Génération automatique de modèles pour la supervision des systèmes dynamiques hybrides : application aux systèmes ferroviaires

Béranger Six

Université de Lille
CCAMY SYSTÈMES

École doctorale **Sciences Pour l'Ingénieur**

Unité de recherche **CRIStAL**

Thèse présentée par

Béranger SIX

Soutenue le **27 septembre 2018**

En vue de l'obtention du grade de docteur de l'Université de Lille

Disciplines **Informatique, automatique**

Génération automatique de modèles pour la supervision des systèmes dynamiques hybrides

Application aux systèmes ferroviaires

Thèse dirigée par	Belkacem Ould Bouamama	Directeur de thèse
	Anne-Lise Gehin	Co-directrice de thèse

Composition du jury

Rapporteurs	Wolfgang Borutzky	Professeur au Bonn-Rhein-Sieg University
	Noureddine Manamanni	Professeur à l'Université de Reims
Examinateurs	Bouchra Ananou	Maitre de Conférence à l'Université de Aix-Marseille
	Ludovic Macaire	Professeur à l'Université de Lille
	Jean-François Arnold	Docteur à ALSTOM
	Jérémy Patrix	Docteur à CCAMY Systèmes
Invité	Sylvain Lintz	Directeur de CCAMY Systèmes

Université de Lille
CCAMY SYSTÈMES

École doctorale **Sciences Pour l'Ingénieur**

Unité de recherche **CRIStAL**

Thèse présentée par

Béranger SIX

Soutenue le **27 septembre 2018**

En vue de l'obtention du grade de docteur de l'Université de Lille

Disciplines **Informatique, automatique**

Génération automatique de modèles pour la supervision des systèmes dynamiques hybrides

Application aux systèmes ferroviaires

Thèse dirigée par	Belkacem Ould Bouamama	Directeur de thèse
	Anne-Lise Gehin	Co-directrice de thèse

Composition du jury

Rapporteurs	Wolfgang Borutzky	Professeur au Bonn-Rhein-Sieg University
	Noureddine Manamanni	Professeur à l'Université de Reims
Examinateurs	Bouchra Ananou	Maitre de Conférence à l'Université de Aix-Marseille
	Ludovic Macaire	Professeur à l'Université de Lille
	Jean-François Arnold	Docteur à ALSTOM
	Jérémy Patrix	Docteur à CCAMY Systèmes
Invité	Sylvain Lintz	Directeur de CCAMY Systèmes

Université de Lille
CCAMY SYSTÈMES

Doctoral School **Sciences Pour l'Ingénieur**

University Department **CRIStAL**

Thesis defended by

Béranger SIX

Submitted on **27th September, 2018**

In order to become Doctor from Université de Lille

Academic Fields **Automatics, computer science**

Automated Model builder for supervision of Hybrid Dynamic Systems

Applied on a railway rolling stock system

Thesis supervised by	Belkacem Ould Bouamama	Supervisor
	Anne-Lise Gehin	Co-Supervisor

Committee members

Referees	Wolfgang Borutzky	Professor at Bonn-Rhein-Sieg University
	Noureddine Manamanni	Professor at Université de Reims
Examiners	Bouchra Ananou	Associate Professor at Université de Aix-Marseille
	Ludovic Macaire	Professor at Université de Lille
	Jean-François Arnold	PhD at ALSTOM
	Jérémy Patrix	PhD at CCAMY Systèmes
Guest	Sylvain Lintz	Founding CEO of CCAMY Systèmes

Mots clés : systèmes dynamiques hybrides, détection de défaut, graphes de lien, méthodes de simulation, logiciels – développement, chemin de fer – matériel roulant, bond graphs hybrides

Keywords: hybrid dynamical systems, fault detection, bond graphs, simulation methods, software development, rolling stock systems, hybrid bond graphs

Cette thèse a été préparée dans les laboratoires suivants.

CRIStAL

Centre de Recherche en Informatique, Signal et Automatique de Lille
CNRS UMR 9189
Avenue Paul Langevin
59650 Villeneuve d'Ascq
Site web : `www.cristal.univ-lille.fr`

Ccamy Systèmes

Entreprise Ccamy Systèmes
Parc de la Ronce
109 Contre-Allée Route de Neufchâtel
76230 Isneauville
SIRET : 49297223700068
Site web : `http://ccamy-systemes.fr/`

Je dédie ce travail
à ma femme et ma famille

Génération automatique de modèles pour la supervision des systèmes dynamiques hybrides

Application aux systèmes ferroviaires

Résumé

Ce travail de thèse présente différentes contributions pour la génération automatique de modèles représentant les Systèmes Dynamiques Hybrides (SDH) caractérisés par plusieurs modes de fonctionnement. Les composants du système (notamment les capteurs) peuvent être manuellement sélectionnés ou automatiquement exportés à partir des données de Conception Assistée par Ordinateur (CAO) ; ces éléments sont ensuite interconnectés pour reproduire le modèle complet du système industriel. À partir de ce modèle, des schémas-blocs de simulation et de diagnostic, ainsi que la Matrice de Signature de Fautes (FSM) seront produits. L'approche est basée sur les Bonds Graph Hybrides ; la présence de commutations engendre des dynamiques variables (notamment des changements de causalité). Pour lever ces verrous, différents algorithmes sont proposés. En comparaison des logiciels existants, les algorithmes proposés sont applicables sur les systèmes continus, discrets ou hybrides. Les théories et algorithmes développés sont appliqués sur un système ferroviaire de freinage électropneumatique.

Automated Model builder for supervision of Hybrid Dynamic Systems

Applied on a railway rolling stock system

Abstract

This thesis work contributes to perform an automated model builder for Hybrid Dynamic Systems (HDS) with numerous modes. Technological components including sensors with an iconic format can be automatically exported from a computer-aided design (CAD) scheme or manually drag from a database and interconnected, so as to produce the overall HDS model, following industrial technological schemes. Once the model has been created, block diagram for simulation and diagnosis and a Fault Signature Matrix (FSM) could be generated. The theory and algorithm behind the software are based on Hybrid Bond Graphs (HBG). The switching behaviour engenders variable dynamics (particularly causal changes). To solve this problematic, news algorithm are performed. Compared with developed programs for automated modelling, the presented algorithms are valid for continuous, discrete and hybrid systems. The theory is illustrated by an industrial application which consists of the pneumo-electrical control of rolling stock.

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille – CNRS UMR 9189 – Avenue Paul Langevin – 59650 Villeneuve d'Ascq – Site web : `www.cristal.univ-lille.fr`

Remerciements

Je remercie Monsieur Lintz, pour ces cinq années de R&D communes ; sans lui, cette thèse n'aurait pu avoir lieu. J'exprime mon profond respect au professeur Belkacem Ould Bouamama, directeur de recherche à Polytech'Lille et à Madame Anne-Lise Gehin, maître de conférences. J'ai pu durant ces trois années apprécier leur encadrement qui a toujours été humain, d'une grande rigueur, et exemplaire. Toute ma reconnaissance va également à l'ensemble des membres du jury qui ont accepté de participer à l'examen de ce travail. Je remercie le Professeur Wolfgang Borutzky, et le Professeur Noureddine Manamanni, d'avoir accepté de rapporter ce document. Je les remercie pour l'intérêt qu'ils ont porté à ce travail. Je remercie également les membres du jury : Madame Bouchra Ananou, Monsieur Ludovic Macaire, Monsieur Jean-François Arnold et Monsieur Jérémy Patrix d'avoir accepté d'examiner ma thèse

Un grand merci à toute l'équipe de CCAMY pour leur présence sur ce projet. Sans réussir à tous les citer, je tiens également à remercier tous les doctorants de l'équipe CRISTAL pour leur sympathie et leur aide : Ibrahim, Marmoud, Joelle, Ryad, Mohamed. Un grand merci à tous les professeurs de l'équipe MOCIS pour leurs remarques toujours pertinentes et constructives. Merci à Yves Blin, Monsieur Cougnon, Madame de Turckheim ; sans votre école, ce travail ne se serait certainement pas écrit. Merci à mes parents de m'avoir littéralement traîné à l'école les premiers jours ; à 30 ans j'y suis toujours. Enfin merci à ma femme Magali pour sa compréhension et son soutien durant cette dernière année.

Acronymes

	.
BG	Bond Graph.
BGH	Bond Graph Hybride.
CAO	Conception Assistée par Ordinateur.
DB	Determining Bond.
EPICS	Experimental Physics and Industrial Control System.
FSM	Matrice de Signatures de Défauts (Fault Signature Matrix).
HIL	Hardware In the Loop.
MCO	Maintien en Condition Opérationnelle.
MOTHS	MOdeling and Transformation of HBGs for Simulation.
PRES	Pôle de Recherche et d'Enseignement Supérieur.
RRA	Relation de Redondance Analytique.
RRAG	Relation de Redondance Analytique Globale.
SAS	Système Affine par Morceaux.
SCAP	Sequential Causality Assignement Procedure.
SCV	Structure de Commande Variable.
SDH	Système Dynamique Hybride.
SIL	Software In the Loop.
SLS	Système Linéaire à Sauts.
XML	Extensible Markup Language.

Table des matières

Liste des tableaux

Table des figures

Introduction Générale

Cadre de la thèse

Les résultats et contributions scientifiques de ce mémoire ont été obtenus dans le cadre d'une collaboration entre le Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStTAL, CNRS UMR 9189) et la société CCAMY Systèmes.

Cadre académique

Les travaux ont été dirigés par Monsieur Belkacem Ould Bouamama (Professeur à l'Université de Lille) et Madame Anne-Lise Gehin (Maitre de conférences à l'Université de Lille) au sein de l'équipe Méthodes et Outils pour la Conception Intégrée de Systèmes (MOCIS).

Les travaux réalisés par l'équipe MOCIS concernent la conception intégrée de systèmes de commande et de supervision pour les processus multiphysiques (électrique, mécanique, thermique, etc.). La thématique générale qui caractérise l'équipe MOCIS au sein du groupe est la conception intégrée de systèmes complexes décrits par trois couches de modèles hiérarchiques : microscopique, mésoscopique, et macroscopique. L'originalité des travaux de l'équipe est l'utilisation de formalismes graphiques unifiés (bond graph, hypergraphes, digraph, etc.) pour la représentation des trois couches de modèles, mais aussi pour aborder de façon simultanée et cohérente les différents aspects de la conception de systèmes automatisés qui sont la modélisation, l'analyse, la commande, et la surveillance. L'aspect graphique des modèles traités ainsi que leurs propriétés sont exploités pour l'implémentation informatique de ces procédures et pour élaborer les algorithmes de tolérance aux fautes. L'équipe se base sur une approche intégrée s'appuyant sur des outils graphiques unifiés constituant ainsi le socle fédérateur de l'équipe et lui donnant sa cohérence qui contribue à sa notoriété au niveau national et international. Des travaux à caractère fondamental sont menés conjointement

dans les trois thématiques de recherche complémentaires suivantes : Modélisation de systèmes complexes, Analyse structurelle pour le pilotage et Plateforme Logicielle pour la conception. Les résultats théoriques sont valorisés dans trois principaux domaines applicatifs : transport, robotique, et énergie. Par rapport aux thématiques de l'équipe, mes travaux concernent d'un point de vue académique la modélisation et le diagnostic des systèmes dynamiques hybrides modélisés à l'aide des Bond Graph Hybrides (BGH). D'un point de vue industriel, il s'agit de réaliser un outil de génération automatique de modèles pour la simulation, la commande et le diagnostic de systèmes hybrides pour des applications à des systèmes mécatroniques principalement ferroviaires.

Cadre industriel

Les travaux de cette thèse ont été réalisés dans le cadre d'un partenariat industriel avec la société CCAMY Systèmes (CCAMY~Systèmes, 2018).

Créée en 2007 par quatre associés experts dans leurs domaines, l'objectif de CCAMY Systèmes est de répondre aux besoins de ses clients dans les domaines de l'électricité, l'instrumentation, l'automatisme, l'informatique industrielle.

CCAMY Systèmes intervient dans toutes les phases de développement d'un projet, aussi bien en management, conseils, études, que dans la réalisation et l'optimisation.

Bien que spécialisée plus particulièrement dans les secteurs du transport (ferroviaire), de l'automobile, de l'énergie, de l'agroalimentaire et de la pharmaceutique, CCAMY Systèmes est capable d'intervenir dans de nombreux domaines.

Contexte général

Depuis 2011, CCAMY Systèmes s'oriente vers la création de nouveaux outils informatiques pour automatiser les procédures de conception des systèmes mécatroniques. Pour cela, une division de développement logiciel a été mise en place. Cette division a réalisé de nombreux outils différents pour la configuration de bancs électroniques, la vérification de schémas électriques selon des règles empiriques, l'extraction d'informations depuis la Conception Assistée par Ordinateur (CAO) à destination des progiciels de gestion. Dans le cadre de ces projets, «Simulatio®» un simulateur automatique de schémas électriques a été développé pour répondre aux besoins des industriels ferroviaires (SNCF, ALSTOM, Bombardier).

Projet antérieur : Simulatio®

Les constructeurs aéronautiques, ferroviaires ou sous-mariniers ont depuis longtemps intégré la simulation dans les diverses phases de développement de projet, dans le but de réduire les ruptures et d'éviter les réitérations entre les cycles de conception et d'industrialisation.

C'est donc dans l'optique de résoudre cette problématique que la société CCAMY Systèmes a engagé, en 2011, des travaux de recherches et a développé Simulatio®.

Simulatio® est une plateforme logicielle réalisée afin de réduire les temps et les coûts de validation des systèmes électriques de contrôle commande complexes. Ce logiciel permet ainsi de produire directement des modèles de simulation à partir des schémas électriques (CAO). Il gère aussi les domaines pneumatique et hydraulique par équivalence électrique.

Simulatio® se décompose en deux phases distinctes : 1. la création des modèles, 2. l'utilisation des modèles pour validation des systèmes sur des architectures Software In the Loop (SIL) ou Hardware In the Loop (HIL). La création de modèles se décompose en quatre étapes (Figure 1) :

1. Les éléments composants un schéma électrique donné sous forme de CAO sont extraits vers un fichier texte exprimé dans un langage standardisé : Extensible Markup Language (XML). Cette étape permet ainsi de s'affranchir de la multitude de formats propriétaires qui existent dans l'industrie. Nous disposons ainsi d'un format standard de CAO listant l'ensemble des éléments et leurs connexions.

2. Ce fichier XML est ensuite converti en un fichier netlist en utilisant une bibliothèque interne de description de composants. Cette netlist est un fichier pour les logiciels de simulation électrique de type SPICE.

3. Les netlist peuvent ensuite être importées dans un logiciel de simulation électrique. CCAMY Systèmes utilise actuellement plusieurs simulateurs électriques selon le besoin, qui permettent de modifier pendant la simulation la valeur des variables représentant les états discrets du système.

4. Les résultats de la simulation peuvent être affichés en temps réel directement sur la CAO (notamment en colorisant les fils selon la valeur des tensions analogiques qui leur sont associées.).

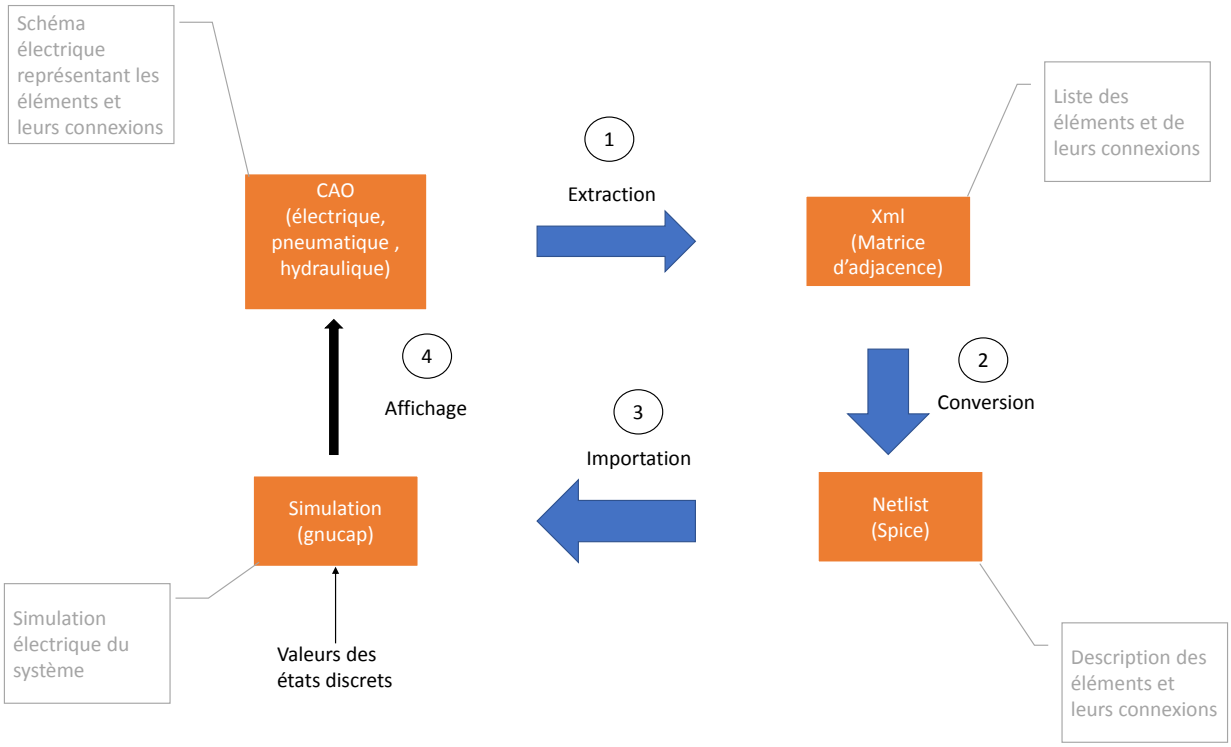


FIGURE 1 – Diagramme de fonctionnement de Simulatio®

Verrous technologique et limites de Simulatio®

Simulatio® a été mis en place sur différents projets. Cependant, lors de sa mise en place sur banc de test, deux problématiques sont apparues : l'impossibilité de simuler des systèmes autres qu'électriques (hormis par analogie très partielle) et des temps de réponse de simulation longs (plusieurs secondes) dus à la complexité des systèmes étudiés, caractérisés par de nombreux modes de fonctionnement.

Projet de recherche : Emulatio®

Pour pallier les lacunes de Simulatio®, CCAMY Systèmes a décidé, en 2015, de développer Emulatio® dans le cadre d'un nouveau projet de recherche. Ce projet effectué dans le cadre de la présente thèse en partenariat avec l'Université de Lille, avec l'équipe MOCIS du laboratoire CRIStALF, a pour objectif de générer automatiquement, à partir des schémas structurels de CAO, des modèles pour la supervision des systèmes de commande hybrides complexes.

L'objectif d'Emulatio® est donc de réaliser une plateforme logicielle pour la modélisation de tout système hybride, en vue de sa simulation pour l'analyse et la commande, mais aussi pour le diagnostic, en termes de détection et localisation de défauts en ligne. A cet effet, il faut intégrer les fonctions logicielles de simulation, utiliser un outil de représentation des systèmes multiphysiques dans un langage unifié. Ce langage doit posséder des propriétés graphiques pour l'implémentation informatique et permettre une représentation du comportement du système sous formes causale et structurelle. Ainsi, l'analyse, la commande et le diagnostic du système seront possibles.

La Figure 2 présente les fonctionnalités attendues de l'outil. Il devra permettre la génération d'un fichier XML standardisé contenant l'ensemble des informations de la CAO. A partir de ce fichier XML et en utilisant une librairie standard de description des composants, un modèle de comportement du système sera généré. Ce modèle comportemental permettra non seulement de simuler le système mais également de détecter des défaillances au niveau des composants internes du système.

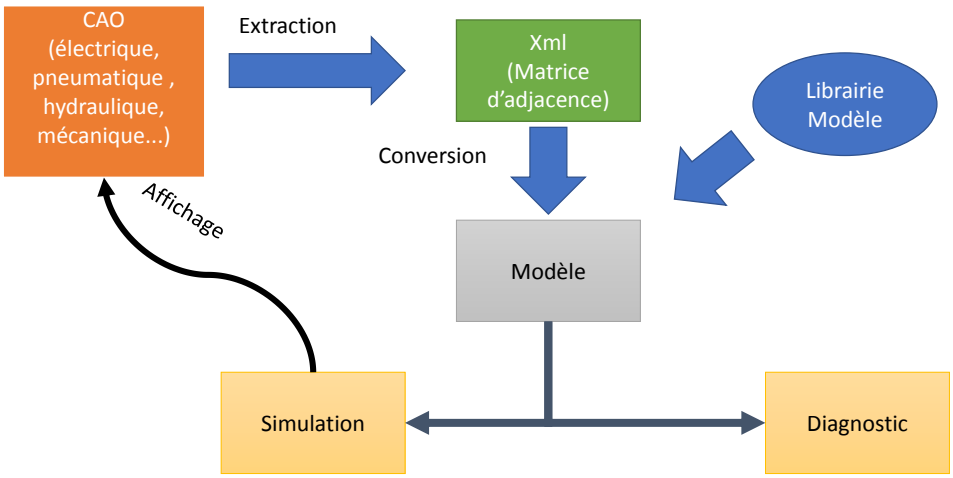


FIGURE 2 – Diagramme fonctionnel d'Emulatio®

Contraintes industrielles

Emulatio® est un projet industriel. Il doit répondre aux contraintes suivantes :

- ➢ Utiliser un seul logiciel pour modéliser l'ensemble des domaines physiques rencontrés par CCAMY Systèmes (électrique, électronique, mécanique, hydraulique et pneumatique).

- ➢ Représenter des systèmes hybrides complexes.
- ➢ Exporter automatiquement les modèles à partir de la CAO.
- ➢ Générer des modèles pour le diagnostic et la simulation à partir du même logiciel.
- ➢ Embarquer les modèles générés sur carte informatique (DSP, FPGA, microP ...) afin de faciliter son utilisation sur banc de test et matériel roulant.
- ➢ Certifier les codes générés suivant les normes du domaine (iec611-31, EN50128).
- ➢ Pouvoir simuler un modèle en moins de 50 millisecondes avec un pas de calcul fixe ou variable.

Problématique et objectif de la thèse

Les systèmes modélisés par CCAMY Systèmes possèdent de nombreux éléments à commutation tels que des diodes, transistors, relais électriques, valves, ou embrayages de systèmes mécaniques. Ces commutations rapides donnent lieu à une forte abstraction de la modélisation du fait que ces changements de mode ont lieu instantanément et peuvent être considérés comme des événements discrets. Ces événements peuvent être autonomes, c'est-à-dire provoqués par des variables continues dont les valeurs franchissent des valeurs seuils ; ou alors ces événements peuvent être contrôlés par des commandes externes (manuelles ou automatiques).

Ces systèmes peuvent être représentés par des modèles dits à structure variable (Cellier et Kofman, 2006). Un modèle est à structure variable si et seulement si ses propriétés structurelles (tel que le nombre d'équations différentielles) dépendent de l'état discret de certains commutateurs. La difficulté des systèmes réalisés par CCAMY Systèmes est liée au très grand nombre de modes qu'ils présentent. Par exemple, pour le métro de Lille plus de 100 contacts de relais électromécaniques existent. C'est-à-dire que si l'on tient compte des états intermédiaires, il existe plus de 2^{100} modes différents. Il nous faut donc développer des algorithmes utilisant une représentation unique pour l'ensemble des modes. Enfin, au-delà de cette problématique scientifique, CCAMY Systèmes associe deux contraintes complexifiant encore cette tâche : gérer les systèmes multiphysiques, et produire un code modulaire (type schéma-bloc, C++ ...) pour faciliter l'implémentation sur cartes. La problématique pourrait donc être résumée par cette interrogation :

Comment superviser des systèmes multiphysiques hybrides possédant un très grand nombre de modes au travers de codes simplement implémentables sur matériel réel ?

Contributions

Les principales contributions de cette thèse sont :

- L'utilisation d'un seul outil fédérateur (les BGH) pour la modélisation dynamique, la simulation, et la détection des défauts d'un système hybride : un seul modèle pour représenter et étudier l'ensemble des modes de fonctionnement.
- La refonte d'un nouveau langage ouvert (Hybrid Bond Graph Modelling Language (HBGML)) pour la représentation des BGH prenant en compte les problématiques de causalité et de représentation graphique.
- La création d'une librairie de composants représentés par des BGH, pour la modélisation des systèmes ferroviaires.
- La création d'algorithmes pour la génération des modèles de simulation et de diagnostic des Systèmes Dynamiques Hybrides (SDH) représentés par des BGH.
- L'implémentation des algorithmes et la mise en application sur un système ferroviaire multiphysique complet.

Liste de publications

Malgré les contraintes industrielles liées à la propriété intellectuelle d'une telle thèse, les résultats de ces travaux de recherche ont fait l'objet de deux conférences internationales avec comité de lecture ainsi que d'une revue scientifique de rang A :

- B. Six et al. (2016). *Railway Direct Braking System Analysis Using Hybrid Bond Graph*. 2016 International Conference on Control, Decision and Information Technologies (CoDIT), p. 034-039
- B. Six et al. (2017). *Automated Model Builder of Hybrid Bond Graph : Application to a Two Level Inverter*. 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED), p. 325-330
- B. Six, B. Ould Bouamama et A. L. Gehin (2018). *Model Builder for Supervision of Hybrid Dynamic Systems : Application to Railway Rolling Stock*. SIMULATION, Status : Under review

Structure du mémoire

Le présent mémoire est organisé autour de quatre chapitres, une introduction et une conclusion. Ces chapitres correspondent à la chronologie du développement des trois années de recherches, conclues par cette rédaction.

Le premier Chapitre définit les systèmes hybrides et propose une classification de ceux-ci. Une revue détaillée des modélisations possibles est exposée. Enfin, les différents éléments du diagnostic et une synthèse sur le choix de modèles sont présentés.

Le second Chapitre rappelle les principes de la modélisation des systèmes continus à l'aide des Bond Graph (BG). Cela est suivi d'une comparaison entre les différentes possibilités offertes pour représenter les éléments caractérisés par des états discrets d'un systèmes hybride. Puis, une revue détaillée des logiciels existants basés sur les BG est développée. Ce chapitre conclut sur l'intérêt de l'utilisation des bond graphs hybrides pour le développement d'un nouvel outil de modélisation et de simulation des systèmes mécatroniques.

Le troisième chapitre traite de la mise en œuvre informatique de la création de modèles pour les SDH. Cette partie regroupe la majorité des contributions scientifiques de cette thèse. La création d'un langage de représentation des BGH et schémas-blocs est ainsi proposée. Puis, de nouveaux algorithmes de génération des modèles de simulation et diagnostic pour BGH sont développés. Enfin, l'informatisation complète du processus est illustrée par un exemple didactique.

Le quatrième chapitre présente une application complète représentant un système de freinage électropneumatique ferroviaire. Cette application valide ainsi les concepts précédemment développés.

Enfin, le travail est conclu par les perspectives industrielles et scientifiques, notamment sur les méthodes d'intégration des modèles de diagnostic.

Bibliographie du présent chapitre

CCAMY~Systèmes (2018). *Votre expert en Contrôle Commande.* URL : `http://ccamy-systemes.fr/fr/`.

Cellier, F. E. et al. (2006). *Continuous System Simulation.* 6th edition. New York : Springer.

Six, B. et al. (2016). *Railway Direct Braking System Analysis Using Hybrid Bond Graph.* 2016 International Conference on Control, Decision and Information Technologies (CoDIT), p. 034-039.

— (2017). *Automated Model Builder of Hybrid Bond Graph : Application to a Two Level Inverter.* 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED), p. 325-330.

Six, B. et al. (2018). *Model Builder for Supervision of Hybrid Dynamic Systems : Application to Railway Rolling Stock.* SIMULATION, Status : Under review.

Chapitre 1

État de l'art : la supervision des Systèmes dynamiques hybrides (SDH)

1.1 Introduction

Le but de ce premier chapitre est de procéder à une revue de l'ensemble des systèmes hybrides et de leur modélisation afin de justifier au mieux les choix réalisés dans la suite de ce mémoire. Dans une première section, nous proposons de classer les systèmes hybrides suivant leurs propriétés mathématiques de commutation. Suite à cette classification, les verrous scientifiques des systèmes modélisés par CCAMY Systèmes seront abordés. Il sera ensuite effectué un rappel sur la supervision des systèmes, en introduisant notamment des notions inhérentes à ce domaine. Enfin, un comparatif des différents outils de modélisations sera présenté, afin de justifier le choix de l'utilisation des BG dans ce mémoire.

1.2 Les systèmes dynamiques hybrides

1.2.1 Définition

Un système physique peut être défini comme une portion de l'univers mis en étude. Cette partie étudiée contient différents éléments connectés ensemble, produisant un résultat (Bertalanffy, 2003).

Un système est défini comme hybride si les différents éléments (ou sous-systèmes) constitutifs de cet ensemble font intervenir :

— des domaines continus et discrets.

— des variables analogiques et des états logiques.

— des équations différentielles, de différence, et logiques.

Il existe principalement deux types de systèmes hybrides :

— Intrinsèquement hybride : présence des deux dynamiques, discrète et continue, dans le même système (domaine aéronautique, automobile, électromécanique).

— Processus continus pilotés par un contrôleur à événements discrets de type automates à états finis (processus en génie des procédés contrôlé par logiciel).

1.2.2 Classification des systèmes hybrides

Dans le cadre de ce travail, nous nous intéressons aux systèmes intrinsèquement hybrides. Selon la manière dont s'effectuent les commutations entre modes, on distingue les Systèmes linéaires à saut, les systèmes affines par morceaux et les systèmes à commutation.

1.2.2.1 Systèmes linéaires à Saut

Un système linéaire à saut (SLS) possèdent différents modes continus (figure 1.1) ; la transition entre les différents modes continus de ce type de systèmes est régie par un processus aléatoire (Vidal, Chiuso et Soatto, 2002). Un exemple éloquent de SLS est le trafic routier : l'événement discret d'un arrêt aléatoire de véhicule entraine un changement de dynamique continue du système (embouteillage par exemple) (Allam et al., 1998).

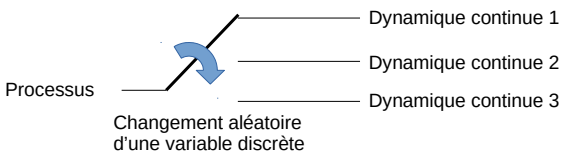


Figure 1.1 – Représentation des systèmes linéaires à saut

Modèle mathématique

Les SLS sont constitués d'une collection de modèles linéaires continus, connectés entre eux par des sauts discrets :

$$x(t+1) = A_{\lambda_t} x(t) + v(t) \tag{1.1}$$

$$y(t) = C_{\lambda_t} x(t) + w(t) \tag{1.2}$$

x, y sont respectivement l'état et la sortie du système.
$\lambda_t \in \{1, ..., s\}$ correspond aux états discrets où s est le nombre de modes différents.
$A_{\lambda_t}, C_{\lambda_t}$ sont les matrices d'états suivant les modes.
$v(t)$, $w(t)$ sont des vecteurs indépendants.

1.2.2.2 Systèmes affines par morceaux

Un Système Affine par Morceaux (SAS) est composé de plusieurs systèmes continus, connexes entre eux. L'ensemble des sous-systèmes continus forment une composition complète du système. Le passage d'un sous-système à un autre correspond à un changement de mode. Ces changements d'état sont déterministes (El Guezar, 2009).

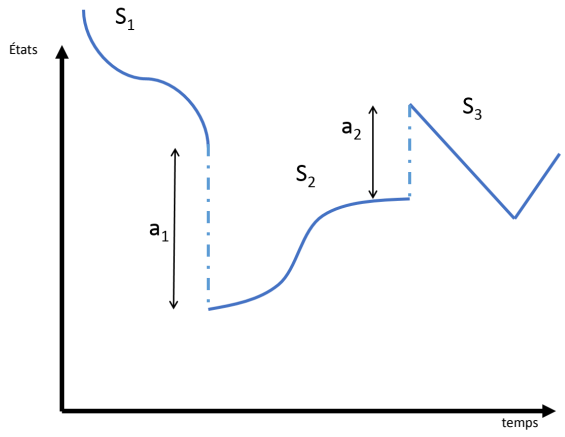


Figure 1.2 – Représentation des systèmes affines constants par morceaux

Modèle mathématique

Un SAS est défini par le modèle suivant (Paoletti et al., 2010) (Xu et Xie, 2014, p19) :

$$dx = A_i x(t)dt + B_i u(t)dt + a_i \tag{1.3}$$

$$y = C_i x(t) + D_i u(t) \tag{1.4}$$

x, y, et u sont respectivement l'état, la sortie et la commande
a_i est la partie affine du système, ce terme est nul pour les systèmes linéaires
$i \in \{1, ..., s\}$ est le mode du système avec s le nombre de modes différents.
L'ensemble des sous-systèmes $S_i{<}A_i, B_i, C_i, D_i, a_i{>}$ forme une partition du système S.

1.2.2.3 Systèmes à commutation

Les systèmes linéaires à commutation (*Switched system*) sont des processus hybrides composés de modes caractérisés par une dynamique continue. La transition entre modes (appelé commutation) est contrôlée par les valeurs de variables discrètes. La figure 1.3 présente le graphe d'état d'un système à commutations possédant deux modes ; où la commutation est contrôlé par la valeur de la variable $S1$. Ces commutations peuvent être autonomes ou contrôlées (cf . Définitions 1.2.1 et 1.2.2). Lorsque les lois de commutation sont parfaitement connues, il est relativement aisé de réaliser l'analyse et la modélisation de ces systèmes. Dans le cas général où les lois qui régissent les commutations ne sont pas parfaitement connues, l'étude de ces systèmes devient bien plus complexe (Liberzon, 2003).

Modèle mathématique

Les systèmes à commutation sont définis par le modèle suivant (Domlan, Ragot et Maquin, 2006) :

$$dx = A_i x(t)dt + Bu(t)dt \tag{1.5}$$

$$y = Cx(t) \tag{1.6}$$

x, y et u sont respectivement l'état, la sortie et la commande, l'ensemble des $A_i \in \{A_1, .., A_n\}$ correspondant aux commutations du système.

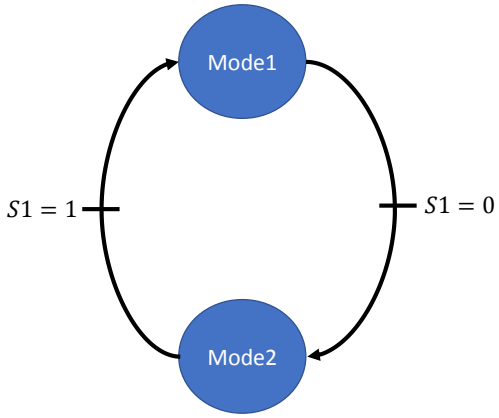


FIGURE 1.3 – Graphe d'état d'un système à commutation possédant deux modes

Définition 1.2.1 (Commutation autonome)**.** *Une commutation est autonome si la valeur de la variable qui contrôle cette commutation est déterminée par l'état interne du système. Par exemple, les changements de mode d'un relais électromécanique s'effectuent de manière autonome puisqu'ils sont fonction de l'intensité du courant de la bobine qui leur est asssociée.*

Définition 1.2.2 (Commutation contrôlée)**.** *Une commutation est contrôlée si la valeur de la variable qui la contrôle est déterminée par une action externe au système. Par exemple, la plupart des contacts des automates programmables industriels peuvent être considérés comme contrôlés puisque leur état dépend de variables informatiques extérieures au système.*

1.2.2.4 Positionnement des systèmes dans le cadre de cette thèse

Le diagramme 1.4 propose une classification des systèmes hybrides. Les applications se décomposent en deux classes, celles dont les commutations sont commandées par des variables aléatoires (systèmes non déterministes) et celles dont les commutations sont commandées par des variables non aléatoires (systèmes déterministes). Les systèmes déterministes peuvent ensuite se décomposer suivant la continuité entre leurs modes.

Les applications développées par CCAMY Systèmes se situent sur des systèmes ferroviaires multi physiques complexes à commutations autonomes et contrôlées. Ces commutations proviennent d'éléments qui sont eux-mêmes des sous-systèmes à commutation (interrupteur, commutateur, etc.) ou des SAS (diodes, valves, etc.).

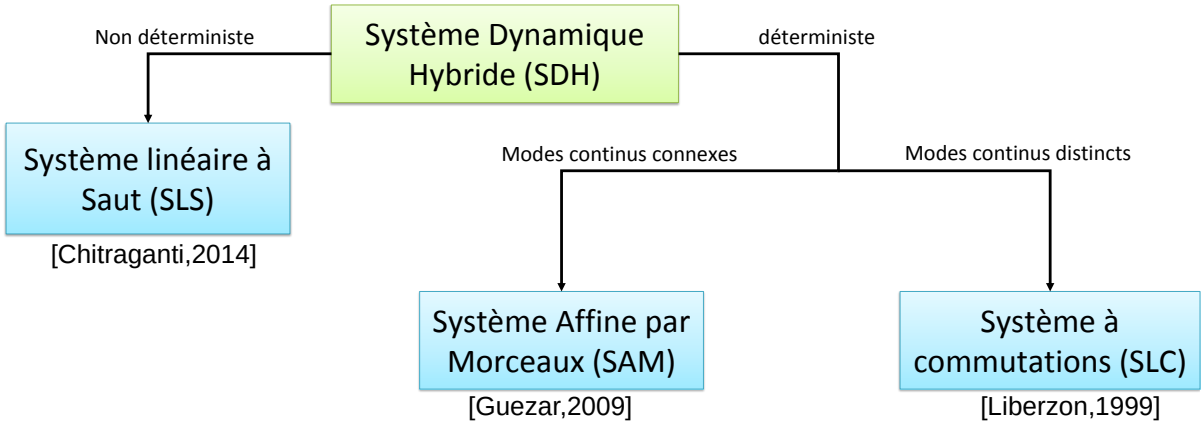


FIGURE 1.4 – Classification des Systèmes dynamiques hybrides

1.2.3 Problématiques des systèmes dynamiques hybrides

Les systèmes hybrides présentent différentes problématiques qu'il est nécessaire de résoudre ou de contourner. Les difficultés à prendre en compte sont liées à la présence de variables discrètes et continues, à la multiplicité des modes de fonctionnement, à la structure variable du système, à l'expression des relations de cause à effet et à la simulation numérique.

1.2.3.1 Hétérogénéité

Un système hybride fait apparaitre des variables continues et discrètes. L'implémentation informatique d'un tel système nécessite la discrétisation des variables continues avec un pas fixe ou variable, alors que les variables discrètes n'existent ou ne varient qu'a certains instants. L'utilisation de ces deux types de paradigmes nécessite des représentations mixtes (SALEH, JOU et NEWTON, 1994).

1.2.3.2 Description de tous les modes

Pour un système hybride à commutation, le nombre de modes est égal à :

$$\prod_{0}^{n} x_i \tag{1.7}$$

où n est le nombre de variables discrètes et x_i le nombre d'états possibles de chaque variable discrète.

La difficulté de représentation de tels systèmes réside dans le fait que, pour chaque composant hybride (transistor, diode, vanne, etc.) présentant, par définition, plusieurs modes de fonctionnement, le nombre de modes est multiplié par le nombre d'états possibles. On a ainsi 2^n modes pour les systèmes ne possédant que des commutateurs à deux états, où n représente le nombre de commutateurs. CCAMY Systèmes modélise des systèmes tels que des métros, où plus de 100 éléments commutatifs sont présents. Autrement dit, cela représente plus de 2^{100} modes différents. Pour simplifier la supervision de tels systèmes, il est donc indispensable de trouver une représentation unique pour l'ensemble des modes.

1.2.3.3 Structure variable

Un modèle est à structure variable si et seulement si ses propriétés structurelles, telles que le nombre d'équations différentielles, dépendent de l'état discret de certains commutateurs (le rang de la matrice d'espace d'état dépend ainsi du mode sélectionné) (Cellier, 1991). Un système à structure variable peut donc être défini par la relation suivante (Yu et Xu, 2003) :

$$\dot{x} = A_i x + B_j u \tag{1.8}$$

les matrices A_i et B_j sont liées au mode de fonctionnement du système autrement dit à son état x. De plus, si la commande u est elle-même variable et discontinue suivant les différents états du système, celui-ci est alors à Structure de Commande Variable (SCV).

La difficulté pour ces systèmes est liée au fait ce que leurs propriétés structurelles telles que l'observabilité, la commandabilité et la diagnosabilité sont fonction du mode dans lequel le système se trouve (Lin et Antsaklis, 2009), (Liu, Lin et Chen, 2013).

1.2.3.4 Causalité variable

Dans le monde physique, les relations entre éléments sont définies par les principes de conservation de l'énergie et de la masse. Ces relations d'égalité ne sont pas ordonnées, c'est-à-dire que l'on ne peut pas dissocier les variables connues des variables dépendantes. Par exemple, il n'existe pas d'expérience physique dans le monde pour savoir si une résistance doit être modélisée comme une chute de tension, ou une source de courant résultant de la différence de potentiel entre ses deux bornes (Cellier, Otter et Elmqvist, 1995). La physique est donc acausale. Cependant afin de calculer les valeurs des énergies, un concept de cause et d'effet est nécessaire. Dans le domaine de la modélisation, la causalité calculatoire (nommée *computing causality* en anglais) définit cet ordre. La causalité des systèmes hybrides, notamment à commutation, est évolutive. Cela signifie que selon le mode dans lequel le système se trouve, l'ordre de calcul des différentes variables n'est pas le même. La figure 1.5 illustre un tel sujet où deux modes sont présents :

— Mode 1 : S1 est à l'état repos. La résistance R est en série avec une source de courant, R est alors équivalent à une chute de tension ($U = RI$). D'un point de vue du schéma-bloc de simulation, U est une sortie et I une entrée

— Mode 2 : S1 est à l'état actif. La source de tension est en série avec la résistance R, qui est alors considérée comme une source de courant ($I = U/R$). D'un point de vue du schéma-bloc de simulation, I est une sortie et U une entrée.

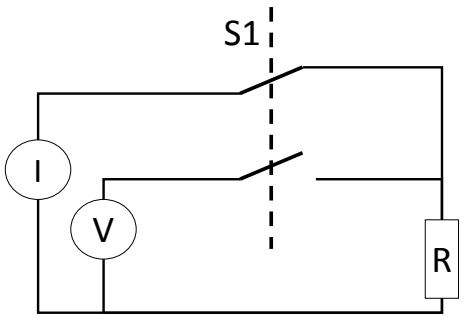


Figure 1.5 – Exemple d'un changement de causalité sur commutation

1.2.3.5 Simulation des systèmes

La simulation numérique scientifique des systèmes repose sur la mise en œuvre de modèles physiques. Ces modèles physiques, lorsqu'ils sont continus peuvent être simplement résolus à l'aide d'analyses numériques d'approximation de solutions d'équations différentielles (Demailly, 2006). Généralement, les systèmes hybrides sont simulés à l'aide des méthodes classiques de résolution de systèmes continus pour chaque mode (Méthodes de Runge-Kutta, Méthodes d'Euler). Dans cette optique, les problématiques de rigidité et de boucles algébriques des simulations des systèmes continus apparaissent : elles sont de plus accrues par la nature hybride du système. La rigidité, nommée *stiffness* en anglais, se définit comme suit : si pour une méthode numérique dans une région définie de stabilité, la résolution doit utiliser des pas de calcul excessivement petits alors le système est rigide dans cet intervalle. Dans le cas des systèmes hybrides à commutation, les sauts sont des changements abrupts qui peuvent engendrer une rigidité sur l'intervalle de saut. La résolution de tels modèles peut donc nécessiter un certain temps (Lambert, 1991).

Une autre difficulté liée à la simulation des systèmes est la présence possible de boucles algébriques. L'équation 1.9 représente un système à boucle algébrique, où $y(t)$ est une fonction dépendante de $y(t)$.

$$y(t) = f(y(t)) \tag{1.9}$$

Si une boucle algébrique ne peut pas être éliminée durant la phase de modélisation du système (Borutzky, 2010), il est tout de même possible de s'affranchir de ce problème en ajoutant un délai lors de son calcul.

1.3 La modélisation des systèmes dynamiques hybrides

Différents formalismes existent pour modéliser les systèmes hybrides. Sans être exhaustifs, nous pouvons citer les approches analytiques, graphiques et événementielles.

1.3.1 Modèles analytiques

Les modèles analytiques représentent les systèmes, à l'aide d'équations basées sur les lois physiques ou déterminée par identification. Ils sont particulièrement adaptés à la simulation des systèmes, puisque les valeurs numériques sont directement disponibles via l'utilisation d'algorithmes de résolution.

Modélisation d'espace état

La modélisation sous forme d'espace état est une représentation minimale de la dynamique du système (cf. équations 1.10) utilisant uniquement l'état (la valeur des grandeurs physiques caractérisant le système), les entrées (valeurs des commandes associées aux actionneurs) et les sorties (valeurs des grandeurs mesurées par les capteurs).

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{cases} \tag{1.10}$$

où u est le vecteur de commande, x le vecteur d'état, y le vecteur de sortie. A, B, C, D sont respectivement les matrices d'état, de commande, d'observation et de commande directe.

Dans le cas des systèmes complexes, cette modélisation est impossible à déduire directement. En effet, l'ensemble des équations qui définissent les sorties et états du système est fonction de toutes les variables physiques. Cela peut rapidement s'avérer excessif. Cette modélisation est donc à considérer comme un résultat attendu de nos développements logiciels plutôt que comme une solution autonome.

Port-Hamiltonien

Une modélisation Port-Hamiltonienne est une représentation énergétique d'un système complexe. Elle est composée de deux éléments (van der Schaft et Maschke, 1995 ; Macchelli, 2002) :

— Une structure de Dirac pour représenter les interconnexions entre fonction flux et énergie.

— Une fonction hamiltonienne de description de l'énergie.

Le système port-Hamiltonien peut être représenté sous la forme d'espace état suivant :

$$\dot{x}(t) = [J(x) - R(x)]\frac{\partial H}{\partial x} + G(x)u(t) \tag{1.11}$$

$$y(t) = G^T(x)\frac{\partial H}{\partial x} \tag{1.12}$$

où G est la matrice port-Hamiltonienne
J, R sont respectivement les matrices d'interconnexion et d'amortissement.
H est l'hamiltonien.
Cette modélisation ne concerne que les systèmes continus. Pour modéliser les systèmes hybrides, les ports paramétrés, complémentaires ou impulsifs ont été créés.

Port Hamiltonien impulsif Dans les publications récentes (Haddad, Nersesov et Chellaboina, 2003), on définit un système Port-Hamiltonien impulsif en ajoutant des variables discrètes (Équation 1.13) :

$$\Delta x(t) = [J_d(x) - R_d(x)]\frac{\partial H}{\partial x} + G_d(x)u_d(t) \tag{1.13}$$

$$y_d(t) = G_d^T(x)\frac{\partial H}{\partial x} \tag{1.14}$$

où $y_d, G_d, et\ u_d$ sont les parties discrètes.

Port Hamiltonien complémentaire Les ports hamiltoniens complémentaires correspondent à l'ajout d'un vecteur complémentaire w représentant les transitions du système (Zainea, 2008).

$$\dot{x}(t) = [J(x) - R(x)]\frac{\partial H}{\partial x} + G(x)u(t) + G_I(x)w \tag{1.15}$$

$$y(t) = G^T(x)\frac{\partial H}{\partial x} + D_w w \tag{1.16}$$

où D_w correspond à la commande discrète.

Port Hamiltonien à port paramétré Les ports hamiltoniens à ports paramétrés traduisent les commutations par modification de la matrice G. Ainsi, pour chaque

mode, on associe une matrice spécifique G_w. L'ensemble des matrices G_w représente une partition de G (Valentin, Magos et Maschke, 2006). Dans l'ensemble des modélisations port-hamiltoniennes hybrides, il faut représenter l'ensemble des équations pour chaque mode, ce qui est excessif pour modéliser les systèmes complexes.

1.3.2 Modèles à événements temporels

DEVS

Ce terme de l'anglais *Discrete Event System Specification* désigne un formalisme de représentation des systèmes hybrides, très fortement orienté informatique. Ce dernier permet de manipuler conjointement les deux concepts d'état et d'événement. Cette représentation se réalise sous la forme suivante (Wainer, 2009 ; Zeigler, 1984 ; Sampath et al., 1995) :

$$M = (X; Y; S; \delta_{int}; \delta_{ext}; \lambda; ta) \tag{1.17}$$

où X, Y, S correspondent respectivement aux entrées, sorties et états du système. δ_{int} est la fonction de transition interne ; δ_{ext} est la fonction externe de transition et λ est la fonction des mesures ta correspond à l'évolution temporelle continue. Afin de faciliter cette écriture très analytique, différentes architectures et méthodologies d'intégrations informatiques ont été proposées.

QSS : L'introduction des méthodes Quantized State Systems (QSS) a permis la modélisation des systèmes continus dans le formalisme DEVS (Bergero et Kofman, 2011 ; Cellier et Kofman, 2006 ; Nutaro, 2010). La méthode QSS repose sur l'idée de la quantification de l'état du système, par opposition à la discrétisation du temps de simulation. Pour une équation spécifiée :

$$\dot{x}(t) = f(x(t), t), \quad x(t_0) = x_0 s \tag{1.18}$$

La méthode QSS de premier ordre permet d'approximer ce système de la manière suivante :

$$\dot{x}(t) = f(q(t), t), \quad q(t_0) = x_0 \tag{1.19}$$

où $q(t)$ est le quantum défini par :

$$q(t) = \begin{cases} x(t) & \text{si } |x(t) - q(t^-)| \geq \Delta Q \\ q(t^-) & \text{sinon} \end{cases} \tag{1.20}$$

Le système continu quantifié possède alors les caractéristiques d'un système à événement discret, celui-ci peut alors être défini par le formalisme DEVS. Cependant cette méthodologie est fortement éloignée de la représentation physique du système. Pour ces raisons, il ne sera pas particulièrement adapté à notre besoin.

1.3.3 Modèles graphiques

Les modèles graphiques représentent les systèmes à l'aide de nœuds connectés entre eux par des liaisons. Les nœuds correspondent aux modèles associés à des sous-systèmes, voir des compoants élémentaires. Les liaisons représentent les connexions entre ces nœuds. Ces modèles utilisent la théorie des graphes pour agencer leurs éléments. Ils sont très largement utilisés dans l'industrie, voire imposés par les normes (John et Tiegelkamp, 2001). En effet, ils permettent de rendre accessible la modélisation à des non-experts de l'automatique au travers de modèles graphiques simplifiés. Enfin, ils sont particulièrement adaptés aux besoins de création hiérarchisée de projets complexes. De plus par leurs concepts graphiques, ces modèles sont bien adaptés pour l'informatisation des différentes procédures d'analyse structurelle (commandalité, observabilité...) et de génération de modèles et d'algorithmes de diagnostic (Samantaray et Ould Bouamama, 2008).

Schémas-blocs

Le schéma-bloc (appelé *block-diagram* en anglais) est un modèle graphique et analytique de représentation des systèmes. Celui-ci est composé de blocs ayant des entrées/sorties et de liaisons orientées qui lient les entrées/sorties entre elles (John et Tiegelkamp, 2001). La fonction associée à chaque bloc est définie soit à l'aide d'un autre schéma-bloc, ou directement par des équations implémentées dans un langage informatique (C, C++, VHDL, etc.).

Cette représentation est très souvent utilisée dans l'industrie, en raison de sa facilité d'implémentation sur les automates programmables industriels : chaque bloc de base est alors fourni par une librairie standard. Cette modélisation est néanmoins difficilement applicable aux SDH, puisqu'elle suppose de modéliser l'ensemble des modes et leurs commutations par de multiples blocs. Comme la modélisation sous-forme d'espace état, cette modélisation doit être considérée comme un résultat attendu de nos développements logiciels plutôt que comme une solution indépendante.

Automates Hybrides

Un automate hybride est une modélisation qui peut être considérée à la fois comme graphique et analytique. Elle consiste à relier des ensembles d'équations analytiques par un digraphe. Chaque ensemble d'équations correspond à un mode spécifique, associés entre eux par des arrêtes correspondant aux conditions de commutation (Buntins et al., 2013 ; Lynch, Segala et Vaandrager, 2003).

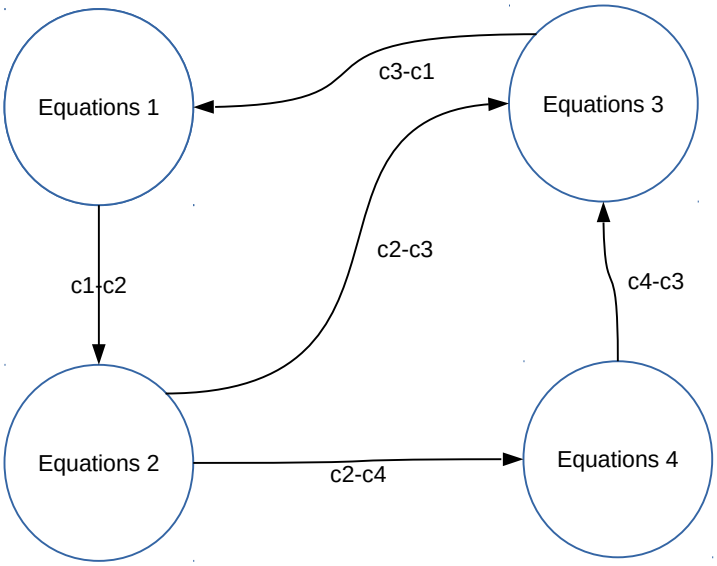


Figure 1.6 – Schéma d'un automate hybride

D'un point de vue plus formel, un automate hybride est défini par :

$$HA = (W, X, Q, O, E, H, D, T) \tag{1.21}$$

où W, X sont respectivement un ensemble de variables externes et un ensemble de variables internes. Q est un ensemble d'états discrets ; O est un ensemble de conditions initiales. E, H sont respectivement les événements externes et internes. D correspond à un ensemble de transitions discrètes. T est un ensemble des trajectoires

Ce type de modèles est bien adapté à la représentation des systèmes hybrides en associant chaque nœud à un mode de fonctionnement et une transition à des conditions

changement de mode. Les conditions de changement de mode, exprimées en fonction des variables des ensembles W et X et des événements des ensembles E et H, permettent d'exprimer des commutations contrôlées ou autonomes. Cependant, ce formalisme ne permet pas de s'affranchir de l'explosion combinatoire des systèmes hybrides complexes (possédant de nombreux modes), puisqu'il fait apparaitre autant d'ensembles d'équations que de combinaisons de modes.

Les réseaux de PETRI hybrides

Les réseaux de Petri peuvent être définis comme une représentation graphique d'un 5-uplet constitué de : {places, transitions, arcs, poids, condition initiales} (Alla et David, 1992). De manière générale, les réseaux de Petri permettent de représenter des systèmes discrets. Cependant, en remplaçant le poids par un réel représentant une quantité temporelle de déplacement, on obtient des réseaux de Petri temporels. En liant ces deux représentations, on obtient alors des réseaux de Petri hybrides. Sur

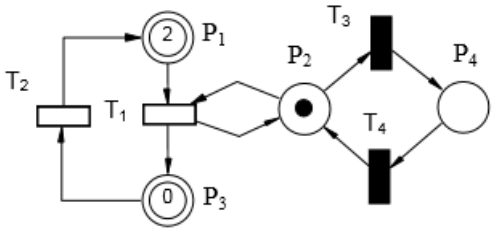


Figure 1.7 – Réseau de Petri hybride

la Figure 1.7, les places P_1 et P_3 correspondent aux modes continus du système. La transition $T2$ représente une commutation autonome. La transition $T1$ représente une commutation contrôlée. Elle est pilotée par une variable discrète pouvant prendre deux valeurs associées aux places $P2$ et $P4$. Les conditions de passage d'une valeur discrète à une autre sont exprimées par les transitions $T3$ et $T4$. La transition T_1 est hybride, elle lie les parties continues et discrètes du système. Ces systèmes sont très facilement implémentables, puisqu'il existe de très nombreuses librairies ouvertes basées sur la représentation des réseaux de Petri. Cependant, cette représentation possède plusieurs défauts pour modéliser les systèmes complexes étant donné que le modèle possède une place par état continu, la problématique combinatoire est la même que pour les automates hybrides.

Bond graph

Le BG est un outil puissant de représentation des systèmes continus (Paynter et Briggs, 1961). Un bond graph est un graphe orienté $G(S; A)$ où S est une collection de nœuds (représentant les éléments physiques du système) et A un ensemble de liens représentant les échanges mutuels de puissances entre ces nœuds. Les BG ont été étendus aux systèmes hybrides via l'ajout d'éléments spécifiques tels que les jonctions contrôlées pour donner un BGH (Mosterman et Biswas, 1998). Le formalisme BGH dispose des atouts suivants :

1. Une approche unifiée permettant la modélisation des systèmes multiphysiques (thermique, hydraulique, électrique, etc.) indépendamment de leur nature.
2. Des aspects fonctionnels et comportementaux qui montrent les aspects topologiques des systèmes complexes dynamiques.
3. Des propriétés structurelles et causales permettant la simulation, l'analyse et le diagnostic des SDH.
4. Une approche graphique orientée objet qui facilite l'implémentation sur matériel embarqué.
5. Le respect des principes physiques de conservations de de la charge, de la masse, de l'énergie.
6. L'utilisation d'un seul modèle pour la représentation de l'ensemble des modes de fonctionnement.

De par l'ensemble des atouts listé ci-dessous, ce formalisme est le plus adaptés pour répondre aux besoins de modélisation des systèmes complexes. Nous reviendrons plus en détail dans le chapitre suivant sur l'approche BG.

1.4 Diagnostic des systèmes dynamiques hybrides (SDH)

La supervision des systèmes est un ensemble d'outils et de méthodes permettant le contrôle de systèmes industriels en condition normale et dégradée (Ould Bouamama, Staroswiecki et Samantaray, 2006). Ces outils incluent notamment le diagnostic en termes de détection et de localisation des défauts (connu en anglais sous le nom de Fault Detection and Isolation (FDI)), mais aussi l'observation des états du système, la

génération de lois de commande et la reconfiguration des loi de commandes en mode dégradé. Dans cette partie, nous nous concentrons sur la mise en œuvre du diagnostic.

Les systèmes peuvent être entravés dans leur bon fonctionnement par différents événements abrupts ou progressifs, localisés ou diffus, liés au vieillissement. Ces non-conformités doivent être traitées pour maintenir une qualité de service suffisante et satisfaire les conditions de sécurité requises. Les contraintes industrielles en termes d'économies, de qualité, de sureté de fonctionnement et de maintien de service rendent indispensable le diagnostic embarqué. Avant de décrire le diagnostic et de classer les différents types de défauts, il est essentiel de rappeler quelques définitions sur ce thème (Laprie, 1996 ; Isermann et Ballé, 1997 ; Zwingelstein, 1995).

Définition 1.4.1 (Défaut)**.** *Un défaut est une déviation non permise d'au moins une des caractéristiques ou paramètres du système. Dans ce mémoire, nous utiliserons indifféremment les termes défauts et fautes.*

Définition 1.4.2 (Défaillance)**.** *Une défaillance est l'incapacité permanente d'un système à accomplir ses fonctions suite à un défaut.*

Définition 1.4.3 (Diagnostic)**.** *Le diagnostic est l'identification de la cause des défaillances à l'aide d'algorithmes basés sur les mesures et la connaissance du système. Cette connaissance peut être empirique, statistique ou basée sur les modèles du système.*

1.4.1 Classification des défauts par localisation

La figure 1.8 représente la chaine de commande complète d'un système. De nombreux défauts peuvent survenir sur les différents éléments du système :

1. Défaut d'un capteur (Du et Cocquempot, 2017) : si la grandeur mesurée n'est pas l'image de la grandeur observée, la commande élaborée peut être incohérente, et de fausses alarmes peuvent être générées.
2. Défaut du processus contrôlé ou défaut physique : si un composant interne n'est plus en mesure d'accomplir sa fonction, le modèle qui le décrit en fonctionnement nominal n'est plus exact et les relations entre entrées et sorties du système deviennent fausses.
3. Défaut d' actionneur (Bouibed et al., 2014) : si le signal délivré par l'actionneur ne correspond pas à celui qu'il devrait être, la variable physique sur laquelle l'actionneur agit, n'est pas modifiée comme elle devrait l'être. Exemple, le débit de la pompe n'est pas égal à celui prévu par sa caractéristique.

4. Défaut du contrôleur (Savkin et Evans, 2002) : si le signal délivré par le contrôleur dévie de celui prévu par l'algorithme les commandes envoyées aux actionneurs ne sont pas les bonnes. Il peut s'agir d'une erreur de conception de la commande, ou d'un défaut du contrôleur lui-même.

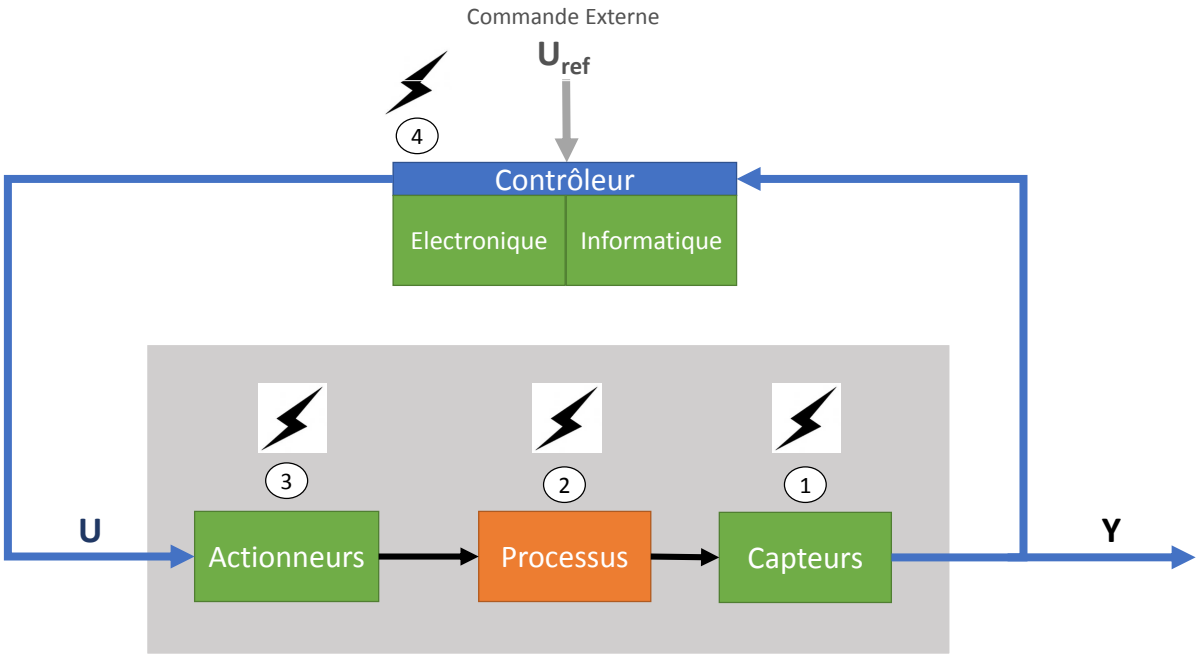


Figure 1.8 – Types de défaut d'un système contrôlé

1.4.2 Classification des fautes par leur nature

Pour les SDH, un défaut peut affecter la dynamique continue ou discrète du système. Une vanne bloquée en position passante par exemple, entrainera le système vers un mode continu non désiré ; de tels défauts sont nommés défauts discrets (Louajri, 2015). À l'opposé, les défauts paramétriques affectent la partie continue du système ; ainsi, une fuite sur un réservoir modifie le modèle du réservoir. Il convient également de noter qu'un défaut paramétrique peut engendrer un défaut discret. Dans le cas par exemple d'une fuite d'un réservoir, cela peut engendrer la non-commutation d'une vanne, ce qui entraine le système dans un mode non désiré.

1.4.3 Les différentes approches du diagnostic

Le diagnostic d'un système s'effectue en trois étapes : la détection de la présence d'un défaut, la localisation (ou isolation) de ce défaut, et l'identification de la cause du défaut. Une fois ces trois étapes effectuées, une décision peut alors être prise.

Toutes les méthodes de diagnostic sont basées sur le même principe, qui consiste à générer un indicateur de faute (appelé aussi résidu) par comparaison entre la sortie fournie par un modèle de référence et celle réelle issue des capteurs. En fonction du type de modèle utilisé, on distingue deux types de méthodes : méthodes dites à base de modèles et sans modèles (Figure 1.9). Les méthodes sans modèles (Patton, Lopez-Toribio et Uppal, 1999 ; Wang, Jiao et Yin, 2017), exploitent des données historiques pour détecter et isoler les défauts. Ces méthodes ne nécessitent pas de modèles analytiques issus de lois physiques complexes, mais la localisation des défauts exige des données provenant du système réel en mode défaillant.

Les méthodes à base de modèles analytiques ne nécessitent aucune action sur le processus réel pour générer les indicateurs de fautes. Leurs performances dépendent principalement de la précision du modèle qui n'est pas simple à obtenir pour des processus complexes. On distingue deux principaux types d'approches à bases de modèles : qualitatives et quantitatives. Les approches qualitatives se basent sur des relations de causes à effets avec pour objectif premier d'identifier l'origine du défaut. Nous pouvons citer les approches DX (Reiter, 1987) et les graphes signés (Iri et al., 1979 ; Chatti, 2013). Ces méthodes qualitatives sont difficilement applicables aux systèmes hybrides, car elles sont plus orientées sur l'identification de l'origine du défaut que sur la quantification de son importance, qui peut être négligeable selon le mode du système.

1.4.4 Détection d'une faute

Dans le cas des approches quantitatives, quelle que soit la solution retenue la détection repose sur la génération de résidus (définition 1.4.4).

Définition 1.4.4 (Résidu)**.** *Un résidu $r(t)$ est un signal qui exprime la différence entre la valeur $\hat{y}(t)$ de la sortie prédite par le modèle de comportement en l'absence de défaillance, et la valeur $y(t)$ réellement mesurée de cette sortie.*

Théorème 1.4.1 (Résidu affecté par une faute)**.** *Un résidu $r(t)$ est affecté par une faute f si l'apparition de la faute f à l'instant t entraine $\mid r(t) \mid > seuil_r$.*

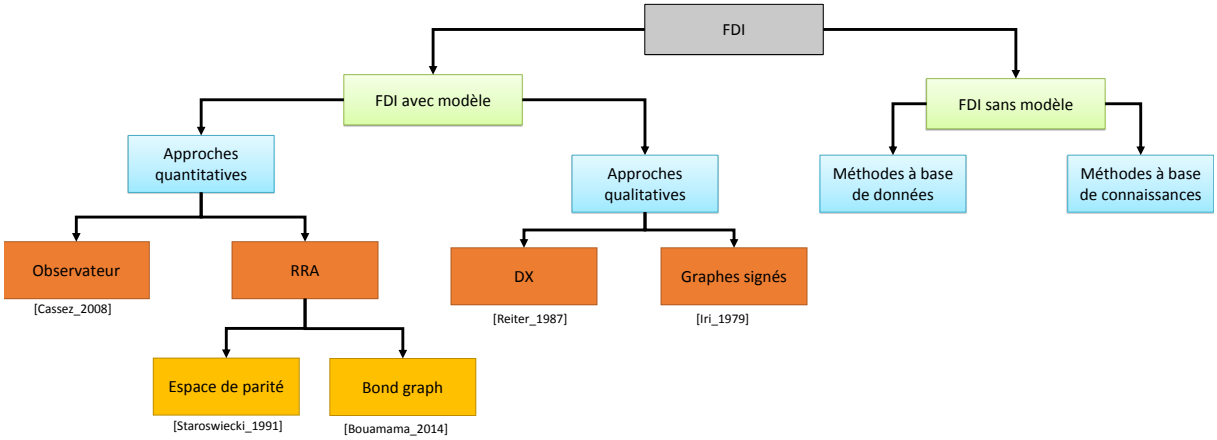


FIGURE 1.9 – Classification des approches FDI

Afin de générer ce résidu, deux solutions sont envisageables, l'utilisation d'observateurs (CASSEZ et TRIPAKIS, 2008) ou de Relations de Redondance Analytique (RRA) (Définition 1.4.5). Les méthodes à base d'observateurs réalisent la génération par différence entre la mesure y et son estimation $\hat{y}$ (Figure 1.10). Elles sont complexes à mettre en œuvre puisque la mesure d'un capteur et son estimation se situent sur deux dynamiques différentes ; il s'agit donc de filtrer fortement la mesure afin de réduire ce décalage. Cela est d'autant plus difficile pour les SDH qui présentent des instabilités lors des changements de modes. Les méthodes à base d'observateurs sont principalement bien adaptées pour la surveillance de défauts actionneurs et capteurs en raison de l'utilisation d'un modèle d'état. De plus, l'augmentation des performances d'isolation exige l'utilisation de bancs observateurs dédiés.

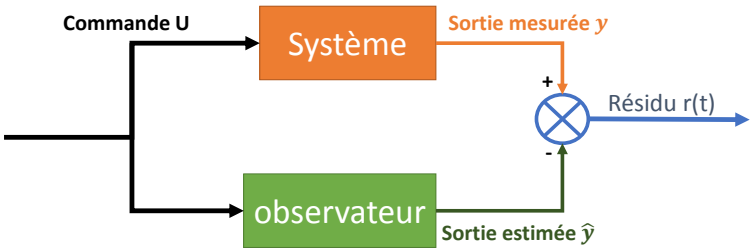


FIGURE 1.10 – Génération de résidu à base d'observateur

Les RRA sont des relations algébro-différentielles (Définition 1.4.5 et Figure 1.11) constituées uniquement de variables connues. Ces relations peuvent alors être évaluées numériquement pour fournir un signal nommé résidu r, $r = Eval(RRA)$. Elles peuvent être directement déduites de l'espace de parité par projections matricielles (Staroswiecki, Cocquempot et Cassar, 1991). Cette méthode analytique nécessite de décrire dans une première étape l'ensemble du système et de ses modes sous forme d'espace état. Dans le cas d'un système décrit par des modèles linéaires et non linéaires sous une forme quelconque, on utilise principalement les graphes bipartis ou les BG. Dans le cas de systèmes hybrides, seuls les BGH peuvent décrire le SDH par un seul modèle. Les BGH par leurs propriétés causales et structurelles, sont exploités pour la génération de RRA valides pour tous les modes de fonctionnement (**Bouamama_2014**). Ces propriétés présentent une motivation supplémentaire pour l'utilisation des BGH pour le diagnostic des systèmes hybrides.

Définition 1.4.5 (RRA)**.** *Une relation de redondance analytique (RRA) est une relation analytique dont l'évaluation numérique donne le résidu. Cette relation ne doit faire intervenir que des variables connues (entrées, sorties, commandes, paramètres).*

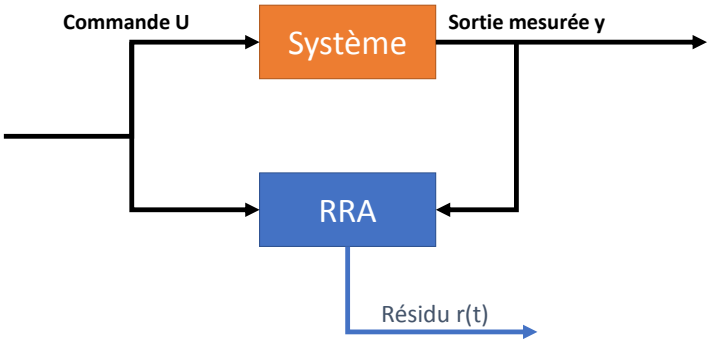


Figure 1.11 – Génération de résidu à base de RRA

1.4.5 Localisation d'une faute

Une fois les défauts détectés, différentes procédures basées sur des signatures répertoriées permettent de localiser l'élément générateur. Cette étape est importante, puisqu'elle permet de réagir aux défauts en s'inscrivant dans le mode dégradé adapté. Les résidus structurés (Définition 1.4.6) sont conçus pour déterminer les causes possibles

d'un défaut à partir de défaillances définies par l'utilisateur (Staroswiecki et Wu, 2004, p.819).

Définition 1.4.6 (Structure (ou Signature) d'un résidu)**.** *La structure (ou signature) d'un résidu r_i par rapport à un ensemble de fautes $F = \{f_j\}$ envisagées pour le système est le mot binaire S_{r_i} composé des bits s_{ij} tels que :*

- $s_{ij} = 1$ *si* r_i *est affecté par* f_j
- $s_{ij} = 0$ *si* r_i *n'est pas affecté par* f_j

À l'aide de ces résidus structurés, et notamment la Matrice de Signatures de Défauts (Fault Signature Matrix) (FSM) (Définition 1.4.8) il est possible de réaliser la détection et l'isolation des défauts. Les signatures de fautes (Définition 1.4.7) permettent en effet de détecter (théorème 1.4.2) et d'isoler (théorème 1.4.3) les fautes. À partir de ces théorèmes, on peut définir les vecteurs de détectabilité (Définition 1.4.9) et les vecteurs d'isolabilité (Définition 1.4.10). Un exemple de ces vecteurs et de la FSM associée est présenté dans le Tableau 1.1.

Définition 1.4.7 (Signature d'une faute)**.** *La signature S_{f_j} d'une faute f_j envisagées pour le système est vecteur binaire défini par :*

- $s_i = 1$ *si* r_i *est affecté par* f_j
- $s_i = 0$ *si* r_i *n'est pas affecté par* f_j

Définition 1.4.8 (Matrice de Signature de Fautes)**.** *L'ensemble des structures S_{r_i} sur l'ensemble des fautes F constitue la matrice des signatures*

D / I	D_1 / I_1	D_2 / I_2	D_3 / I_3	D_4 / I_4
	$F1$	$F2$	$F3$	$F4$
r_1	s_{11}	s_{12}	s_{13}	s_{14}
r_2	s_{21}	s_{22}	s_{23}	s_{24}
r_3	s_{31}	s_{32}	s_{33}	s_{34}

Tableau 1.1 – Matrice de signature de défauts 3x4

Théorème 1.4.2 (Détection d'une faute)**.** *Soit $R = \{r_i\}$ l'ensemble des résidus, une faute est détectée à l'instant t, si il existe : $r_i \in R$ tel que $\mid r_i(t) \mid > seuil_r$*

Théorème 1.4.3 (Localisation faute)**.** *Si $\forall j \neq l$, $S_{f_j} \neq S_{f_l}$ alors f_j est localisable. La signature est différente de toutes les autres signatures.*

Définition 1.4.9 (Vecteur de détectabilité)**.** *Le vecteur de détectabilité D par rapport à un ensemble de pannes F est un ensemble de mots binaires D_j défini par l'équation suivante :*

$$D_j = \begin{cases} 1 \ si \ la \ Faute \ F_j est \ détectable \\ 0 \ sinon \end{cases} \tag{1.22}$$

Définition 1.4.10 (Vecteur d'isolabilité)**.** *Le vecteur d'isolabilité I par rapport à un ensemble de pannes F est un ensemble binaire I_j défini par l'équation suivante :*

$$I_j = \begin{cases} 1 \ si \ la \ Faute \ F_j est \ isolable \\ 0 \ sinon \end{cases} \tag{1.23}$$

Dans le cas particulier des SDH, deux choix sont envisageables pour la construction de la FSM :

— Une matrice par mode de fonctionnement : dans ce cas, il faut connaitre l'état discret du système.

— Une matrice globale pour l'ensemble des modes.

1.5 Synthèse

La représentation des systèmes hybrides multiphysiques présentant de nombreux modes est exhaustive avec la plupart des formalismes de modélisation. Les méthodes Port-Hamiltoniennes et BG sont les seules méthodes présentées capables d'éviter l'explosion combinatoire. Les modèles graphiques tels que les schémas-blocs ou les BG ont une approche hiérarchique (un système peut être décomposé en sous-systèmes et ainsi de suite). Ceci facilite la compréhension des systèmes complexes et facilite l'implémentation informatique notamment dans le cadre d'une conception orientée objet. Les schémas-blocs étant largement utilisés dans l'industrie, leur implémentation sur matériels réels et la certification qui l'accompagne n'est pas une difficulté. La plupart des solutions de modélisation permet d'effectuer le diagnostic des systèmes hybrides. La génération des RRA et de la FSM associée peut être facilitée via l'utilisation de l'utilisation de logiciels spécifiques. On retiendra notamment le logiciel Symbols (**AmalenduMukherjee_2001**), basé sur l'outil BG, et bien adapté au diagnostic des systèmes continus.

Pour toutes ces raisons, le choix a été fait d'utiliser l'outil BGH pour modéliser, simuler et superviser les systèmes dynamiques hybrides conçus par CCAMY Systèmes.

Bibliographie du présent chapitre

Cellier, F. E. et al. (2006). *Continuous System Simulation*. 6th edition. New York : Springer.

Bertalanffy, L. (2003). *General System Theory : Foundations, Development, Applications*. G. Braziller. 324 p.

Vidal, R. et al. (2002). *Observability and Identifiability of Jump-Linear Systems*. Proceedings of the IEEE Conference on Decision and Control. T. 4, p. 3614-3619.

Allam, S. et al. (1998). *Systems with Jumps : Theory and Applications*. Traitement du Signal.

El Guezar, F. (2009). *Modélisation et Simulation Des Systèmes Dynamiques Hybrides Affines Par Morceaux. Exemples En Électronique de Puissance*. Institut National des Sciences Appliquées de Toulouse.

Paoletti, S. et al. (2010). *On the Input-Output Representation of Piecewise Affine State Space Models*. IEEE Transactions on Automatic Control 55.1, p. 60-73.

Xu, J. et al. (2014). *Control and Estimation of Piecewise Affine Systems*. Sous la dir. de J. Xu et al. Woodhead Publishing.

Liberzon, D. (2003). *Switching in Systems and Control*. Réd. par T. Başar. Systems & Control : Foundations & Applications. Boston, MA : Birkhäuser Boston.

Domlan, Elom Ayih et al. (2006). *Systèmes à Commutation : Diagnostic de Fonctionnement et Identification de La Loi de Commutation*. Conférence Internationale Francophone d'Automatique, CIFA'2006. Bordeaux, France, CDROM.

Saleh, R. A. et al. (1994). *Mixed-Mode Simulation and Analog Multilevel Simulation*. Springer Science & Business Media.

Cellier, F. E. (1991). *Continuous System Modeling*. Springer.

Yu, X. et al. (2003). *Variable Structure Systems : Towards the 21st Century*. Springer.

Lin, H. et al. (2009). *Stability and Stabilizability of Switched Linear Systems : A Survey of Recent Results*. IEEE Transactions on Automatic Control 54.2, p. 308-322.

Liu, X. et al. (2013). *Structural Controllability of Switched Linear Systems*. Automatica 49.12, p. 3531-3537.

Cellier, F. E. et al. (1995). *Bond Graph Modeling Of Variable Structure Systems*. Proc. ICBGM'95 (Second International Conference on Bond Graph Modeling and Simulation), Las Vegas, p. 49-55.

Demailly, J. P. (2006). *Analyse numérique et équations différentielles*. 3e édition. Les Ulis, France : EDP Sciences.

Lambert, J. D. (1991). *Numerical Methods for Ordinary Differential Systems : The Initial Value Problem*. New York, NY, USA : John Wiley & Sons, Inc.

Borutzky, W. (2010). *Bond Graph Methodology*. London : Springer London.

Van der Schaft, A. et al. (1995). *The Hamiltonian Formulation of Energy Conserving Physical Systems with External Ports*. AEU - Archiv für Elektronik und Übertragungstechnik 49.5-6, p. 362-371.

Macchelli, A. (2002). *Port Hamiltonian Systems A Unified Approach for Modeling and Control Finite and Infinite Dimensional Physical Systems*. University of Bologna.

Haddad, W. et al. (2003). *Energy-Based Control for Hybrid Port-Controlled Hamiltonian Systems*. Automatica 39.8, p. 1425-1435.

Zainea, M. (2008). *Du Composant à l Automate Hybride Pour La Modélisation et La Simulation Des Systèmes En Communication : Application à l'électronique de Puissance*. IETR - Institut d'Electronique et de Télécommunications de Rennes.

Valentin, C. et al. (2006). *Hybrid Port–Hamiltonian Systems : From Parameterized Incidence Matrices to Hybrid Automata*. Nonlinear Analysis : Theory, Methods & Applications. Hybrid Systems and Applications (5)Hybrid Systems and Applications 65.6, p. 1106-1122.

Wainer, G. A. (2009). *Discrete-Event Modeling and Simulation : A Practitioner's Approach*. Computational analysis, synthesis, and design of dynamic models series. Boca Raton : CRC Press. 494 p.

Zeigler, B. (1984). *Multifacetted Modelling and Discrete Event Simulation*. San Diego, CA, USA : Academic Press Professional, Inc.

Sampath, M. et al. (1995). *Diagnosability of Discrete-Event Systems*. IEEE Transactions on Automatic Control 40.9, p. 1555-1575.

Bergero, F. et al. (2011). *PowerDEVS : A Tool for Hybrid System Modeling and Real-Time Simulation*. Simulation 87, p. 113-132.

Nutaro, J. (2010). *Building Software for Simulation : Theory and Algorithms, with Applications in C++*. Hoboken, N.J : Wiley-Blackwell.

John, K. H. et al. (2001). *IEC 61131-3 : Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools*. Berlin, Heidelberg : Springer-Verlag.

Samantaray, K. et al. (2008). *Model-Based Process Supervision : A Bond Graph Approach*. Advances in Industrial Control. London : Springer-Verlag.

Buntins, M. et al. (2013). *Hybrid Automata as a Modelling Approach in the Behavioural Sciences*. Electronic Notes in Theoretical Computer Science. Proceedings of the first workshop on Hybrid Autonomous Systems 297, p. 47-59.

Lynch, N. et al. (2003). *Hybrid I/O Automata*. Information and Computation 185.1, p. 105-157.

Alla, H. et al. (1992). *Du Grafcet aux réseaux de Petri*. 2ème édition revue et augmentée. Hermes Science Publications.

Paynter, H. M. et al. (1961). *Analysis and Design of Engineering Systems : Class Notes for M.I.T. Course 2.751*. Cambridge, Mass. : M.I.T. Press.

Mosterman, P. J. et al. (1998). *A Theory of Discontinuities in Physical System Models*. Journal of the Franklin Institute 335.3, p. 401-439.

Ould Bouamama, B. et al. (2006). *Software for Supervision System Design in Process Engineering Industry*. IFAC Proceedings Volumes. 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 39.13, p. 646-650.

Laprie, J. L. (1996). *Guide de La Sûreté de Fonctionnement*. Cépaduès.

Isermann, R. et al. (1997). *Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes*. Control Engineering Practice 5.5, p. 709-719.

Zwingelstein, G. (1995). *Diagnostic Des Défaillances : Théorie et Pratique Pour Les Systèmes Industriels*. Hermes Science Publications.

Du, D. et al. (2017). *Fault Diagnosis and Fault Tolerant Control for Discrete-Time Linear Systems with Sensor Fault*. IFAC-PapersOnLine. 20th IFAC World Congress 50.1, p. 15754-15759.

Bouibed, K. et al. (2014). *Actuator and Sensor Fault Detection and Isolation of an Actuated Seat via Nonlinear Multi-Observers*. Systems Science & Control Engineering 2.1, p. 150-160.

Savkin, A. V. et al. (2002). *Hybrid Dynamical Systems - Controller and Sensor Switching*. Springer Science & Business Media.

Louajri, H. (2015). *Centralized and Decentralized Fault Diagnosis of a Class of Hybrid Dynamic Systems : Application to Three Cell Converter*. Lille 1.

Patton, R. J. et al. (1999). *Artificial Intelligence Approaches to Fault Diagnosis*. IEE Colloquium on Condition Monitoring : Machinery, External Structures and Health (Ref. No. 1999/034), p. 5/1-518.

Wang, G. et al. (2017). *Quality-Related Fault Detection Approaches Based on Data Preprocessing*. IFAC-PapersOnLine. 20th IFAC World Congress 50.1, p. 15740-15747.

Reiter, R. (1987). *A Theory of Diagnosis from First Principles*. Artificial Intelligence 32.1, p. 57-95.

Iri, M. et al. (1979). *An Algorithm for Diagnosis of System Failures in the Chemical Process*. Computers & Chemical Engineering 3.1, p. 489-493.

Chatti, N. (2013). *Contribution to the Supervision of Dynamic Systems Using Signed Bond Graph*. Université des Sciences et Technologie de Lille - Lille I.

Cassez, F. et al. (2008). *Fault Diagnosis with Static and Dynamic Observers*. Fundamenta Informaticae 88.4, p. 497-540.

Staroswiecki, M. et al. (1991). *Optimal Design of FDI Systems via Parity Space and Observer Based Approaches*. International Conference on Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91, 143-148 vol.1.

Staroswiecki, M. et al. (2004). *Fault Detection, Supervision and Safety of Technical Processes*. SAFEPROCESS. Elsevier. 1210 p.

Chapitre 2

Modélisation de systèmes mécatroniques à l'aide de Bond Graph Hybrides (BGH)

Ce chapitre a pour but de présenter la solution de modélisation retenue, les BGH, pour la création d'un logiciel de supervision des SDH. Dans un premier temps, nous présenterons la méthodologie BG afin de représenter des systèmes continus. Les possibilités de simulation et de diagnostic à l'aide d'un tel modèle seront développées au travers d'un exemple. Les différentes procédures seront présentées sous forme algorithmique, avec pour objectif de les faire évoluer ensuite pour la gestion des BGH (cf. Chapitre 3). Un état de l'art sur les possibilités de représentation des composants hybrides via le formalisme BG sera présenté. Enfin, il sera conclu par une évaluation des logiciels pour la supervision des SDH basés sur les BGH.

2.1 Rappel sur les bond graphs conventionnels

Le BG est un langage multiphysique qui a largement fait ses preuves d'un point de vue académique et industriel pour la modélisation dynamique des systèmes continus (Paynter et Briggs, 1961). Ce langage permet de modéliser les systèmes indépendamment de leur nature physique. Cet outil a la capacité de représenter des systèmes caractérisés par des variables appartenant aux domaines suivants (cf. figure 2.2) : génie thermique (Abdallah, Gehin et Ould Bouamama, 2017), mécatronique (Merzouki et al., 2012), génie électrique (Šarga et al., 2012), pneumatique et hydraulique (Niu et al., 2015) et chimique.

Le BG est un outil de modélisation basé sur deux grands principes :

— La représentation graphique des échanges de puissance au sein d'un système.

— L'analogie entre les variables de différents domaines physiques.

Plus formellement, un BG est un graphe de liaisons orientées $G(S, A)$ contenant des nœuds S et des arcs A. Les nœuds S modélisent les éléments physiques (capacité, résistance, inertie) ; les arcs A, appelé liaisons dans le cadre des BG, représentent des transferts de puissance instantanés. Ces liaisons sont représentées par des demi-flèches labellisées par une paire de variables effort-flux (e,f) reliant les nœuds entre eux. Le sens de la demi-flèche donne l'orientation (de préférence positive) du transfert d'énergie. À chaque liaison est associée deux variables, une variable d'effort e et une variable de flux f ; le produit de ces deux variables est la puissance échangée p :

$$P(t) = e(t).f(t) \tag{2.1}$$

On appellera ports les réceptacles de connexion d'un nœud vers des liaisons.

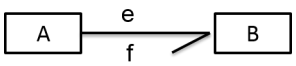


FIGURE 2.1 – Schéma d'un bond graph

Dans la figure 2.1, deux éléments A et B échangent de la puissance dans le sens de A vers B.

Domaine	Effort(e)	Flux(f)
Electrique	Tension $u[V]$	Courant $i[A]$
Mécanique (translation)	Force $F[N]$	Vitesse $v[m/s]$
Mécanique (rotation)	Couple $T[Nm]$	Vitesse angulaire $w[rad/s]$
Hydraulique/Pneumatique	Pression $P[pa]$	Débit $Q[m^3/s]$
Thermodynamique	Température $T[K]$	Flux entropique $\dot{S}[J/K/s]$
Magnétique	Force magnéto-motrice $V[A]$	Flux magnétique $\dot{\phi}$ $[Wb/s]$
Chimique	Potentiel Chimique μ[J/mol]	Flux molaire $\dot{N}$ [mol/s]

FIGURE 2.2 – Différents domaines d'application des bonds graph

2.1.1 La causalité

Telle que définie dans le premier chapitre, la causalité (précisément la causalité calculatoire, en anglais *computational causality*) est un concept qui sépare pour la modélisation la cause de l'effet (Borutzky, 2010). Il s'agit d'une condition indispensable à la résolution des systèmes numériques. L'affectation des causalités, dans les méthodologies à base de BG, peut être automatisée après modélisation ; on convertit alors un «BG acausal» en un «BG causal». Cette possibilité facilite grandement la tâche de création automatisée des modèles numériques pour la simulation et le diagnostic. En effet, cela permet de séparer les tâches génération de modèles et affectation des causalités en utilisant deux algorithmes différents.

Plus formellement, la causalité définit les degrés de liberté du flux ou de l'effort. En effet, de par la dualité flux effort, on impose soit l'effort, soit le flux. Par convention, pour que cette représentation soit graphique, on place une barre orthogonale à chaque liaison. Le positionnement de ce symbole définit la causalité : le trait causal est placé prés (respectivement loin) de l'élément pour lequel l'effort (respectivement le flux) est connu (cf. Figure 2.3 (a)).

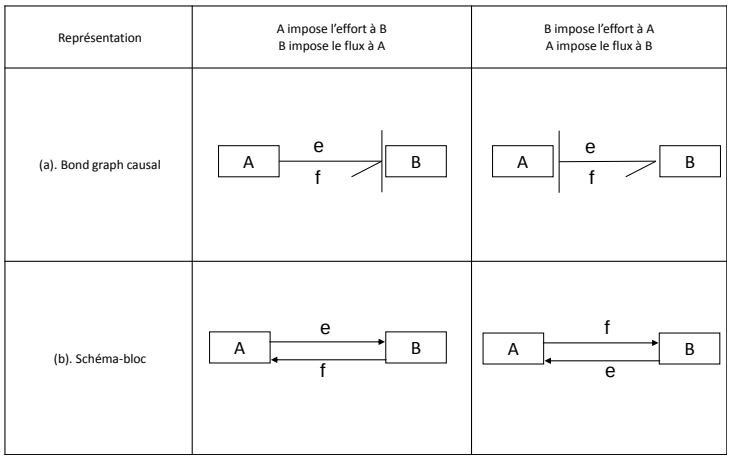


Figure 2.3 – (a) Représentation des deux causalités possibles
(b) Schémas-blocs équivalents

Les éléments des BG d'un point de vue causal peuvent se répartir en quatre classes :

— Les éléments dont la causalité est forcée (les sources).

— Les éléments dont la causalité est libre (les résistances).

— Les éléments qui possèdent des règles de causalité plus complexes (les jonctions, les transformateurs et les gyrateurs).

— Les éléments qui possèdent une causalité préférentielle intégrale ou dérivée (les éléments de stockage d'énergie).

Concernant le choix entre causalités intégrales ou dérivées, on privilégiera un BG intégral pour la création de modèles de simulation, car le calcul intégral est plus robuste aux bruits. En revanche, pour le diagnostic des systèmes, le BG dérivé sera préféré puisque les conditions initiales ne sont pas connues dans un système réel (Loureiro, Merzouki et Bouamama, 2014). Le diagnostic sera ainsi plus aisément embarqué sur un système réel.

2.1.2 Les différents éléments unitaires

Les BG représentant des systèmes continus (dits BG continus) sont constitués de différents éléments appelés nœuds. Pour les systèmes considérés dans ce travail (figure 2.2), l'ensemble des nœuds est :
$S = \{R\}\, U\, \{C\}\, U\, \{I\}\, U\, \{TF\}\, U\, \{GY\}\, U\, \{Se\}\, U\, \{Sf\}\, U\, \{De\}\, U\, \{Df\}\, U\, \{J\}$. Pour plus de détails sur ces éléments, une description complète des éléments conventionnels est présentée en Annexe 1, avec une spécificité sur la représentation des capteurs. En effet, la liaison est souvent représentée par une flèche purement indicative et non par une demi-flèche (principalement dans la communauté française (Borutzky, 2010, p. 62)), car on considère que les capteurs échangent de l'information et non de l'énergie avec le système. Dans ce travail, les détecteurs sont représentés par des éléments nommées De ou Df, suivi d'une demi-flèche orientée vers l'élément détecteur. Les éléments De et Df sont alors respectivement équivalent à des sources de flux ou d'effort nul (Gawthrop et Smith, 1996). Cette représentation permet d'éviter la gestion de multiple types de liaisons dans nos développements logiciels.

2.1.3 Conversion BG vers schémas-blocs

Pour le calcul numérique des modèles BG, il est plus simple de procéder directement à partir de schémas-blocs. Cela facilite l'implémentation par le respect de normes (iec 61131-3) et l'utisation de nombreuses librairies existantes du commerce (Simulink, LABVIEW, FPGA-IP, etc.). Les BG causaux sont automatiquement convertibles en schémas-blocs (théorème 2.1.1), puisqu'ils possèdent des connexions bilatérales (cf. Figure 2.3 (b)).

Théorème 2.1.1. *Un* BG *causal peut être systématiquement converti en schéma-bloc (*Karnopp *et* Rosenberg*, 1968).*

2.2 Supervision informatisée de systèmes mécatroniques continus à l'aide de BG

Cette section a pour objectif de présenter le processus complet de modélisation et supervision d'un système mécatronique continu à l'aide de BG. Pour ce faire, un exemple pédagogique (figure 2.4) de filtre électrique avec moteur sera développé. Les procédures d'assignation de causalité, de génération de schémas-blocs, et de diagnostic seront exposées. L'objectif de cette section est donc de présenter de façon exhaustive l'ensemble des procédures qui permettent de réaliser la simulation et le diagnostic à partir des modèles BG continus. Différents algorithmes seront proposés en vue de l'implémentation des procédures. Cette description exhaustive sera complétée dans le chapitre suivant pour la représentation des SDH.

2.2.1 Système mécatronique en guise de fil conducteur

Considérons à titre d'exemple et comme fil conducteur, le système mécatronique représenté par le schéma structurel de la figure 2.4. Ce système continu est composé d'une source d'alimentation continue V, montée en série avec un circuit de filtrage de type RLC. La tension de ce circuit V_{mes} est mesurée en permanence par un Voltmètre. L'énergie de ce circuit sert à un moteur à courant continu (MCC) qui s'oppose à une charge mécanique CM ; la vitesse de ce moteur est mesurée par deux capteurs $w1m$ et $w2m$. La boite de vitesse permet de réduire le couple appliqué à la charge mécanique.

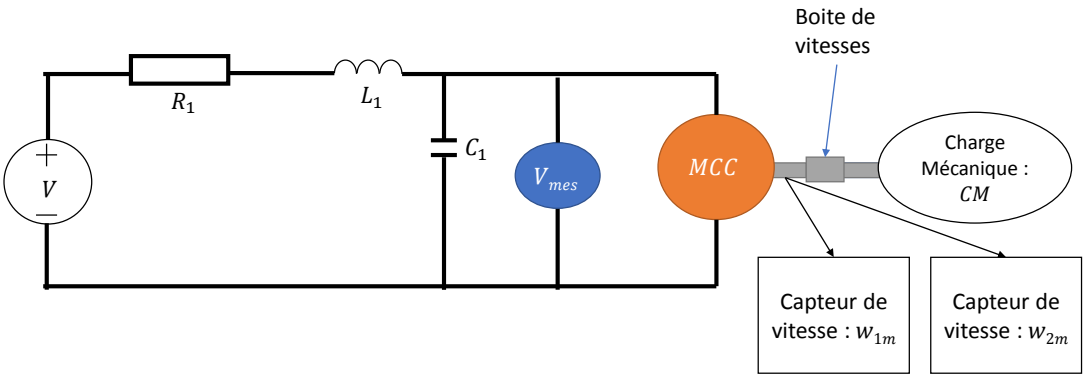


FIGURE 2.4 – Schéma structurel d'un système mécatronique

2.2.2 Création du BG acausal

La conversion du schéma structurel en BG acausal s'effectue en deux étapes :

1. Substituer chaque élément du schéma par son nœud BG équivalent.
2. Connecter l'ensemble des nœuds entre eux.

Le résultat de cette conversion est présenté par la figure 2.5. Les éléments $Se : V$, $R : R_1$, $I : L_1$ substituent respectivement la source de tension, la résistance électrique et l'inductance. Ils sont connectés par une jonction 1, puisque ceux-ci sont en série sur le schéma. Cet ensemble est ensuite connecté en parallèle (jonction 0) à la capacité $C : C_1$, au capteur de tension $De : V_{mes}$ et au moteur électrique représenté par le Gyrateur GY de module K_e pour modéliser la transformation de l'énergie électrique en énergie mécanique. Le moteur est connecté par la jonction 1_b aux capteurs de vitesses $Df : w_{1m}$, $Df : w_{2m}$, à l'arbre de transmission et à la boite de vitesse. L'inertie et les frottements de l'arbre de transmission sont respectivement représentés par les éléments $I : J_1$ et $R : R_f$. La boite de vitesse est modélisée par un transformateur TF de module K_f représentant le rapport de transmission. La charge mécanique $Se : CM$ applique au système une torsion opposée $-CM$.

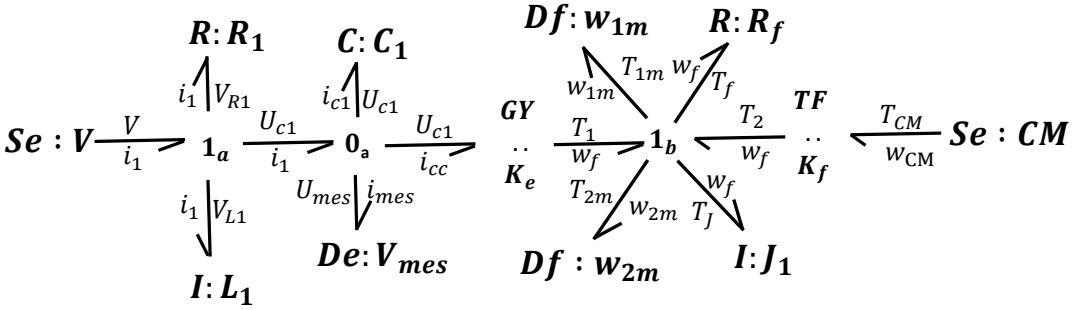


FIGURE 2.5 – BG acausal d'un système mécatronique

2.2.3 Affectation des causalités des systèmes continus

L'affectation des causalités sur un BG acausal continu peut se faire selon la procédure appelée Sequential Causality Assignement Procedure (SCAP) (Karnopp et Rosenberg, 1968). La procédure suivante est arrêtée, dès que l'ensemble des causalités est affecté :

1. Affecter la causalité obligatoire sur une des sources et propager cette causalité sur le BG au travers des éléments : 0, 1, GY, TF aussi loin que possible, en respectant les règles de causalité de chaque élément. Recommencer cette étape tant que toutes les causalités obligatoires des éléments sources ne sont pas affectées.
2. Affecter une causalité préférentielle (intégrale pour la simulation et dérivée pour le diagnostic) sur un des éléments C ou I et propager cette causalité sur le BG au travers des éléments 0, 1, GY, TF aussi loin que possible, en respectant les règles de causalité de chaque élément. Recommencer cette étape jusqu'à ce que tous les éléments C ou I aient une causalité affectée.
3. Enfin, affecter une causalité arbitraire à un élément R et propager cette causalité sur le BG au travers des éléments 0, 1, GY, TF aussi loin que possible en respectant les règles de causalités de chaque élément. Recommencer cette étape tant qu'il existe un élément R ne possédant pas de causalité (Si cette étape est nécessaire, cela signifie qu'une boucle algébrique est présente. Bien que non désirées, ces boucles peuvent être résolues durant la simulation par réduction du modèle ou ajout d'un retard pur et de conditions initiales.).

Mise sous forme d'algorithme

Dans le cadre de ce mémoire, l'objectif est d'informatiser la génération du modèle, dit BGH, représentant un SDH. Ainsi, nous préférons une formulation plus algorithmique de cette procédure, cette formulation est donnée par l'algorithme 1 (cet algorithme sera étendu aux BGH dans le chapitre 3 : algorithme 4). Enfin, il convient de noter que nous avons décidé de marquer les résistances dont la causalité a été affectée à l'étape 3 de la procédure. En effet, ces éléments sont de potentielles boucles causales (Borutzky, 2010)[p.103] ; qui seront prises en compte par les logiciels de simulation.

Algorithme 1 Algorithme équivalent à la SCAP

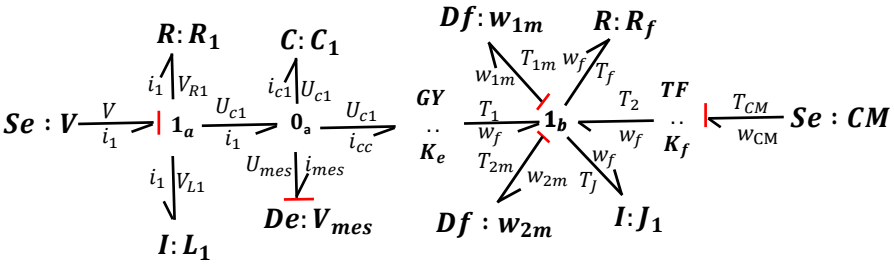
```
1: Push all BG's junction nodes (TF,GY,1,0) in a stack J'
2:
3: for all Source : S_i do
4:     Set imposed causality to the S_i port
5: end for
6: Propagate-causality-on-junction(&J') [Algorithm 2]
7: for all Detector : D_i do
8:     Set imposed causality to the D_i port
9: end for
10: Propagate-causality-on-junction(&J') [Algorithm 2]
11: Push all BG's conservative nodes (C, I) in a stack K'
12: while K' is not empty do
13:     x= K'.pop()
14:     if x port owns a bond with causality unset then
15:         Assign integral(simulation) or derivative(diagnosis) causality to bond
    Propagate-causality-on-junction(&J') [Algorithm 2]
16:     end if
17: end while
18: Push all BG's resistor nodes (R) in a stack R'
19: while R' is not empty do
20:     x= R'.pop()
21:     if x port owns a bond with causality unset then
22:         Assign resistive causality to bond
23:         Propagate-causality-on-junction(&J') [Algorithm 2]
24:         Mark this element as potential causal loop
25:     end if
26: end while
```

Algorithme 2 Algorithme de propagation de la causalité sur jonction standard

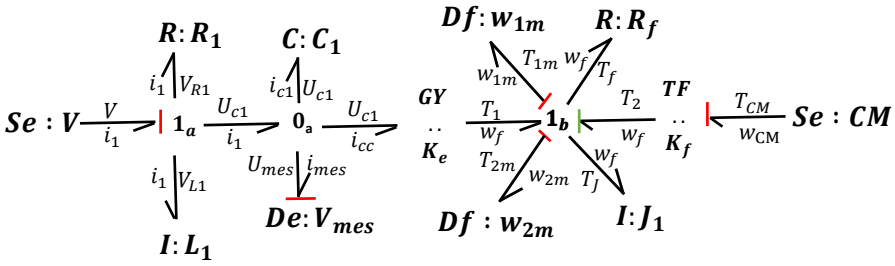
```
1: function PROPAGATE-CAUSALITY-ON-JUNCTION(&J)
2:     repeat
3:         causality-has-changed = false
4:         for all junction j ∈ J  do
5:             if j is a TF or a GY element then
6:                 if one port has its causality set then
7:                     apply causality to other ports
8:                     causality-has-changed = true
9:                     remove j from J
10:                end if
11:            else
12:                if  port : Determining Bond (DB) (definition 4.7.1) has causality set
   then
13:                    apply causality to other ports
14:                    causality-has-changed = true
15:                    remove j from J
16:                else if  all ports expect DB have causality set then
17:                    apply causality to DB
18:                    causality-has-changed = true
19:                    remove j from J
20:                else if  all ports have causality set then
21:                    check if there is no causal conflicts
22:                    remove j from J
23:                end if
24:            end if
25:        end for
26:    until causality-has-changed == true
27: end function
```

Résultat

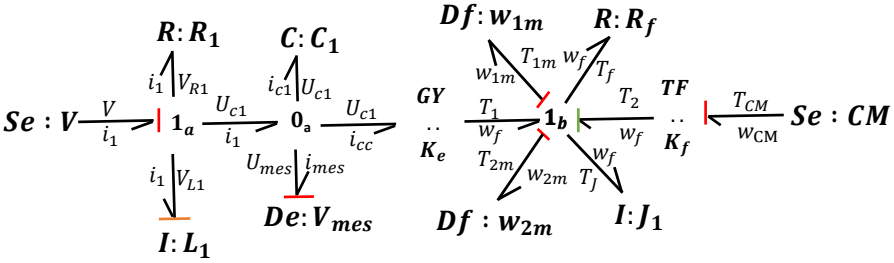
Les figures 2.6 et 2.7 montrent les étapes d'assignation des causalité par application de l'algorithme 1. Les causalités de la figure (a) sont celles imposées par les sources et détecteurs, puis les causalités sont propagées au travers de l'élément $TF : K_f$ (figure (b)). Suite à cela, les causalités ne peuvent plus être propagées au travers des jonctions 0_a, 1_a et 1_b (celles-ci n'ont de causalités affectées ni sur le DB, ni sur l'ensemble des liaisons à l'exception du DB) ; il est donc nécessaire d'affecter une causalité intégrale à un élément et de poursuivre le processus. La figure (c) montre l'affectation d'une causalité intégrale à l'élément $I : L_1$, les causalités sont ensuite de nouveau propagées au travers de la jonction 1_a (figure (d)). À cette étape, les causalités ne peuvent plus être propagées au travers des jonctions 0_a et 1_b ; une causalité intégrale est donc affectée à l'élément $C : C_1$ (figure (e)). Enfin les causalités sont de nouveaux propagées au travers des éléments 0_a, $GY : K_e$ et $1b$ (figure (f)).



(a) BG avec causalités imposées aux sources et détecteurs

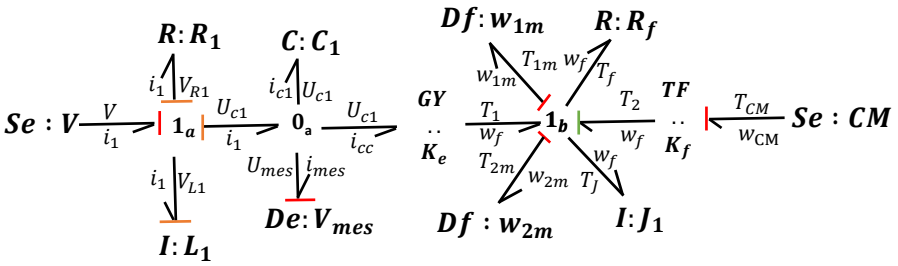


(b) BG après propagation des causalités au travers de l'élément $TF : K_f$

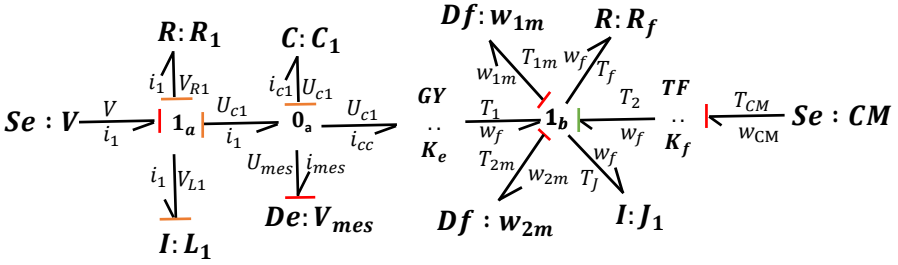


(c) BG après affectation d'une causalité intégrale à l'élément $I: L_1$

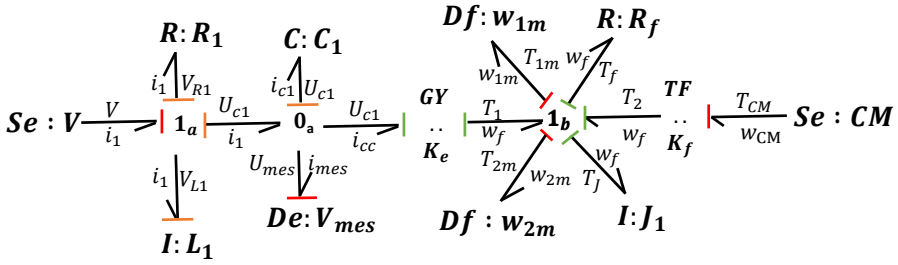
FIGURE 2.6 – Affection des causalités fixes d'un BG de système mécatronique et propagation au travers des éléments



(d) BG après propagation des causalités au travers de la jonction 1_a



(e) BG après affectation d'une causalité intégrale à l'élément $C: C_1$



(f) BG après propagation des causalités au travers des jonctions : 0_a, $GY: K_e$, 1_b

FIGURE 2.7 – Affection d'une causalité intégrale d'un BG de système mécatronique et propagation au travers des éléments

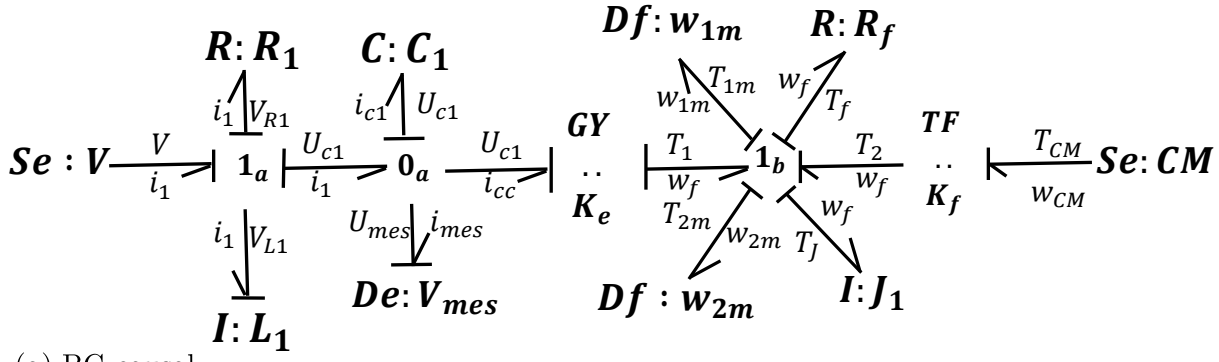
2.2.4 Génération des schémas-blocs pour la simulation

La forme causale d'un BG peut aussi être vue comme une forme compacte de schémas-blocs (Broenink, 1999). La procédure suivante, facilement implémentable, réalise la conversion automatique d'un BG en schémas-blocs :

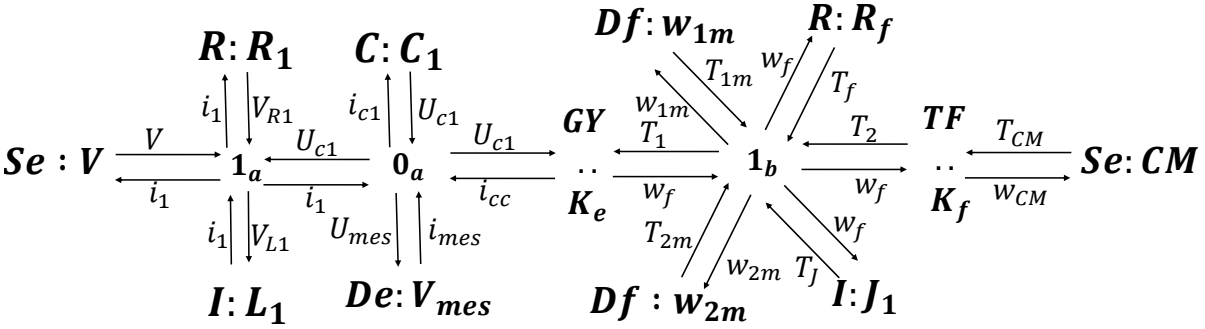
1. Convertir chaque liaison en deux signaux effort et flux. Le sens du signal est déterminé par la causalité.
2. Remplacer les éléments BG par des éléments de schémas-blocs. On déduit les signes des différentes liaisons par le sens des demi-flèches des liaisons BG.
3. Réorganiser graphiquement le schéma-bloc pour le rendre intelligible.

Les différentes étapes de cette procédure sont appliquées sur l'exemple mécatronique. À partir du BG causal (cf figure 2.8 (a)), les liaisons BG sont d'abord remplacées par leurs signaux respectifs (cf figure 2.8 (b)). Ensuite, les éléments BG sont remplacés par leurs éléments schémas-bloc associés (cf figure 2.9 (c)) ; par exemple, la source de tension est convertie en un bloc fournissant la tension du système (puisque la causalité impose l'effort). Enfin, les éléments sont réorganisés graphiquement et les signaux qui ont une de leurs extrémités non-connectée peuvent être supprimés (cf figure 2.9 (d)).

Notes : Dans la figure 2.9, la fonction $f(x)$ définit le coefficient de frottement en fonction de la vitesse angulaire.

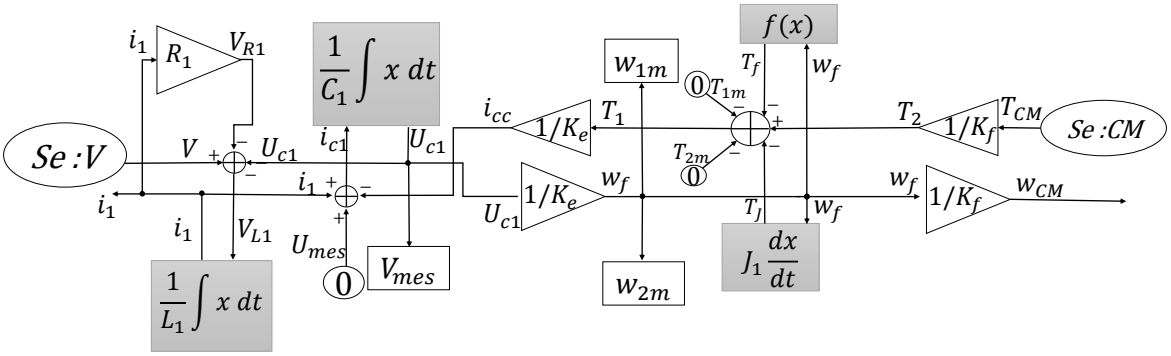


(a) BG causal

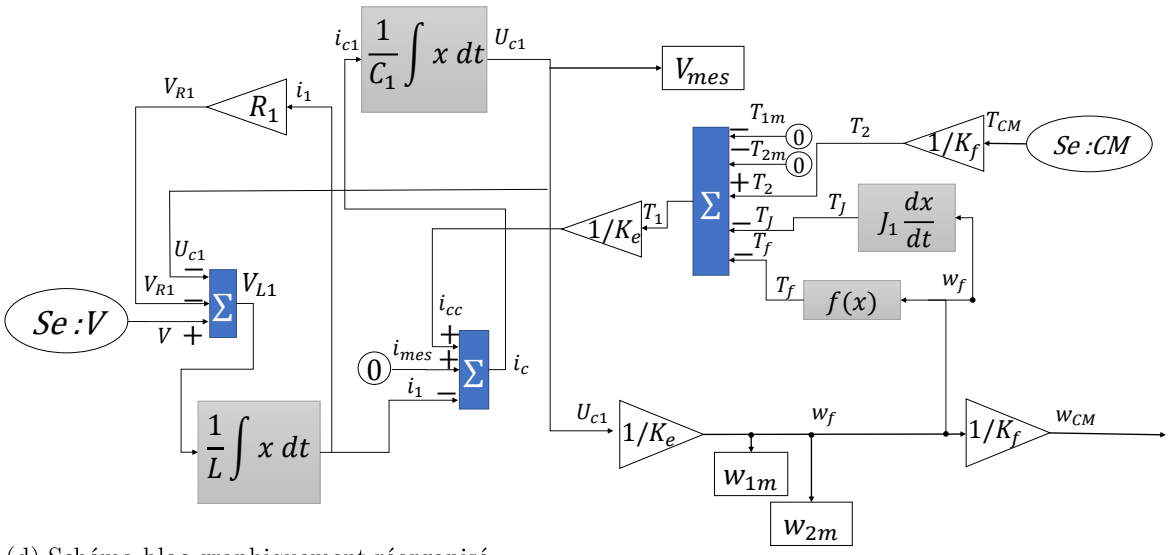


(b) BG avec liaisons substitués par leurs signaux respectifs (efforts/flux)

FIGURE 2.8 – Conversion des liaisons BG causal



(c) BG automatiquement converti en schéma-bloc



(d) Schéma-bloc graphiquement réorganisé

FIGURE 2.9 – Conversion en schéma-bloc du BG causal

2.2.5 Génération de RRA

Dans cette section, nous présentons une approche de génération des RRA, où ces relations s'obtiennent directement à partir des modèles BG par dualisation des détecteurs (Définition 2.2.1) (Abdallah, 2017).

Définition 2.2.1 (Dualiser un détecteur)**.** *Action qui consiste à transformer un détecteur d'effort De (respectivement de flux Df) par une source d'effort SSe (respectivement de flux SSf).*

La génération d'une RRA s'obtient normalement à partir d'un BG en causalité dérivée (pour éviter les problèmes de conditions initiales) par parcours de chemin causal (à partir d'un variable inconnue à une variable connue) pour éliminer les variables inconnues. Ce parcours de chemin causal va alors donner un graphe orienté où le noeud initial est la variable connue (correspondant soit à la sortie d'un capteur, soit à une variable de commande) (Borutzky, 2015 ; Ould Bouamama et al., 2006). On rappelle que la RRA initiale candidate est issue de l'équation de conservation issue d'une jonction connectée à au moins un capteur). Dans le cas de la procédure décrite ci-dessous, le parcours de chemin causal n'est pas explicitement réalisé, celui-ci étant implicitement inclus lors de la génération du schéma-bloc à partir du BG causal (puisque les schémas-blocs sont des graphes orientés).

La procédure se décompose suivant ces différentes étapes :

1. Les relations de redondances matérielles des capteurs sont d'abord générées. Une relation de redondance matérielle est une relation algébrique issue de la différence entre les valeurs fournies par deux capteurs mesurant la même grandeur physique. Ces RRA constituent alors des redondances dites matérielles. Pour la suite de cette procédure, il n'est conservé suite à l'exportation des RRA matérielles qu'au maximum un détecteur connecté à chaque jonction.
2. Les détecteurs sont tous dualisés (cf. définition 2.2.1).
3. Partant d'un détecteur dualisé, on cherche un chemin entre ce détecteur et une autre détecteur dualisé ou une source dont la valeur est connue. La RRA est alors une expression algébrique traduisant les lois de conservation de l'énergie entre les noeuds du chemin trouvé.
4. Affectation des causalités par l'algorithme SCAP au sous-ensemble BG précédemment exporté. Dans la mesure du possible, les éléments C et I sont en causalités dérivées.

5. En cas de conflit de causalité, on supprime la source dualisée entrainant le conflit, puis on reprend la procédure à l'étape 2.
6. Exporter le schéma-bloc à partir du BG causal précédemment obtenu.
7. L'estimation de l'effort (pour les capteurs Df) ou du flux (pour les capteurs De) est le résidu. Il est nul si le système est non défaillant.

Notes : La dualisation des détecteurs peut entrainer des conflits de causalité, puisque l'ajout de source réduit les degrés de liberté du système. Pour pallier ce problème, il est possible de supprimer la source dualisée entrainant le conflit, ce qui entraine cependant des relations contenant plus de variables.

Mise sous forme d'algorithme

Dans le cadre de ce mémoire, l'objectif est d'informatiser la supervision de SDH à l'aide de BG. Ainsi, nous préférons une formulation plus algorithmique de cette procédure, cette formulation est présentée sur l'algorithme 3 (cet algorithme sera étendu pour la gestion des BGH dans le chapitre 3 : algorithme 8).

Algorithme 3 Algorithme de génération des RRA

1: Create BG' a copy of BG
2: ▷ Export hardware redundancy relations
3: **for all** Element sensor s_i in BG' **do**
4: Get junction J_i connected to s_i
5: **for all** Element sensor s_j connected to J_i **do**
6: **if** $s_j \neq s_i$ **then**
7: Remove s_j from BG'
8: The difference of effort (or flow) mesured by s_j and s_i is the residue and should be zero in nominal situation
9: **end if**
10: **end for**
11: **end for**
12: ▷ Export other ARR
13: Dualize all sensors (Definition 2.2.1)
14: **for all** Element sensor s_i in BG' **do**
15: Define BG_{ARR_i} as a new empty bond graph
16: Define L a new empty stack
17: L.push(s_i)
18: **while** L is not empty **do**
19: x= L.pop()
20: **if** $x \notin BG_{ARR_i}$ **then**
21: Add x to BG_{ARR_i}
22: **if** x is a junction connected to a sensor $s_j \neq s_i$ **then**
23: Add s_j to BG_{ARR_i}
24: **else**
25: **for all** node elements w connected to x **do**
26: L.push(w)
27: **end for**
28: **end if**
29: **end if**
30: **end while**
31: Applied SCAP with derivative causality (Algorithme 1)
32: If causal conflict appears at a junction connected to a detector s_j, delete this detector and start again at line 15, if other causal conflict appears model should be revised.
33: Applied Block diagram generation procedure on bond graph BG_{ARR_i} to export block-diagram BD_{ARR_i}.
34: **if** s_i is a De detector **then**
35: The flow at De is the residue and should be zero in nominal situation
36: **else if** s_i is a Df detector **then**
37: The effort at Df is the residue and should be zero in nominal situation
38: **end if**
39: **end for**

Résultat

À partir de cet algorithme, nous pouvons ainsi générer les trois RRA du modèle présenté précédemment. La RRA_3 est la redondance matérielle entre les deux capteurs d'effort, soit : $RRA_3 = w_{1m} - w_{2m}$.

Les RRA_1 (figure 2.10 (a)) et RRA_2 (figure 2.11 (a)) sont générées par dualisation des capteurs. Cependant, pour ces deux générations, un conflit causal apparait. Ce conflit est résolu par l'élimination du capteur à l'origine du conflit (partie hachurée). La génération est alors relancée sans ce capteur, et aboutit à un schéma-bloc (figure 2.10 (b) et figure 2.11 (b)) permettant l'estimation des résidus.

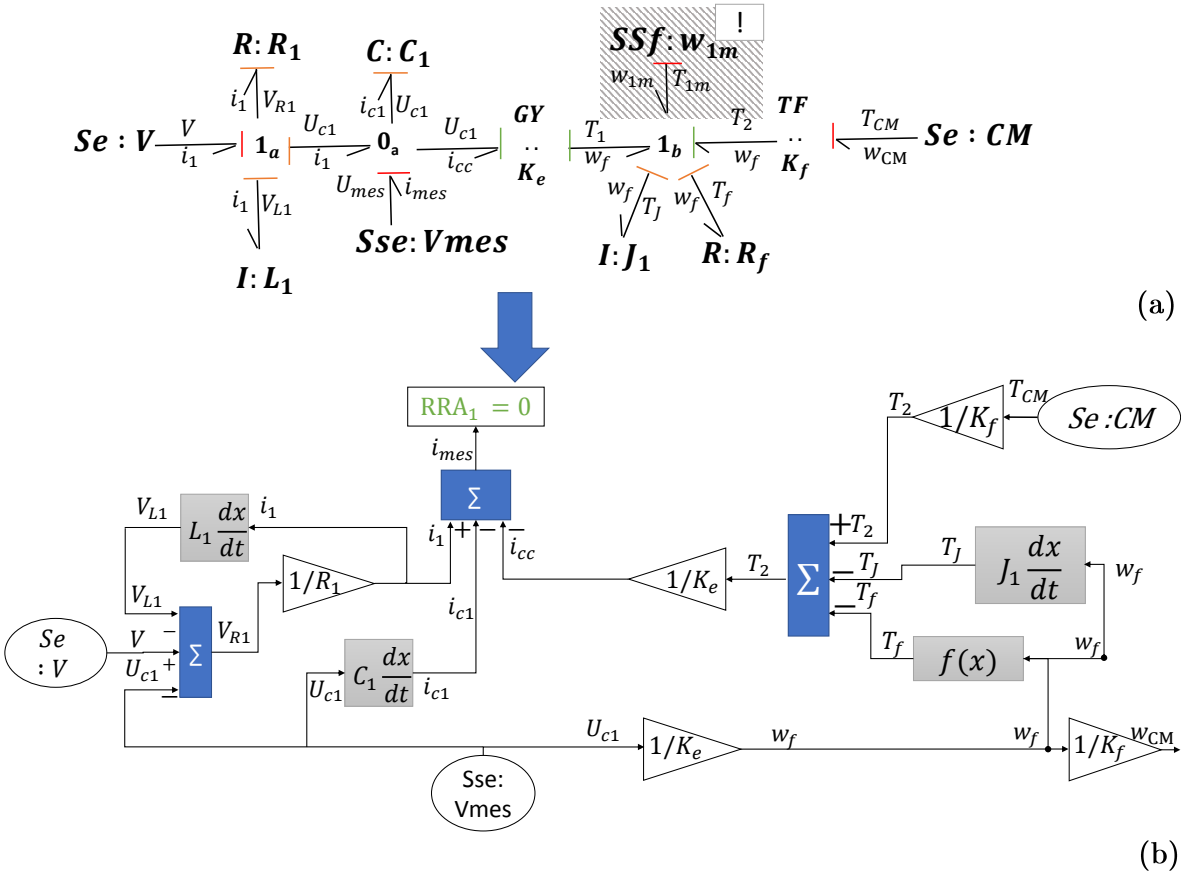


FIGURE 2.10 – Génération RRA1 - (a) BG causal (b) Schéma-bloc similaire

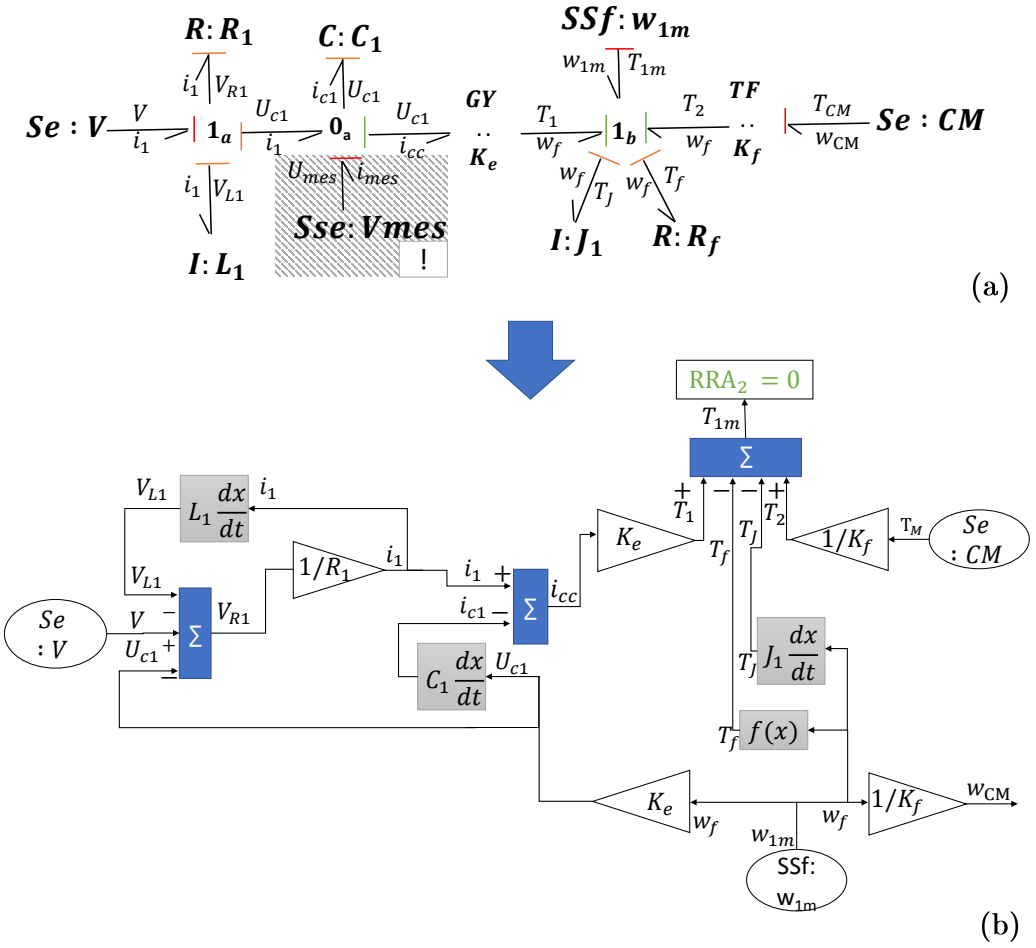


FIGURE 2.11 – Génération RRA2 - (a) BG causal (b) Schéma-bloc similaire

2.2.6 Génération de FSM

La matrice de signatures de fautes permet de détecter et éventuellement de localiser les défauts (cf Chapitre 1). Pour un système complexe, il est exhaustif et source d'omission d'énumérer l'ensemble des défauts qui peuvent potentiellement apparaitre. Une solution consiste à considérer que tout nœud du BG peut être un défaut, puis d'utiliser une approche fonctionnelle (cf. figure 2.4) pour l'associer au composant matériel réellement défaillant. La définition 1.4.6 exprimant la signature d'un résidu peut alors être réécrite selon la définition 2.2.2. La FSM associée au système mécatronique étudié est donnée par le tableau 2.1. Le nombres de capteurs est suffisant pour détecter l'ensemble des défaillances du système, cependant il est possible de ne localiser que les capteurs de vitesse en défauts.

Définition 2.2.2 (Génération de la signature d'un résidu à partir d'une RRA sous forme de BG)**.** *La structure (ou signature) d'un résidu r_i estimé à partir de la* RRA *sous forme causale : BG_{ARR_i}, par rapport à un ensemble de fautes génériques $F = \{f_j\}$ chacune associée à un nœud A_j est le mot binaire S_{r_j} composé des bits s_{ij} tels que :*

— $s_{ij} = 1$ *si* $A_j \in BG_{ARR_i}$

— $s_{ij} = 0$ *sinon*

Ib	0	0	0	0	0	0	0	0	0	1	1	1
Db	1	1	1	1	1	1	1	1	1	1	1	1
Fautes	$Se : V$	$Se : CM$	$R : R_1$	$I : L_1$	$C : C_1$	$GY : K_e$	$TF : K_f$	$R : R_f$	$I : J_1$	$De : V_{mes}$	$Df : w_{1m}$	$Df : w_{2m}$
RRA1	1	1	1	1	1	1	1	1	1	0	1	0
RRA2	1	1	1	1	1	1	1	1	1	1	0	0
RRA3	0	0	0	0	0	0	0	0	0	0	1	1

Tableau 2.1 – FSM du système mécatronique

2.3 Éléments des bond graphs pour les systèmes hybrides

A la différence des systèmes continus, les SDH intègrent des composants dont le comportement diffère selon leur mode de fonctionnement, un interrupteur peut par exemple transmettre ou non de la puissance selon son état ouvert ou fermé. Pour prendre en compte cette spécificité dans la modélisation bond graph, plusieurs solutions existent.

2.3.1 Résistance contrôlée

La résistance contrôlée est une résistance dont la valeur varie selon la valeur de la variable booléenne associée au commutateur. Dans le cas où l'interrupteur est en position fermée ($m = 1$), la valeur de R est égale à 0 ; dans le cas contraire ($m = 0$), elle est infinie. La figure 2.12 donne deux représentations bond graph associées à deux configurations matérielles.

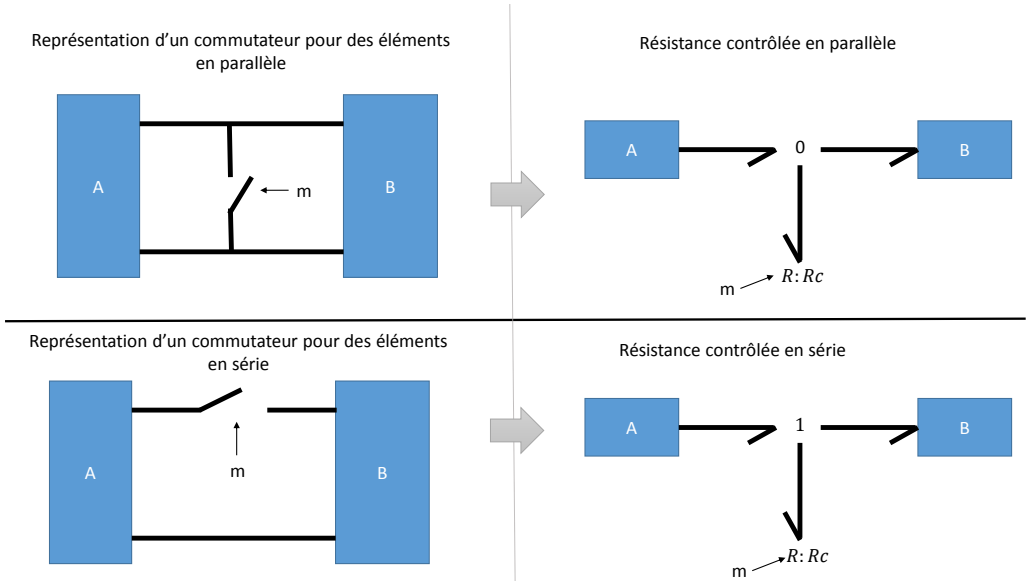


FIGURE 2.12 – Représentations à l'aide de bond graph de deux configurations matérielles à l'aide de résistances contrôlées

Ces valeurs théoriques ne sont en pratique pas possibles, car lorsque la résistance est en causalité conductance, le flux est donné par : $f = \frac{1}{R}e$. Pour pallier ce verrou, deux solutions sont envisageables : imposer une causalité résistance pour ne pas obtenir d'équations impossibles (DAUPHIN-TANGUY, 2000, p. 71), ou imposer une valeur paramétrique minimale et maximale.

2.3.2 Transformateur modulé avec résistance

Les commutations peuvent être représentés par l'association d'un transformateur avec une résistance (Stromberg, Top et Soderman, 1993). Le transformateur modulé représente une commutation idéal, et la résistance représente la résistance de l'élément commutatif à l'état fermé ou ouvert. L'avantage de cette solution par rapport à la résistance contrôlée est l'utilisation directe d'un booléen pour commander l'élément hybride (Dauphin-Tanguy, 2000, p. 72). Deux cas fonctionnels sont possibles, suivant la causalité du transformateur :

— Figure 2.13 (a) : $m = 0$ (état ouvert) l'élément impose un effort nul ; $m = 1$ (état fermé) le commutateur agit comme une résistance de valeur très faible dissipant la puissance.

— Figure 2.13 (b) : $1/m = 0$ (état ouvert) l'élément impose un flux nul ; $1/m = 1$ (état fermé) le commutateur agit comme une résistance de valeur très faible dissipant la puissance.

La figure 2.14 est une représentation à l'aide des bond graphs, de deux configurations matérielles, à l'aide du transformateur contrôlé.

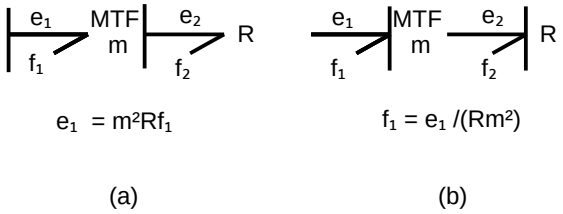


FIGURE 2.13 – Modèle de transformateur contrôlé avec résistance en causalité : (a) résistance (b) conductance

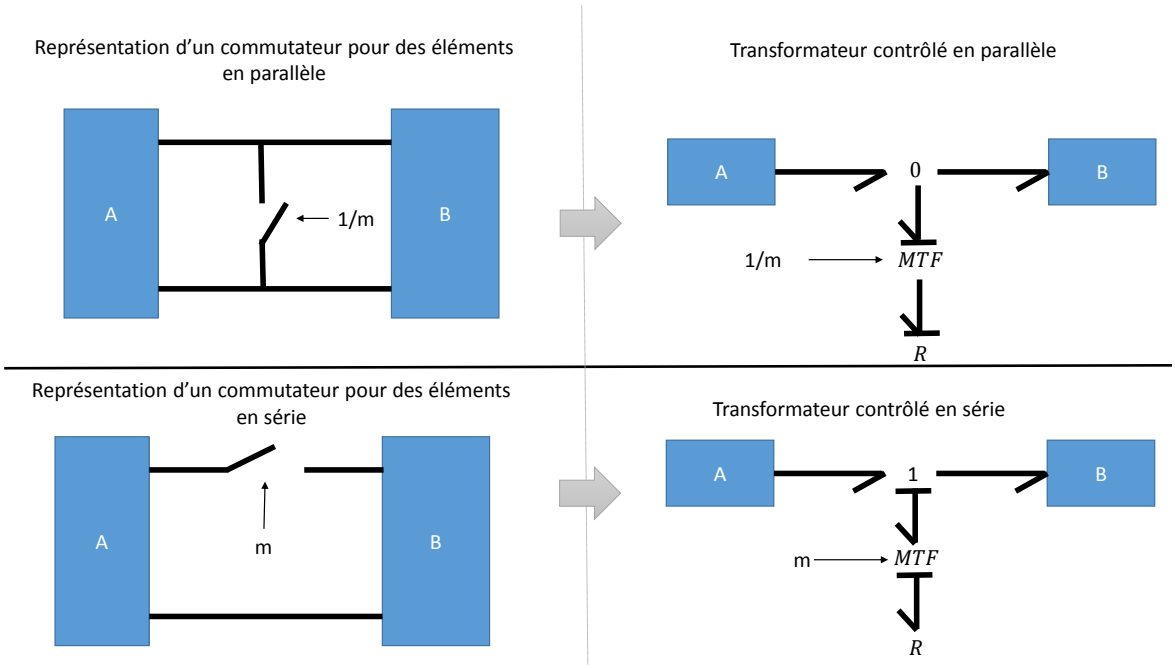


FIGURE 2.14 – Représentations à l'aide des bond graphs de deux configurations matérielles à l'aide de transformateurs contrôlés

2.3.3 Source commutée

La source commutée nommée Sw est une méthode de représentation idéale d'une commutation (Stromberg, Top et Soderman, 1993). Cette modélisation binaire équivaut à connecter et déconnecter, les différents éléments entre eux. Si la source commutée Sw est connectée à une jonction 1, à l'état fermé, cet élément est équivalent à une source d'effort nul ; à l'état ouvert, l'élément est équivalent à une source de flux nul. Réciproquement si cette source est connectée à une jonction 0, à l'état fermé, cet élément est équivalent à une source de flux nul, à l'état ouvert, l'élément est équivalent à une source d'effort nul (Figure 2.15). La causalité de cet élément est dynamique, c'est-à-dire qu'elle change pour chaque commutation.

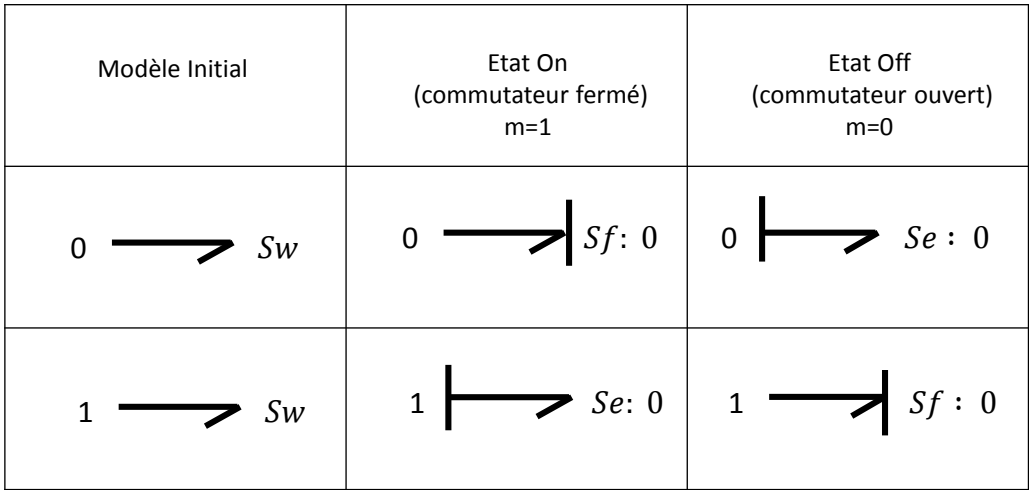


FIGURE 2.15 – Source commutée, représentation des états

La source commutée est obligatoirement reliée à une jonction, 1 si les deux éléments reliés par le commutateur sont en série, et à une jonction 0, si les deux éléments reliés par le commutateur sont en parallèle (Figure 2.16).

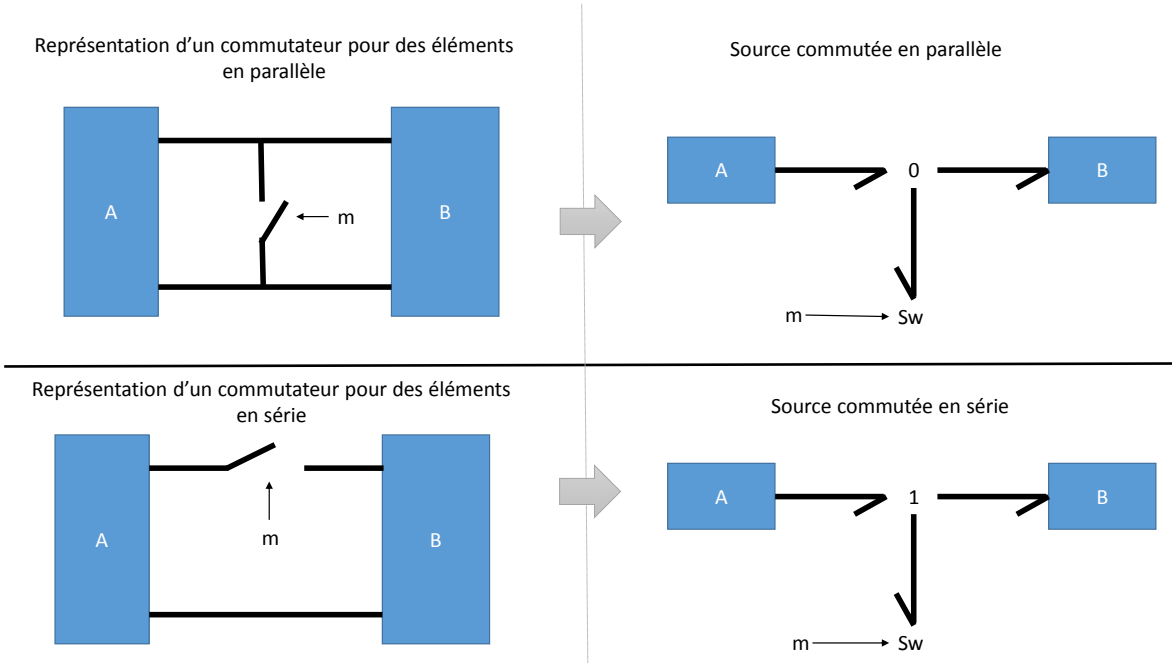


FIGURE 2.16 – Représentations à l'aide des bond graphs de deux configurations matérielles à l'aide de Sources commutées

2.3.4 Jonction contrôlée

Les jonctions contrôlées sont des éléments associés à un signal booléen de commande. À l'identique des jonctions conventionnelles, il existe deux types de jonctions contrôlées $0c$ et $1c$ (Mosterman et Biswas, 1998). Dans le cas où le booléen est à l'état «ON» (interrupteur fermé), la jonction contrôlée agit alors comme une jonction conventionnelle (Figure 2.17). Dans le cas où le booléen est à l'état «OFF» (interrupteur ouvert), la jonction $0c$ impose un effort nul à l'ensemble de ses liaisons (ou un flux nul pour les jonctions $1c$). La difficulté d'utilisation de telles jonctions réside dans le changement de causalité qui peut apparaitre sur commutation. La figure 2.18 présente la représentation à l'aide de bond graph de deux configurations matérielles à l'aide de jonctions contrôlées.

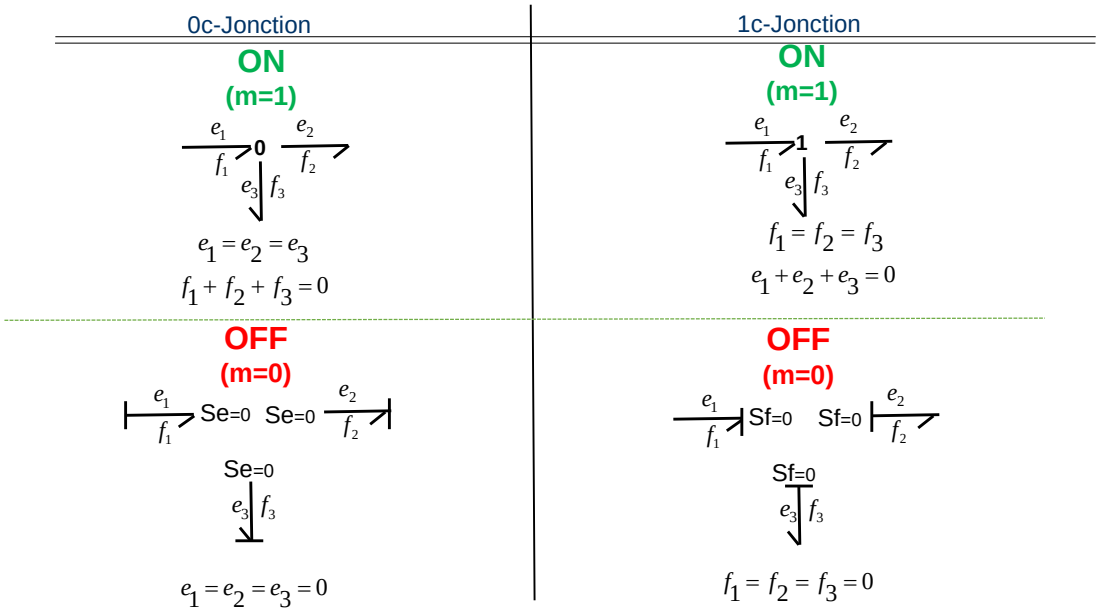


FIGURE 2.17 – Jonction contrôlée

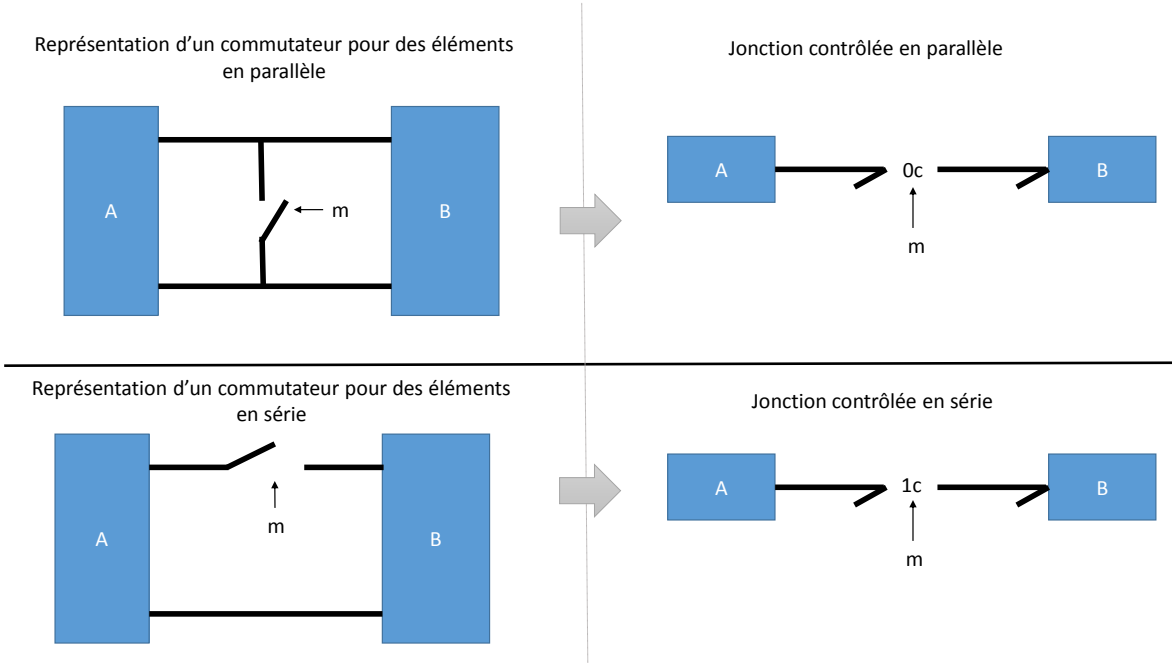


FIGURE 2.18 – Représentations à l'aide de bond graphs de deux configurations matérielles à l'aide de jonctions contrôlées

2.3.5 Jonction à puissance commutée

La jonction à puissance commutée contient deux liaisons déterminantes (Définition 4.7.1) mutuellement exclusives (nommée m et $\bar{m}$ sur la figure 2.19) (Umarikar et Umanand, 2005). C'est à dire que lorsque la liaison m est active, alors $\bar{m}$ ne transfère plus de puissance (la liaison est inactive). Réciproquement, si $\bar{m}$ est actif, m devient inactive. Ces éléments sont encore plus stricts du point de vue de la causalité. En effet, ils imposent la causalité à l'ensemble des ports. Ceci suppose une réflexion sur la causalité dès la modélisation, puisque dès la création du modèle, il est nécessaire de connaitre les liaisons déterminantes.

Pour représenter des éléments commutatifs simples (commutateur à deux états), il est nécessaire d'ajouter une source d'effort ou de flux nul (Figure 2.20). Cet ensemble à l'état ouvert impose un effort (ou un flux) nul à l'ensemble des liaisons. À l'état fermé, cet ensemble agit comme une jonction conventionnelle ; la source d'effort ou de flux nul n'est plus connectée.

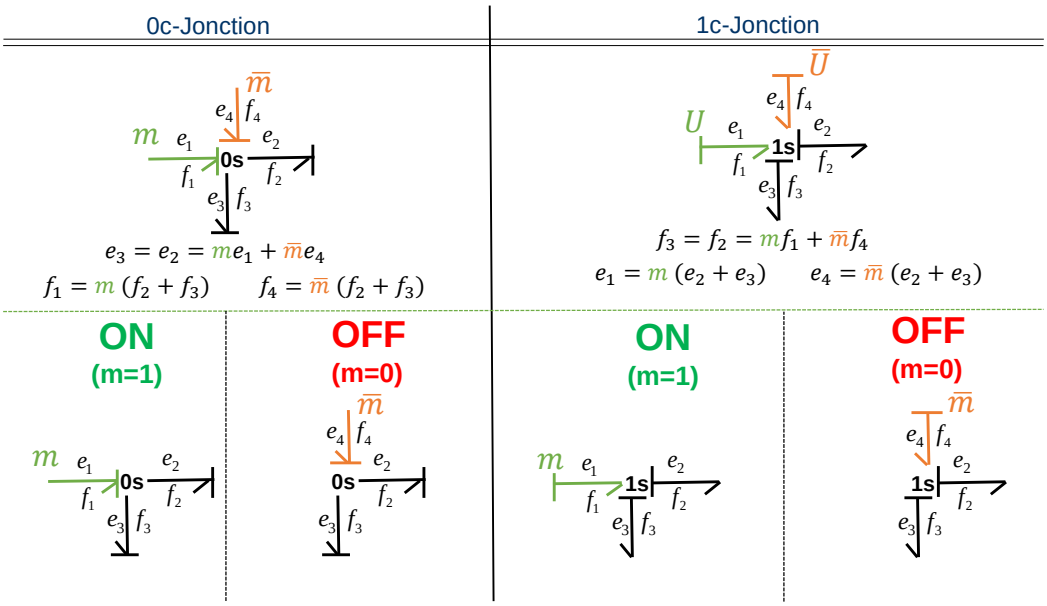


FIGURE 2.19 – Jonction de puissance commutée

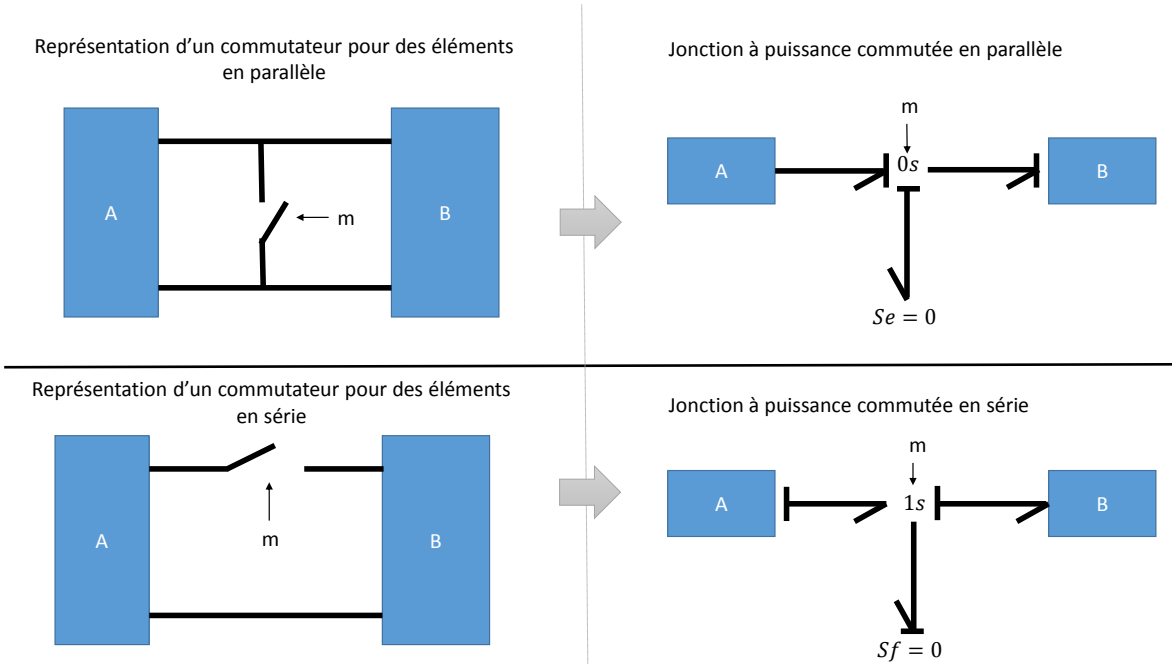


FIGURE 2.20 – Représentations à l'aide de bond graphs de deux configurations matérielles à l'aide de jonctions à puissance commutée

L'intérêt réside dans la représentation des éléments commutatifs complexes (présentant plus que deux états donc modélisés par plus que deux ports) ; dans ce cas, ces jonctions permettent de sélectionner la source d'effort (ou de flux) à l'aide d'un unique élément (Figure 2.21).

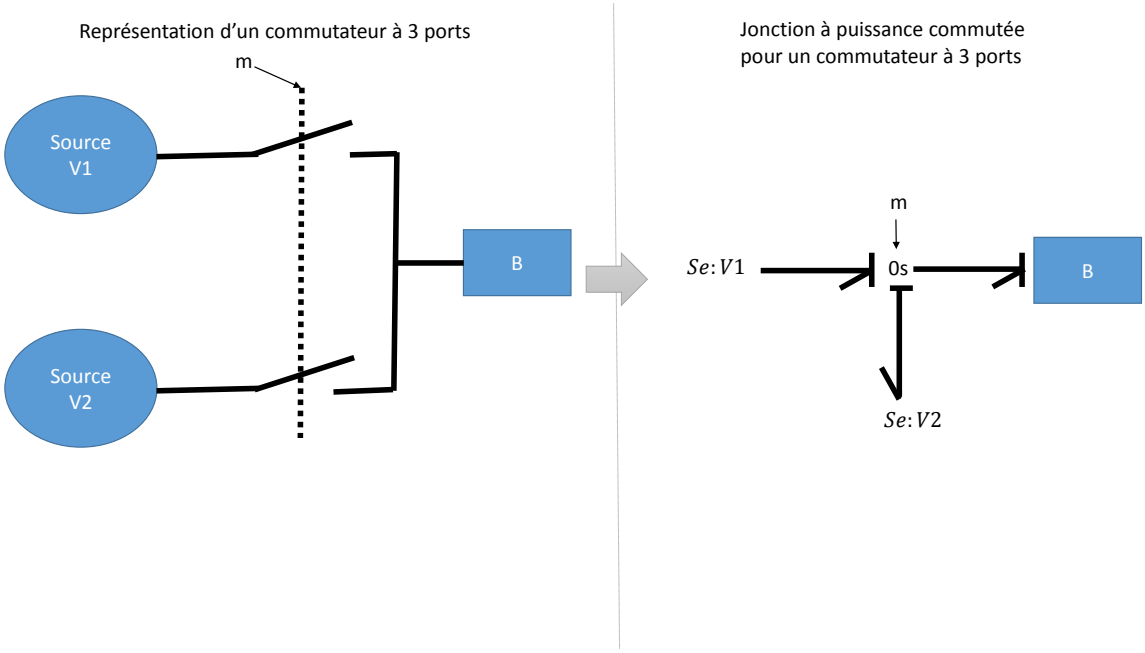


FIGURE 2.21 – Représentations à l'aide de bond graphs d'un commutation complexe (plus de deux ports) à l'aide de jonctions à puissance commutée

2.3.6 «Switching junction» (20sim)

20-sim est un logiciel de modélisation BG proposant un élément commutatif dénommé « Switching junction », possédant 3 ports (bien que le nom porte à confusion, cet élément n'a aucun rapport avec la jonction contrôlée). Comme pour les jonctions conventionnelles, il existe deux types de « switching junction » : $X0$ et $X1$ (20SIM, 2016, p 554). La jonction $X0$ impose deux flux et un effort à ses ports, la jonction $X1$ impose un effort et deux flux à ses ports. La switching junction est donc un élément possédant de fortes contraintes causales.

Dans le cas où le booléen est à l'état «ON» (interrupteur fermé), la « switching junction » agit alors comme une jonction conventionnelle (Figure 2.22). Dans le cas où le booléen est à l'état «OFF» (interrupteur ouvert), la jonction $X0$ impose un effort nul à ses liaisons non déterminantes (ou flux nul pour $X1$). La liaison déterminante se voit imposer un flux nul (jonction $X0$) ou un effort nul (jonction $X1$).

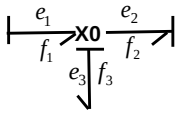
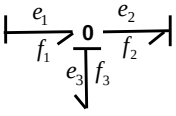
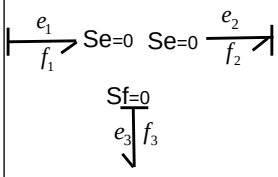
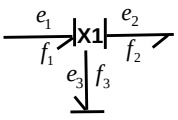
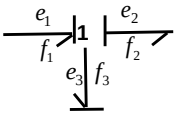
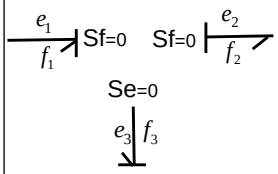
Type	Modèle initial	Commutateur fermé (ON) m=1	Commutateur ouvert (OFF) m=0
X0-Jonction			
X1-Jonction			

FIGURE 2.22 – "Switching junction" (20sim)

Pour implémenter ce type de jonction (Figure 2.23), il est nécessaire de créer un ensemble avec une résistance. Cette résistance de faible valeur représente la résistance de l'élément commutatif à l'état fermé. L'ensemble devient alors totalement équivalent à un transformateur modulé avec résistance.

Notes : À l'origine les jonctions contrôlées étaient différentiées des jonctions standard par un indice lié à la commutation (MOSTERMAN et BISWAS, 1998). Au fil du temps,

la notation communément acceptée devint $X0$ et $X1$. Il est donc regrettable que cette notation soit utilisée par 20sim pour les "switching junction" à causalité fixe. Pour lever toutes ambiguïtés dans cette thèse nous utilisons la notation $0c$ et $1c$ pour définir les jonctions contrôlées.

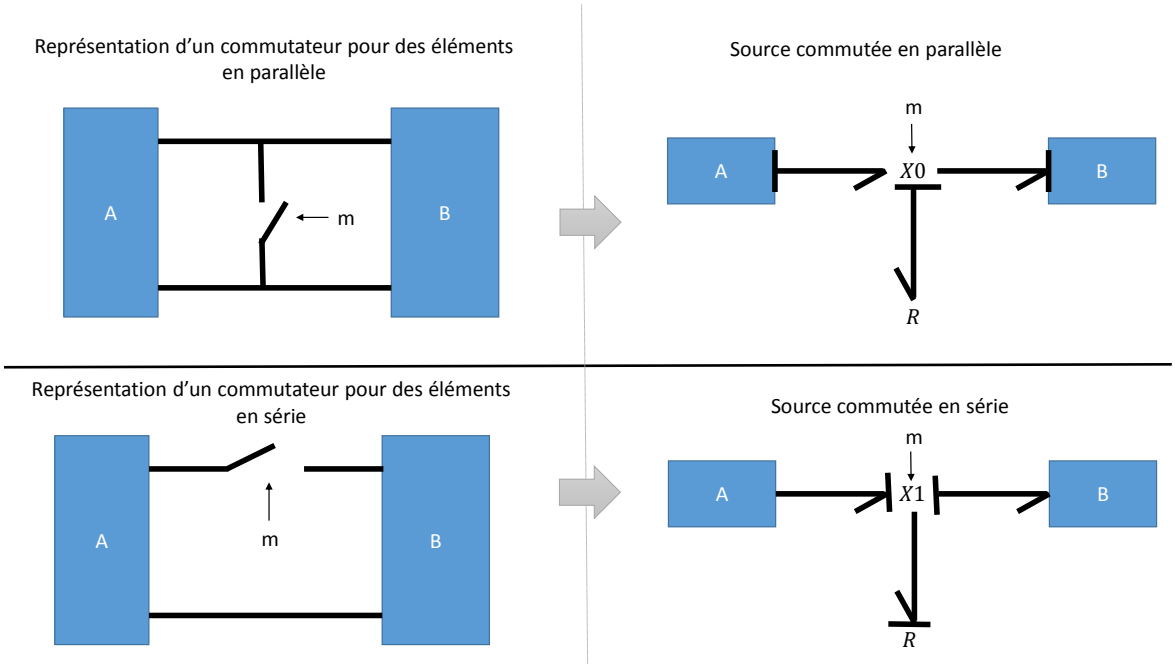


FIGURE 2.23 – Représentations à l'aide de bond graphs de deux configurations matérielles à l'aide de «Switching junction» (20sim)

2.3.7 Tableau de synthèse

La figure 2.24 résume l'ensemble des solutions BG pour représenter les éléments hybrides. Les transformateurs et résistances contrôlées ne modélisent pas les commutations de manière idéale. En effet, un élément résistif de faible valeur est toujours présent, ce qui dans le cas des systèmes complexes peut entrainer des phénomènes de rigidité du système.

Les jonctions à puissance commutée, et switching jonction de 20sim, imposent de fixer strictement la causalité dès la création du modèle. Ce qui est une contrainte forte.

La source commutée, une fois liée à une jonction, équivaut totalement à une jonction contrôlée. Cependant, on lui préférera cette dernière plus compacte. Pour la suite des développements, la jonction contrôlée sera donc privilégiée, car elle modélise les commutations de manière idéale avec des contraintes causales minimales. Concernant

son implémentation, nous devrons cependant gérer au mieux les conflits de causalité, lorsque les ensembles interconnectés ne possèdent pas de degré de liberté sur l'affectation des causalités.

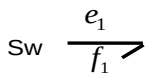
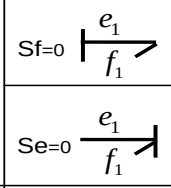
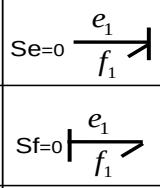
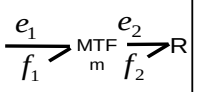
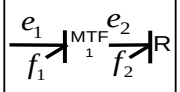
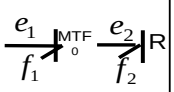
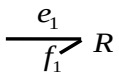
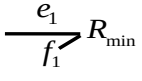
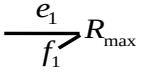
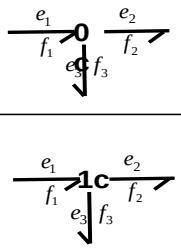
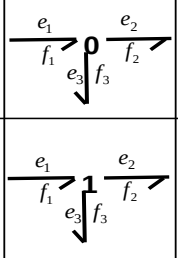
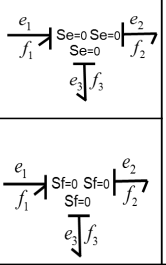
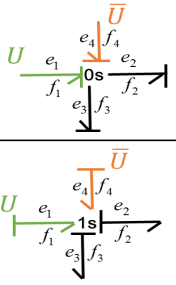
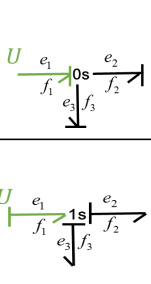
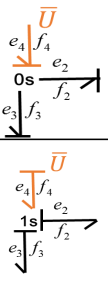
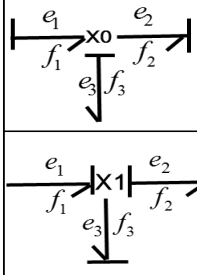
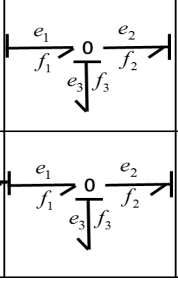
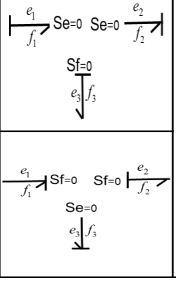
Eléments	Représentation	ON	OFF	Description	Auteur
Source commutée				Commutation entre source de flux nul et effort nul	Stromberg et al 1998
Transformateur Modulé				Module du transformateur change de 1 à 0	Dauphin-Tanguy 2000
Résistance contrôlée				Résistance variable d'une valeur minimale à l'infini	Dauphin-Tanguy 2000
Jonction contrôlée				La jonction est standard à l'état ON / La jonction est une source nulle à l'état OFF	Mosterman and Biswas 1998
Jonction à puissance commutée				Les liaisons U et $\overline{U}$ sont mutuellement exclusives	Umarikar and Uanand 2005
Switching junction				La jonction est standard à l'état ON / La jonction est composée de différentes sources nulles à l'etat OFF	20 sim

FIGURE 2.24 – Synthèse des différents représentation BG des éléments hybrides

2.4 Les logiciels utilisant les bond graphs

De nombreux logiciels utilisant la modélisation à partir des bond graphs pour représenter, voir simuler un système existent. Certains sont à l'état d'abandon ou de prototype, d'autres sont exploités commercialement. Dans cette section, une revue détaillée des logiciels opérationnels est présentée et argumentée.

2.4.1 Logiciels nativement dédiés aux Bonds Graphs

20-sim

20-sim est un logiciel commercial développé par Controlab et Twente University. Ce logiciel est orienté vers la simulation de BG. Les équations peuvent être exportées dans un fichier texte ou dans MATLAB. Une nouvelle extension 20sim-4c permet l'exportation de BG causaux et schémas-blocs à destination d'automates programmables, permettant ainsi l'implémentation de la partie contrôle, directement à partir de BG. La représentation des BGH est possible, via l'utilisation de jonctions, dénommées switching junction «X1» ou «X0» (cf. sous-section 2.3.6).

Symbols 2000

Symbols 2000 est un logiciel qui réalise la simulation et l'analyse des BG représentant les systèmes continus. Il permet l'exportation des systèmes vers MATLAB/SIMULINK au format unitaire, mais n'exporte pas directement des schémas-blocs. Une interface spécifique génère la FSM ; ceci représente un avantage important dans le cas du diagnostic des systèmes. Ce logiciel ne permet pas de représenter les systèmes soumis à des commutations (**AmalenduMukherjee_2001**).

MOTHS

MOdeling and Transformation of HBGs for Simulation (MOTHS) (Roychoudhury et al., 2011) est un logiciel récent, qui possède une approche innovante dans l'exportation directe de schémas-blocs simulink à partir de BGH acausaux. La causalité est recalculée en cours de simulation, selon le mode de fonctionnement du système. Une des conséquences est que le type de grandeur associée au signal peut être modifié (par exemple, un signal qui représentait une tension pourra représenter une intensité). Cette modification n'est pas problématique pour la simulation, puisqu'à tout moment, l'exécution peut être mise en pause. Les valeurs lues peuvent dans ce cas être analysées

a posteriori. Cependant il est difficile d'envisager des solutions de validation HIL, ou des solutions de contrôle avec de telles contraintes. Enfin ce logiciel, n'est ni disponible à la vente, ni ouvert au développement ; il semble uniquement dédié à une utilisation en interne.

LibBondGraph

«LibBondGraph», est un logiciel opensource disponible sur une plateforme de développement ouverte (Dupuis, 2009). Ce projet n'est toutefois plus maintenu et reste très incomplet. Cependant, cet outils est très bien construit autour d'une librairie BG écrite en C++. L'architecture informatique de ce logiciel inspirera partiellement nos constructions.

CAMP-G

CAMP-G est un préprocesseur, permettant la génération d'équations à partir d'une modélisation BG continue (Granda et Reus, 1997). Ces équations peuvent être produites selon les formalismes suivant : MATLAB/SIMULINK, ACSL, DSL, FORTRAN, C. Une fois générés ces éléments doivent ensuite être simulés. Ce logiciel ne traite que de la gestion des BG continus.

2.4.2 Logiciels avec extension permettant l'utilisation des Bonds Graphs

Simcenter Amesim

Simcenter Amesim est un logiciel pour la création et simulation des systèmes physiques (AMESIM, 2016). Initialement, la modélisation des systèmes se réalisait principalement à l'aide de BG. Les éléments discrets sont modélisés de manière non idéale, en effet la causalité est forcée en reliant les entrées d'un sous-modèle à la sortie d'un autre sous-modèle. Depuis le rachat de Imagine, anciennement société éditrice de Amesim, par Siemens, le logiciel fait partie d'une suite logicielle permettant la création des schémas sous CAO, la simulation de ceux-ci, la gestion des stocks et la gestion du cycle de vie du projet (progiciel de gestion). Dans ce cadre, les BG sont devenus une librairie de cette plateforme plus globale.

Dymola/Modelica

Modelica est un langage orienté objet multiphysique. Ce langage dispose d'un environnement graphique : chaque sous-composant du système est représenté par un bloc d'équations, une icône et des ports de connexions. Les connexions représentent un couplage entre les blocs (fils électriques, canalisations, etc.) (Modelica_Association, 2012). Un compilateur de Modelica est nécessaire pour transformer le code en langage C interprétable pour la simulation. Certaines routines permettent l'automatisation partielle du processus de conversion des schémas réalisés sous CAO vers le langage modelica (Juhász et Schmucker, 2008). Modelica est un langage causal et acausal ; ainsi, si elle n'est pas directement définie par l'utilisateur, la causalité du système est déterminée durant la simulation. Cela permet de représenter les systèmes hybrides (avec toutefois des temps de simulation lents, du fait du changement de causalité pendant la simulation).

Dymola est un logiciel de modélisation et simulation développé par Dassault Système. Il utilise le langage de modélisation Modelica. À partir d'une modélisation sous Modelica, il permet la mise en simulation, l'exportation sous MATLAB, et la compilation sous Langage C pour implémentation sur matériel réel.

La modélisation des BG s'effectue avec la libraire BondLib, sous leur forme causale ou acausale (Cellier et Nebot, 2005). Lorsque les éléments sont modélisés sous formes causales, cela se traduit par un bloc. Lorsque les éléments sont traités sous formes acausales, les éléments sont convertis en modèle de type Modelica. Pour représenter les SDH, il faut introduire des éléments commutables à partir des bibliothèques standard Dymola. Ces éléments commutables possèdent des causalités fixes ; cela ajoute de nouvelles contraintes sur l'ensemble du système (Ronkowski, 2008).

Simulink

Les BG avec causalités fixées peuvent être considérés comme des schémas-blocs écrits de manière condensée (Broenink, 1999). Une boite à outils a été développée sous Simulink, pour représenter les BG causaux (Umarikar, Mishra et Umanand, 2006). De nombreuses interfaces sont réalisées avec MATLAB pour générer les équations, interfacer avec du matériel réel, extraire les données et les paramètres du système. Cependant, la nécessité de fixer dès le départ et manuellement la causalité limite grandement l'utilisation de ce logiciel.

2.4.3 Limites des logiciels existants

Le tableau 2.2 synthétise les avantages et inconvénients de chaque logiciel pour la modélisation des systèmes hybrides à partir du langage BGH choisi précédemment. Les logiciels Symbols 2000 et CAMP-G ne peuvent être choisis, puisqu'ils ne permettent pas de modéliser les systèmes hybrides. Dymola et Simulink-Bondlibs ne sont pas des logiciels orientés sur l'utilisation des BGH. 20sim, et Amesim pourraient être de bons candidats ; cependant, ces logiciels ne modélisent pas les commutations de manière idéales. De plus, ces deux logiciels sont totalement propriétaires ; ils ne permettent pas la production automatisée des modèles à l'extérieur de leurs propres interfaces. MOTHS répond à la majorité des problématiques exigées, mais les schémas-blocs générés sont difficilement exploitables pour être embarqués (diagnostic et HIL), du fait que la simulation doit être mis en pause à chaque changement de mode. De plus, ce logiciel n'est ni disponible à la vente, ni ouvert au développement.

Logiciels	simulation	génération de schéma-bloc	génération de codes	diagnostic	Gestion des systèmes hybrides	code ouvert et accessible
20sim	++	–	++	+	avec causalité imposée	-
Symbols2000	++	–	++	++ (inclus la génération de FSM)	non	-
MOHTS	++	+ (limitation sur reconfiguration des connexions)	+	+	totale	–
libBondGraph-1.0.0	+	–	–	-	avec causalité imposée	++
CAMP-G	-	–	–	-	non	–
Amesim	++	-	+	–	avec causalité imposée	–
Dymola	++	++	++	+	avec causalité imposée	-
Simulink-Bondlibs	++	–	–	+	avec causalité imposée	–

TABLEAU 2.2 – Comparaisons des différents logiciels

2.5 Conclusion

Les outils pour modéliser, analyser, simuler un système à l'aide du formalisme BG ont été présentés. Dans un premier temps, nous nous sommes intéressés aux systèmes continus. La procédure d'affectation des causalités sur un modèle BG a été traduite sous forme d'algorithme, ce qui facilite la génération automatique de ce BG. Un deuxième algorithme est proposé pour traduire un BG en schémas-blocs, ceci facilitant la simulation du système. Un troisième algorithme permet d'extraire automatiquement les RRA nécessaires à la réalisation du diagnostic. L'analyse des différentes possibilités pour représenter les éléments hybrides d'un SDH montre que la jonction contrôlée est la solution la plus adaptée, car elle permet de modéliser les commutations de manière idéale avec des contraintes causales minimales. Une revue détaillée des logiciels utilisant la modélisation BG montre qu'il n'existe pas de logiciel ouvert permettant la génération de schémas-blocs ou d'équations, pour la simulation et le diagnostic des SDH possédant un très grand nombre de modes.

En se basant sur cette synthèse, il parait donc intéressant pour répondre à notre besoin de développer un nouveau logiciel complet, en s'inspirant des logiciels existants (notamment libBondGraph-1.0.0 et MOHTS). Ce logiciel devra respecter les contraintes suivantes :

1. Permettre la simulation et la validation SIL et HIL
2. Utiliser des jonctions contrôlées
3. Traiter les systèmes hybrides complexes
4. Générer les outils d'aide à la détection de défauts et à l'isolation
5. Générer les RRA pour la mise en place du diagnostic
6. Générer des schémas-blocs pour le diagnostic embarqué
7. Être standard dans l'entrée des modèles ce qui permettra au choix : la génération automatique de modèles BGH à partir de données industrielles, ou l'édition graphique manuelle.

Bibliographie du présent chapitre

Borutzky, W. (2010). *Bond Graph Methodology*. London : Springer London.

Paynter, H. M. et al. (1961). *Analysis and Design of Engineering Systems : Class Notes for M.I.T. Course 2.751*. Cambridge, Mass. : M.I.T. Press.

Mosterman, P. J. et al. (1998). *A Theory of Discontinuities in Physical System Models*. Journal of the Franklin Institute 335.3, p. 401-439.

Abdallah, I. et al. (2017). *Event Driven Hybrid Bond Graph for Hybrid Renewable Energy Systems Part I : Modelling and Operating Mode Management*. International Journal of Hydrogen Energy.

Merzouki, R. et al. (2012). *Intelligent Mechatronic Systems : Modeling, Control and Diagnosis*. Springer Science & Business Media. 960 p.

Šarga, P. et al. (2012). *Simulation of Electrical System Using Bond Graphs and MATLAB/Simulink*. Procedia Engineering. Modelling of Mechanical and Mechatronics Systems 48, p. 656-664.

Niu, G et al. (2015). *Fault Diagnosis of Locomotive Electro-Pneumatic Brake through Uncertain Bond Graph Modeling and Robust Online Monitoring*. Mechanical Systems and Signal Processing 50-51, p. 676-691.

Loureiro, Rui et al. (2014). *Extension of the Bond Graph Causality Inversion Method for Fault Detection and Isolation*. Mechatronics 24.8, p. 1042-1049.

Gawthrop, P. et al. (1996). *Metamodelling : For Bond Graphs and Dynamic Systems*. Prentice Hall.

Karnopp, D. et al. (1968). *Analysis and Simulation of Multiport Systems : The Bond Graph Approach to Physical System Dynamics*. M.I.T. Press. 248 p.

Broenink, J. F. (1999). *Introduction to Physical Systems Modelling with Bond Graphs*. SiE Whitebook on Simulation Methodologies, p. 1-31.

Abdallah, Ibrahim (2017). *Event-Driven Hybrid Bond Graph : Application : Hybrid Renewable Energy System for Hydrogen Production and Storage*. Lille 1.

Borutzky, W. (2015). *Bond Graph Model-Based Fault Diagnosis of Hybrid Systems*. Springer International Publishing.

Ould Bouamama, B. et al. (2006). *Supervision of an Industrial Steam Generator. Part I : Bond Graph Modelling*. Control Engineering Practice 14.1, p. 71-83.

Dauphin-Tanguy, G. (2000). *Les bond graphs*. Hermes Science Publications.

Stromberg, J. E. et al. (1993). *Variable Causality in Bond Graphs Caused by Discrete Effects*. ICBGM 1993. San Diego.

Umarikar, A. C. et al. (2005). *Modelling of Switching Systems in Bond Graphs Using the Concept of Switched Power Junctions*. Journal of the Franklin Institute 342.2, p. 131-147.

20sim (2016). *20-Sim 4.6 Reference Manual*. Getting Started with 20-sim.

Roychoudhury, I. et al. (2011). *Efficient Simulation of Hybrid Systems : A Hybrid Bond Graph Approach*. Simulation 87.6, p. 467-498.

Dupuis, J. F. (2009). *libBondGraph*. url : `https://sourceforge.net/projects/libbondgraph/?source=navbar`.

Granda, J. J. et al. (1997). *New Developments in Bond Graph Modeling Software Tools : The Computer Aided Modeling Program CAMP-G and MATLAB*. Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics. T. 2, 1542-1547 vol.2.

AMESIM (2016). *Simcenter Amesim Platform : Siemens PLM Software*. url : `https://www.plm.automation.siemens.com/fr/products/lms/imagine-lab/amesim/platform/index.shtml`.

Modelica_Association (2012). *Modelica Language Specification*.

Juhász, T. et al. (2008). *CAD to SIM : CAD Model Conversion for Dymola-Based Mechatronic Simulation*. Tenth International Conference on Computer Modeling and Simulation (Uksim 2008), p. 289-294.

Cellier, F. E. et al. (2005). *The Modelica Bond Graph Library*. Proc. of the Modelica Conference, 2005, p. 57-65.

Ronkowski, M. (2008). *Modelling of Electrical Machines Using the Modelica Bond-Graph Library*. Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th, p. 880-886.

Umarikar, A. C. et al. (2006). *Bond Graph Simulation and Symbolic Extraction Toolbox in MATLAB/SIMULINK*. Indian Institute of Science. 86.1.

Chapitre 3

Génie logiciel pour la création de modèles de supervision des SDH

3.1 Introduction

La production d'un logiciel informatique scientifique industriel est un processus long et complexe. Dans le cadre de ce mémoire, l'accent sera mis sur la résolution des verrous scientifiques, la conception d'architecture et d'algorithmes, ainsi que la mise en application des méthodes sur un exemple industriel. L'étude se déroule donc uniquement dans un contexte de prototypage logiciel. Le développement complet du code et des Interfaces Homme-Machine (IHM) et la validation et certification de cette démarche ne seront pas évoqués ici. Ces étapes nécessitent des investissements plus conséquents, qui constitueront les perspectives industrielles de ces travaux.

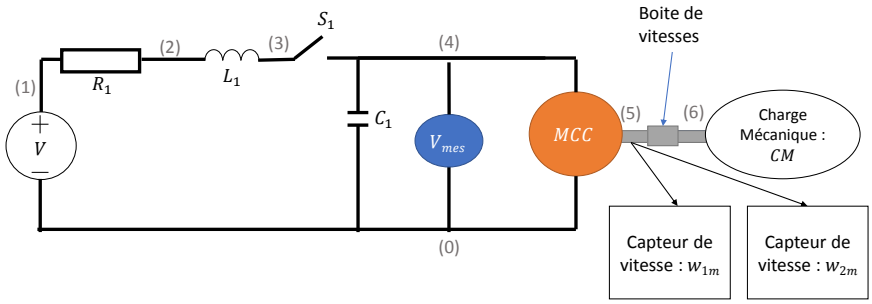
Dans une première partie, l'exemple mécatronique est enrichi d'un élément discret dans l'objectif d'obtenir un SDH. L'architecture logicielle générale du logiciel est ensuite présentée. De nouveaux algorithmes pour gérer les systèmes hybrides sont ainsi conçus. La production d'un langage informatique standard de description des BGH est proposé. Enfin, les choix informatiques de développement du logiciel sont décrits.

3.2 Exemple mécatronique hybride

Le système continu, support pédagogique présenté au chapitre 2 (figure 2.4) est enrichi d'un élément hybride : un contact S1. Cet exemple est modifié par l'ajout d'un contact $S1$ (figure 3.1). Cet élément présente deux modes :

— OFF, le contact est en position ouverte.

— ON, le contact est en position fermée.



s

FIGURE 3.1 – Exemple de système hybride mécatronique

La modélisation BG de ce contact correspond à une jonction contrôlée de type $1c$ (figure 3.2 (a)). En effet, le contact est placé en série entre l'ensemble composé de l'alimentation en tension (V), de la résistance (R_1), de l'inductance (L_1), et l'ensemble composé du moteur électrique (MCC), de la capacité (C_1), et du capteur de tension (V_{mes}). À l'état ouvert (OFF), la jonction est remplacée par des sources de flux nul (figure 3.2 (b)). À l'état fermé (ON), la jonction est remplacée par une jonction conventionnelle (figure 3.2 (c))

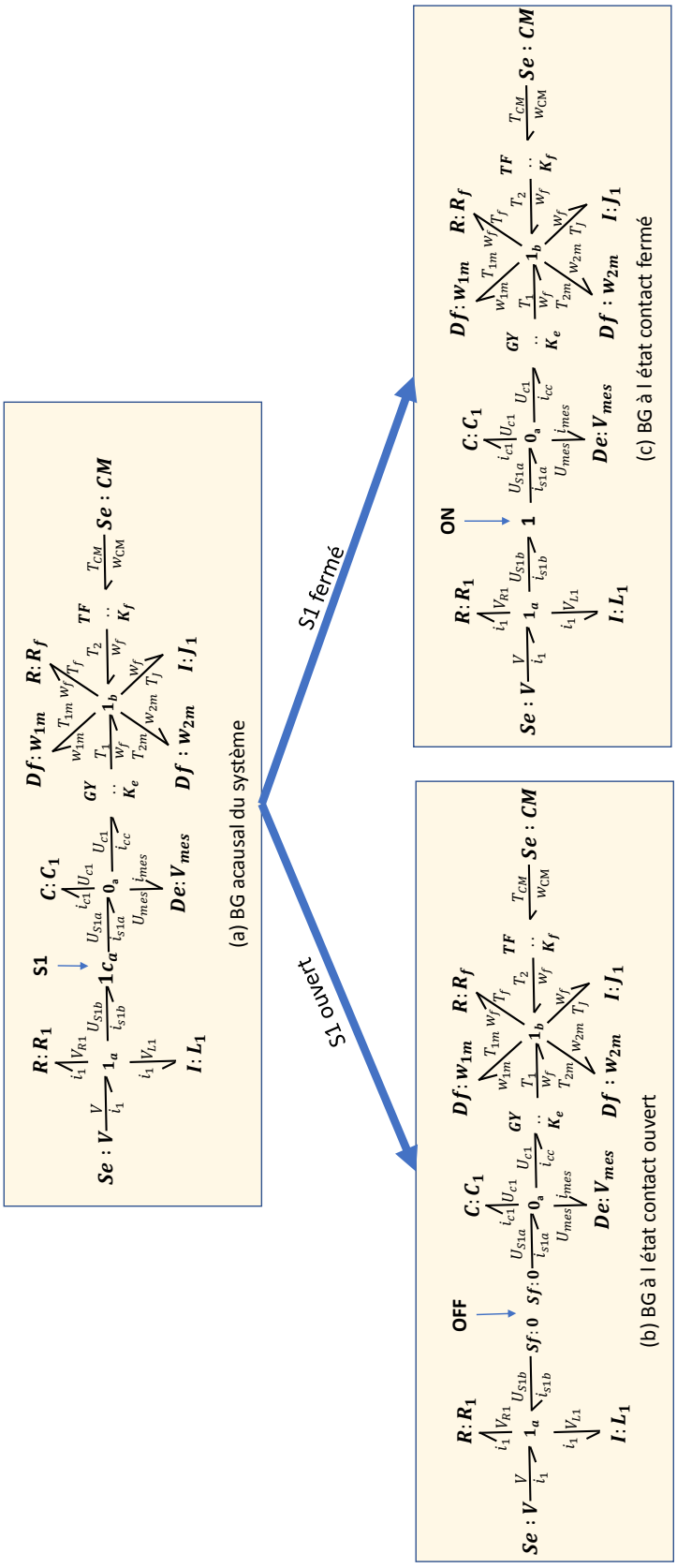


FIGURE 3.2 – BG selon les modes du système

3.3 Architecture logicielle générale

La Figure 3.3 détaille l'architecture générale du logiciel. Le logiciel se décompose en cinq modules qui seront successivement détaillés. Par rapport aux développements précédents (notamment ceux présentés dans le chapitre 2), l'accent est porté sur la difficulté à gérer les changements de causalités liées aux changements de modes.

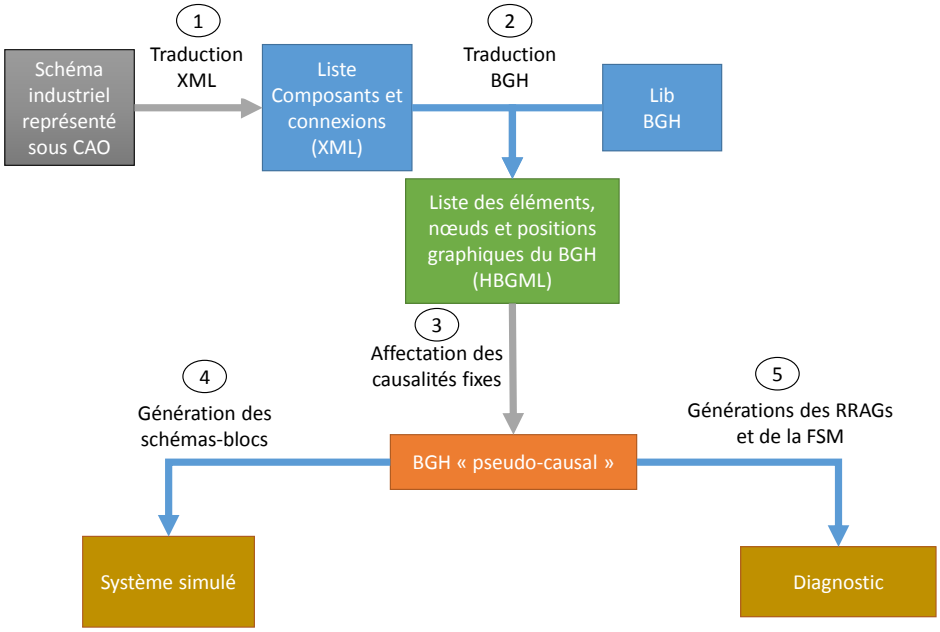


FIGURE 3.3 – Architecture générale logicielle

3.3.1 Module 1 : Conversion de CAO en XML

Le module de conversion de schémas industriels (électrique, pneumatique ou hydraulique) donné sous forme de CAO a été initialement développé dans le cadre du projet Simulatio®. Le résultat de cet outil est un fichier XML listant l'ensemble des composants avec les balises d'informations suivantes :

1. <nom> : le nom de l'instance du composant.
2. <composant> : le type de composant
3. <localisation> : la position graphique du composant sur le logiciel de CAO.
4. <gid> : un identifiant unique (ISO-9834, 2014).
5. <borne> : La liste des bornes du composant (appelé aussi port) contenant, pour chaque borne, un signal de connexion.

La figure 3.4 donne la description XML du composant «R» de appartenant au système mécatronique décrit sur la figure 3.1. Le composant se nomme « R », et son type est « Res » pour résistance. La localisation précise qu'il appartient à la voiture « M1 » représentée sur le premier folio, de la position 339,5 pixels à 340,5 pixels en abscisse et de 203,5 pixels à 204,5 pixels en ordonnée. Ce composant contient deux ports « p1 » et « p2 », respectivement liés aux signaux « (1) » et « (2) ».

```
<appareil>
    <nom>R</nom>
    <composant>Res</composant>
    <localisation>
      <subdivision>=M1</subdivision>
      <emplacement>/1</emplacement>
      <position>0-1a-0-6742!339,5!203,5!340,5!204,5!%</position>
    </localisation>
    <gid>0-1c-0-d67</gid>
    <liste_bornes>
          <nbbornes>2</nbbornes>
          <borne>
            <nom>p1</nom>
            <numero>1</numero>
            <emplacement>/1</emplacement>
            <line_sig>
              <nom>(1)</nom>
            </line_sig>
          </borne>
           <borne>
            <nom>p2</nom>
            <numero>1</numero>
            <emplacement>/1</emplacement>
            <line_sig>
              <nom>(2)</nom>
            </line_sig>
          </borne>
        </liste_bornes>
</appareil>
```

FIGURE 3.4 – Exemple du fichier XML correspondant au contact S1

3.3.2 Module 2 : Conversion en BGH

Le module 2 convertit le XML structurel précédemment exporté en un BG acausal. Les deux étapes pour construire ce BG sont les suivantes :

— Remplacer chaque élément du schéma structurel par son nœud BG associé.

— Connecter l'ensemble des nœuds entre eux.

La première étape est assez aisée ; pour la mener à bien, il convient de sélectionner dans une librairie («lib HBG»), contenant pour chaque type d'éléments, un nœud BGH. Par exemple, le composant «R» précédemment décrit est de type résistance. Une association est faite entre ce type de composant et le nœud R de la librairie BGH. La deuxième étape est plus complexe ; en effet, le fichier XML est structuré autour de nœuds, et non de connexions. Pour les systèmes électriques, pneumatiques et hydrauliques, on utilisera les règles de connexion suivantes (Broenink, 1999) :

1. Si un signal de connexion est associé à plus de deux bornes, alors les bornes sont connectées entre elles par une jonction de type 0.

2. Si un signal de connexion est associé à uniquement deux bornes, alors les bornes sont connectées entre elles par une jonction de type 1.

3. Si un commutateur («contact») est connecté à un ou plusieurs éléments en parallèle, c'est-à-dire que les éléments et le commutateur ont au moins un signal de connexion commun (hors signal de masse) même si l'on retire le commutateur, alors le commutateur est modélisé par une jonction $0c$. Sinon, le commutateur est modélisé par une jonction $1c$ (cf. figure 2.18).

4. Le BG obtenu peut alors être réduit en appliquant les deux règles suivantes : si deux jonctions identiques sont connectées entre elles, on peut supprimer une des deux jonctions, et connecter l'ensemble des liaisons sur la jonction restante. Si une jonction ne contient que deux ports, alors celle-ci peut être supprimée et ses liaisons fusionnées.

Si l'on reprend la partie électrique de l'exemple pédagogique (cf. figure 3.1), le signal de connexion « (4) » relie les composants « C », « Vmes » et « MCC » ; ces composants sont donc connectés grâce à une jonction 0 (suivant la règle 1). Le signal de connexion « (2) » relie uniquement deux éléments R et C, ces éléments sont donc connectés à l'aide d'une jonction 1 (suivant la règle 2). Il en est de même pour la source de tension et la résistance au travers du signal de connexion « (1) ». La règle 4 permet ensuite de réduire la modélisation en supprimant les jonctions 1 interconnectées. Enfin, le contact « S1 » est représenté par une jonction $1c$ puisque si on le retire , les deux ensembles ne sont plus connectés (application de la règle 3). La sauvegarde en mémoire des BGH générés nécessite une représentation informatique spécifique qui sera présentée plus loin dans ce chapitre.

3.3.3 Module 3 : HSCAP

La solution la plus simple d'affectation des causalités est celle qui consiste à les affecter, une fois les variables de contrôle de chaque jonction déterminées. Cependant, pour n jonctions contrôlées d'un BGH, chacune de ces jonctions possède deux états ouvert ou fermé. Ainsi, si l'on représente l'ensemble des états possibles, on obtient 2^n possibilités. Il est donc impossible de mettre en place cette méthode pour les systèmes complexes pour la simulation et le diagnostic temps réel. En effet, chaque utilisation SCAP consomme du temps et des ressources.

C'est pour cette raison qu'un algorithme d'affectation des causalités fixes (cf. définition 3.3.1), nommé Hybrid Sequential Causality Assignement Procedure (HSCAP), a été développé (cf. Définition 3.3.1). Celui-ci est une extension de l'algorithme 1 pour les BGH utilisant les jonctions contrôlées.

Définition 3.3.1 (Causalité fixe)**.** *Une causalité est fixe, si elle est inchangée pour l'ensemble des modes du système.*

Description de l'algorithme HSCAP

L'algorithme HSCAP (cf. algorithme 4) se décompose en trois étapes. Dans la première, les causalités imposées aux sources (Se, Sf) et aux détecteurs (De, Df) sont affectées (par définition, celles-ci sont fixes). Ces causalités sont ensuite propagées au travers des jonctions conventionnelles (TF, GY, $0j$, $1j$). Dans ce cas, ces causalités sont également fixes puisque directement propagées à partir de causalités, elles-mêmes fixes. Lors de la deuxième étape, une pile contenant les jonctions contrôlées est créée. Cette pile sera utilisée à l'étape suivante pour générer l'ensemble des cas possibles. Cependant, les jonctions à causalités forcées (cf. 3.3.2), sont retirées puisque, par définition, la causalité de la liaison déterminante est variable, et les autres causalités des liaisons sont fixes. Enfin, lors de la troisième étape, l'ensemble des modes possibles restants est testé, les causalités fixées étant déterminées par comparaison. Notons que, même si cette dernière étape s'avère consommatrice en termes de temps et de ressources, elle n'est pas incompatible avec nos contraintes, car elle est réalisée hors-ligne (c'est-à-dire avant l'utilisation des schémas-blocs pour la simulation et le diagnostic).

Définition 3.3.2 (Jonction contrôlée à causalités forcées)**.** *Une jonction contrôlée est à causalités forcées si l'ensemble de ses liaisons, à l'exception de la liaison déterminante, ont une causalité fixe. Soit n, le nombre de liaisons de la jonction contrôlée. Pour une*

jonction 1*c, cela signifie qu'il existe* $n-1$ *liaisons à causalité fixe qui imposent l'effort à la jonction pour tous les modes. Pour une jonction* 0*c, cela signifie qu'il existe* $n-1$ *liaisons à causalité fixe qui imposent le flux à la jonction pour tous les modes.*

L'algorithme présenté (cf. algorithme 4) s'inspire fortement des travaux publiés sur le logiciel MOTHS (Roychoudhury et al., 2007). Deux évolutions majeures ont été apportées. Tout d'abord, l'algorithme a été simplifié, par itération sur les éléments, à la place des liaisons. Par ailleurs, les causalités préférentielles (intégrales ou dérivées) ne sont pas imposées ; il existe donc moins de contraintes causales, donc de conflits de causalité, sur la modélisation. Cela permet d'obtenir des modèles de simulation et diagnostic optimisés.

Algorithme 4 Affectation des causalités fixes des BGH

▷ A : Assign imposed causalities
Push all BG's coventional junction nodes (TF,GY,1,0) in a stack J'

for all Source : S_i **do**
    Set imposed causality to the S_i port
end for
Propagate-causality-on-junction($\&J'$) [Algorithm 2]
for all Detector : D_i **do**
    Set imposed causality to the D_i port
end for
Propagate-causality-on-junction($\&J'$) [Algorithm 2]
▷ B : Remove forced controlled junctions (definition 3.3.2)
Push in a stack Jc : all controlled junctions
for all Controlled junction $jc \in Jc$ **do**
    if all non-DB port are assigned, and DB-port is connect to an R, C, or I element **then**
        remove jc from Jc (the DB port could not be in fixed causality)
    end if
end for
▷ C : Set other fixed causalities with brute force method
for all modes of element in Jc **do**
    Apply standard SCAP (algorithme 1)
    Save all bond causalities in a vector b_i
    Check if there is no causal conflict
end for
if a bond stays constant for all vectors b_i, set it as a fixed causality

Algorithme 5 Rappel Chapitre 2 : Algorithme de propagation de la causalité fixe sur jonction standard

```
1: function PROPAGATE-CAUSALITY-ON-JUNCTION(&J)
2:     repeat
3:         causality-has-changed = false
4:         for all junction j ∈ J do
5:             if j is a TF or a GY element then
6:                 if one port has its causality set then
7:                     apply causality to other ports
8:                     causality-has-changed = true
9:                     remove j from J
10:                end if
11:            else
12:                if port : DB (definition 4.7.1) has causality set then
13:                    apply causality to other ports
14:                    causality-has-changed = true
15:                    remove j from J
16:                else if all ports expect DB have causality set then
17:                    apply causality to DB
18:                    causality-has-changed = true
19:                    remove j from J
20:                else if all ports have causality set then
21:                    check if there is no causal conflicts
22:                    remove j from J
23:                end if
24:            end if
25:        end for
26:    until causality-has-changed == true
27: end function
```

Résultat

Cet algorithme a été appliqué sur l'exemple pédagogique. La figure 3.5 (a) montre le résultats de l'affectation puis de la propagation des causalités forcés. Suite à cette étape, la jonction contrôlée $1c_a$, qui n'est pas à causalités forcées, est ajoutée à une pile. Les causalités des deux modes du système (cf. figure 3.5 (b) et (c)) sont ensuite déterminées à l'aide de l'algorithme SCAP conventionnel. Dans ces deux modes, on remarque que d'autres nouvelles causalités affectées sont communes à l'ensembles deux modes, ces causalités sont donc fixes (cf. figure 3.5 (d).

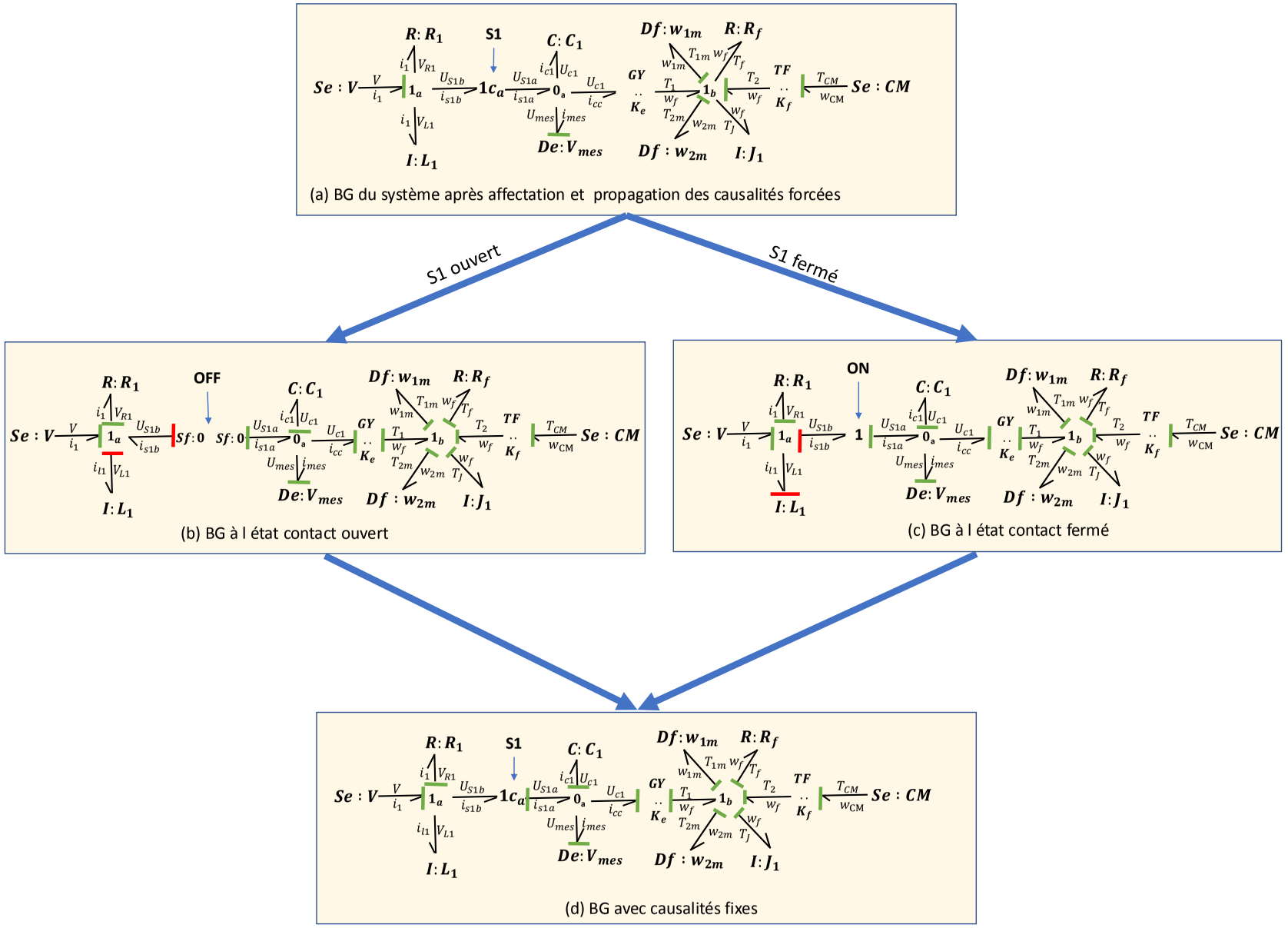


FIGURE 3.5 – Application des différentes étapes d'affectation des causalités fixes

3.3.4 Conflits causaux

Un conflit de causalité apparaît lorsque, pendant ou à la fin d'une procédure d'affection de causalité (HSCAP ou SCAP), des causalités affectées ne respectent pas les règles de causalité d'un nœud (par exemple deux efforts imposés à une jonction 0, deux flux imposés à une jonction 1, etc.). Il paraît difficile de proposer une solution globale pour l'ensemble des conflits causaux ; cela reste un verrou scientifique des BG et des BGH. Afin de résoudre ces conflits, une librairie métier devra ainsi être développée et enrichie à chaque nouveau problème. Un algorithme de résolution d'un conflit causal récurrent est proposé ci-dessous.

Un exemple de problématique : les commutations en série

La figure 3.6 (a) présente un schéma électrique contenant une source de tension en série avec trois éléments : une diode, un contact et une résistance. Un conflit causal apparaît pour les deux commutations lorsque celles-ci sont représentées par des jonctions contrôlées (figure 3.6 (b)). En effet, si les deux jonctions sont à l'état «OFF», le flux est imposé aux deux extrémités de la jonction qui les relient (figure 3.6 (c)). Ce problème peut être reformulé de la manière suivante : il est impossible de calculer les valeurs U_y et I_y, une fois que les deux jonctions sont ouvertes. En effet, comme les commutations sont idéales, le fil y se retrouve à l'état haute impédance (connecté ni à la masse ni à une source de tension) , il parait alors impossible de déterminer les valeurs de tension et de courant associées à ce nœud électrique.

Solution proposée

Ce fil n'étant relié à aucun actionneur ou capteur, la connaissance de la valeur du potentiel associé est inutile. Le conflit causal peut-être résolu par insertion d'une jonction 0_c entre les deux jonctions 1_c. Cette nouvelle jonction 0_c sera contrôlée par un signal, résultant du produit logique (fonction «ET») des commandes des deux commutateurs en série (figure 3.6 (c)). Cette solution revient ainsi à connecter le fil à la masse lorsque les deux éléments sont ouverts (état OFF). Cette solution est généralisée dans l'algorithme 6 ; celui-ci doit être exécuté avant la procédure de HSCAP.

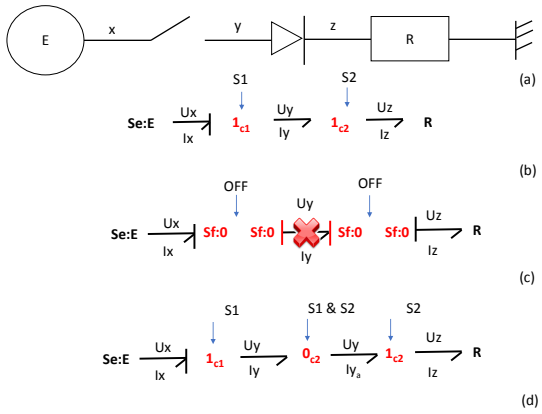


FIGURE 3.6 – (a) Schéma d'un circuit avec deux commutations en série
(b) Représentation BGH du système
(c) Conflit causal lorsque les deux éléments sont à l'état « OFF »
(d) Solution par insertion d'une jonction $0c$

Algorithme 6 Exemple d'un algorithme de résolution de conflits causaux

for all controlled junction : a **do**
  $x := a$
  for all port : $p \in x$ **do**
    $y :=$ connected element to p
    if $y \in R$: already crossed junction **then**
      do nothing
    else if $y \in \{C, I, R, MR, MC, MJ\}$ **then**
      do nothing
    else if $y == 1_c$ and $a == 0_c$ **then**
      do nothing
    else if $y == 0_c$ and $a == 1_c$ **then**
      do nothing
    else if $y == 1_c$ and $a == 1_c$ **then**
      insert a 0_c between a and y
    else if $y == 0_c$ and $a == 0_c$ **then**
      insert a 1_c between a and y
    else if $y == 0$ and $y == 1$ **then**
      $x := y$
      $R \leftarrow y$ Mark already passed element
    end if
  end for
end for

3.3.5 Module 4 : Génération de schémas-blocs pour la simulation

Après avoir affecté les causalités fixes, il est à présent possible de générer un schéma-bloc de simulation. Pour ce faire, l'algorithme 7 a été développé. Celui-ci se décompose en trois étapes majeures. La première consiste à exporter les parties fixes. Puisqu'un BG causal est convertible en un schéma-bloc (cf. théorème 2.1.1), les parties à causalités fixes peuvent donc être directement exportées en schémas-blocs à l'aide de la procédure conventionnelle. L'ensemble des éléments exportés à cette étape est retiré du BGH. Ensuite, à l'aide d'un parcours de graphe, on cherche à déterminer les différents nœuds connexes, pour créer différents sous-ensembles BGH indépendants. Enfin, ces sous-ensembles sont exportés en schémas-blocs pour chaque mode du sous-ensemble. Puis, des éléments de sélection (*switch element*), sont ajoutés à chaque sortie des schémas-blocs ; ceux-ci permettent de sélectionner le mode du sous-ensemble.

Une spécificité est à noter dans cette dernière exportation. Les blocs *intégrateurs* doivent être réinitialisés à chaque changement de modes, puisque leur état précédent n'est plus valide. Pour cela, on ajoute une borne de réinitialisation qui s'active à chaque changement de modes, ainsi qu'une borne de conditions initiales. Les conditions initiales du nouveau mode actif correspondent à l'état du système avant la commutation (Zainea et al., 2005).

Algorithme 7 Génération de schémas-blocs pour les BGH

Push all BG's nodes in a stack S
▷ A. Export elements with fixed causalities
Apply Fixed causality algorithm 4
for all Element $s \in S$ **do**
  if s has all of its connections in fixed causality **then**
    Export the corresponding block diagram and its fixed connections
    remove s from S
  end if
end for
▷ B. Create Subsystem HBG
while S is not empty **do**
  Create a new empty BG, $G(S, A, B)$
  where S are element, A the bond , and B a collection of boolean control signals
  choose first element $s \in S$
  Define S' a new empty stack
  S'.push(s)
  while S' is not empty **do**
    x= S'.pop()
    if $x \in S$ **then**
      Remove x from S
      Add x to G
      if x is a controlled junction **then**
        Add the boolean control signal b of controlled junction to G, if $b \notin G$
      end if
      for all node elements w connected to x, by a bond a **do**
        if $w \in S$ **then**
          S'.push(w)
          Add a to $G(S, A, B)$
        end if
      end for
    end if
  end while
  Add the HBG $G(S, A, B)$ to a list of HBG H
end while
▷ C. Export subsystem Block Diagram
for all $G'(S, A, B) \in H$ **do**
  for all set of boolean B' of B **do**
    Apply Standard causality algorithm (SCAP) (with intergral or derivative causality preference)
    for all Element $s' \in G'(S, A, B)$ **do**
      if s' is a integrator element (C, I) in integral causality **then**
        Export s' with a external reset, and initial condition
        Connect the reset to boolean control
        Export the input, out of the HBG (to connect to initial conditions of other subsystem)
        Add a to $G(S, A, B)$
      else if s' is a derivative element (C, I) in derivative causality **then**
        Export the output, out of the HBG (to connect to initial conditions of other subsystem integrator)
        Export the corresponding block diagram
      else
        Export the corresponding block diagram
      end if
    end for
  end for
  Add a switch at every output controlled by a boolean signal
  Add a delay before each boolean control signal (prevent causal loop on switching)
end for

Résultat

À partir de cet algorithme, nous pouvons ainsi générer le schéma-bloc du modèle présenté précédemment (cf. figure 3.7). Les parties à causalités fixes sont d'abord exportées ($Se : V$, $R : R_1$, $TF : K_f$, $GY : K_e$, $C : C_1$, $Se : CM$, $R : R_f$, $I : J_1$ et la jonction 1_b). Puis, un sous-ensemble contenant les éléments 1_a, $I : L_1$ et $1c_a$ est créé. Celui-ci est exporté en deux schémas-blocs, un pour le mode « ON », un pour le mode « OFF ». Les sorties de ces schémas-blocs sont reliées au reste du système à l'aide de blocs *switch* commandé par l'état contact $S1$. Enfin, la commande de $S1$ et la sortie du bloc dérivatif de l'élément L à l'état «OFF» sont respectivement la commande de réinitialisation et les conditions initiales de l'intégrateur à l'état «ON».

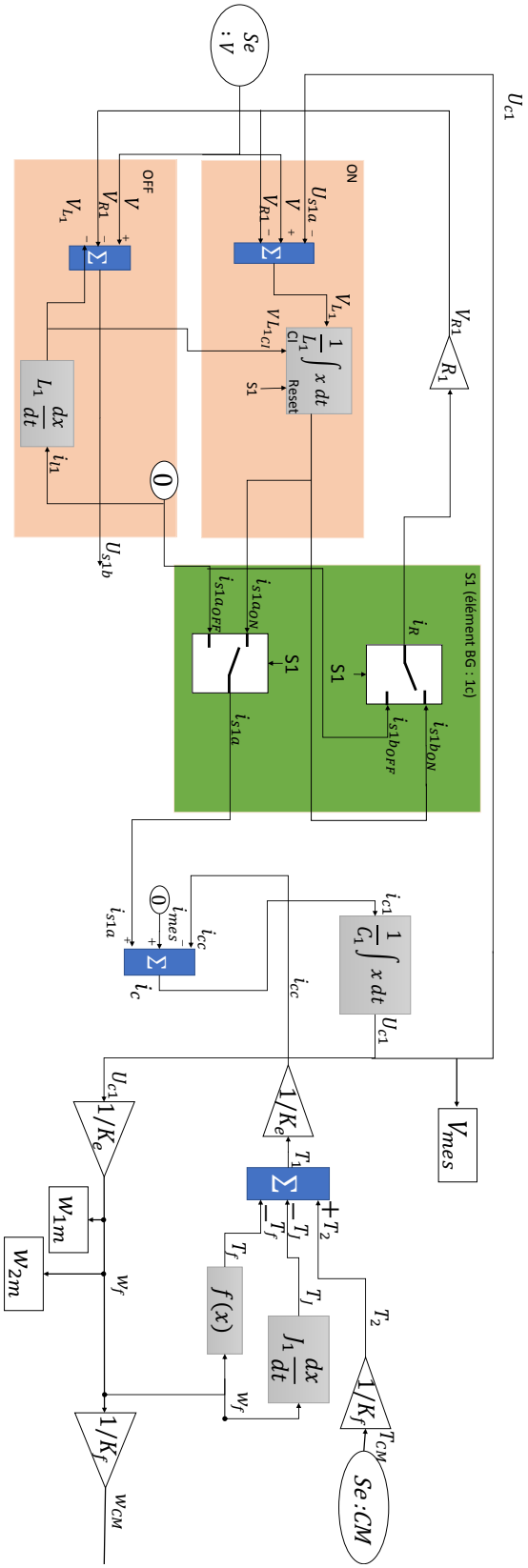


FIGURE 3.7 – schéma-bloc hybride exporté à partir du BGH avec causalités fixes affectées

Boucles algébriques et systèmes d'équations algébro-différentielles

Puisque les BG sont automatiquement convertis en schéma-blocs ; il est très fortement probable que les modèles générés soient de type Équations Différentielles Algébriques (EDA) et possèdent des boucles algébriques pour certains modes. Il faudra donc veiller à utiliser un simulateur de schémas-blocs permettant de traiter ce type de problématiques. Dymola par exemple réduit le système symboliquement, ce logiciel essaye ensuite de générer un nouveau modèle de type Équations Différentielles Ordinaires (EDO) sans boucles algébriques. Simulink utilise des solveurs robustes pour simuler les EDA et résoudre les boucles algébriques (Shampine, Reichelt et Kierzenka, 1999). Dans certains cas, le système n'est pas solvable par ces méthodes, l'ajout de blocs mémoires sur les boucles causales est alors conseillé (SIMULINK, 2018). Afin de réaliser cela on pourra s'aider des éléments marqués comme possibles boucles causales lors de l'exécution l'algorithme 4 (précisément ligne 21 à l'exécution de l'algorithme 1 - ligne 24).

3.3.6 Module 5 : Diagnostic

3.3.6.1 Génération des Relation de Redondance Analytique Globale (RRAG)

Dans le chapitre 2, les RRA étaient générées pour les BG conventionnels à l'aide de l'algorithme 3. Celui-ci exporte un BG dont les capteurs sont dualisés, ne contenant que les informations nécessaires à chaque RRA. Étant donné que les BGH représentent les systèmes avec l'ensemble de leurs modes, on réemploira la démarche précédente. La génération des RRAG se fait par le biais de l'algorithme 8 ; celui-ci contient uniquement deux évolutions par rapport à l'algorithme 3 des BG conventionnels. Premièrement, les causalités fixes sont affectées à l'aide de l'algorithme 4 (HSCAP). Deuxièmement, les schémas-blocs sont logiquement exportés à l'aide de l'algorithme 7.

Algorithme 8 Algorithme de génération des GARR

1: Create BG' a copy of BG
2: ▷ Export hardware redundancy relations
3: **for all** Element sensor s_i in BG' **do**
4: Get junction J_i connected to s_i
5: **for all** Element sensor s_j connected to J_i **do**
6: **if** $s_j \neq s_i$ **then**
7: Remove s_j from BG'
8: The difference of effort (or flow) mesured by s_j and s_i is the residual and should be zero in nominal situation
9: **end if**
10: **end for**
11: **end for**
12: ▷ Export other GARR
13: Dualize all sensors (Definition 2.2.1)
14: **for all** Element sensor s_i in BG' **do**
15: Define BG_{ARR_i} as a new empty bond graph
16: Define L a new empty stack
17: L.push(s_i)
18: **while** L is not empty **do**
19: x= L.pop()
20: **if** $x \notin BG_{ARR_i}$ **then**
21: Add x to BG_{ARR_i}
22: **if** x is a junction connected to a sensor $s_j \neq s_i$ **then**
23: Add s_j to BG_{ARR_i}
24: **else**
25: **for all** node elements w connected to x **do**
26: L.push(w)
27: **end for**
28: **end if**
29: **end if**
30: **end while**
31: Applied HSCAP to set fixed causality (Algorithme 4)
32: Applied Block diagram generation algorithm (algorithm 7) on bond graph BG_{GARR_i} to export block-diagram BD_{GARR_i}.
33: If causal conflict appears at a junction connected to a detector s_j, delete this detector and start again at line 15, if other causal conflict appears model should be revised.
34: **if** s_i is a De detector **then**
35: The flow at De is the residue and should be zero in nominal situation
36: **else if** s_i is a Df detector **then**
37: The effort at Df is the residue and should be zero in nominal situation
38: **end if**
39: **end for**

Résultat

À partir de cet algorithme, nous pouvons ainsi générer les trois RRAG du modèle présenté précédemment. La $RRAG_3$ est issu de la redondance matérielle entre les deux capteurs d'effort, soit : $RRAF_3 = Df : w_{1m} - Df : w_{2m}$.

Les $RRAG_1$ (figure 3.8 (a)) et $RRAG_2$ (figure 3.9 (a)) sont générées par dualisation des capteurs. Comme dans le cas de l'exemple sans contacteur, un conflit causal apparaît pour ces deux générations. Ce conflit est résolu par l'élimination du capteur à l'origine du conflit (partie hachurée). La génération est alors relancée sans ce capteur, et aboutit à un schéma-bloc (figure 3.9 (b) et figure 3.9 (b)) permettant l'estimation des résidus, à partir des entrées du système ($Se : CM$, $Se : V$) et de la commande du contacteur ($S1$). Enfin, notons que dans ce cas cette méthode a permis de ne pas devoir connaître les états initiaux du système, puisque l'ensemble des causalités des éléments C et I sont des causalités dérivées. Cela valide l'intérêt des algorithmes développés.

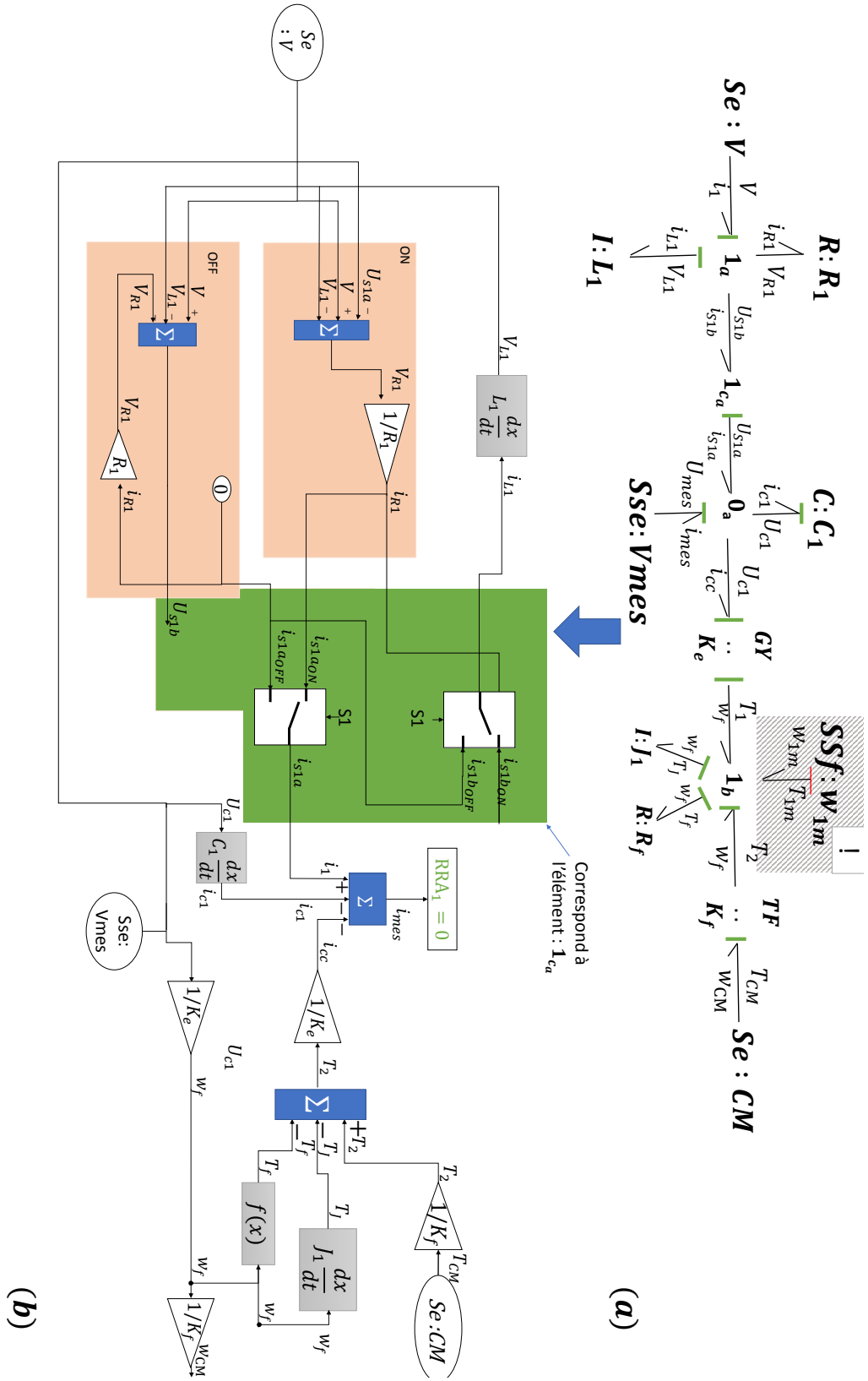


FIGURE 3.8 – (a.) BGH de génération de la RRAG1 (b.) schéma-bloc de la RRAG1

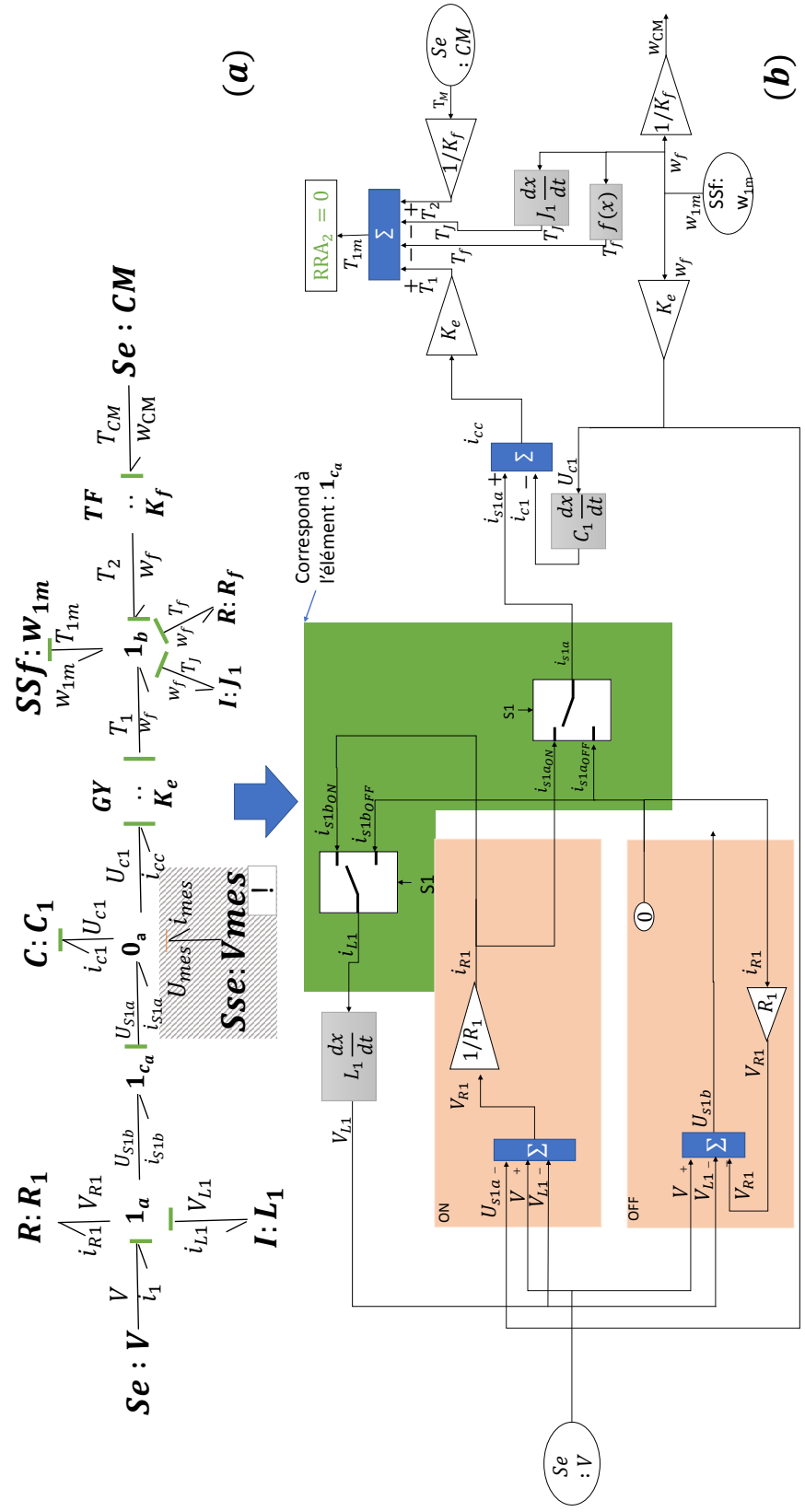


FIGURE 3.9 – (a.) BGH de génération de la RRAG2 (b.) schéma-bloc de la RRAG2

3.3.6.2 Génération de la FSM globale

Dans le chapitre 1, nous avons rappelé que deux choix sont envisageables pour la construction de la FSM d'un système hybride :

— Une matrice par mode : dans ce cas, il faut connaître l'état discret du système.

— Une matrice globale pour l'ensemble des modes : dans ce cas l'isolation et la détection des défauts nécessiteront davantage de résidus.

Toutefois, sans détection avancée, il est impossible de déterminer si un défaut est lié à la commutation du système, ou s'il impacte un mode particulier. On ne peut donc jamais connaître, suite à un défaut, l'état discret du système. Une matrice globale sera donc préférée pour l'ensemble des modes. Pour la générer, le théorème 2.2.2 des BG conventionnels a été généralisé aux BGH. Le résultat de cette génération est la FSM globale présenté sur le tableau 3.1. Celle-ci n'a que très peu évolué par ajout du contact. L'erreur du contact ($1c_a$) est bien détectable, mais non localisable.

Ib	0	0	0	0	0	0	0	0	0	0	1	1	1
Db	1	1	1	1	1	1	1	1	1	1	1	1	1
Fautes	$Se : V$	$Se : CM$	$R : R_1$	$1c_a$	$I : L_1$	$C : C_1$	$GY : K_e$	$TF : K_f$	$R : R_f$	$I : J_1$	$De : V_{mes}$	$Df : w_{1m}$	$Df : w_{2m}$
RRA1	1	1	1	1	1	1	1	1	1	1	0	1	0
RRA2	1	1	1	1	1	1	1	1	1	1	1	0	0
RRA3	0	0	0	0	0	0	0	0	0	0	0	1	1

Tableau 3.1 – FSM globale du système mécatronique hybride

3.4 Langages de description de Bond Graphs

À partir de l'architecture, on remarque que la description du BG est un élément particulièrement structurant du logiciel. Cette description stockée en mémoire devra contenir les données suivantes :

— les nœuds conventionnels (R, C, L, etc.) et hybrides ($1c$, $0c$).

— Les ports de chaque nœud et les règles de causalités associées.

— les liaisons avec leurs origines et leurs destinations.

— les causalités pour la simulation et le diagnostic (on devra pouvoir identifier les causalités fixes).

— les paramètres de l'élément représenté.

— les données permettant le positionnement de l'élément lors de la génération des graphes (abscisse, ordonnée, taille, image associée, etc.)

Ce langage devra aussi être hiérarchisé, afin de pouvoir regrouper les éléments en ensemble (pour représenter les composants complexes). Pour faciliter le développement des algorithmes, le clonage devra être intégré à ce langage (c'est-à-dire pouvoir immédiatement dupliquer une description). Afin de visualiser le BGH, des éléments graphiques seront inclus. La sérialisation (soit l'enregistrement automatisé) de l'état d'avancement permettra d'arrêter et de reprendre le processus à tout moment. Enfin dans un souci de simplicité, ce langage permettra non seulement de représenter les BGH, mais aussi les schémas-blocs. Afin de répondre à la totalité des besoins définis précédemment, il est préférable d'enrichir un langage éprouvé. Après consultation de la littérature existante, deux langages le Structured Interdisciplinary Description Of Physical Systems (SIDOPS) (une description est consultable en Annexe2) et le langage Bond Graph Modelling Language (BGML), semblent être de bons candidats. Dans ce mémoire, le choix s'est finalement porté sur l'amélioration du BGML, puisque celui-ci est basée sur un format informatique standard, le XML.

3.4.1 BGML et HBGML

BGML est un langage de balisage initialement conçu pour l'échange de modèles BG entre logiciels (Borutzky, 2006). Bien que ce soit son objectif premier, il peut parfaitement être utilisé comme langage de description principal d'un BG, puisqu'il contient l'ensemble des éléments nécessaires à sa représentation (causalité, nœuds,

liaisons, ports, etc.). En outre, la grammaire et le vocabulaire sont strictement décrits dans un fichier XML Schema Description (XSD), qui permet la vérification des BGML. Enfin, de nombreux outils permettent la lecture et l'écriture, voire la construction d'architectures informatiques, autour du XSD. Pour l'ensemble de ces raisons, le BGML a été choisi pour la construction de notre logiciel. La suite de cette section présentera ainsi les différents éléments constitutifs de ce langage, ainsi que les modifications et nouvelles fonctionnalités apportées (ces éléments seront marqués d'un astérisque «*» permettant de les identifier aisément). Enfin la dernière partie traitera des éléments ajoutés à ce langage pour la gestion des BGH ; nous proposons finalement de renommer ce langage HBGML.

3.4.1.1 Les différents éléments structurels

BGBD_root

Il s'agit de l'élément de plus haut niveau ; tout fichier BGML doit contenir une instance unique de celui-ci. Cet élément est totalement virtuel et formel ; son rôle est de simplifier la sérialisation des fichiers, en standardisant l'entrée. Cet élément possède un seul sous-élément, de type *BGBD_model* (figure 3.10).

BGBD_model

Le *BGBD_model* encapsule un BG complet, ou l'un des éléments unitaires des BG (C, I, R, etc.). Il est constitué des éléments suivants :

— *name* : le nom de l'élément.

— *id* : un identifiant unique. Celui-ci sera généré par le logiciel pour identifier l'élément.

— *info* : Les informations du modèle (description du bondgraph, auteur, révisions, date de création...).

— *position** : Les coordonnées graphiques de l'élément (x^* : abscisse, y^* : ordonnée, r^* : l'angle de rotation de l'image, s^* : le facteur d'échelle de l'image).

— *graphic* : l'image associée à l'élément dans un format vectoriel.

— au choix l'un des trois éléments suivants : un *composed-submodel*, c'est-à-dire un élément contenant un BG complet (nœuds et liaisons), un élément unitaire des BG nommé *BGBD_element* (C, I, R, etc.), ou un *path_to_BGBD**, un nouvel

élément, qui est un chemin vers un autre fichier BGML contenant la description de l'élément (*BGBD_root*). Cette dernière possibilité permet la création de librairies.

Par rapport à l'ancienne version du BGML, l'élément *model* a été supprimé. Celui-ci ne présente plus d'intérêt, étant donné que nous avons la maîtrise de la création logicielle ; il est donc possible d'ajouter de nouveaux types de modèles.

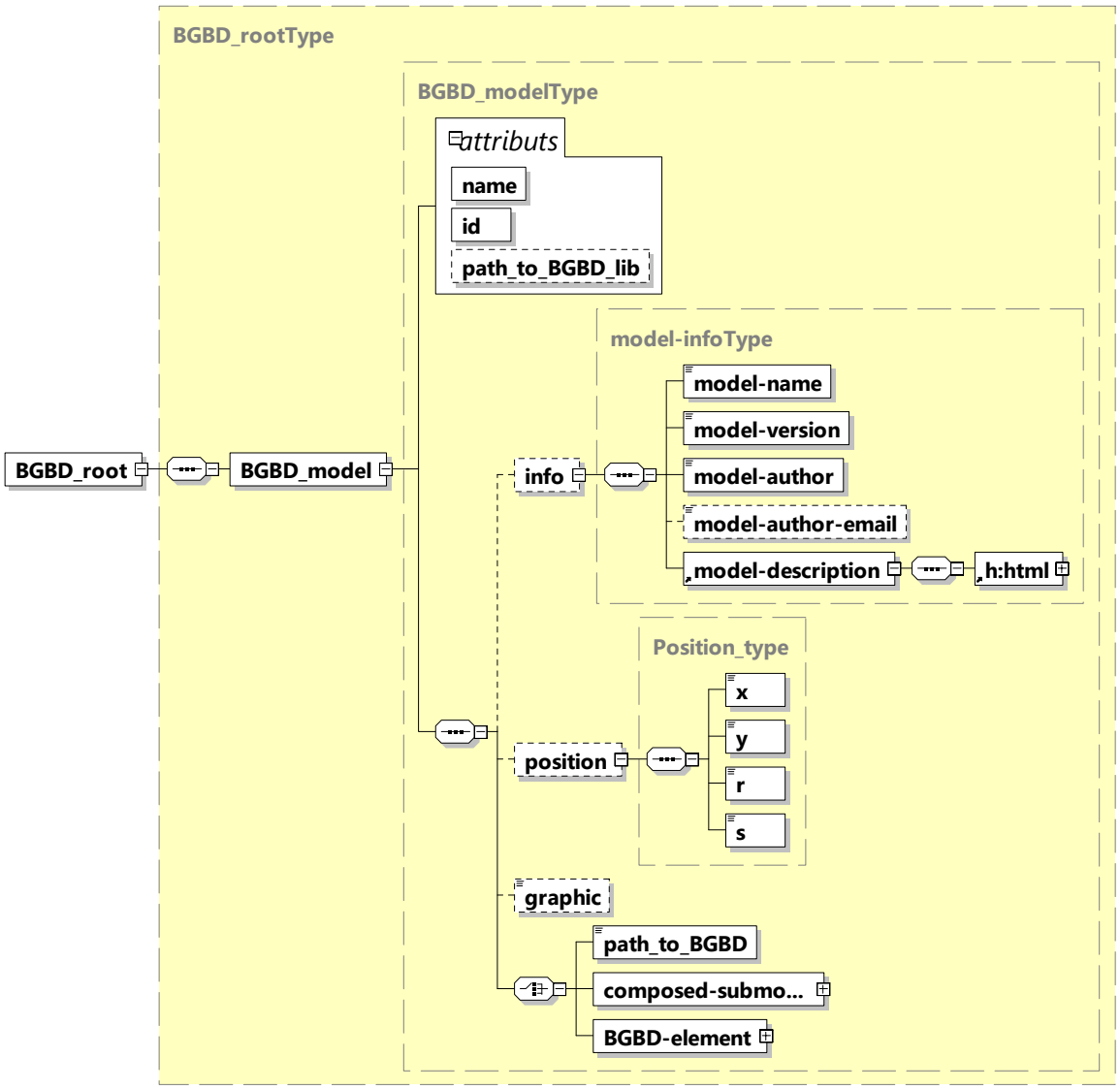


FIGURE 3.10 – Description de l'élément *BGBD_root*

BGBD_element Le *BGBD_element* est l'élément de plus bas niveau structurel (cf. figure 3.11). C'est au choix un élément bond graph (*BG_elem**) ou un élément schéma-bloc (*BD_elem**). Ces derniers étant abstraits, ils ne peuvent être utilisés directement. Pour les utiliser, différents « enfants » (*children*) ont été créés reprenant les éléments des BG conventionnels ($S = \{R\}\ U\ \{C\}\ U\ \{I\}\ U\ \{TF\}\ U\ \{GY\}\ U\ \{Se\}\ U\ \{Sf\}\ U\ \{De\}\ U\ \{Df\}\ U\ \{J\}$), ou des éléments standard de schéma-blocs (générateur de constante, addition, multiplication, etc.).

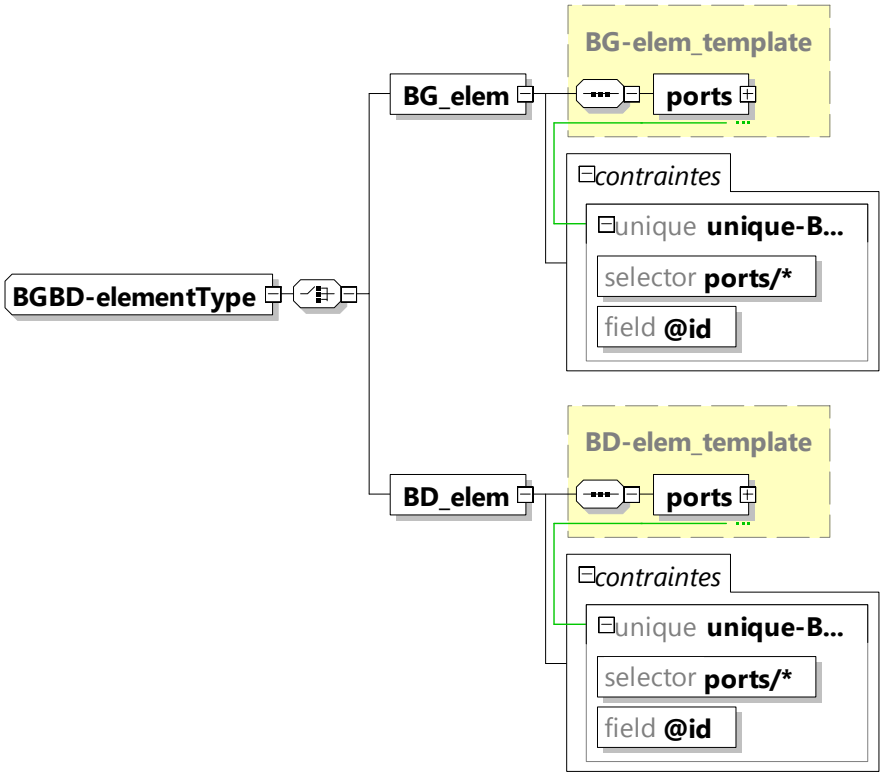


FIGURE 3.11 – Description de l'élément *BGBD_element*

La figure 3.12 présente un exemple d'élément BG, nommé *BG-store*. Celui-ci dérive directement de la classe *BG_elem**; il possède des ports par héritage. Un attribut *type* définit s'il s'agit d'une capacité ou d'une inductance. La valeur de K définit la conductance (C) ou l'inductance (L). Des conditions initiales peuvent être associées à ce nœud. Enfin, les éléments *int** et *deriv_1** sont des chemins de fichier, correspondant aux schémas-blocs associés respectivement aux BG intégral et dérivé.

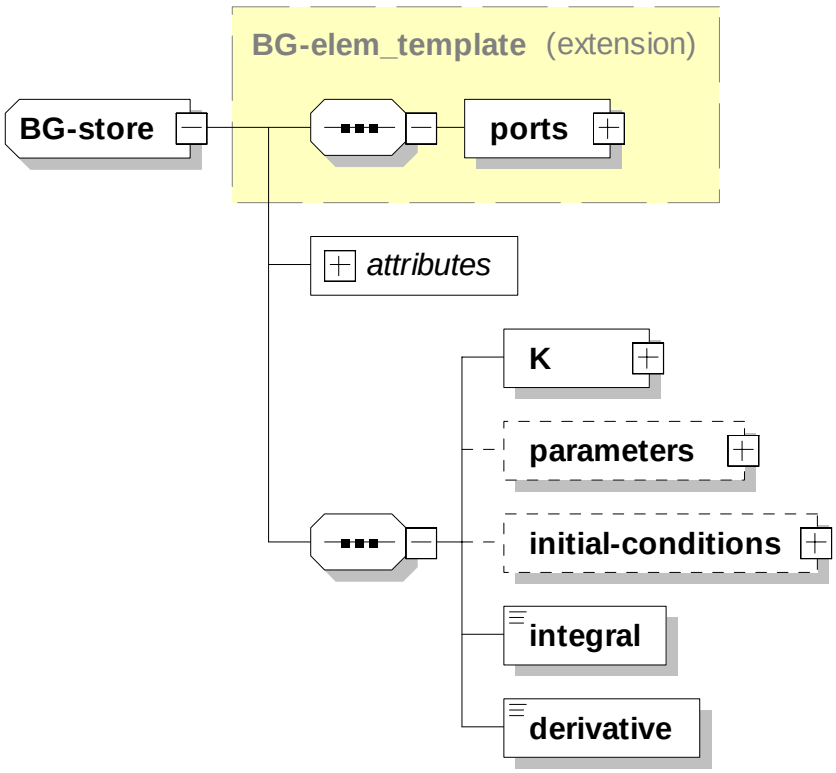


FIGURE 3.12 – Description de l'élément de stockage d'énergie

Le *BG-port** Pour la création de BG à mots hiérarchisés, il est nécessaire de transmettre la puissance entre deux BG au travers de ports. Pour réaliser cette fonction, l'élément *BG-port* a été créé ; celui-ci lie un port externe au BG (dans le cas du BGML un *composed_model*), avec l'intérieur du BG. La figure 3.13 présente un exemple d'utilisation. Dans celui-ci, les ports externes de deux BG (BG 1 et BG 2) sont reliés l'un à l'autre. Chaque *BG_port** est relié au port externe, à l'aide de l'attribut *ref_id** (cf. figure 3.14) qui donne le numéro de port associé. Les règles d'affection des causalités de ces deux éléments sont les suivantes :

— Si l'effort est imposé à un BG-port, alors le port lié se voit imposer le flux. Réciproquement, si le flux est imposé à un BG-port, alors l'autre port lié se voit imposer l'effort.

— Si l'effort est imposé à un port, alors le BG-port lié se voit imposer le flux. Réciproquement, si le flux est imposé à un port, alors l'autre BG-port lié se voit imposer l'effort.

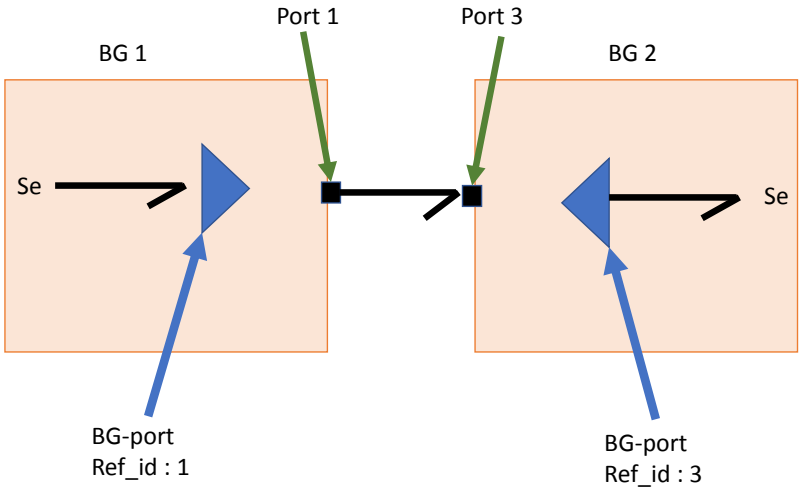


FIGURE 3.13 – Exemple d'utilisation d'un *BG-port*

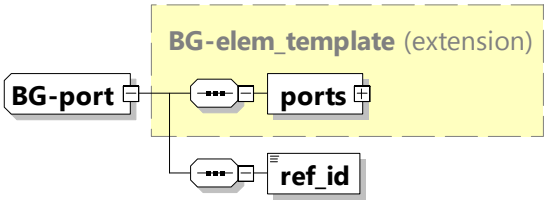


FIGURE 3.14 – XSD du *BG-port*

Il existe des *BD_port**, d'entrée et de sortie, de type logique et analogique.

Port Les ports sont les réceptacles des liaisons entre nœuds BG ou éléments de schémas-blocs. Dans le cadre de ce langage, cinq ports différents sont définis : un port de puissance pour les BG ; pour les schémas-blocs quatre ports, deux de signaux analogiques d'entrée ou de sortie et deux de signaux logiques d'entrée ou de sortie. Tous ces éléments héritent d'une classe abstraite commune nommée *portType*. Le *power_port* utilisé pour les transfert de puissance est constitué des balises suivantes (cf. figure 3.15) :

— *name* : le nom du port.

— *id* : un nouvel attribut id unique pour identifier le port.

— *dimension* : La dimension du port.

— *orientation* : donne le sens préférentiel de transfert de puissance.

— *domain* : le domaine de puissance au choix : électrique, hydraulique, pneumatique, mécanique de rotation et de translation.

— *flow* : le nom associé au flux.

— *effort* : le nom associé à l'effort.

— *causality* : les règles de causalité associées à l'élément pour la simulation.

— *causality_diag** : un nouvel attribut définissant les règles de causalité associées à l'élément pour le diagnostic (par exemple, une causalité dérivée préférentielle pour les éléments C et I).

L'attribut *causality* donne les règles de causalité à appliquer pour chaque port ; cet attribut est choisi parmi plusieurs chaînes de caractères (*restricted string**) (on pourra se reporter à l'Annexe 1 pour rappel des règles de causalité) :

— *fixed effort out, fixed flow out* : pour les éléments Se, Sf, De, Df.

— *preferred effort out, preferred flow out* : pour les éléments C, I.

— *complex TF, and complex GY* : pour les éléments TF, GY .

— *complex 0j*, complex 1j** : pour les jonctions $1j$, $0j$.

— *indifferent* : pour l'élément R.

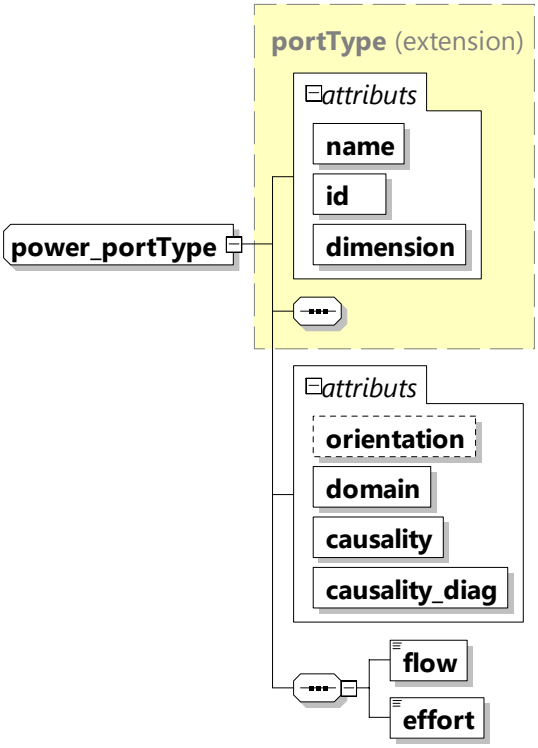


FIGURE 3.15 – Description de l'élément port

Connexions

Un BG est constitué de nœuds et de liaisons, auxquels correspond la partie connexion. Puisque les schémas-blocs et BG sont représentés au sein d'un même langage, l'élément *connections* contiendra trois types de connexions : connexion logique (*line_logic*), connexion analogique (*line*), et connexion de puissance (*bond*). L'élément *bond* est constitué des données suivantes (cf. figure 3.16) :

— *name* : le nom de la liaison.

— *id* : l'id de la liaison.

— *dimension* : la dimension de la liaison.

— *flow* : le nom associé aux flux.

— *effort* : le nom associé à l'effort.

— *coord* : les *id* des deux éléments connectés (l'id du port et du composant).

— *graphic_lines** : une collection de points par lesquels le tracé graphique de la liaison passera.

— *causality** : un booléen ; si « vrai », l'effort est imposé au port *to* ; si « faux », le flux est imposé au port *to*.

— *causality_diag** : un booléen au fonctionnement similaire à l'attribut *causality* ; celui-ci sera utilisé pour la génération de RRAG.

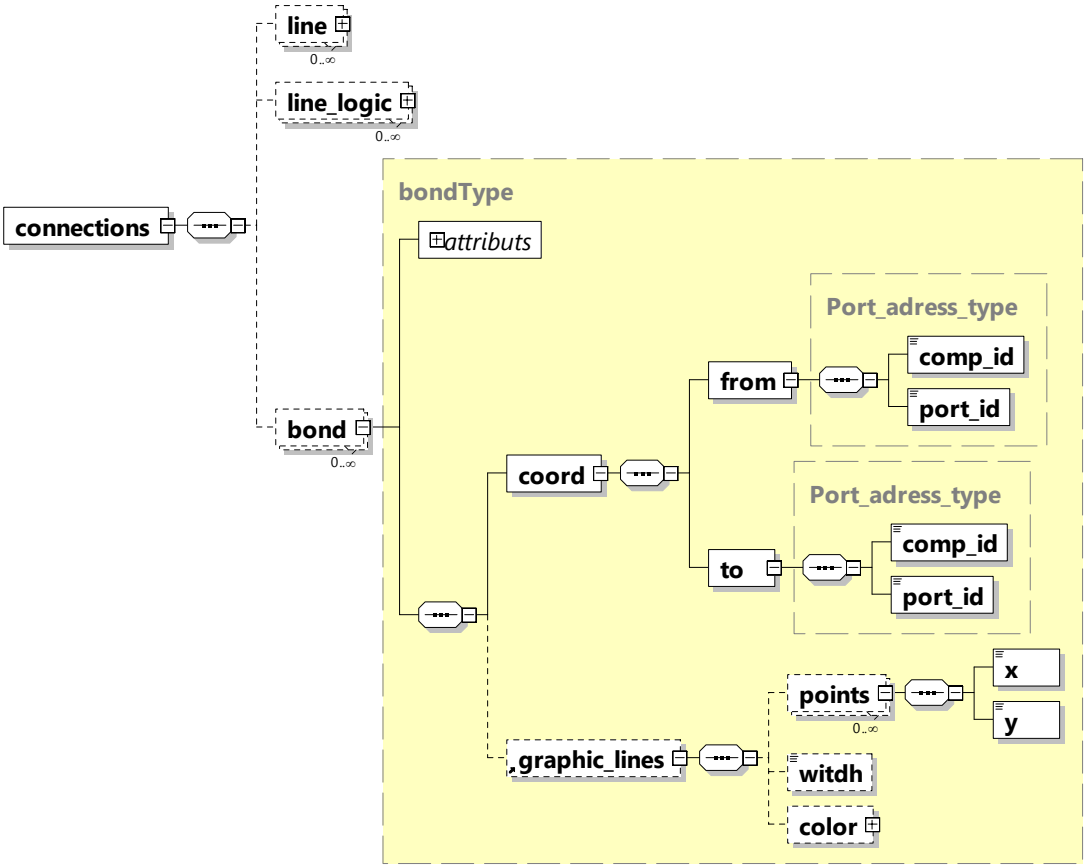


FIGURE 3.16 – Description de l'élément *connections*

3.4.1.2 Représentation des éléments hybrides

Afin de représenter les BGH, d'autres éléments sont ajoutés. Pour pouvoir distinguer l'apport de ces nouveaux éléments, ce langage sera renommé HBGML.

Jonction Contrôlée

*BGH-controlled_junction** est un nouvel élément qui représente la jonction contrôlée (cf Chapitre2). Cet élément, à l'identique des autres éléments conventionnels, hérite directement de l'élément *BG_elem** (cf figure 3.17). Pour rappel la causalité des ports des jonctions contrôlées est particulière ; à l'état ON, celle-ci est identique aux jonctions conventionnelles, à l'état OFF, un effort (ou flux) nul est imposé à l'ensemble des ports. Deux nouveaux attributs ont été créés pour représenter ces règles, nommés *complex 0jc** et *complex 1jc**.

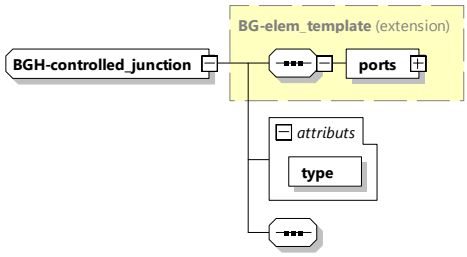


FIGURE 3.17 – Description de l'élément *BGH-controlled_junction**

Liaisons hybrides

Une liaison hybride (cf. définition 3.4.1), héritant directement de la liaison standard, a été développée. Cette liaison, nommée *bond_hybrid*, permet la mise en évidence graphique des liaisons modifiables sur changement de mode (dans notre cas, ces liaisons seront affichées en gris).

Définition 3.4.1 (Liaison hybride)**.** *Une liaison est hybride, si au moins une de ses extrémités est connectée à un élément hybride (jonction contrôlée, jonction à puissance commutée, etc.).*

Causalité fixe

Pour le développement de futurs algorithmes, il est intéressant de pouvoir identifier les causalités fixes (cf. définition 3.3.1). Pour cela, un attribut, nommé *fixed_causality**, permet d'identifier ces liaisons.

3.4.2 Synthèse

Dans cette sous-section, une évolution du langage BGML, pour permettre description complète des BGH, a été développée. Ce langage contient maintenant l'ensemble des éléments nécessaires au développement d'un logiciel

3.5 Architecture informatique détaillée

La production du logiciel s'effectue avec les trois exigences suivantes modularité, réutilisabilité et simplicité des codes produits. Dans ce cadre, le choix de la Programmation Orientée Objet (POO) paraît assez évident. De plus, ce logiciel possède des contraintes d'industrialisations sur le plan de la qualité, et des temps de réponse. Le C++ a donc été choisi pour répondre à ces contraintes. Cette section décrit ensuite les choix de plateforme informatiques et le choix patrons de conception utilisés pour le la mise au point du logiciel, avec l'objectif de réduire au maximum les temps de conception.

3.5.1 Le choix de la plateforme Qt

Qt est une interface orientée objet pour le C++. Qt propose des composants d'interface graphique (*widgets*), d'accès aux données, de connexion réseau, de gestion des files d'exécution, d'analyse XML, etc. Qt permet la portabilité des applications (Windows, Linux) par simple recompilation du code source. En raison de tous ces avantages, le développement du logiciel se fera sur cette plateforme.

Le framework : Graphics View

Le framework (*Graphics View*) est un canevas permettant d'interagir avec un grand nombre d'éléments graphiques 2D personnalisés, ainsi qu'un widget d'affichage (intégrant zoom et rotation), pour visualiser ces éléments. Il inclut une architecture de propagation d'événements qui autorise des interactions précises (clics, mouvements, déplacements, etc.) avec les éléments de la scène. *Graphics View* utilise un arbre BSP (Binary Space Partitioning) pour trouver très rapidement les éléments en conséquence, il peut afficher de très grandes scènes en temps réel, ce même avec des millions d'éléments. Ce framework d'affichage est donc parfaitement adapté aux besoins de visualisation des BGH.

Pour utiliser ce framework, trois composants principaux existent :

1. Les objets, *QGraphicsItem* sont les différents éléments qui se placent sur une scène. Dans la modélisation BG, ils représenteront l'affichage graphique de chaque composant unitaire (C, L, R, ...) ou des connexions entre ces éléments (liaisons, signaux).
2. La scène, *QGraphicsScene* est le canevas contenant l'ensemble des objets. Une scène, pour être visible, doit être attachée à une vue. Celle-ci correspond à

l'affichage d'un BG (*composed_model* pour les HBGML).

3. La vue, *QGraphicsView*, est un moniteur d'affichage de scènes. Elle affiche la scène, et transmet tous les événements (souris, clavier, etc.) vers leur destination. Cet élément est une interface entre l'utilisateur et la scène.

3.5.2 Gestion des XML : Codesynthesis

CodeSynthesis XSD est une plateforme open source d'utilisation des XML. Pour un XSD, cette application génère un code C++ standard, qui représente la structure du XML, mais aussi, les méthodes associées de lecture/écriture et de sérialisation du XML. Un objet structuré sera associé à chaque élément ou attribut défini dans le HBGML. Une fois lu, l'ensemble de la structure est stocké en mémoire. Enfin, des paramètres de customisation permettent à chacune des classes générées d'hériter d'autres classes. Cette notion sera utile, puisque les objets hériteront ainsi de *widgets* Qt, pour affichage ; des méthodes de parcours du BGH seront insérées (Codesynthesis, 2014).

3.5.3 Architecture orientée objet

Patron de conception général

La structure de l'architecture est fortement inspirée des patrons de conception (*Design Pattern*) Modèle-Vue-Contrôleur (MVC) (Lasater, 2010). Cependant, afin d'utiliser les fonctionnalités de personnalisation de CodeSynthesis, il a été décidé de regrouper le modèle et la vue. Le modèle hérite ainsi de la vue, ces deux éléments étant systématiquement mis à jour l'un et l'autre pendant les évolutions du logiciel. La figure 3.18 présente l'architecture de haut niveau de l'application sous le format d'un digramme de classe suivant la norme Unified Modeling Language (UML) (Fowler, 2003). Une fenêtre principale nommée *MAIN_WINDOWS* possède l'instance de plus haut niveau de *composed_model*. Ce dernier contient le BGH en mémoire, ainsi que les éléments graphiques nécessaires à son affichage et à sa modification manuelle (la création d'un nouvel ensemble se fait par modification d'un *composed_model* vide). Le contrôleur est appelé par la fenêtre principale au travers de menus ; ce contrôleur contient les différents algorithmes d'exportation des schémas-blocs de utilisés pour représenter les RRAG, etc. Pour cela, les différents algorithmes du contrôleur utilisent le BGH stocké dans l'élément de haut niveau *composed_model*.

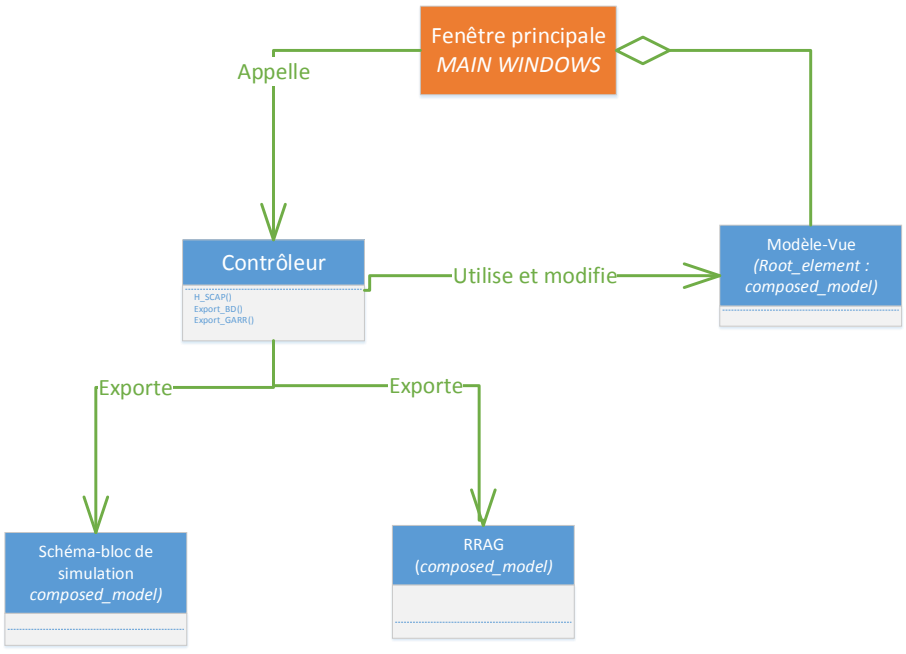


Figure 3.18 – Patron de conception général de l'application (Diagramme de classe UML)

Gestion et Affichage du modèle BG

Dans le cadre de ce développement CodeSyntesis génère un code C++ la lecture et l'écriture des fichiers HBGML. De plus, afin de faciliter la maintenance et le développement de l'outil, les objets générés héritent d'éléments graphiques du framework *Graphics View*. Cette architecture de développement s'inspire fortement du patron de conception : *Abstract Factory* (Lasater, 2010). La figure 3.19 représente le diagramme de classe de l'application (suivant la norme UML).

Les éléments *QGraphicsScene*, *QGraphicsItem* et *QGraphicsItemGroup* sont les éléments de base de la librairie graphique *Graphics View*. De ces éléments héritent respectivement les classes *QGraphicsScene_BGBD*, *QGraphicsItemGroup_BGBD*, et *QGraphicsPathItem_cnx* ; ces dernières ajoutent des méthodes, attributs et événements pour permettre de faire le lien entre l'affichage et le HBGML. Les classes générées par CodeSynthesis héritent de ces objets.

Composed_model est un élément contenant un BGH. Il hérite de *QGraphicsScene_BGBD* afin de récupérer les fonctions et attributs nécessaires à mise en place du canevas (c'est à dire la visualisation graphique du BGH) ; il possède une listes de liens (*connexions*) et un ou plusieurs éléments BGH (*BGBD_model*). L'élément *BGBD_model* hérite d'un objet graphique (*widget*) nommé *QGraphicsItemGroup_BGBD*. La classe *BGBD_model* correspond soit à un *composed_model* soit à un *BGBD_element*, cette spécification se faisant par agrégation. *BGBD_element* correspond soit à un élément BGH (*BG_element**), soit à un élément schéma-bloc (*BD_element**), également par agrégation,

Les classes (*BG_element**) et (*BD_element**) sont des classes abstraites. Pour les utiliser, les éléments BGH (C, I, Se , etc.) ou schémas-blocs (addition, soustraction, etc.) hériteront ainsi de ceux-ci. Ces éléments hérités possèdent des ports de connexion pour les liaisons.

Connections contient des liens de différents types : liaisons hybrides de BGH (*bond_hybridType*), liaison conventionnelle de BGH (*bondType*), liens analogiques de schéma-bloc (*lineType*), ou liens logiques de schéma-bloc (*logic_lineType*). L'ensemble de ces connexions est directement ou indirectement dérivé de *QGraphicsPathItem_cnx*, une ligne graphique dessinée à partir d'un ensemble de points.

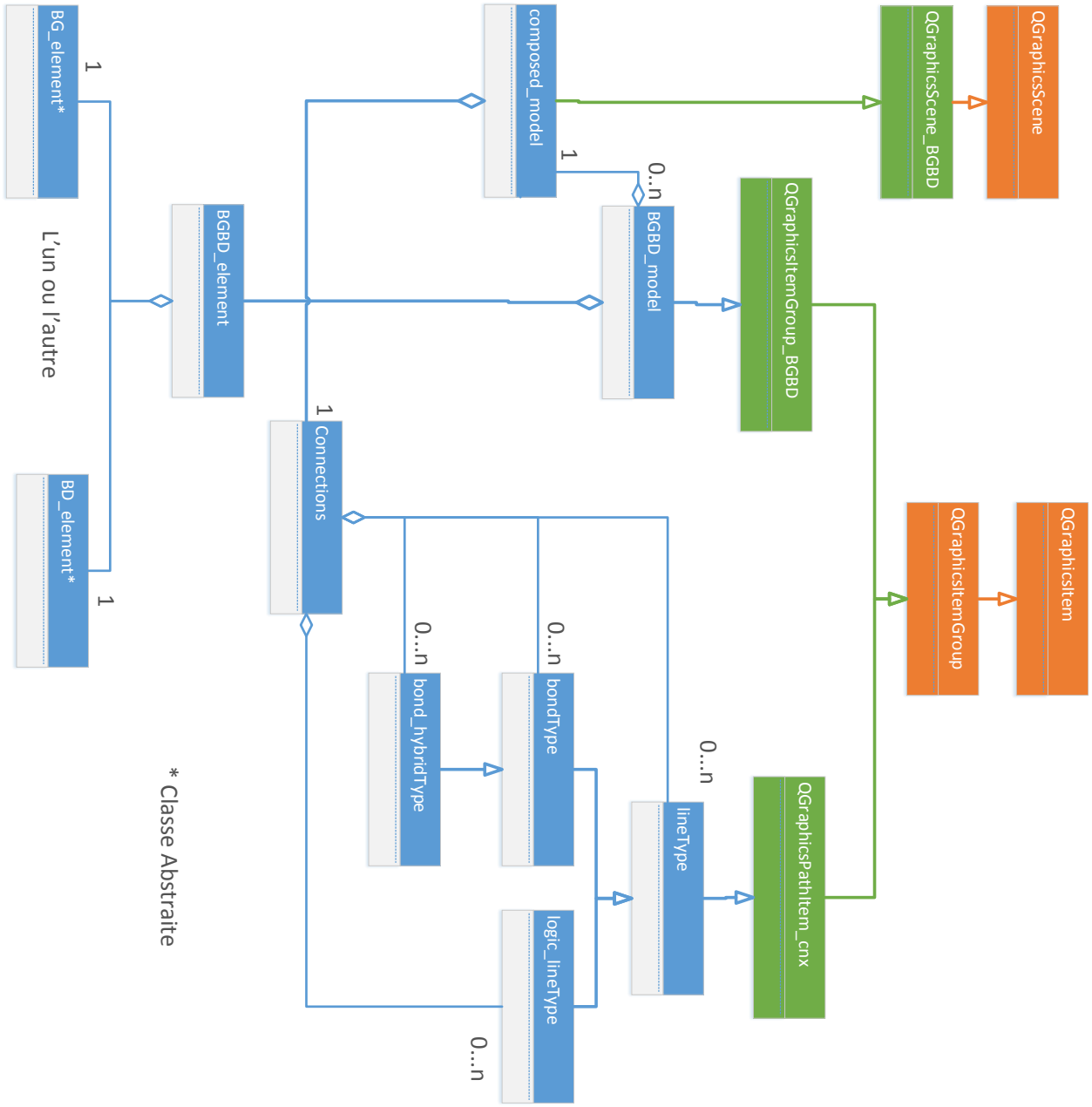


FIGURE 3.19 – Diagramme de classe du modèle est de la vue

3.6 Conclusion

Dans ce chapitre, une architecture informatique détaillée d'un logiciel de simulation et de diagnostic des SDH utilisant la modélisation BGH a été présentée. Partant des schémas industriels représentant le système étudié, différents algorithmes sont proposés pour générer automatiquement les schémas-blocs utilisés pour la simulation et le diagnostic du système. De nouvelles fonctionnalités ont été ajoutées au langage BGML, pour permettre la description des BGH et l'utilisation de ce langage, non plus pour l'échange de données, mais comme structure d'un logiciel. Enfin, différents patrons de conception ont été choisis ; ceux-ci ont été modifiés pour respecter au mieux notre besoin.

A la suite de ce chapitre, un prototype logiciel, répondant à notre problématique de supervision des SDH complexes contenant de très nombreux modes a donc été construit.

Celui-ci devra être testé sur système réel, avant son industrialisation. Cette étape d'industrialisation comportera l'écriture de l'ensemble des éléments de codes, la refonte des IHM pour une plus grande fluidité, la certification des méthodes obtenues, et la création d'un simulateur de schémas-blocs indépendants.

Par ailleurs, la position graphique des éléments provenant de la CAO n'est pas toujours exploitable, on pourra s'inspirer de logiciels tels que graphiz afin résoudre cette contrainte technique (Ellson et al., 2003).

D'un point de vue plus académique, on peut proposer les perspectives d'amélioration suivantes :

— Poursuivre le développement d'algorithmes de résolution des conflits causaux.

— Optimiser le BGH, par l'écriture de nouvelle règles sur la réduction de jonctions utilisée dans la modélisation d'un SDH (Broenink, 1999).

— Trouver des règles autres que celle de la suppression des jonctions à causalités forcées pour affectation des causalités fixes. Cela permettrait de réduire, le plus possible, l'utilisation de la force brute pour l'affectation des causalités fixes, consommatrice de temps.

— Travailler sur le filtrage des bruits de mesure et sur les incertitudes paramétriques.

Bibliographie du présent chapitre

Broenink, J. F. (1999). *Introduction to Physical Systems Modelling with Bond Graphs*. SiE Whitebook on Simulation Methodologies, p. 1-31.

ISO-9834 (2014). *ISO/IEC 9834-8 :2014*.

Roychoudhury, I. et al. (2007). *A Method for Efficient Simulation of Hybrid Bond Graphs*. International Conference on Bond Graph Modeling (Icbgm), p. 177-184.

Zainea, M. et al. (2005). *Automatic Simulink Model Building for Physical Switching Systems*.

Shampine, L. et al. (1999). *Solving Index-1 DAEs in MATLAB and Simulink*. SIAM Review 41.3, p. 538-552.

SIMULINK (2018). *Algebraic Loops - MATLAB & Simulink - MathWorks France*. url : `https://fr.mathworks.com/help/simulink/ug/algebraic-loops.html#bsjdo3y-1` (visité le 23/04/2018).

Borutzky, W. (2006). *BGML – a Novel XML Format for the Exchange and the Reuse of Bond Graph Models of Engineering Systems*. Simulation Modelling Practice and Theory 14.7, p. 787-808.

Codesynthesis (2014). *Tree/Customization Guide - Code Synthesis Wiki*. url : `http://wiki.codesynthesis.com/Tree/Customization_guide`.

Lasater, C. G. (2010). *Design Patterns*. Wordware Publishing, Inc. 306 p.

Fowler, M. (2003). *UML Distilled : A Brief Guide to the Standard Object Modeling Language*. 3 edition. Boston : Addison-Wesley Professional. 208 p.

Ellson, J. et al. (2003). *Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools*. Graph Drawing Software. Springer-Verlag, p. 127-148.

Chapitre 4

Application industrielle

Ce chapitre a pour objet de présenter les résultats obtenus, sur un système réelle par application des méthodes et algorithmes précédemment développés dans ce mémoire. L'application, constituée d'un système électro-pneumatique ferroviaire complet, est décrite dans une première partie. Après une description du fonctionnement du freinage et des éléments le constituant, un BGH à mots sera développé à partir de la CAO ; il donne la structure générale du système. Ensuite, une description complète des composants et de leur modélisation BGH sera réalisée. Le BGH à mots sera étendu sous une forme complète puis implémenté dans le prototype logiciel sous la forme d'un modèle HBGML. Les différents algorithmes de génération des schémas-blocs seront ensuite exécutés ; les schémas-blocs seront simulés pour validation, en incluant certains défauts discrets et paramétriques.

L'application est un système électropneumatique ferroviaire (très légèrement modifié pour des raisons de confidentialité).

4.1 Description générale du système

4.1.1 Alimentation en énergie

Quatre sources d'énergie différentes sont utilisées, pour alimenter et contrôler les équipements du train (Figure 4.1). La haute tension électrique (HT) distribue 750 volts aux moteurs et ConVertisseurs Statiques (CVS). D'un point de vue BGH, cela correspond à un effort (tension) commun représenté par une jonction 0. Le CVS convertit la HT en moyenne tension (MT) afin d'alimenter les compresseurs pneumatiques et les systèmes de climatisation. Le CVS produit également une basse tension (BT) utilisée pour l'alimentation des composants électroniques, et le contrôle d'éléments discrets (relais électromécaniques, transistors, etc.). L'énergie pneumatique est produite par des compresseurs. Ce flux est transmis à un sécheur qui retire l'humidité de l'air. Une fois déshumidifié, ce flux est transmis aux différentes voitures par un conduit principal. Cette énergie pneumatique alimente dans chaque voiture les équipements suivants : portes, freins, et suspensions. Un manomètre vérifie la pression du conduit principal. Les réservoirs de pression (90 L sur la figure 4.2) agissent sur le système comme des filtres de premier ordre. Les connecteurs intervoitures pneumatiques et électriques peuvent être assimilés à des résistances négligeables par rapport au système complet.

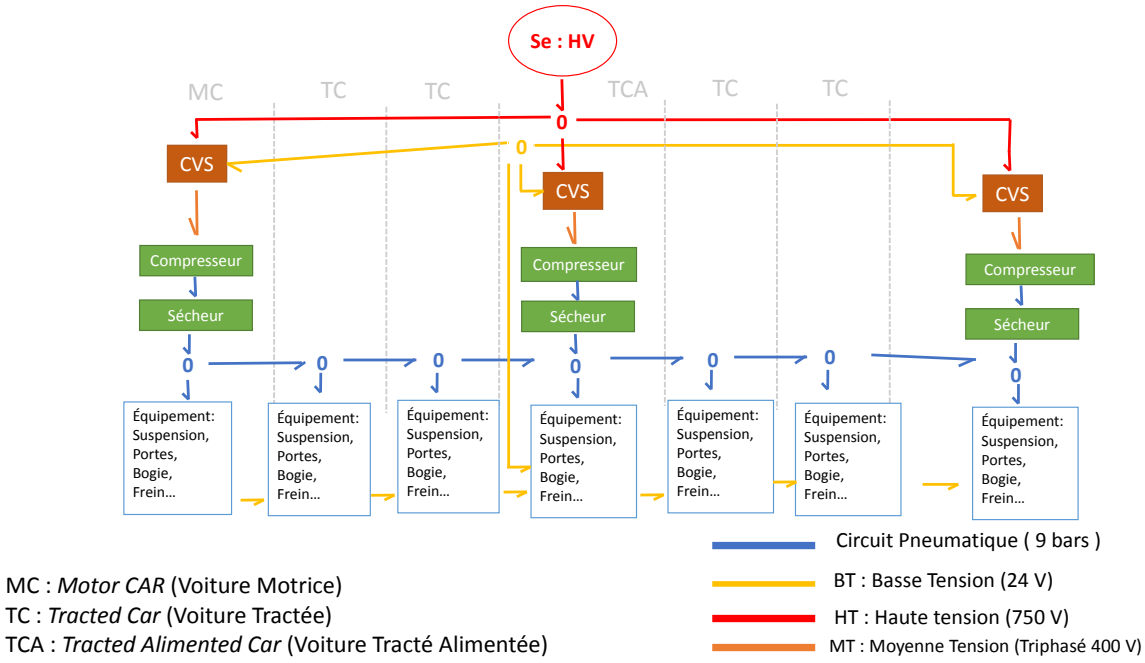


FIGURE 4.1 – Bond graph d'un circuit complet électro-pneumatique ferroviaire

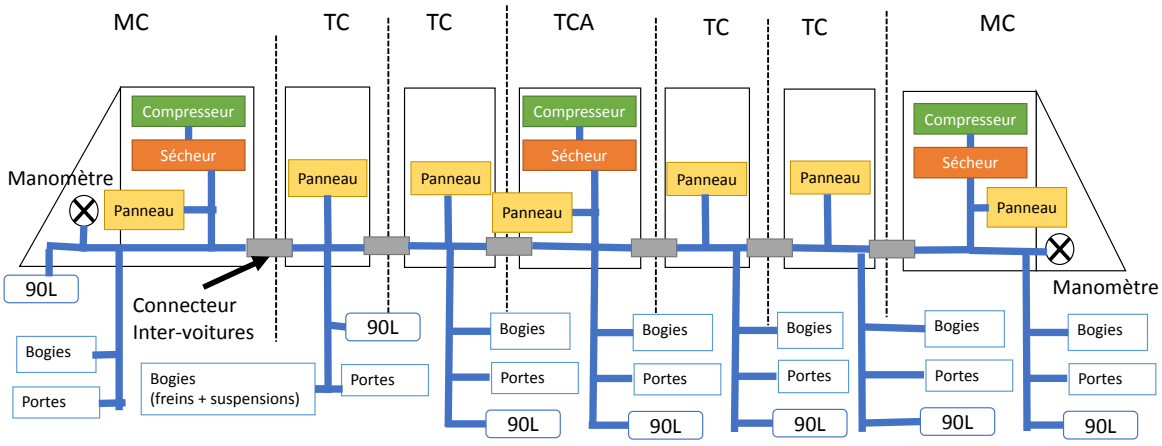


FIGURE 4.2 – Schéma structurel des circuits pneumatiques

4.1.2 Unité de freinage d'une roue d'un bogie

Les systèmes de freinage doivent être des composants à haut niveau de fiabilité. L'énergie pneumatique est couramment utilisée, car elle est plus robuste aux bruits électromagnétiques et vibratoires. Un autre avantage de tels systèmes est qu'ils continuent de fonctionner en mode dégradé, en cas de défaillance légère. Par exemple, si une vanne est légèrement ouverte, le système appliquera quand même une friction de freinage réduite. Le schéma structurel (figure 4.3) représente la partie pneumatique et mécanique du système de freinage d'une roue bogie. Celui-ci a deux entrées pneumatiques redondantes. La première, provenant directement du conduit principal, est régulée par le détendeur A_1 ; celle-ci est utilisée pour le freinage d'urgence. La deuxième entrée provient des suspensions ; celle-ci est régulée par le poids du système, via les coussins d'air des suspensions. Cette deuxième entrée est utilisée pour le freinage de service (cas nominal) puisqu' elle est plus souple de par sa régulation qui prend en compte le poids du train. Un réservoir de 20 litres filtre la dynamique de pression provenant des suspensions. Les systèmes de clapet antiretour G_1 sont équivalents à une porte directionnelle ; ils empêchent le retour de pression depuis le frein.

En freinage de service, la vanne U_2 est ouverte, les vannes U_1 et U_3 sont fermées. Dans ce mode, la pression provient des suspensions ; elle est régulée par le poids du train appliqué sur celles-ci. Ce flux pneumatique traverse ensuite les éléments U_2 et G_1, pour appliquer une pression aux plaquettes. Ces plaquettes exercent une force transversale sur le disque, ce qui freine la roue.

En freinage d'urgence, la vanne U_3 est fermée, les vannes U_1 et U_2 sont ouvertes. Ainsi, en plus de la pression provenant du conduit de suspension, la conduite principale aide au freinage. Les clapets antiretour G_1 permettent la redondance du système, puisqu'ils agissent comme une porte logique « OU » entre les deux flux. L'élément A_1 maintient une pression maximale de neuf bars, pour ne pas abimer le système. En effet, si une trop forte pression est appliquée aux plaquettes, celles-ci peuvent mécaniquement casser ou fusionner avec le disque.

Dans le mode d'échappement, les vannes U_1 et U_2 sont fermées, et U_3 est ouverte. Le ressort impose une force aux plaquettes, qui exercent une pression sur le système ; comme la vanne U_3 est ouverte, l'énergie pneumatique s'évacue. Ainsi, les freins se desserrent.

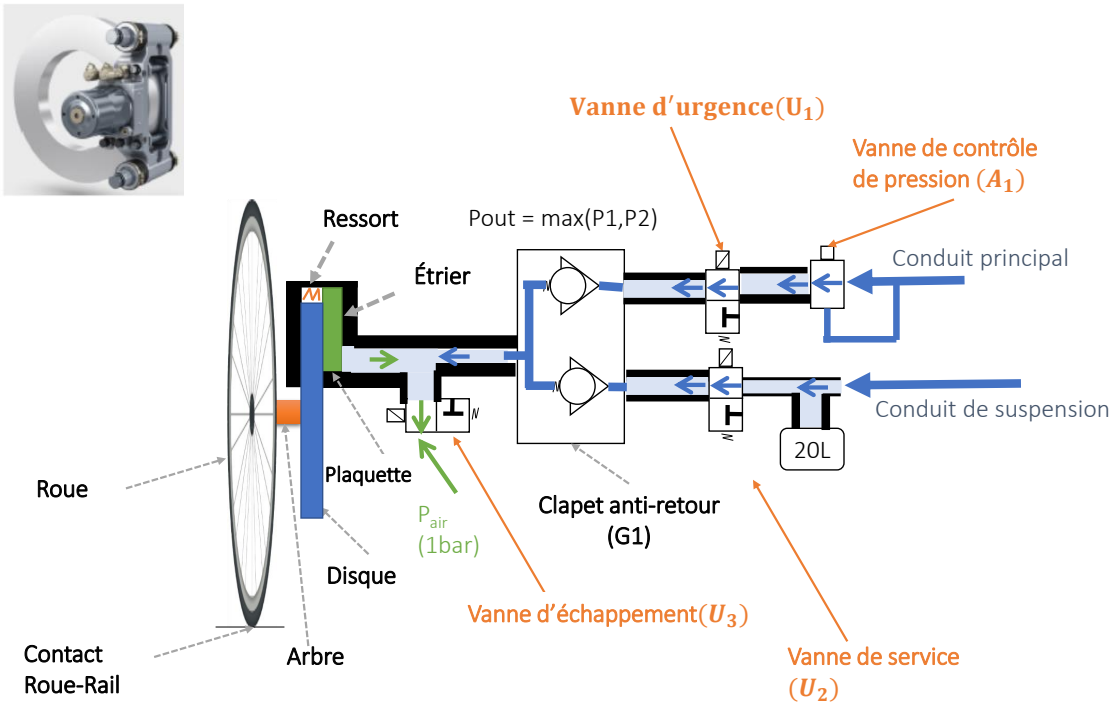


FIGURE 4.3 – Schéma structurel d'un système de freinage direct

4.2 Création du XML structurel

La figure 4.3 est d'abord standardisée vers un XML. Ce fichier contient ainsi l'ensemble des éléments présentés avec pour chacun le nom de l'élément, la localisation, le composant associés à l'élément, ses bornes, et les connexions entre les éléments. La figure 4.4 présente une partie de ce fichier. Sur celle-ci sont présents les deux éléments nommés « clapet_1 » et « clapet_2 », ils sont du même type (composant : « clapet »). Ces deux éléments possèdent chacun deux bornes : une d'entrée « in » et une de sortie « out ». Leurs bornes de sortie sont reliées ensemble par le signal de nom : « (5) ». Leurs bornes d'entrée sont respectivement connectées au conduit principal (« (4p) ») et au conduit de suspension (« (4s) »).

```
<appareil>
    <nom>clapet_1</nom>
    <composant>clapet</composant>
    <localisation>
      <subdivision>=MC</subdivision>
      <emplacement>/1</emplacement>
    </localisation>
    <gid>0-1c-a-db7</gid>
    <liste_bornes>
          <nbbornes>2</nbbornes>
           <borne>
            <nom>in</nom>
            <numero>1</numero>
            <emplacement>/1</emplacement>
            <line_sig>
              <nom>(2)</nom>
            </line_sig>
          </borne>
          <borne>
            <nom>out</nom>
            <numero>2</numero>
            <emplacement>2</emplacement>
            <line_sig>
              <nom>(5)</nom>
            </line_sig>
          </borne>
        </liste_bornes>
</appareil>
<appareil>
    <nom>clapet_2</nom>
    <composant>clapet</composant>
    <localisation>
      <subdivision>=MC</subdivision>
      <emplacement>/1</emplacement>
    </localisation>
    <gid>0-1d-10-d54</gid>
    <liste_bornes>
          <nbbornes>1</nbbornes>
           <borne>
            <nom>in</nom>
            <numero>1</numero>
            <emplacement>/1</emplacement>
            <line_sig>
              <nom>(4)</nom>
            </line_sig>
          </borne>
          <borne>
            <nom>out</nom>
            <numero>2</numero>
            <emplacement>2</emplacement>
            <line_sig>
              <nom>(5)</nom>
            </line_sig>
          </borne>
        </liste_bornes>
</appareil>
```

4.3 Génération du BGH en utilisant la bibliothèque des composants

4.3.1 Création du Bond graph à mots à partir du XML

Le bond graph à mots (cf. figure 4.12) représente la modélisation du système à l'aide d'un BG hiérarchisé, chaque nœuds est alors un sous-modèle décrit lui-même par un BG. Dans le cadre de développements automatisés, cela permet représenter chaque élément de CAO par un BGH équivalent. De plus, cela met en évidence les échanges d'énergie entre composants. Une librairie contenant l'ensemble des composants du système a été créée ; c'est-à-dire : . Les deux clapets et le cylindre sont liés par un effort commun, représenté à l'aide d'une jonction 0. Les éléments sont inter-connectés par des liaisons de puissance (cf. figure 4.12). Ces liaisons sont associées aux variables suivantes :

— P_{4p} et Q_{4p} respectivement la pression et le débit entre le conduit principal les clapets antiretour "P"

— P_{4s} et Q_{4s} respectivement la pression et le débit entre le conduit de suspension les clapets antiretour "S"

— Q_{5p} respectivement le débit de sortie du clapet antiretour "P", Q_{5s} le débit de sortie du clapet antiretour "S" et Q_5 le débit d'entrée du cylindre.

— P_{5p}, P_{5s}, P_5 référant à la pression entre le cylindre et les clapets .

— F_1 la force appliquée sur les plaquettes par le cylindre et V_1 la vitesse des plaquettes de frein.

— ω_1 la vitesse angulaire de la roue et T_1 son couple.

4.3.2 Bibliothèque des composants ferroviaires de Bond graphs

La librairie Bond Graph associe à tout composant d'un système un modèle de comportement donné sous la forme d'une représentation BG et des équations associées. Dans le cas du système étudié, la librairie BG contient les descriptions des 7 sous-systèmes mis en évidence figure 4.5.

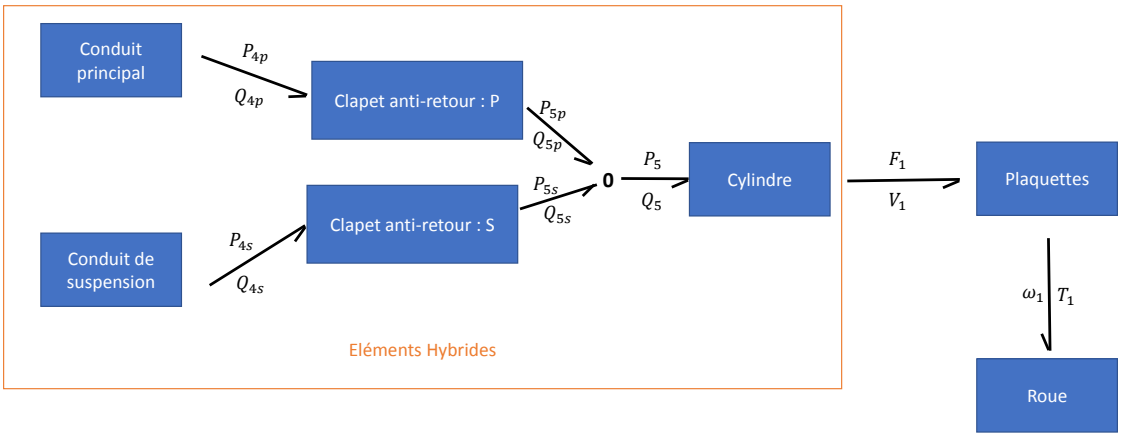


FIGURE 4.5 – Bond graph à mots

Conduit principal

Le conduit principal est constitué d'une entrée de pression de cinq bars maximum ; celle-ci est représentée par un élément MSe, en série avec une électrovanne dénommée EV_p.

Les électrovannes Le système possède trois électrovannes ; ces vannes, pilotées par une entrée électrique idéale, sont des éléments hybrides. À l'état ouvert (« ON »), elles laissent passer un débit légèrement ralenti par la résistance du système. À l'état fermé (« OFF »), le système coupe totalement le flux. Cet élément est modélisé par une jonction $1c$, reliée à un élément R. Deux ports permettent de connecter cet élément à son environnement ; une variable booléenne contrôle l'état du système (cf. figure 4.6). La résistance R est utilisée lorsque la vanne est ouverte (« ON »). En causalité résistive, l'équation 4.1 détermine la pression ; en causalité conductive, l'équation 4.2 détermine le débit.

$$\Delta P = (\frac{Q}{K_v})^2 \tag{4.1}$$

$$Q = K_v.sign(\Delta P).\sqrt{\mid \Delta P \mid} \tag{4.2}$$

où ΔP, Q sont respectivement la pression et le débit. $sign(\Delta)$ est une fonction qui vaut 1 si Δ positif, -1 sinon. Le paramètre K_v est déterminé à partir d'abaque constructeur ; il vaut 30 $\frac{m^3}{h}$.

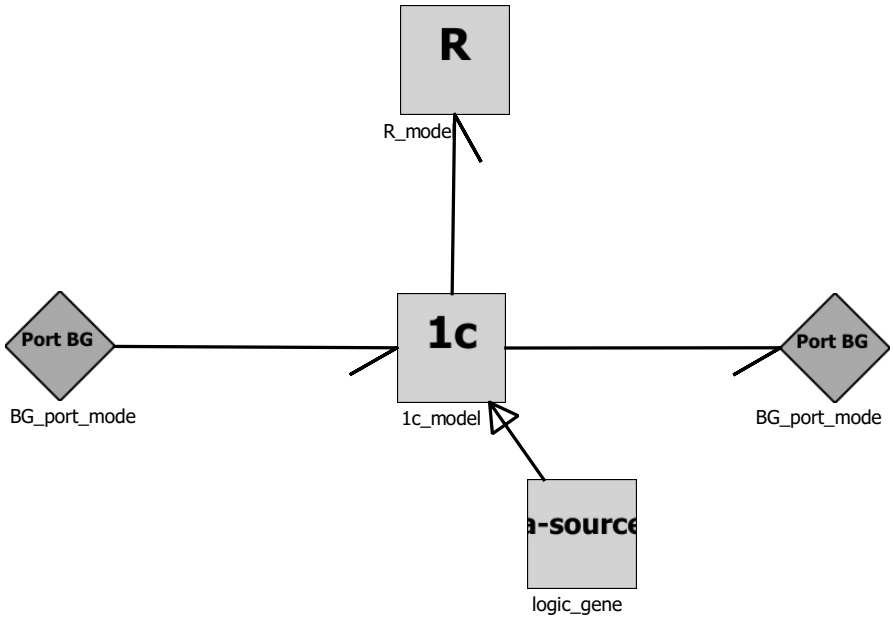


FIGURE 4.6 – Représentation de l'électrovanne

Conduit de suspension

Le conduit de suspension (cf. figure 4.7) est composé d'une entrée provenant des suspensions régulée par le poids du système. La dynamique de cette entrée (représentée par un élément MSe) est filtrée par un réservoir de vingt litres (élément C), qui contient un capteur de pression De. Ces éléments sont en série avec une électrovanne qui contrôle l'admission du flux.

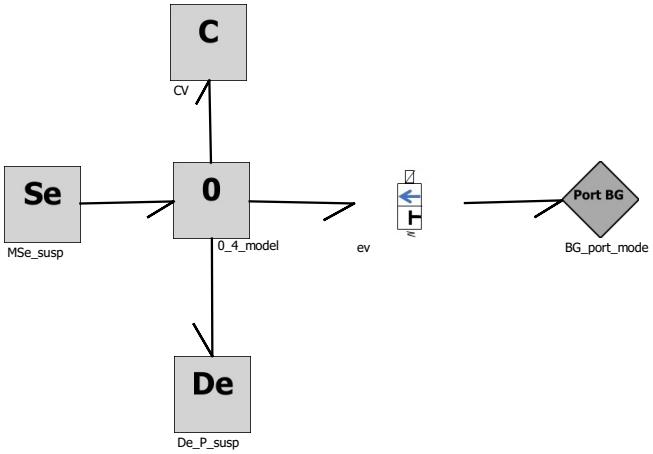


FIGURE 4.7 – BG du conduit de suspension

Les réservoirs Les réservoirs de pression sont des éléments stockant un volume d'air comprimé. En ce sens, ils peuvent être assimilés à un élément capacitif de premier ordre. La conductance C_f est estimée par l'équation 4.3 (MACIA et THALER, 2005, p. 96).

$$C_f = \frac{V_{tank}}{n.p} \tag{4.3}$$

où n est le coefficient polytropique, dans le cas d'un changement rapide de pression $n = 1.4$; et p est la pression de fonctionnement nominal. V_{tank} est le volume du réservoir.

Clapet anti-retour

Les clapets anti-retour laissent passer le flux d'air dans un sens, mais le bloquent dans l'autre. Leur fonctionnement est semblable à une diode électrique. On le représentera à l'aide d'une jonction contrôlée ; celle-ci est commandée par une inégalité entre les mesures de pression en amont et en aval du clapet. La figure 4.8 présente le modèle

utilisé, puisque les clapets sont en série avec les autres éléments du système, la jonction 1_c est préférée.

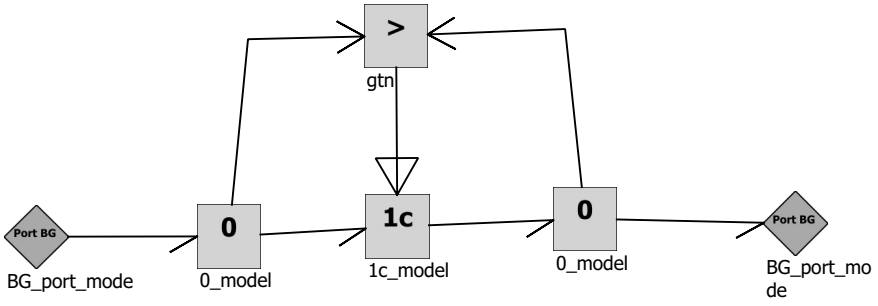


FIGURE 4.8 – Clapet anti-retour

Cylindre

La partie nommée cylindre contient (cf. figure 4.9) un réservoir de cinq litres (représenté par un capacité C_C), deux capteurs de pressions (représenté par les détecteurs De_{P92_m} et De_{P91_m}), une résistance nommée $R : Cc$ représente les effets de pertes de pression par frottement, un piston (modélisé par l'élément $TF : A_p$), et une électrovanne permettant l'échappement de l'air (modélisé par l'élément EV_{de}) vers l'extérieur (représenté par l'élément $Se : P_{air}$). La pression exercée sur le piston impose une force à l'extérieur (vers les plaquettes). Le piston est représenté par un transformateur linéaire défini par l'équation 4.4.

$$F = pA \tag{4.4}$$

oOù p est la pression d'entrée, F la force appliquée par le système aux plaquettes, et A un coefficient de transfert constant (équivalent à la surface pour un piston simple) qui dans ce modèle vaut $A = 0.122$.

Plaquettes

L'ensemble dénommé plaquettes est composé d'un système de plaquettes, de ressorts pour le desserrage, de la fixation, et des plaquettes elles-mêmes. La représentation BGH est disponible en figure 4.10.

L'élément $C : Cp$ représente les ressorts du système ; ils repoussent les freins en

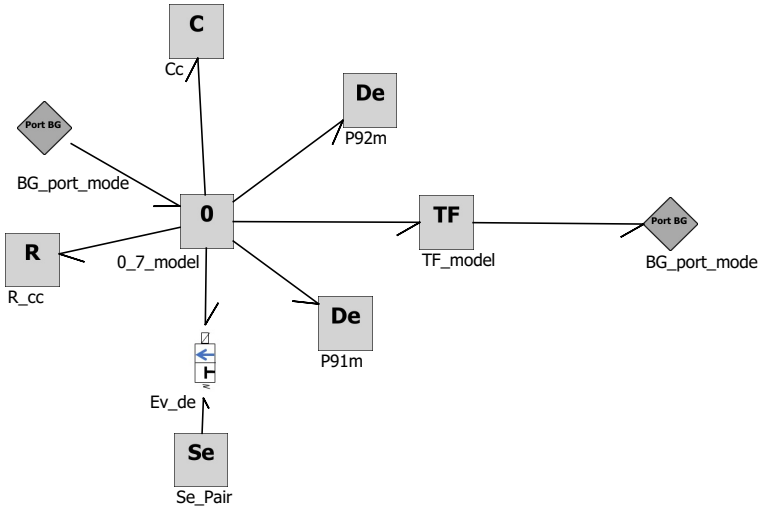


FIGURE 4.9 – BG du cylindre

desserrage. La force élastique de cet élément est définie par l'équation 4.5.

$$F = k. \int v.dt \tag{4.5}$$

où v est la vitesse de mouvement des plaquettes, F la force, et k la constante de raideur du ressort égale à $k = 1000kgf/cm = 980000N/m$. Ce système est fixé en son centre à un point d'attache sur le train ; celui-ci impose donc une vitesse nulle au système par rapport au train, représenté par le nœud $Sf : 0$.

$I : Mp$ est l'énergie cinétique de translation liée à la masse du système ; celle-ci est définie par l'équation 4.6

$$F = m\frac{dV}{dt} \tag{4.6}$$

avec m la masse du système ($m = 37\ kg$), F la force cinétique, et V la vitesse radiale de déplacement.

Ce système possède deux modes, représentés par la jonction $1c_4$: plaquettes collées au disque (état « ON »), plaquettes distantes du disque (état « OFF »). À l'état « ON », les plaquettes sont contre les disques ; elles imposent donc une vitesse nulle au piston. À l'état « OFF », les plaquettes sont éloignées du disque, et n'imposent donc aucun effort. Cet effort nul est modélisé par l'élément $Se : 0$. L'équation 4.7 définit le signal

de contrôle de la jonction $1c_4$.

$$H(V_3, F_3', t) = \begin{cases} if\ H(t-t_s) = 0 \begin{cases} 0\ if\ \int V_3(t)dt > x_{ref} \\ 1\ otherwise \end{cases} \\ if\ H(t-t_s) = 1 \begin{cases} 0\ if\ F_3 < F_s \\ 1\ otherwise \end{cases} \end{cases} \tag{4.7}$$

Où V_3 et F_3 sont respectivement la vitesse du système et la force appliquée au disque par friction, t est le temps, t_s le pas de calcul. F_s est la force statique de friction. x_{ref} est l'épaisseur du disque. $\int V_3(t)dt$ définit la position des plaquettes. A l'état initial, les plaquettes sont décollées du disque ($H(0) = 0$). Ensuite, l'équation est régie par deux modes :

— Soit au pas de calcul précèdent, la plaquette n'est pas collée au disque ($H(t-t_s) = 0$). Dans ce cas, tant que la position des plaquettes est éloignée du disque, $\int V_3(t)dt > x_{ref}$, celles-ci restent décollées du disque ($H(t-t_s) = 0$).

— Soit au pas de calcul précèdent, la plaquette est collée au disque ($H(t-t_s) = 1$). Dans ce cas, les plaquettes restent collées au disque ($H(t) = 1$) si la force appliquée par le système n'est pas supérieure à la force de friction, F_s, entre le disque et les plaquettes ; sinon, si la force appliquée par le système est supérieure à la force de friction alors les plaquettes se décollent du disque ($H(t) = 0$).

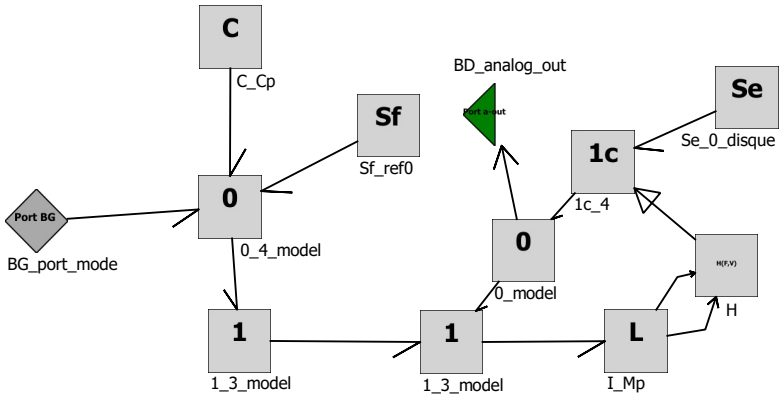


FIGURE 4.10 – BG de l'ensemble plaquette

Roue

La figure 4.11 présente le BG associé à l'ensemble constitué de la roue et du disque. La source d'effort $MSe : \mu$ représente la friction de la plaquette sur le disque ; cette fonction est complexe à modéliser, puisqu'elle dépend de la vitesse angulaire, de la température et de la microstructure et qualité des matériaux. À cause de cette complexité, les industriels préfèrent utiliser des relations empiriques déterminées à l'aide d'essais (équation 4.8).

$$F_6 = F_0 \ \mu \tag{4.8}$$

μ est le coefficient de friction ; avec les matériaux utilisé, il se situe entre 0.35 et 0.45. F_6 est la force appliquée par le disque à la roue, F_0 est la force appliquée par les plaquettes au disque.

Un transformateur $TF : 1/r$ représente la conversion idéale de la force de freinage sur le disque, en un couple opposé à la vitesse (équation 4.9).

$$\begin{cases} T = F.r \\ V = w.r \end{cases} \tag{4.9}$$

où T est le couple de freinage, F la force de friction, V la vitesse radiale, w la vitesse angulaire et r le rayon du disque (0.14 m).

Sf est la vitesse imposée par le train à la roue. L'énergie cinétique stockée par la masse de la roue est représentée par le nœud $I : Jw$, celle-ci est déterminée par l'équation 4.10.

$$T = m\frac{d\omega}{dt} \tag{4.10}$$

Où T le couple, ω la vitesse angulaire, m la masse du système roue-disque soit 3500 kg.

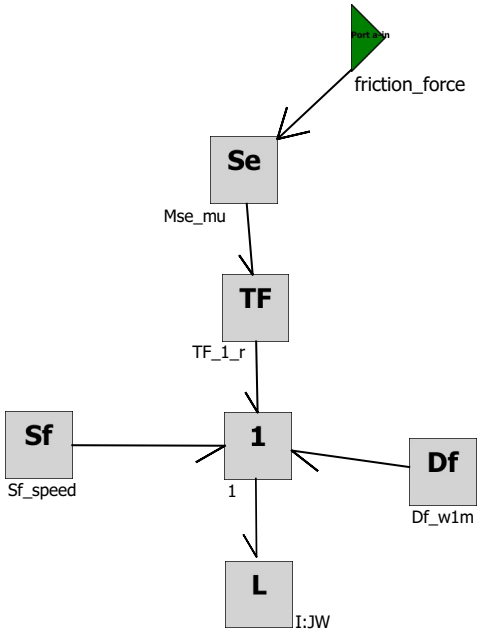


FIGURE 4.11 – BG de la roue

4.3.3 Implémentation sur Emulatio® du système

La figure 4.12 présente l'IHM du logiciel Emulatio®. A gauche, un menu de sélection permet de glisser et déposer des éléments BGH si besoin ; à droite, le canevas présente le BGH (un bond graph à mots dans cet exemple). Enfin, la barre de menu permet d'effectuer les actions logicielles (exécuter des algorithmes, importer un XML, sauvegarder, quitter, etc.). Le schéma présenté, sur la figure 4.12, est issu du remplacement de chaque composant exporté de la CAO par son équivalent de la libraire ferroviaire.

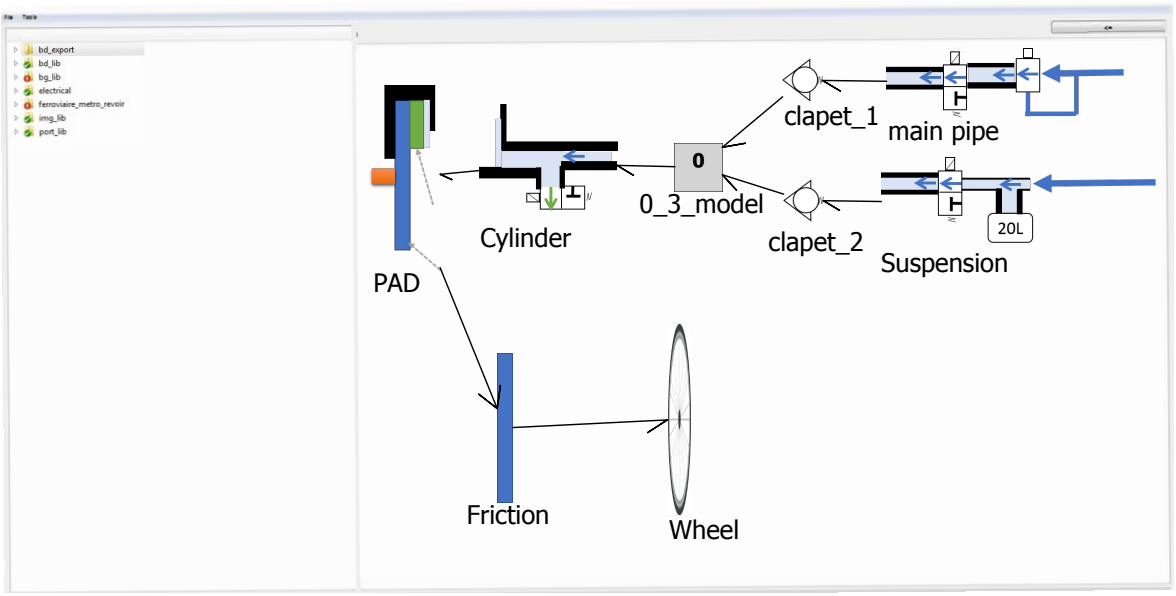


FIGURE 4.12 – Représentation Bond Graph à mots sur le logiciel Emulatio®

4.3.4 Bond graph complet

Le BGH complet du système de freinage est présenté en figure 4.13. Celui-ci reprend l'ensemble des éléments décrits précédemment. Il a été redessiné pour offrir une vision complète du fonctionnement du système. Suite à l'exécution de l'algorithme 6 sur l'ensemble du BG du frein, celui-ci détecte qu'il est nécessaire d'ajouter une jonction 0_c afin d'éviter tout conflit de causalités.

4.4 Affectation des causalités fixes : HSCAP

L'algorithme 4 d'affectation des causalités fixes a été appliqué au système. Le résultat est présenté en figure 4.14. Aucun conflit de causalités n'est à déplorer. Ceux-ci ont été précédemment résolus en amont, par insertion de jonctions 0_c complémentaires. Les éléments $CV : 20L$ et $I : JW$ sont cependant en causalités dérivées, ce qui ralentira et perturbera légèrement la simulation.

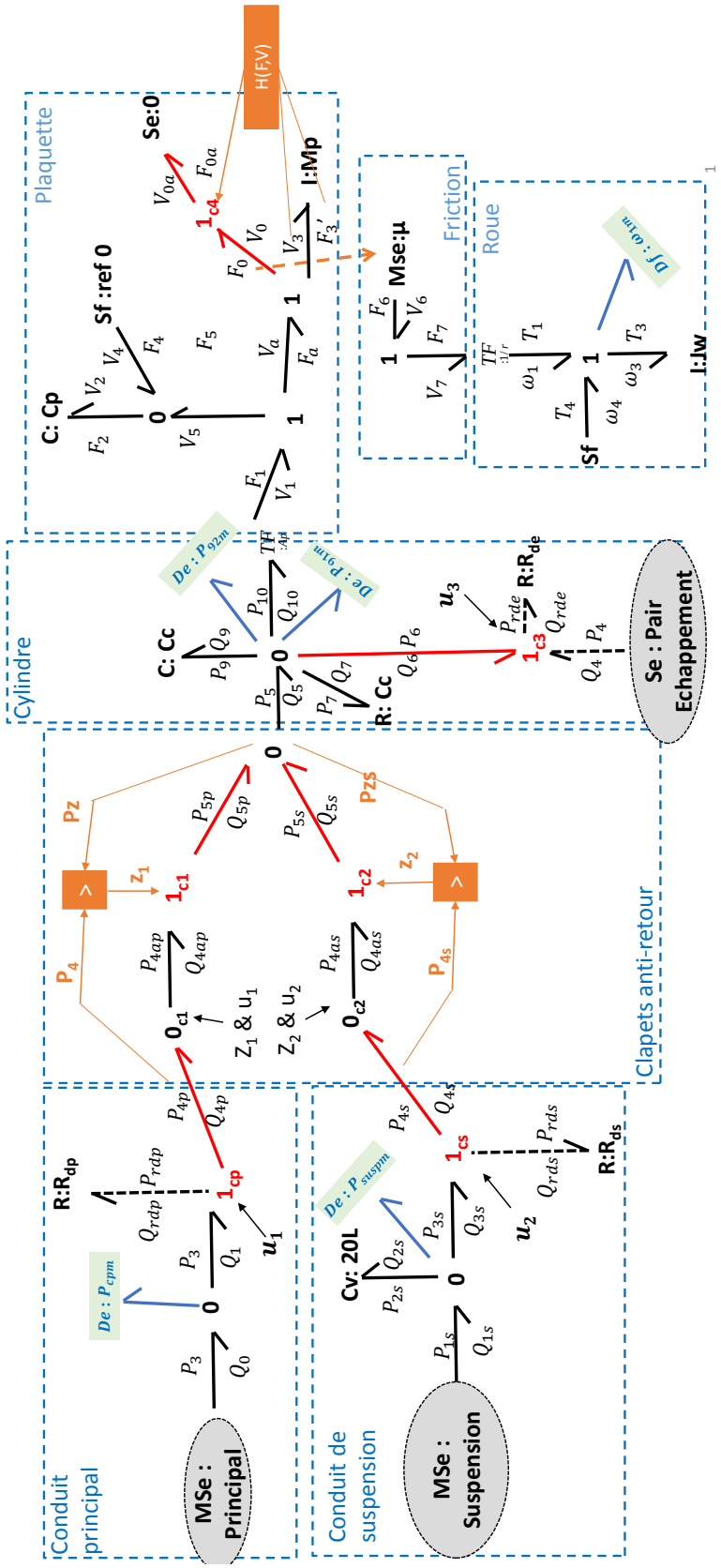


FIGURE 4.13 – BGH acausal complet du système de freinage

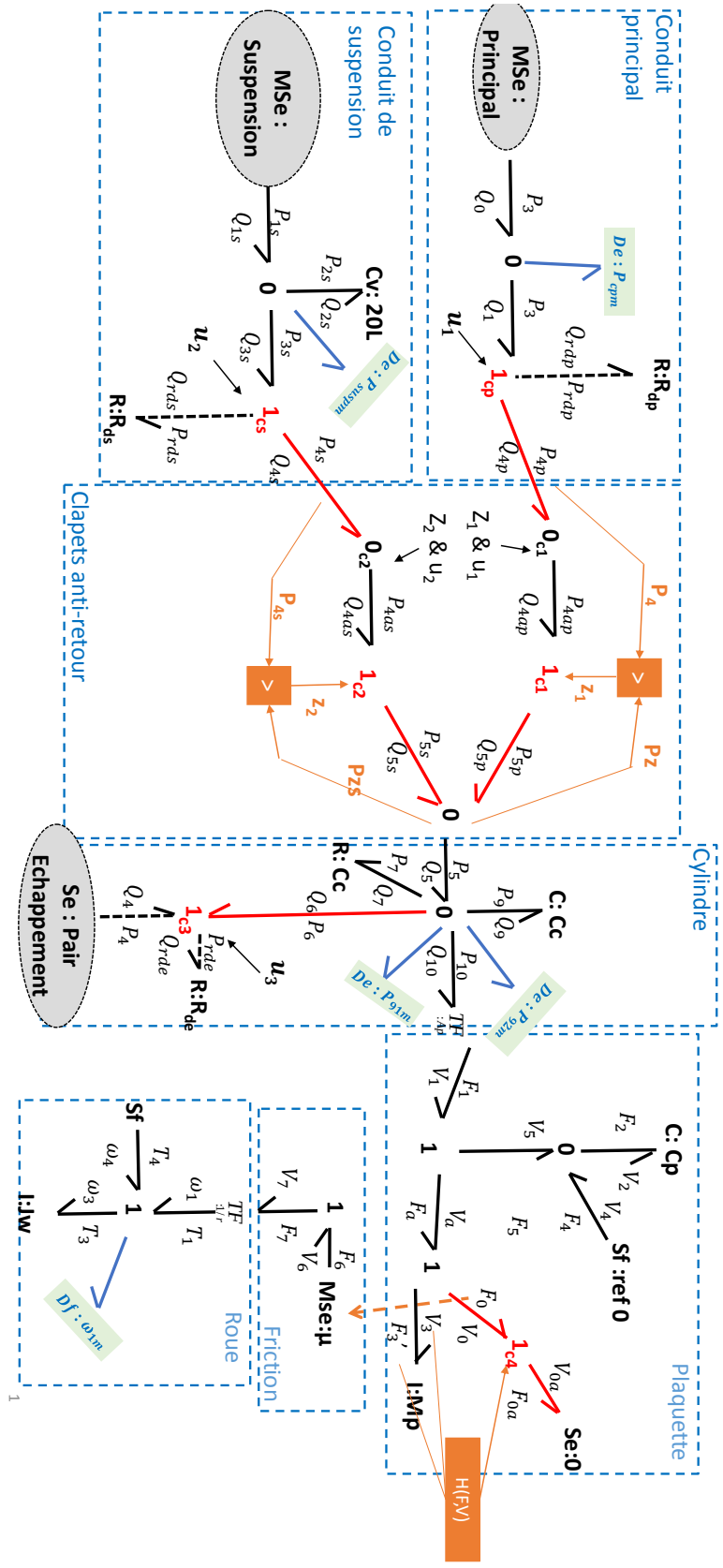


FIGURE 4.14 – BGH causal complet du système de freinage

4.5 Génération de schémas-blocs pour la simulation

L'algorithme 7 de génération des schémas-blocs est utilisé pour générer les blocs de simulation à partir du BGH dont les causalités fixes sont affectées. Cet algorithme exporte dans un premier temps l'ensemble des nœuds qui possèdent des liaisons à causalités fixes ; dans un deuxième temps il génère quatre différents sous-ensembles (cf. figure 4.15) dont la causalité est variable. Le résultat complet de cet algorithme est un schéma-bloc. Celui-ci est retranscrit sous Simulink pour simulation (cf. figure 4.16) puisque le logiciel ne gère pas encore la simulation des modèles. Enfin, pour éviter les boucles causales, des retards purs sont ajoutés à l'ensemble des entrées de commandes des électrovannes.

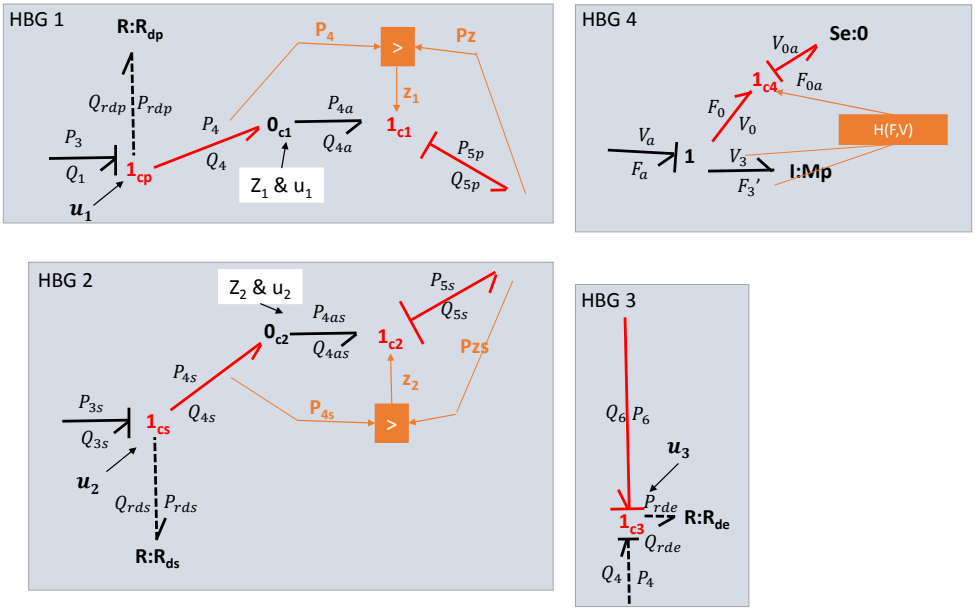


FIGURE 4.15 – Les différents ensembles hybrides du bond graph

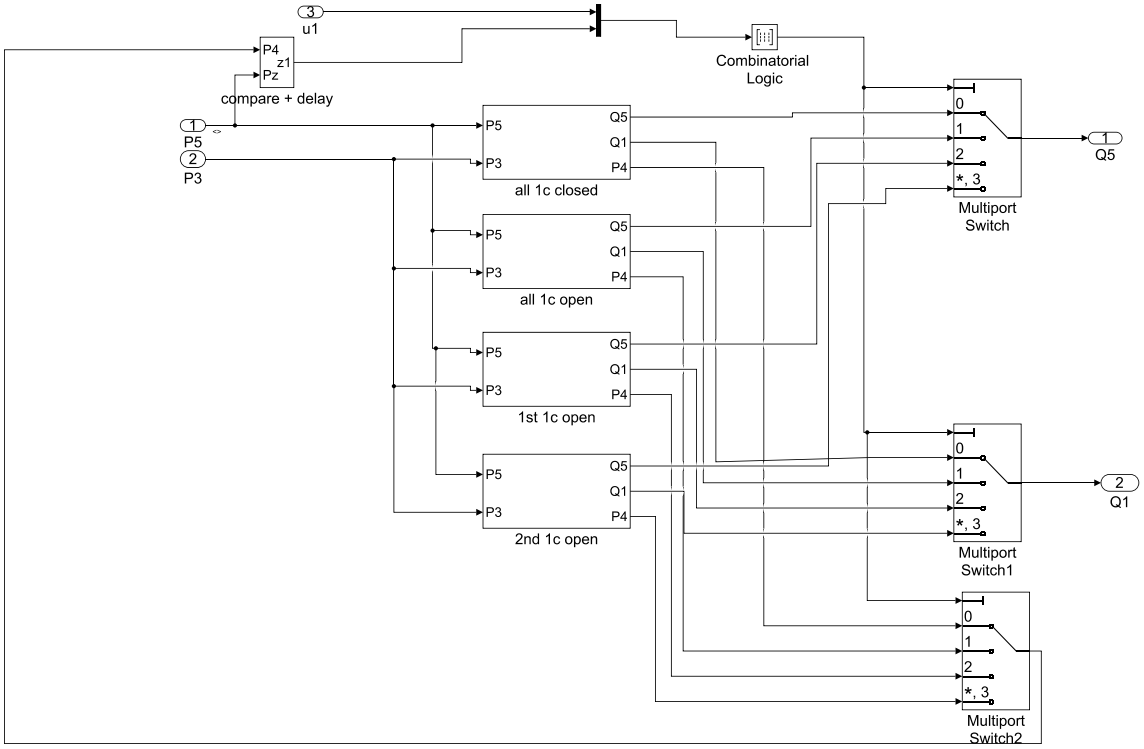


FIGURE 4.16 – Schéma bloc de l'ensemble "HBG 2"

Simulation des schémas-blocs

Le schéma-bloc est simulé afin de vérifier son fonctionnement. La simulation est effectuée à l'aide du logiciel Simulink, avec un pas de calcul dynamique. La figure 4.17 présente les résultats de l'enchaînement de trois événements : un freinage de service, un freinage d'urgence et un desserrage des freins (échappement). Les graphiques de gauche représentent l'état des électrovannes ; si la valeur est nulle, l'électrovanne est ouverte et laisse passer le flux ; si la valeur est de 1, l'électrovanne est fermée. Les graphiques de droite présentent la vitesse angulaire de la roue imposée en entrée, la pression déterminée par le capteur P_{92m}, et le couple de freinage produit par le système. La dynamique du système est assez longue ; en effet, le système possède un retard pur de 710 ms sur le premier freinage, puis un retard de 100 ms pour atteindre 90 % de la commande de couple désiré. Cette dynamique lente du système est liée au temps de pressurisation du système. Celle-ci peut être compensée par la commande d'échappement. L'électrovanne $u3$ ne se fermant que quand la pression est inférieure à 3.5 bar, le système n'est pas totalement dépressurisé à l'échappement, ce qui améliore ses performances.

La figure 4.18 est le résultat de la commande décrite précédemment. Lorsque le système atteint une pression minimale de 3.5 bar, la vanne d'échappement u^3 se ferme. Le système réagit alors en moins de 400 ms pour atteindre 90 % du freinage désiré. Les lois de commande doivent bien sûr être améliorées à l'aide d'outils tels que cette simulation, afin d'obtenir des délais encore plus courts.

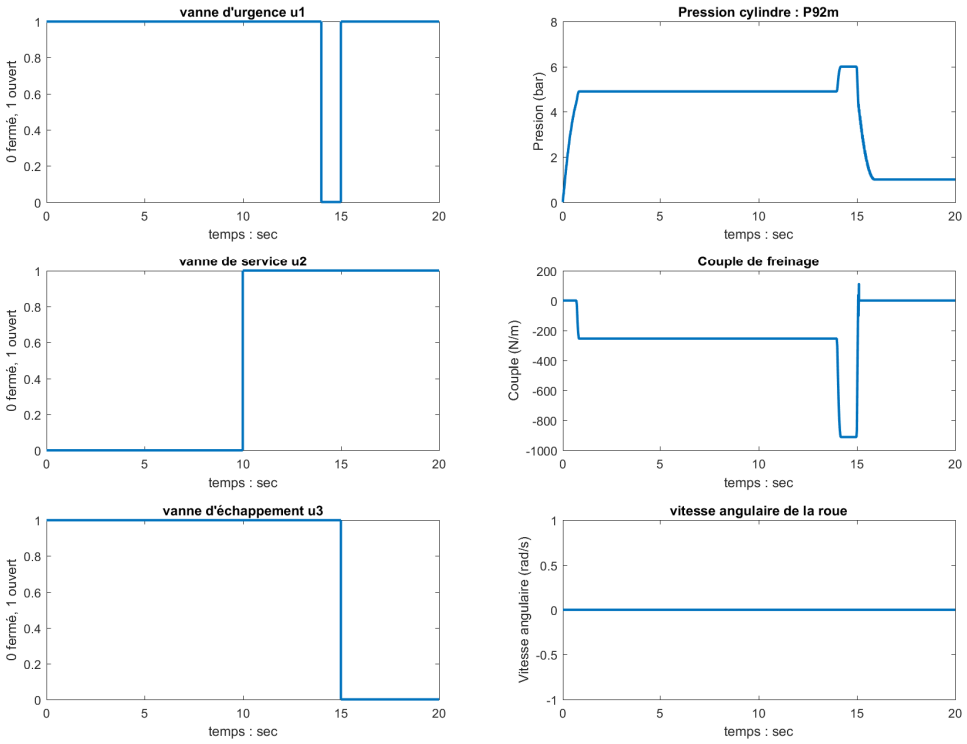


FIGURE 4.17 – Simulation d'un freinage de service et d'urgence, suivie d'un desserrage des freins

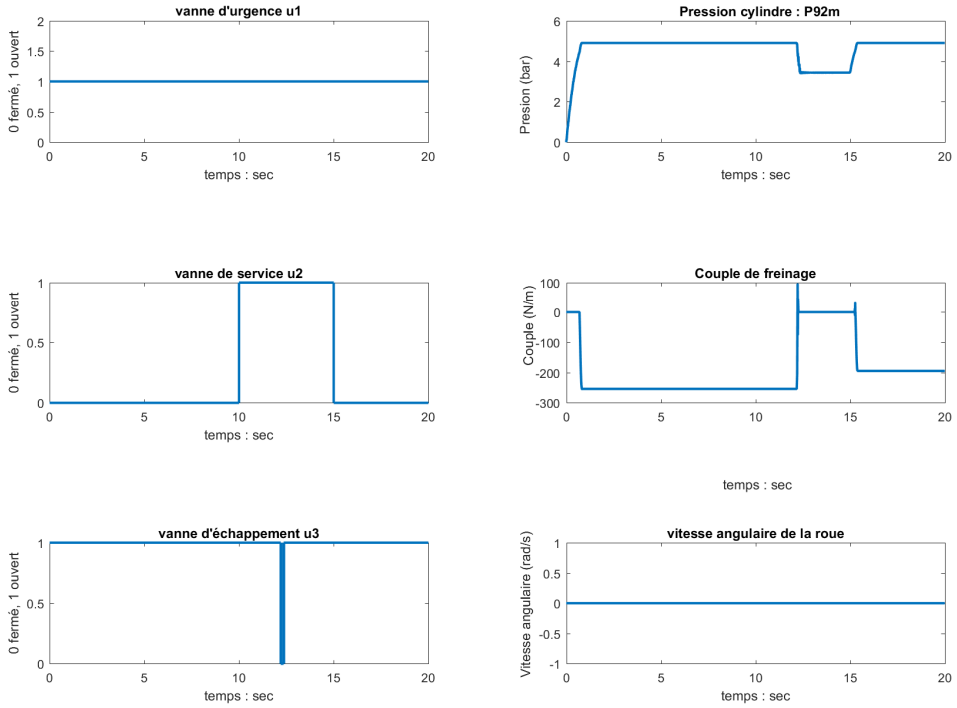


FIGURE 4.18 – Simulation de la commande de l'échappement stoppée à 3.5 bar

4.6 Diagnostic du système

Cette section présente le diagnostic du système développé. Dans un premier temps, elle analyse les relations de redondances analytiques générées ; une simulation validera le processus, et la FSM sera alors produite.

4.6.1 Génération des RRAG

Les RRAG sont générées par application de l'algorithme 8 à partir du BGH. Les quatre premières relations (équation 4.11) des relations de redondances matérielles. La première est liée à la redondance de deux capteurs, P_{92_m} et P_{91_m}. Les deuxième, troisième et quatrième, relations sont liées à des redondances entre les capteurs et les entrées du modèle (w_{1_m} redondé avec la vitesse du train pour $GARR_2$, P_{sup_m} redondé avec la pression d'entrée des suspensions pour $GARR_3$, et P_{cp_m} redondé avec la pression d'entrée principale pour $GARR_4$).

$$\begin{cases} GARR_1 = P_{92_m} - P_{91_m} \\ GARR_2 = w_{1_m} - w_4 \\ GARR_3 = Se_{susp} - P_{sup_m} \\ GARR_4 = Se_{cp} - P_{cp_m} \end{cases} \tag{4.11}$$

La cinquième relation ($GARR_5$) est plus complexe ; elle est générée à partir du capteur $P92m$, elle correspond à l'estimation du flux par ce capteur qui doit être nul. L'algorithme 8, génère dans un premier temps un BGH (cf. figure 4.19) qui contient les données nécessaires à la production d'un schéma-bloc représentant cette relation de redondance (cf. figure 4.20).

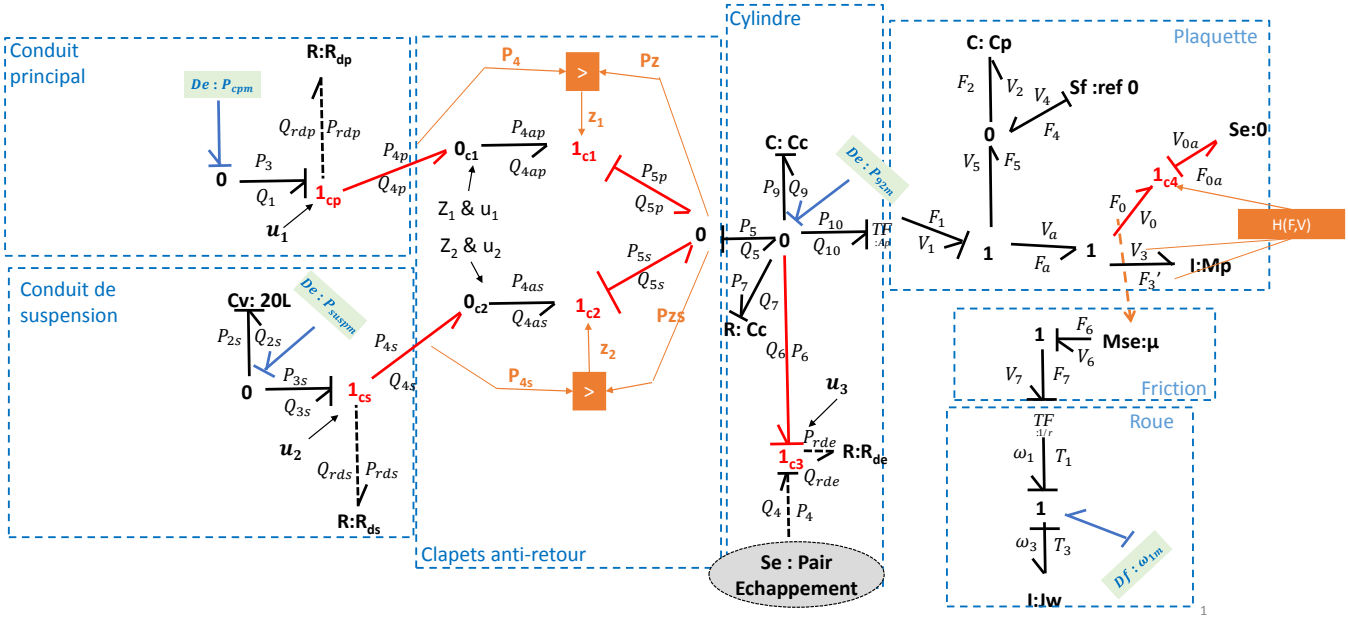


FIGURE 4.19 – BGH de génération de la cinquième RRAG ($GARR_5$)

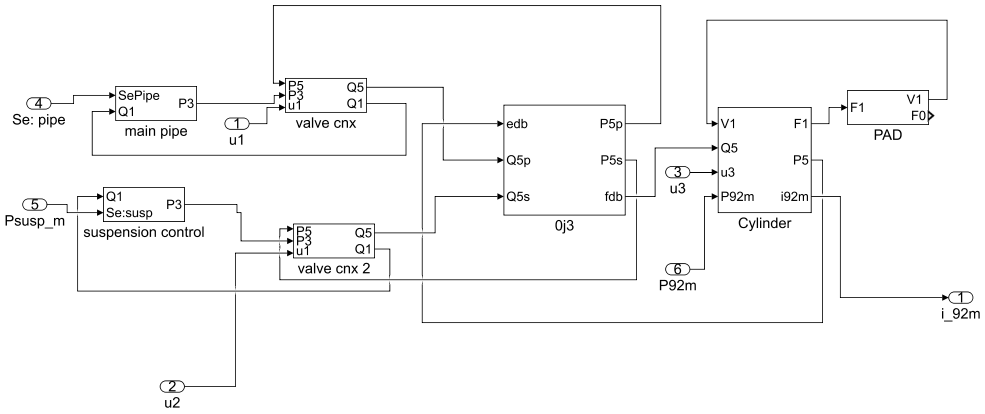


FIGURE 4.20 – schéma-bloc de génération de la cinquième RRAG ($GARR_5$)

Simulation des RRAG

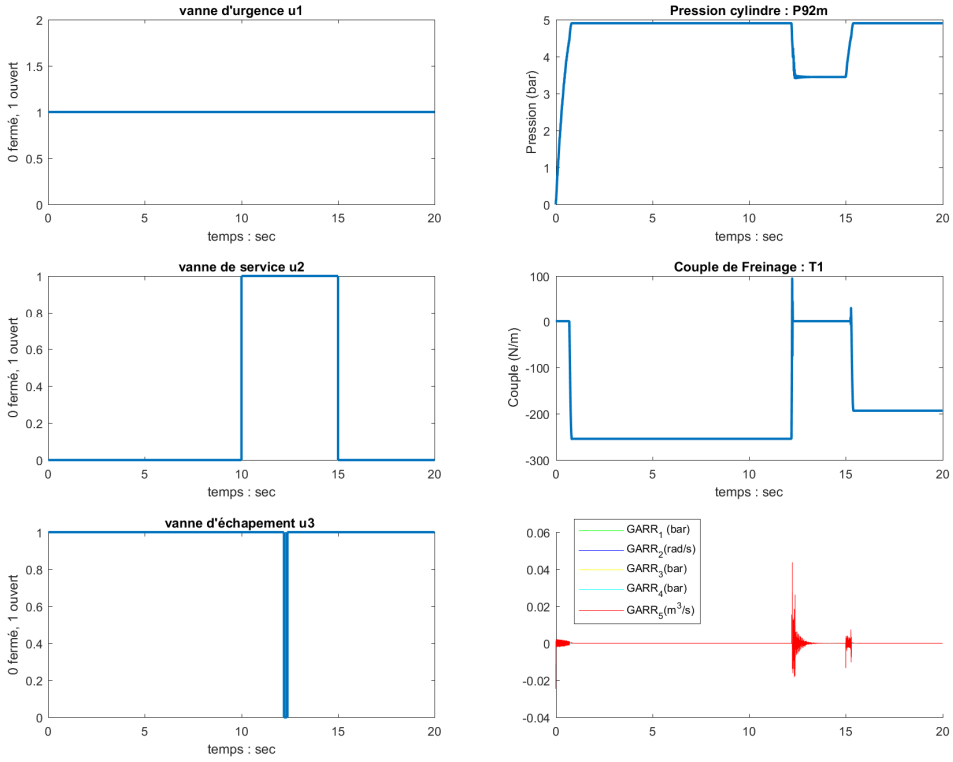


FIGURE 4.21 – Simulation du système sans défaut

La figure 4.21 correspond aux résultant de la simulation du système sans aucun défaut présent. L'évaluation de toutes les $RRAG$ donne des valeurs de résidus presque nulles. Il est à noter cependant que sur commutation, l'évaluation de la cinquième relation présente un retard de stabilisation ($GARR_5$) ; il est donc nécessaire d'ajouter un filtrage afin d'éviter de fausses alarmes.

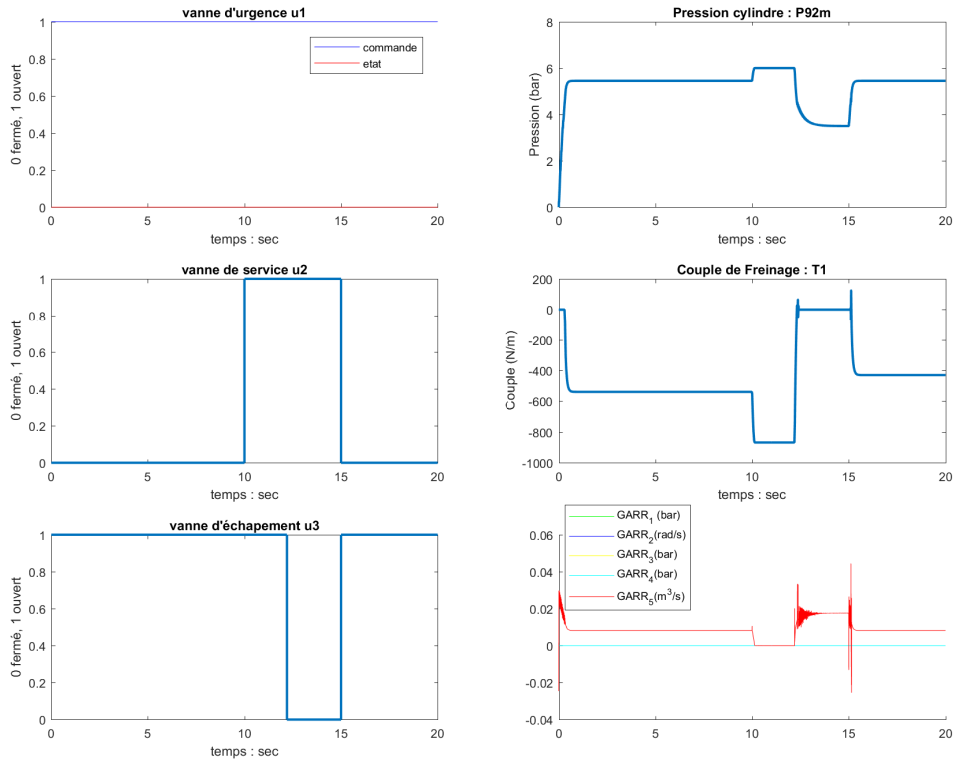


FIGURE 4.22 – Simulation d'un défaut de blocage de la vanne d'admission du conduit principal

La figure 4.22 présente les résultats de simulation du système possédant un défaut associé à un élément discret. Dans ce scénario, la vanne d'admission du conduit principal reste bloquée à l'état ouvert. Le résultat de l'évaluation de la cinquième relation ($GARR_5$) est différent de la valeur nulle pendant un long délai, cela permet la détection de ce défaut.

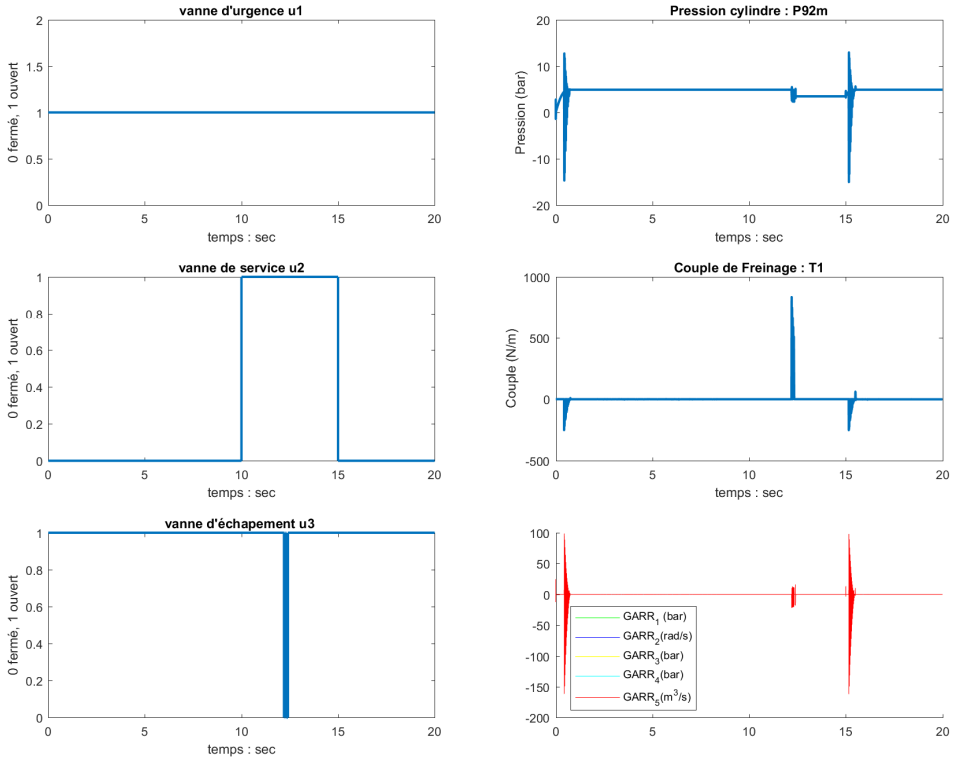


FIGURE 4.23 – Simulation d'une fuite dans le cylindre de frein

La figure 4.23 représente la simulation d'une fuite dans le circuit principal. Pour ce faire, la valeur du paramètre Cc est modifiée. L'évaluation de la cinquième relation de redondances analytiques montre un défaut clairement identifiable.

4.6.2 Génération de la FSM globale

Les relations $GARR_2$, $GARR_3$ et $GARR_4$ expriment la différence entre la mesure d'un capteur et la valeur d'une entrée de commande appliquée au système. Dans le cas où l'estimation d'une de ces relations est non nulle, on peut donc en déduire qu'il existe soit un défaut du capteur, soit de l'entrée du système. Or la valeur de cette entrée de système n'est pas disponible en réalité (elle l'est uniquement en simulation) ; ces relations ne doivent donc pas être intégrées à la FSM du système. Le résultat de l'application du théorème 2.2.2 sur les autres $GARR$ est présenté dans le tableau 4.1.

Ib	0	0	0	0	0	0	0	0	0	1	1	0	0
Db	1	1	1	1	1	1	1	1	1	1	1	1	1
Fautes	vanne service($u2$)	vanne échappement($u3$)	vanne urgence ($u1$)	clapets anti-retours	chambre cylindre	Ressorts	plaquettes	piston	ω_{1m}	P_{92_m}	P_{P91_m}	$Psupsm$	$Pcpm$
$GARR_1$	0	0	0	0	0	0	0	0	0	1	1	0	0
$GARR_5$	1	1	1	1	1	1	1	1	1	1	0	1	1

Tableau 4.1 – FSM du système électropneumatique

4.7 Conclusion

Dans ce chapitre, une application complète de freinage a été présentée. Cette application est utilisée pour illustrer et valider la méthodologie développée dans le chapitre 3 (cf. figure 3.3). D'autres applications seront cependant nécessaires pour fiabiliser le logiciel.

Concernant le système de freinage électropneumatique, on peut en conclure la faisabilité du diagnostic automatisé. Cependant, afin de réussir au mieux cette démarche, deux actions doivent être entreprises : il s'agit d'augmenter le nombre de capteurs et de filtrer les RRAG afin d'éviter les fausses alarmes. L'augmentation du nombre de capteurs peut aisément être réalisée, par l'ajout de mesures de pression, puisque ces éléments ont un coût faible. La difficulté réside dans la standardisation des entrées de mesure provenant d'un fournisseur extérieur (le fournisseur du frein). Le filtrage constitue un point scientifique très étudié actuellement ; de nombreux travaux existent sur le sujet. On pourra notamment citer les filtres par apprentissage (l'apprentissage pouvant se réaliser en partie en simulation) ; ceux-ci pourraient également permettre d'automatiser cette tâche de création de règles de filtrage (Abbeel et al., 2005).

Bibliographie du présent chapitre

Macia, N. F. et al. (2005). *Modeling and Control of Dynamic Systems*. Cengage Learning.

Abbeel, P. et al. (2005). *Discriminative Training of Kalman Filters*. In Proceedings of Robotics : Science and Systems.

Conclusion générale

Synthèse

Dans cette thèse, nous avons développé la méthodologie de conception d'un prototype logiciel de création automatisée de modèles, pour les systèmes multiphysiques hybrides possédant un très grand nombre de modes.

Dans un premier temps, les systèmes ont été classés selon leurs types de commutations. Les applications exploitées par CCAMY Systèmes possèdent des éléments à commutation autonome ou contrôlée. Dans le cadre de la modélisation de tels systèmes multiphysiques, le choix s'est porté sur l'utilisation du langage BGH, car il permet de décrire l'ensemble des modes.

Dans le second chapitre, après une présentation des BG conventionnels, nous avons montré l'intérêt de l'emploi de la jonction contrôlée, qui modélise les commutations idéales avec des contraintes causales minimales. À l'heure actuelle, aucun outil industriel n'utilise les jonctions contrôlées pour la modélisation des SDH. Partant de ce constat, il est apparu nécessaire de développer un nouvel outil pour répondre à ce besoin.

L'importance de la représentation des BGH au travers d'un langage moderne a été prouvée, lors de la présentation de l'architecture générale. Pour répondre à ce besoin, le langage BGML a été enrichi d'éléments hybrides et de balises en vue de son implémentation informatique. Puis, de nouveaux algorithmes de générations des schémas-blocs utilisés pour la simulation et de diagnostic ont été présentés. Enfin, les patrons de conception MVC et Abstract Factory ont été consultés et modifiés, afin de développer un logiciel qui puisse être facilement maintenu.

Dans le dernier chapitre, nous avons ensuite développé et validé une librairie ferroviaire de représentation des systèmes électropneumatiques. La librairie et le prototype logiciel ont ensuite été appliqués sur un système de freinage.

Finalement, cette démarche a permis la création d'un prototype logiciel fédérateur de simulation et diagnostic des systèmes hybrides, possédant de nombreux modes à

l'aide des BGH (notamment des jonctions contrôlées).

Perspectives

Les résultats obtenus par application du prototype sur l'application de freinage sont prometteurs. Cependant, de nombreuses améliorations doivent être apportées afin d'obtenir un logiciel et une démarche entièrement opérationnels.

Perspectives scientifiques

1 **Résolution des conflits causaux :** Un algorithme de résolution de conflit causal des jonctions contrôlées connectées de même type a été proposé. Suite à l'utilisation du prototype sur d'autres applications, il est fortement probable que d'autres algorithmes de gestion de ces conflits causaux devront être développés.

2 **Amélioration de l'affectation des causalités fixes :** L'algorithme d'affectation des causalités fixes pourrait être enrichi à l'aide de règles telles que celle utilisée pour les jonctions à causalité forcée. Cela permettrait de réduire, autant que faire se peut, l'utilisation de la force brute, consommatrice de temps.

3 **Réduction du modèle :** Dans ce mémoire, la réduction du modèle BGH n'a pas fait l'objet de développements particuliers (on supprime uniquement les jonctions identiques connectées entre elles). Cependant, de nombreuses règles restent à développer pour réduire la taille du modèle.

Perspectives industrielles

1 **Implémentation logicielle et matérielle de l'exécution de schémas-blocs :** Les schémas-blocs produits sont actuellement redessinés sous un autre outil pour leur exécution (par exemple Simulink). Pour une plus grande autonomie, le développement d'un module de simulation de ces schémas devra être développé. Enfin, pour une utilisation sur matériel roulant ou banc de test, les modèles générés doivent être embarqués sur carte informatique (DSP, FPGA, microP, etc.).

2 **Industrialisation du logiciel :** La création d'une IHM conviviale, la fiabilisation des logiciels, et le travail de codage complet de la théorie présentée sont des éléments nécessaires à une utilisation plus globale de l'outil. Ces développements

devront se réaliser par une équipe informatique complète ; pour répondre aux standards du domaine.

3 **Certification :** CCAMY Systèmes a développé le logiciel Emulatio® principalement pour le domaine ferroviaire. Il est donc nécessaire que le logiciel et les schémas-blocs respectent les règles du domaine, à savoir l'EN50128, et l'iec-61131-3.

4 **Amélioration du HBGML :** Le langage HBGML a été mis sous licence libre (précisément LGPLv3) et publié ouvertement sur GitHub. Si d'autres développeurs font évoluer ce langage, CCAMY Systèmes pourra réexploiter ces évolutions.

Annexe 1 : Les éléments des Bondgraph continus

Les BG continus sont constitués de différents éléments appelés nœuds. Pour les systèmes considérés dans ce travail (figure 2.2), l'ensemble des nœuds est :
$S = \{R\}\ U\ \{C\}\ U\ \{I\}\ U\ \{TF\}\ U\ \{GY\}\ U\ \{Se\}\ U\ \{Sf\}\ U\ \{De\}\ U\ \{Df\}\ U\ \{J\}$.

Les sources

Se Se est une source active d'effort. Cet élément correspond par exemple à une source de tension ou de pression idéale. La causalité de cet élément est obligatoire ; Se impose l'effort. Cette source peut être modulée au travers d'une commande extérieure, elle devient alors une source modulée d'effort, notée, MSe.

Sf Sf est une source active de flux. Cet élément correspond par exemple à une source de courant idéale ou de débit idéale. La causalité de cet élément est obligatoire ; Sf impose le flux. Cette source peut être modulée au travers d'une commande extérieure, elle devient alors une source modulée de flux, notée, MSf.

L'élément dissipatif : R

R représente un élément dissipatif d'énergie tels qu'une résistance électrique, des amortisseurs mécaniques, une résistance thermique, etc. Cet élément résistif est défini par une relation de résistance (équation 4.12) ou de conductance (équation 4.13). La causalité de l'élément R est totalement libre.

$$e(t) = \Phi_R(f(t)) \tag{4.12}$$

$$f(t) = \Phi_G(e(t)) \tag{4.13}$$

où Φ_G et Φ_R sont des bijections.

Les éléments de stockage d'énergie : C et I

Les éléments C et I sont des éléments passifs de stockage d'énergie. Les ressorts mécaniques, capacités électriques, cuves hydrauliques sont des exemples d'éléments représentés par les nœuds de type C. Les inerties mécaniques, inductances, seront modélisées par un élément I. L'élément C est défini par une relation donnant le flux en fonction de la dérivée de l'effort (Équation 4.14). L'élément I est défini par une relation donnant l'effort en fonction de la dérivée du flux (Équation 4.15). La causalité des éléments C et I est préférentielle, ce qui signifie que, pour le diagnostic il sera préféré une causalité dérivée, et pour la simulation une causalité intégrale. Cependant, s'il est impossible de respecter cette causalité préférentielle, cela ne représente pas un verrou pour l'utilisation du BG.

$$f(t) = \Phi_C(\dot{e(t)}) \tag{4.14}$$

$$e(t) = \Phi_I(\dot{f(t)}) \tag{4.15}$$

où Φ_I et Φ_C sont des fonctions possédant un unique inverse.

Les gyrateurs et transformateurs : GY et TF

Les gyrateurs et transformateurs sont des éléments conservateurs de puissance à deux ports. Le gyrateur est défini par l'équation 4.16 et le transformateur par l'équation 4.17. Les boites de vitesses ou convertisseurs électriques statiques sont par exemple modélisés par des transformateurs, tandis que les moteurs à courant continu peuvent être modélisés par des gyrateurs. La causalité des transformateurs et gyrateurs est propagée automatiquement suivant la causalité d'un des deux ports, suivant ces deux règles :

— Pour les transformateurs, si un port impose l'effort à l'élément alors l'autre port impose le flux. Réciproquement, si un port impose le flux alors l'autre port impose l'effort.

— Pour les gyrateurs si un port impose l'effort à l'élément alors l'autre port impose également l'effort. Réciproquement, si un port impose le flux alors l'autre port impose également le flux.

$$\begin{cases} e_1 = g.f_2 \\ e_2 = g.f_1 \end{cases} \tag{4.16}$$

$$\begin{cases} e_1 = m.e_2 \\ f_2 = m.f_1 \end{cases} \tag{4.17}$$

où e_1, e_2 sont les efforts aux deux ports de l'élément, et f_1, f_2 les flux respectifs aux bornes. r et m sont deux constantes.

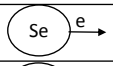
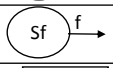
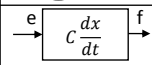
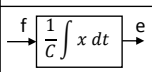
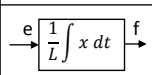
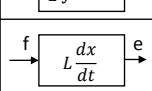
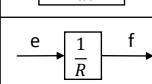
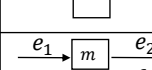
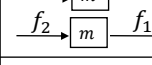
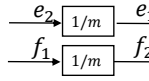
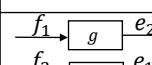
Types	Eléments	Représentations BG	Schémas-bloc	Equations
Sources	Effort			$e = Cste$
	Flux			$f = Cste$
Eléments Passifs	Capacité			$f = C\frac{de}{dt}$
				$e = \frac{1}{C}\int f\,dt$
	Inertie			$f = \frac{1}{L}\int e\,dt$
				$e = L\frac{df}{dt}$
	Résistance			$f = \frac{1}{R}e$
				$e = R\,f$
Jonctions à deux ports	Transformateur			$e_2 = m\,e_1$ $f_1 = m\,f_2$
				$e_1 = \frac{1}{m}\,e_2$ $f_2 = \frac{1}{m}\,f_1$
	Gyrateur			$e_1 = g\,f_2$ $e_2 = g\,f_1$
				$f_1 = \frac{1}{g}\,e_2$ $f_2 = \frac{1}{g}\,e_1$

FIGURE 4.24 – Les différents éléments des BG

Tableau de synthèse

Le tableau 4.24 synthétise les différents éléments développés ci-dessus, avec leurs équations et schémas-blocs équivalents. Dans les équations, les variables e et f correspondent respectivement aux variables d'effort et de flux. Pour faciliter la compréhension de ce tableau, les équations associées aux éléments : C, I, R sont considérés de premier ordre.

Les capteurs

De De est un capteur d'effort ; il représente par exemple un capteur de tension. Sa liaison, qui est souvent représentée par une flèche pleine, est purement indicative, puisque celle-ci n'a pas d'impact sur la dynamique du système. Il convient de noter qu'il existe une représentation équivalente de ces détecteurs comme des sources de flux nul (Gawthrop et Smith, 1996). La causalité de cet élément peut être plus spécifique, si nous sommes dans des conditions de diagnostic. Afin d'éviter la gestion de types de liaisons différentes pour les détecteurs et les autres éléments, nous utiliserons cette notation dans nos développements logiciels et dans ce mémoire.

Df Df est un capteur de flux, il est le dual du capteur d'effort De.

Les jonctions à conservation de puissance

Pour la modélisation des systèmes physiques réels, il parait utile de posséder des mailles de transfert de puissance sans pertes d'énergie. En effet, de nombreux éléments tels que les fils électriques, les tuyaux hydrauliques ou les conducteurs thermiques ont des pertes de puissance généralement négligeables. Pour représenter ces éléments en BG, les jonctions ont été développées. Une jonction est un élément multi-ports défini par l'équation 4.18.

$$\sum_{i=0}^{n} p_i(t) = 0 \tag{4.18}$$

où n est le nombre de ports de la jonction.

Jonction 0 La jonction 0 est une jonction à iso-effort ; tous les éléments connectés possèdent le même effort. Cette jonction respecte l'équation 4.19. Par analogie électrique,

cette jonction correspond à la loi des nœuds. La règle de causalité est strictement la suivante : un seul est unique effort est imposé à la jonction.

$$\begin{cases} \sum_{i=0}^{n} f_i(t) = 0 \\ e_1 = e_2 = ... = e_i = ... = e_n \end{cases} \tag{4.19}$$

où n est le nombre de ports de la jonction.

Jonction 1 La jonction 1 est une jonction à iso flux ; tous les éléments connectés possèdent le même flux. Cette jonction respecte l'équation 4.20. Par analogie électrique, cette jonction correspond à la loi des mailles. La règle de causalité est strictement la suivante : un seul est unique flux est imposé à la jonction.

$$\begin{cases} \sum_{i=0}^{n} e_i(t) = 0 \\ f_1 = f_2 = ... = f_i = ... = f_n \end{cases} \tag{4.20}$$

où n est le nombre de ports de la jonction.

Causalité des jonctions 1 et 0 À partir des contraintes de causalité définies précédemment, on peut déduire les deux règles suivantes d'assignation de causalité pour les jonctions :

1. Si une jonction possède une liaison déterminante (Définition 4.7.1), alors les autres liaisons sont en causalités opposées.
2. Si les causalités de toutes les liaisons, à l'exception d'une seule, sont affectées et que la condition 1 ne s'applique pas, alors la dernière liaison est en causalité opposée aux autres.

Définition 4.7.1 (Liaisons déterminantes)**.** *La liaison déterminante appelée « DB » en anglais est la liaison qui a une causalité unique. C'est donc la liaison qui impose l'effort pour les jonctions 0, et celle qui impose le flux pour les jonctions 1.*

Schéma-bloc équivalent des jonctions 1 et 0 Une fois la causalité des éléments déterminée, il est possible de convertir ces jonctions en schémas-blocs (Figure 4.25).

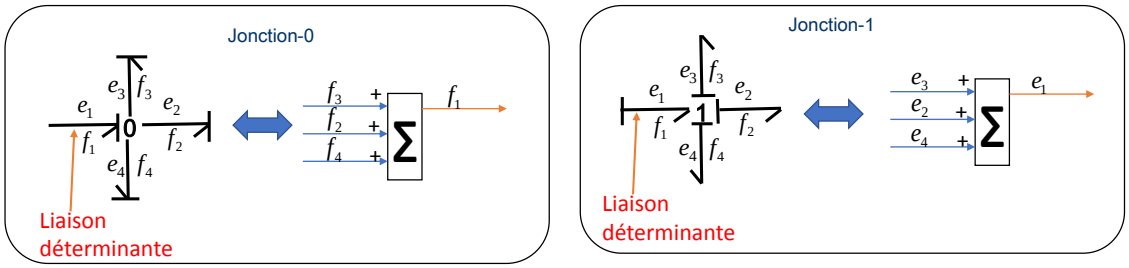


FIGURE 4.25 – Schéma-bloc équivalent des jonctions 1 et 0

Bibliographie du présent chapitre

Gawthrop, P. et al. (1996). *Metamodelling : For Bond Graphs and Dynamic Systems.* Prentice Hall.

Annexe 2 : Le langage de description SIDOPS

SIDOPS (ou SIDOPS+ dans sa dernière version) est un langage textuel de description des BG (Breunese et Broenink, 1997 ; Borutzky, 2010). Ce langage est constitué de classes, définissant les éléments BG, et de connexions, représentant les liaisons entre éléments BG.

Classes

Une classe est constituée des éléments suivants :

1. Le nom de la classe représentant le type d'élément modélisé.
2. La listes des ports avec pour chacun : la causalité préférentielle (dérivée ou intégrale) et l'orientation préférentielle.
3. Les paramètres caractéristiques de l'élément.
4. Les équations reliant les variables de flux et d'effort associées à chaque port.

La figure 3.10 présente un exemple de classe modélisant un élément C. Il possède un unique port p, dont la causalité est « preferred effort p » (effort appliqué en sortie). La capacité électrique C est décrite par un nombre réel. Enfin, l'effort est calculé par le produit de l'intégrale du flux et de l'inverse de la capacité électrique C.

Dans une deuxième version SIDOPS+, les classes sont polymorphiques par héritage ; par exemple, une classe PID hérite des opérations «intégrale» et «proportionnelle» de la classe PI et se spécifie par l'opération «dérivée».

Connexions

Les connexions se définissent au travers de symboles textuels. Par exemple, pour connecter deux éléments A et B, de A vers B, on écrira : A<=n=B (où n est la dimension

```
class C1 version 1
# C1: 1-port C energy store with 1-dimensional port
interface
ports: p
causality restrictions
preferred effort p
orientation restrictions
fixed in p
outputs: real state
parameters
real C
equations
state = int(p.f)
p.e = (1/C) * state
```

FIGURE 4.26 – SIDOPS : définition de la classe d'un élément C

de la liaison). Si l'on veut définir la causalité de la liaison, le symbole «|» est inséré. Ainsi, A|<=n=B, représentera un effort imposé à l'élément A. Dans le cas des éléments à ports multiples, l'identification du port se fera à l'aide du symbole «/» ; par exemple : A<=n=B/p signifie que le port «p» de l'élément B est connecté à A.

Synthèse

SIDOPS est un langage contenant l'ensemble des éléments permettant de décrire un BG. En ajoutant les classes associées aux jonctions contrôlées, il permettrait de représenter un BGH. SIDOPS présente néanmoins l'inconvénient majeur de n'être pas basé sur un format d'échange standard (JSON, XML, YAML) dans le monde informatique. La lecture et modification des fichiers est complexe, même à l'aide d'analyseurs syntaxiques tels que Bison ou Boost. Enfin, chaque nœud est décrit à l'aide d'équations analytiques alors que notre objectif industriel est de générer des schémas-blocs.

Bibliographie du présent chapitre

Borutzky, W. (2010). *Bond Graph Methodology*. London : Springer London.

Breunese, A. P. J. et al. (1997). *Modeling Mechatronic Systems Using The Sidops+ Language*. The Society for Computer Simulation International, p. 301-306.

Bibliographie

CCAMY~Systèmes (2018). *Votre expert en Contrôle Commande.* URL : `http://ccamy-systemes.fr/fr/`.

Cellier, F. E. et al. (2006). *Continuous System Simulation.* 6th edition. New York : Springer.

Six, B. et al. (2016). *Railway Direct Braking System Analysis Using Hybrid Bond Graph.* 2016 International Conference on Control, Decision and Information Technologies (CoDIT), p. 034-039.

— (2017). *Automated Model Builder of Hybrid Bond Graph : Application to a Two Level Inverter.* 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED), p. 325-330.

Six, B. et al. (2018). *Model Builder for Supervision of Hybrid Dynamic Systems : Application to Railway Rolling Stock.* SIMULATION, Status : Under review.

Bertalanffy, L. (2003). *General System Theory : Foundations, Development, Applications.* G. Braziller. 324 p.

Vidal, R. et al. (2002). *Observability and Identifiability of Jump-Linear Systems.* Proceedings of the IEEE Conference on Decision and Control. T. 4, p. 3614-3619.

Allam, S. et al. (1998). *Systems with Jumps : Theory and Applications.* Traitement du Signal.

El Guezar, F. (2009). *Modélisation et Simulation Des Systèmes Dynamiques Hybrides Affines Par Morceaux. Exemples En Électronique de Puissance.* Institut National des Sciences Appliquées de Toulouse.

Paoletti, S. et al. (2010). *On the Input-Output Representation of Piecewise Affine State Space Models.* IEEE Transactions on Automatic Control 55.1, p. 60-73.

Xu, J. et al. (2014). *Control and Estimation of Piecewise Affine Systems.* Sous la dir. de J. Xu et al. Woodhead Publishing.

Liberzon, D. (2003). *Switching in Systems and Control.* Réd. par T. Başar. Systems & Control : Foundations & Applications. Boston, MA : Birkhäuser Boston.

Domlan, Elom Ayih et al. (2006). *Systèmes à Commutation : Diagnostic de Fonctionnement et Identification de La Loi de Commutation.* Conférence Internationale Francophone d'Automatique, CIFA'2006. Bordeaux, France, CDROM.

Saleh, R. A. et al. (1994). *Mixed-Mode Simulation and Analog Multilevel Simulation.* Springer Science & Business Media.

Cellier, F. E. (1991). *Continuous System Modeling.* Springer.

Yu, X. et al. (2003). *Variable Structure Systems : Towards the 21st Century.* Springer.

Lin, H. et al. (2009). *Stability and Stabilizability of Switched Linear Systems : A Survey of Recent Results*. IEEE Transactions on Automatic Control 54.2, p. 308-322.

Liu, X. et al. (2013). *Structural Controllability of Switched Linear Systems*. Automatica 49.12, p. 3531-3537.

Cellier, F. E. et al. (1995). *Bond Graph Modeling Of Variable Structure Systems*. Proc. ICBGM'95 (Second International Conference on Bond Graph Modeling and Simulation), Las Vegas, p. 49-55.

Demailly, J. P. (2006). *Analyse numérique et équations différentielles*. 3e édition. Les Ulis, France : EDP Sciences.

Lambert, J. D. (1991). *Numerical Methods for Ordinary Differential Systems : The Initial Value Problem*. New York, NY, USA : John Wiley & Sons, Inc.

Borutzky, W. (2010). *Bond Graph Methodology*. London : Springer London.

Van der Schaft, A. et al. (1995). *The Hamiltonian Formulation of Energy Conserving Physical Systems with External Ports*. AEU - Archiv für Elektronik und Übertragungstechnik 49.5-6, p. 362-371.

Macchelli, A. (2002). *Port Hamiltonian Systems A Unified Approach for Modeling and Control Finite and Infinite Dimensional Physical Systems*. University of Bologna.

Haddad, W. et al. (2003). *Energy-Based Control for Hybrid Port-Controlled Hamiltonian Systems*. Automatica 39.8, p. 1425-1435.

Zainea, M. (2008). *Du Composant à l Automate Hybride Pour La Modélisation et La Simulation Des Systèmes En Communication : Application à l'électronique de Puissance*. IETR - Institut d'Electronique et de Télécommunications de Rennes.

Valentin, C. et al. (2006). *Hybrid Port–Hamiltonian Systems : From Parameterized Incidence Matrices to Hybrid Automata*. Nonlinear Analysis : Theory, Methods & Applications. Hybrid Systems and Applications (5)Hybrid Systems and Applications 65.6, p. 1106-1122.

Wainer, G. A. (2009). *Discrete-Event Modeling and Simulation : A Practitioner's Approach*. Computational analysis, synthesis, and design of dynamic models series. Boca Raton : CRC Press. 494 p.

Zeigler, B. (1984). *Multifacetted Modelling and Discrete Event Simulation*. San Diego, CA, USA : Academic Press Professional, Inc.

Sampath, M. et al. (1995). *Diagnosability of Discrete-Event Systems*. IEEE Transactions on Automatic Control 40.9, p. 1555-1575.

Bergero, F. et al. (2011). *PowerDEVS : A Tool for Hybrid System Modeling and Real-Time Simulation*. Simulation 87, p. 113-132.

Nutaro, J. (2010). *Building Software for Simulation : Theory and Algorithms, with Applications in C++*. Hoboken, N.J : Wiley-Blackwell.

John, K. H. et al. (2001). *IEC 61131-3 : Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools*. Berlin, Heidelberg : Springer-Verlag.

Samantaray, K. et al. (2008). *Model-Based Process Supervision : A Bond Graph Approach*. Advances in Industrial Control. London : Springer-Verlag.

Buntins, M. et al. (2013). *Hybrid Automata as a Modelling Approach in the Behavioural Sciences*. Electronic Notes in Theoretical Computer Science. Proceedings of the first workshop on Hybrid Autonomous Systems 297, p. 47-59.

Lynch, N. et al. (2003). *Hybrid I/O Automata*. Information and Computation 185.1, p. 105-157.

Alla, H. et al. (1992). *Du Grafcet aux réseaux de Petri*. 2ème édition revue et augmentée. Hermes Science Publications.

Paynter, H. M. et al. (1961). *Analysis and Design of Engineering Systems : Class Notes for M.I.T. Course 2.751*. Cambridge, Mass. : M.I.T. Press.

Mosterman, P. J. et al. (1998). *A Theory of Discontinuities in Physical System Models*. Journal of the Franklin Institute 335.3, p. 401-439.

Ould Bouamama, B. et al. (2006). *Software for Supervision System Design in Process Engineering Industry*. IFAC Proceedings Volumes. 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 39.13, p. 646-650.

Laprie, J. L. (1996). *Guide de La Sûreté de Fonctionnement*. Cépaduès.

Isermann, R. et al. (1997). *Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes*. Control Engineering Practice 5.5, p. 709-719.

Zwingelstein, G. (1995). *Diagnostic Des Défaillances : Théorie et Pratique Pour Les Systèmes Industriels*. Hermes Science Publications.

Du, D. et al. (2017). *Fault Diagnosis and Fault Tolerant Control for Discrete-Time Linear Systems with Sensor Fault*. IFAC-PapersOnLine. 20th IFAC World Congress 50.1, p. 15754-15759.

Bouibed, K. et al. (2014). *Actuator and Sensor Fault Detection and Isolation of an Actuated Seat via Nonlinear Multi-Observers*. Systems Science & Control Engineering 2.1, p. 150-160.

Savkin, A. V. et al. (2002). *Hybrid Dynamical Systems - Controller and Sensor Switching*. Springer Science & Business Media.

Louajri, H. (2015). *Centralized and Decentralized Fault Diagnosis of a Class of Hybrid Dynamic Systems : Application to Three Cell Converter*. Lille 1.

Patton, R. J. et al. (1999). *Artificial Intelligence Approaches to Fault Diagnosis*. IEE Colloquium on Condition Monitoring : Machinery, External Structures and Health (Ref. No. 1999/034), p. 5/1-518.

Wang, G. et al. (2017). *Quality-Related Fault Detection Approaches Based on Data Preprocessing*. IFAC-PapersOnLine. 20th IFAC World Congress 50.1, p. 15740-15747.

Reiter, R. (1987). *A Theory of Diagnosis from First Principles*. Artificial Intelligence 32.1, p. 57-95.

Iri, M. et al. (1979). *An Algorithm for Diagnosis of System Failures in the Chemical Process*. Computers & Chemical Engineering 3.1, p. 489-493.

Chatti, N. (2013). *Contribution to the Supervision of Dynamic Systems Using Signed Bond Graph*. Université des Sciences et Technologie de Lille - Lille I.

Cassez, F. et al. (2008). *Fault Diagnosis with Static and Dynamic Observers*. Fundamenta Informaticae 88.4, p. 497-540.

Staroswiecki, M. et al. (1991). *Optimal Design of FDI Systems via Parity Space and Observer Based Approaches*. International Conference on Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91, 143-148 vol.1.

Staroswiecki, M. et al. (2004). *Fault Detection, Supervision and Safety of Technical Processes*. SAFEPROCESS. Elsevier. 1210 p.

Abdallah, I. et al. (2017). *Event Driven Hybrid Bond Graph for Hybrid Renewable Energy Systems Part I : Modelling and Operating Mode Management*. International Journal of Hydrogen Energy.

Merzouki, R. et al. (2012). *Intelligent Mechatronic Systems : Modeling, Control and Diagnosis*. Springer Science & Business Media. 960 p.

Šarga, P. et al. (2012). *Simulation of Electrical System Using Bond Graphs and MATLAB/Simulink*. Procedia Engineering. Modelling of Mechanical and Mechatronics Systems 48, p. 656-664.

Niu, G et al. (2015). *Fault Diagnosis of Locomotive Electro-Pneumatic Brake through Uncertain Bond Graph Modeling and Robust Online Monitoring*. Mechanical Systems and Signal Processing 50-51, p. 676-691.

Loureiro, Rui et al. (2014). *Extension of the Bond Graph Causality Inversion Method for Fault Detection and Isolation*. Mechatronics 24.8, p. 1042-1049.

Gawthrop, P. et al. (1996). *Metamodelling : For Bond Graphs and Dynamic Systems*. Prentice Hall.

Karnopp, D. et al. (1968). *Analysis and Simulation of Multiport Systems : The Bond Graph Approach to Physical System Dynamics*. M.I.T. Press. 248 p.

Broenink, J. F. (1999). *Introduction to Physical Systems Modelling with Bond Graphs*. SiE Whitebook on Simulation Methodologies, p. 1-31.

Abdallah, Ibrahim (2017). *Event-Driven Hybrid Bond Graph : Application : Hybrid Renewable Energy System for Hydrogen Production and Storage*. Lille 1.

Borutzky, W. (2015). *Bond Graph Model-Based Fault Diagnosis of Hybrid Systems*. Springer International Publishing.

Ould Bouamama, B. et al. (2006). *Supervision of an Industrial Steam Generator. Part I : Bond Graph Modelling*. Control Engineering Practice 14.1, p. 71-83.

Dauphin-Tanguy, G. (2000). *Les bond graphs*. Hermes Science Publications.

Stromberg, J. E. et al. (1993). *Variable Causality in Bond Graphs Caused by Discrete Effects*. ICBGM 1993. San Diego.

Umarikar, A. C. et al. (2005). *Modelling of Switching Systems in Bond Graphs Using the Concept of Switched Power Junctions*. Journal of the Franklin Institute 342.2, p. 131-147.

20sim (2016). *20-Sim 4.6 Reference Manual*. Getting Started with 20-sim.

Roychoudhury, I. et al. (2011). *Efficient Simulation of Hybrid Systems : A Hybrid Bond Graph Approach*. Simulation 87.6, p. 467-498.

Dupuis, J. F. (2009). *libBondGraph*. URL : `https://sourceforge.net/projects/libbondgraph/?source=navbar`.

Granda, J. J. et al. (1997). *New Developments in Bond Graph Modeling Software Tools : The Computer Aided Modeling Program CAMP-G and MATLAB*. Computational

Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics. T. 2, 1542-1547 vol.2.

AMESIM (2016). *Simcenter Amesim Platform : Siemens PLM Software*. URL : `https://www.plm.automation.siemens.com/fr/products/lms/imagine-lab/amesim/platform/index.shtml`.

Modelica_Association (2012). *Modelica Language Specification*.

Juhász, T. et al. (2008). *CAD to SIM : CAD Model Conversion for Dymola-Based Mechatronic Simulation*. Tenth International Conference on Computer Modeling and Simulation (Uksim 2008), p. 289-294.

Cellier, F. E. et al. (2005). *The Modelica Bond Graph Library*. Proc. of the Modelica Conference, 2005, p. 57-65.

Ronkowski, M. (2008). *Modelling of Electrical Machines Using the Modelica Bond-Graph Library*. Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th, p. 880-886.

Umarikar, A. C. et al. (2006). *Bond Graph Simulation and Symbolic Extraction Toolbox in MATLAB/SIMULINK*. Indian Institute of Science. 86.1.

ISO-9834 (2014). *ISO/IEC 9834-8 :2014*.

Roychoudhury, I. et al. (2007). *A Method for Efficient Simulation of Hybrid Bond Graphs*. International Conference on Bond Graph Modeling (Icbgm), p. 177-184.

Zainea, M. et al. (2005). *Automatic Simulink Model Building for Physical Switching Systems*.

Shampine, L. et al. (1999). *Solving Index-1 DAEs in MATLAB and Simulink*. SIAM Review 41.3, p. 538-552.

SIMULINK (2018). *Algebraic Loops - MATLAB & Simulink - MathWorks France*. URL : `https://fr.mathworks.com/help/simulink/ug/algebraic-loops.html#bsjdo3y-1` (visité le 23/04/2018).

Borutzky, W. (2006). *BGML – a Novel XML Format for the Exchange and the Reuse of Bond Graph Models of Engineering Systems*. Simulation Modelling Practice and Theory 14.7, p. 787-808.

Codesynthesis (2014). *Tree/Customization Guide - Code Synthesis Wiki*. URL : `http://wiki.codesynthesis.com/Tree/Customization_guide`.

Lasater, C. G. (2010). *Design Patterns*. Wordware Publishing, Inc. 306 p.

Fowler, M. (2003). *UML Distilled : A Brief Guide to the Standard Object Modeling Language*. 3 edition. Boston : Addison-Wesley Professional. 208 p.

Ellson, J. et al. (2003). *Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools*. Graph Drawing Software. Springer-Verlag, p. 127-148.

Macia, N. F. et al. (2005). *Modeling and Control of Dynamic Systems*. Cengage Learning.

Abbeel, P. et al. (2005). *Discriminative Training of Kalman Filters*. In Proceedings of Robotics : Science and Systems.

Breunese, A. P. J. et al. (1997). *Modeling Mechatronic Systems Using The Sidops+ Language*. The Society for Computer Simulation International, p. 301-306.

Génération automatique de modèles pour la supervision des systèmes dynamiques hybrides

Application aux systèmes ferroviaires

Résumé

Ce travail de thèse présente différentes contributions pour la génération automatique de modèles représentant les Systèmes Dynamiques Hybrides (SDH) caractérisés par plusieurs modes de fonctionnement. Les composants du système (notamment les capteurs) peuvent être manuellement sélectionnés ou automatiquement exportés à partir des données de Conception Assistée par Ordinateur (CAO) ; ces éléments sont ensuite interconnectés pour reproduire le modèle complet du système industriel. À partir de ce modèle, des schémas-blocs de simulation et de diagnostic, ainsi que la Matrice de Signature de Fautes (FSM) seront produits. L'approche est basée sur les Bonds Graph Hybrides ; la présence de commutations engendre des dynamiques variables (notamment des changements de causalité). Pour lever ces verrous, différents algorithmes sont proposés. En comparaison des logiciels existants, les algorithmes proposés sont applicables sur les systèmes continus, discrets ou hybrides. Les théories et algorithmes développés sont appliqués sur un système ferroviaire de freinage électropneumatique.

Automated Model builder for supervision of Hybrid Dynamic Systems

Applied on a railway rolling stock system

Abstract

This thesis work contributes to perform an automated model builder for Hybrid Dynamic Systems (HDS) with numerous modes. Technological components including sensors with an iconic format can be automatically exported from a computer-aided design (CAD) scheme or manually drag from a database and interconnected, so as to produce the overall HDS model, following industrial technological schemes. Once the model has been created, block diagram for simulation and diagnosis and a Fault Signature Matrix (FSM) could be generated. The theory and algorithm behind the software are based on Hybrid Bond Graphs (HBG). The switching behaviour engenders variable dynamics (particularly causal changes). To solve this problematic, news algorithm are performed. Compared with developed programs for automated modelling, the presented algorithms are valid for continuous, discrete and hybrid systems. The theory is illustrated by an industrial application which consists of the pneumo-electrical control of rolling stock.

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille – CNRS UMR 9189 – Avenue Paul Langevin – 59650 Villeneuve d'Ascq – Site web : `www.cristal.univ-lille.fr`