26 – Arbre de défaillance modélisant la structure du système pour le cas du report de charge

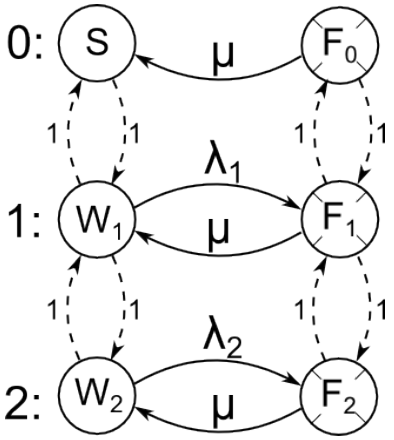

Figure 2.27 – PMC *Po* modélisant le comportement des pompes P_1 et P_2

sollicitées dans leur mode 1, tandis que lorsque la porte ET est également requise, elles sont sollicitées dans leur mode 2. En effet [22], si $R_{sur} = 1$ alors $M_{sur} = 1$ et $M_{P_1} = M_{P_2} = max(1.M_{nom}, 2.M_{sur}) = 2$.

Par ailleurs, la stratégie de redondance est relativement simple. On utilisera donc des commutateurs se comportant comme des gâchettes de BDMP (cf. Figure 2.13). Dans cet exemple, on fait de plus l'hypothèse que cette stratégie est toujours appliquée correctement. En effet, aucune information ne permet de déterminer par quel moyen cette stratégie est appliquée. Cependant, comme on l'a vu dans l'exemple précédent, il est possible de prendre en compte l'échec de la stratégie, au prix d'une modification de la machine de Moore qui la spécifie. La Figure 2.28 récapitule cette modélisation.

22. Les abréviations *nom* et *sur* désignent respectivement les portes "Echec en régime nominal" et "Echec en régime surchargé".

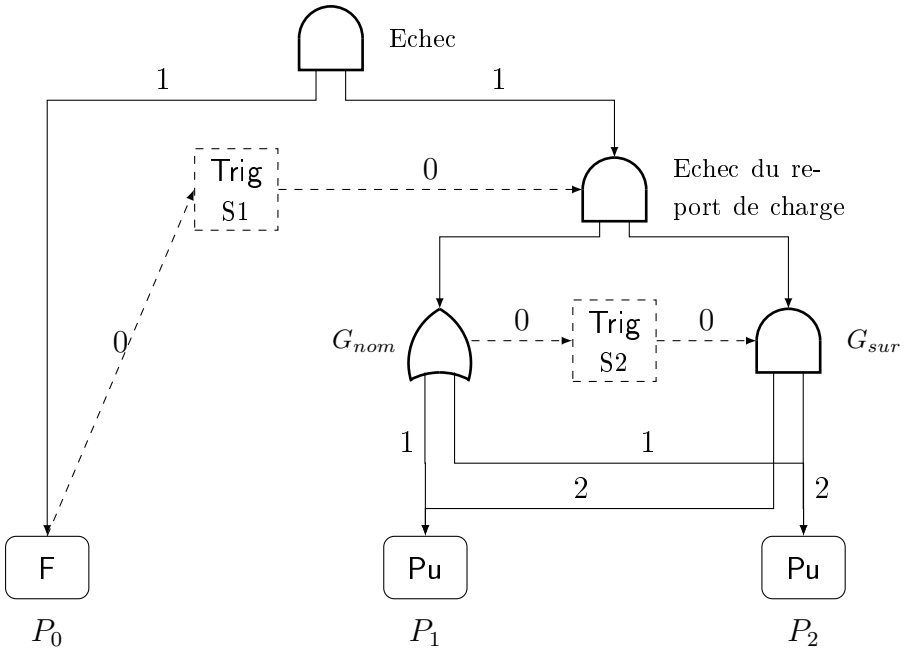

Figure 2.28 – Vue *arbre de défaillance enrichi* du modèle GBDMP des pompes multi-états

Afin d'illustrer le principe du report de charge, nous considérons le scénario dysfonctionnel : $\mathbf{f_{P_0}} \rightarrow \mathbf{f_{P_1}} \rightarrow \mathbf{f_{P_2}} \rightarrow \mathbf{r_{P_1}}$. Le comportement suivant est attendu :

- Initialement, la pompe P_0 est active, et les pompes P_1 et P_2 sont inactives ;
- suite à la défaillance de P_0, les pompes P_1 et P_2 sont activées dans leur régime nominal ;
- suite à la défaillance de P_1, les pompes P_1 et P_2 sont activées dans leur régime surchargé, P_2 réalise la fonction ;
- suite à la défaillance de P_2, la fonction est perdue, mais cela n'a pas d'incidence sur le mode d'opération des pompes ;
- suite à la réparation de P_1, cette pompe réalise la fonction (en régime surchargé).

Le tableau 2.6 montre que le modèle retranscrit bien le comportement attendu.

2.4.2.3 Les vannes bloquantes

Pour le dernier exemple, nous reprenons le cas des vannes bloquantes décrit dans la sous-section 2.2.3.3. L'arbre de défaillance doit représenter le système dans deux configurations, correspondant aux deux phases de la mission à réaliser. Les deux configurations

sequence	0	$\xrightarrow{f_{P_0}}$ 1	$\xrightarrow{f_{P_1}}$ 2	$\xrightarrow{f_{P_2}}$ 3	$\xrightarrow{r_{P_1}}$ 4
X_{P_0}	W	F_2	F_2	F_2	F_2
X_{P_1}	S	W_1	F_2	F_2	W_2
X_{P_2}	S	W_1	W_2	F_2	F_2
F_{Echec}	$False$	$False$	$False$	$True$	$False$

Tableau 2.6 – Comportement décrit par le modèle pour un scénario impliquant les composants P_0, P_1 et P_2

concernent les deux vannes, ce qui est correctement reporté sur l'arbre de défaillance dessiné sur la Figure 2.29. Par ailleurs, on prévoit une feuille dont le rôle sera de générer les événements de changement de phase. Naturellement, ces événements *spontanés* ne pourront être directement responsables d'une défaillance ou d'une réparation, mais ils pourront être la cause d'une commutation.

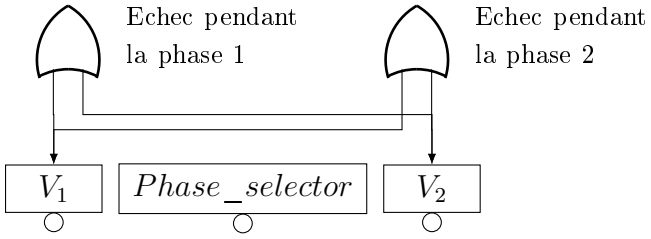

Figure 2.29 – Arbre de défaillance modélisant la structure du système pour le cas des vannes bloquantes

Le comportement d'une vanne est modélisé par le PMC Va (pour **Va**nne) représenté sur la Figure 2.30. La chaîne de Markov correspondant au mode 0 (resp. au mode 1) modélise une vanne lorsqu'elle est censée être fermée (resp. ouverte). Lorsque une vanne se bloque dans sa position courante, rien ne permet de le savoir *a priori* car elle réalise correctement sa fonction. La défaillance ne sera donc révélée que lors de la commutation dans l'autre mode (à l'ouverture ou à la fermeture selon le cas). Il n'y a pas d'arc entre F (fermée) et BO_0 (bloquée ouverte, devant être fermée), ni entre O (ouverte) et BF_1 (bloquée fermée, devant être ouverte), car une vanne ne peut se bloquer que dans sa position courante. Lorsque le système réalise la phase 1, les vannes V_1 et V_2 doivent être respectivement ouverte et fermée. Les arcs reliant la porte représentant cette phase aux feuilles représentant les vannes doivent donc être respectivement étiquetés par 1 et 0 (et vice versa pour la phase 2).

Le PMC Mi (pour profil de **Mi**ssion) représenté sur la Figure 2.31 sera utilisé pour générer les événements de changement de phase (δ est le taux de transition permettant de

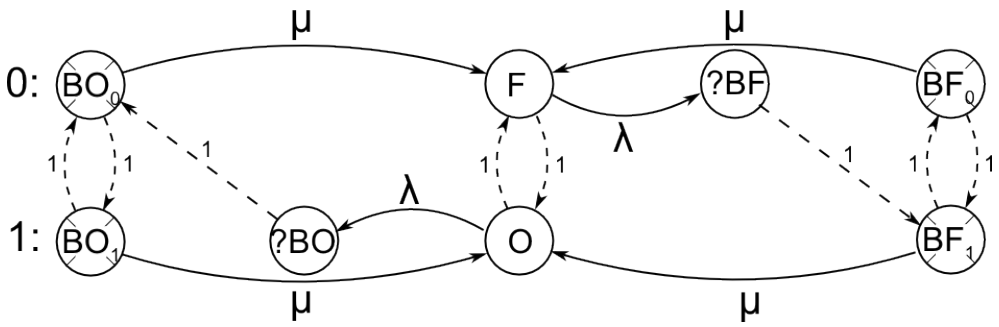

Figure 2.30 – PMC Va modélisant le comportement des vannes V_1 et V_2

déterminer la fréquence d'occurrence de ces événements). Pour que la mission commence systématiquement par la première phase, il faut que la probabilité d'être dans l'état φ_1 initialement soit 1.

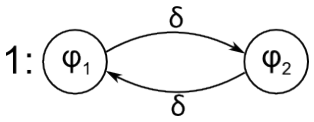

Figure 2.31 – PMC Mi générant les événements de changement de phase

La stratégie de commutation entre les phases est simple :

- Initialement, la phase 1 est effectuée ;
- puis, lorsque un événement de changement de phase survient, la phase courante est interrompue et l'autre est démarrée.

Cette stratégie peut être spécifiée par la machine de Moore représentée sur la Figure 2.32 (là encore, on fait l'hypothèse que la stratégie est appliquée correctement).

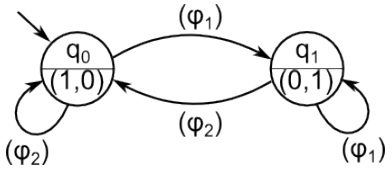

Figure 2.32 – Machine de Moore spécifiant la stratégie de reconfiguration $M3$, pour piloter le changement de phase.

La machine doit être associée à un commutateur relié à un nœud d'entrée, et deux nœuds de sortie. Le nœud d'entrée est la feuille *Phase_selector* (la variable considérée est la variable d'état du PMC associé), et les nœuds de sortie sont les portes représentant les configurations dans chacune des phases. Ainsi, la vue *arbre de défaillance enrichi* du modèle complet peut être générée (cf. Figure 2.33).

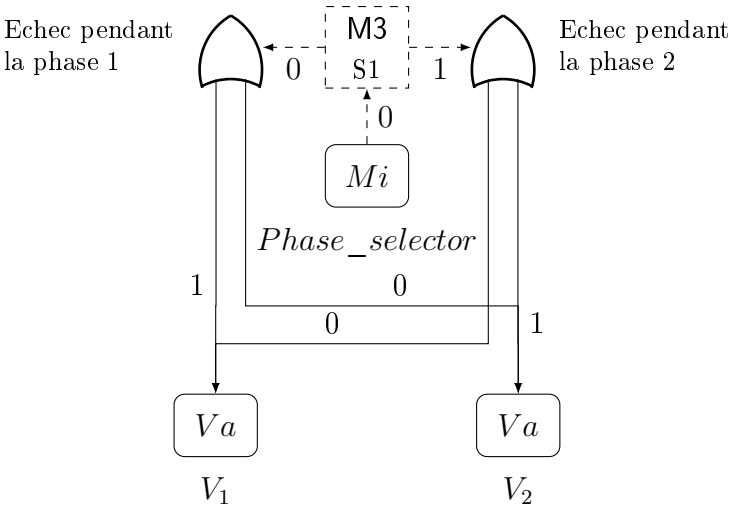

Figure 2.33 – Vue *arbre de défaillance enrichi* du modèle GBDMP des vannes bloquantes

2.5 Bilan

Dans ce chapitre un nouveau formalisme de modélisation a été élaboré en étendant et généralisant les principes qui font la force des BDMP. Les GBDMP offrent de nouvelles perspectives de modélisation qui ont été illustrées sur trois petits exemples représentatifs. En particulier, grâce à l'ajout de machines de Moore dans les primitives du formalisme, n'importe quelle stratégie de reconfiguration du système peut être prise en compte de manière réaliste et précise dans un modèle GBDMP, alors que le formalisme BDMP n'en offre qu'une représentation basique et unique. De plus, l'échec de ces stratégies est également modélisable, ce qui est original et pertinent puisque ces échecs ont un impact certain sur la sûreté de fonctionnement des systèmes.

Par ailleurs, le formalisme GBDMP permet une modélisation plus compacte et intuitive lorsque le système est constitué de composants commutables sur plus de deux modes d'opération. En effet, une feuille de BDMP ne peut modéliser que des composants ayant deux modes d'opération. Cette limitation a été levée avec les GBDMP. En conséquence, un composant du système coïncide exactement avec un élément de base du modèle GBDMP associé et ce, quelles que soient la nature du composant et la spécificité de son comportement dysfonctionnel. Cette pratique de modélisation facilite la construction et la compréhension du modèle et garantit que sa taille reste raisonnable.

Le tableau 2.7 récapitule en quoi le formalisme GBDMP enrichit les capacités de modélisation du formalisme BDMP.

BDMP	GBDMP
Les mécanismes de commutation introduits par les gâchettes se réfèrent à une stratégie de reconfiguration unique.	Les mécanismes de commutation introduits par les commutateurs se réfèrent à une stratégie entièrement spécifiable grâce à une machine de Moore.
Ces mécanismes n'échouent jamais.	L'échec de ces mécanismes peut être pris en compte, au prix d'une modification de la machine de Moore.
Si un composant a plus de deux modes d'opération, son comportement doit être distribué entre plusieurs feuilles du modèle.	Un composant du système correspond à une unique feuille du modèle, quelle que soit sa nature.
Un PMP permet de modéliser plusieurs modes de défaillance d'un composant, mais cette information ne peut être exploitée par le reste du modèle car seul son état binaire de défaillance est pris en compte.	Les commutateurs permettent d'exploiter les informations précises contenues dans les PMC.

Tableau 2.7 – Apports du formalisme GBDMP par rapport au formalisme BDMP

Le chapitre suivant est consacré à l'analyse de sûreté de fonctionnement d'un système critique basée sur un modèle GBDMP.

Chapitre 3
Analyse d'un modèle GBDMP

Sommaire

3.1 Introduction 91

3.2 Analyse qualitative 91

3.2.1 Définition des Séquences de Coupe Minimales d'un système dynamique, réparable et reconfigurable 92

3.2.1.1 Formalisation dans la théorie des langages 92

3.2.1.2 Critère de minimalité d'une séquence de coupe pour les systèmes dynamiques réparables 95

3.2.1.3 Critère de minimalité d'une séquence de coupe pour les systèmes dynamiques, réparables et reconfigurables 97

3.2.2 Calcul de l'ensemble des SCM à partir d'un modèle GBDMP 99

3.2.2.1 Traduction d'un modèle GBDMP en un automate à états fini 99

3.2.2.2 Algorithme de calcul de l'ensemble des SCM 104

3.2.2.3 Complexité algorithmique 106

3.2.3 Etude comparative 107

3.3 Analyse quantitative 113

3.3.1 Méthode de construction automatique de chaîne de Markov limitée 114

3.3.2 Traduction d'un modèle GBDMP en un modèle AltaRica . . 116

3.3.2.1 Brève présentation du langage AltaRica 117

3.3.2.2 Principe de la traduction 119

3.3.3 Etude comparative 123

3.4 Bilan 128

3.1 Introduction

A présent que l'on dispose d'un formalisme de modélisation de la SdF des systèmes dynamiques, réparables et reconfigurables, nous allons voir comment réaliser des analyses basées sur ces modèles. Ce chapitre présente deux techniques d'analyse.

L'extraction de l'ensemble des Séquences de Coupe Minimales (SCM) est une technique d'analyse *qualitative* qui a en particulier fait l'objet de la thèse [Chaux, 2013] pour les systèmes dynamiques et réparables. Cet ensemble fini correspond à l'information minimale nécessaire et suffisante pour caractériser tous les scénarios de défaillance du système. Le résultat de cette analyse est important pour l'expert de SdF car il lui permet d'identifier rapidement les faiblesses du système. Dans la section 3.2, cette technique est rappelée et étendue au cas des systèmes dynamiques réparables et reconfigurables. En outre, une étude comparative est menée afin de mettre en évidence l'intérêt de la modélisation GBDMP par rapport à la modélisation BDMP pour l'analyse qualitative.

La construction de chaîne de Markov limitée, à partir d'une description haut niveau d'un système, est une technique permettant de réaliser des analyses *quantitatives* de la SdF des systèmes complexes de grande taille. Cette technique a en particulier fait l'objet de la thèse [Brameret, 2015]. Un outil a été développé dans le cadre de cette thèse pour implémenter la méthode en partant d'un modèle décrit dans le langage AltaRica. Afin de pouvoir réutiliser l'outil en question, les principes d'un traducteur de modèle GBDMP dans le langage AltaRica sont présentés dans la section 3.3. De même que pour l'analyse qualitative, une étude comparative est menée dans cette section afin de montrer, sur un exemple simple, l'intérêt des GBDMP pour prendre en compte les effets de plusieurs stratégies de reconfiguration (et leurs possibles échecs) dans les analyses quantitatives.

3.2 Analyse qualitative

Dans cette section, la définition des SCM, proposée dans [Chaux et al., 2013] pour les système dynamiques réparables, est rappelée et est modifiée pour prendre en compte des systèmes reconfigurables. Un algorithme est ensuite proposé pour calculer l'ensemble des SCM. Enfin, une étude comparative est menée sur le cas étudié dans [Chaux et al., 2012].

3.2.1 Définition des Séquences de Coupe Minimales d'un système dynamique, réparable et reconfigurable

Dans le contexte des études de sûreté d'un système critique, une séquence de coupe est une succession d'occurrences d'événements (ou scénario) conduisant à la première défaillance du système. [Chaux et al., 2013] définit l'ensemble des SCM comme le plus petit ensemble de séquences d'événements nécessaire et suffisant pour représenter toutes les séquences de coupe. Etant donné deux séquences de coupe, il convient donc d'expliciter ce qui fait que l'une "représente" l'autre.

3.2.1.1 Formalisation dans la théorie des langages

Afin de formaliser l'analyse qualitative, cette section s'appuie sur les bases de la théorie des langages (cf. Annexe C pour un rappel des notations et des définitions élémentaires).

Dans ce cadre, l'ensemble des événements faisant évoluer un système constitue un alphabet Σ, et les scénarios caractéristiques de son comportement sont les mots d'un *langage d'évolution* $\mathcal{L}_E$ construit sur Σ. Nous noterons $\sigma \in \mathcal{L}_E$ une séquence d'évolution du système et $\sigma_{(i)} \in \Sigma$ le *i-ème* élément de la séquence σ. L'ensemble des séquences conduisant à un état défaillant du système est appelé *langage défaillant* est se note $\mathcal{L}_F \subseteq \mathcal{L}_E$. Le *langage des séquences de coupe* $\mathcal{L}_{SC}$ est enfin défini formellement par :

$$\mathcal{L}_{SC} = \{\sigma \in \mathcal{L}_F | \forall \sigma' \in Pref(\sigma) \wedge \sigma' \neq \sigma, \sigma' \notin \mathcal{L}_F\}$$

Une séquence étant une succession d'occurrences d'événements possible, en posant $\epsilon \in \mathcal{L}_E$, tout préfixe d'une séquence est également une séquence. Par conséquent, $\mathcal{L}_E$ est *préfixe clos* et donc *régulier*. D'après le théorème de Kleene, il est donc toujours possible de construire un automate à états fini déterministe $\mathcal{A} =< \Sigma, Q, q_0, Q_M, \delta >$ dont le langage *reconnu* $\mathcal{L}(\mathcal{A})$ est le *langage d'évolution* du système. De plus, en marquant les états dans lesquels le système est considéré défaillant, le langage *marqué* $\mathcal{L}_m(\mathcal{A})$ est le *langage défaillant* du système. Enfin, les séquences de coupe correspondent exactement aux chemins permettant de rejoindre un état marqué en partant de l'état initial et en ne passant que par des états non marqués. Ce langage correspond également au langage marqué de l'automate $\mathcal{A}$ duquel on aurait retiré toutes les transitions sortant d'un état marqué.

En guise d'illustration, nous allons modéliser par un automate le comportement d'un système constitué de deux composants (A et B) en redondance passive[1]. Le comportement des composants est décrit par un PMC de type F (cf. Figure 2.2). La stratégie de reconfiguration pour la redondance est du type "au plus tôt" (cf. sous-section 2.4.2.1) :

1. Si A est non défaillant, alors A et B doivent être respectivement actif et inactif.

2. Si A est défaillant, alors A et B doivent être respectivement inactif et actif.

L'automate modélisant le comportement de ce système est dessiné sur la Figure 3.1. Les configurations 1 et 2 mentionnées ci-dessus sont représentés par les états respectivement en blanc et en gris.

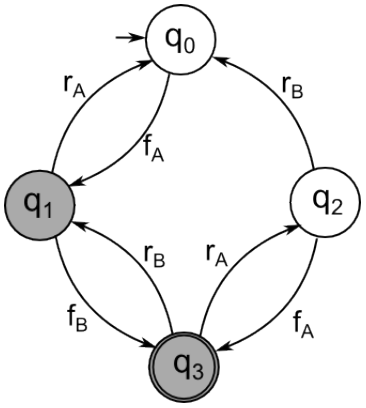

Figure 3.1 – Modèle de deux composants en redondance passive

Considérons à présent que le succès de la stratégie de reconfiguration n'est pas garanti et qu'il dépend de l'état d'un troisième composant C. Lorsque celui-ci défaille, le système reste bloqué dans sa configuration courante et ce, jusqu'à ce que C soit réparé (il n'a qu'un seul mode de défaillance : *frozen*). Le nouveau comportement est dessiné sur la Figure 3.2. Cet automate montre bien que lorsque C est défaillant, la défaillance de A (resp. B) suffit à faire défaillir le système s'il est bloqué dans la configuration 1 (resp. 2). De plus, les composants ne peuvent défaillir que lorsqu'ils sont actifs, ce qui explique pourquoi le composant A (resp. B) ne peut défaillir que dans la configuration 1 (resp. 2).

Dans le cas des systèmes réparables, l'ensemble des séquences de coupe n'est pas fini. L'objectif de l'analyse qualitative est donc de déterminer un sous-ensemble fini de

1. On suppose que deux réparateurs sont constamment disponibles.

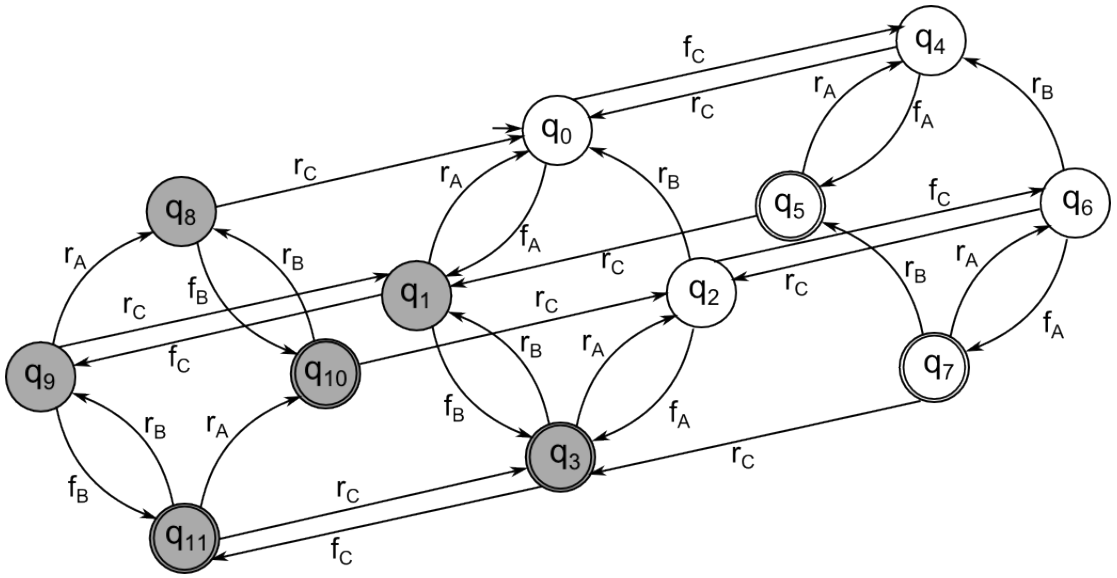

Figure 3.2 – Modèle de deux composants en redondance passive et d'un composant responsable de la commutation

$\mathcal{L}_{SC}$ nécessaire et suffisant pour *représenter* $\mathcal{L}_{SC}$ du point de vue d'un expert. Ce sous-ensemble est appelé *ensemble des Séquences de Coupe Minimales* et est noté SCM. Le qualificatif "minimale" appelle la définition d'une relation d'ordre. Dans le cas des séquences de coupe, il convient donc de formaliser la relation d'ordre $\mathcal{R}$ "représente", permettant de définir l'ensemble $\mathcal{L}_{SCM}^{\mathcal{R}}$ comme l'ensemble des éléments minimaux de $\mathcal{L}_{SC}$ pour la relation $\mathcal{R}$:

$$\mathcal{L}_{SCM}^{\mathcal{R}} = \{\sigma \in \mathcal{L}_{SC} | \forall \sigma' \in \mathcal{L}_{SC}, \sigma' \mathcal{R} \sigma \Longrightarrow \sigma' = \sigma\}$$

La première relation à laquelle on peut penser pour jouer ce rôle est l'inclusion simple des séquences, qui est formellement définie ci-dessous.

Définition 3.1. *Soit* $(\sigma, \sigma') \in \mathcal{L}_{SC}^2$,

$$\sigma \subseteq \sigma' \Longleftrightarrow \sigma' \in \Sigma^* \sigma_{(1)} \Sigma^* ... \Sigma^* \sigma_{(|\sigma|)} \Sigma^*$$

Ainsi, pour l'exemple exposé ci-dessus (cf. Figure 3.2), on a $\mathcal{L}_{\overline{\subseteq}SCM}^{\subseteq} = \{f_A f_B, f_C f_A\}$. En particulier, $f_A f_C r_A f_B \notin \mathcal{L}_{\overline{\subseteq}SCM}^{\subseteq}$ car $f_A f_B \subseteq f_A f_C r_A f_B$. Ce critère de minimalité n'est pas satisfaisant, car un expert ne considèrerait vraisemblablement pas que la séquence $f_A f_B$ représente la séquence $f_A f_C r_A f_B$. En effet, contrairement à la première, cette deuxième séquence permet de comprendre l'influence de la défaillance du composant C : échec de la commutation de B vers A, suite à la réparation de A. Les deux séquences

sont donc porteuses d'une information différente. C'est cependant cette relation qui a été utilisée dans [Chaux et al., 2012] afin de calculer l'ensemble des SCM, pour le cas d'étude que nous reprendrons dans la sous-section 3.2.3. Ce choix été justifié car les échecs de la commutation n'étaient pas pris en compte.

Une autre relation à laquelle on pense naturellement est liée à la longueur des séquences permettant de rejoindre les différents états défaillants. Cette seconde relation est définie ci-après.

Définition 3.2. *Soit* $(\sigma, \sigma') \in \mathcal{L}_{SC}^2$,

$$\sigma \leq \sigma' \iff (|\sigma| \leq |\sigma'|) \wedge (\delta(\sigma) = \delta(\sigma'))$$

Les SCM, selon cette relation d'ordre, correspondent donc aux chemins les plus courts permettant de rejoindre chaque état marqué en partant de l'état initial et en ne passant que par des états non marqués. Pour l'exemple ci-dessus, on a donc : $\mathcal{L}_{SCM}^{\leq} = \{f_A f_B, f_C f_A, f_A f_C f_B, f_A f_C r_A f_B\}$ [2]. Une fois de plus, ce critère de minimalité n'est pas satisfaisant, car un expert considèrerait vraisemblablement que le scénario $f_A f_C f_B$ est représenté par le scénario $f_A f_B$. En effet, la défaillance du composant C ne change rien au fait que le système a besoin qu'au moins un des deux composants A et B soit non défaillant pour fonctionner.

Il est donc nécessaire de déterminer une nouvelle relation d'ordre entre les séquences, qui sera nécessairement liée à leur signification en terme de SdF.

3.2.1.2 Critère de minimalité d'une séquence de coupe pour les systèmes dynamiques réparables

Dans [Chaux et al., 2013], l'objet d'étude est un système dynamique réparable réalisant une unique phase de mission, donc non reconfigurable au sens de cette thèse. Dans ce cadre, les scénarios dysfonctionnels du système peuvent être décrits uniquement avec des événements de panne et de réparation des composants. Etant donné $\mathcal{C}$ l'ensemble des composants du système, on peut donc partitionner l'alphabet comme tel :

$$\Sigma = \Sigma_F \cup \Sigma_R = \bigcup_{c \in \mathcal{C}} \Sigma_F^c \cup \bigcup_{c \in \mathcal{C}} \Sigma_R^c$$

2. Le chemin le plus court pour rejoindre l'état q_7 est $f_A f_B r_A f_C f_A$, mais ce n'est pas une séquence de coupe.

où Σ_F^c et Σ_R^c sont respectivement constitués des événements de défaillance et de réparation du composant c. Ce partitionnement permet de définir une fonction pour déterminer l'ensemble des composants défaillants à l'issue d'une séquence :

$$FC : \begin{array}{ccc} \mathcal{L}_E & \longrightarrow & \mathcal{C} \\ \sigma & \longmapsto & \{c \in \mathcal{C} | \ \ |\sigma_{|\Sigma_F^c}| > |\sigma_{|\Sigma_R^c}|\} \end{array}$$

Aussi, selon [Chaux et al., 2013], une séquence de coupe est représentée par une autre, si elle peut être construite à partir de la seconde, à l'aide des deux constructeurs suivants utilisés de manière itérée et combinée ou non :

1. ajout, au cours de la séquence, d'un événement de défaillance :

$$f_1 : \begin{array}{ccc} \mathcal{L}_E & \longrightarrow & \mathcal{P}(\mathcal{L}_E) \\ \sigma & \longmapsto & \left\{\sigma' \in \mathcal{L}_E | \exists u \in \Sigma_F, \exists (\sigma_1, \sigma_2) \in (\Sigma^*)^2, \sigma = \sigma_1\sigma_2 \wedge \sigma' = \sigma_1 u \sigma_2\right\} \end{array}$$

2. distribution ordonnée d'un ensemble d'événements ne modifiant pas l'ensemble des composants défaillants :

$$f_2 : \begin{array}{ccc} \mathcal{L}_E & \longrightarrow & \mathcal{P}(\mathcal{L}_E) \\ \sigma & \longmapsto & \left\{\sigma' \in \mathcal{L}_E | \exists \theta \in \Sigma^*, \exists (\sigma_0, ..., \sigma_{|\theta|}) \in (\Sigma^*)^{|\theta|+1}, \sigma = \sigma_0 ... \sigma_{|\theta|} \wedge \right. \\ & & \left. \sigma' = \sigma_0 \theta_{(1)} \sigma_1 ... \sigma_{|\theta|-1} \theta_{(|\theta|)} \sigma_{|\theta|} \wedge FC(\sigma) = FC(\sigma')\right\} \end{array}$$

Ces définitions sont étendues aux langages :

$$\text{Soit } \mathcal{L} \subseteq \mathcal{L}_E, f_1(\mathcal{L}) = \bigcup_{\sigma \in \mathcal{L}} f_1(\sigma) \text{ et } f_2(\mathcal{L}) = \bigcup_{\sigma \in \mathcal{L}} f_2(\sigma)$$

Selon cette acception, la relation d'ordre permettant de construire l'ensemble des SCM peut se définir formellement ainsi :

Définition 3.3. *Soit* $(\sigma, \sigma') \in \mathcal{L}_{SC}^2$,

$$\sigma \models \sigma' \Longleftrightarrow \exists n \in \mathbb{N} | \sigma' \in f_2(f_1^n(\sigma))$$

L'ensemble des SCM résultant est le plus petit ensemble de séquences de coupe permettant de construire au moins toutes les séquences de coupes :

$$\begin{cases} \mathcal{L}_{SC} \subseteq \bigcup\limits_{n \in Card(\mathcal{C})} f_2(f_1^n(\mathcal{L}_{SCM}^{\models})) \\ \forall \mathcal{L} \subseteq \mathcal{L}_{SC}, \ \ \mathcal{L}_{SC} \subseteq \bigcup\limits_{n \in Card(\mathcal{C})} f_2(f_1^n(\mathcal{L})) \Longrightarrow \mathcal{L}_{SCM}^{\models} \subseteq \mathcal{L} \end{cases}$$

Pour l'exemple proposé ci-dessus, on a $\mathcal{L}^{\models}_{SCM} = \{f_A f_B, f_C f_A, f_A f_C r_A f_B\}$. On peut en particulier vérifier que :

- la séquence $f_A f_C f_B$ peut être construite en ajoutant l'événement de défaillance f_C à la séquence $f_A f_B$, conformément au constructeur f_1 ;
- la séquence $f_C f_A r_C f_B$ peut être construite en distribuant $f_C r_C$ dans la séquence $f_A f_B$, conformément au constructeur f_2 ;
- en revanche, la séquence $f_A f_C r_A f_B$ *ne* peut *pas* être construite en distribuant $f_C r_A$ dans la séquence $f_A f_B$, conformément au constructeur f_2 : en effet, cela modifierait l'ensemble des composants défaillants.

Ce critère de minimalité se justifie au regard de la cohérence des systèmes dynamiques réparables, qui garantit en particulier que toute séquence obtenue par extension d'une séquence défaillante, à l'aide des deux constructeurs ci-dessus, est défaillante :

$$\forall \mathcal{L} \subseteq \mathcal{L}_F, \forall n \in \mathbb{N}, f_2(f_1^n(\mathcal{L})) \subseteq \mathcal{L}_F$$

3.2.1.3 Critère de minimalité d'une séquence de coupe pour les systèmes dynamiques, réparables et reconfigurables

Pour les systèmes reconfigurables, comme on l'a vu dans le chapitre précédent, les scénarios dysfonctionnels du système ne peuvent plus être décrits uniquement avec des événements de défaillance et de réparation des composants. Ainsi, par exemple, l'ajout d'un événement de changement de phase dans une séquence défaillante (resp. non défaillante) peut engendrer une séquence non défaillante (resp. défaillante), sans pour autant modifier l'ensemble des composants défaillants.

Afin d'illustrer ce phénomène, nous reprenons la première version de l'exemple précédent, i.e. sans prendre en compte le composant C (cf. Figure 3.1). Considérons à présent que le système réalise deux phases de mission. Dans la première phase, les composants sont en redondance passive, tandis que dans la seconde, ils sont tous deux nécessaires à la réalisation de la fonction (fonctionnement en série). Le système a donc trois configurations possibles : A est actif seul, B est actif seul ou A et B sont tous deux actifs. Le comportement dysfonctionnel associé à ce système multi-phases est représenté par l'automate dessiné sur la Figure 3.3 (le symbole φ correspond à l'événement de changement

de phase). Pour ce système, l'ajout de l'événement φ à la fin de la séquence défaillante φf_A engendre la séquence non défaillante $\varphi f_A \varphi$.

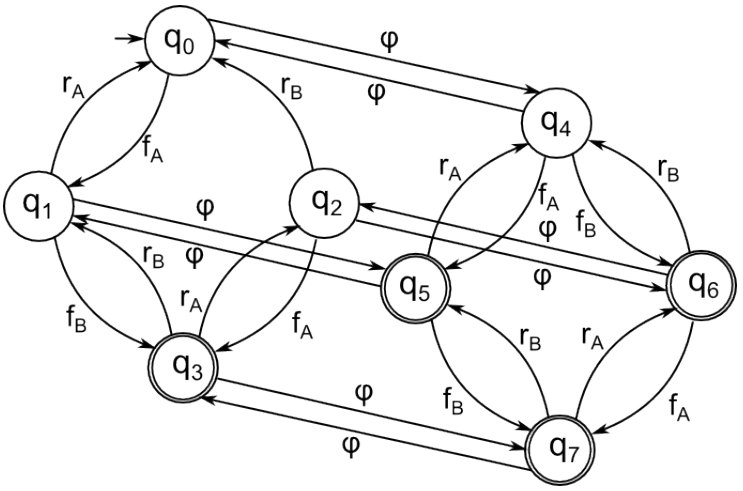

Figure 3.3 – Automate à état modélisant le comportement dysfonctionnel d'un système bi-phasé à deux composants

La notion de cohérence des systèmes dynamiques réparables ne peut donc pas être directement étendue aux systèmes reconfigurables. Nous allons donc proposer une nouvelle acception de la minimalité d'une séquence de coupe.

Cette nouvelle définition repose sur le concept de *coupe minimale*. Les coupes minimales d'un système statique sont les plus petits ensembles de composants, dont les défaillances impliquent celle du système (cf. [Rauzy, 2001]). Elles sont donc définies à l'aide de la relation d'inclusion entre les ensembles de composants du système. Par ailleurs, comme on l'a évoqué au chapitre 1, un système dynamique se différencie d'un système statique par la nécessité de prendre en compte l'ordre d'occurrence des événements caractéristiques de son évolution. L'ensemble des SCM doit donc être défini à l'aide d'une relation qui préserve cet ordre (comme l'inclusion des séquences). Toutefois, l'état dysfonctionnel d'un système dynamique est, comme pour un système statique, intimement lié à celui de ses composants. Cette observation nous amène à formuler le postulat suivant :

Postulat. *Une séquence de coupe est caractérisée non seulement par l'ordre d'occurrence des événements qu'elle représente, mais également par l'ensemble des composants défaillants à l'issue de celle-ci.*

Aussi, une *séquence de coupe minimale* peut être vue comme une *coupe minimale* pour laquelle on précise un ordre d'occurrence des événements y conduisant.

Ce nouvel éclairage permet de définir formellement la relation d'ordre que nous recherchons [3], ainsi que l'ensemble des SCM :

Définition 3.4. *Soit* $(\sigma, \sigma') \in \mathcal{L}_{SC}^2$,

$$\sigma \Subset \sigma' \iff (\sigma \subseteq \sigma') \wedge (FC(\sigma) \subseteq FC(\sigma'))$$

Définition 3.5. *Ensemble des Séquences de Coupe Minimales d'un système dynamique :*

$$\mathcal{L}_{SCM} = \mathcal{L}_{SCM}^{\Subset} = \{\sigma \in \mathcal{L}_{SC} | \forall \sigma' \in \mathcal{L}_{SC}, \sigma' \Subset \sigma \Longrightarrow \sigma' = \sigma\}$$

Cette nouvelle définition est suffisamment générique pour convenir aux systèmes dynamiques, qu'ils soient réparables ou non, reconfigurables ou non. De fait, on peut vérifier que :

$$\forall (\sigma, \sigma') \in \mathcal{L}_{SC}^2, \quad \sigma \models \sigma' \Longrightarrow \sigma \Subset \sigma'$$

La réciproque étant fausse, la nouvelle relation d'ordre est moins restrictive que celle proposée dans [Chaux et al., 2013]. Elle permet en particulier d'obtenir des résultats satisfaisants pour le premier exemple (cf. Figure 3.2) : $\mathcal{L}_{SCM}^{\Subset} = \mathcal{L}_{SCM}^{\models}$; ainsi que pour le second (cf. Figure 3.3) : $\mathcal{L}_{SCM}^{\Subset} = \{f_A f_B, f_A \varphi, \varphi f_A, \varphi f_B\}$.

3.2.2 Calcul de l'ensemble des SCM à partir d'un modèle GBDMP

A présent que l'on a formellement défini ce qu'était l'ensemble des SCM, nous présentons comment extraire cet ensemble d'un modèle GBDMP. La première étape consiste à expliciter l'automate à états fini que décrit par intention un modèle GBDMP, puis nous proposerons un algorithme pour construire automatiquement l'ensemble des SCM d'un tel modèle.

3.2.2.1 Traduction d'un modèle GBDMP en un automate à états fini

Etant donné $\mathcal{M} =< V, E, \kappa, \upsilon, str, pmc >$ un modèle GBDMP bien formé, l'objectif de cette sous-section est de construire l'automate $\mathcal{A} =< \Sigma, Q, q_0, Q_M, \delta >$ décrit par $\mathcal{M}$. Afin que l'automate construit soit déterministe, on fait l'hypothèse que l'état initial du modèle est unique. Formellement, cela implique que pour toute feuille l de dimension k :

$$\forall i \in [\![0, k-1]\!], \ \exists! x \in \mathcal{X}_i^{pmc(l)} | p0_i^{pmc(l)}(x) = 1 \tag{3.1}$$

3. On peut vérifier simplement qu'il s'agit bien d'une relation d'ordre.

Afin d'illustrer la démarche, nous allons partiellement traduire le GBDMP représenté sur la Figure 3.4 en automate à états fini.

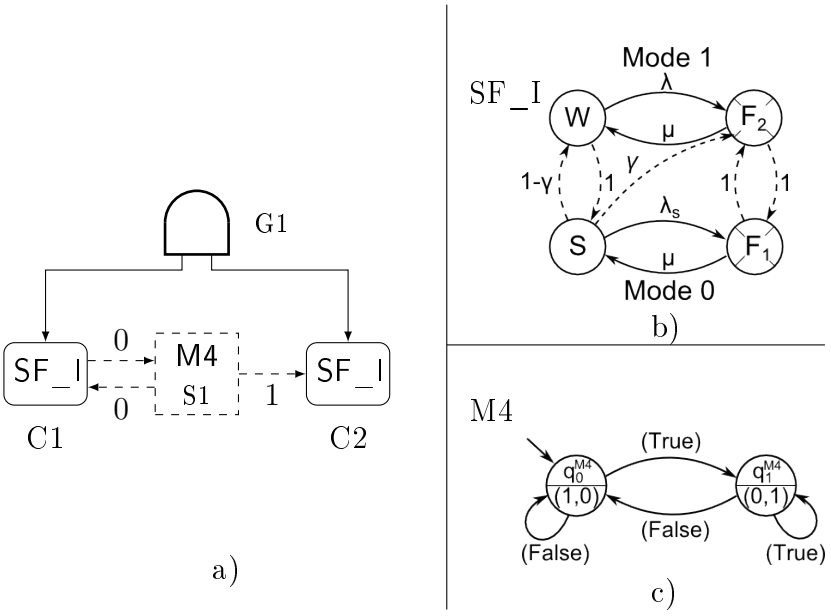

Figure 3.4 – Exemple simple de modèle GBDMP

Dans le cadre des GBDMP, les événements à considérer correspondent exactement aux changements d'état d'un PMC associé à une feuille (i.e. les flèches en traits pleins et en pointillés sur les représentations graphiques). Ainsi, un événement $e^l_{o\to d}$ peut être défini par un triplet $< l, o, d >$, avec :

- $l \in L$, la feuille génératrice de l'événement ;
- $(o, d) \in \left(\mathcal{X}^{pmc(l)}\right)^2$, les états du PMC associés à l, origine et destination de la transition considérée.

Comme on l'a vu au chapitre précédent, l'ensemble des événements peut être partitionné entre les événements *spontanés* et les événements *provoqués*. Aussi, pour une feuille l de dimension k, on a :

$$\begin{cases} \Sigma^l_S = \left\{ e^l_{o\to d} | \exists i \in [\![0, k-1]\!], (o,d) \in \left(\mathcal{X}_i^{pmc(l)}\right)^2 \wedge A_i^{pmc(l)}(o,d) \neq 0 \right\} \\ \Sigma^l_P = \left\{ e^l_{o\to d} | \exists (i,j) \in [\![0, k-1]\!]^2, i \neq j \wedge (o,d) \in \mathcal{X}_i^{pmc(l)} \times \mathcal{X}_j^{pmc(l)} \wedge f_{i\to j}^{pmc(l)}(o,d) \neq 0 \right\} \end{cases}$$

On peut donc proposer un premier partitionnement de l'alphabet :

$$\Sigma = \Sigma_S \cup \Sigma_P = \bigcup_{l\in L} \Sigma^l_S \cup \bigcup_{l\in L} \Sigma^l_P \tag{3.2}$$

Ainsi, pour le modèle proposé, on a :

$$\Sigma_S^{C1} = \{e_{W\to F_2}^{C1}, e_{F_2\to W}^{C1}, e_{S\to F_1}^{C1}, e_{F_1\to S}^{C1}\}$$
$$\Sigma_P^{C1} = \{e_{W\to S}^{C1}, e_{S\to W}^{C1}, e_{S\to F_2}^{C1}, e_{F_1\to F_2}^{C1}, e_{F_2\to F_1}^{C1}\}$$

Par ailleurs, chacun de ces événements est considéré comme événement *neutre*, de *défaillance* ou de *réparation*, selon qu'il fait varier ou non l'ensemble des composants défaillants. Aussi, pour une feuille l, on a :

$$\begin{cases} \Sigma_F^l = \left\{ e_{o\to d}^l \in \Sigma^l | o \notin \mathcal{X}_F^{pmc(l)} \wedge d \in \mathcal{X}_F^{pmc(l)} \right\} \\ \Sigma_R^l = \left\{ e_{o\to d}^l \in \Sigma^l | o \in \mathcal{X}_F^{pmc(l)} \wedge d \notin \mathcal{X}_F^{pmc(l)} \right\} \\ \Sigma_N^l = \Sigma^l \backslash \left(\Sigma_F^l \cup \Sigma_R^l \right) \end{cases}$$

On peut donc proposer un second partitionnement de l'alphabet :

$$\Sigma = \Sigma_N \cup \Sigma_F \cup \Sigma_R = \bigcup_{l\in L} \Sigma_N^l \cup \bigcup_{l\in L} \Sigma_F^l \cup \bigcup_{l\in L} \Sigma_R^l \tag{3.3}$$

Ainsi, pour le modèle proposé, on a :

$$\Sigma_F^{C1} = \{e_{W\to F_2}^{C1}, e_{S\to F_1}^{C1}, e_{S\to F_2}^{C1}\}$$
$$\Sigma_S^{C1} = \{e_{F_2\to W}^{C1}, e_{F_1\to S}^{C1}\}$$
$$\Sigma_N^{C1} = \{e_{W\to S}^{C1}, e_{S\to W}^{C1}, e_{F_1\to F_2}^{C1}, e_{F_2\to F_1}^{C1}\}$$

Les états d'un modèle GBDMP correspondent à des assignations de valeurs à ces variables caractéristiques (cf. section 2.3.2). Dans la suite, nous noterons $X(q)$ la valeur assignée à X dans l'état q. Par ailleurs, on a montré au chapitre précédent que les valeurs des status de *défaillance*, de *réquisition* et d'*activité* peuvent être déduites des valeurs des variables d'états des PMC et machines de Moore associés aux feuilles et aux commutateurs (cf. formules 2.1, 2.2, 2.3 et 2.4). La donnée des valeurs de ces variables d'état suffit donc à déterminer l'état du modèle [4].

L'espace d'état Q étant décrit par intention, celui-ci ne peut être construit que récursivement, en partant de l'état initial q_0, et en identifiant pour chaque nouvel état rencontré quels sont les événements pouvant survenir. L'initialisation du modèle est évidemment liée aux états initiaux des machines de Moore et aux distributions de probabilités initiales des PMC. Sous réserve que l'hypothèse 3.1 soit vérifiée pour chaque

4. Par exemple, dans la formule 3.4, qui décrit l'état initial q_0, les valeurs des M_l utilisées dans la seconde ligne sont déduites des valeurs attribuées aux U_s dans la première ligne.

feuille $l \in L$, l'état initial peut être déterminé grâce à la formule suivante :

$$\begin{cases} \forall s \in S,\ U_s(q_0) = q_0^{str(c)} \\ \forall l \in L,\ p0_{M_l(q_0)}^{pmc(l)}(X_l(q_0)) = 1 \end{cases} \tag{3.4}$$

Pour l'exemple proposé, l'état initial q_0 est alors tel que :

$$\begin{cases} U_{S1}(q_0) = q_0^{M4} \\ X_{C1}(q_0) = W \\ X_{C2}(q_0) = S \end{cases}$$

Il est évident qu'un événement $e_{o\to d}^{l}$ ne peut survenir que si le PMC associé à la feuille l est dans l'état o. Par ailleurs, on sait qu'un événement *spontané* ne peut survenir que si toutes les feuilles sont dans le mode d'opération correspondant à leur statut d'activité. Dans le cas contraire, les événements *provoqués* conduisant les feuilles dans le bon mode d'opération peuvent survenir. Ces informations permettent de définir une fonction $\mathcal{E}$, qui détermine les événements pouvant survenir dans un état donné :

$$\begin{array}{rccl} & Q & \longrightarrow & \mathcal{P}(\Sigma) \\ \mathcal{E} : & & & \begin{cases} \text{si } \forall l \in L, X_l(q) \in \mathcal{X}_{M_l(q)}^{pmc(l)} & \{e_{o\to d}^{l} \in \Sigma_S | X_l(q) = o\} \\ \text{sinon} & \{e_{o\to d}^{l} \in \Sigma_P | X_l(q) = o \wedge d \in \mathcal{X}_{M_l(q)}^{pmc(l)}\} \end{cases} \\ & q & \longmapsto & \end{array}$$

Pour l'exemple proposé, on a donc :

$$\mathcal{E}(q_0) = \{e_{W\to F_2}^{C1}, e_{S\to F_1}^{C2}\}$$

L'occurrence d'un événement a pour conséquence directe de modifier l'état d'une feuille du modèle. Par ailleurs, on rappelle que les autres variables du modèle ne doivent être actualisées que suite à l'occurrence d'un événement *spontané*, ou lorsque toutes les feuilles ont été commutées dans le bon mode. Aussi, pour un état donné, si les événements pouvant survenir sont des événements *provoqués* concernant plus d'une feuille, les variables d'états des commutateurs ne doivent pas être modifiées suite à l'occurrence d'un de ces événements, car il restera encore au moins une feuille à commuter. Il est donc intéressant de remarquer que l'ordre d'occurrence des événements *provoqués* n'a pas d'incidence sur l'évolution du modèle. La fonction de transition caractéristique d'un

modèle GBDMP peut donc à présent être formellement définie [5] :

Soit $q \in Q$, si $e^l_{o\to d} \in \mathcal{E}(q)$, alors $\delta(q, e^l_{o\to d}) = q'$ tel que,

- si $Card(\{l \in L | \mathcal{E}(q) \cap \Sigma^l_P \neq \varnothing\}) > 1$:

$$\begin{cases} & X_l(q') = d \\ \forall l' \in L\backslash\{l\}, & X_{l'}(q') = X_{l'}(q) \\ \forall s \in S & U_s(q') = U_s(q) \end{cases} \tag{3.5}$$

- sinon :

$$\begin{cases} & X_l(q') = d \\ \forall l' \in L\backslash\{l\}, & X_{l'}(q') = X_{l'}(q) \\ \forall s \in S & U_s(q') = trans^{str(s)}(U_s(q), Input_s(q')) \end{cases}$$

Ainsi, l'espace d'état Q décrit par un modèle GBDMP peut se construire récursivement de la manière suivante :

$$\begin{cases} q_0 \in Q \\ \forall q \in Q, \forall e^l_{o\to d} \in \mathcal{E}(q), \delta(q, e^l_{o\to d}) \in Q \end{cases} \tag{3.6}$$

Enfin, pour déterminer l'ensemble des états marqués, il est nécessaire à ce stade de choisir le *nœud cible* du modèle $nc \in N$ dont le status de *défaillance* correspond à la perte de la fonction ciblée par l'analyse. En outre, seul un état *stable* -c'est à dire un état que l'on ne peut quitter que sur occurrence d'un événement *spontané*- pourra être considéré comme défaillant :

$$Q_M = \{q \in Q | \mathcal{E}(q) \subseteq \Sigma_S \wedge F_{nc}(q) = True\} \tag{3.7}$$

Afin d'illustrer cette traduction, la Figure 3.5 représente une partie de l'automate construit à partir du GBDMP donné en exemple. Les rectangles en traits pleins (resp. pointillés) correspondent aux états stables (resp. instables) du GBDMP. L'unique état marqué est encadré deux fois.

Dans cette sous-section ont été formalisé les outils nécessaires pour traduire un modèle GBDMP $\mathcal{M}$ bien formé en un automate à états fini déterministe $\mathcal{A}$, sous réserve que l'hypothèse 3.1 soit vérifiée pour chaque feuille $l \in L$, et qu'un *nœud cible* ait été désigné : $\mathcal{M} \xrightarrow{3.2,\ 3.4,\ 3.5,\ 3.6,\ 3.7} \mathcal{A}$.

5. Cette définition introduit pour chaque commutateur $s \in S$ une fonction $Input_s : Q \longrightarrow \Sigma_I^{str(s)}$, qui fournit la valeur des entrées de la machine de Moore associée à s pour un état du modèle GBDMP donné. On a montré au chapitre précédent que si le modèle est bien formé, pour chaque $s \in S$ et étant donné un état $q \in Q$, la valeur des entrées $Input_s(q)$ se déduit des valeurs $(X_l(q))_{l\in L}$.

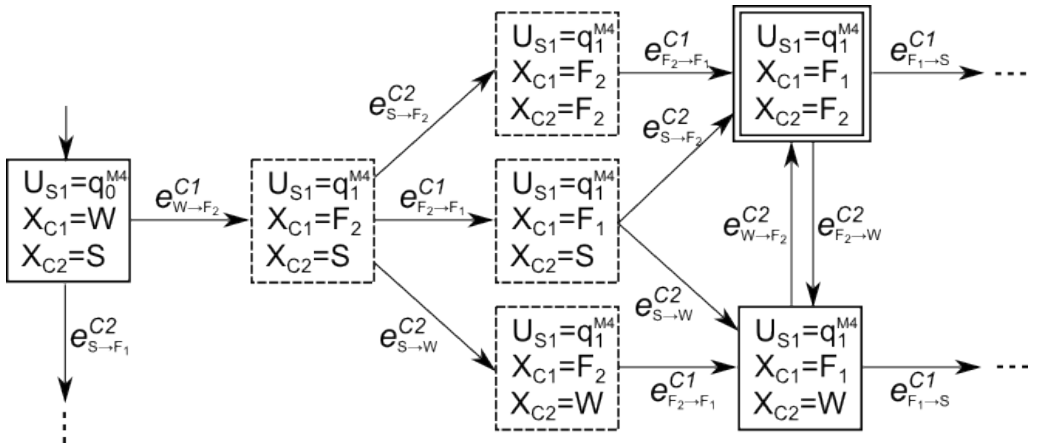

Figure 3.5 – Construction partielle de l'automate équivalent au GBDMP proposé.

3.2.2.2 Algorithme de calcul de l'ensemble des SCM

Le principe de l'algorithme 2 est de parcourir en largeur l'espace d'état décrit par un GBDMP et de construire l'ensemble des SCM "à la volée". Le parcours en largeur permet de garantir la possibilité de déterminer la minimalité d'une séquence en fonction des SCM déjà trouvées. En effet, une séquence ne peut être représentée que par elle-même ou par une séquence plus courte. Ainsi, lorsqu'une nouvelle séquence de coupe est identifiée, on la compare aux SCM pour savoir si elle est déjà représentée par l'une d'entre elle. Dans le cas contraire, on l'ajoute à l'ensemble des SCM (cf. lignes 17 et 18).

Par souci de clarté, les événements *neutres provoqués* seront masqués dans les séquences. En effet, il est raisonnable de considérer que l'occurrence de ces événements (qui modélisent des commutations de composants prévues par une stratégie de reconfiguration) n'apporte pas d'information utile pour la compréhension d'un scénario de défaillance. Comme les événements de *réparation provoqués* n'existent pas dans un modèle GBDMP bien formé, les séquences de coupe seront projetées sur l'alphabet $\Sigma_S \cup \Sigma_F$ (cf. lignes 17 et 18). Aussi, les lignes 12 et 13, traduisent la non augmentation de la taille de la séquence après projection, suite à l'ajout d'un événement *neutre provoqué*.

En outre, on définit deux critères d'arrêt de l'exploration :

1. on n'explore pas au delà du premier état défaillant rencontré (en effet, les séquences que l'on découvrirait ne seraient pas des séquences de coupe, par définition) ;

2. on n'explore pas au delà d'un état déjà rencontré dans la séquence considérée (il est évident qu'une séquence de coupe contenant une boucle est représentée par la même séquence sans boucle).

Les lignes 14 à 16 de l'algorithme traduisent la négation de ces critères d'arrêt : si l'état considéré q_{cur} n'est pas défaillant, et n'a jamais été rencontré dans la séquence considérée σ_{cur}, alors on poursuivra l'exploration à partir de cet état, sinon, si q_{cur} est défaillant, alors σ_{cur} est nécessairement une séquence de coupe.

Il est intéressant de remarquer que cet algorithme de recherche des SCM est valable quel que soit la relation d'ordre considérée, sous réserve que celle-ci soit liée à l'accroissement de la taille des séquences. Pour appliquer l'algorithme avec une autre relation d'ordre, il suffit de remplacer la relation $\Subset$ dans la ligne 17.

Algorithme 2 Calcul de l'ensemble des SCM d'un modèle GBDMP

Inputs:
- $< V, E, \kappa, \upsilon, str, pmc >$ un modèle GBDMP bien formé.
- $< \Sigma, Q, q_0, Q_M, \delta >$ l'automate équivalent au modèle (cf. formules 3.2, 3.4, 3.5, 3.6, 3.7).

Outputs: $\mathcal{L}_{SCM}$ l'ensemble des SCM représentatives des séquences de coupe du modèle projetées sur l'alphabet $\Sigma_S \cup \Sigma_F$.

1: *// Initialisation :*
2: $\mathcal{L}_{SCM} := \varnothing$
3: $seqOfLastLength := \{\epsilon\}$
4: $seqOfCurLength := \varnothing$
5: *// Boucle principale :*
6: **while** $seqOfLastLength \neq \varnothing$ **do**
7: **for all** $\sigma_{last} \in seqOfLastLength$ **do**
8: $q_{last} = \delta(q_0, \sigma_{last})$
9: **for all** $u \in \mathcal{E}(q_{last})$ **do**
10: $q_{cur} = \delta(q_{last}, u)$
11: $\sigma_{cur} = \sigma_{last} u$
12: **if** $u \in \Sigma_P \cap \Sigma_N$ **then**
13: $seqOfLastLength := seqOfLastLength \cup \{\sigma_{cur}\}$
14: **else if** $q_{cur} \notin Q_M \wedge \nexists \sigma \in Pref(\sigma_{last}) | \delta(q_0, \sigma) = q_{cur}$ **then**
15: $seqOfCurLength := seqOfCurLength \cup \{\sigma_{cur}\}$
16: **else if** $q_{cur} \in Q_M$ **then**
17: **if** $\nexists \sigma_{min} \in \mathcal{L}_{SCM} | \sigma_{min} \Subset \sigma_{cur|\Sigma_S \cup \Sigma_F}$ **then**
18: $\mathcal{L}_{SCM} = \mathcal{L}_{SCM} \cup \{\sigma_{cur|\Sigma_S \cup \Sigma_F}\}$
19: **end if**
20: **end if**
21: **end for**
22: **end for**
23: $seqOfLastLength := seqOfCurLength$
24: $seqOfCurLength := \varnothing$
25: **end while**

3.2.2.3 Complexité algorithmique

L'objectif de cette sous-section est d'estimer une borne supérieure de la complexité de l'algorithme 2, en fonction des paramètres du modèle GBDMP $\mathcal{M}$ fourni en entrée.

La première étape consiste à borner la taille de l'espace d'état décrit par $\mathcal{M}$. On a vu qu'un état du modèle était complètement déterminé par la valeur des variables d'état des PMC et machines de Moore respectivement associés aux feuilles et aux commutateurs. Ainsi $Card(Q) \leq dim_{\mathcal{M}}$, avec :

$$dim_{\mathcal{M}} = \prod_{l \in L} Card(\mathcal{X}^{pmc(l)}) \prod_{s \in S} Card(Q^{str(s)})$$

L'algorithme 2 construit toutes les séquences de coupe sans boucle, ce qui revient à explorer tous les chemins du graphe représentant l'automate équivalent à $\mathcal{M}$ ne passant pas deux fois dans le même état, partant de l'état initial q_0 et finissant dans le premier état marqué rencontré. Dans le pire des cas, l'automate est fortement connexe, et seul un état est marqué (différent de l'état initial). Dans cette éventualité, le nombre de chemins possibles pour rejoindre l'état marqué en partant de l'état initial (sans passer deux fois par le même état) est le nombre d'arrangements d'états de longueur maximale $dim_{\mathcal{M}}$, dont le premier élément est l'état initial et le dernier est l'état marqué [6] :

$$\sum_{k=0}^{dim_{\mathcal{M}}-2} A^k_{dim_{\mathcal{M}}-2} = \sum_{k=0}^{dim_{\mathcal{M}}-2} \frac{(dim_{\mathcal{M}}-2)!}{(dim_{\mathcal{M}}-2-k)!}$$

Le terme le plus grand de cette somme est $(dim_{\mathcal{M}} - 2)!$, donc ce dénombrement est majorée par $(dim_{\mathcal{M}} - 1)!$.

Par ailleurs, pour chacun des chemins rencontrés, des tests sont réalisés, parmi lesquels le plus complexe est le test d'inclusion d'une séquence dans une autre qui nécessite de parcourir la plus longue des deux (dans le pire des cas). Ce test est donc de complexité linéaire en fonction de la taille maximale d'un chemin (i.e. $\mathcal{O}(dim_{\mathcal{M}})$). Cependant, pour chaque nouvelle séquence minimale, il est réalisé autant de fois que l'on a déjà trouvé de séquence minimale. Or, dans le pire des cas, toutes les séquences sont minimales. Le test d'inclusion est donc réalisée au maximum $(dim_{\mathcal{M}} - 1)!$ fois par nouvelle séquence trouvée.

6. Ici, la notation A^k_n ne désigne pas une matrice de taux de transitions de chaîne de Markov, mais le nombre d'arrangements de k éléments parmi n.

Finalement la complexité de l'algorithme est majorée par le carré de la factorielle des dimensions du modèle $\mathcal{M}$:

$$\mathcal{O}(dim_{\mathcal{M}}(dim_{\mathcal{M}}-1)!(dim_{\mathcal{M}}-1)!) < \mathcal{O}(dim_{\mathcal{M}}!^2)$$

La grande complexité algorithmique de l'algorithme pourrait faire douter de l'efficacité de cette méthode d'analyse. Cependant, l'ensemble des SCM est construit "à la volée" par ordre croissant de longueur. Or, il est raisonnable de considérer qu'au sens de l'analyse qualitative, l'ordre croissant de longueur des séquences correspond à l'ordre décroissant d'intérêt pour l'expert de SdF. Cette limitation calculatoire n'est donc pas dramatique pour la pertinence de cette approche car, en pratique, l'expert n'aura pas intérêt à laisser le calcul se poursuivre au delà des séquences de faible longueur (moins de dix événements). La validité de cette affirmation sera appuyée par les traitements de cas qui seront réalisés dans la sous-section suivante et dans le chapitre 4.

3.2.3 Etude comparative

En vue de justifier par un exemple l'intérêt de la modélisation GBDMP du point de vue de l'analyse qualitative, nous allons reprendre le cas d'étude modélisé et analysé (qualitativement) dans [Chaux et al., 2012]. Le système étudié a pour fonction de fournir un débit d'eau sous pression à un générateur de vapeur (GV) de centrale nucléaire à eau pressurisée. Un schéma descriptif est reporté sur la Figure 3.6.

La partie directement en contact avec la matière d'œuvre est décomposée en deux sous-systèmes de pompes redondées. Un premier groupe de pompes doit assurer l'extraction d'un débit d'eau suffisant d'un condensateur (hors cadre du système). Ce débit d'eau doit ensuite être mis sous pression par le second groupe de pompes. Les pompes d'extraction ($Ex1$,$Ex2$ et $Ex3$) sont en redondance passive 2 parmi 3 (redondance R_A), et les pompes de pressurisation ($Pr1$ et $Pr2$) sont en redondance passive 1 parmi 2 (redondance R_B).

De plus, les pompes d'extraction sont alimentées électriquement par un sous-système hautement redondé. Les pompes peuvent être alimentées par deux lignes électriques en redondance passive (redondance R_C). Chaque ligne est alimentée par un tableau électrique ($TEa1$ et $TEa2$). Deux sources en redondance passive (redondance R_D) permettent d'apporter un flux électrique à ces tableaux : le réseau ($Grid$) ou un moteur

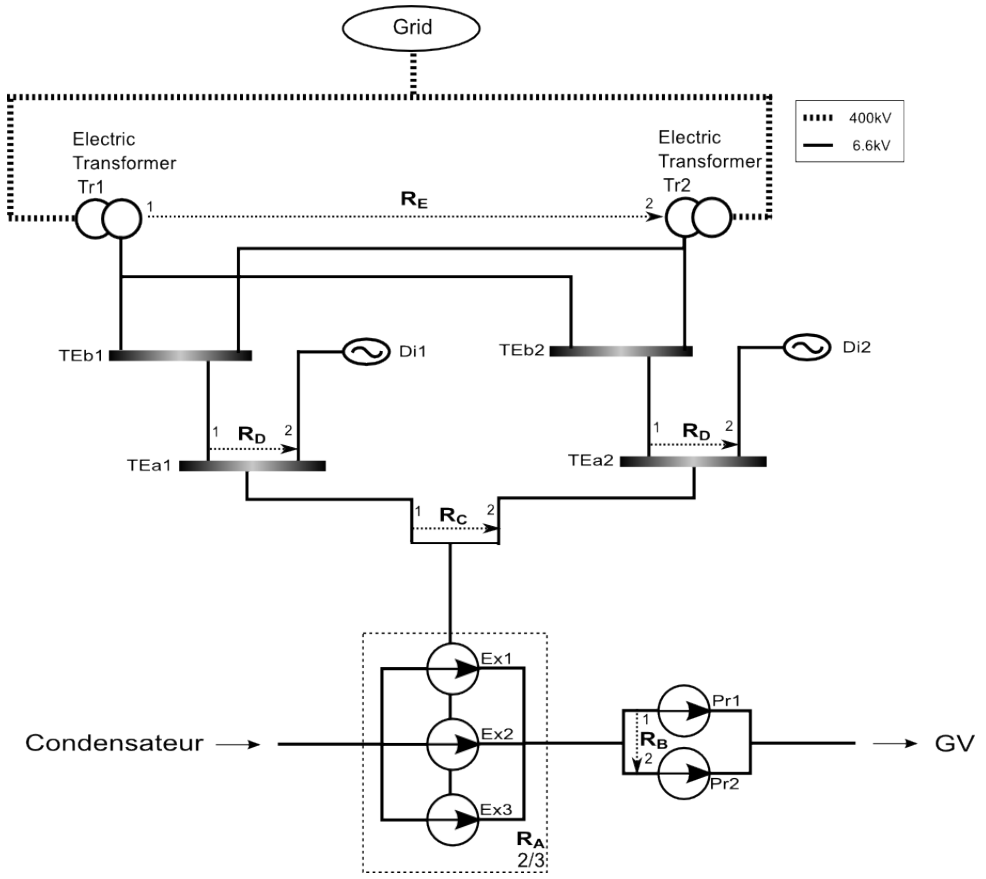

Figure 3.6 – Système hydraulique, représenté avec son alimentation électrique

diesel ($Di1$ et $Di2$, connectés respectivement à $TEa1$ et $TEa2$). Enfin, deux transformateurs ($Tr1$ et $Tr2$) en redondance passive (redondance R_E) débitent un courant à une tension adéquate vers les deux lignes, via deux nouveaux tableaux électriques ($TEb1$ et $TEb2$ connectés respectivement à $TEa1$ et $TEa2$).

Le modèle BDMP $\mathcal{M}_1$ de ce système proposé dans [Chaux et al., 2012] est reporté sur la figure 3.7. Pour construire ce modèle, les auteurs de la publication ont été contraints de respecter les deux hypothèses imposées par les BDMP (mentionnées dans la sous-section 2.2.3) :

- le seul mécanisme de commutation possible est celui traduit par une gâchette de BDMP : *l'activation et la désactivation de la destination sont respectivement corrélées à la défaillance et la réparation de l'origine. L'origine, elle, reste toujours active* ;
- son succès est garanti.

Afin de montrer que ces choix de modélisation imposés par le formalisme BDMP ont

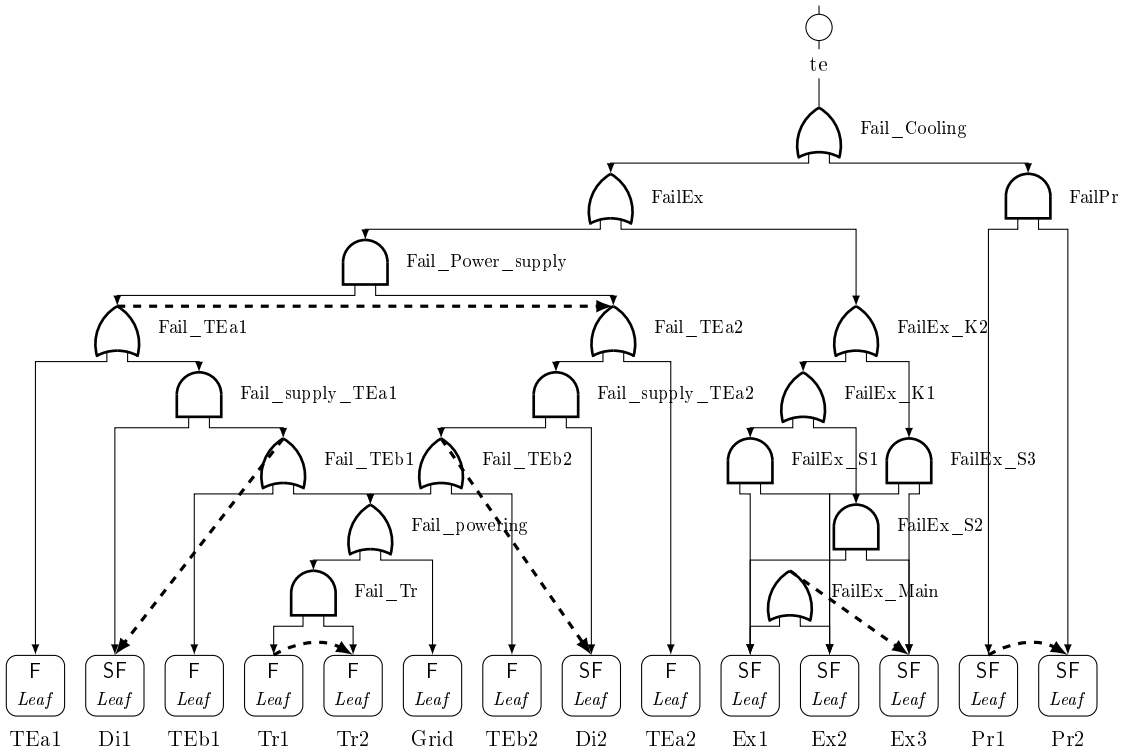

Figure 3.7 – Modèle BDMP $\mathcal{M}_1$ du système hydraulique et de son alimentation électrique ([Chaux et al., 2012])

un impact sur les résultats de l'analyse qualitative, nous allons enrichir le modèle grâce aux primitives introduites avec les GBDMP, et comparer les SCM obtenues à partir des deux modèles.

Nous considérons à présent que le succès des commutations introduites par une redondance passive est dépendant d'un composant fictif modélisant la perte du moyen physique de pilotage de cette redondance. Les feuilles R_A, R_B, R_C, R_D et R_E (en gris sur la Figure 3.8) représentent le comportement dysfonctionnel de ces composants fictifs. Pour simplifier, nous modéliserons ces comportements à l'aide de PMC de type F. De plus, on décide arbitrairement que la stratégie de reconfiguration associée à ces redondances est du type "au plus tard" (cf. sous-section 2.4.2.1). Le modèle GBDMP $\mathcal{M}_1$ résultant de ces deux modifications est représenté sur la figure 3.8. Les machines de Moore auxquelles fait référence le modèle sont représentées sur la Figure 3.9.

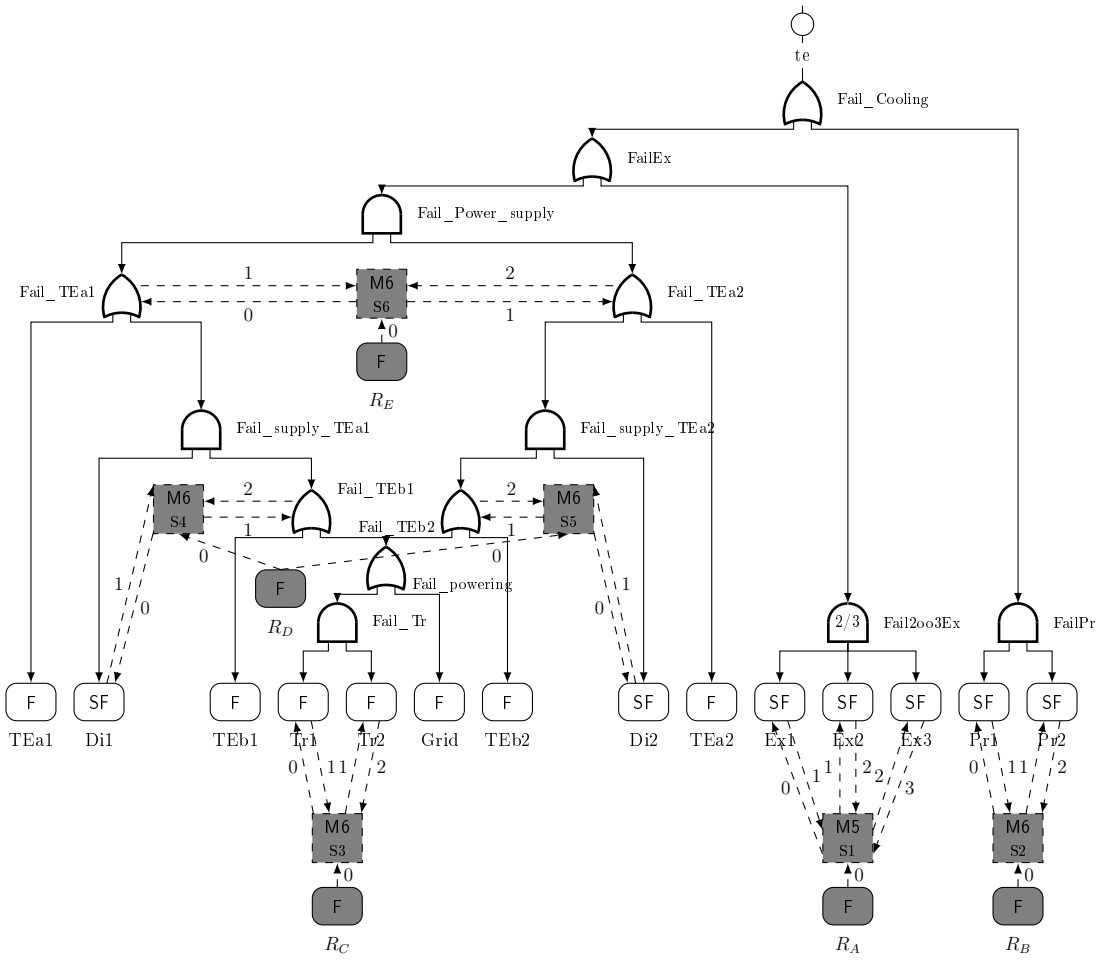

Figure 3.8 – Modèle GBDMP $\mathcal{M}_2$ du système hydraulique et de son alimentation électrique

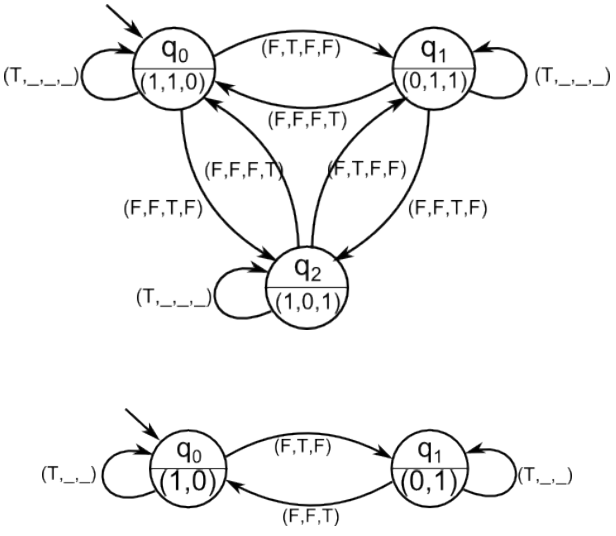

Figure 3.9 – Machines de Moore spécifiant des stratégies de reconfiguration "au plus tard", au succès non garanti, pour une redondance 2 parmi 3 : M5 (en haut), et 1 parmi 2 : M6 (en bas).

Afin d'alléger l'écriture des séquences dans la suite, étant donné que toutes les feuilles sont associées à un PMC de type SF et F[7], on notera :

- f_a^l, l'événement $e^l_{W\to F_2}$ (défaillance en mode actif de la feuille l) ;
- f_p^l, l'événement $e^l_{S\to F_1}$ (défaillance en mode passif de la feuille l) ;
- r_a^l, l'événement $e^l_{F_2\to W}$ (réparation en mode actif de la feuille l) ;
- r_p^l, l'événement $e^l_{F_1\to S}$ (réparation en mode passif de la feuille l).

L'ensemble des SCM extrait de $\mathcal{M}_1$, et en considérant la relation d'ordre $\subseteq$ (cf. [Chaux et al., 2012]), est différent de celui calculé par l'algorithme 2 appliqué à $\mathcal{M}_2$ ci-dessus, et en considérant la relation d'ordre $\Subset$. Les séquences qui ne sont pas dans l'intersection des deux ensembles s'expliquent par au moins une des trois raisons suivantes :

1. *La relation d'ordre $\Subset$ est moins restrictive que $\subseteq$ pour le calcul des SCM.*
   Par exemple, $f_a^{Grid} f_a^{Di1} r_p^{Grid} f_a^{TEa2} f_a^{TEb1} \in \mathcal{L}^{\Subset}_{SCM}(\mathcal{M}_2) \backslash \mathcal{L}^{\subseteq}_{SCM}(\mathcal{M}_1)$.
   En effet, $f_a^{Grid} f_a^{Di1} f_a^{TEa2} \subseteq f_a^{Grid} f_a^{Di1} r_p^{Grid} f_a^{TEa2} f_a^{TEb1}$
   mais $f_a^{Grid} f_a^{Di1} f_a^{TEa2} \notin f_a^{Grid} f_a^{Di1} r_p^{Grid} f_a^{TEa2} f_a^{TEb1}$
   car $FC(f_a^{Grid} f_a^{Di1} f_a^{TEa2}) = \{Grid, Di1, TEa2\}$
   et $FC(f_a^{Grid} f_a^{Di1} r_p^{Grid} f_a^{TEa2} f_a^{TEb1}) = \{Di1, TEa2, TEb1\}$

2. *La prise en compte de l'échec des commutations implique de nouveaux comportements dysfonctionnels.*
   Par exemple, les nouvelles SCM $f_a^{R_B} f_a^{Pr1}$ et $f_a^{Pr1} f_a^{R_B} r_p^{Pr1} f_a^{Pr2}$ sont symptomatiques de l'échec respectivement du *remplacement* et du *rétablissement* de $Pr1$.

3. *La modification de la stratégie de commutation implique des comportements dysfonctionnels différents.*

   - Par exemple, $f_a^{Ex1} r_p^{Ex1} f_a^{Ex3} f_a^{Ex2} \in \mathcal{L}^{\Subset}_{SCM}(\mathcal{M}_2) \backslash \mathcal{L}^{\subseteq}_{SCM}(\mathcal{M}_1)$. Cette séquence traduit le fait que, selon le modèle $\mathcal{M}_2$, le service n'a pas été rétabli sur $Ex1$ suite à sa réparation. Il n'y a alors que dans cette situation que le système peut défaillir suite aux défaillances en mode actif de $Ex3$ et $Ex2$, survenues

7. On rappelle que ces types de PMC ne modélisent pas de défaillance à la sollicitation. Aussi, les séquences ne contiendront que des événements *spontanés*.

dans cet ordre. Les seuls SCM de longueur inférieur conduisant à la défaillance de $Ex2$ et $Ex3$ sont $f_p^{Ex3} f_a^{Ex2}$ et $f_a^{Ex2} f_a^{Ex3}$, lesquels ne sont pas inclues dans la séquence en question. Par ailleurs, selon le modèle $\mathcal{M}_1$, le service est rétabli sur $Ex1$ suite à sa réparation : le composant $Ex3$ est désactivé et ne peut alors pas défaillir en mode actif. La séquence en question n'appartient donc pas au langage d'évolution du modèle $\mathcal{M}_1$.

- En revanche, $f_a^{TEa1} f_a^{TEb2} r_p^{TEa1} \boldsymbol{f_p^{Di2}} f_a^{TEa1} \in \mathcal{L}_{SCM}^{\subseteq}(\mathcal{M}_1) \backslash \mathcal{L}_{SCM}^{\Subset}(\mathcal{M}_2)$ car, selon le modèle $\mathcal{M}_1$, le rétablissement de l'alimentation électrique sur la ligne 1 suite à la réparation de $TEa1$ est immédiat, donc $Di2$ ne peut défaillir qu'en mode **passif**. Selon le modèle $\mathcal{M}_2$, le rétablissement n'est pas immédiat donc $Di2$ ne peut défaillir qu'en mode **actif**, ce qui donnerait la séquence $f_a^{TEa1} f_a^{TEb2} r_p^{TEa1} \boldsymbol{f_a^{Di2}} f_a^{TEa1}$, qui n'est pas minimale car $f_a^{TEa1} f_a^{TEb2} f_a^{Di2} \Subset f_a^{TEa1} f_a^{TEb2} r_p^{TEa1} f_a^{Di2} f_a^{TEa1}$.

On peut remarquer que les apparitions ou disparitions de certaines SCM sont dues à plusieurs raisons. Par exemple l'apparition de la séquence $f_a^{Pr1} r_p^{Pr1} f_a^{R_B} f_a^{Pr2}$ s'explique par la combinaison des trois raisons ci-dessus.

Une sélection de l'ensemble $\mathcal{L}_{SCM}^{\Subset}(\mathcal{M}_2)$ de longueur inférieure ou égale à 6 est reportée dans le tableau 3.1 (les nouvelles séquences sont indiquées en gras, et le nombre de SCM par longueur est indiqué en détaillant les différences avec les résultats obtenues dans [Chaux et al., 2012]). A titre d'indication, sur une machine de travail standard (fréquence du processeur : $2,9\ GHz$), il n'a fallu qu'une minute pour générer les 200 premières séquences, puis deux minutes de plus pour générer les 100 suivantes. La dernière (1944ème) a quant à elle été trouvée au bout d'environ 45 minutes. Par ailleurs, toutes les séquences de longueur inférieure ou égale à 5 ont été explorées en moins de 5 minutes, tandis que l'exploration des séquences de longueur 6 a duré 50 minutes de plus.

nombre par longueur	sélection de séquences
Longueur 2 14 (9+**5**)	$f_a^{Ex1}\, f_a^{Ex2}$ $f_a^{TEa1}\, f_a^{TEa2}$ $\boldsymbol{f_a^{R_B}\, f_a^{Pr1}}$ $\boldsymbol{f_a^{R_D}\, f_a^{Grid}}$
Longueur 3 37 (19+**18**)	$f_d^{Di1}\, f_a^{Grid2}\, f_a^{Di2}$ $f_a^{TEa1}\, f_a^{TEb2}\, f_a^{Di2}$ $\boldsymbol{f_a^{R_E}\, f_a^{Grid}\, f_a^{Di1}}$ $\boldsymbol{f_a^{R_C}\, f_a^{R_D}\, f_a^{TEb1}}$ $\boldsymbol{f_a^{TEa1}\, f_a^{TEb2}\, f_a^{Di2}}$
Longueur 4 93 (39+**54**)	$f_a^{Tr1}\, f_p^{Di1}\, f_a^{Tr2}\, f_a^{TEa2}$ $\boldsymbol{f_a^{Ex1}\, r_p^{Ex1}\, f_a^{Ex3}\, f_a^{Ex2}}$ $\boldsymbol{f_a^{Pr1}\, f_a^{R_B}\, r_p^{Pr1}\, f_a^{Pr2}}$ $\boldsymbol{f_a^{Pr1}\, r_p^{Pr1}\, f_a^{R_B}\, f_a^{Pr2}}$
Longueur 5 240 (17-**7**+**230**)	$f_a^{Tr1}\, f_a^{TEb1}\, f_a^{Di1}\, f_p^{Di2}\, f_a^{Tr2}$ $\boldsymbol{f_a^{Grid}\, r_p^{Grid}\, f_a^{R_D}\, f_p^{Di2}\, f_a^{Di1}}$ $\boldsymbol{f_a^{Tr1}\, f_a^{R_E}\, f_a^{R_D}\, r_p^{Tr1}\, f_a^{Tr2}}$ $\boldsymbol{f_a^{Grid}\, f_a^{Di1}\, r_p^{Grid}\, f_a^{TEa2}\, f_a^{TEb1}}$
Longueur 6 1560 (45-**45**+**1560**)	$\boldsymbol{f_a^{R_E}\, f_a^{TEb1}\, f_a^{Di1}\, r_p^{TEb1}\, f_p^{Di21}\, f_a^{Tr1}}$ $\boldsymbol{f_a^{Grid}\, f_a^{Di1}\, r_p^{Grid}\, f_a^{TEa2}\, f_p^{R_E}\, f_a^{Tr1}}$ $\boldsymbol{f_a^{Tr1}\, f_a^{Tr2}\, f_a^{R_D}\, f_a^{Di1}\, r_p^{Tr1}\, f_a^{Di2}}$ $\boldsymbol{f_a^{TEa1}\, f_a^{Tr1}\, r_p^{TEa1}\, f_a^{Tr2}\, f_p^{Di1}\, f_a^{TEa2}}$

Tableau 3.1 – Extrait des résultats de l'analyse qualitative pour le système hydraulique

3.3 Analyse quantitative

Cette section présente une méthode d'analyse quantitative basée sur une description haut niveau d'un système critique - développée dans [Brameret et al., 2015] - et montre comment utiliser cette méthode à partir d'un modèle GBDMP.

3.3.1 Méthode de construction automatique de chaîne de Markov limitée

La difficulté de construire de très grandes chaînes de Markov et d'évaluer des indicateurs probabilistes de SdF à partir de celles-ci a été évoquée dans le premier chapitre. Afin de réduire cette difficulté, les auteurs de [Brameret et al., 2015] proposent de limiter la construction à une partie de la chaîne suffisamment petite pour permettre le calcul des indicateurs attendus et suffisamment représentative du comportement dysfonctionnel du système pour que l'erreur de calcul soit négligeable (cf. Figure 3.10).

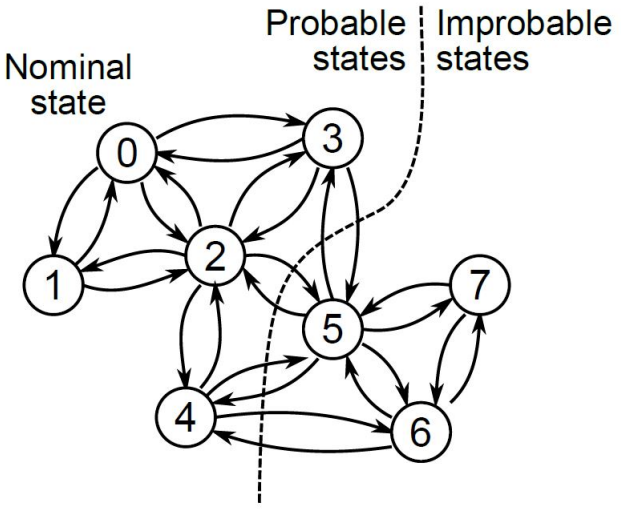

Figure 3.10 – Principe de la construction de Chaîne de Markov limitée [Brameret, 2015]

Afin de déterminer quelle partie de la chaîne doit être construite, les auteurs introduisent la notion de *facteur de pertinence probabiliste* d'un état. Ce facteur peut se comprendre intuitivement comme une mesure de la probabilité que l'état soit visité pendant un scénario. Il est donc basé sur la compétition entre les transitions sortantes d'un état. Etant donné une chaîne de Markov $\mathcal{Z} =< \mathcal{X}, A, p0 >$, si au temps t_0 cette chaîne est dans l'état $x_0 \in \mathcal{X}$, et qu'un événement survient au temps t_1, la probabilité pour que la chaîne arrive dans l'état $x \in \mathcal{X}$ suite à l'occurrence de cet événement est liée aux taux de probabilité des transitions sortantes de x :

$$\frac{A(x_0, x)}{\sum_{x' \in \mathcal{X}} A(x_0, x')}$$

Ainsi, le *facteur de pertinence probabiliste* d'un état se définit récursivement de la manière suivante :

$$R : \begin{array}{ccc} \mathcal{X} & \longrightarrow & [0,1] \\ x & \longmapsto & \max\left(p0(x), \max\limits_{x_0 \in \mathcal{X}}\left(R(x_0)\dfrac{A(x_0,x)}{\sum\limits_{x' \in \mathcal{X}} A(x_0,x')}\right)\right) \end{array} \tag{3.8}$$

Le principe de l'approche proposée dans [Brameret et al., 2015] est de construire l'espace d'état du système par ordre décroissant du *facteur de pertinence probabiliste* des états et d'arrêter l'exploration avant que la chaîne devienne trop grande pour être traitable. Les auteurs de la publication montrent comment utiliser l'algorithme de Dijkstra (cf. [Dijkstra, 1959]) pour procéder à cette construction ordonnée de la chaîne de Markov. En effet, cet algorithme permet de garantir l'exploration d'un graphe qui maximise le *facteur de pertinence probabiliste* des états.

Par ailleurs, afin de mesurer l'erreur produite par l'approche, les auteurs proposent d'ajouter à la chaîne un état "puits" (*sink* en anglais), pour agréger tous les états non explorés. La Figure 3.11 illustre comment cet état est construit : lorsque la construction est interrompue, toutes les transitions frontalières (i.e. les transitions sortant d'un état exploré et allant vers un état inexploré candidat) sont redirigées vers l'état puits. Selon l'interprétation que l'on donne à cet état, le calcul effectué sur la chaîne avec puits fournira une borne minimale ou maximale de l'indicateur recherché. Par exemple, considérer que le système modélisé est défaillant (resp. non défaillant) dans l'état puits, revient à faire l'hypothèse qu'il l'est dans tous les états non explorés. Ainsi, le calcul de la probabilité d'être dans un état défaillant de la chaîne partielle avec puits donne une borne maximale (resp. minimale) de l'indisponibilité du système.

Cette approche permet donc de calculer une approximation de l'indicateur de probabilité attendu (calcul sur la chaîne partielle sans état puits), ainsi que des bornes minimales et maximales de celui-ci (calculs sur la chaîne partielle avec états puits). La qualité des bornes dépend de la nature du système étudié et de la durée de l'étude [8]. Toutefois, les nombreuses expérimentations effectuées, dans le cadre de la thèse [Brameret, 2015], montrent qu'en moyenne, pour les systèmes constitués de composants fiables [9], un calcul effectué sur plus d'1% de la chaîne, et sur une durée d'étude compatible avec

8. En effet, puisqu'aucune transition ne sort d'un état puits, l'erreur grandit avec la durée de l'étude.
9. Un composant fiable a des taux de défaillance très inférieurs aux taux de réparation.

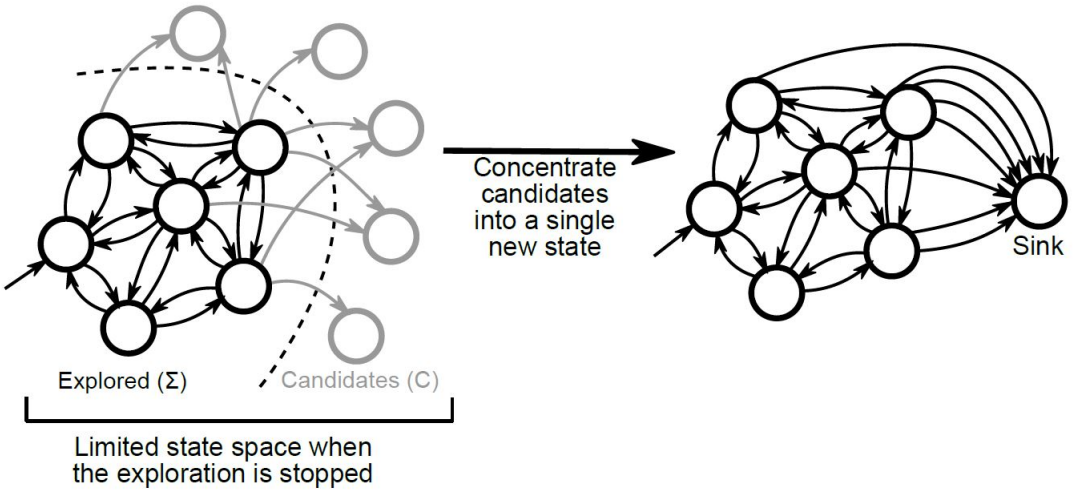

Figure 3.11 – Agrégation des états non explorés dans un état puits [Brameret et al., 2015]

les durées d'exploitation du système, permet d'obtenir une évaluation de l'indicateur recherché avec moins d'1% d'erreur. Cet important travail expérimental a donc permis de valider l'efficacité de la méthode, sur des cas représentatifs.

[Brameret et al., 2015] propose un algorithme pour utiliser cette méthode d'analyse quantitative à partir d'un modèle AltaRica. Par ailleurs, un outil a été développé dans le cadre de la thèse [Brameret, 2015], pour implémenter cet algorithme. Afin d'utiliser la méthode à partir d'un modèle GBDMP, nous avons donc deux possibilités : implémenter un nouvel algorithme pour l'adapter directement aux GBDMP ou définir un traducteur de modèle GBDMP en modèle AltaRica et utiliser l'outil existant. C'est la deuxième solution qui a été retenue dans le cadre de ce travail. En effet, la communauté utilisatrice du langage AltaRica et des outils dédiés est importante et tend à s'agrandir. Aussi, créer un lien entre la modélisation GBDMP et ce langage permet une meilleure diffusion de notre travail. De plus, les utilisateurs d'AltaRica pourront se servir de ce lien comme d'un outil méthodologique pour construire des modèles AltaRica, à partir d'un formalisme de modélisation haut niveau se représentant graphiquement.

3.3.2 Traduction d'un modèle GBDMP en un modèle AltaRica

Dans un premier temps, les bases du langage AltaRica sont rappelées, puis, le principe de la traduction d'un modèle GBDMP en modèle AltaRica est décrit.

3.3.2.1 Brève présentation du langage AltaRica

Nous limiterons cette présentation à la dernière version du langage : AltaRica 3.0 [10]. Ce langage est la combinaison du formalisme Guarded Transition System (GTS) décrit ci-dessous dans la définition 3.6 et du paradigme System Structure Modeling Language (S2ML). Ces deux facettes d'un modèle AltaRica traduisent respectivement le comportement et la structure du système. Le paradigme S2ML permet en particulier de composer des modèles GTS élémentaires à l'aide d'opérateurs tels que la *connexion* et la *synchronisation* (décrits ci-dessous), et ajoute des outils classiques de hiérarchisation tels que l'*agrégation*, l'*héritage* et le *référencement* (qui ne seront pas décrits dans le cadre de cette thèse).

Définition 3.6. *Un GTS est un 5-tuple* $< \mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{A}, i >$ *tel que :*

- $\mathcal{V} = \mathcal{V}_S \cup \mathcal{V}_F$ *est un ensemble de variables, partitionné entre les variables d'états* $\mathcal{V}_S$ *et les variables de flux* $\mathcal{V}_F$ *;*
- $\mathcal{E}$ *est un alphabet d'événements ;*
- $\mathcal{T}$ *est un ensemble de transitions, i.e. de 3-tuple* $< e, \mathcal{G}, \mathcal{P} >$ *tel que :*
  - $e \in \mathcal{E}$ *est un événement ;*
  - $\mathcal{G}$ *est une expression Booléenne construite sur* $\mathcal{V}$*, appelée la garde ;*
  - $\mathcal{P}$ *est une instruction appelée la post-condition, qui permet d'actualiser les variables d'états* $\mathcal{V}_S$ *;*
- $\mathcal{A}$ *est une instruction appelée l'assertion, qui permet d'actualiser les variables de flux* $\mathcal{V}_F$ *;*
- i *est une assignation "initiale" des variables d'états* $\mathcal{V}_S$*, et "par défaut" des variables de flux* $\mathcal{V}_F$*.*

La définition 3.6 fait référence à la notion d'*instruction* définie ci-dessous :

Définition 3.7. *L'ensemble des instructions d'un GTS est strictement décrit tel que :*

- "*skip*" *est une instruction (l'instruction vide) ;*

10. Une présentation plus détaillée de ce langage peut être consultée dans la thèse [Prosvirnova, 2014])

- *soit* $v \in \mathcal{V}$ *et* E *est une valeur ou une expression mathématique permettant d'en calculer une, "*$v := E$*" est une instruction ;*

- *soit* I *une instruction et* C *une expression Booléenne, "*$if\ C\ then\ I$*" est une instruction ;*

- *soit* I_1 *et* I_2 *deux instructions, la composition parallèle "*$\{I_1; I_2\}$*" est une instruction.*

Comme pour les GBDMP, un GTS décrit un espace d'état par intention. Cet espace peut être construit selon la dynamique suivante :

- l'assignation i décrit l'état initial ;

- si la garde $\mathcal{G}$ est vérifiée dans l'état s, alors la transition $< e, \mathcal{G}, \mathcal{P} >$ est franchissable depuis l'état s ;

- si la transition $< e, \mathcal{G}, \mathcal{P} >$ est franchie depuis l'état s, alors, dans un premier temps, les variables d'états $\mathcal{V}_S$ sont actualisées en appliquant l'instruction $\mathcal{P}$. Dans un second temps, les variables de flux $\mathcal{V}_F$ sont réinitialisées à leur valeur par défaut, puis actualisées en appliquant l'instruction $\mathcal{A}$. L'assignation résultante de ces actualisations décrit un nouvel état.

Deux modèles GTS peuvent être composés, soit par produit libre (si l'intersection des alphabets et de l'ensemble des variables est vide), soit par *connexion* (i.e. en ajoutant des instructions dans l'assertion pour décrire les interactions entre les variables de flux des deux modèles), soit par *synchronisation*. Plusieurs opérateurs de synchronisation ont été définis, mais nous n'en décrirons qu'un dans la Définition 3.8 (le seul qui sera utilisé pour la traduction d'un GBDMP en GTS).

Définition 3.8. *Soit* $< e_1, \mathcal{G}_1, \mathcal{P}_1 >$ *et* $< e_2, \mathcal{G}_2, \mathcal{P}_2 >$ *deux transitions de GTS, l'événement de synchronisation* $?e_1 \& ?e_2$ *est associé à une transition* $<?e_1 \& ?e_2, \mathcal{G}, \mathcal{P} >$ *telle que :*

- $\mathcal{G} = \mathcal{G}_1 \vee \mathcal{G}_2$

- $\mathcal{P} = \{if\ \mathcal{G}_1\ then\ \mathcal{P}_1; if\ \mathcal{G}_2\ then\ \mathcal{P}_2\}$

Enfin, le formalisme de GTS peut être étendu à celui de GTS Temporisé et Stochastique (TSGTS) grâce à l'ajout de délais et de données probabilistes aux événements du modèle. Dans ce cas, lors de la définition d'un événement e, on doit renseigner la valeur $delay(e)$ qui correspond à un délai fixe (potentiellement nul) ou à une distribution de probabilités permettant d'en déterminer un aléatoirement. Dans le cas où le délai est fixe, on doit également indiquer la valeur de $expectation(e)$ qui correspond à la probabilité que e survienne s'il est en concurrence avec plusieurs événements devant survenir à la même date.

Le paradigme S2ML introduit les notions de *classes* pour désigner les briques génériques, vouées à être instanciées dans le modèle, et de *blocs* pour désigner les constructions résultant de la composition et de la hiérarchisation des instances de ces classes. Ces notions sont respectivement issues des paradigmes de programmation *orientée objet* (cf. [Abadi and Cardelli, 1996]) et *orientée prototype* (cf. [Noble et al., 1999]). D'un point de vue méthodologique, pour la modélisation d'un système, les *classes* servent à modéliser le comportement élémentaire des composants, et les *blocs* servent à modéliser la structure du système et les interactions entre les composants. La construction d'un modèle AltaRica consiste donc à décrire un TSGTS au sein de *classes* et de *blocs*.

3.3.2.2 Principe de la traduction

Dans cette sous-section, nous ne présenterons que le principe de la traduction, car la définition d'un traducteur formel impliquerait de rappeler la syntaxe du langage AltaRica, ce qui n'est pas l'objet du présent document. L'annexe D propose un exemple de traduction d'un modèle GBDMP simple dans le langage AltaRica.

a) Construction des classes

La première étape de la traduction consiste à construire les classes de base décrivant les primitives du formalisme GBDMP.

La classe *Nœud* est décrite par 2 variables de flux entières M et R, dont la valeur par défaut est 1, et une variable de flux Booléenne F dont la valeur par défaut est $False$ (ces trois variables correspondent aux trois statuss caractéristiques d'un nœud de GBDMP).

Pour chaque k-PMC $P =< (\mathcal{Z}_i^P)_{0\leq i<k}, \mathcal{X}_F^P, (f_{i\to j}^P)_{(i,j)\in[\![0,k-1]\!]^2} >$ référencé dans le modèle GBDMP (i.e. $\exists l \in L | pmc(l) = P$), on décrit une nouvelle classe à l'aide de :

- une variable d'état X, prenant ses valeurs dans $\mathcal{X}^P \cup \{None\}$ et initialisée à $None$

(avant l'initialisation, on ne sait pas dans quel mode le PMC doit être) ;

- deux variables de flux $isFaulty$ et $shouldBeInMode$ (resp. Booléenne et entière), dont les valeurs par défaut peuvent être choisies arbitrairement ;

- des événements et transitions de trois types :

  - d'initialisation : $\forall x \in \mathcal{X}^P | \exists i \in [\![0, k-1]\!], p0_i^P(x) \neq 0$

    $$\begin{cases} init\_x \in \mathcal{E} \text{ avec } delay(init\_x) = 0 \text{ et } expectation(init\_x) = p0_i^P(x) \\ < init\_x,\ (X = None) \wedge (shouldBeInMode = i),\ X := x > \in \mathcal{T} \end{cases}$$

  - provoqués : $\forall (x,y) \in \left(\mathcal{X}^P\right)^2 | \exists (i,j) \in ([\![0, k-1]\!])^2, i \neq j \wedge f_{i\to j}^P(x,y) \neq 0$

    $$\begin{cases} prov\_x\_to\_y \in \mathcal{E} \text{ avec } delay(prov\_x\_to\_y) = 0 \\ \qquad\qquad\qquad\qquad \text{et } expectation(prov\_x\_to\_y) = f_{i\to j}^P(x,y) \\ < prov\_x\_to\_y,\ (X = x) \wedge (shouldBeInMode = j),\ X := y > \in \mathcal{T} \end{cases}$$

  - spontanés : $\forall (x,y) \in \left(\mathcal{X}^P\right)^2 | \exists i \in [\![0, k-1]\!], A_i^P(x,y) \neq 0$

    $$\begin{cases} spont\_x\_to\_y \in \mathcal{E} \text{ avec } delay(spont\_x\_to\_y) = exp(A_i^P(x,y)) \\ < spont\_x\_to\_y,\ (X = x) \wedge (shouldBeInMode = i),\ X := y > \in \mathcal{T} \end{cases}$$

- une unique instruction dans l'assertion :

$$isFaulty := \bigvee_{x \in \mathcal{X}_F^P} (X = x)$$

Pour chaque machine de Moore $M =< Q^M, q_0^M, \Sigma_I^M, \Sigma_O^M, trans^M, out^M >$ référencée dans le modèle GBDMP (i.e. $\exists s \in S | str(s) = M$), on décrit une nouvelle classe à l'aide de :

- une variable d'état U, prenant ses valeurs dans Q^M et initialisée à q_0^M ;

- une variable de flux $nextU$, prenant ses valeurs dans Q^M et dont la valeur par défaut est q_0^M ;

- des variables de flux $I_0, ..., I_a$ et $O_0, ..., O_b$ dont le nombre $(a + b + 2)$ et les types sont à déterminer selon la nature de Σ_I^M et Σ_O^M. Les valeurs par défaut de ces variables peuvent être choisies arbitrairement.

- un événement *update* avec $delay(update) = 0$ et $expectation(update) = 1$;
- une transition $< update,\ U \neq nextU,\ U := nextU >$;
- une assertion composée à l'aide des instructions suivantes :
  - $\forall (q, i, q') \in Q^M \times \Sigma_I^M \times Q^M | i = (i_0, ..., i_a) \wedge q' = trans^M(q, i)$:

$$if\ \ (U = q) \wedge \bigwedge_{k=0}^{a} (I_k = i_k)\ then\ nextU := q'$$

  - $\forall (q, o) \in Q^M \times \Sigma_I^M | o = (o_0, ..., o_b) \wedge o = out^M(q)$:

$$if\ nextU = q\ then\ \{O_0 := o_0; ...; O_b := o_b\}$$

b) Construction du bloc

Ces classes sont ensuite instanciées dans un bloc global. C'est également à cette étape que l'on connecte les instances entre elles, selon la structure du modèle GBDMP, en ajoutant des instructions dans l'assertion du modèle GTS sous-jacent. Le processus de construction de ce bloc est décrit par l'algorithme 3.

c) Synchronisation des transitions immédiates

Finalement, afin de respecter strictement la dynamique des GBDMP, il reste à synchroniser les transitions immédiates. En effet, suite à l'occurrence d'un événement *spontané*, si plusieurs feuilles doivent être commutées, les variables de flux ne sont pas sensées être ré-évaluées entre chaque occurrence d'événements *provoqués* permettant ces commutations. Les transitions correspondantes doivent donc être synchronisées. De même, les transitions d'actualisation des variables d'état des machines de Moore doivent être synchronisées avec les transitions *provoqués* pour éviter des évolutions imprévues du type :

1. occurrence d'un événement *spontané* $\Longrightarrow$ actualisation des variables $\Longrightarrow$ des feuilles doivent être commutées et l'état de certaines machines de Moore doit être actualisé ;
2. choix arbitraire de franchir en premier la transition permettant d'actualiser les feuilles $\Longrightarrow$ les variables sont actualisées en prenant en compte des variables d'état de machines de Moore qui ne sont pas à jour $\Longrightarrow$ conséquences non envisagées.

Algorithme 3 Construction d'un bloc AltaRica correspondant à un modèle GBDMP

Inputs:
- $< V, E, \kappa, \upsilon, str, pmc >$ un modèle GBDMP bien formé ;
- la classe *Noeud*, ainsi que les classes traduisant chaque k-PMC et machines de Moore référencés dans le modèle.

Outputs: le *bloc* AltaRica correspondant au modèle GBDMP.

1: *// Instanciation des classes*
2: **for all** $n \in N$ **do**
3: instancier la classe *Noeud*
4: **if** $n \in L$ **then**
5: instancier la classe correspondante à $pmc(n)$
6: **end if**
7: **end for**
8: **for all** $s \in S$ **do**
9: instancier la classe correspondante à $str(s)$
10: **end for**
11: *// Connexion des instances*
12: **for all** $n \in N$ **do**
13: *// Calcul des statuss de défaillance (cf. formule 2.1 et 2.2)*
14: **if** $n \in L$ **then**
15: $A := A \cup \{n.F := pmc(n).isFaulty\}$
16: **else**
17: $A := A \cup \{if\ Card(\{n' \in \Gamma^+_{\mathcal{G}_F}(n) | n'.F \vee \neg n'.R\}) \geq \kappa(n)\ then\ n.F := True\}$
18: **end if**
19: *// Calcul des statuss de réquisition (cf. formule 2.3)*
20: **if** $\exists s \in S | (s, n) \in E_S$ **then**
21: $A := A \cup \{n.R := s.O_{\upsilon(s,n)}\}$
22: **end if**
23: *// Calcul des statuts d'activité (cf. formule 2.4)*
24: **if** $\Gamma^-_{\mathcal{G}_F}(n) = \varnothing$ **then**
25: $A := A \cup \{n.M := n.R\}$
26: **else**
27: $A := A \cup \{n.M := n.R * \max_{g \in \Gamma^-_{\mathcal{G}_F}(n)} (M_g.\upsilon((g, n)))\}$
28: **end if**
29: *// Actualisation du mode des PMC*
30: **if** $n \in L$ **then**
31: $A := A \cup \{pmc(n).shouldBeInMode := n.M\}$
32: **end if**
33: *// Calcul des entrées des machines de Moore*
34: **if** $\exists s \in S | (n, s) \in E_S$ **then**
35: **if** $s.I_{\upsilon(n,s)} \in \{False, True\}$ **then**
36: $A := A \cup \{s.I_{\upsilon(n,s)} := n.F\}$
37: **else**
38: $A := A \cup \{s.I_{\upsilon(n,s)} := pmc(n).X\}$
39: **end if**
40: **end if**
41: **end for**

Lorsque la commutation d'une feuille n'est pas déterministe (en particulier en cas de défaillances à la commutation), cette étape constitue la principale difficulté de la traduction. En effet dans ce cas, on ne peut pas se contenter de synchroniser toutes les transitions immédiates du modèle à l'aide de l'opérateur introduit dans la sous-section 3.3.2.1, car certaines de ces transitions sont mutuellement exclusives (en particulier, la commutation normale et l'échec à la commutation d'un composant ne peuvent pas être synchronisés). Aussi, l'unique solution est de créer autant de transitions de synchronisation qu'il y a de combinaisons maximales de transitions non mutuellement exclusives deux à deux. En pratique, cette solution n'est viable que si le nombre de composants pouvant défaillir à la sollicitation est faible. Un exemple de telles synchronisations est présenté dans l'annexe D.

3.3.3 Etude comparative

Nous allons à présent valider l'intérêt de la modélisation GBDMP pour les analyses quantitatives. L'objectif est de montrer que les différentes stratégies de reconfiguration et leurs possibles échecs ont un impact sensible sur les indicateurs probabilistes de SdF d'un système reconfigurable. Nous allons donc choisir un système reconfigurable et faire varier les stratégies de reconfiguration, ainsi que la prise en compte ou non de leurs échecs. Puis, nous effectuerons des calculs de disponibilité basés sur ces différentes variantes et confronterons les résultats.

Pour cette étude, nous considérons le système reconfigurable le plus simple possible : une redondance passive entre deux composants. En effet, si on arrive à identifier un impact de la modélisation GBDMP sur les résultats d'analyse pour un exemple aussi petit que celui-ci, on pourra raisonnablement admettre que cet impact s'observera également sur un exemple de taille plus importante. Aussi, les chaînes de Markov pourront être générées entièrement, et les résultats obtenus seront exacts.

Dans la suite, on choisira arbitrairement des valeurs numériques pour les données d'entrée :

$$\left|\begin{array}{rcl} \lambda & = & 0.001\ h^{-1} \\ \lambda_S & = & 0.0005\ h^{-1} \\ \mu & = & 0.1\ h^{-1} \\ \gamma & = & 0.2 \end{array}\right.$$

La Figure 3.12 représente la première version $\mathcal{M}_1$ du modèle GBDMP pour cet exemple simple. Les feuilles modélisent deux composants à deux modes d'opération pouvant défaillir à la sollicitation (ce comportement est modélisé par des 2-PMC de type SF_I). La stratégie de reconfiguration entre les deux composants est celle d'une gâchette de BDMP, de telle sorte que ce modèle soit équivalent à un modèle BDMP.

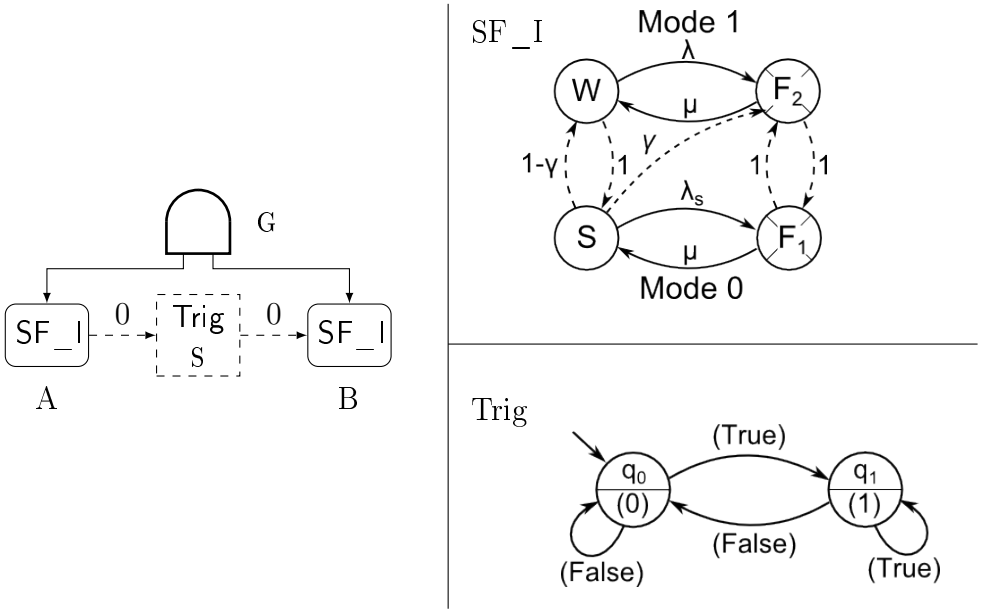

Figure 3.12 – Première version du modèle : $\mathcal{M}_0$

Dans la seconde version du modèle (cf. Figure 3.13), on ajoute un troisième composant responsable de l'application de la stratégie de reconfiguration. Le modèle $\mathcal{M}_1$ n'a pas d'équivalent BDMP.

Le graphique reporté sur la Figure 3.14 trace les courbes d'indisponibilité du système en fonction du temps écoulé depuis le début d'une mission, calculées à partir des modèles $\mathcal{M}_0$ et $\mathcal{M}_1$. Ce résultat montre bien l'impact attendu de la prise en compte dans le modèle de l'échec du mécanisme de commutation sur l'évaluation de la disponibilité du système. Le calcul de la disponibilité du système mené sur le modèle $\mathcal{M}_1$ est plus pessimiste que celui mené sur le modèle $\mathcal{M}_0$, mais également plus réaliste.

A présent nous introduisons deux nouvelles versions du modèle. Rien ne change ni dans la structure, ni dans les comportements dysfonctionnels des composants. La seule modification apportée concerne la stratégie de reconfiguration entre les deux composants :

- dans le modèle $\mathcal{M}_2$, on remplace la stratégie de la gâchette par une stratégie de

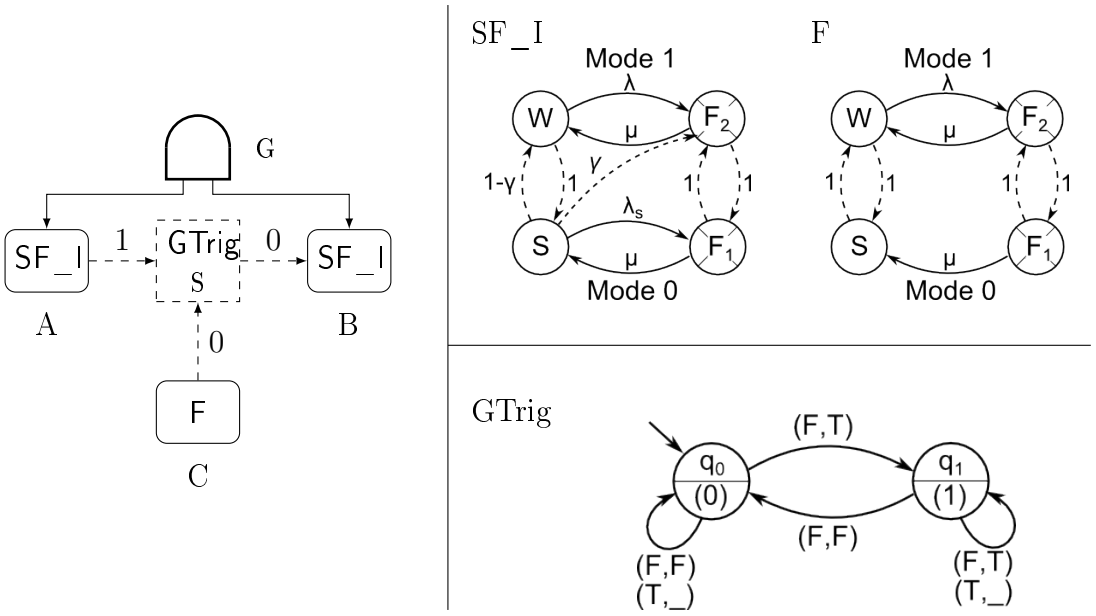

Figure 3.13 – Deuxième version du modèle : $\mathcal{M}_1$

commutation "au plus tôt". Cela signifie qu'on ne se contente plus d'activer le composant B tant que A est défaillant, on désactive aussi le composant A. On rappelle que ces commutations sont réalisées sous réserve que le composant C est non défaillant.

- dans le modèle $\mathcal{M}_3$, on remplace cette stratégie par une stratégie de commutation "au plus tard". Cela signifie que l'on ne privilégie plus l'activation de A dès qu'il est disponible. Les conditions de commutation de A vers B et de B vers A sont symétriques : le composant actif doit être défaillant, tandis que l'autre ne doit pas l'être. De nouveau, ces commutations sont réalisées sous réserve que le composant C est non défaillant.

Ces deux stratégies de reconfiguration sont respectivement spécifiées par les machines de Moore représentées sur les Figure 3.15 et 3.16.

Le graphique reporté sur la Figure 3.17 permet de confronter les trois stratégies de reconfiguration. Il apparaît clairement que, pour ce système, la stratégie de la gâchette offre une meilleure disponibilité que les deux autres. Cela s'explique par l'absence de risque de défaillance à la commutation du composant A, selon cette stratégie [11]. A l'inverse, pour les deux autres stratégies, le risque de défaillance à la commutation de ce composant est non nul, ce qui augmente l'indisponibilité du système. D'autre part, les

11. On rappelle que l'origine d'une gâchette n'est jamais désactivée puis réactivée.

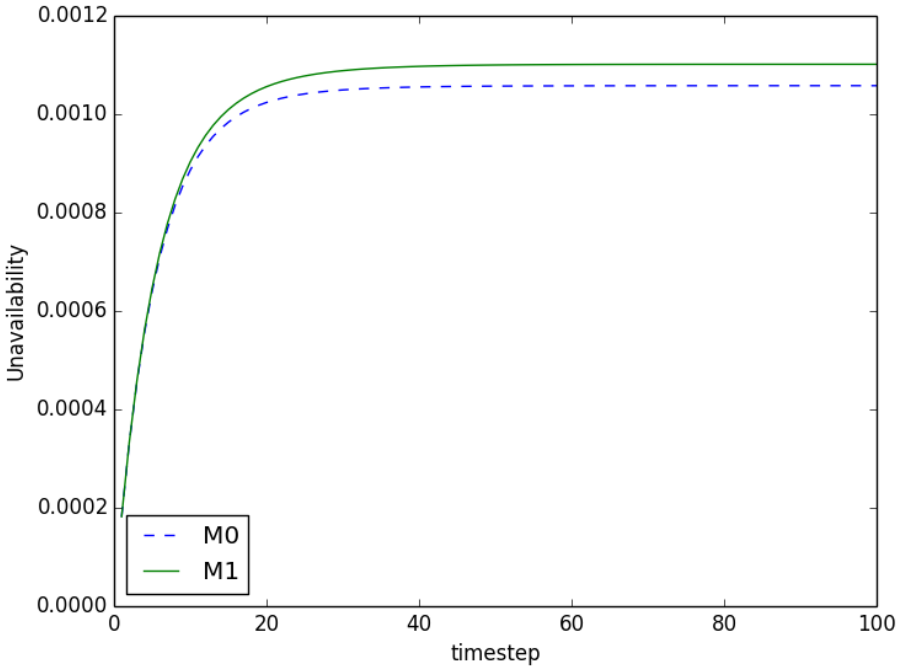

Figure 3.14 – Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}_0$ et $\mathcal{M}_1$

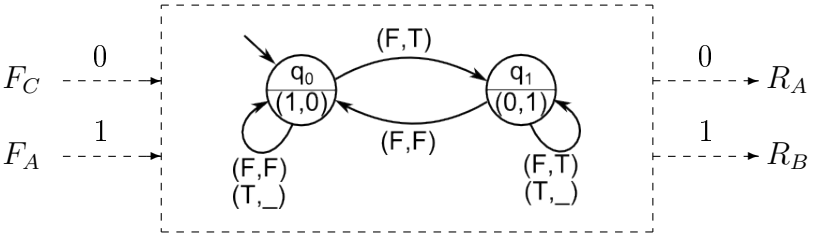

Figure 3.15 – Machine de Moore caractérisant la troisième version du modèle $\mathcal{M}_2$

courbes montrent que la stratégie "au plus tard" est meilleure que la stratégie "au plus tôt" en terme de disponibilité pour ce système. L'explication vient du fait que la stratégie "au plus tard" minimise le nombre de commutations. La stratégie "au plus tôt" en revanche, génère des commutations superflues pour la réalisation de la fonction, ce qui a pour effet d'augmenter le risque de défaillance à la sollicitation des composants A et B et donc l'indisponibilité du système. Il est important de noter que cette observation ne signifie pas que la stratégie "au plus tard" est intrinsèquement préférable à la stratégie "au plus tôt". En effet, privilégier l'utilisation du composant A peut se légitimer pour des raisons de performances autres que la disponibilité du système (par exemple si la puissance consommée par A est inférieure à celle consommée par B).

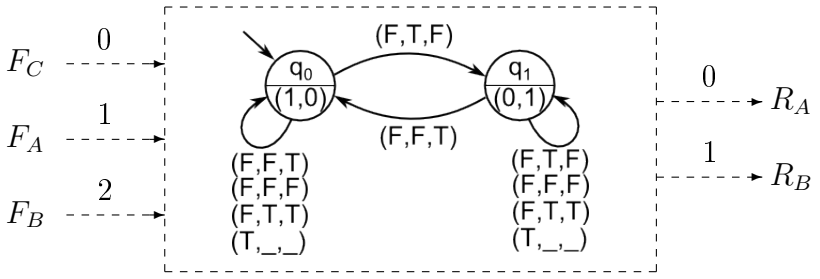

Figure 3.16 – Machine de Moore caractérisant la quatrième version du modèle $\mathcal{M}_3$

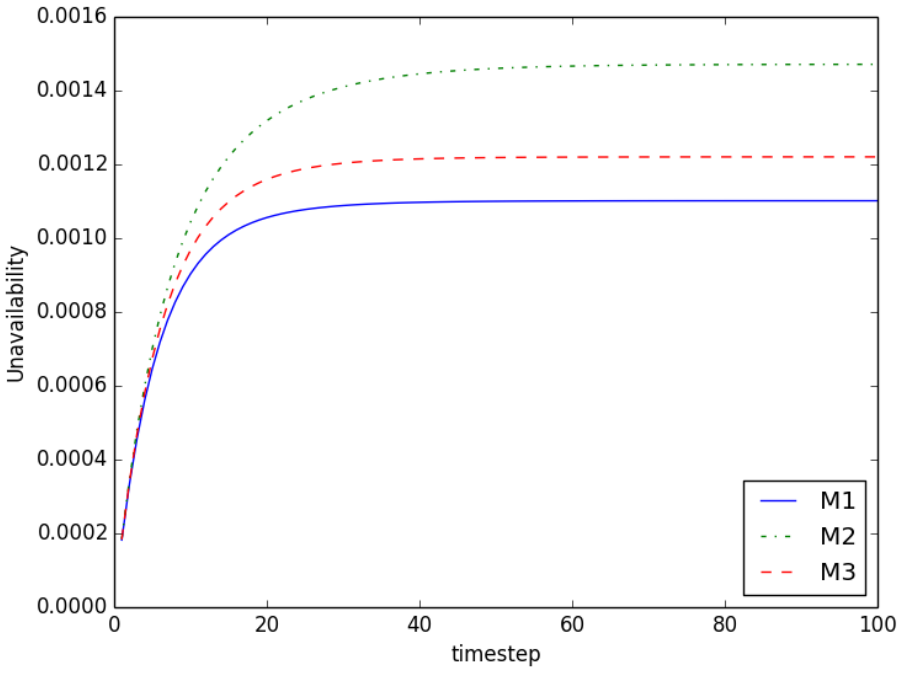

Figure 3.17 – Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}_1$, $\mathcal{M}_2$ et $\mathcal{M}_3$

Pour la dernière étude comparative, nous allons appliquer une modification mineure aux trois dernières versions du modèle. La modification porte sur le comportement dysfonctionnel des composants : le taux de réparation, alors que le composant est inactif, est doublé. Cette modification a une interprétation physique très réaliste : un composant a plus de chance d'être réparé s'il est inactif.

Les nouvelles versions du modèle seront désignés par $\mathcal{M}'_1$,$\mathcal{M}'_2$ et $\mathcal{M}'_3$. La Figure 3.18 montre que dans ce cas, la stratégie de la gâchette est devenue la moins bonne des trois. En effet, puisque dans les deux autres stratégies, le composant A est désactivé lorsqu'il est défaillant, il met deux fois moins de temps à être réparé que dans la stratégie de la gâchette, ce qui diminue l'indisponibilité du système.

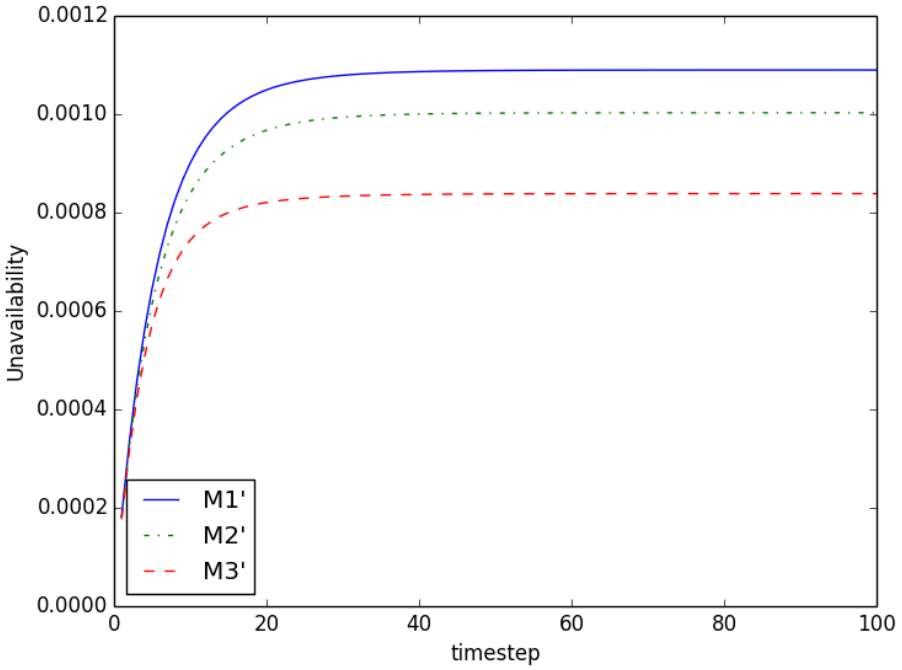

Figure 3.18 – Confrontation des résultats du calcul de disponibilité mené sur les modèles $\mathcal{M}'_1$, $\mathcal{M}'_2$ et $\mathcal{M}'_3$

Cette sous-section a permis de montrer l'intérêt de la modélisation GBDMP pour l'analyse quantitative de la SdF des systèmes dynamiques, réparables et reconfigurables. En effet, ces trois études comparatives n'auraient pas pu être menées dans le cadre d'une modélisation BDMP. C'est grâce au pouvoir d'expression des GBDMP que les différentes variantes du système ont pu être discriminées dans les modèles.

3.4 Bilan

Dans ce chapitre, deux techniques d'analyse de SdF basées sur un modèle GBDMP ont été présentées : une méthode d'analyse qualitative étendant la portée des travaux de [Chaux et al., 2013], et une méthode d'analyse quantitative reposant sur ceux de [Brameret et al., 2015].

Une nouvelle définition de la minimalité d'une séquence de coupe, formalisée dans la théorie des langages, a été proposée pour les systèmes dynamiques. Cette définition est suffisamment générale pour convenir aux systèmes dynamiques, réparables et reconfigurables.

En outre, un algorithme de recherche des SCM d'un système à partir d'un modèle

GBDMP de celui-ci a été proposé. L'exécution de cet algorithme implique la traduction du modèle en automate à états fini, laquelle a été formellement exposée.

Par ailleurs, le principe d'un traducteur de modèle GBDMP dans le langage AltaRica 3.0 a également été décrit. Cette traduction permet en particulier d'utiliser un outil développé dans le cadre de la thèse [Brameret, 2015], afin de réaliser des calculs d'indicateurs probabilistes de la SdF d'un système, à partir d'un modèle GBDMP de celui-ci.

Enfin, des études comparatives réalisées pour les deux types d'analyse ont permis de mettre en exergue l'intérêt de la modélisation GBDMP pour l'analyse de SdF des systèmes dynamiques, réparables et reconfigurables. En effet, on a montré que le pouvoir d'expression de ce formalisme permettait de constater, dans les résultats qualitatifs comme dans les résultats quantitatifs, l'impact des différentes stratégies de reconfiguration de ce type de système, ainsi que des possibles échecs de la réalisation de ces stratégies.

Chapitre

4

Application : partie d'une centrale nucléaire à eau pressurisée

Sommaire

4.1 Introduction 133

4.2 SAGE : un outil pour l'édition et la manipulation des modèles 133

4.3 Description du processus de conception 136

4.4 Description du cas 139

4.4.1 Le procédé physique 140

4.4.2 Le contrôle-commande 143

4.4.3 Analyse préliminaire des risques 146

4.4.4 Maintenance des échangeurs 147

4.5 Modélisation GBDMP du système 148

4.5.1 Construction des PMC 148

4.5.2 Modélisation de l'architecture opérationnelle de CC 150

4.5.3 Construction de la structure du GBDMP pour la partie procédé 151

4.5.4 Construction des machines de Moore 152

4.5.4.1 Stratégies gérées par la partie CC du système 152

4.5.4.2 Autres stratégies de reconfiguration 155

4.5.5 Modèle complet 157

4.6 Analyse 160

4.6.1 Identification de scénarios critiques 160

4.6.2 Disponibilité des fonctions 163

4.7 Bilan 167

4.1 Introduction

Dans ce chapitre, la modélisation GBDMP et l'analyse de SdF basée sur un modèle GBDMP sont expérimentées sur un cas d'étude de taille plus importante que les exemples traités dans les chapitres précédents. Il s'agit d'une version très simplifiée d'une partie de centrale nucléaire à eau pressurisée. Cette version a été conçue en partenariat avec des collaborateurs industriels du projet CONNEXION dans le but d'illustrer plusieurs problématiques industrielles, en ce qui concerne la modélisation et l'analyse de SdF des systèmes complexes.

Afin de rendre possible la réalisation de ce traitement de cas et étant donné le contexte industriel dans lequel ce travail a été mené [1], un prototype d'outil logiciel a été développé dans le cadre de la thèse. La section 4.2 de ce chapitre a pour objet la présentation de ce prototype d'outil.

L'utilisation de cet outil s'inscrit dans un processus de conception d'architecture opérationnelle de Contrôle-Commande (CC) conçu dans le cadre du projet CONNEXION. Afin de replacer l'analyse de SdF basée sur un modèle GBDMP dans son contexte d'application, ce processus de conception sera décrit dans la section 4.3.

Enfin, le système considéré est respectivement décrit, modélisé et analysé dans les sections 4.4, 4.5 et 4.6.

4.2 SAGE : un outil pour l'édition et la manipulation des modèles

SAGE (pour *Safety Analysis in a GBDMP Environment*) est un prototype d'outil logiciel à l'usage de l'ingénieur expert en SdF. Il s'agit d'une implémentation du formalisme GBDMP, et des algorithmes définis au sein de ce mémoire, dans le langage de programmation Python 3 [2].

L'outil permet avant tout de construire des modèles grâce à une description textuelle, de les représenter graphiquement et de les simuler à l'aide d'une interface graphique (utilisant la bibliothèque *pygame* [3]). Pour déterminer automatiquement la disposition

1. Rappelons que le projet de recherche CONNEXION est un projet de recherche visant à fournir des outils logiciels innovants.
2. L'implémentation représente plus de 1600 lignes de code.
3. *pygame* est une bibliothèque Python librement disponible sur le site *http ://www.pygame.org*, conçue pour faciliter le développement de petites applications vidéoludiques.

des éléments du modèle, l'outil Graphviz est utilisé (cf. [Gansner and North, 2000]) mais si l'utilisateur n'est pas satisfait de cette disposition il peut la modifier directement sur l'interface graphique (et sauvegarder la nouvelle) à l'aide d'une manipulation intuitive. Un exemple de copie d'écran du simulateur graphique de l'outil est reporté sur la Figure 4.1. Maintenir le *clic droit* sur une feuille (resp. un commutateur) permet d'ouvrir une

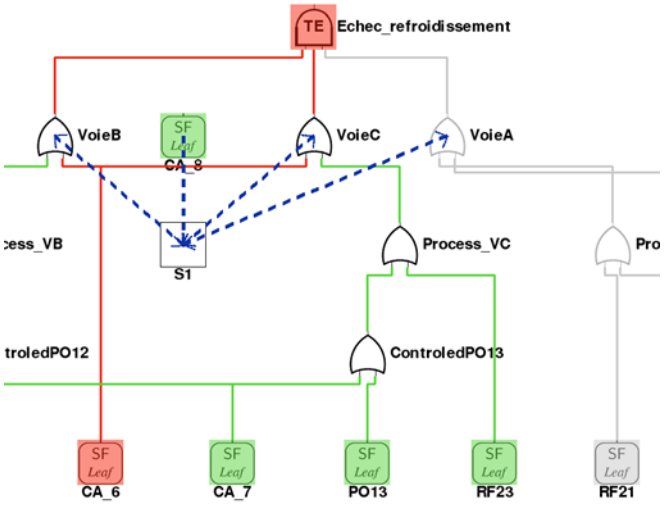

Figure 4.1 – Copie d'écran du simulateur graphique de l'outil SAGE

bulle contextuelle, dans laquelle se dessine la représentation graphique du PMC (resp. de la machine de Moore) qui lui est associé (cf. Figure 4.2). Enfin, un code couleur permet de représenter l'état courant du modèle. Ainsi, dans la vue *arbre de défaillance enrichi*, une feuille est colorée :

- en gris lorsqu'elle est inactive (mode 0) et non défaillante ;
- en vert lorsqu'elle est active (mode > 0) et non défaillante ;
- en orange lorsqu'elle est inactive et défaillante ;
- en rouge lorsqu'elle est active et défaillante.

La racine principale est également colorée en vert ou en rouge pour identifier au premier coup d'œil si la fonction cible est correctement réalisée ou non. Dans les vues *machine de Moore*, l'état courant est coloré en vert. Finalement, dans la vue *PMC*, l'état courant est coloré en vert, et les états accessibles suite à l'occurrence d'un événement *spontané*, sont colorés en mauve. Un *clic gauche* sur l'un de ces états permet de déclencher l'occurrence

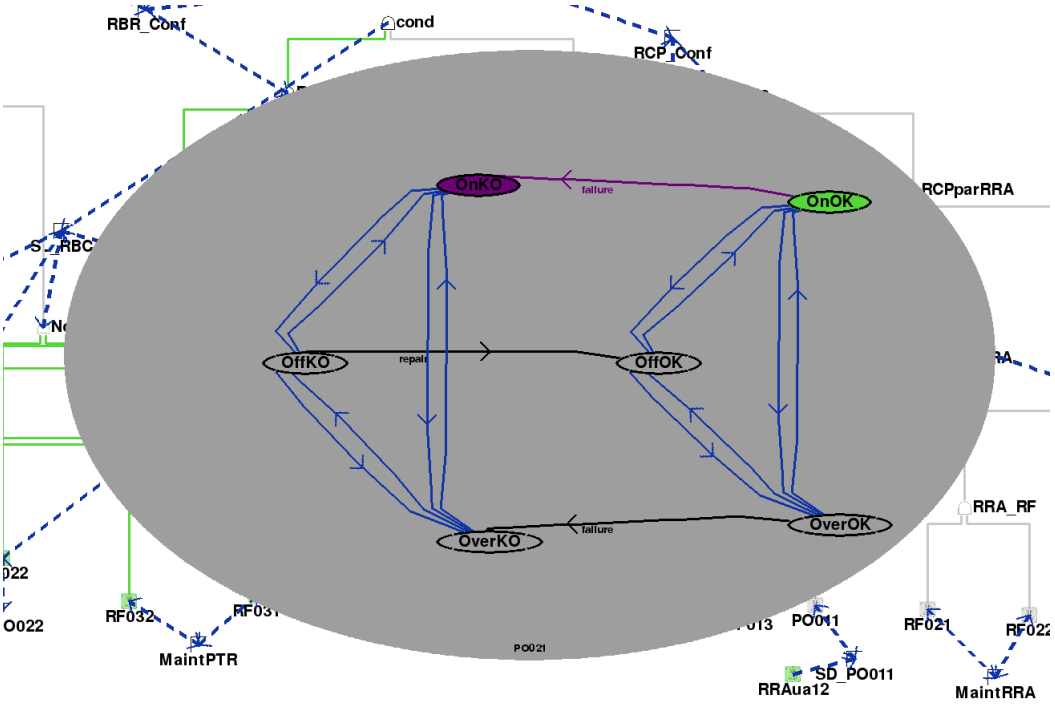

Figure 4.2 – Copie d'écran de la vue *PMC*, via le simulateur graphique de l'outil SAGE

de l'événement conduisant dans cet état, et de propager ses effets conformément à l'algorithme 1 [4] (actualisation des variables, occurrence des événements *provoqués*...). Un *clic gauche* directement sur une feuille permet de générer arbitrairement un des événements *spontanés* relatifs à cette feuille, pouvant survenir dans l'état courant.

Une fois le modèle construit, l'utilisateur peut paramétrer les analyses qu'il souhaite réaliser. Pour l'analyse qualitative, il peut indiquer la longueur maximale des SCM à trouver. L'algorithme 2 sera donc appliqué jusqu'à ce que toutes les séquences de longueur inférieure ou égale à celle-ci soient explorées. L'outil retourne les résultats dans un fichier, qu'il aura rempli à la volée. De cette manière, si le calcul est arrêté prématurément pour une raison quelconque, l'utilisateur pourra néanmoins consulter un résultat partiel. Il n'est pas toujours aisé de comprendre les SCM, lorsque leur taille dépasse 3 ou 4 événements. Aussi, le simulateur graphique du modèle permet d'aider à l'interprétation des résultats de l'analyse qualitative.

Pour l'analyse quantitative, l'utilisateur choisit la taille de la chaîne de Markov à générer (i.e. le nombre maximal de transitions), la durée du scénario (en heures) et le pas de temps du calcul. L'outil procède alors en trois temps :

4. Le calcul du niveau de profondeur des sommets du modèle utilise un algorithme de tri topologique de graphe orienté acyclique, tel que celui qui est décrit dans l'annexe A.

1. traduction du modèle en langage AltaRica, conformément aux principes décrits dans la sous-section 3.3.2.2 et à l'algorithme 3. Le résultat de cette traduction est inscrit dans un fichier au format *.alt* (le format de fichier lisible par les outils développés dans le cadre du projet AltaRica) ;

2. appel à l'outil *LimmMark* développé dans le cadre de la thèse [Brameret, 2015] [5] ;

3. génération d'un graphe représentant l'évolution de l'indisponibilité au cours du temps de la mission. Si la chaîne de Markov est trop grande pour être calculée entièrement (ce qui est généralement le cas), trois courbes sont tracées : l'estimation de l'indisponibilité (fournie en considérant la chaîne sans l'état puits), ainsi que ses bornes (fournies en considérant la chaîne avec l'état puits).

Le prototype d'outil SAGE a été utilisé pour traiter le cas d'étude qui fait l'objet principal de ce chapitre. S'il ne constitue en aucun cas un outil opérationnel pour une utilisation industrielle, il est suffisamment fonctionnel pour jouer le rôle de démonstrateur pour cette thèse.

4.3 Description du processus de conception

Cette section a pour objet de décrire dans les grandes lignes le processus de conception d'architecture opérationnelle de CC développé dans le cadre du projet CONNEXION, afin de replacer dans son contexte l'application de l'analyse de SdF basée sur un modèle GBDMP. Un schéma représentant le processus de conception est reporté sur la Figure 4.3, et sera commenté par la suite.

Avant de procéder à cette conception, la partie opérative du système, ainsi que la plateforme matérielle de la partie CC, ont déjà été conçues et validées. L'objectif est donc de concevoir et de valider l'architecture opérationnelle de CC, c'est-à-dire l'architecture fonctionnelle répartie sur la plateforme matérielle, en tenant compte de toutes les contraintes physiques et temporelles, ainsi que des exigences exprimant les besoins du projet de l'installation. Comme l'illustre la Figure 4.4, la partie opérative (ou procédé) du système (ici, une partie de centrale nucléaire à eau pressurisée) a été décomposée en Systèmes Elémentaires (SE) en interactions. La confrontation de plusieurs données

5. L'utilisation de cet outil externe est soumise aux modalités décrites dans la licence "Utilisateur" de la plateforme *OpenAltaRica*, accessible depuis le site internet *http ://openaltarica.fr*.

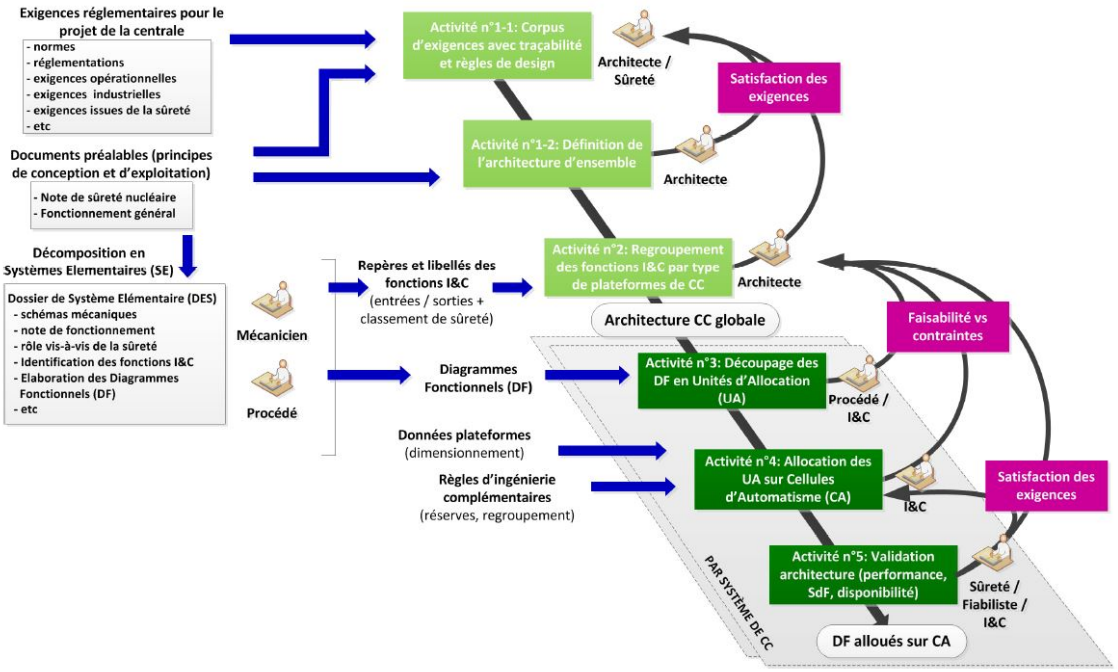

Figure 4.3 – Schéma du processus de conception d'une architecture opérationnelle de CC

(exigences, descriptions physique des SE...) permet aux ingénieurs d'élaborer les Diagrammes Fonctionnels (DF), décrivant les fonctions de commande vouées à être implémentées sur la plateforme de CC, pour contrôler la partie opérative. Ainsi, une partie de ces DF décrit les stratégies de reconfiguration du système.

Les premières activités du processus (cf. activités n°1.1, 1.2 et 2 sur la Figure 4.3) visent à constituer l'architecture fonctionnelle globale du CC, tout en veillant à la traçabilité des exigences. C'est à ces étapes que sont définies les grandes fonctions de commande [6] (désignées par les fonctions I&C sur le schéma, pour *Instrumentation and Control* en anglais) et qu'elles sont distribuées sur les différents types de plateformes matérielles, notamment selon leur classement de sûreté. Cette première opération d'allocation ne pose aucun problème technique, car le niveau de granularité est suffisamment élevé et les contraintes réglementaires suffisamment restrictives pour que ses dimensions restent de taille humaine. Aussi, ces activités étant plutôt liées à des problématiques de gestion des données, nous ne nous y attarderons pas plus.

L'activité n°3 consiste en un découpage des DF - objet abstrait jusqu'alors -, en Unités d'Allocation (UA). L'UA correspond à la brique fonctionnelle élémentaire insé-

6. L'attribut "grand" qualifie ici le niveau de granularité.

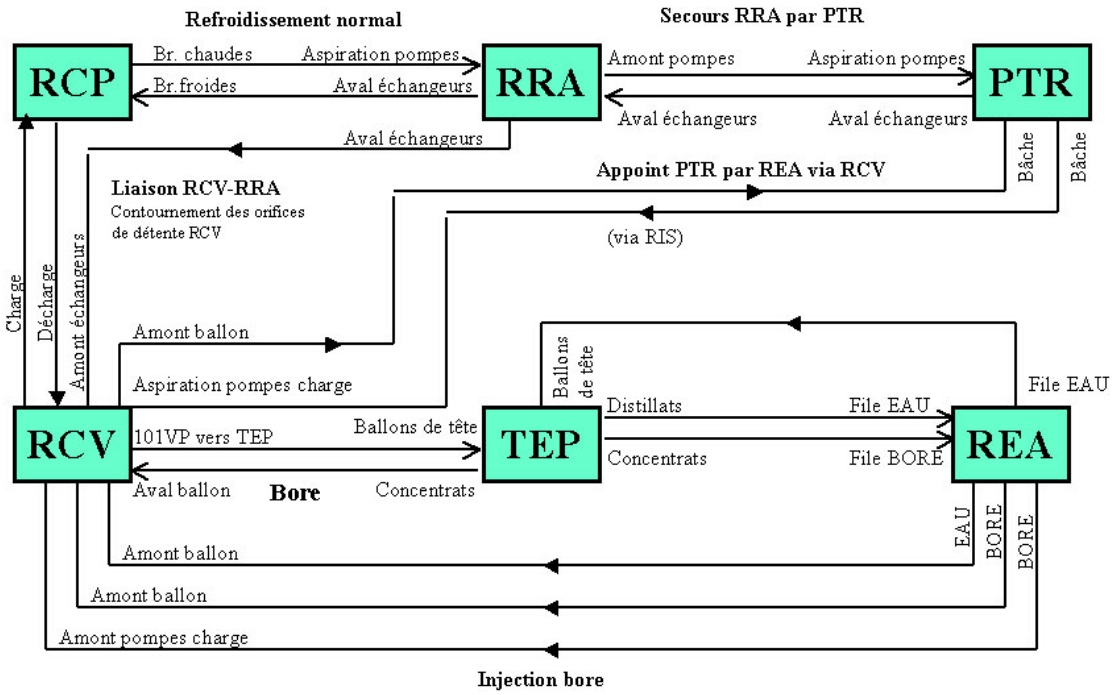

Figure 4.4 – Les six systèmes élémentaires d'une partie de centrale nucléaire à eau pressurisé et leurs interconnexions

cable de l'architecture[7]. Elle est décrite par une planche SCADE (du nom d'un logiciel développé par l'entreprise *Esterel Technologies*, un des partenaires du projet), prête à être implémentée sur une unité matérielle de commande élémentaire, appelée Cellule d'Automatisme (CA). Cette étape est l'occasion de préciser la cible de chaque UA, au niveau de la partie opérative, et de dupliquer certaines d'entre elles, pour établir des redondances (actives) de fonctions de commande.

L'activité suivante consiste à distribuer les UA sur les CA, le résultat étant une architecture opérationnelle de CC. Cette seconde opération d'allocation est irréalisable manuellement, car ses dimensions sont trop importantes (plusieurs milliers d'UA à allouer sur plusieurs centaines de CA). La résolution de ce problème par recherche d'atteignabilité dans un réseau d'automates communicants a fait l'objet de la thèse [Lemattre, 2013]. Cette méthode est très efficace pour trouver une solution au problème, mais l'est beaucoup moins pour trouver une solution optimale. Deux approches alternatives ont donc été proposées pour déterminer plus rapidement une solution optimale, selon plusieurs critères, tout en intégrant des contraintes d'allocation supplémentaires. La première idée est d'utiliser un algorithme génétique (cf. [Nasri et al., 2015]), mais cela implique de

7. On arrive donc au niveau de granularité le plus fin.

disposer préalablement d'un ensemble de solutions. La seconde approche utilise un algorithme de *Bin Packing* et semble particulièrement prometteuse en termes d'efficacité et de passage à l'échelle (cf. [Benazouz and Faure, 2015]).

Enfin, la dernière activité est celle qui concerne directement ce travail de thèse. Il s'agit de valider que l'architecture opérationnelle définie à l'étape précédente satisfait aux exigences relatives aux performances temporelles et à la SdF. L'évaluation des performances temporelles se rapporte à des problèmes de calcul de temps de cycle dans un réseau, ce qui est assez éloigné du propos de cette thèse. En revanche, la validation du respect des exigences de SdF se fait à l'aide d'analyses de SdF basées sur les modèles. Dans le secteur de l'industrie nucléaire, ces analyses sont appelées Analyses Qualitatives de Sûreté (AQS, pour les analyses qualitatives) et Etudes Probabilistes de Sûreté (EPS, pour les analyses quantitatives). Ainsi, lorsque l'ingénieur expert en SdF a la charge de valider une architecture opérationnelle de CC, il doit dans un premier temps construire un modèle de SdF du système bouclé. Puis, des analyses basées sur ce modèle devront permettre de s'assurer que l'architecture proposée n'est ni à l'origine d'une (ou plusieurs) SCM trop courte(s) au regard des exigences, ni responsable d'une disponibilité trop faible de telle ou telle fonction du système bouclé. On peut faire l'hypothèse que la partie logicielle du CC est conforme à sa spécification. Aussi, les seuls éléments susceptibles de défaillir dans le système bouclé sont les composants de la partie opérative, les CA de la partie CC, ainsi que certains composants matériels assurant la communication entre les deux parties (bus, répartiteurs, modules d'entrée/sortie, etc...).

Nous allons voir dans la suite, comment le formalisme GBDMP peut être exploité avantageusement pour effectuer cette activité de validation sous l'angle de la SdF.

4.4 Description du cas

Dans cette section, nous décrivons le système à analyser. Il s'agit d'une version très simplifiée d'une partie de centrale nucléaire à eau pressurisée[8].

8. Cette description n'est pas conforme à la réalité pour des raisons de confidentialité et de simplification, mais présente un panel de problématiques réalistes.

4.4.1 Le procédé physique

Nous avons choisi de limiter l'étude à deux SE (cf. Figure 4.4) : le PTR (Traitement et Refroidissement des Piscines) et le RRA (Refroidissement du Réacteur à l'Arrêt, cuve fermée). Par ailleurs, une centrale nucléaire est un système multi-phases, les différentes phases de mission étant appelées *états de tranche*. Une équipe d'opérateurs a la charge de conduire la centrale, c'est à dire de déclencher les événements de changement d'état de tranche. Le temps passé dans chaque état de tranche varie en fonction de nombreux facteurs (notamment des besoins de production de la centrale). Pour ce cas d'étude, nous n'observerons que trois états de tranche [9] :

- RP : Réacteur en Puissance ; c'est la phase de production, elle dure en moyenne 42 jours (soit environ 1000 heures).
- APR : Arrêt Pour Rechargement ; lors de cette phase de mission, la cuve du réacteur est ouverte, afin de sortir le réactif consommé et de faire entrer le nouveau réactif. Cette phase dure en moyenne 4 jours (soit environ 100 heures).
- Tran : état transitoire entre RP et APR ; c'est au cours de cette phase que la puissance du réacteur est augmentée (pour passer de APR à RP) ou abaissée (pour passer de RP à APR). Cette phase dure en moyenne 2 jours (soit environ 50 heures).

L'unique fonction réalisée par le système RRA est le Refroidissement du Circuit Primaire (RCP) pendant les phases transitoires. Lorsque le réacteur est arrêté pour rechargement, la cuve est ouverte, donc RRA ne doit pas être utilisé, et lorsqu'il est en puissance, c'est un autre SE (hors périmètre du cas) qui a la charge de le refroidir. Comme l'illustre la Figure 4.5, le système RRA est constitué de deux pompes (011PO et 012PO), deux échangeurs thermiques (015RF et 016RF), deux vannes manuelles (013VP et 014VP) et deux vannes automatisées (207VP et 211VP). La fonction de refroidissement peut être réalisée grâce à une pompe et un échangeur. Ainsi, quatre lignages permettent d'assurer cette fonction :

- 011PO - 013VP - 015RF
- 011PO - 014VP - 016RF

9. On rappelle que les valeurs indiquées ne sont pas réalistes.

- 012PO - 013VP - 015RF
- 012PO - 014VP - 016RF

Les pompes sont des composants simples : ils n'ont que les deux modes d'opération *actif* et *inactif*. Les échangeurs ne sont pas instrumentés, ils ne peuvent donc pas être activés ou désactivés. En revanche, on distinguera les modes *plein* ou *vide*, selon qu'un flux d'eau les traverse ou non. Les vannes sont de type TOR (Tout Ou Rien), on distinguera donc les modes *ouvert* et *fermé*. Les vannes manuelles permettent à des opérateurs d'interrompre le flux dans les échangeurs afin d'intervenir sur ceux-ci. Les vannes automatisées permettent d'assurer la liaison avec le système PTR, afin de reporter la charge de RRA sur PTR, en cas d'échec des quatre lignages, ce qui fait l'objet de la fonction SRRA, décrite ci-après.

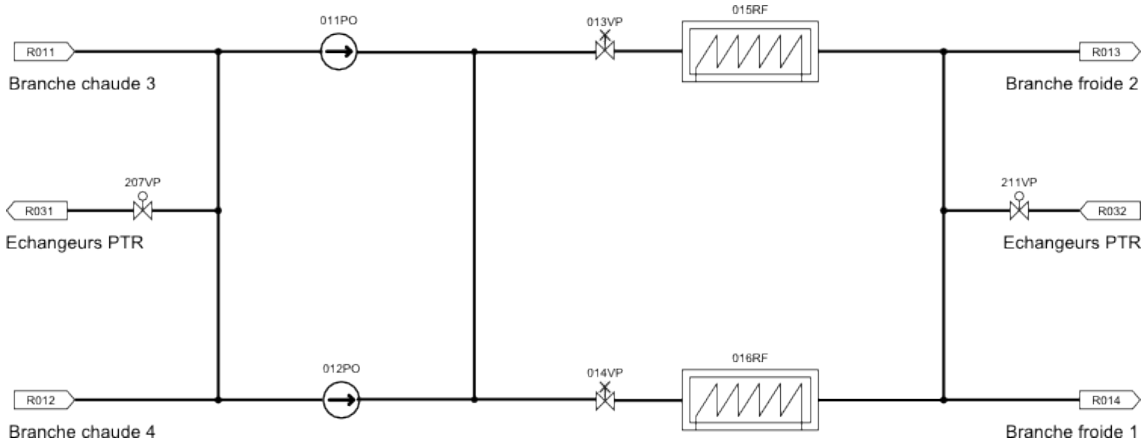

Figure 4.5 – Schéma mécanique du système élémentaire RRA

Le PTR est dimensionné pour réaliser trois fonctions, qui seront requises ou non, selon l'état de tranche :

- RPC : Refroidissement de la Piscine Combustible ; cette piscine contient les stocks de réactifs (neufs et usagés). Ces stocks doivent être immergés en permanence (pour l'étude considérée), donc cette fonction doit être assurée dans les trois états de tranche APR, RP et Tran.
- RPR : Refroidissement de la Piscine Réacteur ; lorsque la cuve du réacteur est ouverte (pour le rechargement), celui-ci doit être immergé. Ainsi, la piscine du réacteur n'est remplie - et donc ne doit être refroidie - que dans l'état de tranche APR.

- SRRA : Secours de RRA ; cette fonction est une redondance fonctionnelle passive de celle devant être réalisée par RRA. Aussi n'est-elle requise que dans les phases transitoires, en cas d'échec de RRA.

Comme le montre la Figure 4.6, PTR est constitué de deux pompes (021PO et 022PO), deux échangeurs thermiques (025RF et 026RF) et deux vannes manuelles (022VP et 023VP). De même que pour RRA, quatre lignages permettent de réaliser chaque fonction de refroidissement :

- 021PO - 023VP - 025RF
- 021PO - 024VP - 026RF
- 022PO - 023VP - 025RF
- 022PO - 024VP - 026RF

De plus, ce système est dimensionné pour pouvoir réaliser deux fonctions de refroidissement simultanément, sur chaque lignage. Ainsi, à la différence des pompes de RRA, celles de PTR ont trois modes d'opération, à l'instar de celles qui ont été décrites dans la sous-section 2.2.2.2 : *Off*, *On* et *Over*. Les échangeurs et vannes manuelles sont, quant à eux, identiques à ceux de RRA.

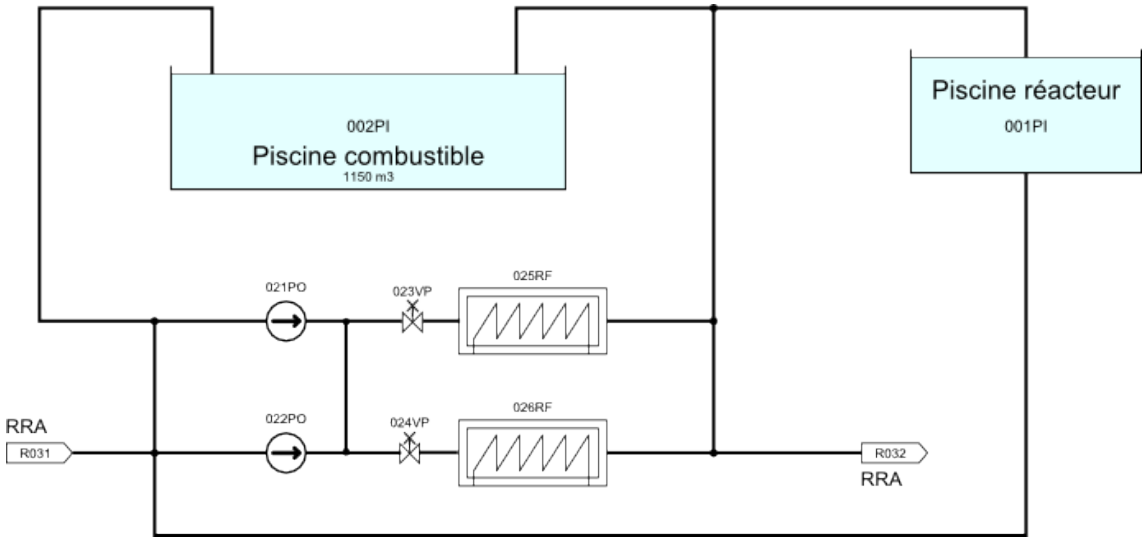

Figure 4.6 – Schéma mécanique du système élémentaire PTR

4.4.2 Le contrôle-commande

A présent nous allons décrire les stratégies de reconfiguration de ces deux SE, qui ont été spécifiées à travers les UA. On rappelle que ces UA sont décrites à l'aide de planches SCADE. Ces planches sont construites à l'aide de boîtes génériques contenant pour la plupart un diagramme logique, sauf SC2 (pour *Sélecteur de Composant de type 2*) qui contient un graphe d'état, analogue à une machine de Moore. Comme ces planches font référence à de nombreuses variables de CC sortant du cadre de cette étude, seuls des extraits épurés sont présentés ici.

Pour le système RRA, deux UA traduisent trois stratégies de reconfiguration. La planche SCADE spécifiant l'UA1 est reportée sur la Figure 4.7. Cette UA traduit en premier lieu le fait que la fonction RCP est requise uniquement dans l'état de tranche transitoire (désigné par ETAT2 sur la figure). Cette stratégie est spécifiée par la boîte RCP. La seconde boîte (selFRRA) est un sélecteur permettant de déterminer laquelle des deux pompes doit être activée en fonction de leurs états dysfonctionnels, afin d'assurer cette fonction lorsqu'elle est requise. Via le graphe d'état associé, cette boîte décrit une redondance passive entre les deux pompes, conformément à la stratégie de reconfiguration suivante :

- Initialement, seule la pompe 011PO est démarrée.
- Si aucune pompe n'est défaillante : ne rien changer.
- Si une pompe est défaillante : la pompe défaillante doit être inactive, et la pompe non défaillante doit être active.
- Si les deux pompes sont défaillantes : elles doivent être inactives.

Par ailleurs, en cas d'échec de la fonction de refroidissement (que ce soit à cause des pompes ou des échangeurs), les vannes 207VP et 211VP doivent être ouvertes afin de reporter la charge du système RRA sur le système PTR. Dès que la fonction peut être rétablie sur RRA, ces vannes doivent être refermées. Cette stratégie de reconfiguration est portée par l'UA2, dont la planche SCADE est représentée sur la Figure 4.8.

Pour le système PTR, trois UA traduisent trois stratégies de reconfiguration. Premièrement, l'UA3 traduit le fait que la fonction de refroidissement de la piscine du réacteur

11. DS : Demande de Secours (symptomatique d'un défaut) ; OE : Ordre d'Enclenchement
12. OO : Ordre d'Ouverture ; OF : Ordre de Fermeture

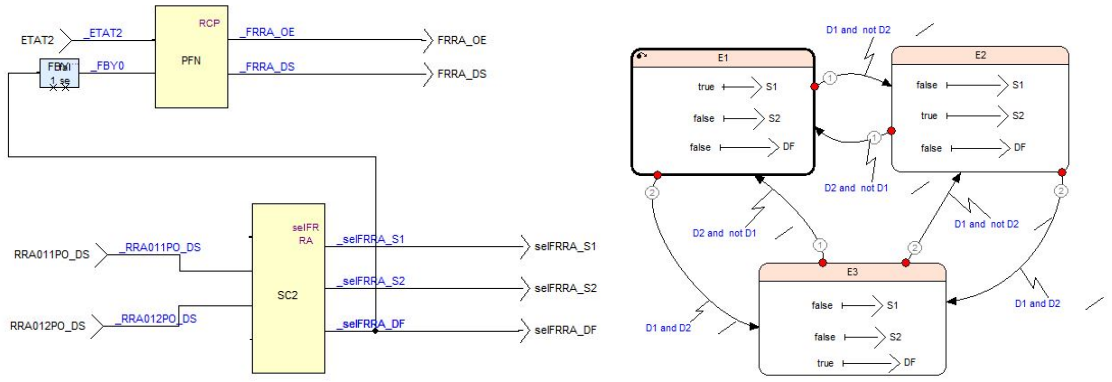

Figure 4.7 – Planche SCADE spécifiant l'UA1 (à droite, contenu d'une boîte de type SC2) [11].

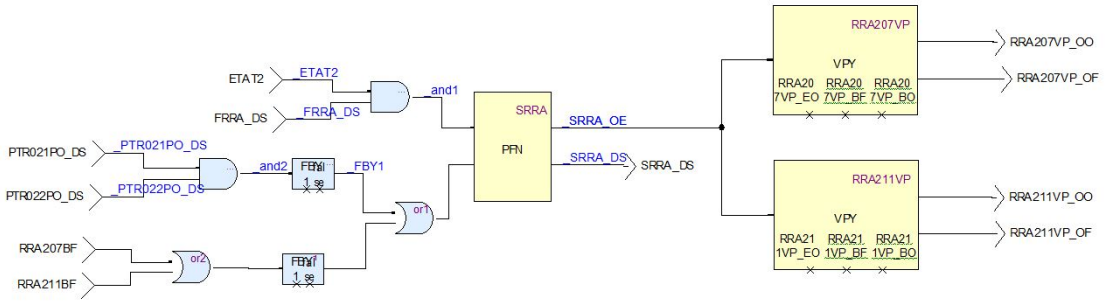

Figure 4.8 – Planche SCADE spécifiant l'UA2 [12].

est requise uniquement dans l'état de tranche APR. Les unités d'allocation 4 et 5 décrivent quant à elles les stratégies de redondance consistant à déterminer dans quel mode d'opération doivent être commutées les pompes 021PO et 022PO. Les planches SCADE correspondantes à ces UA sont reportées sur la Figure 4.9. En l'absence de défaillance, la pompe 021PO est dédiée à la réalisation de la fonction RPC (qui doit être assurée dans tous les états de tranche considérés), tandis que la pompe 022PO est dédiée à la réalisation des fonctions RPR (qui ne doit être assurée que dans l'état APR) et SRRA (qui ne doit être assurée que dans l'état Tran, si le système RRA est indisponible). Ainsi, l'UA4 décrit la stratégie de reconfiguration suivante :

- Si la pompe 021PO est défaillante, elle doit être inactive (logique intrinsèque à la boîte générique PPS).

- Sinon, elle doit être active tant que la fonction RPC est requise (toujours vrai dans notre cas).

- De plus, si la pompe 022PO est défaillante, alors que les fonctions RPR ou SRRA sont requises, la pompe 021PO doit être commutée sur le mode *Over*, pour assumer la charge de 022PO.

L'UA5 décrit quant à elle la stratégie de reconfiguration suivante :

- Si la pompe 022PO est défaillante, elle doit être inactive.
- Sinon, elle doit être active tant que l'une des fonctions RPR ou SRRA est requise.
- De plus, si la pompe 021PO est défaillante, alors que la fonction RPC est requise, la pompe 022PO doit être commutée sur le mode *Over*, pour assumer la charge de 021PO.

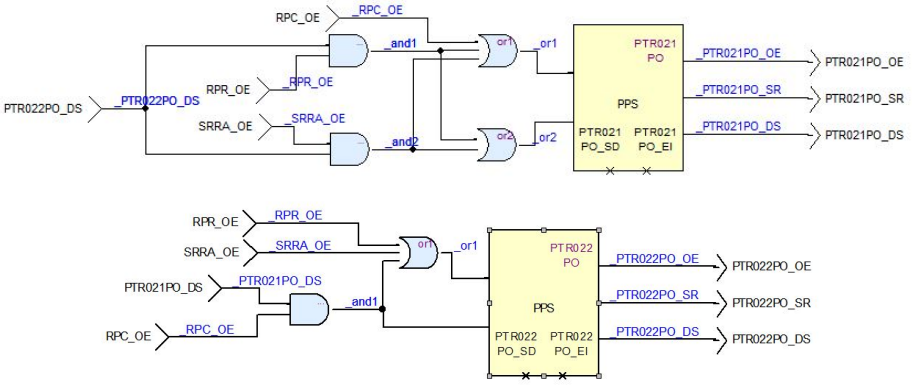

Figure 4.9 – Planches SCADE spécifiant l'UA4 (en haut) et l'UA5 (en bas)[13].

Chacune de ces UA a été dupliquée et les deux UA résultantes ont été allouées à des CA différentes, de telle sorte que cette duplication soit une forme de redondance active. Ainsi, l'allocation suivante a été générée par l'un des outils mentionnés dans la section 4.3[14] :

- Les UA UA1, UA2 et UA4' sont allouées à la CA1 ;
- Les UA UA3 et UA5' sont allouées à la CA2 ;
- Les UA UA1' et UA2' sont allouées à la CA3 ;
- Les UA UA3' et UA4 sont allouées à la CA4 ;

13. SR : Sur-Régime (ordre d'enclenchement en mode *Over*) ; EI : Etat Indisponible ; SD : Symptôme de Défaut

14. Notons que d'autres UA hors cadre de cette étude, peuvent avoir été allouées à ces CA.

- L'UA5 est allouée à la CA5.

Les CA communiquent entre elles et avec la partie procédé via un couple de bus en redondance active. Cette architecture opérationnelle est représentée sur la Figure 4.10.

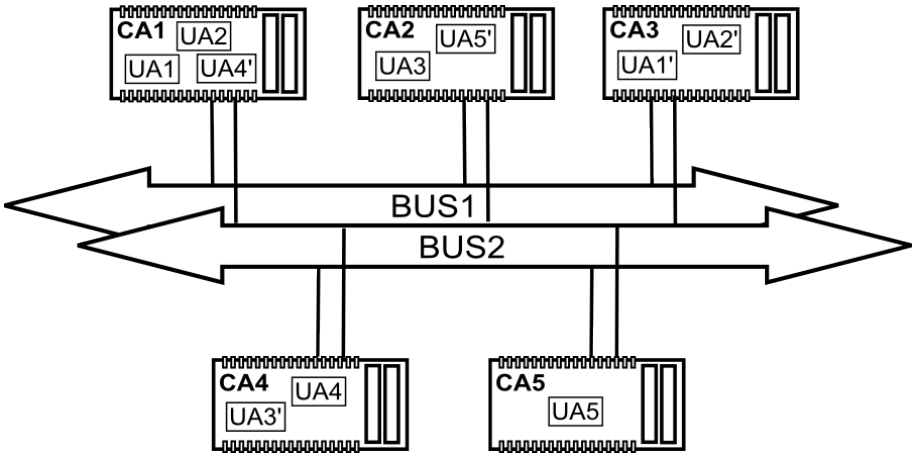

Figure 4.10 – Schéma de l'architecture opérationnelle de CC

4.4.3 Analyse préliminaire des risques

Cette sous-section présente les données de fiabilité des composants résultant d'une analyse préliminaire des risques [15].

Les pompes du système RRA sont des composants tels que ceux décrits par un PMC de type SF (cf. Figure 2.2), avec $\lambda = 3.10^{-5}h^{-1}$, $\lambda_S = 1.10^{-5}h^{-1}$ et $\mu = 4,2.10^{-2}h^{-1}$ [16], excepté qu'ils ne peuvent être réparés que lorsqu'ils sont inactifs. Les pompes du système PTR sont des composants tels que ceux décrits par un PMC de type Po (cf. Figure 2.27), avec $\lambda = 3.10^{-5}h^{-1}$ et $\mu = 4,2.10^{-2}h^{-1}$, excepté qu'ils ne peuvent être réparés que lorsqu'ils sont inactifs.

Les vannes automatisées peuvent refuser de s'ouvrir avec une probabilité de $4,5.10^{-3}$, et de se fermer avec une probabilité de $1,5.10^{-3}$. Dans les deux cas, elles sont réparées avec un taux de probabilité $8,4.10^{-2}h^{-1}$. En revanche, les vannes manuelles ne sont soumises à aucune défaillance, car les opérateurs qui agissent dessus veillent toujours à ce qu'elles soient correctement ouvertes ou fermées.

Lorsqu'un flux d'eau les traverse, les échangeurs thermiques peuvent s'encrasser avec un taux de probabilité $10^{-3}h^{-1}$. L'encrassement d'un échangeur n'est pas considéré

15. On rappelle que les valeurs numériques fournies sont volontairement non conforme à la réalité.
16. Ce taux de probabilité correspond à un temps de réparation moyen d'environ 24 heures.

comme une défaillance, car cela ne l'empêche pas d'assurer correctement son service. Par contre, en cas d'encrassement, les tuyaux d'un échangeur risquent de rompre avec un taux de probabilité $10^{-2}h^{-1}$. La rupture d'un échangeur est une défaillance dans la mesure où le service n'est plus assuré dans ce cas. Un échangeur peut être nettoyé avec un taux de probabilité $0,1h^{-1}$ et peut être réparé avec un taux de probabilité $0,01h^{-1}$ mais, pour ce faire, le flux d'eau le traversant doit être interrompu [17].

Finalement, les composants de contrôle (CA et bus de communication) sont toujours actifs. Ils peuvent défaillir avec un taux de probabilité $6,5.10^{-6}h^{-1}$ et être réparés avec un taux de probabilité $4,2.10^{-2}h^{-1}$. Nous ferons l'hypothèse que lorsqu'ils sont défaillants, leurs sorties sont figées à leur dernière valeur avant la défaillance.

4.4.4 Maintenance des échangeurs

En plus des stratégies de reconfiguration automatisées, gérées par les UA du CC, il existe deux autres stratégies appliquées par des opérateurs ayant la charge d'assurer la maintenance des échangeurs [18]. On a vu dans la sous-section précédente que lorsqu'un échangeur atteignait un certain niveau d'encrassement, le risque de défaillance de celui-ci devenait significatif. C'est pourquoi les opérateurs assurent non seulement une maintenance *corrective* en cas de défaillance, mais aussi une maintenance *préventive* en cas d'encrassement. Dans les deux cas, la réparation est du type *"as good as new"*, c'est à dire que l'échangeur est comme neuf suite à l'opération de maintenance. Pour intervenir sur un échangeur les opérateurs doivent interrompre le flux d'eau le traversant, ce qu'ils peuvent faire en fermant la vanne qui se trouve en amont. Pour assurer la continuité du service, les opérateurs ne doivent pas intervenir simultanément sur 015RF et 016RF, ainsi que sur 025RF et 026RF. De plus, la stratégie de maintenance diffère entre ces deux groupes d'échangeurs. En effet, pour une raison non détaillée, pour le premier groupe, les opérateurs ne privilégient pas la maintenance de l'un des deux échangeurs, tandis que pour le second, la maintenance de 025RF est prioritaire à celle de 026RF. Ainsi, si 015RF et 016RF sont tous les deux encrassés ou défaillants, ils sont réparés dans l'ordre FIFO (*First In First Out*) ; tandis que pour 025RF et 026RF, les opérateurs réparent 025RF,

17. L'hypothèse Markovienne est difficilement justifiable pour la modélisation de ce type de phénomène. Son adoption reste néanmoins pertinente, car elle constitue la moins mauvaise approximation pour traiter ce cas, à l'aide d'une modélisation purement discrète.

18. L'intervention humaine dans la réparation des autres composants n'est pas considérée pour cette étude.

quitte à interrompre la réparation de 026RF. En revanche, pour les deux groupes, la maintenance d'un échangeur défaillant est prioritaire à celle d'un échangeur seulement encrassé.

4.5 Modélisation GBDMP du système

4.5.1 Construction des PMC

Compte tenu de l'analyse préliminaire des risques (cf. la sous-section 4.4.3), nous pouvons construire des PMC pour modéliser le comportement dysfonctionnel des composants matériels du système bouclé.

Le comportement dysfonctionnel des pompes peut être modélisé par des PMC de type SF pour les pompes du système RRA et de type Po pour celles du système PTR [19]. La représentation graphique de ces PMC est rappelée sur la Figure 4.11, avec les valeurs numériques fournies par l'analyse préliminaire des risques.

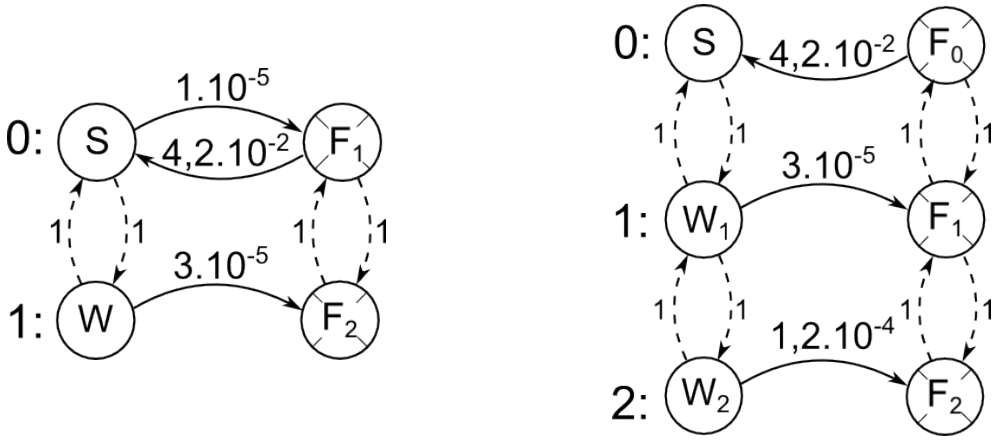

Figure 4.11 – PMC modélisant le comportement dysfonctionnel d'une pompe du système RRA (à gauche : SF) et du système PTR (à droite : Po)

Les vannes seront modélisées par des PMC de type VM (pour les Vannes Manuelles) et VA (pour les Vannes Automatisées), représentés sur la Figure 4.12. Conformément aux résultats de l'analyse préliminaire des risques, les vannes manuelles ne défaillent pas, tandis que les vannes automatisées peuvent défaillir à la commutation (refus d'ouverture ou de fermeture, avec des probabilités différentes) et ainsi rester bloquées en position ouverte ou fermée.

19. Les valeurs des taux de réparation sont choisies pour traduire le fait que les pompes ne peuvent être réparées que lorsqu'elles sont inactives.

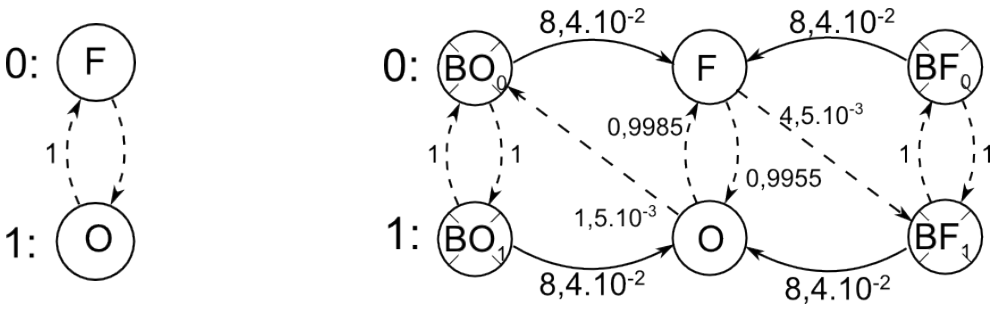

Figure 4.12 – PMC modélisant les vannes manuelles (à gauche : VM) et les vannes automatisées (à droite : VA)

Le comportement des Echangeurs Thermiques sera modélisé par un PMC de type ET, représenté sur la Figure 4.13. Ce PMC est construit sur 6 états qui sont obtenus en combinant les modes d'opération P (Plein) et V (Vide), et les modes de défaillance OK (intact), E (Encrassé) et KO (défaillant). On peut remarquer que l'encrassement et la défaillance d'un échangeur ne peuvent survenir que lorsque ce dernier est "plein" (i.e. lorsqu'un flux d'eau le traverse). L'échangeur ne peut alors être réparé que lorsqu'il est "vide".

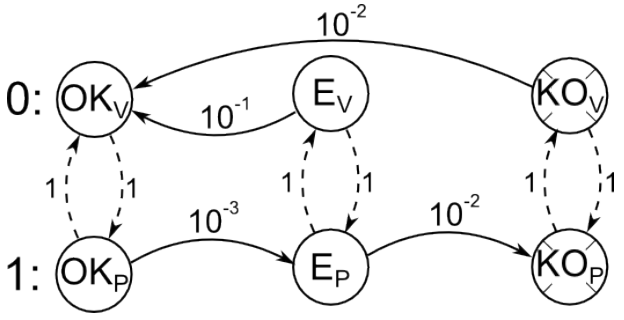

Figure 4.13 – PMC de type ET modélisant le comportement dysfonctionnel d'un échangeur thermiques

Le comportement dysfonctionnel des composants de contrôle (CA ou bus de communication) peut être modélisé à l'aide d'un PMC de type UC (Unité de Contrôle), représenté sur la Figure 4.14. Ce PMC très simple est construit sur deux états : W (Working) et F (Failure). On rappelle que lorsqu'une feuille de GBDMP est de dimension 1, l'unique mode de son PMC associé porte le numéro 1 (cf. Règle 5).

Finalement, un dernier PMC doit être construit pour modéliser la conduite de la centrale. Ce PMC est appelé Co (Conduite) et est représenté sur la Figure 4.15. On peut remarquer que l'inverse des taux de probabilité correspondent aux durées moyennes de

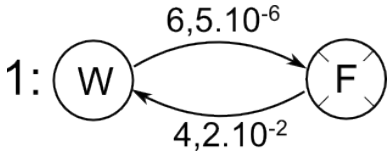

Figure 4.14 – PMC de type *UC* modélisant un composant de contrôle

séjour dans chacun des états de tranche.

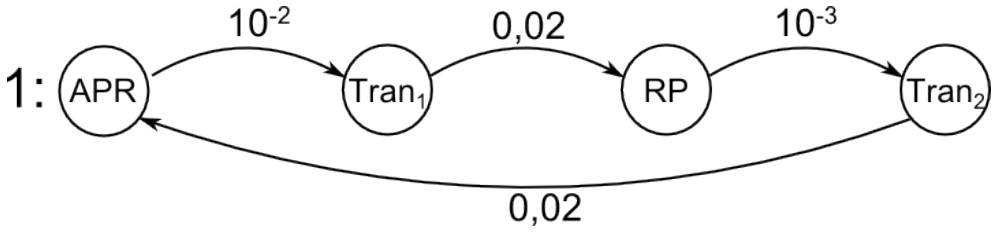

Figure 4.15 – PMC de type *Co* modélisant la conduite de la centrale

4.5.2 Modélisation de l'architecture opérationnelle de CC

Nous allons à présent construire la partie du GBDMP, permettant de modéliser les conditions de succès de chacune des stratégies de reconfiguration gérées par la partie CC. Cette partie est représentée sur la Figure 4.16.

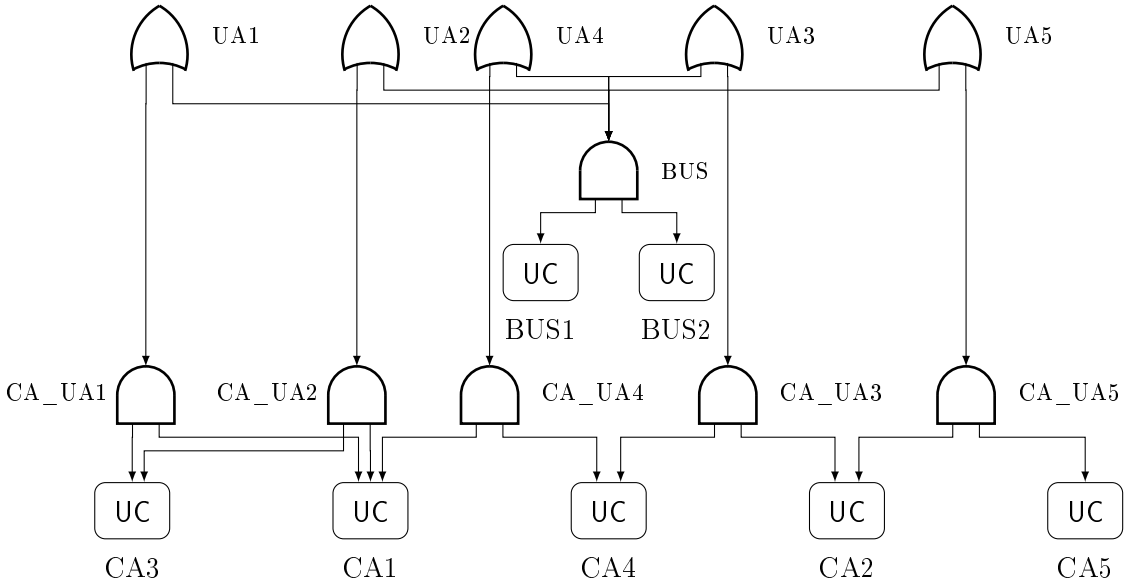

Figure 4.16 – Vue *arbre de défaillance enrichi* du modèle GBDMP pour l'architecture opérationnelle de CC

Chaque stratégie de reconfiguration échoue lorsque défaillent soit les deux bus de communication redondants, soit les deux CA dans lesquels ont été allouées les deux UA redondantes spécifiant la stratégie en question. Par exemple, la stratégie de redondance entre les pompes du système RRA est décrite par l'UA1 et l'UA1', qui ont respectivement été implémentées dans la CA1 et la CA3 (cf. Figure 4.10). Aussi, cette redondance est indisponible lorsque BUS1 et BUS2 sont défaillants, ou lorsque CA1 et CA3 sont défaillants.

4.5.3 Construction de la structure du GBDMP pour la partie procédé

Avant de modéliser les stratégies de reconfiguration du système, nous construisons la structure du GBDMP, en se basant uniquement sur la décomposition fonctionnelle du procédé. Cette structure est représentée sur la Figure 4.17.

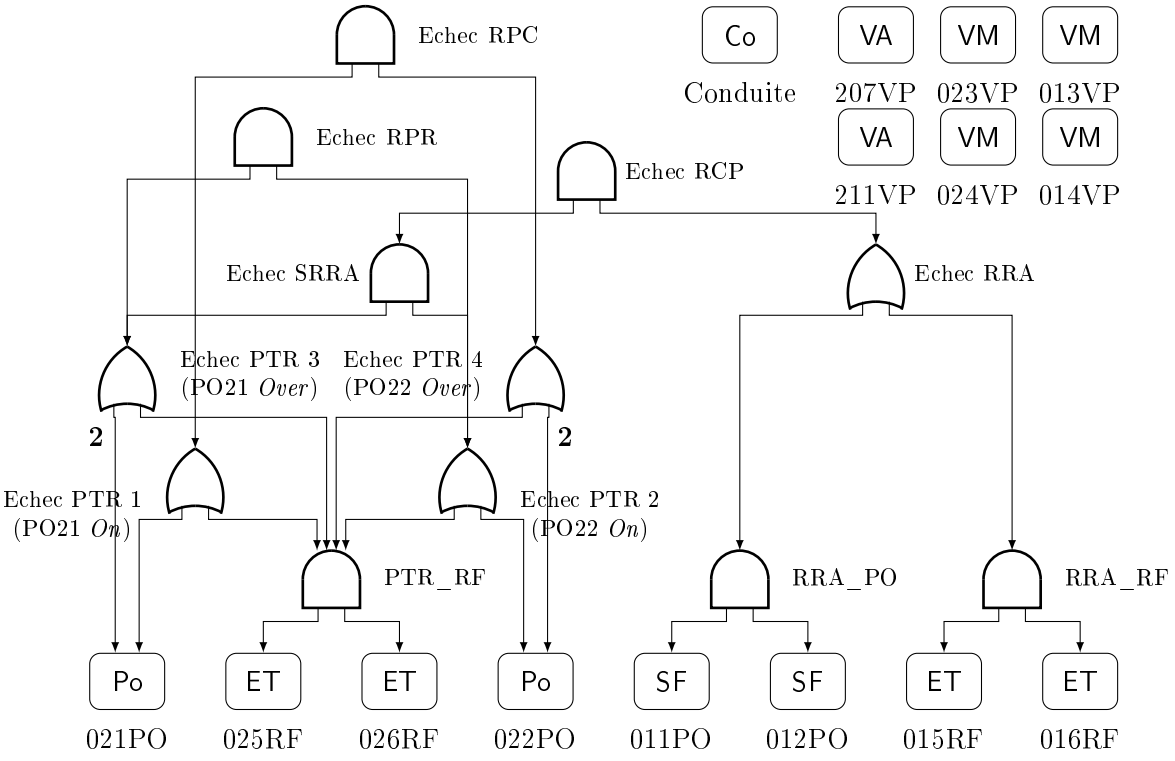

Figure 4.17 – Structure du modèle GBDMP pour la partie procédé du système

En racines de cette structure arborescente, on retrouve les trois fonctions principales du système :

- La fonction RCP (Refroidissement du Circuit Primaire) peut être réalisée soit par RRA soit par PTR (grâce à la sous-fonction SRRA) :

  - Pour réaliser cette fonction par RRA, il faut qu'au moins une pompe parmi 011PO et 012PO, et un échangeur parmi 015RF et 016RF soient disponibles.
  - La fonction SRRA est réalisée soit par la pompe 022PO dans le mode *On*, soit par la pompe 021PO dans le mode *Over* (et par un échangeur parmi 025RF et 026RF).

- La fonction RPR (Refroidissement de la Piscine Réacteur) a les mêmes critères de succès que la fonction SRRA.

- La fonction RPC(Refroidissement de la Piscine Combustible) est réalisée soit par la pompe 021PO dans le mode *On*, soit par la pompe 022PO dans le mode *Over* (et par un échangeur parmi 025RF et 026RF).

Enfin, les feuilles modélisant les vannes et la conduite de la centrale seront connectées à cette structure via les commutateurs.

4.5.4 Construction des machines de Moore

A présent, nous décrivons les commutateurs du modèle pour traduire les stratégies de reconfiguration du système.

4.5.4.1 Stratégies gérées par la partie CC du système

Pour toutes les stratégies gérées par la partie CC du système, l'entrée numéro 0 de la machine de Moore correspond au statut de *défaillance* du sous-système de CC nécessaire à la réalisation de la stratégie.

Premièrement, la machine de Moore permettant de déterminer dans quel état de tranche la fonction RCP est requise est représentée sur la Figure 4.18. Elle modélise l'une des deux stratégies décrites par l'UA1 (et l'UA1'). Ainsi, la machine de Moore contrôle le statut de *réquisition* de la fonction RCP, en fonction de l'état de tranche, sous réserve que la porte UA1 soit non défaillante.

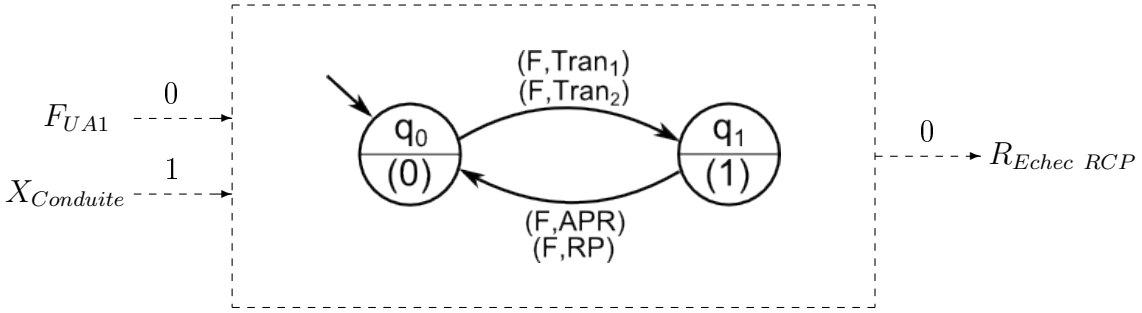

Figure 4.18 – Commutateur S1 (contenant la machine de Moore M_{req1}) contrôlant la réquisition de la fonction RCP

L'UA1 décrit également la stratégie de redondance pour les pompes du système RRA. Aussi, la machine de Moore correspondante (cf. Figure 4.19) ressemble fortement au contenu de la boîte générique de type $SC2$, sur la planche SCADE spécifiant l'UA1 (cf. Figure 4.7). La seule différence est que cette machine de Moore prend en compte l'échec de la reconfiguration. Ainsi, pour pouvoir changer d'état, la porte UA1 doit être non défaillante (i.e. l'entrée numéro 0 doit être à $False$). Les entrées numéro 1 et 2 doivent respectivement être reliées aux statuts de *défaillance* des pompes 011PO et 012PO.

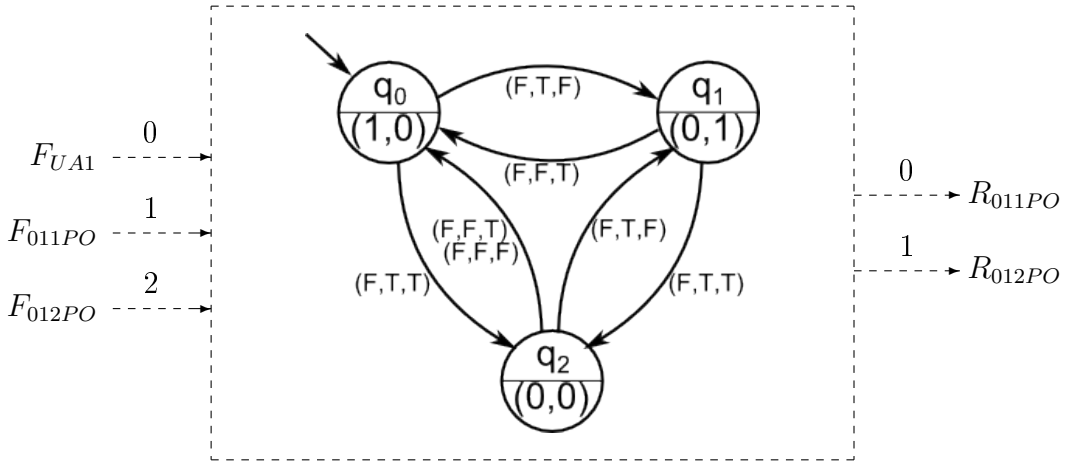

Figure 4.19 – Commutateur S2 (contenant la machine de Moore M_{red1}) modélisant la redondance entre les pompes du système RRA

La stratégie de sauvegarde du système RRA en cas de défaillance est portée par l'UA2 (et l'UA2'). Elle est modélisée par la machine de Moore représentée sur la Figure 4.20. Celle-ci contrôle l'ouverture des deux vannes automatisées (VP207 et VP211) en fonction du statut de *défaillance* du système RRA (entrée numéro 1), sous réserve que la porte UA2 soit non défaillante.

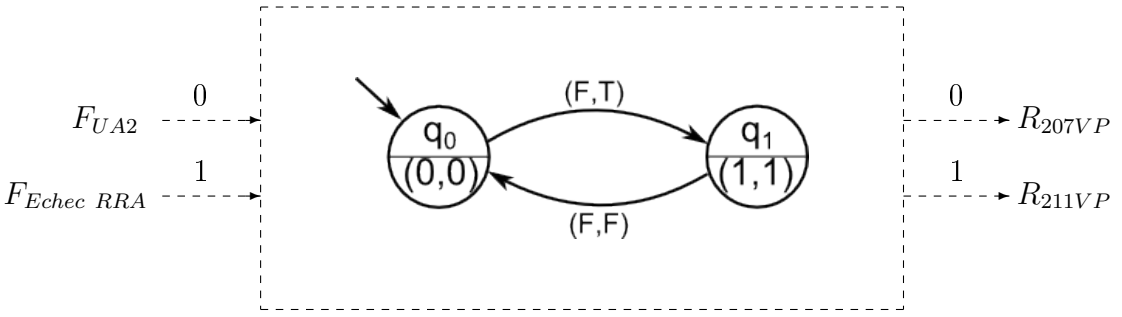

Figure 4.20 – Commutateur S3 (contenant la machine de Moore M_{sauv}) modélisant la stratégie de sauvegarde du RRA

La Figure 4.21 représente la machine de Moore permettant de déterminer dans quel état de tranche la fonction RPR est requise, selon la stratégie décrite par l'UA3 (et l'UA3'). Ainsi, la machine de Moore contrôle le statut de *réquisition* de la fonction RPR, en fonction de l'état de tranche, sous réserve que la porte UA3 soit non défaillante.

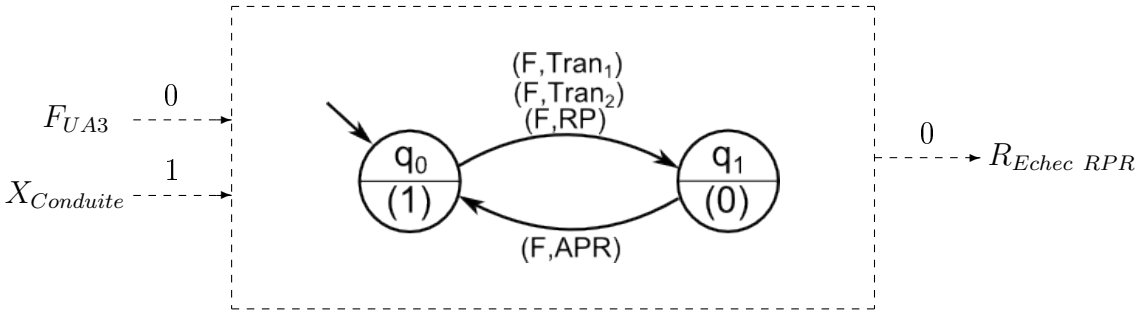

Figure 4.21 – Commutateur S4 (contenant la machine de Moore M_{req2}) contrôlant la réquisition de la fonction RPR

L'UA4 (resp. UA5) décrit dans quel mode la pompe 021PO (resp. 022PO) doit être activée en fonction de son statut de *défaillance* et de celui de 022PO (resp. 021PO). Le commutateur modélisant cette stratégie de reconfiguration pour la pompe 021PO [20] est reporté sur la Figure 4.22.

Cette machine traduit le fait que la pompe 021PO :

- ne doit pas être requise lorsqu'elle est défaillante (ni dans le mode *On*, ni dans le mode *Over*) ;
- doit être requise dans le mode *On* si elle et la pompe 022PO ne sont pas défaillantes ;

20. Pour la pompe 022PO, la même machine de Moore convient également mais les variables d'entrée et de sortie changent. Dans ce cas, les entrées sont dans l'ordre : F_{UA5}, F_{021PO} et F_{022PO} ; et les sorties sont dans l'ordre : $R_{Echec\ PTR\ 2}$ et $R_{Echec\ PTR\ 4}$.

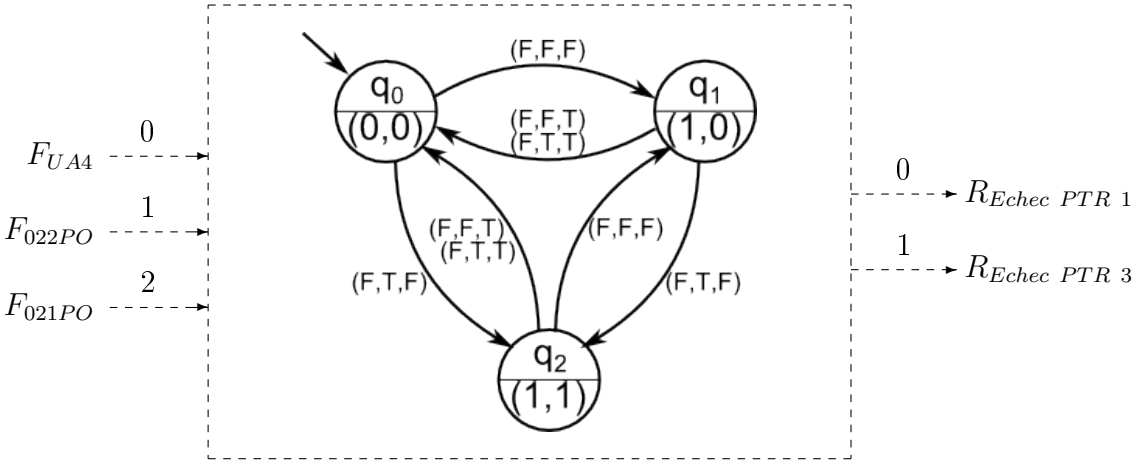

Figure 4.22 – Commutateur S5a (contenant la machine de Moore M_{red2}) modélisant la redondance entre les pompes du système PTR

- doit être requise dans le mode *Over* si elle n'est pas défaillante et la pompe 022PO l'est.

Ainsi, si la feuille modélisant la pompe considérée est sollicitée par une branche de l'arbre (selon l'état de tranche), ce commutateur permet de déterminer dans quel mode elle doit être commutée. Une fois de plus, cette stratégie n'est réalisée qu'en cas de disponibilité de la porte UA4 (resp. UA5).

4.5.4.2 Autres stratégies de reconfiguration

Les stratégies de reconfiguration pour la maintenance des échangeurs thermiques doivent également être modélisées à l'aide de machines de Moore. Nous faisons l'hypothèse que les opérateurs qui appliquent ces stratégies ne commettent pas d'erreur. Les figures 4.23 et 4.24 représentent respectivement les commutateurs modélisant la stratégie de reconfiguration des échangeurs des systèmes RRA et PTR.

Pour le système RRA, la symétrie est parfaite entre la maintenance des deux échangeurs : dès que l'un d'eux est encrassé, il est isolé pour la maintenance (i.e. on ferme la vanne qui l'alimente en eau), jusqu'à ce qu'il soit remis à neuf, ou que l'autre échangeur soit dans un état plus dégradé. En revanche, pour le système PTR, on constate que la maintenance du premier échangeur est prioritaire par rapport à celle du second. Ainsi, à niveau égal d'usure, on isolera toujours le premier échangeur.

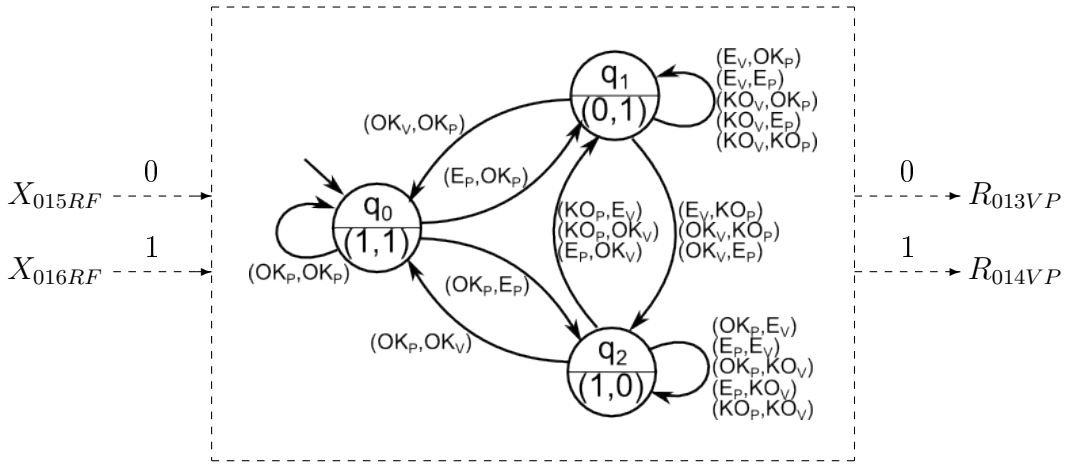

Figure 4.23 – Commutateur S6 (contenant la machine de Moore M_{mtn1}) modélisant la stratégie de maintenance des échangeurs de RRA

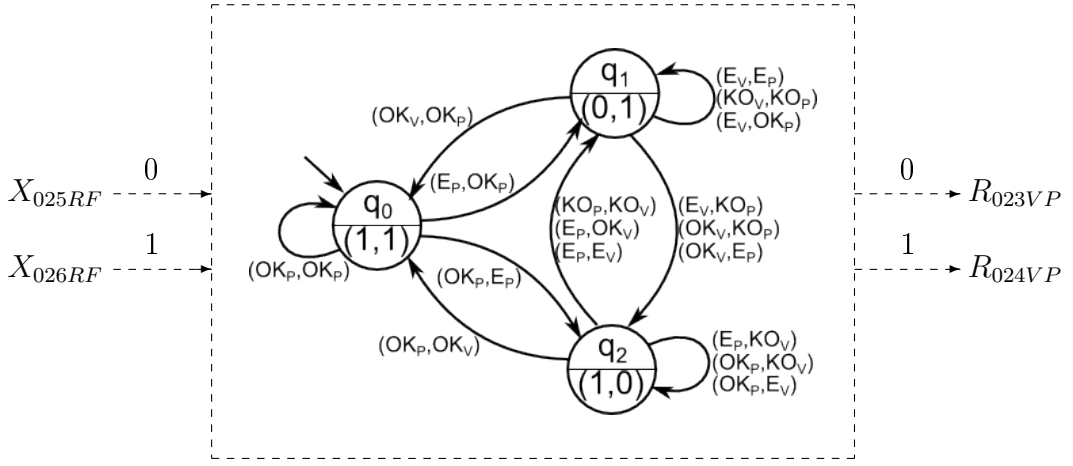

Figure 4.24 – Commutateur S7 (contenant la machine de Moore M_{mtn2}) modélisant la stratégie de maintenance des échangeurs du PTR

Finalement, deux nouvelles machines de Moore sont construites afin de prendre en compte les effets de la position ouverte ou fermée des vannes. Ainsi, pour les vannes manuelles permettant d'isoler les échangeurs, la machine de Moore représentée sur la Figure 4.25 indique que l'échangeur 015RF est traversé par un flux d'eau tant que la vanne 013VP est en position ouverte (le principe est le même pour les couples (016RF,014VP), (025RF,023VP) et (026RF,024VP)).

Pour les vannes automatisées, permettant de provoquer le secours de RRA par PTR, la machine de Moore 4.26 indique que la fonction SRRA n'est requise que lorsque les deux vannes sont en position ouverte. Si l'une des deux est bloquée en position fermée, alors le système PTR n'a aucun moyen de secourir le système RRA.

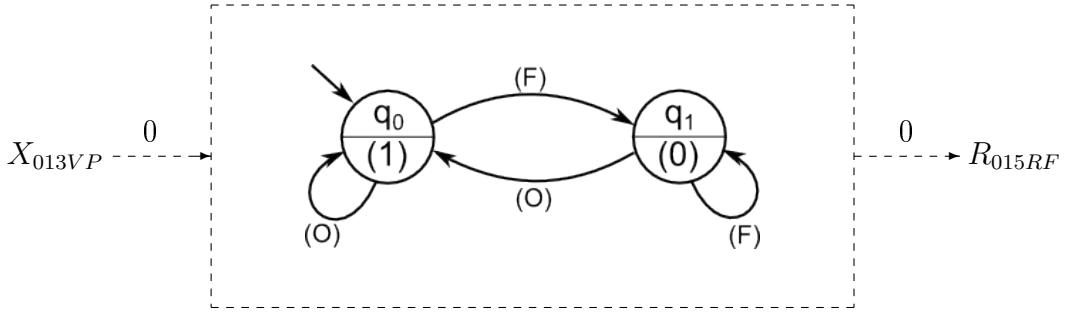

Figure 4.25 – Commutateur S8a (contenant la machine de Moore M_{flux1}) modélisant l'effet des vannes manuelles

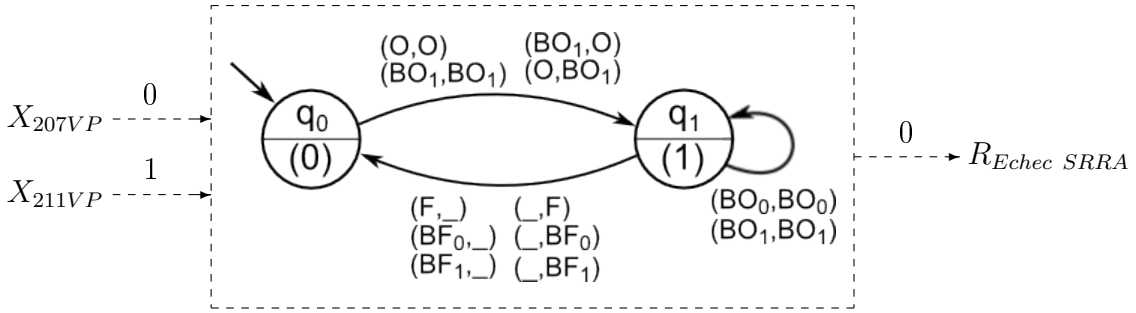

Figure 4.26 – Commutateur S9 (contenant la machine de Moore M_{flux2}) modélisant l'effet des vannes automatisées

4.5.5 Modèle complet

Finalement, les Figure 4.27, 4.28 et 4.29 présentent respectivement les vues *PMC*, *machine de Moore* et *arbre de défaillance enrichi* du modèle complet. On peut vérifier que ce modèle est bien formé, car il respecte les règles de construction définies dans la section 2.3.

Il est important de remarquer que dans la perspective d'une utilisation industrielle du formalisme GBDMP, les experts en charge de la construction des modèles pourront constituer au fur et à mesure une bibliothèque de modèles élémentaires (PMC et machine de Moore), adaptée à leur domaine technique. De cette manière, le temps consacré à la phase de modélisation ainsi que le nombre d'erreur qu'elle génèrerait seraient considérablement diminués.

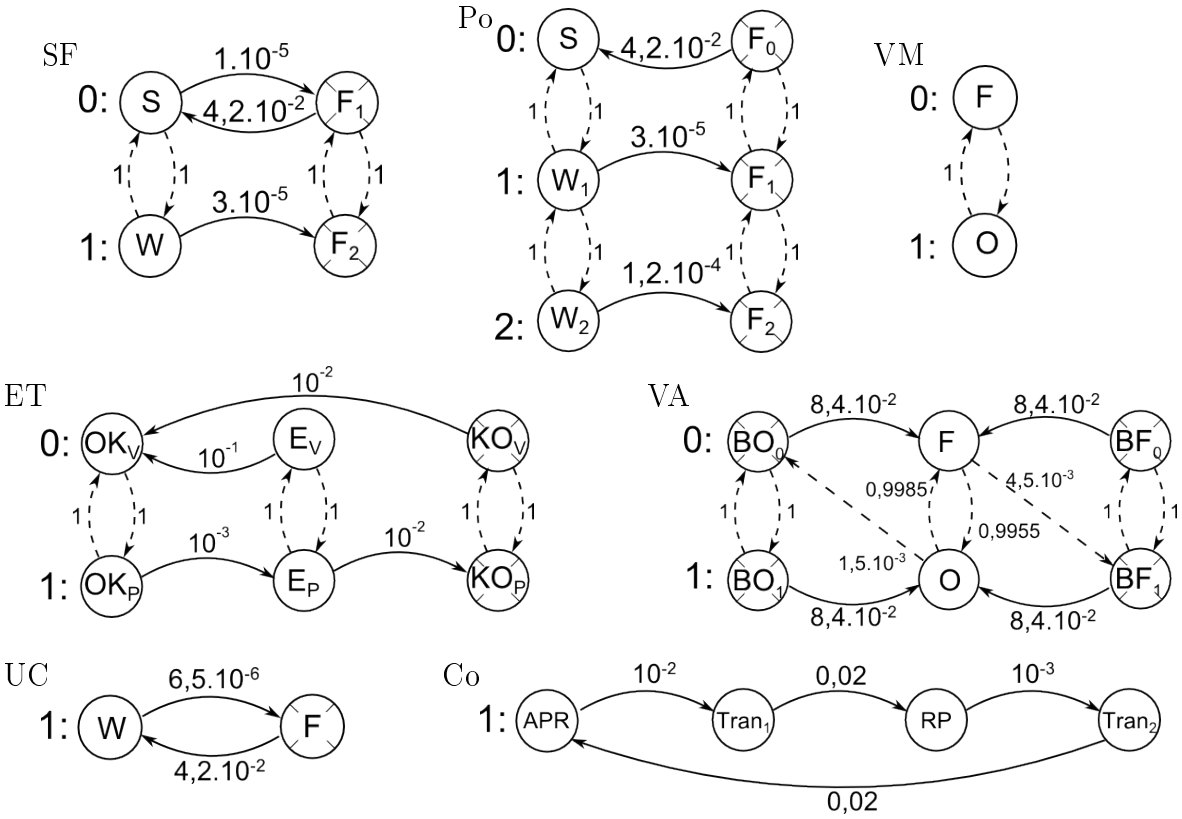

Figure 4.27 – Vue *PMC* du modèle GBDMP pour les systèmes PTR et RRA

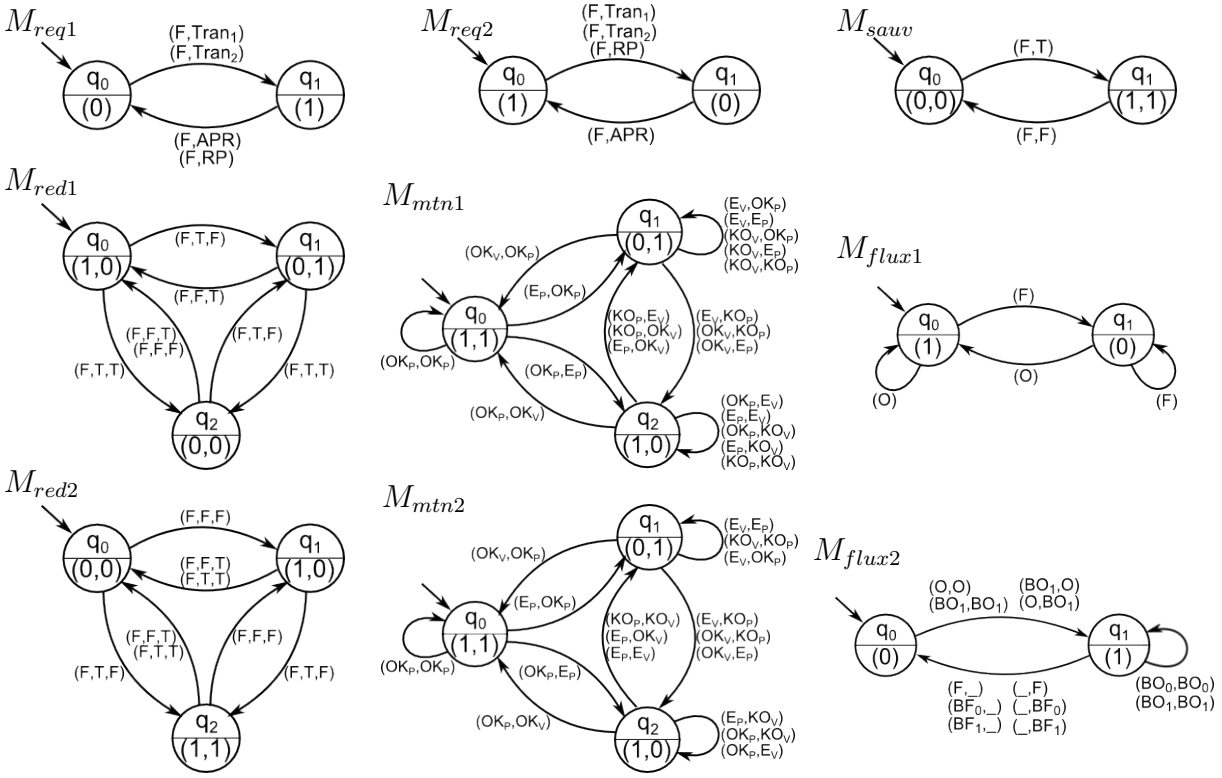

Figure 4.28 – Vue *Machine de Moore* du modèle GBDMP pour les systèmes PTR et RRA

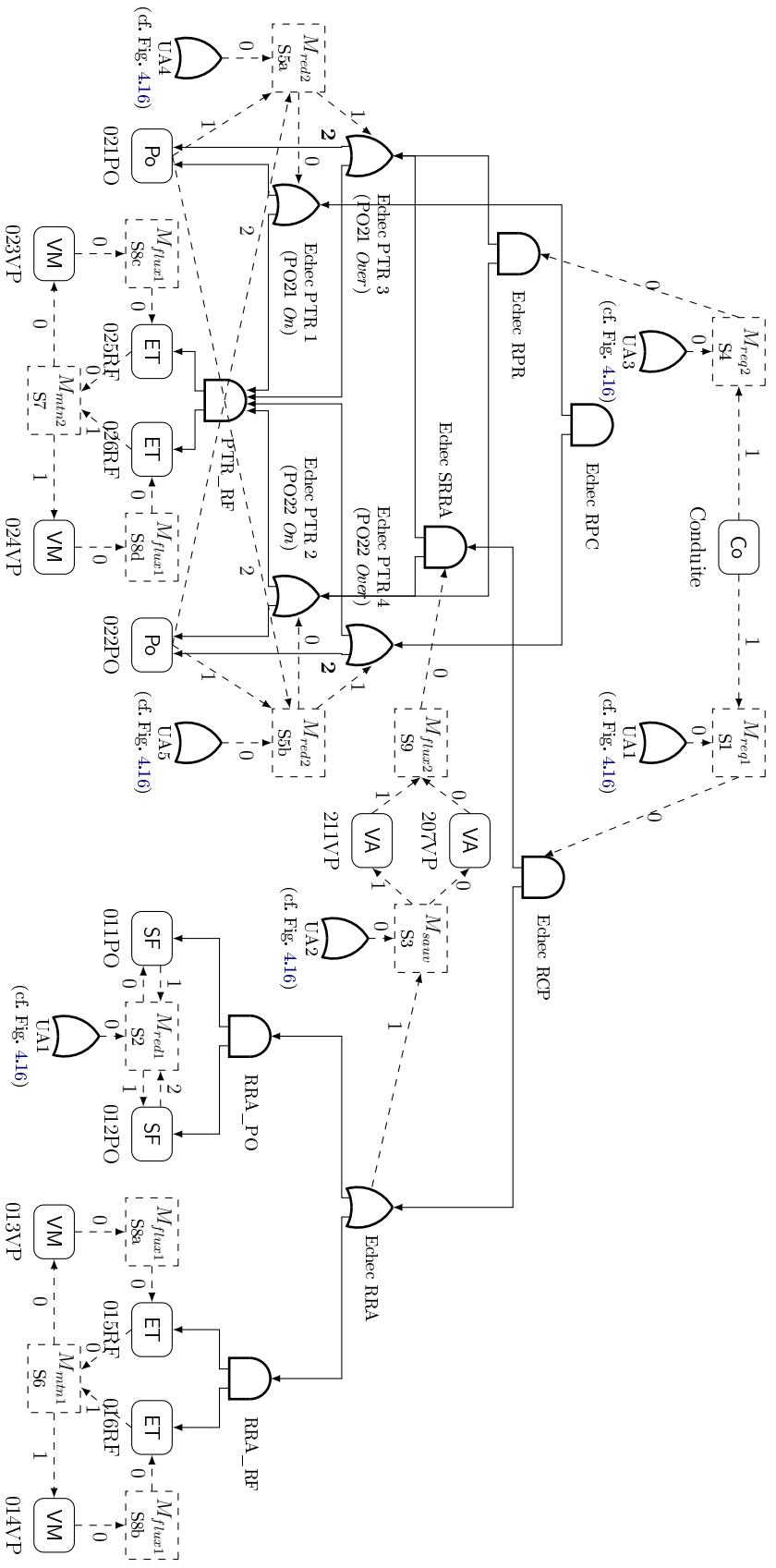

Figure 4.29 – Vue *arbre de défaillance enrichi* du modèle GBDMP pour les systèmes PTR et RRA

4.6 Analyse

Pour terminer ce traitement de cas, nous allons à présent réaliser des analyses basées sur le modèle construit dans la section précédente, en appliquant les deux techniques décrites dans le chapitre 3.

4.6.1 Identification de scénarios critiques

L'algorithme d'extraction des SCM d'un modèle GBDMP (cf. sous-section 3.2.2) permet d'identifier les scénarios de défaillance les plus courts, représentatifs du comportement dysfonctionnel du système.

Selon l'étude que l'on souhaite réaliser, il faut choisir un état de tranche initial ainsi que la fonction cible. Par exemple, on a pu identifier 23 SCM conduisant à la perte de la fonction RPC, en partant de l'état de tranche initial APR [21]. Ces séquences sont reportées dans le Tableau 4.1 [22].

On peut retrouver certains aspects du comportement décrit dans la section 4.4, à travers ces SCM. Par exemple, la séquence $\boldsymbol{f^{022PO}_{W_1 \to F_1}\, e^{Conduite}_{APR \to Tran_1}\, f^{021PO}_{W_1 \to F_1}}$ peut s'expliquer de la manière suivante [23] :

- Défaillance en mode *On* de la pompe 022PO, la pompe 021PO est commutée dans le mode *Over*, afin d'assurer la fonction RPR.
- Changement d'état de tranche : la fonction RCP est requise et est réalisée par le système RRA. Par ailleurs, la fonction RPR n'est plus requise, donc la pompe 021PO est commutée dans le mode *On*, afin d'assurer l'unique fonction RPC.
- Défaillance en mode *On* de la pompe 021PO, la fonction RPC est perdue.

Cette séquence est minimale car seule la présence de l'événement $\boldsymbol{e^{Conduite}_{APR \to Tran_1}}$ explique que la pompe 021PO puisse défaillir en mode *On* après la défaillance de 022PO.

La séquence de coupe $\boldsymbol{f^{021PO}_{W_1 \to F_1}\, f^{CA1}_{W \to F}\, f^{CA4}_{W \to F}\, r^{021PO}_{F_0 \to S}}$ est surprenante car elle se termine par un événement de réparation. Elle est pourtant symptomatique d'un compor-

21. L'outil SAGE a permis d'obtenir ce résultat en un peu plus d'une minute.

22. Pour améliorer la lisibilité, dans la suite, les événements sont préfixés par : $f!$ pour les événements de défaillance à la commutation, f pour les autres événements de défaillance, r pour les événements de réparation et e pour les événements neutres.

23. On rappelle que dans l'état de tranche APR (état de tranche initial), les fonctions RPC et RPR sont requises et sont respectivement réalisées par les pompes 021PO et 022PO.

nombre par longueur	séquences
Longueur 2 2	$f^{021PO}_{W_1\to F_1} f^{022PO}_{W_2\to F_2}$ $f^{022PO}_{W_1\to F_1} f^{021PO}_{W_2\to F_2}$
Longueur 3 5	$f^{BUS1}_{W\to F} f^{BUS2}_{W\to F} f^{021PO}_{W_1\to F_1}$ $f^{BUS2}_{W\to F} f^{BUS1}_{W\to F} f^{021PO}_{W_1\to F_1}$ $f^{CA2}_{W\to F} f^{CA5}_{W\to F} f^{021PO}_{W_1\to F_1}$ $f^{CA5}_{W\to F} f^{CA2}_{W\to F} f^{021PO}_{W_1\to F_1}$ $\boldsymbol{f^{022PO}_{W_1\to F_1} e^{Conduite}_{APR\to Tran_1} f^{021PO}_{W_1\to F_1}}$
Longueur 4 8	$f^{CA1}_{W\to F} f^{CA4}_{W\to F} f^{022PO}_{W_1\to F_1} f^{021PO}_{W_1\to F_1}$ $f^{CA4}_{W\to F} f^{CA1}_{W\to F} f^{022PO}_{W_1\to F_1} f^{021PO}_{W_1\to F_1}$ $\boldsymbol{f^{021PO}_{W_1\to F_1} f^{CA1}_{W\to F} f^{CA4}_{W\to F} r^{021PO}_{F_0\to S}}$ $f^{021PO}_{W_1\to F_1} f^{CA4}_{W\to F} f^{CA1}_{W\to F} r^{021PO}_{F_0\to S}$ $f^{CA1}_{W\to F} f^{021PO}_{W_1\to F_1} f^{CA4}_{W\to F} r^{021PO}_{F_0\to S}$ $f^{CA4}_{W\to F} f^{021PO}_{W_1\to F_1} f^{CA1}_{W\to F} r^{021PO}_{F_0\to S}$ $\boldsymbol{e^{025RF}_{OK_P\to E_P} e^{026RF}_{OK_P\to E_P} f^{026RF}_{E_P\to KO_P} f^{025RF}_{E_P\to KO_P}}$ $\boldsymbol{e^{026RF}_{OK_P\to E_P} e^{025RF}_{OK_P\to E_P} f^{026RF}_{E_P\to KO_P} f^{025RF}_{E_P\to KO_P}}$
Longueur 5 8	$\boldsymbol{f^{021PO}_{W_1\to F_1} f^{BUS1}_{W\to F} f^{BUS2}_{W\to F} r^{021PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}}$ $f^{021PO}_{W_1\to F_1} f^{BUS2}_{W\to F} f^{BUS1}_{W\to F} r^{021PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{022PO}_{W_1\to F_1} f^{BUS1}_{W\to F} f^{BUS2}_{W\to F} r^{022PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{022PO}_{W_1\to F_1} f^{BUS2}_{W\to F} f^{BUS1}_{W\to F} r^{022PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{BUS1}_{W\to F} f^{021PO}_{W_1\to F_1} f^{BUS2}_{W\to F} r^{021PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{BUS2}_{W\to F} f^{021PO}_{W_1\to F_1} f^{BUS1}_{W\to F} r^{021PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{BUS1}_{W\to F} f^{022PO}_{W_1\to F_1} f^{BUS2}_{W\to F} r^{022PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$ $f^{BUS2}_{W\to F} f^{022PO}_{W_1\to F_1} f^{BUS1}_{W\to F} r^{022PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}$

Tableau 4.1 – Ensemble des SCM (de longueur inférieure ou égale à 5) conduisant à la perte de la fonction RPC

tement dysfonctionnel du à l'échec d'une stratégie de reconfiguration. Elle peut en effet s'expliquer ainsi :

- Défaillance de 021PO (dans le mode *On*) : conformément à la stratégie qu'elles appliquent, l'UA4 provoque la désactivation de 021PO, tandis que l'UA5 provoque la commutation de 022PO dans le mode *Over*, de manière à assurer la fonction RPC malgré la défaillance.

- Défaillance de CA1 puis de CA4 : perte de la stratégie appliquée par l'UA4.

- Réparation de 021PO : l'UA5 provoque la commutation de 022PO dans le mode *On*, de sorte que cette pompe n'assure plus que la fonction RPR. Comme la stratégie appliquée par l'UA4 est perdue, la pompe 021PO n'est pas réactivée, donc la fonction RPC est perdue.

Dans la séquence $\boldsymbol{f^{021PO}_{W_1\to F_1} f^{BUS1}_{W\to F} f^{BUS2}_{W\to F} r^{021PO}_{F_0\to S} f^{022PO}_{W_2\to F_2}}$, la défaillance des deux bus de communication provoque non seulement la perte de la stratégie appliquée par l'UA4, mais aussi celle appliquée par l'UA5. Aussi, après la réparation de 021PO, la pompe 022PO reste dans le mode *Over*, et les deux fonctions RPR et RPC sont donc toujours correctement réalisées. Par contre, après la défaillance de 022PO (en mode *Over*), le CC défaillant échoue à remettre la pompe 021PO en service, et les fonctions sont donc perdues. Cette séquence de coupe est bien minimale car, bien que la SCM $\boldsymbol{f^{021PO}_{W_1\to F_1} f^{022PO}_{W_2\to F_2}}$ soit incluse dedans, les composants défaillants à l'issue des deux séquences sont différents.

Finalement, les séquences $\boldsymbol{e^{025RF}_{OK_P\to E_P} e^{026RF}_{OK_P\to E_P} f^{026RF}_{E_P\to KO_P} f^{025RF}_{E_P\to KO_P}}$ et $\boldsymbol{e^{026RF}_{OK_P\to E_P} e^{025RF}_{OK_P\to E_P} f^{026RF}_{E_P\to KO_P} f^{025RF}_{E_P\to KO_P}}$ permettent de constater l'asymétrie entre les échangeurs 025RF et 026RF, sur le plan de leur maintenance. En effet, dans la première séquence, le premier échangeur à s'encrasser est 025RF, dont la maintenance est prioritaire à celle de 026RF, lequel reste donc en service même après son propre encrassement. A l'inverse, dans la seconde séquence, l'encrassement de l'échangeur 025RF survient après celui de 026RF. La maintenance de celui-ci étant prioritaire, 025RF est remis en service, bien qu'il soit toujours encrassé. Aussi, dans les deux cas, seule la défaillance de 026RF peut survenir après cela. Sa maintenance devient alors plus urgente que celle de 025RF, qui est remis en service, et devient à son tour susceptible de défaillir.

En faisant varier la fonction cible et l'état de tranche initial, on peut trouver d'autres SCM caractéristiques du comportement dysfonctionnel du système. Par exemple, parmi les SCM conduisant à l'échec de la fonction RCP en partant d'un état de tranche transitoire, la séquence $\boldsymbol{f^{011PO}_{W_1\to F_1} f^{012PO}_{W_1\to F_1} f^{!207VP}_{F\to BF_1}}$ permet de vérifier que le refus d'ouverture de la vanne 207VP empêche le secours du système RRA par le système PTR. Mais pour cette fonction RCP, la SCM la plus intéressante pour l'analyste est vraisemblablement la séquence $\boldsymbol{f^{CA1}_{W\to F} f^{CA3}_{W\to F} f^{011PO}_{W_1\to F_1}}$. En effet, deux stratégies de redondances sont mises en place pour assurer cette fonction : la redondance passive entre les pompes 011PO et

012PO, et le secours de RRA par PTR. Or, ces deux stratégies sont respectivement appliquées par les couples d'UA redondantes (UA1,UA1') et (UA2,UA2'). Ces deux couples d'UA ont été allouées au même couple de CA (CA1,CA3). C'est pourquoi la défaillance de ces deux CA suffit à perdre les deux stratégies de redondance, après quoi la simple défaillance de 021PO provoque la perte de la fonction RCP. Cette séquence peut être vue comme une faiblesse de l'architecture opérationnelle de CC, qui peut être corrigée en séparant ces deux couples d'UA dans l'allocation.

4.6.2 Disponibilité des fonctions

Pour obtenir des résultats quantitatifs, l'outil SAGE a été utilisé pour traduire le modèle GBDMP dans le langage AltaRica. Puis, l'outil de génération limitée de Chaîne de Markov, développé dans la thèse [Brameret, 2015], a permis d'estimer les valeurs de disponibilité des trois fonctions principales du système.

Une fonction est indisponible seulement si elle n'est pas réalisée alors qu'elle est requise. Pour mesurer l'indisponibilité de la fonction RPR, il faut donc calculer la probabilité qu'elle soit perdue alors que le système est dans l'état de tranche APR. Ce paramétrage se fait aisément dans le modèle traduit en langage AltaRica : il suffit d'écrire dans le bloc modélisant le système, la ligne :

```
observer Boolean unavailability = (PMC_Conduite.X==APR) and (RPR.F==true);
```

De manière analogue, pour calculer l'indisponibilité de la fonction RCP, il faudra écrire la ligne :

```
observer Boolean unavailability = (PMC_Conduite.X==Tran) and (RCP.F==true);
```

La fonction RPC est toujours requise pour le cas d'étude considéré, donc la ligne à écrire est :

```
observer Boolean unavailability = (RPC.F==true);
```

Pour les trois Figures 4.30, 4.31 et 4.32, les trois courbes tracées sur le graphe représentent l'évolution de l'indisponibilité estimée de la fonction considérée, avec ses deux bornes minimales et maximales, au cours d'une mission. Pour notre cas, une mission correspond à un cycle des états de tranche et dure donc 1200 heures. On rappelle que les valeurs approximées sont obtenues en considérant la chaîne réduite sans état puits, et que les bornes sont obtenues en la considérant avec l'état puits.

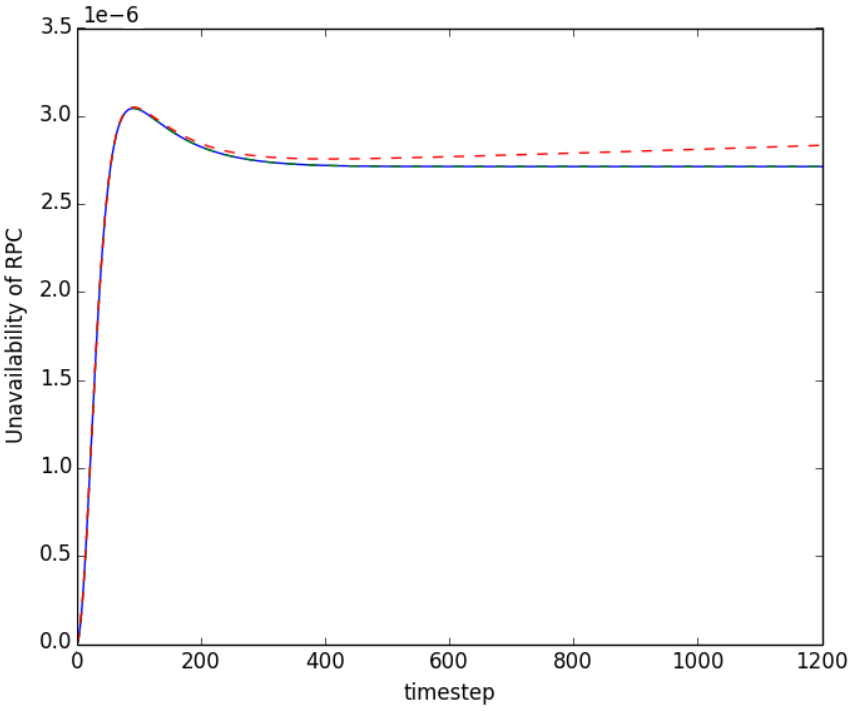

Figure 4.30 – Estimation de l'indisponibilité de la fonction RPC

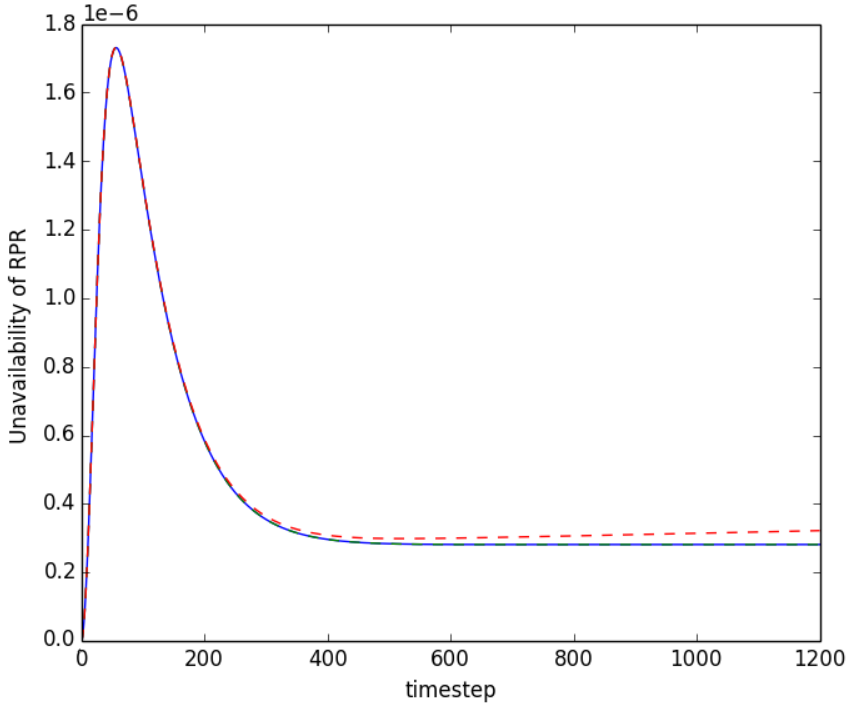

Figure 4.31 – Estimation de l'indisponibilité de la fonction RPR

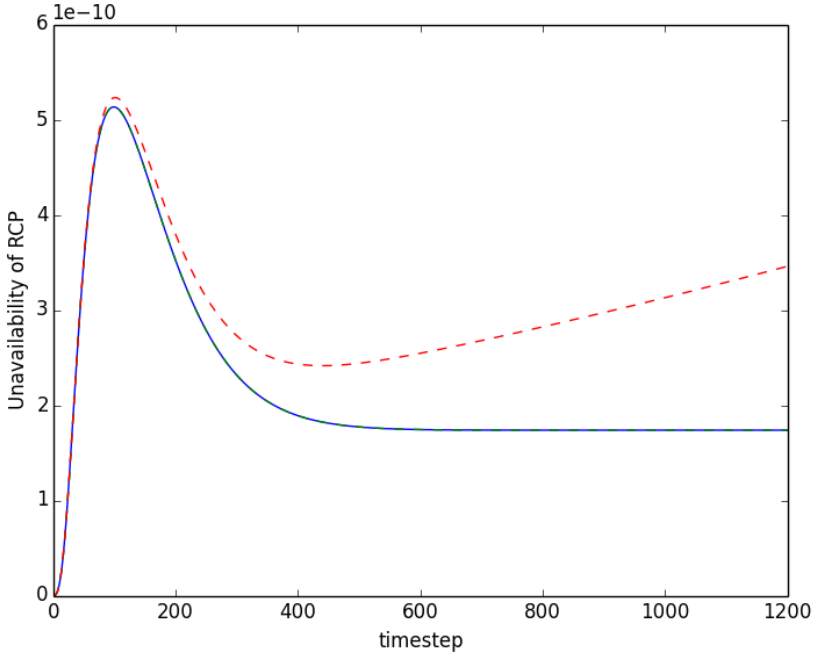

Figure 4.32 – Estimation de l'indisponibilité de la fonction RCP

Sur ces trois courbes, au début de la mission, l'indisponibilité atteint une valeur maximale, car durant les états de tranche APR et Tran, le système est davantage sollicité que pendant l'état de tranche RP. Ce dernier état est prépondérant dans la mission (il dure plus de 80% du temps de mission), ce qui explique la stabilisation rapide des indisponibilités à leur valeur asymptotique. On peut remarquer que l'indisponibilité de la fonction RCP est infime, ce qui n'est pas une surprise car de nombreuses solutions redondantes permettent de la réaliser. De surcroît cette fonction est requise pendant seulement 8% du temps de la mission (en moyenne).

Les valeurs asymptotiques et maximales de ces indicateurs de SdF sont reportées dans le Tableau 4.2. Dans le cadre d'une EPS, ces résultats seraient à confronter avec les valeurs ciblées par les exigences de SdF.

Fonctions		**RPC**	**RPR**	**RCP**
nombre de transitions de la chaîne de Markov		300 000	400 000	1,5 million
temps de calcul (en minutes)		11	16	56
Indisponibilité maximale	Borne haute	$\mathbf{3,050.10^{-6}}$	$\mathbf{1,730.10^{-6}}$	$\mathbf{5,231.10^{-10}}$
	approximation	$\mathbf{3,044.10^{-6}}$	$\mathbf{1,730.10^{-6}}$	$\mathbf{5,133.10^{-10}}$
	borne basse	$\mathbf{3,044.10^{-6}}$	$\mathbf{1,730.10^{-6}}$	$\mathbf{5,133.10^{-10}}$
Indisponibilité asymptotique	Borne haute	$\mathbf{2,834.10^{-6}}$	$\mathbf{3,219.10^{-7}}$	$\mathbf{3,459.10^{-10}}$
	approximation	$\mathbf{2,713.10^{-6}}$	$\mathbf{2,814.10^{-7}}$	$\mathbf{1,738.10^{-10}}$
	Borne basse	$\mathbf{2,713.10^{-6}}$	$\mathbf{2,813.10^{-7}}$	$\mathbf{1,738.10^{-10}}$

Tableau 4.2 – Récapitulatif des paramètres et résultats du calcul de disponibilité

4.7 Bilan

Ce chapitre a permis d'illustrer l'approche développée dans cette thèse sur un cas d'étude représentatif de plusieurs problématiques industrielles. On a montré que la modélisation GBDMP permettait de prendre en compte des stratégies de reconfiguration complexes et variées d'un système. En particulier, on a pu modéliser avec précision des stratégies de redondances, de sauvegarde et de maintenance (préventive et corrective). De plus, le système considéré était multi-phases et comportait des composants multi-états. La modélisation GBDMP a également permis de prendre en compte la perte des stratégies de reconfiguration due aux défaillances de l'architecture opérationnelle de CC.

Les capacités de modélisation des GBDMP ont ainsi permis de qualifier et de quantifier la SdF du système bouclé, c'est à dire en prenant en compte dans l'étude non seulement la partie procédé mais aussi la partie CC.

La réalisation de cette étude de cas a été rendue possible grâce au prototype d'outil SAGE, qui a été développé dans le cadre de cette thèse. Sous réserve qu'une version opérationnelle de cet outil soit industrialisée, on a montré que son utilisation par un ingénieur expert en SdF s'inscrit parfaitement dans un processus de conception d'architecture opérationnelle de CC.

Conclusions & Perspectives

Le présent mémoire de thèse a introduit le formalisme GBDMP (Generalized Boolean logic Driven Markov Processes) pour la modélisation de la SdF des systèmes dynamiques, réparables et reconfigurables. Ce formalisme conserve le principe fondamental et pertinent des BDMP (arbre de défaillance construit sur des processus de Markov en interactions), et le développe afin d'étendre son pouvoir d'expression, pour traduire avec précision les stratégies de reconfiguration du système. Le rôle de ces stratégies est d'exploiter les flexibilités du système (à travers les modes d'opération de ses composants), pour atteindre divers objectifs, notamment pour des raisons de redondances passives et pour des changements de phase de mission. Le formalisme proposé permet en outre de prendre en compte la défaillance des composants responsables de l'application des stratégies de reconfiguration, et donc de traduire les échecs de celles-ci. La définition du formalisme constitue la principale contribution de cette thèse. Une aide à la modélisation GBDMP a également été fournie sous forme d'une réflexion méthodologique à la construction des modèles, illustrée par plusieurs exemples.

Les modèles GBDMP sont destinés à être analysés afin d'obtenir des indicateurs qualitatifs et quantitatifs de la SdF du système étudié. Aussi, deux méthodes d'analyse basées sur un modèle GBDMP ont été décrites dans ce mémoire. La première fournit un résultat qualitatif : l'ensemble des SCM (Séquences de Coupe Minimales). Afin de spécifier ce résultat, le critère de minimalité d'une séquence de coupe d'un système dynamique a été formellement défini dans le cadre de la théorie des langages rationnels. Cette définition étend la portée de celle décrite dans [Chaux et al., 2013]. Un algorithme permettant de générer l'ensemble des SCM à la volée à partir d'un modèle GBDMP a ensuite été proposé. La seconde méthode d'analyse permet d'estimer des indicateurs probabilistes de SdF (notamment la disponibilité du système). L'estimation est calculée sur une chaîne de Markov partielle, générée à partir d'un modèle GBDMP. La méthode de génération partielle de chaîne de Markov a été développée dans le cadre de la thèse [Brameret, 2015], et a été outillée à partir d'un modèle AltaRica. Compte tenu de cet

existant, nous avons décrit comment traduire un modèle GBDMP en langage AltaRica, avant d'utiliser l'outil en question. Deux études comparatives ont finalement montré l'impact, tant qualitatif que quantitatif, des stratégies de reconfiguration et de leurs possibles échecs, sur la SdF du système. Ces aspects n'ayant pu être pris en compte que grâce aux capacités de modélisation du formalisme GBDMP, les études comparatives menées ont ainsi permis de souligner la pertinence du formalisme pour l'analyse de SdF basée sur les modèles.

Finalement, l'approche a été validée expérimentalement sur un cas d'étude représentatif des problématiques industrielles. Afin de traiter un cas de cette dimension, un prototype d'outil logiciel pour la modélisation et l'analyse de SdF a été développé. L'application sur le cas d'étude, a permis en outre d'introduire les processus métiers types permettant d'illustrer l'utilisation des GBDMP dans la perspective d'une valorisation industrielle de ces travaux de thèse.

Au terme de ces travaux se dessinent de nombreuses perspectives. Premièrement, d'un point de vue opérationnel, Il serait intéressant de valoriser ce travail dans l'industrie, par le développement - en qualité industrielle - d'un outil support aux activités d'ingénierie liées aux études de SdF. Le contexte privilégié pour cette industrialisation est vraisemblablement le langage FIGARO [Bouissou et al., 1991], ainsi que les environnements de développement (KB3, Visual Figaro...) et outils associés (YAMS, FIGSEQ...). En effet, ce contexte est, à ce jour, celui qui a porté avec succès le formalisme BDMP. En outre, sa maturité (plus de trois décennies de développement) se traduit par de nombreuses fonctionnalités existantes, indispensables à une utilisation industrielle : gestion de base de modèles, édition graphique, vérificateur de respect des règles de construction... Cette palette de fonctionnalités permettrait en particulier d'envisager le développement d'une méthode de synthèse automatique de GBDMP bien formés.

Par ailleurs, pour l'analyse qualitative d'un modèle GBDMP, l'algorithme proposé dans ce mémoire réalise une exploration en largeur de l'espace d'état. Cependant, l'ordre d'exploration des séquences d'une même longueur est arbitraire. Pour améliorer l'efficacité de la méthode, il pourrait être envisagé de spécifier cet ordre d'exploration, à l'aide d'heuristiques. On pourrait par exemple utiliser le facteur de pertinence probabiliste d'un état, défini pour l'analyse quantitative, afin de guider cette exploration en largeur.

Les analyses de SdF basées sur les modèles sont des opérations coûteuses. Aussi,

avant de passer à cette étape, se pose légitimement la question épineuse de la validité des modèles. Etant donné que le formalisme GBDMP est basé sur des modèles à états (chaînes de Markov, machines de Moore), il serait très intéressant de développer des méthodes de vérification formelle de propriétés dynamiques sur des modèles GBDMP (par exemple en utilisant des techniques de *Model Checking*).

Du point de vue de la modélisation, plusieurs idées pourraient être développées afin d'améliorer la précision des modèles. D'une part, l'hypothèse d'homogénéité des processus de Markov élémentaires du modèle devraient être levée pour pouvoir considérer une plus grande diversité de comportements dysfonctionnels. D'autre part, certains aspects temporels pourraient être intégrés aux modèles en ajoutant des retards à l'occurrence des événements *provoqués*. En effet, dans la pratique, la reconfiguration d'un système n'est pas instantanée.

Finalement, une dernière perspective beaucoup moins immédiate sort du cadre de l'analyse de SdF basée sur les modèles, pour aller vers celui de la synthèse de contrôleur tolérant aux fautes. Pour expliquer cette perspective, il faut remarquer qu'un modèle GBDMP, privé de ses commutateurs, est un Processus de Décision Markovien (cf. l'ouvrage [Puterman, 2014] sur les PDM). La fonction de récompense de ce PDM peut être paramétrée pour mesurer, par exemple, la disponibilité ou la fiabilité du système. Les commutateurs des GBDMP permettraient alors de décrire une politique de décision pour ce PDM. Déterminer la politique de décision qui optimise la fonction de récompense d'un PDM est un problème classique de programmation dynamique. Aussi, déterminer les stratégies de reconfiguration (i.e. les commutateurs) qui optimisent la disponibilité d'un système modélisé à l'aide d'un GBDMP se ramène à ce problème. Comme pour les chaînes de Markov classiques, les PDM sont laborieux à construire. Par conséquent, l'idée de combiner les qualités de modélisation des GBDMP, avec l'efficacité des techniques de détermination de politique optimale pour un PDM, devrait aboutir à une méthode innovante et originale de synthèse de contrôleur tolérant aux fautes.

Glossaire

- **ADD :** Arbre de Défaillance Dynamique
- **APR :** Arrêt Pour Rechargement (phase de mission)
- **BDMP :** *Boolean logic Driven Markov Processes*
- **CA :** Cellule d'Allocation
- **CC :** Contrôle-Commande
- **CONNEXION :** COntrôle-commande Nucléaire Numérique pour l'EXport et la rénovatION
- **DF :** Diagramme Fonctionnel
- **DS :** Demande de Secours
- **EI :** Etat Indisponible
- **FIFO :** *First In First Out*
- **GBDMP :** *Generalized Boolean logic Driven Markov Processes*
- **GTS :** *Guarded Transition System*
- **OE :** Ordre d'Enclenchement
- **OF :** Ordre de Fermeture
- **OO :** Ordre d'Ouverture
- **PMC :** Processus de Markov Commuté
- **PMP :** Processus de Markov Piloté
- **PTR :** Traitement et Refroidissement des Piscines (système)

- **RCP :** Refroidissement du Circuit Primaire (fonction)
- **RP :** Réacteur en Puissance (phase de mission)
- **RPC :** Refroidissement de la Piscine Combustible (fonction)
- **RPR :** Refroidissement de la Piscine du Réacteur (fonction)
- **RRA :** Refroidissement du Réacteur à l'Arrêt, cuve fermée (système)
- **SAGE :** *Safety Analysis in a GBDMP Environment*
- **SCM :** Séquence de Coupe Minimale
- **SD :** Symptôme de Défaut
- **SdF :** Sûreté de Fonctionnement
- **SE :** Système Elémentaire
- **SR :** Sur-Régime
- **SRRA :** Secours du RRA
- **TOR :** Tout Ou Rien
- **Tran :** Transitoire (phase de mission)
- **UA :** Unité d'Allocation

Notations

Dans cette thèse, de nombreux formalismes basés sur les notions d'états et de transitions sont introduits. Les notations qui seront employées dans ce document, pour chacun de ces formalismes, sont répertoriées ici.

Notations relatives aux automates à états fini et aux machines de Moore

$\mathcal{A} =< \Sigma, Q, Q_0, Q_M, \delta >$ est un automate à états fini :

- Σ est l'*alphabet* (i.e. un ensemble de symboles).
  - u, u', u_1 et u_2 sont quatre symboles quelconques de Σ.
  - $\sigma, \sigma', \sigma_1$ et σ_2 sont quatre exemples de séquences quelconques construites par concaténation finie de symboles de Σ.
- Q est l'ensemble fini des états.
  - q, q', q_1 et q_2 sont quatre états quelconques de Q.
  - $Q_0 \subseteq Q$ est le sous-ensemble des états dits *initiaux* (s'il y en a plusieurs).
  - $q_0 \in Q$ est l'état *initial* (s'il est unique).
  - $Q_M \subseteq Q$ est le sous-ensemble des états dits *marqués* (ou finaux).
- δ désigne la fonction de transition.

Une machine de Moore est un automate à entrées/sorties. De ce fait, les notations ressemblent à celles d'un automate. $M =< Q^M, q_0^M, \Sigma_I^M, \Sigma_O^M, trans^M, out^M >$ est une machine de Moore :

- Σ_I^M et Σ_O^M sont les *alphabets* respectivement d'entrée et de sortie.

  - Dans le cadre de ce travail, les symboles d'entrée des machines de Moore seront des familles de $\boldsymbol{a}$ valeurs ($\boldsymbol{a} \in \mathbb{N}^*$). Aussi, $\boldsymbol{i} = (\boldsymbol{i_k})_{\boldsymbol{k} \in [\![\boldsymbol{0},\boldsymbol{a}]\!]}$ et $\boldsymbol{i'} = (\boldsymbol{i'_k})_{\boldsymbol{k} \in [\![\boldsymbol{0},\boldsymbol{a}]\!]}$ sont deux symboles quelconques de $\boldsymbol{\Sigma_I}$.
  - Dans le cadre de ce travail, les symboles de sortie des machines de Moore seront des familles de $\boldsymbol{b}$ valeurs ($\boldsymbol{b} \in \mathbb{N}^*$). Aussi, $\boldsymbol{o} = (\boldsymbol{o_k})_{\boldsymbol{k} \in [\![\boldsymbol{0},\boldsymbol{b}]\!]}$ et $\boldsymbol{o'} = (\boldsymbol{o'_k})_{\boldsymbol{k} \in [\![\boldsymbol{0},\boldsymbol{b}]\!]}$ sont deux symboles quelconques de $\boldsymbol{\Sigma_O}$.

- $\boldsymbol{Q^M}$ est l'ensemble fini des états.
  - $\boldsymbol{q}, \boldsymbol{q'}, \boldsymbol{q_1}$ et $\boldsymbol{q_2}$ sont quatre états quelconques de $\boldsymbol{Q^M}$.
  - $\boldsymbol{q_0} \in \boldsymbol{Q^M}$ est l'état *initial* (s'il est unique).
- $\boldsymbol{trans^M}$ est la fonction de transition (équivalent à $\boldsymbol{\delta}$ pour les automates).
- $\boldsymbol{out^M}$ est la fonction de sortie.
- $\mathbb{M}$ désigne l'ensemble de toutes les machines de Moore.

Notations relatives aux processus de Markov

$\boldsymbol{\mathcal{Z}} =< \boldsymbol{\mathcal{X}}, \boldsymbol{A}, \boldsymbol{p0} >$ est une chaîne de Markov :

- $\boldsymbol{\mathcal{X}}$ est l'ensemble fini des états.
  - $\boldsymbol{x}$ et $\boldsymbol{y}$ sont deux états quelconques de $\boldsymbol{\mathcal{X}}$.
- $\boldsymbol{A}$ est la matrice des *taux de transitions* entre les états de $\boldsymbol{\mathcal{X}}$.
  - $\boldsymbol{o}$ et $\boldsymbol{d}$ sont deux états de $\boldsymbol{\mathcal{X}}$ tels que $\boldsymbol{A(o,d)} \neq \boldsymbol{0}$. On dira qu'ils sont respectivement *origine* et *destination* d'une *transition*.
- $\boldsymbol{p0}$ est la distribution initiale de probabilité dans les états de $\boldsymbol{\mathcal{X}}$.

$\boldsymbol{P} =< \boldsymbol{\mathcal{Z}_0}, \boldsymbol{\mathcal{Z}_1}, \boldsymbol{\mathcal{X}_F}, \boldsymbol{f_{0\to1}}, \boldsymbol{f_{1\to0}} >$ est un Processus de Markov Piloté :

- $\boldsymbol{\mathcal{Z}_0}$ et $\boldsymbol{\mathcal{Z}_1}$ sont deux chaînes de Markov.
- $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}_0} \cup \boldsymbol{\mathcal{X}_1}$ est l'union des ensembles d'états des deux chaînes de Markov $\boldsymbol{\mathcal{Z}_0}$ et $\boldsymbol{\mathcal{Z}_1}$.
  - $\boldsymbol{\mathcal{X}_F} \subseteq \boldsymbol{\mathcal{X}}$ est le sous-ensemble des états dits *défaillants*.

- $\boldsymbol{f_{0\to1}}$ et $\boldsymbol{f_{1\to0}}$ sont les fonctions de transfert.

$\boldsymbol{P} =< (\boldsymbol{\mathcal{Z}_i^P})_{0\leq i<k}, \boldsymbol{\mathcal{X}_F^P}, (\boldsymbol{f_{i\to j}^P})_{(i,j)\in[\![0,k-1]\!]^2} >$ est un Processus de Markov Commuté à $\boldsymbol{k}$ modes ($\boldsymbol{k} \in \mathbb{N}^*$) :

- $(\boldsymbol{\mathcal{Z}_i^P})_{0\leq i<k}$ est une famille de chaînes de Markov.
- $\boldsymbol{\mathcal{X}^P} = \bigcup_{i=0}^{k-1} \boldsymbol{\mathcal{X}_i^P}$ est l'union des ensembles d'états des chaînes de Markov de $(\boldsymbol{\mathcal{Z}_i^P})_{0\leq i<k}$.
  - $\boldsymbol{\mathcal{X}_F^P} \subseteq \boldsymbol{\mathcal{X}^P}$ est le sous-ensemble des états dits *défaillants*.
- $(\boldsymbol{f_{i\to j}^P})_{(i,j)\in[\![0,k-1]\!]^2}$ est une famille de fonctions de transfert.
- $\mathbb{P}$ désigne l'ensemble de toutes les machines de Processus de Markov Pilotés.

Notations relatives aux Systèmes de Transitions Gardées

$< \mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{A}, i >$ est un Système de Transitions Gardées :

- $\mathcal{V}$ est l'ensemble des variables.
  - v est une variable quelconque de $\mathcal{V}$.
  - $v := \boldsymbol{E}$ est une *assignation* ($\boldsymbol{E}$ est une valeur ou une expression mathématique permettant d'en calculer une).
  - i est une assignation particulière des variables $\mathcal{V}$.
  - $\mathcal{V}_{\boldsymbol{F}} \subseteq \mathcal{V}$ est le sous-ensemble des variables de flux.
  - $\mathcal{V}_{\boldsymbol{S}} \subseteq \mathcal{V}$ est le sous-ensemble des variables d'état.
- $\mathcal{E}$ est l'alphabet des *événements*.
  - e est un événement quelconque de $\mathcal{E}$.
- $\mathcal{T}$ est un ensemble de *transitions*, i.e. de 3-tuple $< e, \mathcal{G}, \mathcal{P} >$.
  - $\mathcal{G}$ est la *garde*.
  - $\mathcal{P}$ est la *post-condition*.
- $\mathcal{A}$ est l'*assertion*.

Bibliographie

[IEE, 1990] (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84.

[SAE, 1996] (1996). SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.

[INC, 2007] (2007). *Systems Engineering Vision 2020*. International Council on Systems Engineering, Seattle, INCOSE Technical Operations edition. TP-2004-004-02, Version 2.03.

[Abadi and Cardelli, 1996] Abadi, M. and Cardelli, L. (1996). *A theory of objects*. Springer New York.

[Adeline et al., 2010] Adeline, R., Cardoso, J., Darfeuil, P., Humbert, S., and Seguin, C. (September 2010). Toward a methodology for the altarica modelling of multi-physical systems. In *Proc. 19th European Safety & Reliability Conference (ESREL'10)*, pages 190–197, Rhodes (Greece).

[Ajmone Marsan et al., 1994] Ajmone Marsan, M., Balbo, G., Donatelli, S., Franceschinis, G., and Conte, G. (1994). *Modelling with generalized stochastic Petri nets*. John Wiley & Sons.

[Alcaraz-Mejia et al., 2006] Alcaraz-Mejia, M., Lopez-Mellado, E., and Ramirez-Trevio, A. (April 2006). Fault recovery of manufacturing systems based on controller reconfiguration. In *Proc. IEEE/SMC International Conference on System of Systems Engineering*, Los Angeles (California - USA). 6 pages.

[Andrews and Moss, 1993] Andrews, J.-D. and Moss, T.-R. (1993). *reliability and Risk Assessment*. John Wiley & Sons.

[Batteux et al., 2013] Batteux, M., Prosvirnova, T., Rauzy, A., and Kloul, L. (July 2013). The altarica 3.0 project for model-based safety assessment. In *Proc. 11th*

IEEE International Conference on Industrial Informatics (INDIN'2013), pages 741–746, Bochum (Germany).

[Benazouz and Faure, 2015] Benazouz, M. and Faure, J. (2015). Safety-Level Aware Bin-Packing Approach for Control Functions Assignment. In *The 15th IFAC Symposium on Information Control Problems (INCOM 2015)*, Ottawa (Canada).

[Berge, 1973] Berge, C. (1973). *Graphs et Hypergraphs*. Elsevier.

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

[Blodget et al., 2003] Blodget, B., McMillan, S., and Lysaght, P. (March 2003). A light-weight approach for embedded reconfiguration of FPGAs. In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE'2003)*, pages 399–400, Munich (Germany).

[Bobbio and Raiteri, 2004] Bobbio, A. and Raiteri, D.-C. (January 2004). Parametric Fault Trees with dynamic gates and repair boxes. In *Proc. Reliability and Maintainability Symposium (RAMS'2004)*, pages 459–465, Los Angeles (California - USA).

[Boiteau et al., 2006] Boiteau, M., Dutuit, Y., Rauzy, A., and Signoret, J.-P. (2006). The AltaRica data-flow language in use : modeling of production availability of a multi-state system. *Reliability Engineering & System Safety*, 91(7) :747–755.

[Bon and Collet, 1994] Bon, J.-L. and Collet, J. (1994). An algorithm in order to implement reliability exponential approximations. *Reliability Engineering & System Safety*, 43(3) :263–268.

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

[Bouissou, 2005] Bouissou, M. (2005). Dix petits problèmes de modélisation (en sûreté de fonctionnement des systèmes). *Revue Phoebus*, 34. 13 pages.

[Bouissou, 2007] Bouissou, M. (June 2007). A generalization of Dynamic Fault Trees through Boolean logic Driven Markov Processes (BDMP). In *Proc. 16th European Safety & Reliability Conference (ESREL'07)*, Stavanger (Norway). 7 pages.

[Bouissou, 2009] Bouissou, M. (June 2009). Using BDMP (Boolean logic Driven Markov Processes) for multi-state system analysis. In *Proc. Mathematical Methods in Reliability (MMR 2009)*, Moscow (Russia). 4 pages.

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

[Bouissou et al., 1991] Bouissou, M., Bouhadana, H., Bannelier, M., and Villatte, N. (1991). Knowledge modelling and reliability processing : Presentation of the FIGARO language and associated tools. In *Proc. 10th International Conference on Computer Safety, Reliability and Security (SafeComp'91)*, pages 69–75, Trondheim (Norway).

[Bouissou and Dutuit, 2004] Bouissou, M. and Dutuit, Y. (June 2004). Reliability analysis of a dynamic phased mission system. In *Proc. Mathematical Methods in Reliability (MMR 2004)*, Santa Fe (New Mexico - USA). 4 pages.

[Bouissou and Lefebvre, 2002] Bouissou, M. and Lefebvre, Y. (January 2002). A path-based algorithm to evaluate asymptotic unavailability for large markov models. In *Proc. Reliability and Maintainability Symposium (RAMS 2002)*, pages 32–39, Seattle (Washington - USA).

[Brameret, 2015] Brameret, P.-A. (2015). *Assessment of reliability indicators from automatically generated partial Markov chains.* PhD thesis, Ecole Normale Supérieure de Cachan, Cachan (France).

[Brameret et al., 2015] Brameret, P.-A., Rauzy, A., and Roussel, J.-M. (2015). Automated generation of partial Markov chain from high level descriptions. *Reliability Engineering & System Safety*, 139 :179–187.

[Bryant, 1986] Bryant, R.-E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8) :677–691.

[Burdick et al., 1977] Burdick, G.-R., Fussell, J.-B., Rasmuson, D.-M., and Wilson, J.-R. (1977). Phased mission analysis : A review of new developments and an application. *IEEE Transactions on Reliability*, R-26 :43–49.

[Cancila et al., 2009] Cancila, D., Terrier, F., Belmonte, F., Dubois, H., Espinoza, H., Gérard, S., and Cuccuru, A. (October 2009). SOPHIA : a modeling language for model-based safety engineering. In *Proc. Model-based Architecting of Cyber-Physical and Embedded Systems (ACES-MB 2009)*, pages 11–25, Denver (Colorado - USA).

[Carer et al., 2002] Carer, P., Bellvis, J., Bouissou, M., Domergue, J., and Pestourie, J. (September 2002). A new method for reliability assessment of electrical power supplies with standby redundancies. In *Proc. 7th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, Napoli (Italy). 6 pages.

[Chaux, 2013] Chaux, P.-Y. (2013). *Formalisation de la cohérence et calcul des Séquences de Coupe Minimales pour les systèmes binaires dynamiques réparables.* PhD thesis, Ecole Normale Supérieure de Cachan, Cachan (France).

[Chaux et al., 2012] Chaux, P.-Y., Roussel, J.-M., Lesage, J.-J., Deleuze, G., and Bouissou, M. (June 2012). Systematic extraction of minimal cut sequences from a BDMP model. In *Proc. 21th European Safety & Reliability Conference (ESREL'12)*, Helsinki (Finland). session 16B, 8 pages.

[Chaux et al., 2011] Chaux, P.-Y., Roussel, J.-M., Lesage, J.-J., Deleuze, G., and Bouissou, M. (September 2011). Qualitative analysis of a bdmp by finite automaton. In *Proc. 20th European Safety & Reliability Conference (ESREL'11)*, pages 2055–2057, Troyes (France).

[Chaux et al., 2013] Chaux, P.-Y., Roussel, J.-M., Lesage, J.-J., Deleuze, G., and Bouissou, M. (September 2013). Towards an unified definition of minimal cut sequences. In *Proc. 4th International Workshop on Dependable Control of Discrete Systems (DCDS 2013)*, volume 4 (1), York (UK). 6 pages.

[Codetta-Raiteri, 2005] Codetta-Raiteri, D. (2005). 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.

[Crow, 1975] Crow, L.-H. (1975). Reliability Analysis for Complex, Repairable Systems. Technical report, U.S. Army Material Systems Analysis Activity.

[David et al., 2010] David, P., Idasiak, V., and Kratz, F. (2010). Reliability study of complex physical systems using SysML. *Reliability Engineering & System Safety*, 95(4) :431–450.

[Dijkstra, 1959] Dijkstra, E.-W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1) :269–271.

[Distefano and Liudong, 2006] Distefano, S. and Liudong, X. (January 2006). A new approach to modeling the system reliability : dynamic reliability block diagrams. In *Proc. Reliability and Maintainability Symposium (RAMS06)*, pages 189–195, Newport Beach (California - USA).

[Dugan et al., 1992] Dugan, J.-B., Bavuso, S.-J., and Boyd, M.-A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3) :363–377.

[Dugan et al., 1993] Dugan, J.-B., Bavuso, S.-J., and Boyd, M.-A. (1993). Fault trees and Markov models for reliability analysis of fault-tolerant digital systems . *Reliability Engineering & System Safety*, 39(3) :291–307.

[Dugan et al., 1990] Dugan, J.-B., Bavuso, S.-J., and Boyd, M.-A. (January 1990). Fault trees and sequence dependencies. In *Proc. Reliability and Maintainability Symposium (RAMS1990)*, pages 286–293, Los Angeles (California - USA).

[Durga Rao et al., 2009] Durga Rao, K., Gopika, V., Sanyasi Rao, V.-V.-S., Kushwaha, H.-S., Verma, A.-K., and Srividya, A. (2009). Dynamic Fault Tree analysis using Monte Carlo simulation in probabilistic safety assessment. *Reliability Engineering & System Safety*, 94(4) :872–883.

[Faraut et al., 2010] Faraut, G., Piétrac, L., and Niel, E. (October 2010). Control law synthesis and reconfiguration using SCT. In *Proc. Conference on Control and Fault-Tolerant Systems (SysTol 2010)*, pages 576–581, Nice (France).

[Feiler and Rugina, 2007] Feiler, P.-H. and Rugina, A. (2007). Dependability modeling with the Architecture Analysis & Design Language (AADL). Technical report, DTIC Document.

[Fourneau et al., 2007] Fourneau, J.-M., Pekergin, N., and Younes, S. (2007). Censoring Markov chains and stochastic bounds. In *Formal Methods and Stochastic Models for Performance Evaluation*, pages 213–227. Springer.

[Fukuda, 1991] Fukuda, M. (1991). *Reliability and degradation of semiconductor lasers and LEDs*. Artech House.

[Fussell, 1976] Fussell, J.-B. (1976). Fault tree analysis : concepts and techniques. In *Generic Techniques In Systems Reliability Assessment*, pages 133–162. Noordhoff.

[Gansner and North, 2000] Gansner, E. and North, S. (2000). An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11) :1203–1233.

[Garrett et al., 1995] Garrett, C.-J., Guarro, S.-B., and Apostolakis, G.-E. (1995). The Dynamic Flowgraph Methodology for assessing the dependability of embedded software systems. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5) :824–840.

[Golumbic, 2004] Golumbic, M.-C. (2004). *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier.

[Gonzalez Berlanga et al., 2008] Gonzalez Berlanga, S.-I., Niel, E., Zouari, B., and Blanquart, J.-P. (July 2008). A contribution to the validation of operating mode switching : application to satellite. In *Proc. 17th IFAC World Congress*, volume 17, pages 6566–6571, Seoul (South Korea).

[Gudemann and Ortmeier, 2010] Gudemann, M. and Ortmeier, F. (November 2010). A Framework for Qualitative and Quantitative Formal Model-Based Safety Analysis. In *Proc. IEEE 12th International Symposium on High-Assurance Systems Engineering (HASE 2010)*, pages 132–141, San Jose (California - USA).

[Ionescu et al., 2014] Ionescu, D.-R., Brinzei, N., and Pétin, J.-F. (September 2014). Assessment of the critical events sequences of systems by means of probabilistic lan-

guages. In *Proc. 23th European Safety & Reliability Conference (ESREL'14)*, pages 217–224, Wroclaw (Poland).

[Joshi et al., 2007] Joshi, A., Binns, P., and Vestal, S. (June 2007). Automatic generation of Fault Trees from AADL models. In *Proc. Dependable Systems Networks : Workshop on Architecting Dependable Systems (DSN-WADS 2007)*, Edinburgh (Scotland - UK). Session D (7 pages).

[Joshi et al., 2006] Joshi, A., Heimdahl, M.-P., Miller, S.-P., and Whalen, M.-W. (February 2006). Model-Based Safety Analysis. Technical Report NASA/CR-2006-213953, NASA.

[Kriaa et al., 2012] Kriaa, S., Bouissou, M., and Pietre-Cambacedes, L. (October 2012). Modeling the stuxnet attack with BDMP : Towards more formal risk assessments. In *Proc. 7th International Conference on Risk and Security of Internet and Systems (CRiSIS)*, Cork (Ireland). 8 pages.

[Laprie et al., 1995] Laprie, J.-C., Arlat, J., Blanquart, J.-P., Costes, A., Crouzet, Y., Deswarte, Y., Fabre, J.-C., Guillermain, H., Kaâniche, M., Mazet, C., Powell, D., Rabéjac, C., and Thévenod, P. (1995). *Guide de la sûreté de fonctionnement*. Cépaduès.

[Lemattre, 2013] Lemattre, T. (2013). *Allocation de fonctions de commande de systèmes critiques par recherche d'atteignabilité dans un réseau d'automates communicants*. PhD thesis, Ecole Normale Supérieure de Cachan.

[Lisnianski and Levitin, 2003] Lisnianski, A. and Levitin, G. (2003). *Multi-state system reliability : assessment, optimization and applications*. World Scientific.

[Liu et al., 2007] Liu, D., Zhang, C., Xing, W., Li, R., and Li, H. (2007). Quantification of cut sequence set for fault tree analysis. In *High performance computing and communications*, pages 755–765. Springer.

[Markov, 1971] Markov, A.-A. (1971). Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In Howard, R., editor, *Dynamic Probabilistic Systems (Volume I : Markov Models)*, pages 552–577. John Wiley & Sons.

[Marseguerra et al., 1998] Marseguerra, M., Zio, E., Devooght, J., and Labeau, P.-E. (1998). A concept paper on dynamic reliability via monte carlo simulation. *Mathematics and Computers in Simulation*, 47(2-5) :371–382.

[Meduna, 2012] Meduna, A. (2012). *Automata and languages : theory and applications*. Springer.

[Meister and Persoz, 2000] Meister, E. and Persoz, M. (2000). Contribution de la mécanique probabiliste à l'intégrité des cuves des réacteurs nucléaires. *Revue Française de Mécanique (RFM)*, 4 :247–252.

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

[Meshkat et al., 2002] Meshkat, L., Dugan, J.-B., and Andrews, J.-D. (2002). Dependability analysis of systems with on-demand and active failure modes, using Dynamic Fault Trees. *IEEE Transactions on Reliability*, 51(2) :240–251.

[Meshkat et al., 2003] Meshkat, L., Xing, L., Donohue, S.-K., and Ou, Y. (July 2003). An overview of the phase-modular fault tree approach to phased mission system analysis. In *Proc. International Conference on Space Mission Challenges for Information Technology (SMC-IT 2003)*, Pasadena (California - USA). 10 pages.

[Molloy, 1982] Molloy, M.-K. (1982). Performance analysis using Stochastic Petri Nets. *IEEE Transactions on Computers*, 100(9) :913–917.

[Moore, 1956] Moore, E.-F. (1956). Gedanken-experiments on sequential machines. *Annals of Mathematical Studies*, 34 :129–153.

[Moore and Shannon, 1956] Moore, E.-F. and Shannon, C.-E. (1956). Reliable circuits using less reliable relays. *Journal of the Franklin Institute*, 262(4) :281–297.

[Muntz et al., 1989] Muntz, R.-R., De Souza e Silva, E., and Goyal, A. (1989). Bounding availability of repairable computer systems. *IEEE Transactions on Computers*, 38(12) :1714–1723.

[Murata, 1989] Murata, T. (1989). Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :pp. 541–580.

[Nasri et al., 2015] Nasri, I., Pétin, J.-F., and Bicking, F. (September 2015). An integer coded genetic algorithm based on a replacement procedure for designing operational control architectures of critical systems. In *Proc. 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2015)*, Luxembourg (Luxembourg). 6 pages.

[Nke and Lunze, 2011] Nke, Y. and Lunze, J. (June 2011). Online control reconfiguration for a faulty manufacturing process. In *Proc. 3rd International Workshop on Dependable Control of Discrete Systems (DCDS 2011)*, pages 19–24, Saarbrucken (Germany).

[Noble et al., 1999] Noble, J., Taivalsaari, A., and Moore, I. (1999). *Prototype-Based Programming : Concepts, Languages and Applications*. Springer.

[OMG, 2005] OMG (2005). *Uml 2.0 Infrastructure specification*. Object Management Group.

[Paoli et al., 2011] Paoli, A., Sartini, M., and Lafortune, S. (2011). Active fault tolerant control of Discrete Event Systems using online diagnostics. *Automatica*, 47(4) :639–649.

[Papadopoulos and Maruhn, 2001] Papadopoulos, Y. and Maruhn, M. (July 2001). Model-based synthesis of fault trees from Matlab-Simulink models. In *Proc. International Conference on Dependable Systems and Networks (DSN 2001)*, pages 77–82, Goteberg (Sweden).

[Papadopoulos and McDermid, 1999] Papadopoulos, Y. and McDermid, J.-A. (1999). Hierarchically performed hazard origin and propagation studies. In *Computer Safety, Reliability and Security*, pages 139–152. Springer.

[Papadopoulos et al., 2011] Papadopoulos, Y., Walker, M., Parker, D., Rüde, E., Hamann, R., Uhlig, A., Grätz, U., and Lien, R. (2011). Engineering failure analysis and design optimisation with HiP-HOPS. *Engineering Failure Analysis*, 18(2) :590–608.

[Pietre-Cambacedes and Bouissou, 2010] Pietre-Cambacedes, L. and Bouissou, M. (2010). Attack and defense modeling with bdmp. In *Computer Network Security*, pages 86–101. Springer.

[Pietre-Cambacedes et al., 2011] Pietre-Cambacedes, L., Deflesselle, Y., and Bouissou, M. (May 2011). Security modeling with BDMP : From theory to implementation. In *Proc. Conference on Network and Information Systems Security (SAR-SSI)*, La Rochelle (France). 8 pages.

[Piriou et al., 2014a] Piriou, P.-Y., Faure, J.-M., and Deleuze, G. (2014a). A meta-model to support the integration of dependability concerns into Systems Engineering processes : an example from power production. *IEEE Systems Journal*, PP(99). (10 pages).

[Piriou et al., 2015] Piriou, P.-Y., Faure, J.-M., and Lesage, J.-J. (May 2015). Modeling standby redundancies in repairable systems as guarded preemption mechanisms. In *Proc. 5th International Workshop on Dependable Control of Discrete Systems (DCDS 2015)*, number 5, pages 147–153, Cancun (Mexico).

[Piriou et al., 2014b] Piriou, P.-Y., Faure, J.-M., and Lesage, J.-J. (September 2014b). Control-in-the-loop Model Based Safety Analysis. In *Proc. 24th European Safety & Reliability Conference (ESREL'14)*, pages 655–662, Wroclaw (Poland).

[Prosvirnova, 2014] Prosvirnova, T. (2014). *AltaRica 3.0 : une approche orientée modèles pour la Sûreté de Fonctionnement.* PhD thesis, Ecole Polytechnique, Palaiseau (France).

[Prosvirnova et al., 2013] Prosvirnova, T., Batteux, M., Maarouf, A., and Rauzy, A. (2013). GraphXica : a language for graphical animation of models. In *Proc. 22th European Safety & Reliability Conference (ESREL'13).*

[Puterman, 2014] Puterman, M.-L. (2014). *Markov Decision Processes : discrete stochastic dynamic programming.* John Wiley & Sons.

[Rauzy, 2001] Rauzy, A. (2001). Mathematical foundations of minimal cutsets. *IEEE Transactions on Reliability*, 50(4) :pp. 389–396.

[Rauzy, 2008] Rauzy, A. (2008). Guarded Transition Systems : a new States/Events formalism for Reliability studies. *Journal of Risk and Reliability*, 222(4) :pp. 495–505.

[Rauzy, 2011] Rauzy, A. (2011). Sequence algebra, sequence decision diagrams and dynamic fault trees. *Reliability Engineering & System Safety*, 96(7) :pp. 785–792.

[Renault et al., 1999] Renault, I., Pilliere, M., Villatte, N., and Mouttapa, P. (January 1999). KB3 : computer program for automatic generation of fault trees. In *Proc. Reliability and Maintainability Symposium (RMS 1999)*, pages 389–395, Washington DC (USA).

[Ridoux, 1999] Ridoux, M. (1999). AMDEC - moyen. In *Techniques de l'Ingénieur*. AG4220 edition.

[Ruijters and Stoelinga, 2014] Ruijters, E.-J.-J. and Stoelinga, M.-I.-A. (2014). Fault Tree Analysis : A survey of the state-of-the-art in modeling, analysis and tools. Technical Report TR-CTIT-14-14, Centre for Telematics and Information Technology, University of Twente, Enschede.

[Schmidt, 2015] Schmidt, K.-W. (2015). Computation of supervisors for reconfigurable machine tools. *Discrete Event Dynamic Systems*, 25(1-2) :125–158.

[Stewart, 1994] Stewart, W.-J. (1994). *Introduction to the numerical solution of Markov chains*, volume 1. Princeton University Press.

[Tang and Dugan, 2004] Tang, Z. and Dugan, J. (2004). Minimal Cut Set/Sequence generation for Dynamic Fault Trees. In *Proc. Reliability and Maintainability Symposium, (RAMS 2004)*, pages 207–213, Los Angeles (California, USA).

[Vesely et al., 1981] Vesely, W.-E., Goldberg, F.-F., Roberts, N.-H., and Haasl, D.-F. (1981). *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission. volume NUREG-0492.

[Villemeur, 1988] Villemeur, A. (1988). *Sûreté de fonctionnement des systèmes industriels*. Eyrolles.

[Walker and Papadopoulos, 2009] Walker, M. and Papadopoulos, Y. (2009). Qualitative temporal analysis : Towards a full implementation of the Fault Tree Handbook. *Control Engineering Practice*, 17(10) :pp. 1115–1125.

[Xiang et al., 2011] Xiang, J., Yanoo, K., Maeno, Y., and Tadano, K. (June 2011). Automatic Synthesis of Static Fault Trees from System Models. In *Proc. 5th International Conference on Secure Software Integration and Reliability Improvement (SSIRI 2011)*, pages 127–136, Jeju island (South Korea).

Annexe

A

Théorie des graphes : définitions et notations

Cette annexe rappelle les définitions et notations des éléments de théorie des graphes utilisés dans ce mémoire. Une présentation plus complète de la théorie des graphes peut être consultée dans l'ouvrage [Berge, 1973].

Un *graphe orienté* $\mathcal{G}$ est un 2-tuple $< V, E >$ tel que $E = V \times V$; V est appelé l'ensemble des *sommets* (ou *vertices* en anglais), et E est l'ensemble des *arcs* (ou *edges* en anglais).

Soit $\mathcal{G} =< V, E >$ un graphe orienté, on définit le vocabulaire et les notations suivantes :

- Le *voisinage* $\Gamma_{\mathcal{G}}(v)$ d'un sommet v est l'ensemble de ses sommets *adjacents* (i.e. qui lui sont reliés par un arc). Cet ensemble est l'union des sommets *antécédents* $\Gamma^{-}_{\mathcal{G}}(v)$ et *successeurs* $\Gamma^{+}_{\mathcal{G}}(v)$:

$$\Gamma^{-}_{\mathcal{G}} : \begin{cases} V & \longrightarrow & \mathcal{P}(V) \\ v & \longmapsto & \{a \in V | (a, v) \in E\} \end{cases} \qquad \Gamma^{+}_{\mathcal{G}} : \begin{cases} V & \longrightarrow & \mathcal{P}(V) \\ v & \longmapsto & \{s \in V | (v, s) \in E\} \end{cases}$$

$$\Gamma_{\mathcal{G}} : \begin{cases} V & \longrightarrow & \mathcal{P}(V) \\ v & \longmapsto & \Gamma^{-}_{\mathcal{G}}(v) \cup \Gamma^{+}_{\mathcal{G}}(v) \end{cases}$$

- Le *degré* $d_{\mathcal{G}}(v)$ d'un sommet v est le nombre d'arcs connectés à ce sommet. Ce nombre est la somme du *demi-degré intérieur* $d^{-}_{\mathcal{G}}(v)$ (nombre d'arcs entrants) et

du *demi-degré extérieur* $\boldsymbol{d^+_{\mathcal{G}}(v)}$ (nombre d'arcs sortants) :

$$\boldsymbol{d^-_{\mathcal{G}}} : \begin{cases} \boldsymbol{V} & \longrightarrow & \mathbb{N} \\ \boldsymbol{v} & \longmapsto & \boldsymbol{Card\,(E \cap (V \times \{v\}))} \end{cases} \qquad \boldsymbol{d^+_{\mathcal{G}}} : \begin{cases} \boldsymbol{V} & \longrightarrow & \mathbb{N} \\ \boldsymbol{v} & \longmapsto & \boldsymbol{Card\,(E \cap (\{v\} \times V))} \end{cases}$$

$$\boldsymbol{d_{\mathcal{G}}} : \begin{cases} \boldsymbol{V} & \longrightarrow & \mathbb{N} \\ \boldsymbol{v} & \longmapsto & \boldsymbol{d^-_{\mathcal{G}}(v) + d^+_{\mathcal{G}}(v)} \end{cases}$$

- Un *chemin* est une suite de sommets adjacents :
  Soit $\boldsymbol{k > 1}$,

  $\boldsymbol{(v_i)_{i \in [\![0, k-1]\!]} \in V^k}$ est un chemin de longueur $\boldsymbol{k-1} \iff \boldsymbol{\forall i \in [\![1, k-1]\!], v_i \in \Gamma^+_{\mathcal{G}}(v_{i-1})}$

  On définit ainsi la relation d'*accessibilité* (au sens strict et au sens large), comme la fermeture transitive de $\boldsymbol{E}$:

  Soit $\boldsymbol{(v_1, v_2) \in V^2}$: $\quad \boldsymbol{v_2 \succ_{\mathcal{G}} v_1} \iff$ il existe un chemin entre $\boldsymbol{v_1}$ et $\boldsymbol{v_2}$
  $\boldsymbol{v_2 \succcurlyeq_{\mathcal{G}} v_1 \iff v_2 \succ_{\mathcal{G}} v_1 \vee v_2 = v_1}$

- Un *circuit* est un chemin dont les deux extrémités sont identiques. On définit ainsi la relation de *forte connexité* :

  Soit $\boldsymbol{(v_1, v_2) \in V^2}$: $\qquad \boldsymbol{v_2 \rightleftarrows_{\mathcal{G}} v_1 \iff (v_2 \succcurlyeq_{\mathcal{G}} v_1) \wedge (v_1 \succcurlyeq_{\mathcal{G}} v_2)}$

  On peut vérifier que cette relation est une relation d'équivalence. Soit $\boldsymbol{v \in V}$, on note $\boldsymbol{[v]}$ la classe d'équivalence de $\boldsymbol{v}$ selon la relation $\boldsymbol{\rightleftarrows_{\mathcal{G}}}$:

  $$\boldsymbol{\forall (v_1, v_2) \in V^2, [v_1] = [v_2] \iff v_1 \rightleftarrows_{\mathcal{G}} v_2}$$

  On définit le graphe des composantes fortement connexes d'un graphe $\boldsymbol{\mathcal{G}}$:
  $\boldsymbol{\mathcal{G}/\rightleftarrows_{\mathcal{G}} = < V', E' >}$ tel que :

  - $\boldsymbol{V'}$ est l'ensemble des classes d'équivalence de $\boldsymbol{V}$ pour la relation $\boldsymbol{\rightleftarrows_{\mathcal{G}}}$.
  - $\boldsymbol{\forall (v_1, v_2) \in V^2, ([v_1], [v_2]) \in E' \iff ((v_1, v_2) \in E) \wedge ([v_1] \neq [v_2])}$

Un *graphe orienté acyclique* est un graphe orienté pour lequel la relation d'accessibilité est une relation d'ordre (éventuellement partielle). Un tel graphe vérifie donc la propriété :

$$\boldsymbol{\forall (v_1, v_2) \in V^2, v_1 \rightleftarrows_{\mathcal{G}} v_2 \iff v_1 = v_2}$$

Un *tri topologique* d'un graphe orienté $\mathcal{G} =< V, E >$ est une application strictement croissante de V dans $\mathbb{N}$:

Soit $f : V \to \mathbb{N}$ un tri topologique de $\mathcal{G}$: $\quad \forall (v_1, v_2) \in V^2, v_2 \succ_{\mathcal{G}} v_1 \Longrightarrow f(v_2) > f(v_1)$

[Golumbic, 2004] montre qu'un tel tri topologique existe toujours pour un graphe orienté acyclique, et propose un algorithme pour en construire un (cf. algorithme 4).

Algorithme 4 Construction d'un tri topologique pour un graphe orienté acyclique.

Inputs: $\mathcal{G} =< V, E >$ un graphe orienté acyclique.
Outputs: $f : V \to \mathbb{N}$ un tri topologique de $\mathcal{G}$.

```
1:  done := ∅
2:  i := 0
3:  while done ≠ V do
4:     for all v ∈ V do
5:        if Γ⁻_G(v)\done = ∅ then
6:           f(v) := i
7:           done := done ∪ {v}
8:           i := i + 1
9:        end if
10:    end for
11: end while
```

Annexe B

Niveaux de profondeur des sommets d'un modèle GBDMP

Théorème. *Soit* $< \boldsymbol{V, E, \kappa, v, str, pmc} >$ *un modèle GBDMP bien formé. Il existe une fonction* $\boldsymbol{Level : V \to \mathbb{N}}$ *vérifiant la condition 2.6 :*

$$\forall (\boldsymbol{v_1, v_2}) \in \boldsymbol{V^2} \begin{cases} (\boldsymbol{v_1, v_2}) \in \boldsymbol{E_F} & \Rightarrow \quad \boldsymbol{Level(v_2) > Level(v_1)} \\ (\boldsymbol{v_1, v_2}) \in \boldsymbol{E_S} & \Rightarrow \quad \boldsymbol{Level(v_1) \geq Level(v_2)} \end{cases}$$

Démonstration. On rappelle que $\boldsymbol{\mathcal{G}_F} =< \boldsymbol{N, E_F} >$, $\boldsymbol{\mathcal{G}_S} =< \boldsymbol{V, E_S} >$ et $\boldsymbol{\mathcal{G} = \mathcal{G}_F \cup \overline{\mathcal{G}_S}} =< \boldsymbol{V, \{(v_1, v_2) \in V^2 | (v_1, v_2) \in E_F \vee (v_2, v_1) \in E_S\}} >$.

1. Montrons que $\boldsymbol{\mathcal{G}/\rightleftarrows_{\mathcal{G}}}$ est un graphe orienté acyclique, i.e. que $\boldsymbol{\succcurlyeq_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}}}$ est une relation d'ordre :

   - $\boldsymbol{\succcurlyeq_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}}}$ est réflexif par définition de la relation d'accessibilité au sens large ;
   - la transitivité est évidente ;
   - antisymétrie :

   $$\begin{aligned} ([\boldsymbol{x}] \succcurlyeq_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}} [\boldsymbol{y}]) \wedge ([\boldsymbol{y}] \succcurlyeq_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}} [\boldsymbol{x}]) &\implies (\boldsymbol{x} \succcurlyeq_{\mathcal{G}} \boldsymbol{y}) \wedge (\boldsymbol{y} \succcurlyeq_{\mathcal{G}} \boldsymbol{x}) \\ &\implies \boldsymbol{x} \rightleftarrows_{\mathcal{G}} \boldsymbol{y} \\ &\implies [\boldsymbol{x}] = [\boldsymbol{y}] \end{aligned}$$

   Remarque : ceci est vrai quelque soit le graphe $\boldsymbol{\mathcal{G}}$.

2. Soit $\boldsymbol{f}$ un tri topologique de $\boldsymbol{\mathcal{G}/\rightleftarrows_{\mathcal{G}}}$. Montrons que l'application $\boldsymbol{Level :} \begin{cases} \boldsymbol{V} & \longrightarrow & \mathbb{N} \\ \boldsymbol{v} & \longmapsto & \boldsymbol{f([v])} \end{cases}$ vérifie la condition 2.6 :

- Soit $(v_1, v_2) \in V^2$,

$$\begin{aligned}(v_1, v_2) \in E_F &\overset{\text{déf. de } \mathcal{G}}{\Longrightarrow} v_2 \succ_{\mathcal{G}} v_1 \\ &\overset{\text{Règle 7}}{\Longrightarrow} \neg(v_1 \rightleftarrows_{\mathcal{G}} v_2) \quad \text{(cf. Règle 7)} \\ &\Longrightarrow [v_1] \neq [v_2]\end{aligned}$$

  Or,

$$\begin{aligned}((v_1, v_2) \in E_F) \wedge ([v_1] \neq [v_2]) &\overset{\text{déf. de } \mathcal{G}/\rightleftarrows_{\mathcal{G}}}{\Longrightarrow} [v_2] \succ_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}} [v_1] \\ &\overset{\text{déf. de } f}{\Longrightarrow} f([v_2]) > f([v_1]) \\ &\overset{\text{déf. de } Level}{\Longrightarrow} Level(v_2) > Level(v_1)\end{aligned}$$

  D'où $(\mathbf{v_1}, \mathbf{v_2}) \in \mathbf{E_F} \Rightarrow \mathbf{Level}(\mathbf{v_2}) > \mathbf{Level}(\mathbf{v_1})$.

- Soit $(v_1, v_2) \in V^2$,

$$\begin{aligned}(v_1, v_2) \in E_S &\overset{\text{déf. de } \mathcal{G}}{\Longrightarrow} v_1 \succ_{\mathcal{G}} v_2 \\ &\overset{\text{déf. de } \mathcal{G}/\rightleftarrows_{\mathcal{G}}}{\Longrightarrow} ([v_1] \succ_{\mathcal{G}/\rightleftarrows_{\mathcal{G}}} [v_2]) \vee ([v_1] = [v_2]) \\ &\overset{\text{déf. de } f}{\Longrightarrow} f([v_1]) \geq f([v_2]) \\ &\overset{\text{déf. de } Level}{\Longrightarrow} Level(v_1) \geq Level([v_2])\end{aligned}$$

  D'où $(\mathbf{v_1}, \mathbf{v_2}) \in \mathbf{E_S} \Rightarrow \mathbf{Level}(\mathbf{v_1}) \geq \mathbf{Level}(\mathbf{v_2})$

□

Annexe C

Théorie des langages : définitions et notations

Cette annexe rappelle les définitions et notations des éléments de théorie des langages utilisés dans ce mémoire. Une présentation plus complète de la théorie des langages peut être consultée dans l'ouvrage [Meduna, 2012].

Un *alphabet* est un ensemble fini de *symboles* (ou *lettres*).

Un *mot* $\boldsymbol{\sigma}$ est une concaténation d'un nombre fini de symboles. La longueur du mot $\boldsymbol{\sigma}$ est notée $|\boldsymbol{\sigma}|$ et correspond au nombre de symboles composant le mot. L'unique mot vide de symbole (i.e. $|\boldsymbol{\sigma}| = \mathbf{0}$) est noté $\boldsymbol{\epsilon}$.

La *concaténation* est un opérateur associatif et **non commutatif** entre des symboles ou des mots dont $\boldsymbol{\epsilon}$ est l'élément neutre. La concaténation des mots $\boldsymbol{\sigma_1}$ et $\boldsymbol{\sigma_2}$ se note $\boldsymbol{\sigma_1\sigma_2}$.

La *projection* d'un mot $\boldsymbol{\sigma} \in \boldsymbol{\Sigma_1}$ sur un alphabet $\boldsymbol{\Sigma_2}$ est une fonction permettant de restreindre $\boldsymbol{\sigma}$ sur les symboles de l'alphabet $\boldsymbol{\Sigma_2}$ et est noté $\boldsymbol{\sigma_{|\Sigma_2}}$. Cette fonction peut se définir récursivement ainsi :

$$
\begin{array}{rccl}
 & \boldsymbol{\Sigma_1^*, \Sigma_2} & \longrightarrow & \boldsymbol{\Sigma_2^*} \\
\boldsymbol{projection :} & \boldsymbol{\sigma} & \longmapsto & \boldsymbol{\sigma_{|\Sigma_2}} = \begin{cases} \text{si } \boldsymbol{\sigma} = \boldsymbol{\epsilon} & \boldsymbol{\epsilon} \\ \text{si } \exists \boldsymbol{\sigma'} \in \boldsymbol{\Sigma_1^*} \wedge \boldsymbol{u} \notin \boldsymbol{\Sigma_2} | \boldsymbol{\sigma} = \boldsymbol{\sigma' u} & \boldsymbol{\sigma'_{|\Sigma_2}} \\ \text{si } \exists \boldsymbol{\sigma'} \in \boldsymbol{\Sigma_1^*} \wedge \boldsymbol{u} \in \boldsymbol{\Sigma_2} | \boldsymbol{\sigma} = \boldsymbol{\sigma' u} & \boldsymbol{\sigma'_{|\Sigma_2} u} \end{cases}
\end{array}
$$

Un *langage* $\boldsymbol{\mathcal{L}}$ construit sur l'alphabet $\boldsymbol{\Sigma}$ est un ensemble de mots composés avec des symboles de $\boldsymbol{\Sigma}$.

Le langage comportant tous les mots de longueur $\boldsymbol{n}$ qu'il est possible de construire à l'aide des symboles de $\boldsymbol{\Sigma}$ est noté $\boldsymbol{\Sigma^n}$. Le langage comportant tous les mots qu'il est

possible de construire à l'aide des symboles de $\boldsymbol{\Sigma}$ est appelé *fermeture de Kleene de* $\boldsymbol{\Sigma}$ et est noté $\boldsymbol{\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n}$.

Un mot $\boldsymbol{\sigma_1}$ est dit *préfixe* d'un mot $\boldsymbol{\sigma}$ s'il existe un mot $\boldsymbol{\sigma_2 \in \Sigma^*}$ tel que $\boldsymbol{\sigma = \sigma_1\sigma_2}$. L'ensemble de tous les préfixes d'un mot $\boldsymbol{\sigma}$ est appelé *fermeture préfixielle de* $\boldsymbol{\sigma}$ et est noté $\boldsymbol{Pref(\sigma)}$. Cette définition est étendue aux langages : $\boldsymbol{Pref(\mathcal{L}) = \bigcup_{\sigma\in\mathcal{L}} Pref(\sigma)}$. Un langage qui est égal à sa fermeture préfixielle est dit *préfixe-clos*.

Un *langage régulier* (ou *rationnel*) sur un alphabet $\boldsymbol{\Sigma}$ est un langage qui peut s'écrire à partir des langages singletons sur $\boldsymbol{\Sigma}$, à l'aide d'un nombre fini d'unions, de concaténations et de fermetures de Kleene.

Un *automate à états fini* est un 5-uplet $\boldsymbol{< \Sigma, Q, Q_0, Q_M, \delta >}$ où :

- $\boldsymbol{\Sigma}$ est un *alphabet* ;
- $\boldsymbol{Q}$ est un ensemble fini d'*états* ;
- $\boldsymbol{Q_0 \subseteq Q}$ est l'ensemble des états *initiaux* ;
- $\boldsymbol{Q_M \subseteq Q}$ est l'ensemble des états *marqués* (ou finaux) ;
- $\boldsymbol{\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)}$ est une *fonction de transition*.

Un automate *déterministe*, est tel que $\boldsymbol{Card(Q_0) = 1}$ et $\boldsymbol{\forall (q,u) \in Q\times\Sigma, Card(\delta(q,u)) = 1}$. Tout automate non déterministe a un équivalent déterministe.

Le *langage reconnu* par un automate à états fini est l'ensemble des mots permettant d'atteindre un état de l'automate à partir d'un état initial.

Le *langage marqué* par un automate à états fini est l'ensemble des mots reconnus par cet automate permettant d'atteindre un état marqué à partir d'un état initial.

Le *théorème de Kleene* établit une équivalence entre les langages réguliers et les langages reconnus par les automates à états finis.

Annexe D

Exempe de traduction de modèle GBDMP vers un modèle AltaRica

Les Figures D.1 et D.2 représentent respectivement un modèle GBDMP et sa traduction dans le langage AltaRica [1].

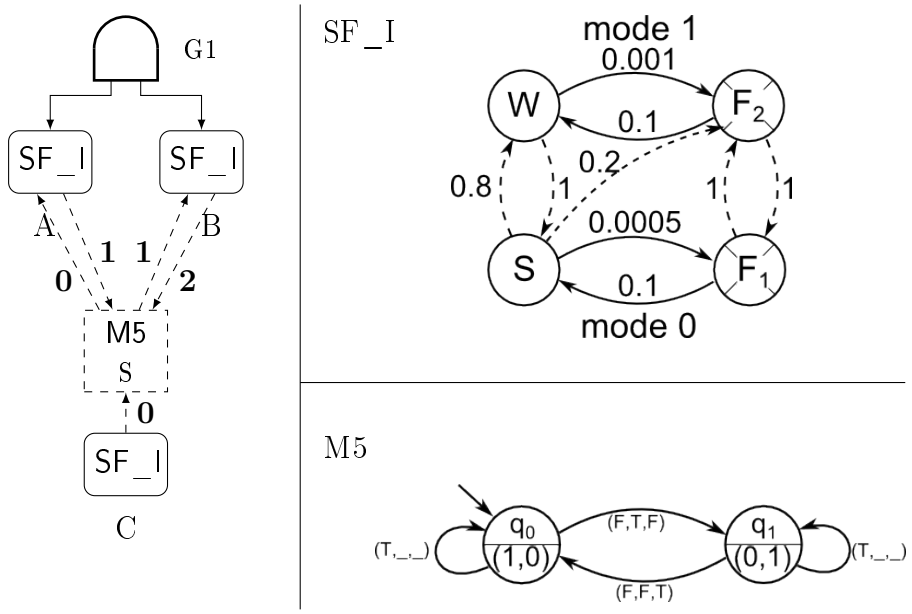

Figure D.1 – Modèle GBDMP à traduire

Dans la traduction en langage AltaRica du comportement d'un PMC de type SF_I, on a tenu compte du fait que les événements ***prov_S_to_W*** et ***prov_S_to_F1*** (respectivement activation normale et défaillance à la sollicitation) sont mutuellement exclusifs et par conséquent, ne peuvent pas être synchronisés. Etant donné que trois feuilles sont associées à un tel PMC dans le modèle, cette distinction des synchronisations génère une combinatoire de 8 ($= \mathbf{2^3}$) transitions synchronisées au niveau du système.

1. On considère que l'on s'intéresse à la défaillance de la porte ***G1***.

Dans chacune de ces transitions, un choix est fait entre les deux événements ci-dessus pour chacun des composants du modèle. Aussi, chaque fois que les PMC doivent être commutés (et les machines de Moore actualisées), l'une de ces transitions est choisie pour être franchie. Les probabilités de franchissement de ces transitions (valeur de l'attribut **expectation**) sont calculées en fonction des probabilités associées aux deux événements ci-dessus.

```
class Node
    parameter Integer FatherWeight_0 = 1;

    Integer M (reset = 0);
    Boolean F (reset = false);
    Boolean R (reset = true);
    Boolean FatherMode_0 (reset = 1);
    Integer maxFatherModes (reset = 1);
  assertion
    maxFatherModes := FatherMode_0*FatherWeight_0;
    M := if R then maxFatherModes else 0;
end

class Leaf extends Node;
    Boolean PMCIsFaulty (reset = false);
  assertion
    F := PMCIsFaulty;
end

class Gate extends Node;
    parameter Integer kind = 1;
    Boolean SonIsAvailable_0 (init = false);
    Boolean SonIsAvailable_1 (init = false);
  assertion
    F := (if SonIsAvailable_0 then 1 else 0) +
         (if SonIsAvailable_1 then 1 else 0) >= kind;
end

class PMC
    Boolean isFaulty (reset = false);
    Integer shouldBeInMode (reset = 0);
end

class Moore
    Integer U (init = 0);
    Integer nextU (reset = 0);

    event update (delay=0, expectation = 1.0);
  transition
    update: not (U==nextU) -> U:=nextU;
    hide update;
end
```

```
domain States_SF_I {None, S, F1, W, F2}
class PMC_Of_Kind_SF_I extends PMC;
    States_SF_I X (init = None);

    event init_S (delay=0, expectation=1.0);
    event init_W (delay=0, expectation=1.0);
    event prov_S_to_W (delay=0, expectation=0.8);
    event prov_S_to_F2 (delay=0, expectation=0.2);
    event prov_F1_to_F2 (delay=0, expectation=1.0);
    event prov_W_to_S (delay=0, expectation=1.0);
    event prov_F2_to_F1 (delay=0, expectation=1.0);
    event spont_S_to_F1 (delay=exponential(0.0005));
    event spont_F1_to_S (delay=exponential(0.2));
    event spont_W_to_F2 (delay=exponential(0.001));
    event spont_F2_to_W (delay=exponential(0.1));
    event Switching_0 (delay=0, expectation=0.8);
    event Switching_1 (delay=0, expectation=0.2);
  transition
    init_S: X==None and shouldBeInMode==0 -> X:=S;
    init_W: X==None and shouldBeInMode==1 -> X:=W;
    prov_S_to_W: X==S and shouldBeInMode==1 -> X:=W;
    prov_S_to_F2: X==S and shouldBeInMode==1 -> X:=F2;
    prov_F1_to_F2: X==F1 and shouldBeInMode==1 -> X:=F2;
    prov_W_to_S: X==W and shouldBeInMode==0 -> X:=S;
    prov_F2_to_F1: X==F2 and shouldBeInMode==0 -> X:=F1;
    spont_S_to_F1: X==S -> X:=F1;
    spont_F1_to_S: X==F1 -> X:=S;
    spont_W_to_F2: X==W -> X:=F2;
    spont_F2_to_W: X==F2 -> X:=W;
    Switching_0: ?init_S & ?prov_W_to_S & ?prov_S_to_W &
             ?prov_F1_to_F2 & ?prov_F2_to_F1 & ?init_W;
    Switching_1: ?init_S & ?prov_W_to_S & ?prov_S_to_F2 &
             ?prov_F1_to_F2 & ?prov_F2_to_F1 & ?init_W;

    hide init_S, init_W, prov_S_to_W, prov_S_to_F2,
         prov_F1_to_F2, prov_W_to_S, prov_F2_to_F1,
         Switching_0, Switching_1;
  assertion
    isFaulty := X==F1 or X==F2;
end
```

```
class Moore_Of_Kind_M5 extends Moore;
    Boolean I0 (reset = false);
    Boolean I1 (reset = false);
    Boolean I2 (reset = false);
    Boolean O0 (reset = false);
    Boolean O1 (reset = false);
  assertion
    nextState :=
      if curState==0 then
        if (I0==false) and (I1==true) and (I2==false)
        then 1 else 0
      else if curState==1 then
        if (I0==false) and (I1==false) and (I2==true)
        then 0 else 1
      else 1;
    if nextState==0 then {O0:=true; O1:=false;}
    if nextState==1 then {O0:=false; O1:=true;}
end

block trad_GBDMP
    PMC_Of_Kind_SF_I PMC_A,PMC_C,PMC_B;
    Trigger_Of_Kind_GGPll S;
    Leaf C ;
    Gate G (kind = 2);
    Leaf A,B (FatherWeight_0=1);
    observer Boolean unavailability = G.F;

    event Switching_0 (delay=0, expectation=0.512);
    event Switching_1 (delay=0, expectation=0.128);
    event Switching_2 (delay=0, expectation=0.128);
    event Switching_3 (delay=0, expectation=0.032);
    event Switching_4 (delay=0, expectation=0.128);
    event Switching_5 (delay=0, expectation=0.032);
    event Switching_6 (delay=0, expectation=0.032);
    event Switching_7 (delay=0, expectation=0.008);
```

```
  transition
    Switching_0: ?S.update & ?PMC_A.Switching_0 &
             ?PMC_C.Switching_0 & ?PMC_B.Switching_0;
    Switching_1: ?S.update & ?PMC_A.Switching_0 &
             ?PMC_C.Switching_0 & ?PMC_B.Switching_1;
    Switching_2: ?S.update & ?PMC_A.Switching_0 &
             ?PMC_C.Switching_1 & ?PMC_B.Switching_0;
    Switching_3: ?S.update & ?PMC_A.Switching_0 &
             ?PMC_C.Switching_1 & ?PMC_B.Switching_1;
    Switching_4: ?S.update & ?PMC_A.Switching_1 &
             ?PMC_C.Switching_0 & ?PMC_B.Switching_0;
    Switching_5: ?S.update & ?PMC_A.Switching_1 &
             ?PMC_C.Switching_0 & ?PMC_B.Switching_1;
    Switching_6: ?S.update & ?PMC_A.Switching_1 &
             ?PMC_C.Switching_1 & ?PMC_B.Switching_0;
    Switching_7: ?S.update & ?PMC_A.Switching_1 &
             ?PMC_C.Switching_1 & ?PMC_B.Switching_1;

  assertion
    PMC_A.Mode := A.M;
    PMC_C.Mode := C.M;
    PMC_B.Mode := B.M;
    A.PMCIsFaulty := PMC_A.isFaulty;
    C.PMCIsFaulty := PMC_C.isFaulty;
    B.PMCIsFaulty := PMC_B.isFaulty;
    G.SonIsAvailable_0 := A.F or not A.R;
    G.SonIsAvailable_1 := B.F or not B.R;
    A.FatherMode_0 := G.M;
    B.FatherMode_0 := G.M;
    S.I0 := C.F;
    S.I1 := A.F;
    S.I2 := B.F;
    A.R := S.O0;
    B.R := S.O1;
end
```

Figure D.2 – Traduction du modèle en langage AltaRica

université PARIS-SACLAY | ÉCOLE DOCTORALE Sciences et technologies de l'information et de la communication (STIC)

Titre : Contribution à l'analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables

Mots clés : *Sûreté de Fonctionnement, Systèmes dynamiques réparables, Reconfiguration, BDMP Généralisés, Séquences de Coupe Minimales, machine de Moore.*

Résumé : Dans les travaux existants, les analyses basées sur les modèles de la Sûreté de Fonctionnement (SdF) d'un système automatisé sont généralement focalisées uniquement sur la partie procédé. Aussi, les stratégies de reconfiguration du procédé - réalisées par le contrôle-commande - ne sont souvent pas modélisées, sinon de manière imprécise et sans échec possible. Pourtant, ces stratégies ont un impact certain sur la SdF du système bouclé, qui doit être pris en compte dans les modèles afin d'améliorer la pertinence des analyses. Le travail dont rend compte cette thèse contribue à la modélisation et à l'analyse de la SdF des systèmes dynamiques, réparables et reconfigurables. Premièrement, un nouveau formalisme de modélisation est proposé pour prendre en compte avec précision les différentes stratégies de reconfiguration du système avec leurs possibles échecs. Ce formalisme développe et généralise le principe des BDMP (Boolean logic Driven Markov Processes en anglais), auxquels il associe des machines de Moore afin de spécifier formellement les stratégies de reconfiguration. Dans un second temps, deux techniques d'analyse basées sur un modèle GBDMP (BDMP Généralisé) sont décrites. Ces techniques permettent d'obtenir un résultat qualitatif : l'ensemble des plus courtes Séquences de Coupe Minimales (SCM), ainsi qu'un résultat quantitatif : indicateur probabiliste de la disponibilité du système. Finalement, la modélisation GBDMP et l'analyse de SdF basée sur un modèle GBDMP sont expérimentées sur un cas d'étude représentatif de plusieurs problématiques industrielles liées au secteur de la production d'énergie électrique.

Title : Contribution to model based safety analysis for dynamic repairable reconfigurable systems

Keywords : *Model Based Safety Analysis, Repairable Dynamic system, Reconfiguration, Generalized BDMP, Minimal Cut Sequences, Moore Machine.*

Abstract : Existing works on Model Based Safety Analysis of an automated system generally focus on the process part. Process reconfiguration strategies that are driven by the control are often modeled without failure and with a lack of accuracy. However these strategies have a real impact on the safety of the closed-loop system. In order to improve the relevance of analysis, this impact has to be captured in models. This thesis contributes to modeling and analysis of dynamic repairable reconfigurable systems. Firstly a new modeling formalism is proposed to relevantly take into account different reconfiguration strategies that can fail. This formalism develops and generalizes the principle of Boolean logic Driven Markov Processes (BDMP), and enriches it with Moore machine for formally specifying reconfiguration strategies. In a second stage, two analysis techniques based on a Generalized BDMP (GBDMP) model are described. These techniques allow to obtain a qualitative result: the set of shortest Minimal Cut Sequences (MCS), and a quantitative result: probabilistic indicator of system availability. Finally, a case study coming from the electric power production field is addressed. This case study shows how several industrial problems can be solved in GBDMP framework.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France