N° d'ordre : 836

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR D'UNIVERSITE

spécialité PRODUCTIQUE :
AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

par

Abdellah AZMANI

Analyse qualitative d'un Bond-Graph par les techniques de l'Intelligence Artificielle. Contribution à la conception et à la réalisation du logiciel d'aide à la modélisation ARCHER.

soutenue le 19 décembre 1991 devant le jury d'examen

MM.	P. BORNE	Président
	J.P. DELAHAYE	Rapporteur
	S. SCAVARDA	Rapporteur
	M. BENREJEB	Examinateur
	J.P. CASSAR	Examinateur
Mme	G. DAUPHIN-TANGUY	Examinateur

Thèse dirigée par Mme G. DAUPHIN TANGUY, Professeur à Ecole Centrale Lille

AVANT-PROPOS

Table de matières

CHAPITRE 4 : SPECIFICITES INFORMATIQUES DU PROJET ARCHER

Introduction générale

Introduction générale

La modélisation suscite de plus en plus d'intérêt et est nécessaire à toute étude préalable d'un système dynamique.

A travers le monde, et dans les différents laboratoires concernés, l'automatisation de cette phase préalable à l'étude de n'importe quel système préoccupe de nombreux spécialistes de tout bord et fait l'objet d'une multitude de projets. Ces derniers ont tous un point commun : chercher comment traduire les informations, relatives au système étudié, sous une forme représentant le plus fidèlement possible le modèle réel.

Pour cela, mettre au point un logiciel de modélisation des systèmes physiques paraît une solution idéale.

Au laboratoire d'Automatique et d'Informatique Industrielle de Lille (Ecole Centrale de Lille), ce projet de logiciel a pour nom ARCHER. Il repose sur la méthodologie bond-graph (*) et sur l'utilisation des techniques de l'intelligence artificielle.

Le choix de l'outil bond-graph est justifié par les points suivants : tout d'abord, le type de représentation peut être généralisé à tous les systèmes physiques ; puisque le langage graphique mis en oeuvre fonctionne par analogie et représente d'une façon unique les phénomènes physiques analogue, quelque soit le domaine physique concerné.

Cela est rendu possible grâce à la notion de causalité (Rosenberg [1987]). Celle-ci caractérise les relations de cause à effet existant dans un système. Elle permet à la fois l'écriture et l'organisation des équations mathématiques associées au modèle bond-graph (Rosenberg [1974]), et la mise en évidence des propriétés structurelles du système dynamique (Suda et Hatanaka [1986], Sueur et Dauphin-Tanguy [1989, 1991.a]). Notre contribution dans ce domaine réside dans la proposition d'une méthode d'affectation automatique de la causalité.

Cependant, il faut noter que la modélisation par bond-graph demande à l'utilisateur un investissement préalable et un apprentissage de l'outil. L'idée primordiale qui nous a amenés à développer ARCHER est de libérer l'utilisateur de cette phase préliminaire

* Voir Paynter [1961], Karnopp et Rosenberg [1983], Breedveld [1984].

et de lui permettre de bénéficier des grands avantages de la démarche bond-graph de façon tout à fait transparente pour lui.

Pour cette raison, nous avons prévu un compilateur avec un analyseur syntaxique et un analyseur sémantique. Il permet de transformer le modèle physique en bond-graph, et de détecter des contradictions par rapport aux lois physiques relatives à la nature des éléments.

L'affectation de la causalité tient une place importante dans la méthodologie bond-graph pour les informations causales qu'elle fournit et la mise en équation du système physique. Nous avons cherché à proposer une nouvelle méthode de l'affectation de la causalité. Elle passe par plusieurs étapes successives, dites étapes de "causalité obligatoire", "causalité préférentielle", et "causalité décisionnelle". Elle repose sur des objets "représentants causaux" pour faire des "déductions causales". A ce niveau des erreurs sémantiques peuvent être décelées.

La "causalité décisionnelle" se base sur des règles statiques (que nous avons définies) pour provoquer les éventuels conflits causaux, le but est de générer des classes d'objets "compétitions causales" dont la résolution s'appuie sur certaines règles dynamiques.

Lorsque le système est de rang maximal (tous les éléments de stockage d'énergie sont indépendants), ces "compétitions causales" vont être directement solvables à partir des règles statiques. Le bond-graph causal correspond à celui que l'on obtient avec les méthodes classiques, telle SCAP (Karnopp et Rosenberg [1975, 1983].

En revanche, lorsqu'une causalité mixte (existence d'éléments de stockage d'énergie dépendants) est obligatoire, l'affectation de la causalité ne se fait pas d'une manière séquentielle. En effet, les règles dynamiques permettent de déterminer la causalité adéquate.

L'autre volet de notre recherche est consacrée à la détermination, à partir du bond-graph, de l'équation d'état sous forme d'expressions formelles. La méthode que nous proposons s'intitule "Méthode des Boucles et Chemins Causaux : MBCC". Elle est graphique et se base sur la règle de Mason, appliquée à certaines parties du bond-graph. Nous la démontrons à partir des règles numériques de Rosenberg.

La MBCC présente l'originalité d'être : pratique (elle se base uniquement sur le parcours et sur l'analyse des chemins causaux et des boucles causales) ; complète

(tous les cas de figure susceptibles d'exister dans un modèle bond-graph, boucles algébriques, causalité dérivée..., sont pris en considération) ; facile à programmer (elle ne génère aucune situation non déterministe et se passe du calcul matriciel) ; rapide (que ce soit en utilisation manuelle ou sur micro-ordinateur, mais aussi par rapport aux méthodes existantes). Au point de vue pratique, la MBCC est parfaitement au point, elle s'est bien prêtée à la philosophie d'Archer.

Dans le cas d'un système non linéaire, nous avons adopté deux stratégies : l'application de la MBCC à des systèmes partiellement non linéaires ou la simplification des expressions mathématiques du modèle d'état pour la simulation.

Cette méthode s'applique également dans le cas des éléments multi-ports.

La dernière partie de notre travail est consacrée à l'étude algorithmique d'Archer et à son analyse informatique.

La validation de nos algorithmes s'est basée sur la résolution des problèmes liés à "l'admissibilité", à "la complexité" et à "la terminaison". En outre, celui de la gestion de la mémoire a tout particulièrement retenu notre attention. Pour franchir les obstacles qui en découlent nous avons eu recours aux techniques de l'intelligence artificielle et aux concepts de la programmation par objet .

Les problèmes, posés par les questions de choix et de décision, qui nécessitent la recherche soit d'heuristiques soit de régles nous ont incité à concevoir, pour certains modules d'Archer, un générateur de connaissances se comportant en système expert.

Nous avons voulu également munir Archer d'un modules qui permet la traduction en langage naturel adapté aux domaines de l'utilisateur et d'un autre qui effectue le calcul formel des expressions mathématiques du modèle.

Pour la partie programmation, nous avons utilisé deux langages (ceci a été possible grâce à la conception modulaire d'Archer) : Turbo Prolog et Turbo Pascal Windows.

Dans un premier chapitre, nous nous consacrons, après un bref rappel des éléments de base des bond-graphs, à la présentation d'Archer et du principe de son fonctionnement. Nous détaillerons ensuite la première phase d'Archer qui, par l'intermédiaire d'un compilateur, génère la représentation symbolique d'un bond-graph à partir de la description d'un système physique.

Au deuxième chapitre, nous proposons une méthode de l'affectation automatique de la causalité qui se fonde sur une analyse qualitative du bond-graph, et des algorithmes pour déterminer toutes les informations nécessaires à l'exploitation de la méthodologie bond-graph.

Le troisième chapitre, traite de la détermination de l'équation d'état par une nouvelle méthode que nous proposons dans le cadre de ce travail et qui est basée sur l'analyse qualitative des relations causales d'un bond-graph.

Un quatrième chapitre est consacré à la spécificité informatique du projet Archer : conception, problèmes rencontrés, solutions éventuelles et particularités.

Une annexe donnera les grandes lignes des algorithmes concernant la détermination des équations mathématiques associées au modèle ainsi que ceux liés à l'analyse structurelle du système étudié.

Chapitre 1

ARCHER

et la Modélisation par Bond-Graph

Introduction

Le concept bond-graph a été introduit par H.M. Paynter [1961] pour mettre en évidence les échanges de puissance entre les éléments d'un système physique.

Cette idée a été formalisée par D. Karnopp et R.C. Rosenberg [1975, 1983] et a permis son application dans de nombreux domaines. Depuis, les bond-graphs ont été généralisés par P. C. Breedveld [1984].

Par conséquent, plusieurs types de systèmes dynamiques ont pu être modélisés et simulés. P. C. Breedveld et al. [1991] ont présenté une large bibliographie.

Dans une première partie, nous présenterons quelques éléments de la méthodologie bond-graph. Nous nous consacrerons ensuite à exposer d'une manière générale le projet Archer.

Nous montrerons, dans une troisième partie, la première étape d'Archer qui consiste à générer, sous une forme symbolique, le bond-graph associé au modèle physique, à partir de la description de ce dernier.

I. Modélisation par les bond-graphs

Nous rappelerons ici de façon très succinte les éléments de base.

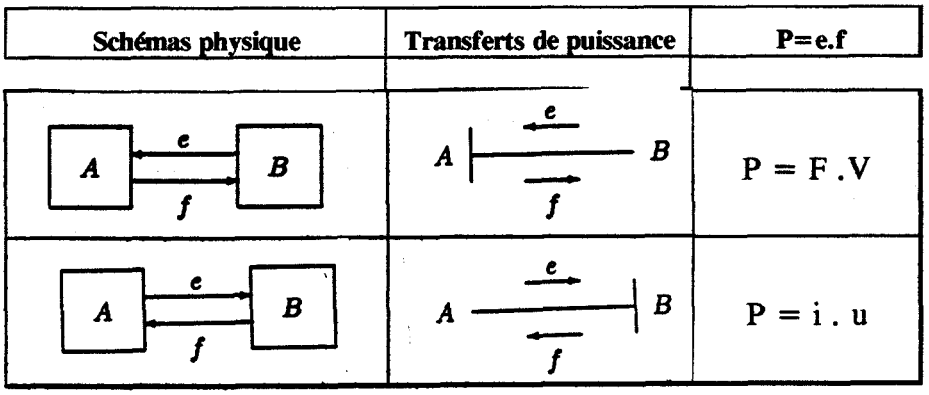
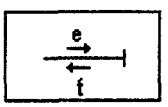


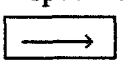
figure 1.1 : exemples de représentation de transferts de puissance

Un bond-graph est une représentation graphique des mécanismes d'interaction, de dissipation et de stockage d'énergie d'un système dynamique. L'interconnexion entre deux éléments d'un système échangeant de la puissance est représentée par un lien. A ce dernier on associe deux variables de puissance, effort (**e**) et flux (**f**). Leur produit exprime la puissance (**P = e.f**), portée par le lien. Le sens positif de celle-ci est indiqué par une demi-flèche. La figure 1.1 montre deux exemples de liens, quant au tableau 1.2 il présente leur signification dans divers domaines de la physique.

Les variables p et q représentent respectivement l'impulsion et le déplacement.

Un trait causal peut être affecté, ou l'autre des extrémités du lien est toujours représenté vers le trait 

d'une manière, exclusive, à l'une selon des règles précises. L'effort causal qui caractérise la relation de cause à effet existant entre deux éléments du système. La causalité est totalement indépendante de l'orientation du flux de puissance (sens d'une demi-flèche).

Quand la puissance d'un lien est nulle ou négligeable, à cause de la présence dans le système d'un dispositif spécifique (amplificateur, capteur, ...), le lien est représenté par un simple signal : 

Les éléments bond-graphs peuvent être classés de la manière suivante :
-éléments actifs : sources d'énergie S_e, S_f.
-éléments passifs de stockage d'énergie : élément capacitif **C**, élément inertiel **I**.
-éléments dissipatifs d'énergie : élément résistif **R**
-éléments de Jonction : **0**, **1**, **TF**, **GY**

SYSTEME PHYSIQUE	FLUX Unités	EFFORT Unités	DEPLACEMENT Unités	IMPULSION Unités
GENERALISE	f(t)	e(t)	$q(t)=\int f(t)\,dt$	$p(t)=\int e(t)\,dt$
MECANIQUE EN TRANSLATION	V, vitesse linéaire m / s	F, force N	x, distance m	P, impulsion N s
MECANIQUE EN ROTATION	ω, vitesse angulaire rad / s	τ, couple N m	θ, angle radians	P, impulsion angulaire N m s
ELECTRIQUE	i, courant A	e, tension Volt	q, charge électrique Coulomb	λ, flux Weber
MAGNETIQUE	Φ_B, dérivée du flux magnétique Volt	fmm, force magnétomotrice A	Φ, flux magnétique Weber	Ne se définit pas
FLUIDIQUE	Q, débit m^3/s	P, pression Pa	V, volume m^3	Γ, impulsion de pression Pa s
CHIMIQUE	J, flux molaire moles / s	μ, potentiel chimique J/mole	M, masse molaire moles	Ne se définit pas
THERMIQUE	q', flux de chaleur J / s	T, température °K	q, énergie calorifique J	Ne se définit pas
THERMODYNAMIQUE	S', flux d'entropie J / °K s	T, température °K	S, entropie J / °K	Ne se définit pas
BIOLOGIQUE (1)	J, flux molaire moles / s	μ, potentiel chimique J/mole	M, masse molaire moles	Ne se définit pas
PHYSIOLOGIQUE (2) (processus de diffusion)	J_k, flux de potassium moles / s	$\Delta\mu_k$, potentiel chimique J/mole	n_k, masse molaire moles	Ne se définit pas
ECOLOGIQUE et AGRICOLE (modèle Masse-Energie) (3)	γ, flux de matière Kg / s	ξ, énergie spécifique J/Kg	M, matière Kg	E, énergie J
ECONOMIQUE	Sf, taux de flux de marchandises	e, prix unitaire des marchandises unité monétaire	q, inventaire	λ, impulsion économique unité monétaire par unité de temps
ACOUSTIQUE	da, déplacement d'air m^3/s	pa, pression d'air N/m^2	va, volume d'air m^3	ia, impulsion de pression d'air N/m^2 s

(1) Ce type de système peut être traité par analogie avec des systèmes mécaniques.

(2) Ce type de système peut aussi être représenté par le processus du métabolisme ATP ou le processus de synthèse des protéines.

(3) Ce type de système peut être traité par analogie avec des systèmes mécaniques, fluidiques, thermiques et thermodynamiques.

tableau 1.1

Le tableau 1.2 énumère les propriétés des éléments de base.

LES ÉLÉMENTS DE BASE BOND-GRAPH ET LEURS PROPRIÉTÉS.

Eléments bond-graph	Symboles	loi générique	Exemples physiques	Causalités
Variables de puissance	effort e flux f		force, couple, tension, pression... vitesse, vit. ang., courant, débit vol....	
Variables d'énergie	moment p déplacement q	$p=\int e\,dt$ $q=\int f\,dt$	impulsion, flux (self),... déplacement, charge, volume...	
Eléments actifs (sources)	$Se \xrightarrow{e}$ $Sf \xrightarrow[f]{}$	e indépendant de f f indépendant de e	pesanteur, générateur de tension,... générateur de courant, pompe,...	$Se \xrightarrow{e}$, e donné $Sf \vdash\!\!\!\rightarrow$, f donné
Eléments passifs 1-port	$\xrightarrow[f]{e} R$	$\Phi_R(e,f)=0$	amortisseur, résistance, restriction hydraulique, frottement...	$\vdash\!\!\!\rightarrow R \quad e=F_R(f) \quad (e=Rf)$ $\rightarrow\!\vert R \quad f=F_R^{-1}(e) \quad (f=1/Re)$
	$\xrightarrow[f=\dot{q}]{e} C$	$\Phi_C(e,q)=0$	ressort, condensateur, réservoir, élasticité,...	$\vdash\!\!\!\rightarrow C \quad e=F_C(q) \quad (e=q/C)$ $\rightarrow\!\vert C \quad q=F_C^{-1}(e) \quad (q=Ce)$
	$\xrightarrow[f]{e=\dot{p}} I$	$\Phi_I(p,f)=0$	masse, inertie, self,...	$\rightarrow\!\vert I \quad f=F_I(p) \quad (f=p/I)$ $\vdash\!\!\!\rightarrow I \quad p=F_I^{-1}(f) \quad (p=If)$

tableau 1.2 : propriétés des éléments de base d'un bond-graph

Quant à la figure 1.2, elle indique les relations caractérisant les jonctions.

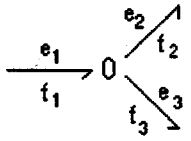
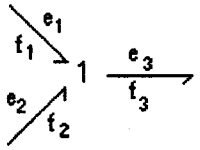
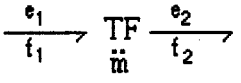
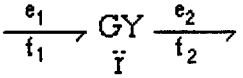
			
$e_1 = e_2 = e_3$ $f_1 - f_2 - f_3 = 0$	$f_1 = f_2 = f_3$ $e_1 + e_2 - e_3 = 0$	$e_1 = m\, e_2$ $f_2 = m\, f_1$	$e_1 = r\, f_2$ $e_2 = r\, f_1$

figure 1.2 : relations caractérisant les jonctions

Le tableau suivant présente deux exemples de systèmes physiques et leur bond-graph associé.

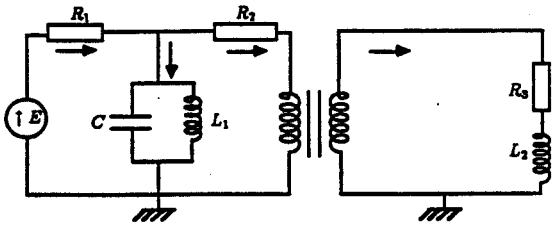
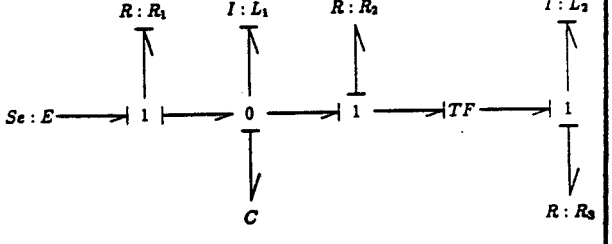
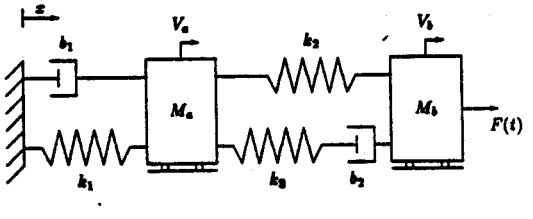
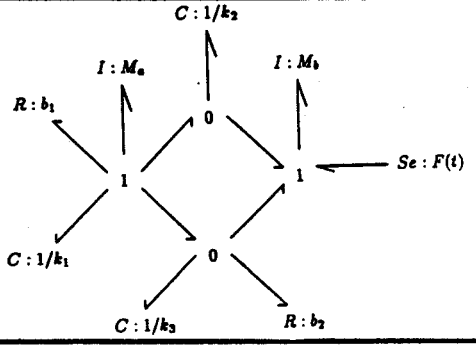
Système physique	son bond-graph causal
	
	

figure 1.3

II. Le projet Archer

Archer est un logiciel d'aide à la modélisation et à l'analyse des systèmes dynamiques. Il utilise la méthodologie bond-graph et les techniques de l'intelligence artificielle, et possède quelques concepts de la programmation orientée objet.

Archer est aussi un produit informatique ayant le profil et les caractéristiques des logiciels modernes : convivialité, réponse en temps réel, traitement en langage naturel....

Pour modéliser et simuler les systèmes physiques à partir des bond-graphs, d'autres produits existent : ENPORT (R. C. Rosenberg [1974, 1980, 1985, 1987]) - CAMP (J. J. Granda [1982]) - CAMAS (J. F. Broenink [1990])... On trouve dans J. Montbrun et al. [1991] une présentation globale de tous les produits utilisant la méthodologie bond-graph.

Chacun de ces produits possède ses propres caractéristiques et cible un public ou un domaine bien particulier. La théorie des bond-graphs s'enrichie d'année en année en méthodes, en applications et en théories (une études bibliographique a été réalisée par P. C. Breedveld et al. [1991]). Cependant, aucun logiciel ne peut prétendre couvrir tous les domaines. En plus chacun possède des particularités informatiques spécifiques.

Cela justifie le développement d'un nouveau logiciel "ARCHER" qui se consacre plutôt à des problèmes de modélisation et d'analyse structurelle pour optimiser la construction de modèles valides. D'autre part, sa grande particularité est qu'il est conçu pour être utilisé également par des non bond-graphistes.

L'élaboration d'Archer a débuté par une étude de faisabilité qui a donné lieu à quelques publications.

Depuis, le produit s'est enrichi et compliqué. L'expertise des chercheurs du laboratoire dans le domaine de la modélisation est progressivement intégrée à ARCHER, ce qui est tout à fait réalisable de part la structure modulaire du logiciel.

II.1 Le processus d'Archer

Nous consacrerons cette partie à l'étude des différentes phases suivies par Archer pour atteindre ses objectifs. Nous présenterons les étapes de son processus sous forme d'un automate à états.

II.1.1 Automate simulant le processus d'Archer

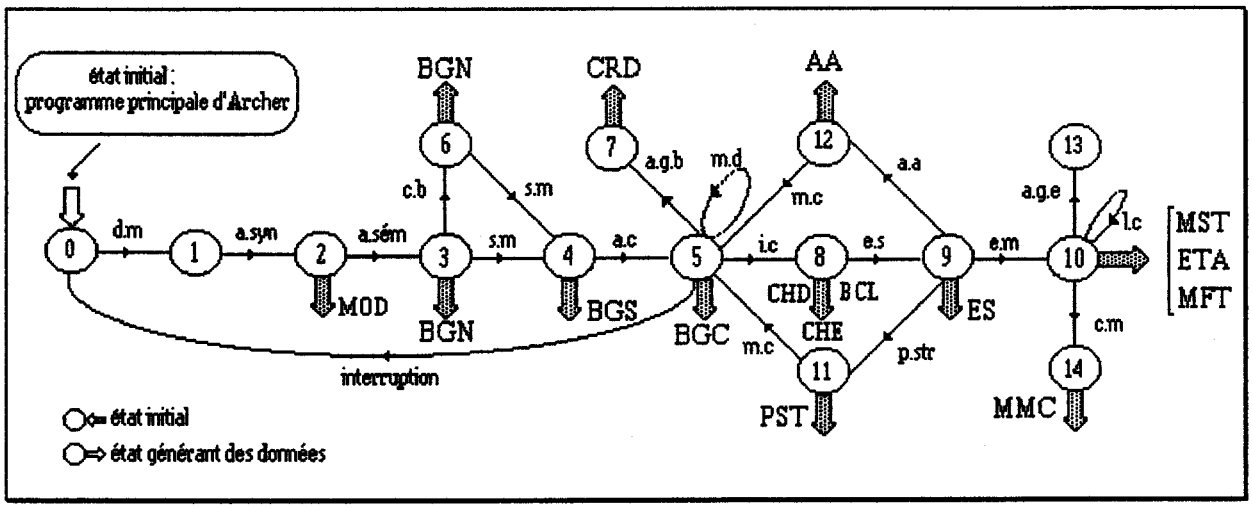


figure 1.4 : Automate simulant le processus d'Archer

Avec :

d.m : description du modèle physique (mode texte, mode graphique)

a.syn : analyse syntaxique des modèles physiques

a.sém : analyse sémantique

a.c : affectation de la causalité

c.b : couplage entre bond-graphs

a.g.b : affichage graphique du bond-graph

i.c : détermination des informations causales

m.d : modification des modèles

(analyseur sémantique des modèles physiques)

e.s : détermination des entrées-sorties

m.s : détermination de la matrice de structure

l.c : détermination d'une liaison causale indirecte

e.m : détermination des équations mathématiques

a.g.e : affichage graphique des équations.

a.str : analyse structurelle du modèle

a.a : autres analyses

m.c : manipulation causale

c.m : conversion des opérations mathématiques en données exploitables par les logiciels de simulation (ACSL, TUTSIM, ...)

interruption : erreur sémantique de la description du système physique et prise en main de sa correction par l'utilisateur.

Chaque état terminal q_F produit un ensemble de données qui vont enrichir la base de connaissances d'Archer en faits exploitables par les états qui succèdent à q_F ; et qui sont enregistrées dans des bases de données. Leurs noms sont composés du même préfixe que celui du modèle et par un suffixe indiquant le type des données générées à partir de q_F. Ces suffixes ont les significations suivantes :

MOD : données du modèle physique
BGN : données du modèle bond-graph non simplifié
BGS : données du modèle bond-graph simplifié
BGC : données du modèle bond-graph causal
CRD : coordonnées du bond-graph par rapport à l'écran
CHE : données des chemins causaux élémentaires
CHD : données des chemin causaux directs
BCL : données des boucles causales
ES : données des entrées-sorties du système
BCD : données des boucles causales disjointes
MST : données des matrices de structure
MFT : données de matrice des fonctions de transfert
ETA : données des matrices d'état
PST : données des propriétés structurelles
AA : autres informations diverses (bond-graph réciproque ...)
MMC : modèle mathématique converti

II.1.2 Algorithme associé à l'automate précédent

Cet algorithme représente le cas général d'une session d'Archer. Ses lignes, que l'on peut retracer comme ci-dessous, correspondent aux actions associées aux états de l'automate :

Décrire le modèle physique (en mode texte ou graphique) ;
Compiler le modèle physique (analyse syntaxique et analyse sémantique)
si couplage avec d'autres bond-graphs, alors déterminer un modèle bond-graph unique non simplifié ;
Simplifier le modèle bond-graph ;
Affecter la causalité et éditer éventuellement le modèle bond-graph sur un périphérique de sortie ;
Déterminer les informations causales ;
Déterminer les entrées-sorties ;
Déterminer les équations mathématiques et les éditer ou les préparer pour la simulation ;
Analyser la structure du modèle bond-graph ou Procéder à d'autres manipulations de ce type.

II.1.3 Commentaire de l'automate à états

L'état initial correspond au programme principal d'Archer. Il se charge de la gestion de son environnement. Il est très explicite et permet d'avoir toutes les informations concernant les données qui ont été générées.

L'état 1 correspond à un "interface utilisateur" où l'on peut décrire le modèle physique étudié sous forme de langage simple. On lance alors une compilation du modèle qui commence par une analyse syntaxique (état 2) et permet de corriger toutes les erreurs (frappe, terme ne se trouvant pas dans un dictionnaire prédéfini et extensible, etc...).

La compilation se poursuit avec l'analyse sémantique (état 3) pour générer le modèle bond-graph non simplifié. Ce dernier possède la même architecture que le modèle physique correspondant.

Aussi dégageons-nous, pendant la phase de sa simplification (état 4), des informations aidant à le dessiner correctement sur un périphérique de sortie (écran, imprimante, table traçante) tout en respectant les symétries (*) contenues dans le système physique (ex : alimentation triphasée d'un moteur).

Avec l'analyseur sémantique, l'affectation de la causalité (état 5) constitue le noyau d'Archer. En effet, elle permet d'affecter la causalité en détectant les anomalies du système physique, d'éviter les problèmes susceptibles d'exister dans les modèles mathématiques (boucles algébriques), de tenir compte des paramètres associés aux éléments (lois régissant le comportement des éléments...).

Cette étape est très importante car l'affectation de la causalité est constamment sollicitée par les méthodes de l'analyse structurelle (commandabilité, observabilité, stabilité), et par d'autres méthodes nécessitant des manipulations causales (bond-graph réciproque, réduction des modèles, estimation des dynamiques...)(**).

---

* Le respect des symétries constitue la phase la plus délicate du dessin automatique du bond-graph à partir de la description du système étudié.

** G. Dauphin-Tanguy et al. [1983] ; G. Dauphin-Tanguy et al. [1985] ; C. Sueur et G. Dauphin-Tanguy [1991] ; C. Sueur [1990].

A partir de la causalité (état 5), on détermine les informations causales (état 8) nécessaires à établir les modèles mathématiques sous forme d'expressions formelles. Les liaisons causales, directes ou indirectes, seront utilisées pendant la détermination des équations du modèle et pendant l'analyse structurelle.

La détermination des entrées-sorties (état 9) se fait par l'intermédiaire d'un interface utilisateur qui offre la possibilité de choisir l'effet de certaines grandeurs physiques sur les éléments. Par exemple pour un système mécanique, il peut choisir parmi la vitesse, le déplacement, la force et le moment ; et pour un système électrique ou électronique, il peut choisir parmi la tension, la charge, le flux et l'intensité.

Remarques 1.1 :

1. La communication avec l'utilisateur se fait en langage naturel propre au domaine étudié. Sa démarche se fait en trois étapes : l'interprétation des caractéristiques des systèmes pour déterminer les grandeurs physiques sous forme de langage naturel, la gestion du dialogue et la réinterprétation des résultats sous une forme symbolique.

2. La détermination des entrées-sorties peut se faire, à partir des données d'un modèle bond-graph simplifié, à n'importe quel niveau. Nous l'avons volontairement placée à l'état 9, parce que les états qui viennent ensuite auront besoin simultanément des informations causales et des données représentant les entrées-sorties.

Ces données permettent de déterminer (état 10) les matrices de transfert dans le cas linéaire, et l'équation d'état du système linéaire (ou partiellement non linéaire) sous forme d'expressions formelles(*).

Les phases d'analyse (états 11 et 12) peuvent se faire sans difficulté par la manipulation de la causalité. Cette opération se réalise aisément parce que la méthode d'affectation de la causalité a été, entre autres, conçue à cet effet.

Les modules correspondant aux états 6, 7 et 13 font l'objet d'autres travaux en cours (pour l'état 6, voir DEA (P. Remy [1990]) et pour les états 7 et 13, voir (R. Bouayad [1991])). Quant à celui de l'état 14, des extensions sont prévues sur une première version consacrée à convertir les données d'Archer sous formes de données d'ACSL.

---

* voir méthode proposée dans le cadre de ce travail au chapitre 3.

II.2 Représentation symbolique des données d'Archer

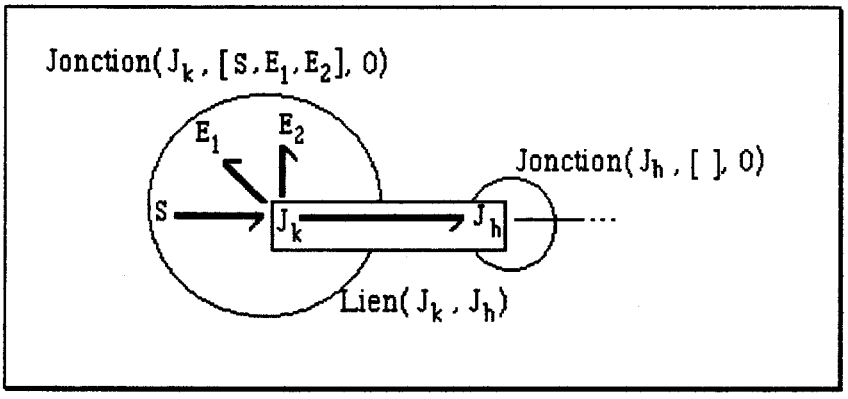


figure 1.5 : forme des représentations symboliques des éléments bond-graph bond-graph

Une jonction est représenté par trois paramètres :

- type de la jonction J de type 0, 1, TF ou GY. k est l'identificateur de la jonction, il est utile pour le traitement informatique.
- liste des éléments qui lui y sont attachés avec S une source de type S_e ou S_f et E un élément de type I, C ou R.
- un entier égal à 0, 1 ou 2 qui correspond à une convention permettant d'identifier le type du "bond" -simple ou multiple- liant deux jonctions. Le 0 indique que J est une simple jonction. Nous verrons plus bas la définition des deux autres et pourquoi nous avons associé cette convention à la jonction plutôt qu'au lien.

Le symbole "lien" correspond uniquement aux liens entre jonctions.
Cette séparation entre les élément un port (représentés dans une liste d'une jonction ou plus exactement) et les éléments n-ports a été déterminée suite ou différentes méthodes graphiques que nous utilisons. D'autant plus que cette séparation répond bien à la philosophie bond-graph car il permet de mettre en évidence la structure à partir du prédicat "lien(.)" et les éléments par le prédicats "Jonction(.)".

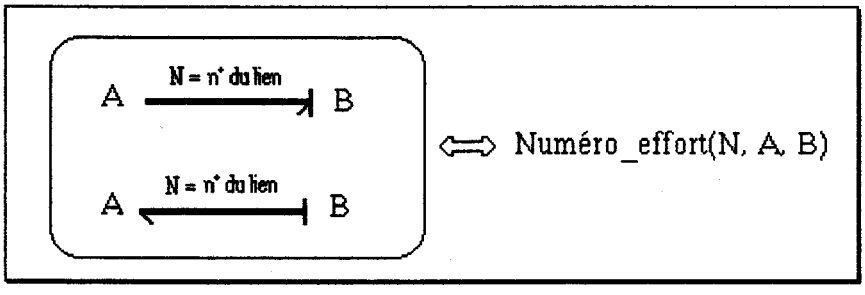
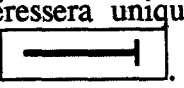


figure 1.6 : prédicat représentant la causalité

Dorénavant, lorsqu'on s'intéressera uniquement à la causalité nous négligerons la demi-flèche sur nos dessins :  .

Les autres, représentations symboliques utilisées par Archer seront exprimés au chapitre 2 et dans l'annexe A.

cas particuliers :

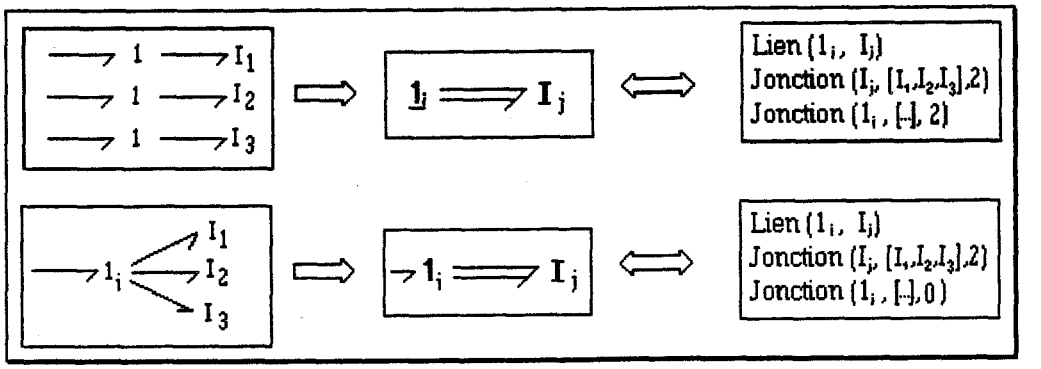


figure 1.7 : transformation de liens simple en multi-liens et leur représentations symbolique.

Le troisième paramètre du prédicat "jonction" servira ici à identifier le type du liens et celui de la jonction. Cela permettra d'identifier pendant les traitements les éléments n-ports de type I, C, R, IC et IR. Ces derniers caractérisent une relation matricielle entre les variables qui lui y sont attachées. Cette information est utilisée également par le module du dessin automatique du bond-graph. Les conventions suivantes sont alors appliqués : Si la somme des "3° paramètres" des jonctions d'un lien est supérieure ou égale à 2 alors le lien sera représenté par un multi-bond.

Si la somme des "3° paramètres" de 3 jonctions consécutives est égale à 6 alors celle du milieu sera souligné (conventions utilisée par les bond-graphiste).

Le cas de La figure 1. se produit souvent pendant l'affectation de la causalité pour résoudre certains conflits causaux. La liste associés à la jonction "champ" est représentée par les numéros des liens qui lui y sont attachés. Pour identifier ce cas par rapport à ceux de la figure 1. , le 3° paramètre prendra ici la valeur 1.

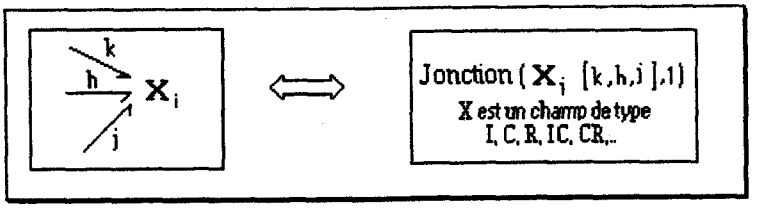


figure 1.8 : représentation symbolique d'un champs

Remarque:

Dans un but de simplification des notations, nous avons choisi de représenter un élément de type I, de paramètre I_1 par la seule valeur de son paramètre. La notation conventionnelle (I : I_1) est bien entendu utilisée lors de l'affichage du bond-graph (état 7).

II.3 Eléments de base pour décrire un modèle (*)

La description d'un modèle se base sur les notions de **série** et de **parallélisme**. Elles aideront à déterminer les éléments qui ont le même effort et ceux qui ont le même flux. Les premiers seront alors attachés à des jonctions-0, quant aux seconds ils seront liés à une jonction-1.

Pour indiquer que deux éléments physiques X et Y sont en série (respectivement en parallèle) on écrit : X + Y (respectivement X / Y).

Les éléments sont désignés par des termes associés à un domaine physique et commençant obligatoirement par une lettre majuscule. Ceci permettra de les identifier par rapport aux points de connexions qui seront représentés par une lettre minuscule suivie d'un nombre. Un terme est composé d'une ou plusieurs lettres suivies d'un identificateur entier, par exemple : Capacité1, Capa5 ou tout simplement C3.

La première partie du terme doit exister dans un dictionnaire dont les éléments sont des triplets indiquant : le domaine, l'élément et son symbole universel. Par exemple ("m", "Ressort" "k") avec "m" pour mécanique et k symbole de la raideur d'un ressort. Si le mot n'existe pas, il faut construire son triplet et l'ajouter.

---

* Une partie de ce travail et de la compilation qui va suivre ont fait l'objet d'un mémoire de D.E.A (D. Ringot [1990]). Nous l'avons reprise depuis pour lui donner une conception orientée objet afin de la généraliser à d'autres domaines comme l'électronique et de faire quelques couplage .

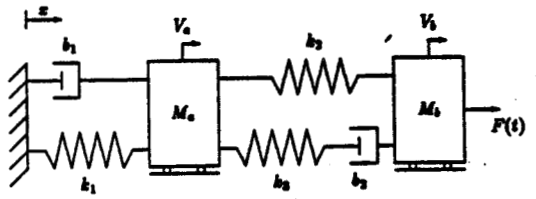
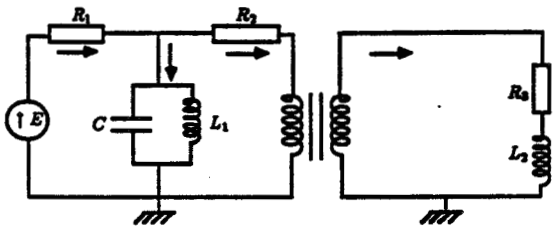
Modèle physique	Sa description dans Archer
	**mécanique** **bâti + Amortisseur1 / Ressort1 + Ma + Ressort2 / (Ressort2 + A2)+Mb F1(Mb)** % A2 = Amortisseur2 { on peut adapter le code des termes}
	**électrique** **m1+E1+R1+a1+R+TFp1+m1 a1+C1/L1+m1 m2+R3+L2+TFs1+m2** { . a1 est un point de connexions . pour les transformateurs on doit les séparés en deux partie}

figure 1.9 : description de deux modèles physiques

Les '{', '}' et '%' du tableau correspondent respectivement à un commentaire sur plusieurs lignes et un commentaire sur une ligne.

III. Compilation des modèles physiques

Archer possède un compilateur basé sur un analyseur syntaxique et un analyseur sémantique. Son but consiste à vérifier des anomalies dans la description d'un modèle physique et à générer un code représentant de façon symbolique un bond-graph (sous forme de prédicats de la database de Turbo Prolog : "**lien** (_,_) et **Jonction** (_,_,_)").

III.1 Analyseur syntaxique

Son principe consiste à vérifier la validité syntaxique des mots entrés par l'utilisateur. Pour cela, il se base sur la grammaire qui possède :

- un alphabet défini par :

X = {**a, ..., z, A, ..., Z, 0, ..., 9, +, /, %, (,), {, }, :, !, ;, ,**}

- une opération de concaténation '*'qui pe
- un automate à états représenté par :

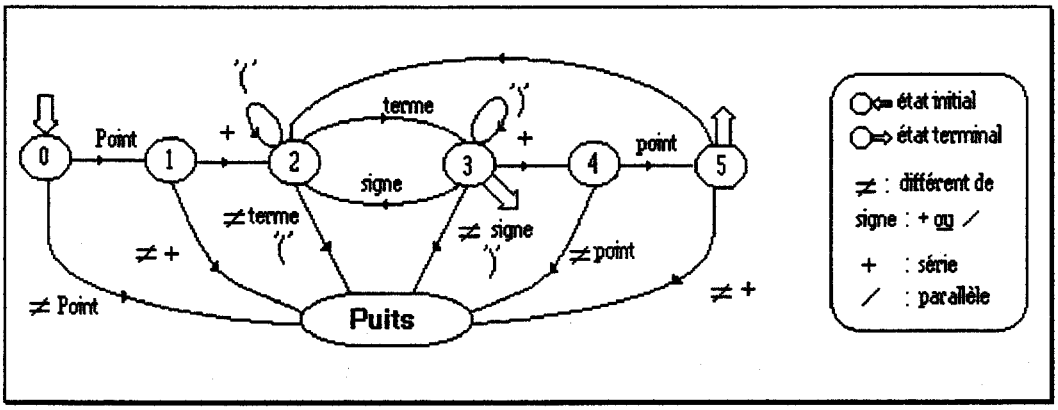


figure 1.10 : automate associé à l'analyse syntaxique de la description du modèle physique.

avec : q_0 : état initial, $Q_F = \{q_3, q_5\}$: ensemble des états terminaux

Cet automate reconnaît un langage dont les mots constituent une suite d'éléments de X. Les mots de X* (ensemble de toutes les concaténations de X) appartenant à ce langage vérifient la syntaxe représentée ici sous la forme classique de Backus :

<mot> ::= <point> <signe> <mot> / <point>
<mot> ::= <terme> <parenth> <mot> / <terme> <signe> <mot> / <terme>

<point> ::= <minuscule> <point>
<point> ::= <nombre>

<terme> ::= <majuscule> <terme2> / '!' <code> <identificateur>
<terme2> ::= <lettre> <terme2> / <lettre> <nombre>
<terme2> ::= <lettre> <nombre>

<signe> ::= '/' / '+'

<parenth> ::= '(' <parenth> / ')' <parenth>
<parenth> ::= ')' / '('

<nombre> ::= <chiffre> <nombre> / <chiffre>
<chiffre> ::= '0' /... / '9'

<lettre> : ::= <minuscule> / <majuscule>
<minuscule> ::= 'a'/ ... / 'z'
<majuscule> ::= 'A'/ ... / 'Z'

<code> ::= 'R' / 'D'
<identificateur> ::= 'e'/ 'E'/ 'm'/ 'H'

Le vocable de la description d'un modèle se base sur les classes suivantes :

- le point représente l'existence d'une connexion dans le modèle
- le signe représente le type de la connexion (série ou parallèle)
- le terme représente un élément physique ou à un élément de couplage entre deux domaines différents.

Le couplage entre deux domaines est indiqué par un '!'. Le code détermine l'élément 2-ports qui couple deux domaines différents. Quant à l'identificateur il permet de connaitre le nouveau domaine, il nous aidera à connaître le type de la jonction qu'il faudra associer aux éléments en série respectivement en parallèle (voir plus bas).

Les codes et les identificateurs ont la signification suivante :
R : utilisation de règles de couplage
D : couplage direct sans intermédiaire
e : électrique
E : Electronique
m : mécanique
H : Hydraulique

Remarquons que l'automate précédent est non déterministe : à l'état 3 le signe '+' conduira à l'état 4 ou à l'état 2.

Nous n'avons pas cherché à le rendre déterministe pour ne pas alourdir le traitement informatique. Cependant cette situation est facilement solvable, il suffit pour cela de tester l'élément qui vient après le signe "+". Cette opération n'est pas coûteuse parce qu'elle est 2-décidable : pour sortir d'une situation non déterministe, on peut se décider au bout de 2 lectures successives.

Les états correspondent à des actions de lecture. A la rencontre d'une erreur syntaxique, on aboutit à un état "**puits**" qui a pour rôle de déterminer la nature de l'erreur, de la signaler par un message et de se positionner sur celle-ci.

Avant de présenter les différents cas d'erreurs, nous allons montrer la spécificité d'un fichier associé à la description du modèle physique.

Le programme se compose de trois parties : les deux premières sont obligatoires, par contre la troisième est optionnelle. Il offre la possibilité de placer des commentaires de deux types différents.

Partie titre : est écrite sur une seule ligne. Elle correspond au domaine principal du système étudié.
Partie description : peut contenir plusieurs lignes décrivant le système.
Partie particularités : permet d'exprimer les particularités de certains éléments. Elle commence toujours par le mot NL suivi de ':', ou uniquement par ':'.

Pour la "partie description", nous avons présenté l'automate qui permet de gérer sa syntaxe. Quant à la "partie particularités", ses éléments sont des triplets qui ont la forme suivante :

(élément physique, une convention = 0 ou 1, une fonction ou un souligné).

L'élément physique doit être déjà rencontré dans la "partie description", la convention 0 (respectivement 1) indique que le flux est fonction de l'effort (respectivement effort est fonction du flux), la fonction correspond à n'importe quelle fonction mathématique. Lorsqu'elle est représentée par un souligné, elle sera alors symbolisée par "Φ" qui représente une fonction non linéaire quelconque.

Par exemple pour la "partie particularité", on aura : NL : (L_1, 0, _) ; (R_2, 1, $\sin^2$) ;

Et pour les triplets, on aura :

(X, 0, _) correspondra à flux = Φ_X(effort) avec X = L_1
(Y, 1, _) (L1, 0, _) correspondra à effort=Φ_Y(flux)
(Z, 1, $\sin^2$) correspondra à effort=$\sin^2$(flux) avec Z=R_2

L'automate associé à la vérification syntaxique de la "partie particularités" est le suivant :

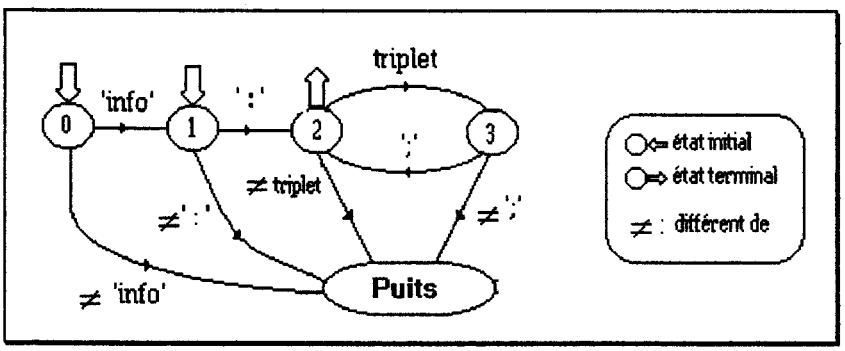


figure 1.10 : Automate reconnaissant le langage de la "partie particularités"

III.2 Traitement des erreurs syntaxiques

Nous allons retracer les différentes erreurs syntaxiques susceptibles de se produire pendant l'écriture du fichier décrivant le modèle physique.

La première erreur peut correspondre à l'absence de la "partie titre" ou à une erreur de frappe dans celle-ci. Les erreurs correspondant à la "partie description" sont celles qui aboutissent à l'état "puits" de l'automate associé, ou celles ne répondant pas correctement à la syntaxe d'un point ou d'un terme. Celles de la "partie particularités" correspondent également à l'état "puits" de son automate ou à l'absence d'un élément physique de l'un de ses triplets dans la "partie description" .

Lorsqu'une erreur est détectée par l'analyseur syntaxique, celui-ci place le curseur sur le mot erroné et envoie un message indiquant le type de l'erreur. Par exemple :

Aucun domaine n'est précisé
Domaine non reconnu
Il manque n ')'
')' mal placée
Le terme X n'appartient pas au domaine courant D
Le terme X a été rencontré précédemment
Deux signes consécutifs rencontrés
Une ligne ne doit pas commencer par un signe
Manque d'un terme entre deux parenthèses
consécutives différentes sont rencontrées
Le deuxième terme d'un triplet est égal à 0 ou 1
...

L'utilisateur a la possibilité d'ajouter ses propres mots pour décrire un élément. Dans ce cas, il doit indiquer son système et le symbole universel le représentant. Il doit déterminer l'élément bond-graph associé à ce dernier. Le triplet d'informations (système physique, élément ajouté, symbole universel) et le couple (symbole universel, élément bond-graph) seront insérés dans deux listes utilisées par l'analyseur syntaxique. La première est utile pour savoir si un élément appartient au domaine concerné et la seconde permettra de générer, pour chaque élément, une information représentée dans Archer par le prédicat :

conv(un indicateur du système, élément physique indexé par son numéro identificateur, élément bond-graph associé indexé par un élément qui peut être différent du précédent).

L'indicateur de système est un code permettant d'identifier le système considéré.
L'indice de l'élément bond-graph ne correspond pas toujours à celui de l'élément physique qui lui est associé, car le système peut contenir deux éléments de nature physique différente (par exemple une résistance et un amortisseur) qui sont tous les deux représentés par l'élément bond-graph "R".

Cette information sera d'une grande utilité pendant les interfaces utilisateur (détermination des entrées-sorties, affectation de la causalité et analyse) et lors de l'affichage des messages et des résultats.

III.3 Analyseur sémantique

Un certain nombre de procédures (Karnopp et Rosenberg [1975, 1983], P. Borne et al. [1991]) permettent de construire graphiquement un bond-graph à partir du schéma représentant le modèle physique.

En ce qui concerne notre analyseur sémantique, il suit la procédure :

-Définir le sens positif de l'orientation de la puissance (on impose à l'utilisateur de décrire son système en fixant un sens positif de transfert d'énergie).

-Attacher, en fonction du domaine étudié, les éléments en série à une jonction-0 ou une jonction-1. Ainsi par exemple pour les systèmes électrique, électronique et hydraulique les éléments en série seront attachés à une jonction-1. Par contre, pour un système mécanique, ils seront attachés à une jonction-0.

En règle générale, si les éléments en série correspondent à des noeuds représentant l'effort, on les attache à des jonction-0. S'ils correspondent à des noeuds caractérisant le flux, ils seront attachés à des jonction-1.

-Placer entre deux jonctions précédentes leurs jonctions duales.

Remarques 1.2 :
1. Pour que le programme puisse traiter n'importe quel système dynamique (voir entre autres, le tableau 1.1), il faut suivre la procédure :

.Définir ses éléments et leur symbole respectif (construction de la liste des triplets précités).
.Associer à chaque symbole son représentant bond-graph.
.Déterminer la convention 0 ou 1 permettant d'appliquer le deuxième point du principe précédent (cette convention = 0 si les éléments en série doivent être attachés à des jonction-0, à une jonction-1 sinon).

2. L'implantation sur machine de cette procédure ne présente aucune difficulté, bien au contraire, on peut rendre son programme très intéressant en lui donnant la possibilité de générer lui même les conventions et les éléments bond-graph associés aux symboles. En effet, cette opération peut se faire par identification avec des systèmes existant ayant le même comportement physique que le système ajouté.

3. Le fait d'associer à chaque système une convention, aidant à déterminer la nature des jonctions symbolisant les noeuds d'effort ou de flux, nous a permis de modéliser des systèmes dynamiques pluridisciplinaires. Ce couplage se fait toujours au niveau du modèle physique. Pour le moment, nous ne traitons pas de blocs préexistants ou prédéfinis, toute la description des systèmes physiques doit se trouver dans le même fichier. Il est prévu de construire une bibliothèque de blocs, leur couplage dépendra alors de certaines règles(*). Celles-ci détermineront l'élément bond-graph 2-port qui va coupler deux domaines différents.

Eventuellement une "partie déclaration" pourrait être ajoutée dans le fichier associé au modèle physique pour permettre de faire des descriptions bien réparties (comme celles rencontrées dans le code des programmes procéduraux).

4. Le compilateur possède l'intérêt de ne pas dépendre des données traitées et d'être réutilisable. Ces deux propriétés font partie des concepts d'une programmation par objet. Les objets correspondent ici aux systèmes physiques composant le système dynamique étudié.

Dans notre analyse, les points sont traités comme des éléments particuliers, ils seront également attachés à des jonctions. C'est pourquoi le modèle bond-graph non simplifié peut être projeté sur le modèle physique.

---

* le couplage des modèles fait l'objet d'une thèse (P. Remy [1990]).

Pour simplifier le modèle bond-graph généré par le compilateur, on applique la procédure suivante :

-Eliminer les jonctions attachées aux points d'absorption de la puissance, (ces points sont par exemple : "batis" pour un système mécanique et "masses" pour un système électrique).

-Appliquer les règles suivantes :

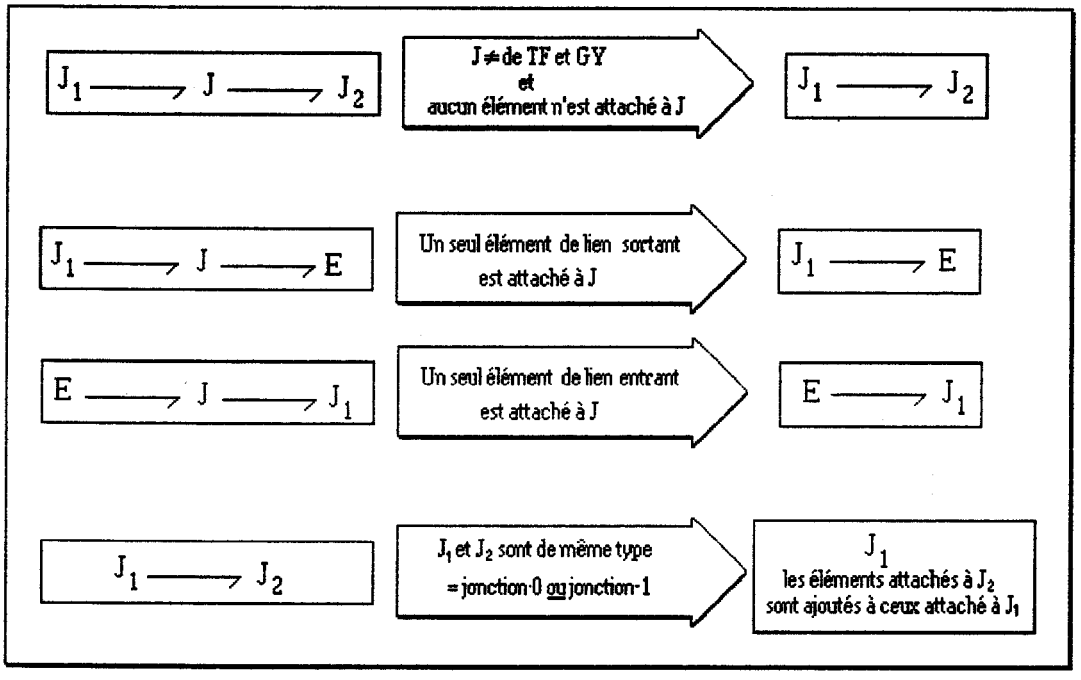


figure 1.11 : règle de simplification

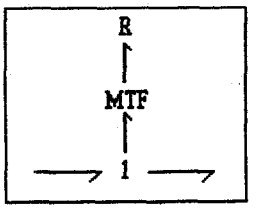
Dans le cadre d'Archer, nous procédons à une simplification permettant dans certains cas (voir figure 1.7) de transformer des liens simples en multi-bonds (multi-liens).

Nous terminerons ce chapitre par les remarques suivantes :

1. Certains systèmes peuvent être symbolisés par des multi-bonds directement par le compilateur. Cette opération dépend de l'élément bond-graph associé aux composants du système physique. En effet comme nous l'avons signalé dans la remarque 1.2.3 , il suffit d'indiquer le nombre de ports qu'un élément bond-graph doit posséder lors de la définition d'un nouveau système.

2. Certains éléments physiques peuvent avoir plusieurs états. Par exemple en électronique de puissance les commutateurs : Switchs (Interrupteurs), Diodes,

Transistors et Thyristors peuvent passants ou bloqués. Ils seront alors représentés par un "MTF" et un élément R :



Dans ce cas les "MTF" sont des jonctions-TF dont le module un booléen.

3. Nous avons présenté l'analyse d'une description du modèle en mode texte. Cependant la nouvelle version d'Archer, que nous développons sous Windows, possède à la fois un mode texte et un mode graphique.

Sur les suggestions des utilisateurs qui testent les modules d'Archer, nous avons gardé la possibilité de décrire le modèle en mode texte. En effet, cette description est beaucoup plus rapide que le mode graphique. En plus la possibilité de placer des commentaires, des informations sur la non-linéarité des éléments et de faire des copies de blocs de texte à l'intérieur d'un modèle ou d'aller les chercher dans d'autres, ainsi que l'aisance de son langage rendent son utilisation intéressante et souple.

Cependant pour rendre Archer très esthétique et moderne, la nouvelle version possédera un interface utilisateur permettant une description graphique du modèle physique.

Pour ne pas réécrire complètement un autre compilateur et pour ne pas avoir pour chaque mode un compilateur différent, nous avons conçu un interface graphique proposant une boîte à outils qui se base entièrement sur l'analyseur syntaxique précédent. L'utilisateur ne touche jamais à la feuille du dessin, il se contente de donner des ordres à partir de la boîte en cliquant sur des "Boutons BitMap Personnalisés" pour dessiner, définir le sens de la puissance, ou pour choisir le type de bifurcation (respectivement jonction) et le nombre de ses branches partant (respectivement entrant) d'un point de connexions.

Une liste est prévue pour sauvegarder les points de connexions, pour permettre aussi de les supprimer, de les lier à d'autres points ou de leur ajouter des branches supplémentaires. Une deuxième liste permettra de coupler des systèmes différents. Chaque fois qu'un domaine physique sera choisi, les "BitMap" associés à ses éléments s'afficheront à l'écran.

Cette méthode, qui est presque terminée actuellement, possède les avantages suivants :

Contrôle complet des opérations qui nous aideront à écrire, simultanément pendant la construction graphique, le modèle physique sous forme texte et sous forme de fichier "BitMap". On pourra alors utiliser l'analyseur sémantique précédent. Ici l'analyseur syntaxique ne sera pas utilisé puisqu'il l'est implicitement. Ceci va compenser les éventuelles pertes de la conversion "graphique ----> texte". En effet, il n'y aura pas d'erreurs de syntaxe, donc pas de gestion de messages etc.

La conversion inverse "texte ----> graphique" est très complexe, car il faut balayer pour chaque point de connexions le fichier pour déterminer sa nature, son type et ses nombres de branches. Dans la majeure partie des cas on peut avoir uniquement la dernière information. Cette conversion est de même type (au sens sémantique) que la "traduction automatique" qui constitue une vieille branche de l'intelligence artificielle. De toute manière cette conversion est sans intérêt particulier et est sans importance pour le déroulement d'Archer.

Les autres avantages résident dans la convivialité de Windows, et la réutilisabilité. En effet le même programme pourra être utilisé sans modification pour l'affichage du bond-graph (état 7 du processus d'Archer) ou pour coupler diverses présentations du système (physique, bond-graph et modèle mathématique). Cette mixité de présentation est souvent utilisée par les spécialistes.

La réutilisabilité est rendue possible car le programme de la construction traite des objets "BitMap" sans se préoccuper de leur contenu. L'implantation de ce noyau vide et réutilisable est rendue possible par la programmation orientée objet. Dans notre cas nous utilisons Turbo Pascal Windows pour atteindre nos objectifs.

Conclusion

Ce chapitre constitue la base de notre travail qui a pour tâche la conception et la réalisation d'une partie du projet Archer.

Ce dernier a pour objectif premier la construction d'un modèle qui se rapproche le plus possible de la réalité. Son principal outil est le bond-graph dont la modélisation est plus explicite que les outils classiques, car elle comprend à la fois les informations structurelles ou graphiques et celles des modèles mathématiques. Nous verrons dans

les chapitres qui vont suivre les méthodes qui lui permettront d'atteindre le but escompté.

Archer est avant tout un projet informatique conçu comme un outil d'aide pour les physiciens. Il fallait donc déterminer un langage qui soit simple à utiliser et qui permet de respecter l'architecture du système pour faciliter sa description.

Nous avons retracé ses caractéristiques générales (notamment sa conception orientée objet et l'utilisation des techniques de l'intelligence artificielle pour résoudre ses problèmes algorithmique -voir chapitre 4-), qui font de lui un produit informatique moderne répondant à plusieurs problèmes rencontrés lors du développement des projets.

Bien qu'Archer ne soit pas complètement terminé (voir chapitre 4 "Inventaire et perspectives d'Archer") le développement de certains de ses modules est plus une question de temps qu'une difficulté informatique.

Chapitre 2

CAUSALITE ET INFORMATIONS CAUSALES

Introduction générale

Dans ce chapitre nous allons étudier tous les aspects de la notion de causalité pour un modèle bond-graph. Ce qui permettra de déterminer les modèles mathématiques et d'analyser la structure d'un bond-graph.

La première partie sera consacrée aux méthodes d'affectation de la causalité. Nous verrons dans un premier temps les règles conventionnelles définies par D. Karnopp et R. C. Rosenberg. Elles montrent le type de causalité que peuvent avoir les éléments bond-graphs.

Nous discuterons ensuite les méthodes existantes, pour lesquelles nous essayerons de montrer les avantages et les inconvénients.

Nous présenterons alors la méthode de l'affectation automatique de la causalité que nous proposons dans le cadre de ce travail. Celle-ci comporte plusieurs algorithmes qui aideront à son implantation sur calculateur.

La deuxième partie sera consacrée à montrer comment extraire les informations causales d'un modèle bond-graph. Des notions comme "chemin causal" et "boucle causale" ainsi que leur signe et leur gain seront définis. Nous proposerons également des méthodes algorithmiques qui serviront à déterminer ces informations.

Les algorithmes seront écrits sous forme "méta-langage", proche du langage Pascal. Les mots réservés comme "Si", "Tant que", ... seront écrits en "**gras**". Les commentaires seront écrites entre des accolades "{ }".

Les procédures ou actions particulières ne seront détaillées que si elles ne sont pas évidentes, ou si elles sont importantes. Elles seront alors identifiées par une écriture en **gras.**

1° Partie : Affectation de la causalité

Introduction

Au cours de cette partie, nous présentons une étude des méthodes d'affectation de la causalité, avec une discussion sur les méthodes existantes.

L'intérêt d'une méthode automatique par rapport à une méthode directe (par l'intermédiaire d'un interface graphique) est défini.

Nous décrivons ensuite la méthode que nous proposons dans le cadre de ce travail.

I. Rappels

La notion de causalité joue un rôle très important dans la théorie des bond-graphs, elle constitue l'un des facteurs, nécessaires pour modéliser un système dynamique et à analyser sa structure. Son affectation correspond à la construction d'une série de décisions reflétant, pour chaque élément 1-port, la relation de cause à effet.

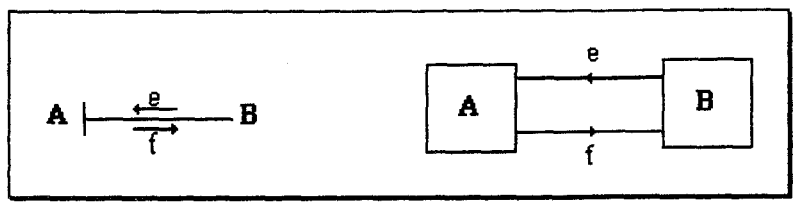
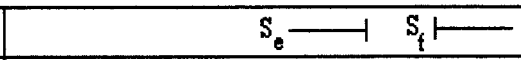
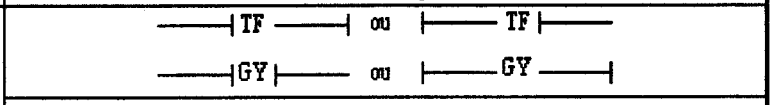
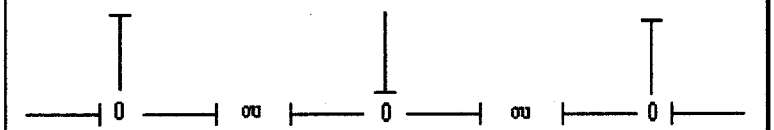
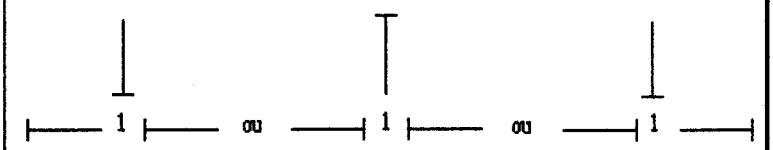
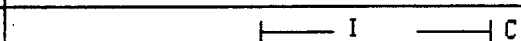
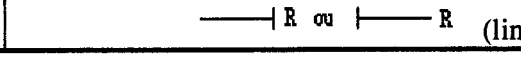


figure 2.1 : signification du trait causal

Les cas de la figure 2.1, présentés dans Karnopp & Rosenberg [1987], montrent les particularités de cette notion appliquée à deux composantes A et B. La relation de cause à effet est représentée dans un bond-graph par deux directions opposées et est indiquée par un trait vertical sur de chaque lien appelé "lien causal" (causal stroke).

L'affectation de la causalité à un bond-graph acausal doit respecter un certain nombre de règles conventionnelles. La table 2.1 présente tous les cas possibles de l'affectation causal pour un élément bond-graph.

Causalité nécessaire	
Causalité restreinte	  
Causalité intégrale	
Causalité dérivée	
Causalité arbitraire	

. table 2.1 : causalité des éléments de base d'un bond-graph

Rappelons ici les trois points essentiels :

. Une source ne peut avoir qu'un type de causalité en fonction de sa nature : effort sortant (flux entrant) pour une source d'effort ; effort entrant (flux sortant) pour une source de flux.

. Les variables d'état sont associées aux éléments dynamiques bond-graphs I et C. Pour des raisons d'ordre numérique on privilégie la causalité intégrale par rapport à la causalité dérivée pour les éléments I et C. On dit alors et que l'ordre du système est maximal. (*)

. Une jonction-0 doit avoir un lien et un seul possédant une causalité effort entrant (flux sortant). Et une jonction-1 doit avoir un lien et un seul possédant une causalité effort sortant (flux entrant).

L'algorithme séquentiel qui va suivre, présenté par Karnopp et Rosenberg [1976, 1983, 1986], permet d'appliquer les règles conventionnelles et de donner un ordre pour affecter la causalité aux éléments.

*

{étape 1}
répéter
Affecter la causalité aux sources en fonction de leur type (S_e ou S_f).
En déduire la causalité des jonctions 0, 1, TF et GY,
qui peut être induite.
jusqu'à ce que toutes les sources aient reçu une causalité

{étape 2}
tant qu'il existe un élément I et un élément C sans causalité **faire**
Assigner la causalité intégrale pour cet élément
En déduire toutes les implications possibles pour les jonctions
fait

{étape 3}
tant qu'il existe un élément R sans causalité **faire**
Assigner une causalité arbitraire à cet élément
En déduire toutes les implications possibles pour les jonctions
fait

Remarquons que cet algorithme convient surtout aux systèmes de rang maximal (tous les éléments dynamiques sont indépendants) et se prête bien à une affectation directe de la causalité à un modèle bond-graph où l'utilisateur affecte, à tel ou tel élément, une causalité en accord avec le modèle physique.

Une affectation automatique de la causalité utilise certes le principe de cet algorithme, mais nécessite néanmoins, lorsqu'une causalité dérivée est inévitable, la connaissance d'informations supplémentaires pour qu'il n'y ait pas de contradictions entre les lois régissant les phénomènes physiques et le modèle causal du bond-graph associé.

Remarques 2.1 :

1. Généralement les éléments R et les jonctions TF et GY ne sont pas prioritaires. Cependant il peut arriver que les éléments physiques qu'ils représentent, répondent à une loi ou à une relation qu'il faut respecter. Prenons par exemple l'élément R ; il peut représenter un élément ou un phénomène physique caractérisé par une loi non linéaire (diode, thyristor, ...) non bijective pour laquelle il faut avoir une causalité effort entrant. Dans ce cas l'algorithme exposé précédemment peut ne pas respecter cette contrainte. Une solution consisterait à traiter l'étape 3 immédiatement après l'affectation de la causalité aux sources. En réalité, le fait de changer l'ordre des

étapes ne constitue pas une solution car d'autres contradictions peuvent surgir.(*)

2. On trouve dans la littérature des méthodes qui, pour des raisons liées à la nature et à la classe du système étudié, ne prennent pas forcément en compte toutes les règles conventionnelles. Ainsi B.J. Joseph et H.R. Martens [1974] ont toléré, pour modéliser une classe de systèmes non linéaires, des jonctions-0 (respectivement des jonctions-1) avec plusieurs liens à effort entrant (respectivement à effort sortant).
G. Dauphin-Tanguy [1990] a montré que, pour modéliser les systèmes électroniques à partir des bond-graphs, une jonction-0 peut avoir, dans certains cas, plus d'un lien avec une causalité effort entrant. Ceci est lié à la nature des composants électroniques (comme les diodes, transistors et thyristors) qui possèdent un état bloqué et un état passant.

II. Méthodes existantes

Les logiciels utilisant la méthodologie bond-graph comme ENPORT, CAMAS, MS-Bond, ... ainsi que des réalisations telle que Delgado [1991], permettent aux utilisateurs d'affecter la causalité par l'intermédiaire d'un interface graphique avec une vérification syntaxique basée sur le respect des règles conventionnelles. L'absence d'une affectation automatique mais surtout de l'analyse sémantique (permettant de guider, d'orienter et de conseiller l'utilisateur avant les phases de modélisation et de simulation) restreint l'utilisation aux seuls bond-graphistes.

Dans ce qui va suivre, nous allons insister sur une méthode -la seule à notre connaissance-d'affectation automatique de la causalité présentée par S.J. Hood et al. [1989].

Son intérêt majeur est lié aux techniques informatiques choisies pour son implantation. Par exemple l'utilisation des tables de transition, généralement rencontrées dans la théorie des automates, apporte une richesse à l'algorithme de Karnopp & Rosenberg, et permet ainsi de réduire le nombre de tests alternatifs qui se schématisent dans les implantations par un nombre important d'imbrications de la technique de programmation structurelle " SI - Alors - Sinon " (if - then - else).

Cette méthode utilise une table principale dont les lignes représentent les éléments bond-graphs S_e, S_f, I, C, R, TF, GY, 0 et 1. Quant aux colonnes, elles indiquent les différents cas de figure qu'on peut rencontrer pour chaque type de noeud, ou encore

---

* Nous verrons plus loin dans la méthode que nous proposons comment résoudre ces problèmes.

les états de départ et les différents états qui peuvent résulter d'une précédente affectation. Une cellule de la table correspond à l'état de la causalité du prochain élément, en fonction de son type. Les valeurs des cellules caractérisent les actions ou routines qui permettent soit d'affecter les causalités obligatoires ou déduites, soit d'indiquer l'état causal de l'élément suivant, soit d'envoyer des messages...

Dans le cas d'un conflit (impossibilité d'affecter une causalité intégrale à un élément I ou C), deux autres tables, liées respectivement aux jonction 0 et 1, sont appelées.

Les états induits par l'utilisation de la table principale sont sauvegardés dans une pile qui contient aussi la trace de la causalité affectée à chaque lien dans l'ordre où elle a été traitée. La rencontre d'un conflit engendrera un appel, en fonction de la jonction, à l'une des deux tables auxiliaires dont les cellules contiennent des actions permettant de gérer la pile. Celle-ci sera alors dépilée jusqu'au moment où son sommet correspond à un élément dont la causalité est arbitraire, en l'occurrence un élément résistif linéaire ou des jonctions TF et GY. On modifie alors sa causalité et on recommence le traitement.

Cette opération peut malheureusement être très coûteuse du fait du nombre important de va et vient dans le modèle bond-graph et d'une fréquente mise en cause de l'état de la pile. En plus, l'absence d'une causalité arbitraire, pendant l'opération de résolution de conflit, arrête le processus d'affectation de causalité et envoie un message d'erreur.

Nous pensons également, que le fait d'obéir seulement aux règles conventionnelles limite son application aux systèmes de rang maximal. Autrement dit ceux qui possèdent des éléments dynamiques dépendants ne pourront pas être traités de manière correcte. Car la non participation de l'utilisateur à certaines décisions et la négligence de la nature physique des éléments peuvent conduire à des résultats inattendus.

III. Nouvelle méthode pour l'affectation automatique de la causalité

Nous allons proposer une méthode générale d'affectation de la causalité à un bond-graph. Elle présente l'intérêt de remédier aux problèmes relevés dans les autres méthodes, et de tenir compte :

-> de la nature et de la loi physiques de l'élément .
-> du souhait de l'utilisateur et de ses préférences.

Ainsi, on note parmi ses avantages :

-> la possibilité qu'elle offre de choisir la causalité (intégrale ou dérivée) des éléments de stockage d'énergie (même si par défaut il privilégie la causalité intégrale).
-> un traitement des boucles algébriques.
-> une réduction éventuelle du nombre de boucles causales.
-> une construction de chemins causaux courts, optimisant ainsi le traitement et la production des informations causales.
-> un ajout, par choix ou après avis de l'utilisateur, d'éléments permettant de résoudre certains conflits causaux (par exemple lorsque l'utilisateur veut absolument avoir un système de rang maximal : tous les éléments I et C en causalité intégrale).
-> l'utilisation des techniques de programmation orientée objet (P.O.O).
-> le recours à des règles d'expertise pour conseiller et aider l'utilisateur à prendre une décision ou, le cas échéant, permettre à ARCHER de les appliquer si l'utilisateur n'arrive pas à se décider pour un choix.

Cette expertise est fondée sur des règles que nous avons définies, et qu'on peut grouper en deux parties : des règles statiques (ou règles de production) qui sont à la base de l'implantation informatique de notre méthode (écrite entre autres en turbo prolog), et des règles, dites "dynamiques", en nombre moins important que les précédentes. C'est cette dernière catégorie que l'utilisateur peut modifier et enrichir, dans la limite d'une syntaxe que nous avons défini.

III.1 Principe et idée de base

Commençons par présenter l'idée de base sur laquelle nous avons axé notre étude :

> **Chercher parmi les liens associés à chaque jonction-0 (respectivement jonction-1) celui qui doit avoir la causalité effort entrant (respectivement effort sortant). Autrement dit, celui dont le trait causal est proche (respectivement loin) de la jonction-0 (respectivement jonction-1).**

Dorénavant nous appellerons ce lien **"le représentant causal de la jonction"** ou tout simplement **"le représentant causal"** (*)

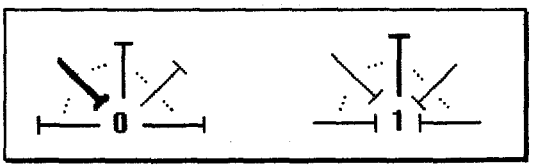


figure 2.2 : exemple de représentants causaux

Les règles de production (règles statiques) ou les choix de l'utilisateur peuvent induire un représentant causal, qui à son tour pourrait donner, après une phase de déduction, un autre représentant causal. Dans le premier cas le représentant causal est associé à un lien entre une jonction (0 ou 1) et un élément de type S_e, S_f, I, C et R, alors que dans le deuxième cas il correspond à un lien entre deux jonctions.

L'algorithme 2.1 retrace les grandes lignes des étapes suivies par notre méthode. Nous reviendrons par la suite sur les parties essentielles (étape 1 et étape 3). Le tout sera illustré par des exemples afin de faciliter la compréhension de notre démarche.

Algorithme 2.1

{Description globale des différentes étapes de la méthode}

Etape 1 {Affectation de la causalité obligatoire}

1ère phase :

*Affecter la causalité aux sources d'effort S_e (respectivement S_f) liées à des jonctions 0 (respectivement 1)

* Affecter la causalité aux éléments munis d'une loi physique particulière.

=> production, dans le cas de l'existence de ce type de source ou de l'existence des lois physiques associées à certains éléments, de la première liste d'objets associés à la classe des "représentants causaux".

2ème phase :

* Cet objet s'occupe de toutes les déductions de l'application causale induite et envoie éventuellement des messages d'erreur (ex : plusieurs sources de classe précédente sont associées à la même jonction)

=> production des premiers liens causaux représentés dans Archer par les prédicats de la base de données : **Numéro_effort (N, X, Y)** tel que $x \xrightarrow{n} y$

En indiquant que, pour le lien numéroté par N, l'effort part de X vers Y (cf. chapitre 1).

Etape 2 {Affectation préférentielle}

1ère phase :

*Adapter l'environnement utilisateur en fonction des systèmes physiques (électriques, électroniques, mécaniques, ...) présents dans le système dynamique

= > production de messages et de dialogues du même type que le langage utilisateur.

2ème phase :

* Imposer la causalité à certains éléments (ou à tous)

= > vérification de la validité de cette opération et production d'une liste de classe d'objets "représentants causaux".

** si pendant la vérification on aboutit à un conflit causal

alors proposition de solution pour résoudre le conflit **fsi**

= > 3 possibilités :

ajouter un élément à une jonction du type du conflit et modification des modèles physique et bond-graph.

ou remettre en cause le type de la causalité imposée à un élément.

ou tolérer le conflit {ici l'utilisateur peut tolérer le conflit et c'est à lui de vérifier la portée des conséquences au niveau des modèles mathématiques (équations d'état, matrice de transfert, ...)}

** Appel à la déduction par l'objet "représentant causal"

= > production des liens causaux indiqués par le prédicat Numéro-effort (Numéro de lien, noeud de départ, noeud d'arrivée).

3ème phase :

* Attribuer une causalité préférentielle à certains éléments

= > liste des choix préférentiels.

** Vérifier la validité de la causalité choisie par l'utilisateur

si conflit **alors** ne pas tenir compte du choix

sinon produire l'objet "représentant causal".

= > liste des "représentants causaux".

fsi

** Traitement de la classe d'objets "représentants causaux"

= > production de liens causaux.

Etape 3 { Affectation logique } (* *)

* Parcourir le bond-graph en profondeur comme s'il s'agissait d'un arbre

= > appliquer à chaque état final les règles statiques d'expertise (cf. plus bas).

---

* Cette étape, qui reflète l'une des originalités de nos travaux, sera étudiée en détails par la suite.

** En cas de conflit ou de choix entre plusieurs éléments de même type

= > dialogue avec l'utilisateur et déclenchement des règles dynamiques d'expertise

= > production de classe d'objets "représentants causaux".

Remarque 2.2 :

L'étape 2 présente un interface utilisateur convivial. Nous verrons, dans le chapitre V, sa spécificité et des exemples d'application de cet algorithme à différents systèmes physiques.

Avant de détailler les points importants de chaque étape, nous allons nous intéresser à la phase de déduction commune à chacune d'elles. En effet, le représentant causal de chaque jonction 0 ou 1, permet d'induire la causalité des autres liens et de produire parfois une autre liste de représentants. Ce qui, du point de vue informatique, va conduire à une récursivité du processus et une hiérarchisation du traitement de chaque objet. A noter que la notion d'objet (cf. pour plus de détail chapitre 5) représente à la fois le type, à savoir ici la classe des représentants, et le module (sous-programmes) qui gère cette classe.

Commençons par exposer les cas de figure susceptibles d'exister dans un modèle bond-graph .

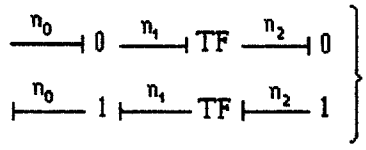
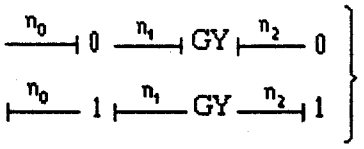
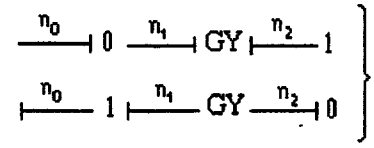
Dans la suite, nous notons J une jonction 0 ou 1 et J_d son dual, c'est-à-dire J_d est une jonction-1 si J est une jonction-0, et inversement .

Classe d'objet : ——⊣ 0 "représentant causal" (lien causal près de la jonction 0)

⊢—— 1 "représentant causal" (lien causal loin de la jonction 1)

Classe de fonction ou règle de déduction :

Règle de déduction — J — J_d $\xrightarrow{n_0}$ 0 $\xrightarrow{n_1}$ 1 $\xrightarrow{n_0}$ 1 $\xrightarrow{n_1}$ 0	n_0 va induire la causalité de n_1
Règle de déduction — J — TF — J_d $\xrightarrow{n_0}$ 0 $\xrightarrow{n_1}$ TF $\xrightarrow{n_2}$ 1 $\xrightarrow{n_0}$ 1 $\xrightarrow{n_1}$ TF $\xrightarrow{n_2}$ 0	n_0 va induire la causalité de n_1 et de n_2

Règle de déduction — J —TF— J 	n_0 va induire la causalité de n_1 et de n_2 Création d'un nouveau représentant causal (n_2) et itération du processus.(*)
Règle de déduction — J —GY— J 	Déduction de la causalité pour n_1 et n_2 à partir de n_0
Règle de déduction — J —GY— J_d 	n_0 va induire la causalité de n_1 et de n_2 Création d'un nouveau représentant causal (n_2) et itération du processus.(*)

Algorithme 2.2 : "déduction causale "

{partie déclaration }
L : liste de couples de type (e_i , J) correspondant à un lien représentant causal.
Cette liste a une structure de pile de type :
Last In First Out.(Dernier Entrant Premier Sortant)
avec e_i : élément bond-graph de type [I, C, R, S_e, S_f, 0, 1, TF, GY]
J : Jonction de type 0 ou 1 .

{Corps de l'algorithme}
Début
répéter
{ dépiler et récupérer le sommet }
(e_i , J) := dépiler (L)
{s'occuper d'affecter la causalité à la liste d'éléments associés à la jonction J}
Traiter_Liste_Jonction (e_i ,J)

---

* La création ou la rencontre d'une classe d'objet peuvent être interprétées comme un message envoyé à l'objet lui même qui s'occupe d'itérer le processus, et qui, une fois terminé, indique à un autre de continuer.

```
        tant que existe une jonction J_h, non marquée, liée à J    faire
            marquer (J_h)
            {appliquer la règle de déduction adéquate}
            Valider_une_Régle_déduction ( J,J_h)
        fait
    jusqu'à  L devient vide
Fin
```

Algorithme 2.2.1

```
Traiter_liste_Jonction (e_i, J)
{traitement de la liste d'éléments de type [ I, C, R, Se, Sf ] attachée à une jonction 0 ou 1 }
Début
    soit LE cette liste
    si J est une jonction 0      alors    Sens : = "e"  {sens effort partant de J = jonction-0 }
                                 sinon    Sens : = "f" {sens flux partant de J = jonction-1 }
    fsi
    Affecter_la_causalité ( J, e_i,  dual(Sens) )
    {-ce sous programme s'occupe à la fois de la numérotation du lien et de l'affectation de la
    causalité dans le bon sens
    - dual est une fonction qui inverse le rôle en "e" (effort) et "f" (flux) }
    pour tout élément de e_j de LE tel que e_j * e_i  faire
        Affecter_la_causalité ( J, e_j, Sens )
    fait
Fin {de traiter liste Jonction}
```

Algorithme 2.2.2

```
Valider_Règle_Déduction (J, J_h) (voir plus haut)
{Ce sous programme s'occupe de déclencher l'une ou l'autre des règles statiques de déduction. Nous
donnons ici une présentation classique de cet algorithme, cependant pour l'implanter sur un ordinateur
il vaut mieux utiliser une approche et un langage de programmation logique et/ou de la P.O.O (cf.
remarques suivantes) afin d'appréhender ce problème.}
```

```
Début
    Affecter_la_causalité (J, J_h, Sens)
    {on suppose ici que Sens est une variable globale }
    si  J_h est duale à J  alors  rien
    sinon
        chercher une jonction J_k (différent) J liée à J_h
```

```
        si  J_h est une jonction TF  alors
            si J_k est une jonction duale à J  alors  Affecter_la_causalité ( J_h , J_k , Sens)
            sinon  { on a le choix entre 3 types de traitement }
             . itérer le processus déduction causal avec une liste contenant un élément [(J_h , J_k)]
            .ou en utilisant une technique de P.O.O, envoyer un message ("existence d'un
            représentant causal"). Cet événement sera capté et traité par l'objet représentant
            causal.
            . ou attendre que la nouvelle liste des représentants causaux soit complètement
            déduite à partir de la liste initiale et appliquer l'un des deux cas précédents.
            fsi
        sinon   { J_k est une jonction GY }
   {la variable d'orientation causale (effort ou flux) doit être dualisée dans une jonction GY }
            Sens := dual (Sens)
            si J_k est une jonction duale de J alors
                    Idem qu'avec TF et telle que J_k est une jonction non duale de J.
            sinon
                    Affecter_la_causalité ( J_h , J_k , Sens ).
            fsi
        fsi
   fsi
Fin {de Valider_Règle_Déduction}
```

Algorithme 2.2.3

```
Affecter_la_causalité (X, Y, Sens)
Début
   rétracter(Numéro_courant(N)) {récupération et effacement de la mémoire de l'information
   donnant le numéro courant du lien }
   N : = N + 1      {incrémentation de ce numéro pour le mettre à jour}
   Insérer ( Numéro_courant(N)) {sauvegarde pour une utilisation future}
   si Sens = "e" alors
      insérer (numéro_effort (N, X, Y)) {Sauvegarde dans la base de données (ou fichier) de
         l'information donnant pour chaque lien, son numéro et le sens de sa variable effort }
   fsi
Fin {Affecter_la_causalité}
```

La partie qui s'occupe de la validité des règles de déduction, ne traite ni le cas d'une cascade de liaisons TF ou GY (TF—TF— ... —TF ou GY— ... —GY), ni celui d'une alternance cascadée de TF et GY (TF—GY— ... TF—GY). Nous avons omis ces cas pour rendre l'algorithme lisible, il suffit pour en tenir compte de remplacer le

test alternatif au niveau des jonctions TF et GY, par une boucle itérative qui permettra d'avancer et d'affecter la causalité tant que la liaison entre des jonctions de même type existe. Le cas d'une alternance entre une jonction TF et GY peut se résoudre par l'ajout d'une boucle itérative permettant, tant que ce type de liaison existe, de dualiser le sens de la causalité et de l'affecter.

L'algorithme qui s'occupe du traitement de la liste d'éléments attachée à une jonction 0 ou 1, est lié à la caractéristique de la représentation formelle d'un modèle bond-graph dans Archer (cf. chapitre 1). En effet, celle-ci associe à toute jonction la liste d'éléments de type I, C, R, Se et Sf, qui lui sont liés. D'où le fait d'utiliser une procédure à part va accélérer le traitement, sachant qu'aucune déduction n'est à espérer dans ce cas. Tout autre type de représentation ne séparant pas un élément d'une jonction, va nécessiter une règle supplémentaire de même type que la règle numéro 1, ou de généraliser celle-ci. Ce qui revient, dans l'algorithme précédent, à fusionner la procédure de traitement de liste avec celle de la validité des règles.

Pour finir cette section, nous notons que la technique de recherche et de déduction, à partir de la classe des représentants causaux, possède l'inconvénient d'associer une numérotation désordonnée aux liens. Cet inconvénient (comme le montre l'exemple de la figure 2.3) est lié seulement à l'esthétique de l'affichage graphique du modèle bond-graph, et n'affecte en rien ni la phase de modélisation, ni celle de l'analyse.

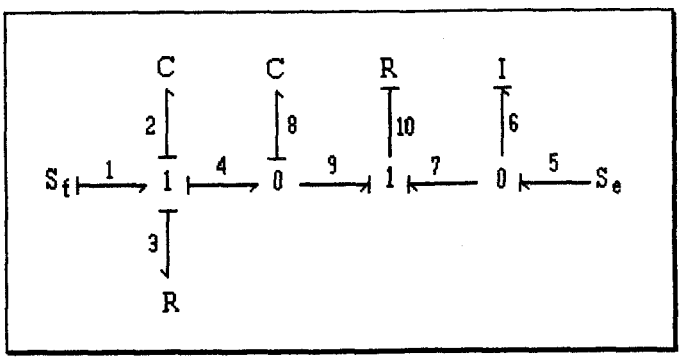


figure 2.3 : exemple de bond-graph causal possédant une numérotation désordonnée.

L'affectation de la causalité pour le modèle bond-graph de la figure 2.3, commence par suivre l'étape 1 qui produit deux représentants causaux à savoir $S_f \xrightarrow{1} 1$ et $0 \xleftarrow{5} S_e$. La partie déduction a affecté la causalité et a numéroté les liens 1, 2, 3, et 4 par application de l'algorithme précédent au premier représentant causal. En suivant la même démarche pour le deuxième, et sachant que la numérotation est automatique, il a été obligatoire d'affecter et de numéroter à partir de 5, les liens 5, 6 et 7. Le reste découle de l'application de l'étape 2 ou de l'étape 3.

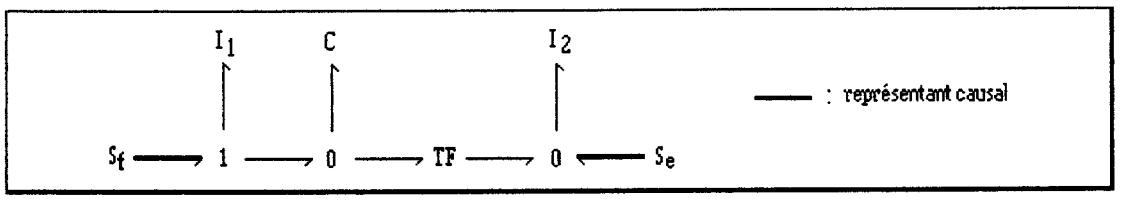
IV. Traitement de la causalité obligatoire (étape 1 de la méthode proposée)

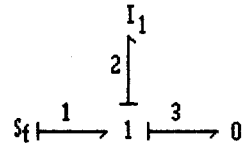
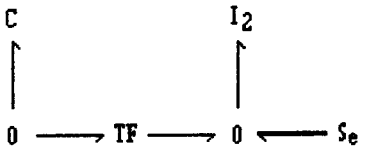
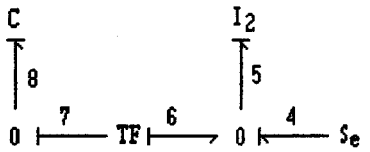
Au cours de cette section, nous allons voir, sur un exemple, comment appliquer la première étape de notre méthode, écrire son algorithme et débattre des différents problèmes susceptibles d'être déduits dans cette partie.

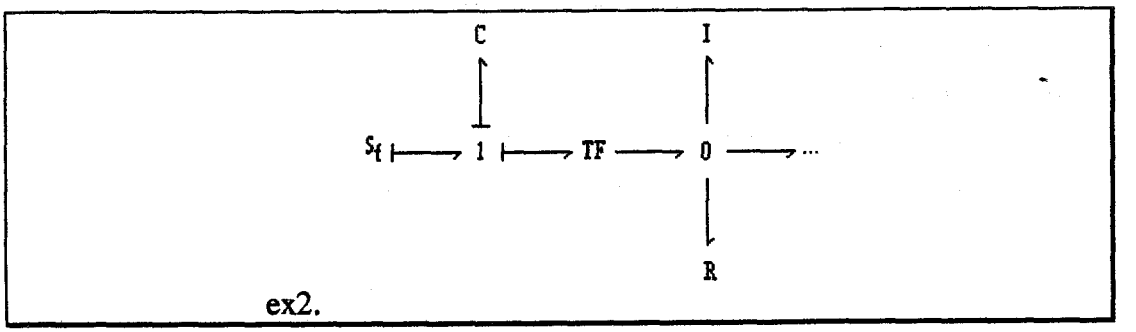
IV.1 Exemples d'application

Les exemples du tableau 2.1 résument les phases de l'application de cette première étape. Lors de la première phase, la classe des représentants causaux de type S_f (source du flux) liée à une jonction-1 ou S_e (source d'effort) attachée à une jonction-0 sera déterminée. Pendant la seconde phase, par application des règles de déduction, et à partir de chaque élément de la classe précédente, une causalité sera affectée à d'autres liens.

ex1.



Affectation de la causalité et déduction causale	Partie restante du bond-graph acausal	Information
		I_1 est obligatoirement en causalité dérivée. Pour changer sa causalité => modifier le modèle physique.
	Vide	C est obligatoirement en causalité dérivée. Pour changer sa causalité => modifier le modèle physique.



ex2.

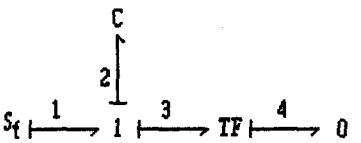
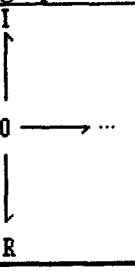
Affectation de la causalité et déduction causale	Partie restante du bond-graph acausal	Information
		Rien

tableau 2.1 : exemples d'application de l'étape 1 "causalité obligatoire"

Cette phase correspond à la colonne 1 du tableau 2.1, la colonne 2 indique chaque fois la partie du modèle bond-graph qui reste à traiter et la colonne 3 renseigne sur la causalité : causalité dérivée ou contradiction causale selon les règles conventionnelles.

Ces informations renvoient une éventuelle anomalie dans la description du modèle physique, ce qui peut être considéré comme le résultat d'une analyse sémantique du modèle bond-graph. L'utilisateur peut les consulter et intervenir si une modification est nécessaire.

Lors de l'implantation sur ordinateur de l'étape 1, ces informations peuvent enrichir le dialogue et l'échange d'informations avec l'utilisateur. Le programme pourra alors s'occuper de proposer et d'apporter les modifications résultant de cet échange.

IV.2. Algorithme de l'étape

Algorithme 2.3

causalité obligatoire

Début

L : = Vide {noté habituellement par ... ou [ ] }

pour chaque jonction J de type 0 ou 1 **faire**

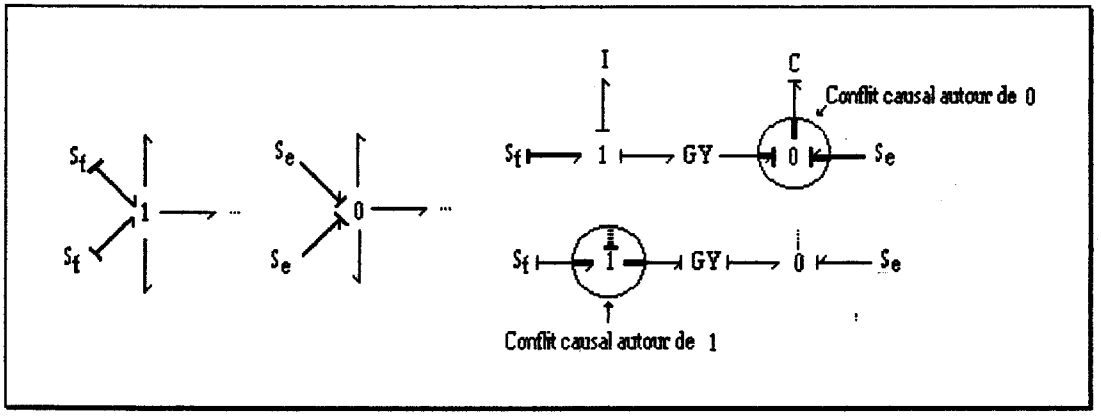
si J est une jonction-0 **alors**

former la liste $L_{Sources}$ des sources

```
        effort "Se" liées à J
    sinon {J est une jonction-1}
        former la liste LSources des sources
        flux "Sf" liées à J
    fsi
    si cardinal (LSources) > 2 alors
        . erreur
        . arrêt
    sinon si Cardinal (LSources) = 1 tel que LSources = [Sj] alors
                empiler ( J, Sj ) : L := L    [(J, Sj)]
    fsi
    Déduction_Causale (L)
  fait
Fin
```

L'erreur qu'on risque de rencontrer dans l'application de cet algorithme à un modèle bond-graph, provoquera l'arrêt du programme et nécessitera obligatoirement une modification du modèle descriptif du système dynamique. Le concepteur peut gérer cette situation de façon à ce que son programme ne s'arrête pas brutalement et entame un dialogue avec l'utilisateur.

La figure 2.4 présente quelques problèmes détectés par l'affectation de la causalité mettant en cause le modèle physique.



a. plusieurs sources flux attachés à une jonction-1

b. plusieurs sources effort attachés à une jonction-0

c. contradictions causales ou conflits causaux

figure 2.4 : quelques cas présentant des contradictions causales

Les figures 2.4.a et 2.4.b montrent les cas, traités dans l'algorithme précédent,

nécessitant obligatoirement une intervention de l'utilisateur. Celui de la figure 2.4.c est décelé pendant la déduction causale, il est plus compliqué à traiter : il suppose à chaque application d'une règle de déduction, conduisant à une récursivité (ou itération du processus : Régles de déduction n°3 et n°4), la mise en place d'un test qui vérifie si la jonction atteinte possède aussi un représentant causal.

Si ce test est positif, une phase de dialogue avec l'utilisateur est entamée dans le but d'une éventuelle modification des modèles physiques et bond-graph. L'utilisateur peut également lorsqu'il peut se justifier par exemple dans le cas de phénomènes de commutation rendant active une seule branche du bond-graph à la fois.

Les représentants causaux imposés par une loi physique, par la nature de son module (fonction non linéaire) ou par l'utilisateur (étape 2) peuvent engendrer lors d'une déduction causale une erreur nécessitant un traitement équivalent.

V. Traitement de l'affectation logique (étape 3 de notre méthode) (*)

l'étape 2 ne présente pas de difficulté particulière, son rôle se limite à adapter l'environnement à un langage naturel, à saisir et à interpréter les réponses de l'utilisateur.

Cette étape constitue l'ossature de la méthode d'affectation de la causalité que nous proposons.

Elle utilise les données associées à la partie acausale (non traitée par l'exécution des étapes 1 et 2) du modèle bond-graph. Cette partie pourrait être :

-vide : dans ce cas l'exécution de l'étape 3 n'aura pas lieu.
-réduite par rapport au modèle initial : seule une partie du modèle a été affectée par les étapes 1 et/ou 2.
-égale au modèle initial : les conditions de l'étape 1 n'ont pas été satisfaites et l'utilisateur n'a pas souhaité imposer une causalité.

Voyons maintenant comment nous allons considérer, représenter et parcourir cette partie acausale pour pouvoir traiter et analyser les différentes situations qu'un modèle

---

* La création ou la rencontre d'une classe d'objet peuvent être interprétées comme un message envoyé à l'objet lui même qui s'occupe d'itérer le processus, et qui, une fois terminé, indique à un autre de continuer.

bond-graph peut engendrer. Mais d'abord rappelons les notions que nous utiliserons.

V.1 Transformation d'un bond-graph en un graphe

On distingue deux classes différentes de conversion de bond-graph en graphe :

La première a été introduite par Bell et Martens [1974], Perelson et Oster [1976]. Ils ont fait une comparaison entre l'étude des systèmes linéaires à partir des bond-graphs et des graphes linéaires. Il s'est avéré qu'un bond-graph ne définit pas (comme c'est le cas pour le graphe linéaire) seulement la topologie ou la structure du système, il représente en plus le transfert de la puissance.

Un bond-graph contient donc deux graphes classiques : un graphe causal en représentant la relation de cause à effet et un graphe associé au transfert de la puissance.

Pour transformer un bond-graph en un graphe simple une information est obligatoirement perdue Ort et Martens [1974]. Pour effectuer l'opération inverse le manque d'une information nécessite l'utilisation de deux graphes pour sa réussite.

Birkett [1989] a donné également une approche de cette conversion à partir des matroïdes.

La seconde classe peut ne pas être considérée comme une transformation car elle ne se substitue pas totalement au modèle bond-graph : c'est une façon de représenter et d'explorer informatiquement un modèle bond-graph selon la théorie des graphes. Ainsi on trouve dans Bouayad et al. [1990] une méthode permettant d'afficher un modèle bond-graph (représenté formellement) sur écran par sa conversion en un graphe. On trouve également dans Azmani et al. [1990] une représentation du modèle bond-graph sous forme d'un graphe permettant de faciliter l'exploitation des informations causales (*).

Dans cette optique nous allons parcourir un modèle bond-graph acausal comme s'il s'agissait d'un graphe non orienté en suivant le sens des demi-flèches, afin de lui affecter une causalité.
La figure 2.5.b présente le graphe associé au bond-graph de la figure 2.5.a. Ses sommets vont correspondre aux jonctions 0,1,TF et GY et ses arêtes sont les liens

---

* voir partie 2.

associant ces jonctions. Ceci est équivalent à la structure d'un bond-graph.

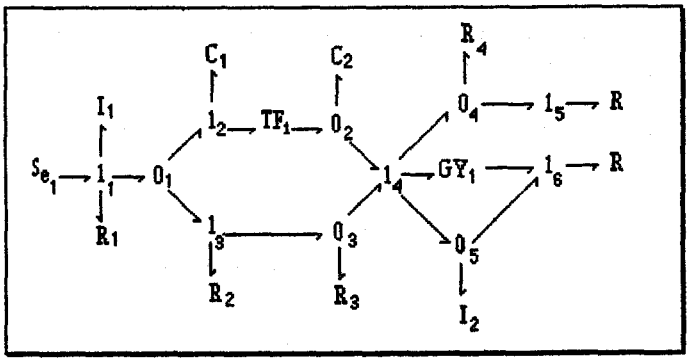


figure 2.5.a exemple d'un modèle bond-graph

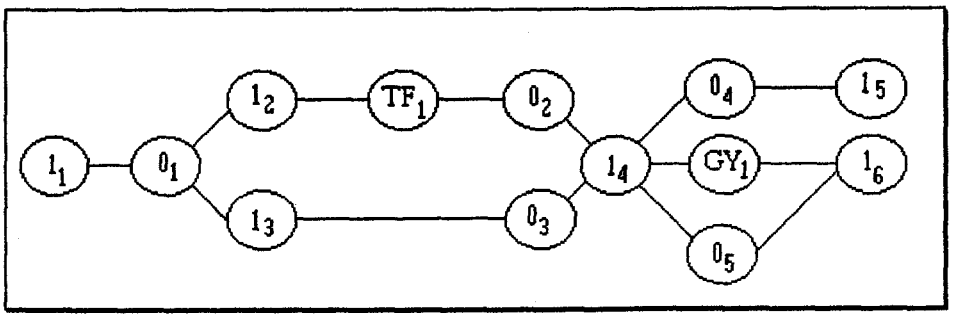


figure 2.5.b : graphe non orienté associé au modèle de la figure 2.5.a

Remarque 2.3 :

Nous avons vu au premier chapitre qu'Archer représente formellement un bond-graph sous la forme **"jonction (J, liste d'éléments associée à J, une convention sur le lien : simple ou multiple)"** et **"lien (J_0, J_1)"**. Cette représentation séparant les éléments associés à une jonction et la relation entre les jonctions facilite plusieurs traitements, et notamment la conversion "bond-graph / graphe". Comme le montre la figure 2.6, cette conversion sera retrouvée de manière implicite si on considère le prédicat lien (J_0, J_1) comme une arête.

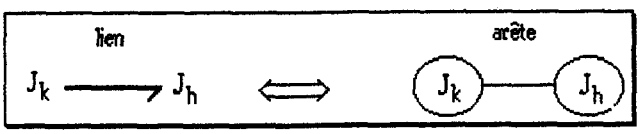


figure 2.6 : équivalence entre un lien et ne arête

La partie acausale (sur laquelle s'applique l'algorithme de "l'affectation logique"), correspond, assez souvent, à un arbre ou à une forêt. D'où le choix de son exploration en profondeur. En partant d'une jonction quelconque et en utilisant les liaisons entre les sommets (prédicat lien (X,Y)), nous sauvegardons ces différentes informations

dans une pile, appelée **pile de parcours**, de type **"LIFO ou DEPS"**, jusqu'à la rencontre d'un **sommet terminal** (jonction dont toutes les informations, sous forme de prédicat "lien (_, _)", ont été utilisées ; ou encore noeud associé au sommet de la pile qui ne possède pas de fils **ou** dont les fils ont été tous traités préalablement). On dépile le sommet de **la pile de parcours** et on analyse sa situation.

La figure 2.7 montre l'arbre associé à l'exploration en profondeur du graphe de la figure 2.5.b.

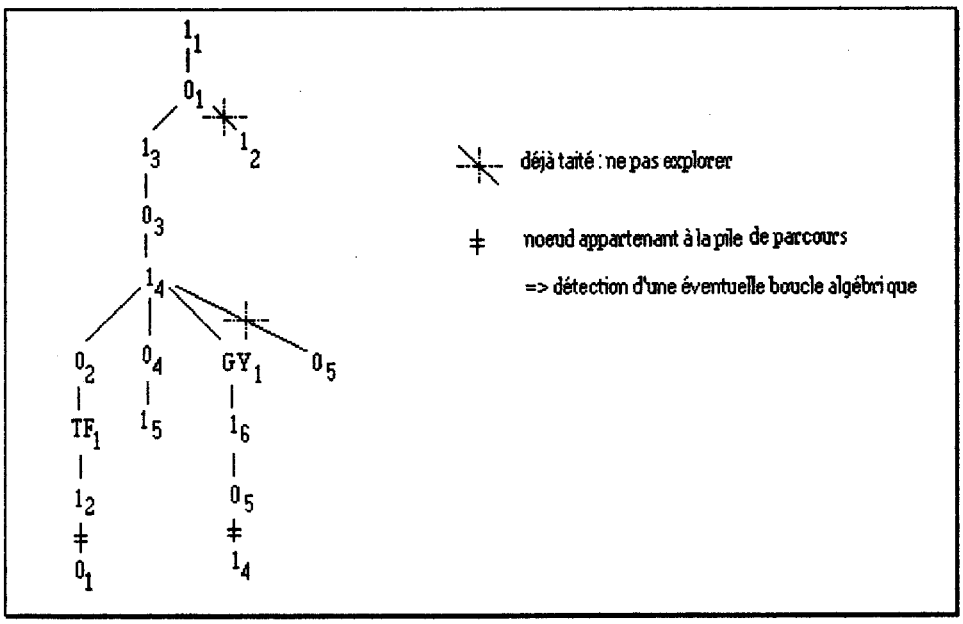


figure 2.7 : arbre représentatif de l'exploration en profondeur du graphe de la figure 2.5.b.

V.2 Définitions et conventions

Définissons les conventions qui vont nous servir à écrire l'algorithme de "l'affectation logique".

- Une jonction-1 (respectivement 0) possédant dans sa liste au moins un élément **I** (respectivement **C**), sera représentée par le triplet suivant :

('**i**', [liste des éléments **I** associés à la **jonction-1**], indice de la jonction dans le modèle)

(respectivement :

('**i**', [liste des éléments **C** associés à la **jonction-0**], indice de la jonction dans le modèle))

- Une jonction-1 (respectivement 0) ne possédant dans sa liste aucun élément **I** (respectivement **C**), sera représentée par le triplet suivant :

('**d**', [liste des éléments **C** associés à la **jonction-1**] **ou** [liste vide si pas de **C**],

indice de la jonction dans le modèle)
(respectivement :
('**d**', [liste des éléments **I** associés à la **jonction-0**] **ou** [liste vide si pas de **I**],
indice de la jonction dans le modèle)) (*)

* l'indicateur '**i**' signale qu'une causalité intégrale sur un élément dynamyque est possible .
l'indicateur '**d**' signale qu'une causalité dérivée sur un élément dynamyque est obligatoire .

Remarque 2.4 :

Les conventions précédentes correspondent au cas où le type de la causalité générale souhaitée est intégral. Si celui-ci correspond à une maximisation de la causalité dérivée pour les éléments de stockage d'énergie, on inverse le rôle des jonctions 0 et 1 et celui de I et C.

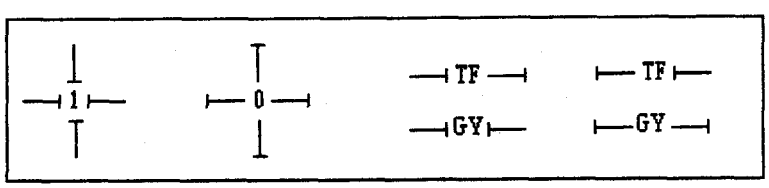
- Par convention les jonctions TF et GY seront représentées en fonction des cas suivants par :

J_p —— TF_k —— J_{D_q} -> ('i', [ ], TF_k)
J_p —— TF_k —— J_q -> ('d', [ ], TF_k)

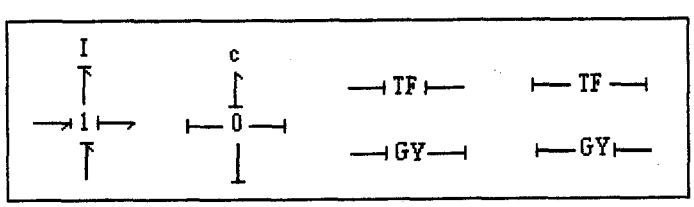
J_p —— GY_k —— J_q -> ('i', [ ], GY_k)
J_p —— GY_k —— J_{D_q} -> ('d', [ ], GY_k)

avec J_p et J_q deux jonctions de même type et J et J_D deux jonctions duales.

Les différentes conventions correspndent aux cas de figure suivants :



noeuds de type 'i'



noeuds de type 'd' : pseudo-conflit causal

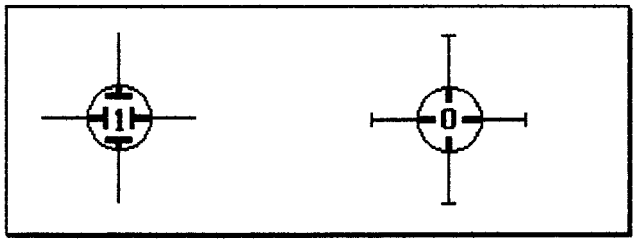


figure 2.8.a : conflits de type 'd' solvables par application des règles d'expertise, provoqué par une jonction-1(resp. 0) ne possédant pas d'élément I (resp.C), et traité au niveau de l'étape3.

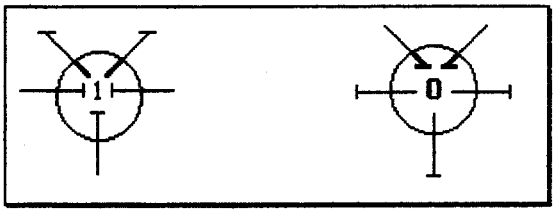


figure 2.8.b : type de conflits nécessitant un dialogue avec l'utilisateur et provoqué par une déduction causale.

Quant aux conflits causaux deux types sont possibles :

- le premier est provoqué par la représentation basée sur les conventions précédentes (cf. figure 2.8.a).

- le second peut être induit par une déduction causale.

Remarque 2.5 :

Le premier conflit est toujours solvable par application des règles statiques (cf. plus bas). Quant au second, sa résolution nécessite l'intervention de l'utilisateur soit pour apporter des modifications au système physique, soit pour tolérer le conflit (cf. remarque 2.2).

L'information sur la présence, dans la liste d'une jonction, possédant une convention de type **'d'**, d'éléments résistifs (de causalité arbitraire) doit être sauvegardée sous forme d'un prédicat (ou fait) de la base de données (ou tout autre type de représentation de données). Ainsi dans ARCHER nous avons choisi de la représenter sous la forme :
"**info_résistive** (type et indice de la Jonction, [liste des éléments R qui lui sont associés])"

Bien entendu cette information n'existe que si la liste des éléments R associée à une jonction est non vide. Elle sera alors très utile pour résoudre un conflit de type 'd' .
Une autre information, essentielle pour l'une des règles dynamiques, est associée à la présence des sources, de type différent de celui de l'étape1 : "affectation obligatoire", dans la liste d'une jonction-0 ou d'une jonction-1. Elle est sauvegardée dans ARCHER sous la forme :
" **info_source** (type et indice de la Jonction)"

Cette information permettra, lorsqu'elle existe, de privilégier l'affectation de la causalité à l'élément ou aux éléments directement **commandables** à partir de la source qu'elle représente.

L'application des conventions précédentes au modèle bond-graph acausal de la figure 2.5.a. aboutira au modèle causal suivant :

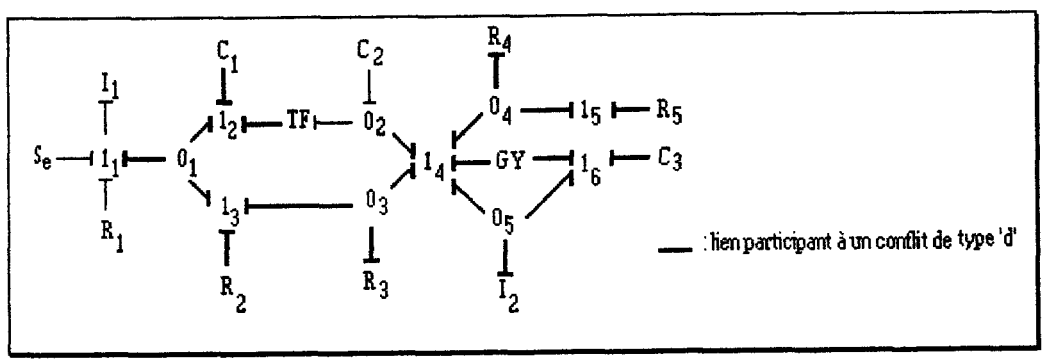


figure 2.9 : application des conventions au modèle bond-graph de la figure 2.5.a

Notons que la causalité du modèle de la figure 2.9 est une affectation virtuelle, c'est-à-dire aucune construction causale n'a eu lieu, c'est seulement le résultat qu'on aurait pu avoir par application des conventions pré-citées.

L'exploration en profondeur de la figure 2.9 sera simulée par l'arbre suivant :

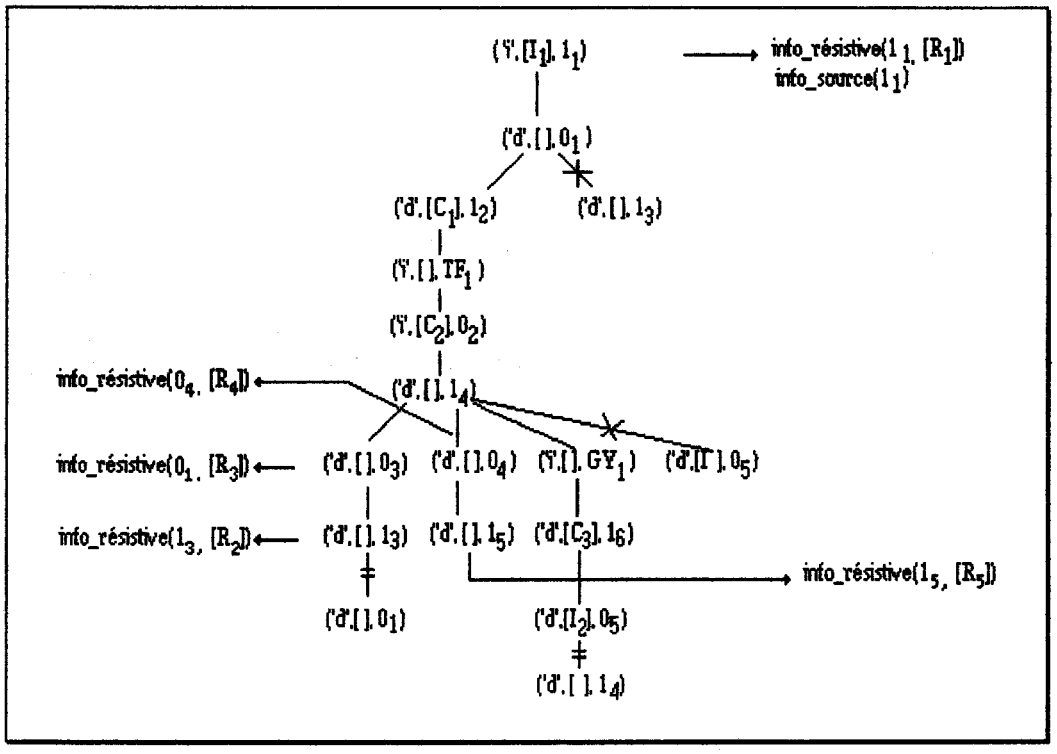


figure 2.10 : arbre simulant le parcours du modèle bond-graph de la figure 2.5.a en tenant compte des conventions définies précédemment.

IV.4. Algorithme de l'affectation logique de la causalité

Algorithme 2.4

affectation logique

{Adaptation des jonctions aux conventions précédentes et exploration en profondeur du bond-graph}

Début

Mettre dans la base de données active la partie restante du modèle acausal du bond-graph initial

Initialiser la pile de parcours

tant que Existe , dans la base de données active, une jonction **J** représentée par

le prédicat **jonction(J, L)** avec L liste des éléments associés à J **faire**

Adapter_Convention _Jonction(J, L, Triplet)

{cette procédure va adapter la convention de la jonction J en fonction des éléments de L. Elle retourne un résultat sous forme d'un triplet de type ('i' ou 'd', [ ] ou [liste d'éléments I ou C], J) }

racine := Triplet

Rétracter de la base de données active le fait " jonction (J, L)"

Explorer_Arbre(racine)

fait

Fin.

Algorithme 2.4.1 :

Adapter_Convention_Jonction (J, L, T)

Début
L_I = complément des éléments de type [R, C, S_e et S_f] par rapport à L
L_C = complément des éléments de type [R, I, S_e et S_f] par rapport à L
L_R = complément des éléments de type [I, C, S_e et S_f] par rapport à L
L_S = complément des éléments de type [I, R, C] par rapport à L
{ sauvegarder l'information sur la présence d'éléments R }
si $L_R \neq [\,]$ **alors**
  insérer dans la base de données active l'information **info_résistive (J, L_R)**
fsi
si $L_S \neq [\,]$ **alors**
  insérer dans la base de données active l'information **info_source (J)**
fsi
si J est une jonction-1 **alors**
  si $L_I = [\,]$ **alors** T := ('d', L_c, J) **sinon** T := ('i', L_I, J) **fsi**
    sinon { J est une jonction-0}
      si $L_c = [\,]$ **alors** T := ('d', L_I, J) **sinon** T := ('i', L_c, J) **fsi**
  fsi
Fin

Remarque 2.6

Dorénavant nous prévilégions dans nos exemples la causalité intégrale, il suffit d'inverser les rôles de 'i' et de 'd' pour les adapter au cas où la causalité dérivée doit être majoritaire. Pour ce qui concerne le traitement en lui même rien n'est à modifier, il faut seulement interpréter "choisir la causalité intégrale" par "choisir la causalité souhaité (intégrale ou dérivée)" et "choisir la causalité dérivée " par "choisir la causalité duale de celle souhaité".

Algorithme 2.4.2 :

Explorer_Arbre (racine)

{racine est de type ('i' ou 'd', [ ] ou [liste d'éléments I ou C], J) }
Début
  PileParcours := PileParcours racine { empiler racine}
  NoeudCourant := racine

```
    tant qu'il existe un lien entre le NoeudCourant.Jonction et une Jonction J faire
        rétracter de la base de données le fait Lien (J, NoeudCourant.Jonction)
        ou Lien (J, NoeudCourant.Jonction)
        si J est déjà dans la Pile de Parcours alors
        {la pile contient une boucle de causalité dont la tête est J}
            Sauvegarder_Boucle_Algébrique(J)
        sinon
            Adapter_Convention(J, L, T)
            Explorer_Arbre(T)
        fsi
    Fait
    {à la sortie de la boucle principale le dernier NoeudCourant correspond à un
    état terminal ou sommet terminal de la Pile de Parcours}
    Traiter_Etat_Terminal (NoeudCourant)
Fin
```

Algorithme 2.4.3

```
Sauvegarder_Boucle_Algébrique (J)
{Sauvegarde de la trace d'une boucle de causalité (éventuelle boucle algébrique).
Cette boucle sera traitée une fois que sa tête correspondra à un état terminal} Début
    Empiler la tête de la boucle algébrique dans PileTête
    Copier la partie de la Pile de Parcours associée à cette boucle
Fin
```

Algorithme 2.4.4

```
Traiter_Etat_Terminal (NoeudCourant)
Début
si NoeudCourant.Jonction est la tête d'une boucle algébrique (PileTête) alors
            Traiter_Boucle_Algébrique (NoeudCourant)
sinon
        si NoeudCourant.Jonction est un élément
            de la dernière boucle algébrique alors
        {former l'information associée à cette jonction à celles des jonctions de
        la boucle déjà traitées }
            Former_Informations_Boucle
```

```
        sinon
            Appliquer_Règles_Statiques (NoeudCourant)
        fsi
    fsi
    Fin {de Traiter_Etat_Pile}
```

Pour écrire les algorithmes correspondant aux procédures Traiter-Boucle-Algébrique, Former-Information-Boucle et Appliquer-Règles-Statiques, nous allons utiliser une règle expertise s'appuyant sur les différentes caractéristiques topologiques d'un modèle bond-graph.

Tout d'abord commençons par illustrer, à l'aide d'un exemple, la démarche que nous allons suivre. Un modèle bond-graph peut avoir un nombre d'éléments en causalité dérivée (sans pour autant savoir lesquels). Et sans indication préalable on ne peut, lors d'une phase d'affectation de la causalité, privilégier un élément sur un autre. Nous qualifions ce problème de "**compétition causale**", il indique qu'une liste d'éléments ou un ensemble de listes d'éléments sont en compétition pour la priorité de telle ou telle causalité.

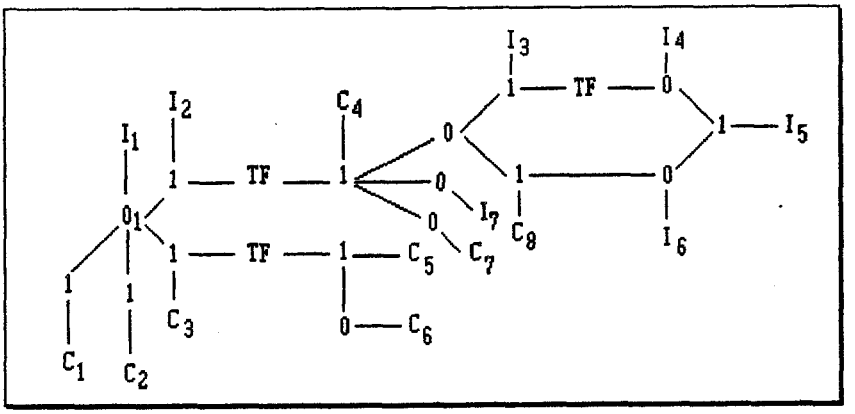


figure 2.11 : exemple de modèle bond-graph possédant des conflits causaux et des compétitions causales

Le tableau suivant donne les listes des éléments de la figure 2.9 associées aux différentes **compétitions causales.**

choisir une dérivée parmi : ([I_3] et [I_4]) ou ([I_4] et [I_5]) d'où à cause des liste simples choisir entre I_3, I_4 et I_5
choisir une dérivée parmi : I_2, I_3 (si il n'est pas choisi précédemment), C_4 , I_7
choisir une dérivée parmi : I_1, C_3 , C_5 , C_6 et entre C1 ou C2

Compétitions causales du modèle bond-graph de la figure 2.11

Le parcours du modèle de la figure 2.11 peut être simulé (selon l'algorithme 2.4) par l'arbre suivant :

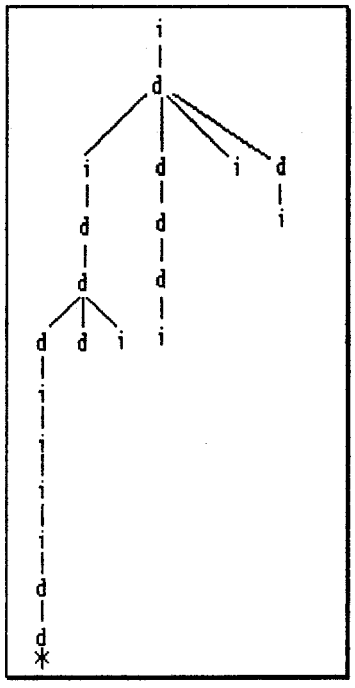


figure 2.12 : arbre simulant le parcours du modèle bond-graph de la figure 2.11

Remarque 2.7 :

Un noeud 'i' avec une liste vide aura toujours soit une jonction-TF soit une jonction-GY.

Ce type de noeud ne rentre jamais en conflit, et à chaque fois qu'il est rencontré pendant la remontée de la pile de parcours on le dépile tout en mémorisant sa présence.

Lors d'une résolution informatique basée sur une exploration en profondeur d'un modèle bond-graph (au sens de l'algorithme 2.4), on doit, pour déceler une compétition causale, soit reconstruire le sous-arbre qui lui est associé, l'explorer et exploiter son information, soit sauvegarder tous les chemins liant la racine aux feuilles de ce sous-arbre et les analyser lorsque la racine devient un état terminal de la pile de parcours. Cette racine doit avoir (à une exception près : lorsqu'elle correspond à celle de l'arbre global(*)) une convention de type 'i'dont la liste est non vide, car les

* Cette exception peut être évitée en initialisant la pile de parcours par une racine fictive de type 'i' : **PileParcours** := [('i',[ ], "").

conflits associés aux descendants d'un noeud de type 'i' ne sont jamais en compétition avec ceux associés à ses ascendants.(*)

Ainsi les différentes compétitions causales vont correspondre aux traitements des sous-arbres suivants, que nous appelons "**arbre de compétition**" :

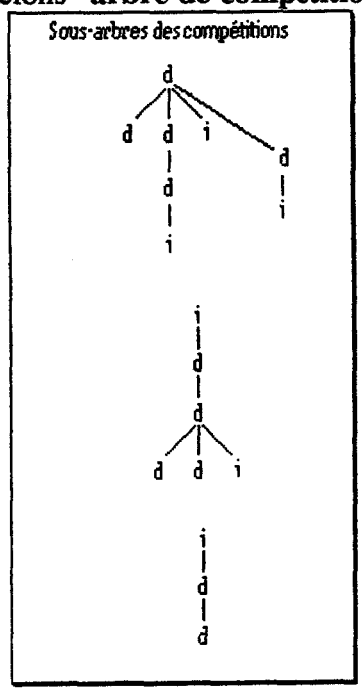


figure 2.13 : exemple d'arbre de compétition

Donc lorsque une racine atteinte, on vérifie s'elle possède une information sur ses descendants (sous forme d'arbre ou de chemins). On construit alors une liste formée à partir d'éléments I et C ou éventuellement des listes d'éléments I ou d'éléments C, en compétition pour une causalité.
Cette liste que nous nommons "**liste des compétitions**" sera analysée selon les règles qui vont suivre.

Son traitement doit tenir compte de toutes les situations possibles. Nous allons les présenter et montrer comment les résoudre.

V.4 Résolution de la liste des compétiotions

Commençons par les cas où les jonctions ont une convention de type 'd', ne possèdent pas d'éléments résistifs (pas de "info_résistive(.)").

---

* Cette constatation constitue l'une des règles que nous allons expliciter dans la suite.

cas n° 1 : $L_{comp} = \{ (Conv, [[E_1, ..., E_n]])\}$

avec E_i de type "I " ou "C" ; Conv de type 'i' ou 'd'

Cette liste sera proposée à l'utilisateur pour qu'il en choisisse l'élément, parmi $[E_1,...,E_n]$, qui doit être en causalité intégrale (respectivement dérivée) **si** Conv='i' (respectivement Conv='d').

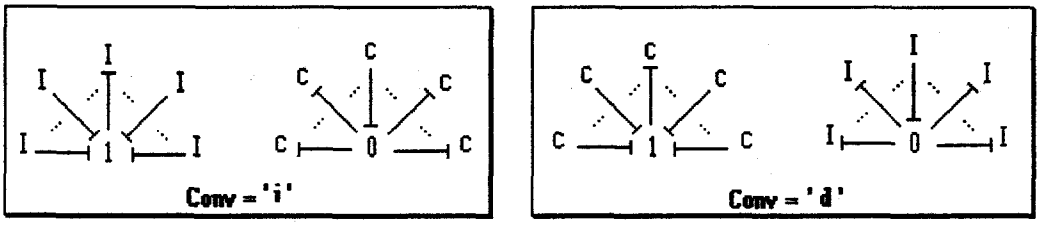


figure 2. 14 : exemples d'une liste de compétition du cas n°1

A noter que dans ce cas, il y a toujours une seule compétition causale. Donc s'il n'y a qu'un seul élément E_i, il prendra alors automatiquement la causalité recherchée.

cas n° 2 : $L_{comp} = \{(Conv,[[E_1, ..., E_i], ..., [E_j, ..., E_k]])\}$

avec E_x de type "I " et/ou de type "C" ; Conv de type 'i' ou 'd'

si Conv='i' **alors** une des sous-listes doit avoir tous ses éléments en causalité dérivée.

Ce cas correspond à un sommet de type 'd' dont la liste d'éléments est vide, dont la jonction ne possède pas d'éléments résistifs et dont tous les fils sont de type 'i'.

Le lien formé par la jonction du sommet et celle de la sous-liste choisie, est le représentant causal de ces deux jonctions.

Pour chacune des autres, on choisit (l'utilisateur ou Archer) l'élément qui doit être en causalité intégrale. Il deviendra alors le représentant de la jonction qui lui est associée.

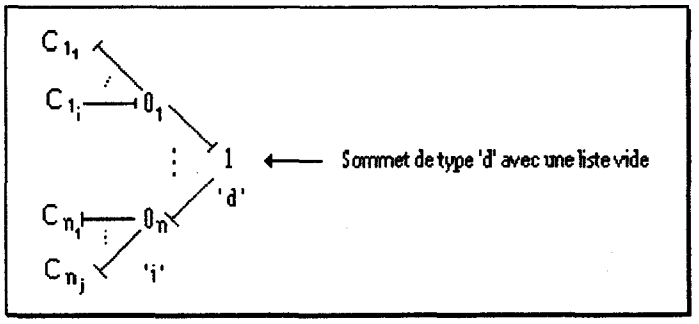


figure 2. 15 : exemple du cas n°1 avec conv='i'

Le nombre de compétitions est égal à celui des sous-listes.

si Conv='d' **alors** {cas où le sommet est obligatoirement un conflit 'd'}

si les jonctions des descendants du sommet sont les mêmes que celle de dernier
alors deux des sous-listes, dont une est obligatoirement associée au sommet,
peuvent garder tous leurs éléments en causalité intégrale.

Les liens formant le chemin entre les deux jonctions associées aux sous-listes choisies, et ayant une place d'ordre impair, sont aussi les représentants causaux de chaque jonction de leurs extrémités.

Pour les autres sous-listes, on cherche pour chacune l'élément qui doit prendre la causalité dérivée. Il est aussi le représentant de sa jonction.

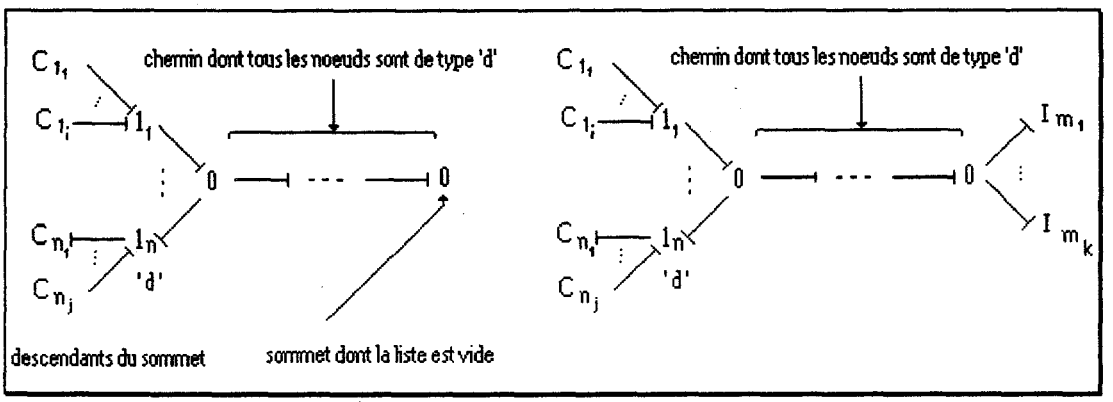


figure 2. 16 : exemple de la 1° possibilité du cas n°2 pour Conv='d'

Dans ce cas le nombre de compétitions est égal à celui des sous-listes **moins 1.**

si les jonctions des descendants du sommet ne sont pas les mêmes, que celle de ce dernier
alors une des sous-listes peut garder tous ses éléments en causalité intégrale.

Si la sous-liste du sommet est choisie **alors** les liens, d'ordre pair, formant le chemin entre la jonction du sommet et celle de la jonction associée au père des noeuds correspondant aux sous-listes des descendants, sont les représentants causaux de leurs extrémités.
sinon les liens, d'ordre impair, formant le chemin entre la jonction de la sous-liste choisie et celle du sommet, sont les représentants causaux de leurs extrémités.

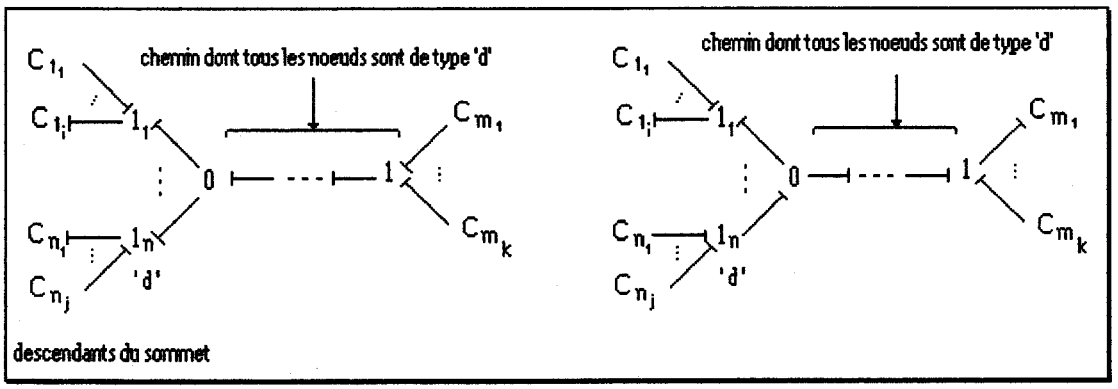


figure 2.17 : deux manière d'affecter la causalité au même modèle bond-graph
cas où les jonctions des descendants et celle du sommet sont duales.

Si la liste du sommet est vide alors une seule possibilité : considère que cette liste a été choisie.

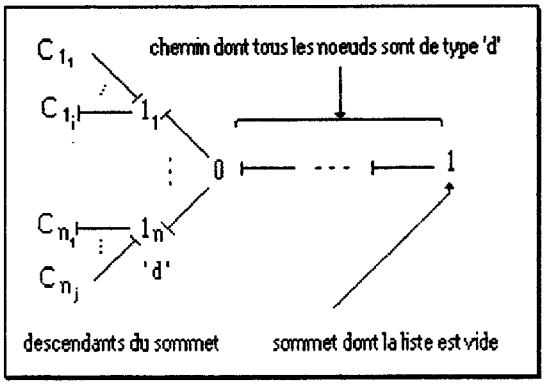


figure 2. 18 : seul cas de la figure 2. si la liste du sommet est vide.

cas n° 3 : (le sommet est de type 'i')

L_{comp} = { ('i', [[S_1, ..., S_i]]), ('d',[[E_h, ..., E_m], ..., [E_n, ..., E_p]])}
avec S_x , E_x de type "I " et/ou de type "C"

Nous distinguons les éléments associés aux sommets par S_x.

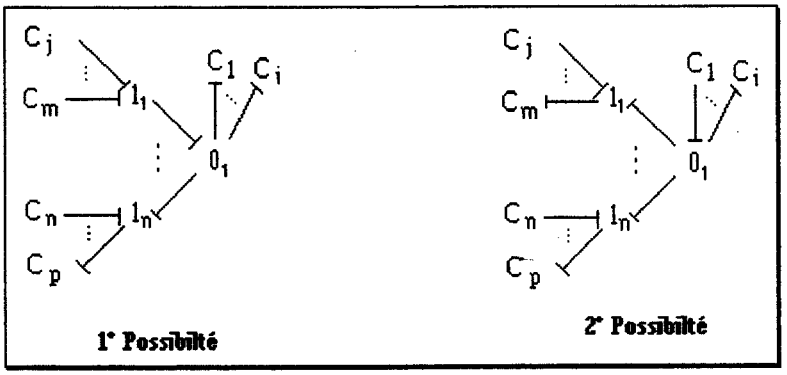


figure 2.19 : Cas où le sommet est de type 'i' et possédant des fils de type 'd'

Comme le montre la figure 2. , on aura deux possibilités à proposer à l'utilisateur :

1° Possibilité :
Une des sous-listes associées aux descendants peut garder la totalité de ses éléments en causalité intégrale.

Le lien joignant la jonction associée à la sous-liste choisie à celle du sommet est un représentant causal de ces deux jonctions. Ceci entraine que la sous-liste associée au sommet aura tous ses éléments en causalité dérivée.

Pour chacune des autres, on détermine le représentant causal de la jonction qui lui est associée.

2° Possibilité : (préférée d' Archer)
Choisir l'élément qui doit prendre la causalité intégrale parmi $[S_1, \ldots, S_i]$. Et pour chacune des autres, choisir celui qui doit avoir la causalité dérivée. Les éléments choisis constituent les représentants causaux des jonctions auxquelles ils sont associés.

Cas n°4 : (le sommet est de type 'd')

$$L_{comp} = \{('d',[\ [S_1, \ldots, S_i], [\ E_1, \ldots, E_i], \ \ldots, [\ E_j, \ldots, E_k]\ ]),$$
$$('i',[\ [\ E_h, \ldots, E_m], \ \ldots, [\ E_n, \ldots, E_p]\ ])\}$$

avec S_x , E_x de type "I " et/ou de type "C"
Nous distinguons les éléments associés aux sommets par S_x.

Plusieurs situations peuvent se produire, celles-ci vont dépendre de la dualité entre les jonctions associées aux descendants de type 'd' et celle du sommet ainsi que de la liste (vide ou pas) de ce dernier.

si ces jonctions sont duales **alors**
Traiter les listes associées aux descendants 'i', c'est-à-dire pour chacune il faut choisir l'élément qui doit prendre la causalité intégrale.

Pour les descendants de type 'd', il faut choisir une sous-liste susceptible de garder tous ses éléments en causalité intégrale. Quant aux autres, il faut déterminer pour chacune le représentant de la jonction qui lui est associé, en choisissant l'élément qui doit prendre la causalité dérivée.

Les liens d'ordre impair par rapport au chemin joignant la jonction associée à la sous-liste choisie et celle du sommet, sont les représentants causaux des jonctions associées à leurs extrémités.

A noter que dans ce cas les éléments de la liste du sommet ($[S_1, ..., S_l]$) auront tous la causalité intégrale.

si ces jonctions sont de même type **alors**

1° Possibilité :

Traiter les listes associées aux descendants 'i', c'est-à-dire pour chacune, il faut choisir l'élément susceptible de prendre la causalité intégrale.

Pour les descendants de type 'd', il faut appliquer le **cas n°2** (avec jonctions duales) à la liste, $L_{comp} = \{('d',[[S_1, ..., S_l], [E_1, ..., E_i], ..., [E_j, ..., E_k]])\}$. Voir aussi le cas où la liste associée au sommet est vide.

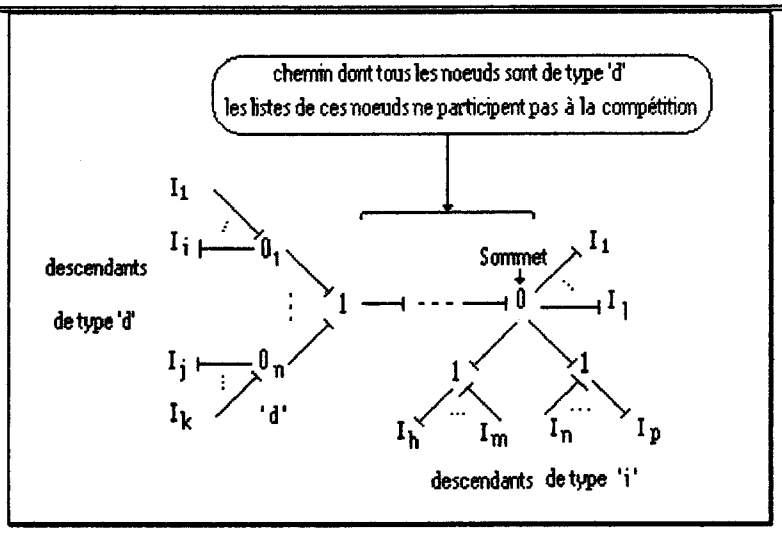


figure2.20 : cas où les jonctions des descendants 'd' sont de même type que celle du sommet.
Exemple de la 1° possibilité , la liste du sommet peut ne pas garder tous éléments en causalité intégrale

2° Possibilité :

-Parmi les sous-listes des descendants de type 'i' choisir celle qui peut garder tous ses éléments en causalité dérivée. Et pour chacune des autres, déterminer l'élément susceptible de prendre la causalité intégrale.

-Parmi les sous-listes des descendants de type 'd' choisir celle qui peut garder tous ses éléments en causalité intégrale. Et pour chacune des autres, déterminer l'élément susceptible de prendre la causalité dérivée.

Cette deuxième possibilité n'est pas affectée si [S_1, ..., S_l] est vide.

Les liens d'ordre impair par rapport au chemin joignant la jonction associée à la sous-liste choisie et celle du sommet, sont les représentants causaux des jonctions associées à leurs extrémités.

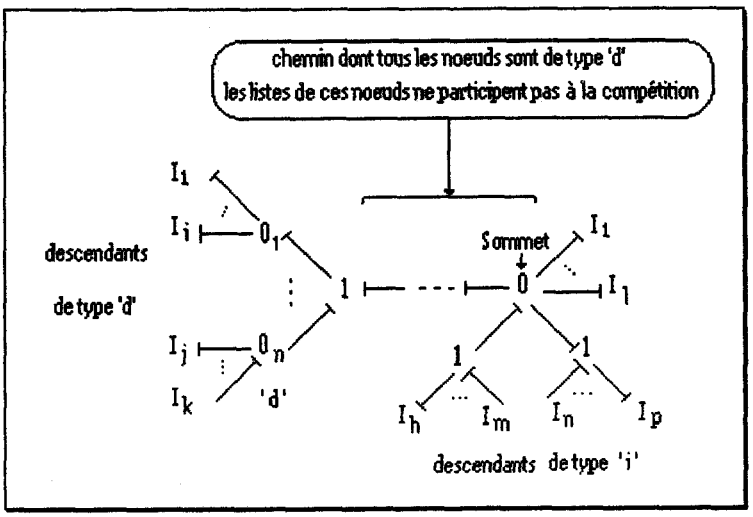


figure2. 21: cas où les jonctions des descendants 'd' sont de même type que celle du sommet. Exemple de la 2° possibilité avec la liste du sommet qui garde tous ses éléments en causalité intégrale

Remarques 2.8

-Les possibilités doivent être interprétées pour être proposées simultanément à l'utilisateur.

-Lorsque les listes d'éléments sont simples (cas le plus fréquent pour les modèles bond-graphs), on peut concaténer celles formant la liste des compétitions et avoir ainsi un seul choix à faire.

-Les possibilités proposées vont aboutir au même nombre de liens en causalité intégrale. Autrement dit, elles donneront un nombre identique maximisant la causalité souhaitée.

-Les noeuds dont les listes sont vides auront la priorité par rapport aux autres.

Dans le cas où des jonctions possédant une causalité de type 'd' ont aussi une information sur des éléments résistifs "info_résiste(.)", nous passerons directement aux cas n°3 et n°4 pour montrer comment tenir compte de ce nouveau paramètre.

Avant d'appliquer les règles du cas n°3, il faut commencer par éliminer de la liste des compétitions les sous-listes de type 'd' dont les jonctions associées possèdent cette information.

Pour le cas n°4 , il faut appliquer les règles suivantes :

si la jonction du sommet et celle de ses descendants de type 'd' sont duales **alors**

Traiter les listes associées aux descendants 'i', c'est-à-dire pour chacune, il faut choisir l'élément qui doit prendre la causalité intégrale.

Pour les descendants de type 'd', il faut choisir une sous-liste susceptible de garder tous ses éléments en causalité intégrale (de préférence associée à une jonction ne possédant pas d'éléments résistifs). Quant aux autres, il faut déterminer, pour chacune de celles dont les jonctions ne possèdent pas une information de type "info_résistive(.)", le représentant de la jonction qui lui est associée, en choisissant l'élément qui doit prendre la causalité dérivée.

Les liens d'ordre impair par rapport au chemin joignant la jonction associée à la sous-liste choisie et celle du sommet, sont les représentants causaux des jonctions associées à leurs extrémités.

A noter que là aussi, les éléments de la liste du sommet ($[S_1, \ldots, S_l]$) auront tous la causalité intégrale.

si la jonction du sommet et celles de ses descendants de type 'd' sont les mêmes **alors**

si les jonctions des sous-listes de $\{[S_1,\ldots,S_l],[E_1,\ldots,E_i],\ldots,[E_j,\ldots,E_k]\}$ possèdent toutes l'information "info_résistive(.)" **alors**

-Traiter les listes des descendants 'i' une à une en choisissant dans chacune l'élément qui doit prendre la causalité intégrale.

-Une des listes résistives doit garder tous ses éléments en causalité "effort entrant". Pour chacune des autres, il faut en choisir une causalité "effort sortant".

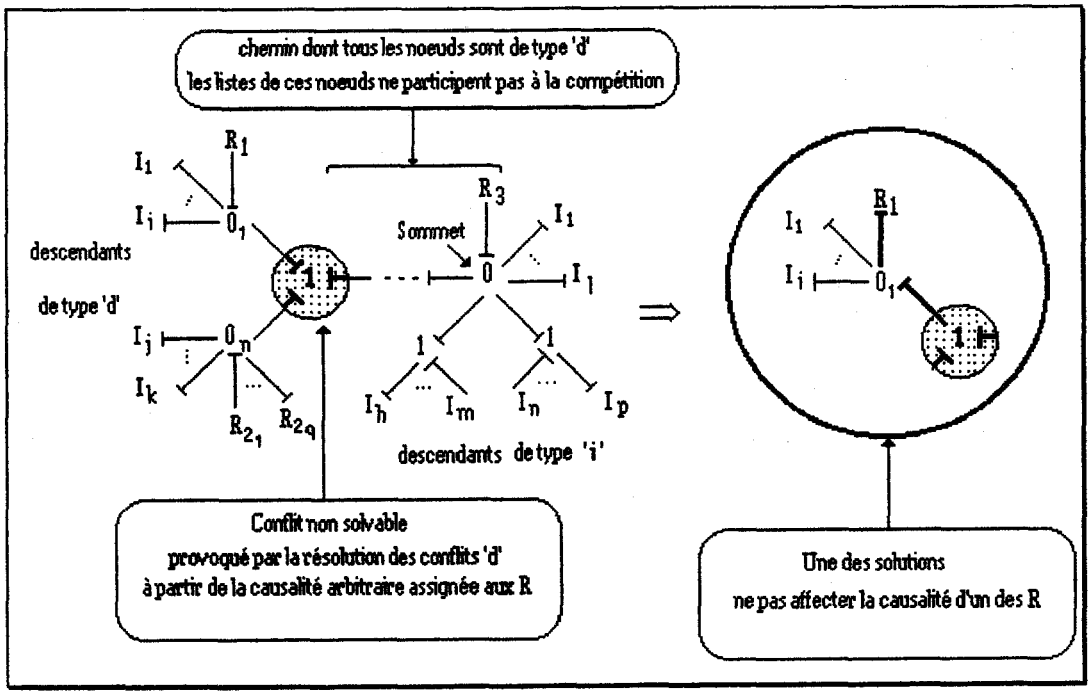


figure 2.22 : cas d'un conflit provoqué par la résolution, à partir d'éléments R , d'une compétition causale

si les jonctions des sous-listes de $\{[S_1,\ldots,S_l],[E_1,\ldots,E_i],\ldots,[E_j,\ldots,E_k]\}$

ne possèdent pas toutes l'information "info_résistive(.)" **alors**

-Traiter les listes des descendants 'i' une à une en choisissant dans chacune l'élément qui doit prendre la causalité intégrale.

-Pour chacune des sous-listes ne possédant pas cette information, choisir l'élément susceptible de prendre la causalité dérivée.

- Pour chacune des listes résistives ; choisir une causalité "effort sortant".

Remarques 2.9

-Les listes des éléments R seront proposées à l'utilisateur pour qu'il choisisse, en fonction du cas, l'élément qui doit prendre la causalité "effort entrant" ou "effort sortant". L'élément choisi correspondra au représentant causal de sa jonction.

-Lorsqu'une des listes résistives ne doit pas être utilisée lors de la résolution d'un conflit (cf. figure 2.22), on propose alors à l'utilisateur de participer au choix de celle-ci.

-Dans Archer, on ne propose pas à l'utilisateur des éléments bond-graph mais les éléments physiques qui leur sont associés. De même les termes de type "causalité intégrale, causalité dérivée, effort entrant, effort sortant" ne sont utilisés qu'à titre indicatif. En revanche, nous proposons le module que peut avoir l'élément physique.

Par exemple, soient des éléments I_1, I_2, ..., I_n représentant des masses M_1, M_2, ..., M_n, nous proposons :
seules une masse parmi [M_1, M_2, ..., M_n] pourra avoir une causalité

$flux = \Phi_{Mi}(\int effort)$, si on cherche une causalité intégrale.

$effort = \Phi_{Mi}^{-1}\left(\frac{dflux}{dt}\right)$, si on cherche une causalité dérivée.

L'exploitation informatique peut être très complexe, en raison de la taille de l'information sauvegardée (certains modèles bond-graph peuvent générer de grands arbres de compétitions possédant plusieurs niveaux en profondeur et en largeur) et du temps associé à son traitement. Donc une sorte de redondance inévitable.

La taille et la forme de la liste de compétition peuvent rendre le dialogue avec l'utilisateur très ambigu, à cause du nombre important d'informations véhiculées. Or certains éléments peuvent ne pas avoir de rapport direct dans le modèle physique. C'est à dire ils peuvent être associés à des éléments physiques complètement disjoints et éloignés les uns des autres. Donc leur appartenance à la même liste de compétition ne fera qu'aggraver l'ambiguïté du dialogue.

Nous avons cherché alors à optimiser les phases de la démarche précédente, par l'utilisation d'**heuristiques**. Le nombre d'éléments en causalité dérivée et en causalité intégrale (avec maximisation de la nature de la causalité -intégrale ou dérivée- souhaitée) est le même. Cependant le modèle bond-graph causal peut parfois être différent (certains éléments I ou C n'ont pas forcément la même causalité) par l'une ou par l'autre démarche.

La taille de la liste de compétition causale pourra être largement réduite par rapport au cas précédent, ce qui rendra cette liste exploitable dans une phase d'échange avec l'utilisateur.

Du point de vue informatique, il n'est pas indispensable de construire la liste des compétitions, car il suffit d'exploiter directement les paramètres du sommet et/ou ceux de l'information sur ses descendants. Nous verrons plus loin que parfois on tient compte seulement des information sur les descendants du sommet.

En résumé, au lieu de s'intéresser aux sous-arbres de niveaux quelconques nous ne tiendrons compte que des sous-arbres de niveaux inférieur ou égal à 3.
Notons que cette démarche se prête bien (contrairement à la première) aux traitements

des boucles de causalité entre des jonctions.

Deux autres points nous ont incités à réduire le niveau de l'arbre de compétition causale (utile pour la détermination des modèles mathématiques et pour l'analyse structurelle des bond-graphes) : la production de courts chemins causaux directs et la réduction du nombre de boucles causales.

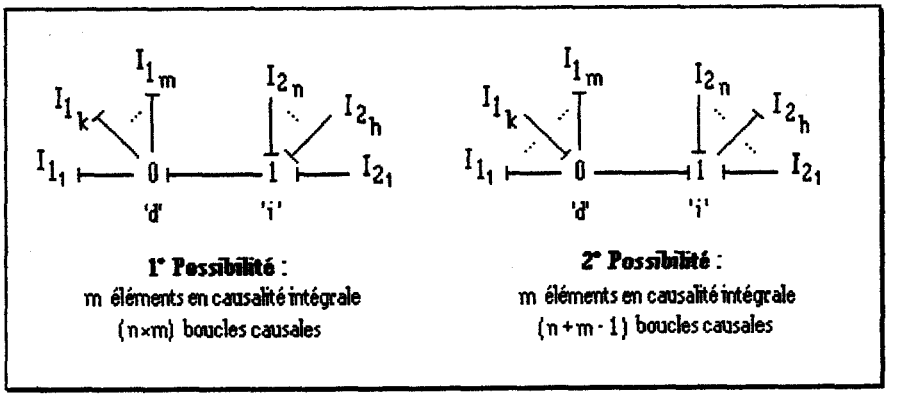


figure 2.23 : exemple de réduction de nombre de boucles causales

Nous arrivons maintenant à la présentation des règles de production qui génèrent, à partir d'un état terminal de la pile, soit la constitution des données pour la liste des compétitions, soit la résolution de celle-ci.

V.5 Règles statiques de production

Ces règles nécessitent la connaissance du type de convention des 3 premiers éléments (partant du sommet) de la pile de parcours et éventuellement de la trace de l'information associée soit aux fils, soit aux petits fils du sommet (état terminal).

Rappelons les conventions associées à un noeud :

- 'i' pour indiquer que la jonction-0 (respectivement jonction-1) de l'élément de la pile (ou du noeud) possède un ou plusieurs éléments de type C (respectivement de type I).

- 'd' pour indiquer que la jonction-0 (respectivement jonction-1) ne possède pas d'élément de type C (respectivement de type I).

Déterminons deux autres types de conventions :

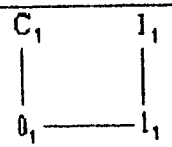
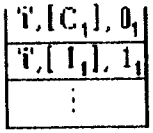
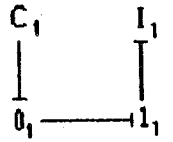
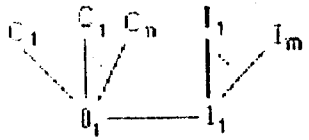
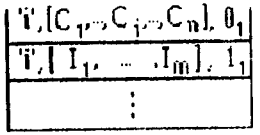
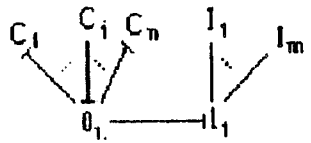
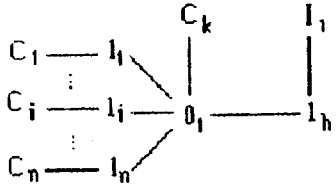
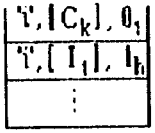
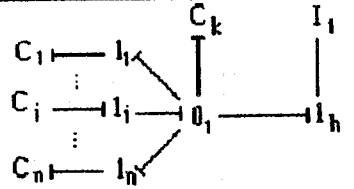
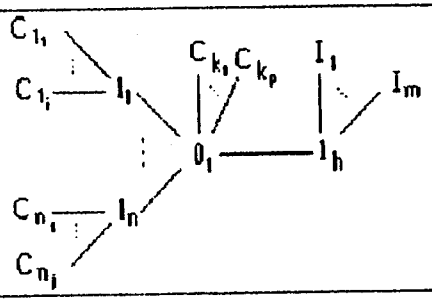
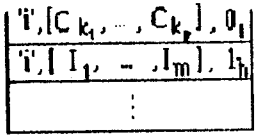
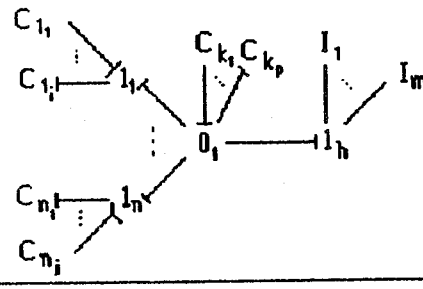
- 'n' pour initialiser la pile de parcours : PileParcours := [('n', [ ],""), ('n', [ ],"")]. Initialisation à deux niveaux de type 'n' pour faciliter la récursivité du traitement. On

peut ainsi traiter trois niveaux à la fois sans avoir à s'occuper, à chaque pas, du fond de la pile. Et lorsqu'on atteint un noeud dont le père et le grand-père sont de type 'n', on arrête le traitement.

- 's', ancienne convention de type 'd' transformée par les règles de production, indique que la jonction associée au noeud est "solvable". C'est-à-dire que la liste des éléments I ou C, associée à la jonction qu'elle représente, aura la même causalité (intégrale ou dérivée) voulue par l'utilisateur, et que son représentant causal sera déterminé par une déduction causale.

Le nombre de combinaisons possibles, de longueur 3, formé à partir des quatre indices est 4^3, soit 64 combinaisons. Nous verrons que certaines situations (ou combinaisons) ne peuvent jamais se présenter, d'autres peuvent être combinées en une seule ...

tableau 2.3

Cas des éléments	Exemple d'une situation bond-graph	Etatde la Pile de Parcours	Informations sur les descendants : fils ou petits-fils	Conftruction de la liste de compétition "L_{comp}"	Solutions Possibles
éléments simples			Rien	$L_{comp} = \{(i', [C_1])\}$	
éléments multiples			Rien	$L_{comp} = \{(i', [C_1, \ldots, C_i, \ldots, C_n])\}$	
éléments simples			$(d, [I_1, \ldots, I_n], [C_1, \ldots, C_i, \ldots, C_n])$	$L_{comp} = \{(i', [C_k]) \; (d, [C_1, \ldots, C_i, \ldots, C_n])\}$ $\Updownarrow$ $L_{comp} = \{(i', [C_k, C_1, \ldots, C_i, \ldots, C_n])\}$	
éléments multiples			$(d, [I_1, \ldots, I_n]$ $[[C_1, \ldots, C_{i_1}]$ $\vdots$ $[C_{n_1}, \ldots, C_{n_j}]])$	$L_{comp} = \{(i', [C_{k_1}, \ldots, C_{k_p}]) \; (d, [[C_1, \ldots, C_{i_1}] \; \vdots \; [C_{n_1}, \ldots, C_{n_j}]])\}$	

A partir du tableau 2.3 nous pouvons déduire la règle suivante :

Règle n°1

si l'état terminal ainsi que son père ont une convention de type 'i'
ou la convention du père es t de type 'n'
alors
-construire la liste des compétitions à partir du sommet et de l'information, si elle existe, sur ses fils de type 'd'.
- traiter la liste des compétitions
- dépiler le sommet

Règle n°2

si la convention de l'état terminal est de type 'i' et possède une information sur ses descendants
et celle de son père est de type 'd'.
alors
- construire la liste des compétitions à partir de cette information
- traiter la liste des compétitions
-dépiler le sommet et l'ajouter dans la liste formée par les descendants de type 'i' du sous sommet (son père) .

La solution la plus complète consiste à tenir compte de la liste associée au sommet dans les compétitions. Et si après la résolution des compétitions un élément de cette liste représente la jonction du sommet alors on ajoute, quand même, ce dernier à la liste des descendants de type 'i' de son père, sinon on dépile tout simplement.

Remarques 2.10 :

-Nous avons préféré de ne pas proposer à l'utilisateur une liste qui aurait déjà fait l'objet d'un choix précédent. Nous rendons ainsi le dialogue "Homme-Machine" moins redondant et plus logique.

-Si on tient compte du sommet, on formera une liste de compétition dont la résolution se fait à partir des règles du cas n°3. On y trouve la possibilité suivante : "deux listes, dont celle du sommet, peuvent garder tous leurs élément en causalité intégrale". Cette dernière va générer un nombre plus important de boucles causales, et moins de courts chemins causaux.

-Nous aurions pu, après avoir traité la liste du sommet dans les compétitions, ne pas la garder pour concourir avec celles de son père, et donc ne pas la proposer une

deuxième fois à l'utilisateur. Mais si jamais le noeud père ne possède pas d'autre descendant, et que sa jonction n'a pas d'élément résistif (cf. figure 2.24), on provoquera alors un conflit non solvable directement. Donc un mauvais modèle causal.

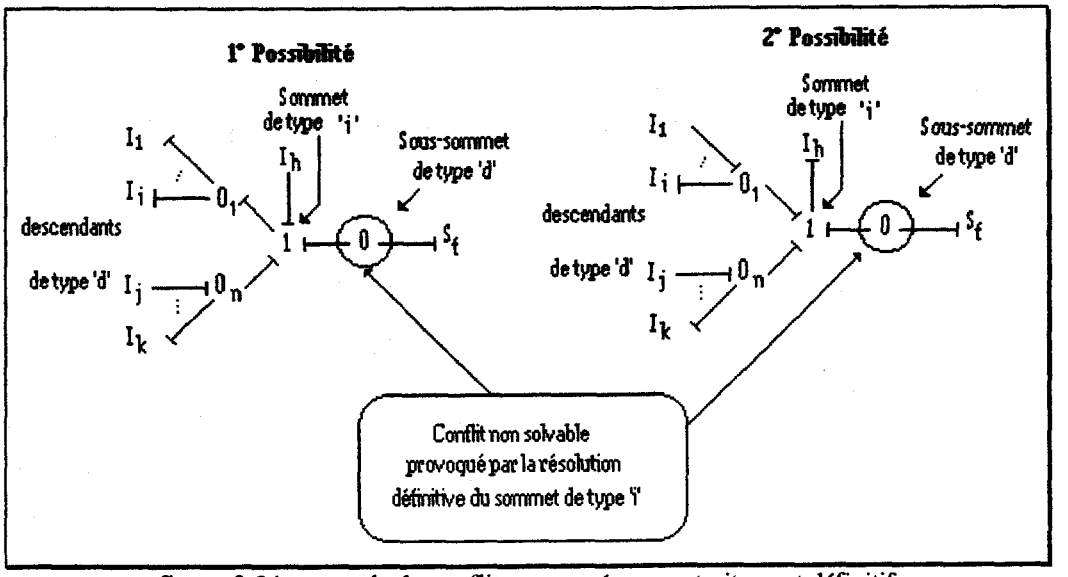


figure 2.24: exemple de conflit provoqué par un traitement définitif d'un sommet de type 'i' et dont le père est de type 'd'.

Règle n°3

si la convention de l'état terminal est de type 'i' et ne possède pas d'information sur ses descendants et celle de son père est de type 'd'. **alors** dépiler le sommet et l'ajouter dans la liste formée par les descendants de type 'i' du sous-sommet (son père) .

Passons maintenant au cas où le sommet est de type 'd'.

Règle n°4

si la convention de l'état terminal est de type 'd' **et** possède des informations sauvegardées sur ses descendants **et** celle de son père est de type 'i' **alors** - dépiler le sommet -construire la liste des compétitions à partir du sommet et des informations sur ses descendants. - traiter la liste des compétitions

La solution la plus complète consisterait (lorsque le représentant causal de la jonction du sommet est parmi les éléments de la liste qui lui est associée) à sauvegarder l'information du sommet avec celles des descendants de son père. Pour la même raison qu'antérieurement, nous avons préféré ne pas proposer une liste plus d'une

fois.

A l'inverse du cas précédent, on tiendra compte du sommet dans la compétition causale, pour éviter tout problème. D'autant plus que le risque est majoré, si la liste associée au sommet est vide.

Règle n°5

si la convention de l'état terminal est de type 'd' **et** ne possède pas d'informations sur ses descendants. **et** celle de son père est de type 'i'

alors -dépiler le sommet et l'ajouter dans la liste formée par les descendants de type 'd' du sous-sommet (son père) .

Règle n°6

si l'état terminal, ainsi que son père et son grand-père ont une convention de type 'd'

et la jonction de cet état ainsi que celles de ses descendants de type 'd' sont de même nature.

et (la liste de l'état terminal est non vide **ou** vide mais sa jonction possède l'information "info_résistive(;)")

alors - dépiler le sommet et l'ajouter à la liste des descendants, de type 'd', de son grand-père.

- construire la liste des compétitions à partir des descendants du sommet.

- traiter la liste des compétitions.

-modifier la convention du nouveau sommet , en remplaçant le 'd' par un 's'.

Lorsqu'un noeud de type 'd' possède des informations sur ses descendants, il y a en principe une compétition entre lui et ses descendants. Faut-il forcément inclure le sommet dans cette compétition ? Les cas de la figure 2.26 apportent des éléments de réponses à cette question.

La participation du sommet aux compétitions comme le montre la figure 2.6 .a et 2.6 .b va induire une perte de deux causalité intégrale. D'où par précaution vaut mieux régler le problème des compétitions uniquement à partir des descendants du sommet et ajouter ce dernier à la liste de type 'd' associée aux descendants de son grand-père.

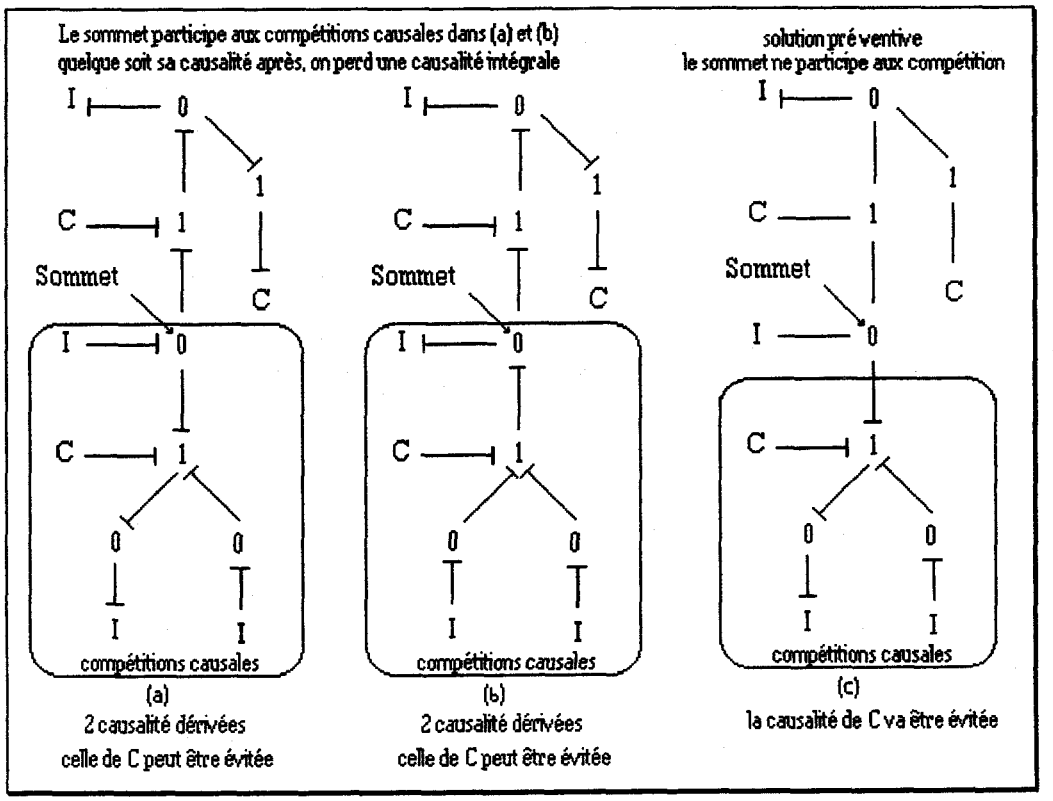


figure2.26

Remarque2.11 :

Les règles suivantes vont être annonçées sans commentaires ni figures, cependant on peut les vérifier sur des exemples de modèles bond-graphs.

Règle n°7

si l'état terminal, ainsi que son père et son grand-père ont une convention de type 'd'

et (la jonction de cet état, ainsi que celles de ses descendants de type 'd' sont duales.

ou (ces jonctions sont les mêmes **et** la liste de l'état terminal est vide mais sa jonction ne possède pas d'éléments résistifs))

alors - construire la liste des compétitions à partir du sommet et de ses descendants.

traiter la liste des compétitions.

Règle n°8

si l'état terminal ainsi que son père ont une convention de type 'd'

et celle de son grand-père est différente de 'd'

alors - construire la liste des compétitions à partir des descendant du sommet

- traiter la liste des compétitions

- dépiler le sommet et l'ajouter dans la liste formée par les descendants de type 'd' du sous-sommet (son père) .

Règle n°9

si l'état terminal ainsi que son grand-père ont une convention de type 'd'
et celle de son père est de type 's' et possède une information sur ses descendants
alors
- dépiler le sommet
- construire la liste des compétitions à partir du sommet et ses descendants
- traiter la liste des compétitions

Règle n°10

si l'état terminal ainsi que son grand-père ont une convention de type 'd'
et celle de son père est de type 's' et ne possède pas d' information sur ses descendants
alors
- dépiler le sommet et l'ajouter dans la liste formée par les descendants de type 'd' de son grand-père.

Règle n°11

si la convention de l'état terminal est de type 's'
et celle de son père est de type 'd'
alors dépiler le sommet

Règle n°12

si la convention de l'état terminal est de type 'd' ou 'i'
et celles de son père et de son grand père sont de type 'n'
alors
- dépiler le sommet
- construire la liste des compétitions à partir du sommet et ses descendants
- traiter la liste des compétitions

Les règles associées à la résolution de la liste de compétitions citées plus haut restent valables ici.

Algorithme 2.4.5

Si on écrit l'algorithme suivant d'une manière classique -proche d'une programmation structurée- il aura une taille très importante et sera illisible par le nombre de tests alternatifs imbriqués. D'où une écriture proche de la programmation logique. Son

principe est basé sur le règles précédentes, il suffit alors de les adapter à tel ou tel langage pour une implantation sur ordinateur.

Quant à nous, nous avons utilisé, pour Archer, une implantation sous forme de prédicat Prolog. Donc une inférence basée sur un chaînage arrière. Cependant nous pensons que la technique des "Frames" est la mieux appropriée à ce type de problème.

Appliquer_Règles_Statiques(NoeudCourant)
Début
Dépiler le sommet de la Pile de Parcours : ($Conv_1$, L_1, J_1)
Consulter le sommet courant : ($Conv_2$, L_2, J_2)
Consulter le sous-sommet : ($Conv_3$, _, _)
{la connaissance de L_3 et J_3 n'est pas importante pour les règles}
Vérifier, en fonction des conventions, si le sommet possède des informations sur ses de connaissances.
{Ce fait déclenchera une des règles statiques}
...
Fin

V.4 Règles dynamiques

Ces règles seront utilisées pendant la phase d'échange de dialogue avec l'utilisateur. Elles vont permettre de l'aider à faire un choix qui s'accorde avec les méthodes de la modélisation et de l'analyse.

Par exemple, si on doit choisir parmi une liste d'éléments de type résistif, et prendre la causalité "effort entrant", il vaut mieux affecter celle-ci à l'élément R qui a le plus petit module, et inversement. Ainsi les gains des boucles associées à cet élément auront un module inférieur à 1.

De la même façon on peut éviter selon des principes exposés dans Van Dijk J., Breedveld [1991] les boucles causales d'ordre 0, qui constituent un prolongement du cas de la boucles des boucles algébriques entre des R. En effet se sont des boucles formées par des éléments de même type (entre deux I, ou deux C, ou deux R). Ce type de boucles peut provoquer des problèmes algébriques pendant le traitement numériques du modèle.

L'information sur la présence d'une source dans la liste d'une jonction, sera utilisée

pour permettre d'affecter la causalité recherchée à l'élément qui doit être directement commandable. De la même manière, on peut sauvegarder l'information sur les capteurs pour aider à choisir l'élément qui doit être directement observable.

A ce niveau, l'utilisateur peut lui même ajouter d'autres règles concernant la stabilité, et la réduction des modèles, etc.

Nous avons vu précédemment que lorsque Archer est sollicité pour choisir, il va -bien entendu- chercher à vérifier si les conditions des deux règles précédentes sont satisfaites, sinon il va privilégier les solutions qui réduisent à la fois le nombre des boucles causales et la taille des chemins causaux (cf. figure 2.23) .

Une autre règle de ce type va permettre d'éviter des boucles de causalité, non solvables, entre des jonctions. Cette règle se base sur les résultats de R.C. Rosenberg et A. N. Andry qui donnent les conditions de la solvabilité de la structure de jonction d'un bond-graph contenant des boucles, ainsi que sur ceux de F. Lorenz et M. Wlasowski (Nous allons la retrouver dans la suite).

V.5 Résolution des boucles algébriques

Pour terminer cette partie, nous allons nous intéresser aux boucles de causalité entre les jonctions. Les exemples suivants montrent les cas de ce type de boucles :

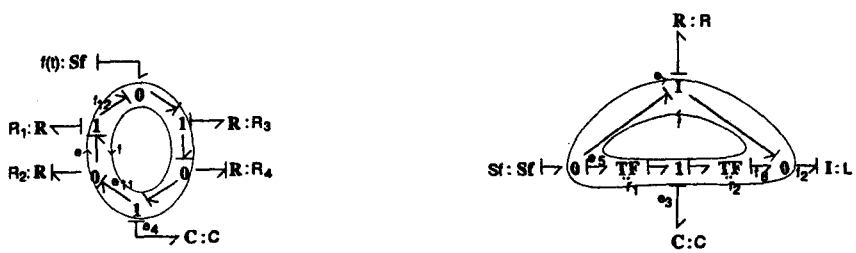


figure2.27 : cas de boucles entre des jonctions
nommées aussi boucles de causalité ou chemin causal clos

R.C. Rosenberg et A. N. Andry [1979] ont exposé les conditions de la solvabilité de ce type de boucle à partir de la causalité et du concept de gain de boucles causales (cf. 2° partie de ce chapitre). Nous nous contentons, sans rentrer dans les détails, de donner la conclusion de cette étude :

La structure de jonction d'un bond-graph contenant des boucles est solvable si le gain de chacune d'elles est différent de +1.

Cette étude a été reprise par F. Lorenz et M. Wlasowski [1989] en associant à chaque groupe de boucles adjacentes, d'une structure de jonctions, un déterminant calculé par application des règles de Mason (Brown [1972]).

Cette structure est solvable (c'est-à-dire sans boucle algébrique) si et seulement si ce déterminant est +1.

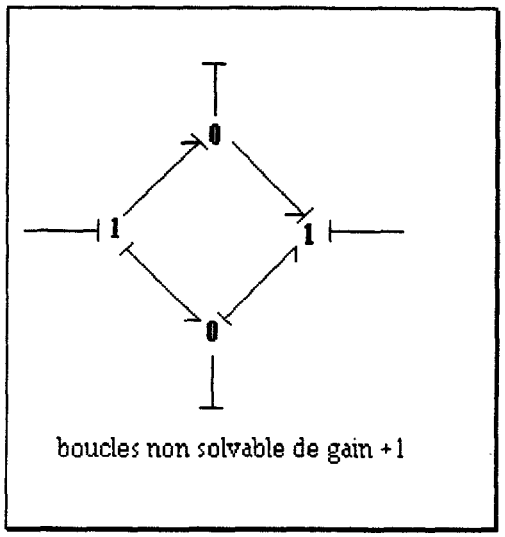


figure 2.28

La première solution est la mieux adaptée à notre méthode, puisqu'elle s'applique aux boucles, formées par des jonctions, une à la fois ; alors que la seconde les prend en considération simultanément.

Pendant l'exploration du graphe, la rencontre d'un noeud appartenant à la pile de parcours (cf. Algorithme 2.4.2) indiquera la présence de ce genre de boucle. La partie de la pile correspondant à cette boucle sera alors sauvegardée (cf. Algorithme 2.4.3).

Lorsqu'un état terminal appartient à la dernière boucle mémorisée, alors il sera traité suivant les règles statiques et l'information qu'il contient sera analysée selon l'algorithme suivant :

Sauvegarder_Boucle_de_causalité(Noeud terminal)

Début
Si Noeud courant a une jonction de type TF ou GY **alors**
Multiplier par le module correspondant à cette jonction le gain de la partie de la boucle déjà traitée .
fsi
Si ce noeud est de type 'd' ou 'i' **alors** Incrémenter de 1 le nombre de noeuds de ce type associés à cette boucle **fsi**
Fin

Lorsque la tête de cette boucle correspond à un état terminal, les renseignements qui lui sont associés seront analysés selon les règles de l'algorithme suivant :

Analyser_Boucle_de_causalité

Début
Si Existe au moins un noeud de type 'i' faisant partie de la boucle
ou le nombre de noeud de type 'd' est impair
ou le signe de cette boucle est négatif
ou le nombre de noeuds de type 'd' est pair avec la jonction d'au moins un noeud possédant une liste d'éléments résistifs "info_résistive(.)"
ou la boucle contient des jonctions TF et/ou GY dont le produit des modules est différent de 1
alors la boucle est solvable
sinon la boucle est non solvable,
{pour la résoudre, un noeud et un seul (nécessairement de type 'd') ne doit pas garder la totalité de ses éléments en causalité intégrale.}
Fin

Remarques 2.12 :

-Nous verrons dans la deuxième partie comment déterminer le signe d'une boucle. Notons que ce signe ne dépend pas de la causalité mais seulement de l'orientation des demi-flèches autour d'une jonction.

-Pour savoir si le gain est différent de 1, on utilise celui formé dans la phase "sauvegarde des information sur la boucle".

Quant à Archer, le fait qu'il ne manipule pas des données numériques l'oblige à

demander à l'utilisateur si ce gain est différent de 1. Si l'utilisateur n'arrive pas à se décider, Archer considère que la boucle est solvable et enregistre la condition de cette solvabilité dans un fichier associé aux particularités(*) du modèle étudié.

-Lorsqu'un un noeud ne peut pas garder la totalité de ses éléments en causalité intégrale, Archer va choisir celui représentant la tête de cette boucle. Car d'une part, les autres seraient déjà traités et les remettre en cause compliquera le processus ; d'autre part le noeud choisi peut ne pas perdre sa causalité intégrale s'il possède un père de type 'd'. Donc on a intérêt à le remettre dans la pile pour le retraiter.

Conclusion

Nous avons présenté le long de cette partie des méthodes algorithmiques de l'affectation de causalité. Les méthodes existantes, à savoir celle de D. Karnopp et R.C. Rosenberg [1987] et celle de S.J. Hood et al. [1989], ne tiennent pas compte de certains paramètres comme la nature des modules des éléments et leurs lois physiques. Le fait de ne pas tenir compte de l'avis de l'utilisateur ne permettra pas une bonne affectation automatique, car celle-ci produira toujours le même bond-graph. Donc on aura toujours les mêmes modèles mathématiques et la même analyse.

Conscient de l'intérêt que représente la causalité dans la modélisation et l'analyse des systèmes dynamiques à partir des méthodologies bond-graphs, nous avons cherché une méthode qui soit en accord avec les résultats existant dans la littérature bond-graph.

Se basant sur des règles d'expertise, la méthode que nous avons proposée permettra, dès la phase de l'affectation causale, d'apporter une sorte d'analyse préalable du modèle étudié.

Elle associe l'utilisateur à toutes ses phases, mais elle peut également prendre en charge toutes les opérations. Et comme elle tiens compte de toutes les éventualités, elle peut paraître compliquée. D'autant plus que le nombre de cas que permet la structure bond-graph (surtout pour les modèles théoriques) est important. En plus, tout programme informatique complet se doit de généraliser son traitement à tous les cas de figure. C'est pourquoi certains de nos exemples ne représenteront peut-être jamais un modèle physique.

---

* ce fichier va contenir toutes les informations associées au modèle étudié au fur et à mesure qu'on exécute une étape d'Archer. Entre autres il y aura le rang du système, ses entrées/sorties, ses domaines physiques, ses éléments directement commandables ou observables, etc.

2° Partie : Informations causales

I. Introduction

Après la phase de l'affectation de la causalité, nous pouvons entamer celles de la modélisation, de l'analyse structurelle ...

Du point de vue pratique, on commence par explorer le modèle bond-graph et extraire les informations sur ses chemins causaux, ses boucles causales etc... Ceci va nous permettre, par application d'un certain nombre de méthodes, d'atteindre les objectifs escomptés.

Lors d'une implantation informatique, nous allons suivre la même démarche en utilisant des représentations symboliques et des techniques algorithmiques (Azmani et al. [1990]). Les méthodes que nous allons proposer dans le cadre de ce travail, utilisent une représentation de données proche de celle de "Prolog" (mais facilement adaptable à d'autres langages) et des techniques d'exploration, que nous avons adaptées au cas d'un bond-graph, rencontrées dans la théorie des graphes.

A cause de ses deux sens de l'orientation causale (effort et flux), le bond-graph correspond en réalité à un di-graphe. Ces deux sens de parcours peuvent être rencontrés dans un même chemin. Ce qui pose un problème quand il s'agit d'un parcours quelconque, car on ne sait pas quel sens on doit privilégier et quelle représentation on doit choisir pour retrouver toutes les informations causales, sans pour autant dépendre d'un sens de parcours ou d'un autre.

Les réponses à ces préoccupations feront l'objet des méthodes que nous verrons par la suite. Mais commençons par formuler les définitions associées aux informations causales.

II. Rappels et définitions

Les définitions que nous allons donner ici constituent une synthèse formée à partir de R.C. Rosenberg et A.N. Andry [1979] concernant les parcours associés à un chemin causal, de F.T. Brown [1972] pour les différents types de boucles causales susceptibles d'exister dans un modèle bond-graph ainsi que N. Suda et T. Hatanaka [1986] pour la relation qui existe entre le signe d'un chemin et l'orientation des demi-flèches.

II.1 Chemins causaux

Définition 2.1

Un chemin causal direct est une séquence d'alternance de liens et de jonctions liant, selon l'orientation causale, un élément X de type {S_e, S_f, I, C, R} à un élément Y de type {I, C, R}.

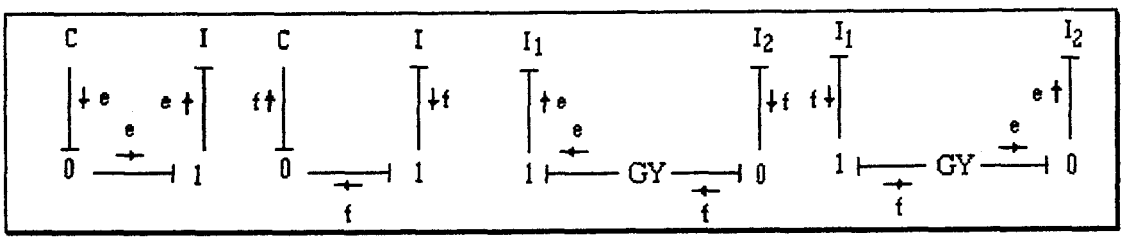
Définition 2.2

Un chemin causal indirect est formé à partir d'une combinaison d'au moins deux chemins causaux directs.

Tout type de chemin causal liant deux éléments X et Y, possède exactement deux parcours causaux : en partant de X pour atteindre Y et en partant de Y pour atteindre X.

On distingue 3 types de parcours causaux :

1) un parcours "**flux**", où seuls les variables flux apparaissent.
2) un parcours "**effort**", où seuls les variables effort apparaissent.
3) un parcours "**mixte**", où il y a à la fois les variables effort et flux.



figures 2.29 différentes manières de parcourir un chemin causal.

Propriétés 2.1

a) Un chemin causal direct sans gyrateurs (jonction GY) possède un parcours flux et un parcours effort.

b) Un chemin causal direct avec gyrateurs possède deux parcours mixtes.

c) Un chemin causal indirect possède nécessairement deux parcours mixtes.

Définition 2.3

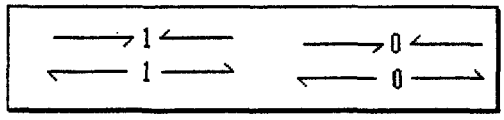
Le signe d'un chemin causal est le produit des signes de ses différents parcours, il est calculé par :

$$\sigma = (-1)^{n0+n1}$$

avec :

n_0= nombre de changements d'orientation des demi-flèches autour d'une jonction-0 lorsque le parcours du chemin causal se fait dans le sens de la variable flux.

n_1= nombre de changements d'orientation des demi-flèches autour d'une jonction-1 lorsque le parcours du chemin causal se fait dans le sens de la variable effort.



figures 3.30 : cas des changements d'orientation dans les demi-flèches autour d'une jonction 0 ou 1.

Lorsque le parcours est de type :

"flux" : $\sigma = (-1)^{n_0}$
"effort" : $\sigma = (-1)^{n_1}$
"mixte" : $\sigma = (-1)^{n_0+n_1}$

Définition 2.4

Le gain d'un chemin causal peut être calculé par :

$$\mathbf{G} = \prod_i r_i^h \prod_j m_j^k$$

avec :

r_i = module de la $i^{ième}$ jonction-GY
m_j= module de la $j^{ième}$ jonction-TF
n= nombre des jonctions GY présentes dans le chemin considéré.
p= nombre des jonctions TF présentes dans le chemin considéré.

h = 0 si n=0

h = 1 si la jonction-GY a la causalité suivante ⊣—— GY ⊢——

h = -1 si la jonction-GY a la causalité suivante ⊢—— GY ——⊣

k = 0 si p=0

k = 1 si la jonction-TF a la causalité suivante ⊢—— TF ⊢——

k = -1 si la jonction-TF a la causalité suivante ⊣—— TF ——⊣

II. 2 Boucles causales

Définition 2.5

Une boucle causale entre deux éléments X-Y de type {I, C, R} est formée à partir des combinaisons des parcours, pris 2 à 2, des différents chemins causaux directs liant X à Y.

Il en résulte qu'une boucle causale possède un parcours "aller" et un parcours "retour". Ils ne représentent pas forcément le même chemin causal direct. Ainsi, si **n** est le nombre de chemins causaux directs, liant deux éléments quelconques (distincts), **n** x **n** est celui des boucles causales possibles.

Ainsi dans un modèle bond-graph deux types de causales sont possibles :

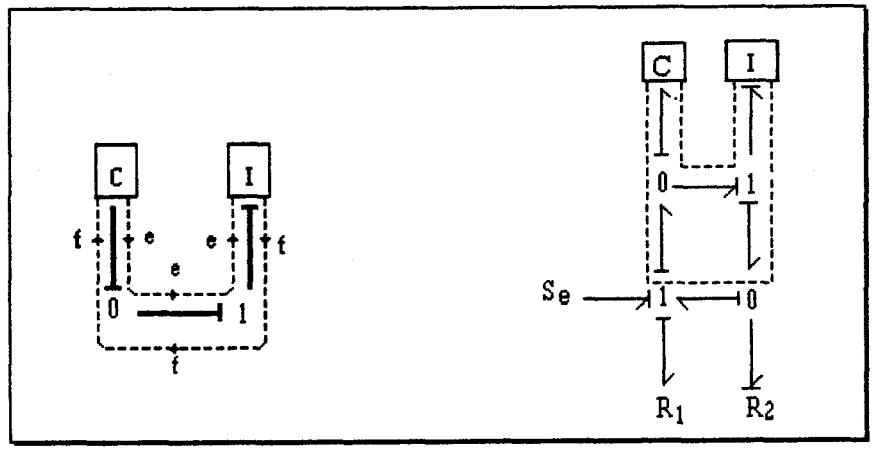


figure 2.31

1) une boucle causale uniforme est formée par les parcours d'un même chemin direct (cas de la figure 2.31.a)

2) une boucle causale ouverte est formée par les parcours de deux chemins directs différents (cas de la figure 2.31.b)

Un type particulier de boucles causales, signalées par F.T. Brown [1971] peut apparaître dans un modèle bond-graph (cf. figure 2.32). Il est dû à l'existence d'un chemin entre un élément et lui même. Ce chemin possède nécessairement un nombre impair de jonctions-GY.

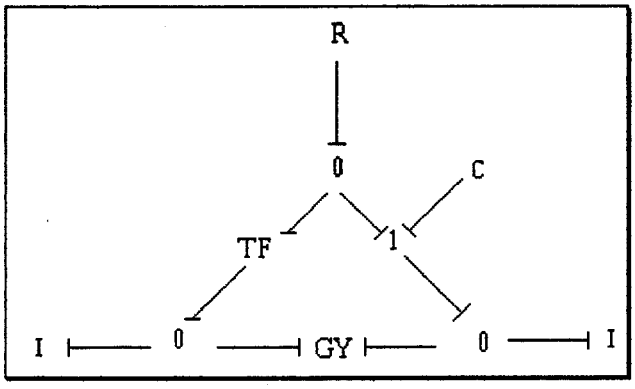


figure 2.32

Remarque 2.13

Ce chemin peut être engendré par la résolution d'une éventuelle boucle de causalité lors de l'affectation de la causalité. Mais il peut être également évité par application de la méthode proposée précédemment. En effet lorsqu'une boucle de causalité ayant un nombre impair de jonctions-GY, n'est pas algébrique (elle possède un nombre impair de conflits de type 'd'), l'utilisateur et/ou Archer doivent choisir une causalité dérivée parmi les listes non vides de ces conflits ou parmi les éléments résistifs associés aux listes. L'élément choisi va former alors un chemin direct entre lui même et produira donc deux boucles causales de même genre, associées à chacun de ses parcours.

Définition 2.6

Le signe d'une boucle causale est égal au produit des signes de son **parcours aller** et de son **parcours retour**.

Définition 2.7

Le gain d'une boucle causale, entre deux éléments X et Y, est égal au produit des gains de son **parcours aller** et de son **parcours retour**, multiplié par les modules de X et deY.

III. Organisation des données

Dans cette section nous allons présenter une méthode qui va nous permettre de déterminer automatiquement tous les chemins causaux et leurs informations (identificateurs des jonctions GY et MGY, identificateurs des jonctions TF et MTF, nombre de demi-flèches inversées autour des jonction-0 et celui autour des jonction-1), ainsi que toutes les boucles causales, leur signe et leur gain.

Comme il a été signalé plus haut, pour faciliter son exploitation, nous allons considérer un modèle bond-graph causal comme un graphe orienté classique. Dans ce cas les éléments de type {I, C, R}, contrairement à la présentation précédente (voir première partie), ne seront pas ignorés. Ainsi on définit ce graphe de la manière suivante :

- les noeuds sont les éléments bond-graph de type {S_e, S_f, I, C, R, 0, 1, TF, GY, MTF, MGY}
- les arcs sont orientés dans le sens de la variable "effort"
- chaque arc "**u**" véhicule deux valeurs : le numéro du lien qui lui est associé et un "drapeau" noté $\gamma(\mathbf{u})$, qui informe sur l'orientation de la puissance .
- deux sous-ensembles se démarquent :

* le premier caractérise les noeuds de départ qui vont correspondre aux éléments de type {S_e, I, C, R, GY, MGY} possédant une causalité effort sortant.

* le second caractérise les noeuds d'arrivée qui vont correspondre aux éléments de type {S_f, I, C, R, GY, MGY} possédant une causalité effort entrant.

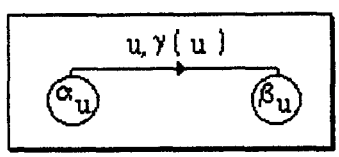


figure 2.34

Dans cette perspective pour ce type de graphe, nous avons cherché une représentation, facile à manipuler. On trouve dans la littérature de la théorie des graphes (M.Minoux. M.Gourdon. [1979], F. Buckley et F. Harary[1990], ...) des outils pour représenter l'existence des connexions dans un graphe. Leur efficacité algorithmique dépend de la nature du problème traité et de la particularité du graphe étudié.

Généralement on a deux familles de représentations : la première utilise la matrice

d'adjacence ou ses dérivées, la seconde utilise la matrice d'incidence ou ses dérivées.

Fréquemment les matrices d'incidence ou d'adjacence ont une faible densité (plusieurs composantes nulles), ce qui coûte cher au niveau de la place mémoire occupée et du temps de traitement de l'information cncernée. Minoux et Gourdon proposent alors d'utiliser les dérivées de ces matrices en ne représentant que leurs termes non nuls. Ces dérivées auront la forme d'un vecteur caractérisant l'existence d'un arc quelconque.

Cette dernière forme correspond bien à la nature de notre problème, nous allons l'adapter de la manière suivante :

Nous définissons :

-> deux vecteurs α et β, construits à partir du graphe orienté associé à un modèle bond-graph causal, tels que pour chaque arc u : $\alpha(u)$ (respectivement $\beta(u)$) va correspondre à son noeud de départ (respectivement à son noeud d'arrivée). L'incidence, comme le montre la figure 2.35 , est implicite, nous allons l'enrichir par une information supplémentaire caractérisant le sens de la demi-flèche (orientation de la puissance).

-> un vecteur γ tel que $\gamma(u)$ est un entier égal à 0 ou à 1.

$$\gamma(u) = 0 \quad \text{si} \quad \alpha(u) \rightarrowtail \beta(u)$$
$$\gamma(u) = 1 \quad \text{si} \quad \alpha(u) \leftarrowtail \beta(u)$$

figure 2.39

-> A partir de α et β, deux sous-ensembles, nommés α' et β', sont déterminés à partir des relations suivantes :

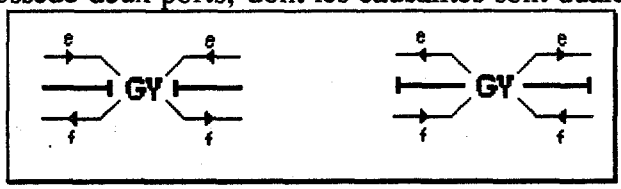
$$\alpha' = \alpha - (\alpha \cap \beta)$$
$$\beta' = \beta - (\alpha \cap \beta)$$

α' est l'ensemble des noeuds de départ, β' est celui des noeuds d'arrivée.

Terminons cette section de définitions par des remarques essentielles pour la compréhension des algorithmes qui vont suivre.

Remarques 2.14

1)Une jonction-GY possède deux ports, dont les causalités sont duales:



Pendant la détermination des chemins, nous considérons séparément chacun des ports comme un élément 1-port. Nous les distinguons par un paramètre, "p" pour premier et "s" pour second, dans l'ordre où ils sont rencontrés.

2) Pour explorer le graphe défini précédemment, nous partons des éléments de α' pour atteindre ceux de β'. Chaque départ à partir d'un élément X de α', va correspondre à l'exploration d'un arbre de racine X des noeuds de type {0, 1, TF, MTF} et des feuilles formées par les éléments de β' directement causalement liées à X.
Cette exploration arbre/arbre nous donnera tous les chemins causaux élémentaires ainsi que les informations qu'ils contiennent à savoir :
n_0, n_1, les TF et les GY présents.

3) Un chemin causal élémentaire "cce" peut lier deux jonctions-GY, ou une jonction-GY et un élément de type {Se, Sf, I, C, R}. Donc il ne peut jamais avoir un parcours de type "mixte".
Nous déduisons qu'un chemin causal direct possède au moins un "cce" : c'est un "cce" sans jonction-GY, ou une combinaison de "cce" possédant au moins une jonction-GY à ses extrémités.

4) L'information sur la présence d'une jonction-GY dans un "cce" sera indiquée seulement pour le premier port : $(GY_i)_p$. On évite ainsi d'allouer deux places mémoire pour la même jonction-GY.

5) Le traitement des jonctions-GY se généralise facilement pour les éléments **I, C, R** multiports.

IV. Algorithmes et approche I.A.

Toute application utilisant, pour son implantation sur calculateur, des méthodes et des techniques I.A. doit faire l'objet d'une étude préalable pour voir comment :

- utiliser la représentation symbolique du langage de programmation choisi ainsi que ses structures de données

- résoudre les éventuelles situations non déterministes, ainsi que des problèmes de non décidabilité.

- réduire la complexité des algorithmes (allocation de la mémoire et temps d'exécution).

C. Towsend [1986], H. Farreny et M. Ghallab [1987], ainsi que plusieurs articles dans Hand Book of I.A [1981, 1982, 1989] ont axé la résolution des problèmes précédents sur l'étude des trois domaines suivants : la représentation, la stratégie et l'organisation. Nous allons y revenir plus en détails dans le chapitre 5.

IV.1. Algorithmes pour la détermination des chemins causaux élémentaires

-> Représentation des données

* sous-graphe (noeud $\in$ α/ α(u), [liste de (numéro du lien : u, β(u), γ(u))]

Il représente un noeud et la liste de ses descendants. Sa construction se fait à partir de α, β, et γ, et se détermine facilement si le modèle bond-graph est représenté symboliquement.

*cce (numéro du cce, extrémité$_i$, extrémité$_j$, [liste de (u : numéro du lien, noeud p, noeud q, γ(u))])

avec : extrémité$_i$ $\in$ α', extrémité$_j$ $\in$ β', noeud p$\in$ α, noeud q $\in$ β.

* demi_flèche_inversée (le numéro du "cce", type de la jonction 0 ou 1)

N = nombre de demi-flèches inversées dans les jonctions 0 ou 1.

si N est pair **alors** le "cce" est positif **sinon** il est négatif **fsi**

Par souci d'une optimisation de l'allocation de la mémoire, nous prendrons, par défaut, N pair et ne sauvegarderons cette information que si N est impair.

* gain_tf (numéro du "cce", [liste de (numéro de la jonction-TF ou MTF, exposant_tf)])

avec par exemple :

exposant_tf = 1 si la jonction-TF a la causalité suivante : ⊢— TF —⊢

exposant_tf = -1 si ,, ,, ,, ,, : —⊣ TF ⊢—

Ce paramètre correspond à l'exposant du module de la jonction-TF_k ou MTF_k, ce dernier sera représenté par " $(m_k)^{exposant\_tf}$ ".

* gain_gy (numéro du "cce",
[liste de (numéro de la jonction-GY ou MGY, exposant_gy)])

avec par exemple :

exposant_gy = 1 si la jonction-TF a la causalité suivante : ⊢— GY —⊣

exposant_gy = -1 si ,, ,, ,, ,, : —⊣ GY ⊢—

Ce paramètre correspond à l'exposant du module de la jonction-GY_k ou MGY_k, ce dernier sera représenté par " $(r_k)^{exposant\_gy}$ ".

-> Algorithme 2.5

```
{contruction et exploration du graphe}
Début
{construction d'un graphe orienté à partir d'un modèle bond-graph causal}
   Liste_fils := [ ] {initialisation à vide de la liste des descendants}

   pour tout élément α(u) de α faire
      Liste_fils := liste_fils + (u, β(u), γ(u))
      tant que ( ∃ v / α(u) = α(v) et u différent de v) faire
         Liste_fils := liste_fils + (v, β(v), γ(v))
         Rétracter de la base de données active le fait : α(v) .
      fait
      Insérer dans la base de données active le fait :
      "sous-graphe(α(u), Liste_fils)"
      Rétracter de la base de données active le fait : α(u).
   fait
   {exploration du graphe}
   pour tout élément α'(u) de α' faire
      Activer le fait de la base de données sous-graphe(α'(u), L)
      Traiter_Arbre (α'(u), L)
   fait
Fin
```

Nous avons choisi d'écrire l'algorithme suivant dans une forme qui fait appel à une écriture à la fois en méta-langage et en programmation logique comme celle de Prolog.

Algorithme 2.5.1

Traiter_Arbre(_, [ ]) :- backtracking pour remonter le graphe et arrêter le dernier processus récursif.

Traiter_Arbre(α'(u), [(u, β(u), γ(u)), L']) :-

Empiler (u, α'(u), β(u), γ(u))

Si β(u)=β'(u) **Alors**

. L'état de la pile correspond à un "cce" entre α'(u) et β'(u)

. Dépiler le sommet

Sinon Si ($\exists$ v/ α(v) = β(u)) **Alors**

. activer le fait de la base de données sous-graphe(α(v), K)

. **Traiter_Arbre** (α(v), K) {appel récurssif}

FSi

FSi

Traiter_Arbre (α'(u), L') {appel récurssif}

Le prédicat "Traiter_Arbre(.)" utilise une organisation basée sur une pile de type LIFO ou DEPS avec la stratégie d'une exploration en "profondeur d'abord" muni d'un processus récursif. Ce dernier s'arrête quand il ne reste plus d'éléments à traiter ou quand il rencontre des heuristiques. Il entame alors un processus de retour arrière (backtracking) pour remonter l'arbre et explorer d'autres branches.

Parmi les heuristiques nous avons :
- Ne pas traîter un descendant s'il est en même temps ascendant. Ce qui se traduit par : "ne pas traîter un noeud s'il est déjà dans la pile".

- Ne pas traîter les branches qui ménent à des feuilles de type sources {S_e, S_f} quand la racine est du même type. C'est-à-dire interdire l'existence d'un chemin causal entre deux sources.

Les informations associées à un "cce" sont déterminées, pendant la réalisation de celui-ci à partir de l'état de la pile, en appliquant à deux éléments consécutifs de la liste du "cce" les règles suivantes :

soient [... ,(u, _, β(u), γ(u)), (v, β(v), _, γ(v)), ...] deux éléments consécutifs quelconques ; on a :

-Si $\beta(u)=\beta(v)$=jonction-0 ou 1 **et** $\gamma(u) + \gamma(v)$ est impair **alors**
Incrémenter le nombre des inversions des demi-flèches associé au type β(u) et β(v).

Si à la fin du traitement de la liste ce nombre est impair **alors**
Sauvegarder l'information :
demi_flèche_inversée (le numéro du "cce", type de la jonction 0 ou 1)

- **Si** $\beta(u)=\beta(v)$=jonction-TF **alors**
  Si $\gamma(u) + \gamma(v) = 2$ **alors** exposant_tf = -1
  Si $\gamma(u) + \gamma(v) = 0$ **alors** exposant_tf = 1

- **Si** $\beta(u)=\beta(v)$=jonction-GY **alors**
  Si $\beta(u) \in \alpha'$ **alors** exposant_gy = 1
  Si $\beta(v) \in \beta'$ **alors** exposant_gy = -1

IV.2. Algorithmes pour la détermination des chemins causaux direct et indirect

Dans ce qui va suivre nous noterons, "**ccd**" le chemin causal direct et "**cci**" le chemin causal indirect.

Représentation des données

* ccd(n° du ccd,
  extrémité$_i$, extrémité$_j$, [liste des numéros des "cce" formant ce "ccd"])

* cci(extrémité$_i$, extrémité$_j$, [liste des numéros des "ccd" formant ce "cci"],
  [liste des éléments intermédiaires])

avec extrémité$_{i \text{ ou } j} \in \alpha$ ou β.

éléments intermédiaires = éléments de type {I, C, R} (différents de extrémité$_i$ et extrémité$_j$) intervenant dans les "ccd" associés au "cci" considéré.

Algorithme 2.6

{détermination des chemins directs}
étape 1 : Convertir tous les "cce"(N, $\alpha'(u) \neq GY$, $\beta'(u) \neq GY$, _)
en un "ccd"(n°, $\alpha'(u)$, $\beta'(u)$, [N])
{n° est une variable qu'il faut initialiser à 0 et qui sera incrémentée
chaque fois que l'on crée un "ccd"}
L := [ ] {initialisation à vide d'une liste d'entiers }

étape 2 : **pour** tout "cce"(N_1, $\alpha'(u) \neq GY$, $(GY_i)_1$, _) **faire**
L := L $\cup$ {N_1}
Appel du "cce"(N2, X, $(GY_i)_2$, _) {deuxième partie de GY_i}
L := L $\cup$ {N_2}
si X $\neq$ GY **alors** Insérer dans la base de données le prédicat
"ccd"(n°, $\alpha'(u)$, X, L)
L:=[ ] {réinitialisation de L pour un nouveau chemin}
sinon aller à l'étape 3
fait
étape 3 : {similaire à l'étape 2}
pour tout "cce"(N_1, $(GY_i)_1$, $\beta'(u) \neq GY$, _) **faire**
L := L $\cup$ {N_1}
Appel du "cce"(N2, $(GY_i)_2$, X, _) {deuxième partie de GY_i}
L := L $\cup$ {N_2}
Si X $\neq$ GY **alors** Insérer dans la base de données le prédicat
"ccd"(n°, X, $\beta'(u)$, L)
L:=[ ] {réinitialisation de L pour un nouveau chemin}
sinon aller à l'étape 2
Fait

L'algorithme de la détermination des chemins indirects peut être écrit de la manière suivante :

étape 1 :
Construire un graphe non orienté dont les noeuds sont les éléments bond-graph de type {I, C, R, S_e, S_f}, correspondant aux extrémités des chemins causaux directs ; et dont les arêtes indiquent l'existence d'un "ccd" entre deux noeuds. Chaque arête sera muni d'un "poids" correspondant au "ccd" qui lui est associé.

étape 2 :
Explorer ce graphe et former tous les chemins indirects entre deux noeuds à partir de chaque chaîne d'arêtes, de longueur supérieure ou égale à 2, liant ces noeuds.

L'exploration d'un graphe non-orienté est généralement délicate à cause de sa structure non-linéaire. Une autre contrainte, liée à la nature de notre application (détermination de tous les chemins indirects possibles), va accroître la compléxité de cet algorithme.

Malgré l'inexistence de situations non-déterministes (elles n'auront lieu que si on tient compte du poids associé à chaque arête), cette complexité peut être exponentielle et peut engendrer un problème combinatoire dès que le modèle bond-graph atteint une taille moyenne (une vingtaine d'éléments de type {I, C, R, S_e, S_f}).

Il devient alors clair que la résolution de ce problème va nécessiter l'intervention d'heuristiques qui interdiront l'exploration de certaines branches particulières. Par exemple :

- ne pas traîter un noeud existant déjà dans la pile.
- ne pas tenir compte des chemins entre deux sources.
- une source ne doit jamais être prise comme noeud intermédiaire. Elle ne peut être qu'un noeud de départ ou d'arrivée.
- un chemin causal ne doit pas passer plus d'une fois par le même lien en empruntant la même orientation causale.

L'ajout de ces heuristiques va réduire certes la complexité, mais celle-ci restera néanmoins assez importante.

Les méthodes de la détermination des équations associées aux modèles mathématiques ainsi qu'à celles de l'analyse structurelle nous ont inspirés pour trouver une solution à ce problème. En effet, nous n'aurons jamais besoin de tous les chemins directs simultanément (contrairement aux autres chemins causaux), cette information devient importante lorsqu'une variable du vecteur d'entrée du système n'est pas directement causalement liée à une variable du vecteur de sortie. Ce critère va réduire largement la compléxité de l'algorithme et facilitera son implantation sur calculateur.

Remarque 2.15

Cet algorithme aura le même principe que celui de la détermination des "cce", sauf que dans ce cas la phase "construction d'un nouveau graphe" peut être évitée en gérant la

liaison entre les différentes extrémités des chemins causaux par l'utilisation d'une pile de parcours. Ainsi l'algorithme peut s'écrire dans le même style que l'algorithme 2.4 en remplaçant le fait "lien(.)" par le fait "ccd(.)".

Algorithme 2.7

Déterminer_CCI (Entrée, Sortie)
{Les éléments de la pile de parcours ont la forme suivante :
(noeud d'arrivée d'un "ccd", numéro du "ccd")}

Début
PileParcours := [(Entrée, 0)]
{initialisation de la pile à la variable Entrée, 0 = chemin fictif}

tant que ∃ un ccd(N_i, Entrée, X,_) ou un ccd(N_j, X, Entrée,_) **faire**
PileParcours:=PileParcours ∪ [(X, N_i ou N_j)] {empiler(X, N_i ou N_j)}
Traiter_Fils(X)
Passer à un autre chemin - s'il existe- d'extrémité "Entrée", non utilisé précédemment.
fait
Fin

Algorithme 2.7.1

Traiter_Fils(X)
Début
tant que ∃ un ccd(N_k, X, Y, _) **ou** un ccd(N_h, X, Y,_) **et** Y, N_k, N_h n'appartiennent pas à PileParcours **faire**
si Y = Sortie **alors**
{l'état de la pile correspond à un chemin indirect entre l'élément du fond et son sommet.}
Copier l'état de la pile et former les paramètres d'un "cci" à partir de cette copie .
sinon
PileParcours:=PileParcours ∪ [(Y, N_k ou N_h)] {empiler(Y, N_k ou N_h)}
traiter_fils(Y) {appel récursif}
fSi
Passer à un autre chemin -s'il existe- d'extrémité X n'appartenant pas à la pile.
fait

{à la sortie de la boucle, X sera terminal et ne correspondra jamais à la variable Sortie}

Dépiler le sommet de la pile de parcours

Fin

Le signe d'un ccd(_,E_i, E_j,_) se calcule, en fonction du sens de parcours (de E_i vers E_j ou de E_i vers E_j), en additionnant le nombre des faits : "demi_flèche_Inversée(.)" appropriée au parcours de chacun de ses "cce". **Si** ce nombre est pair **alors** le chemin est positif, **sinon** il est négatif.

Quant à son gain, il est égal à $\prod_i m^{exposant\_tf(i)}\, r^{exposant\_gy(i)}$,

avec i = numéro d'un de ses "cce".

Le signe d'un "cci" est égal au produit des signes de ses "ccd", son gain est égal au produit des gains de ses "ccd", multiplié par les impédances de ses éléments intermédiaires.

IV.3 Détermination des boucles causales

La méthode que nous allons exposer ici ne fait que suivre les définitions 2.1 et 2.2 pour déterminer les boucles ainsi que leur signe et leur gain. D'autres méthodes (récentes) permettront d'arriver plus rapidement au même résultat en utilisant de la matrice de structure associée à un modèle bond-graph ou en appliquant de l'algèbre de Serius aux chemins causaux directs. Nous y reviendrons dans l'annexe A.

Représentation des données

* Boucle_Causale (n° de la boucle, extrémité$_i$, extrémité$_j$, N_1, N_2)
avec :
extrémité de type {I, C, R}
N_1 numéro du "ccd" associé au Parcours "Aller"
N_2 ,, ,, ,, ,, "Retour"

* Gain_Boucle (n° de la boucle, son signe, son gain)

Algorithme 2.8

{construction des boucles}

pour tout "ccd (N, X, Y, _)" **faire**

L := {N} {L : Liste des "ccd" composant la boucle}

Chercher tous les chemins "**ccd**"(N', X, Y avec N ^{1} N')

L:= L $\cup$ {N'}

avec les couples (PA, PR) de L $_{X}$ L **faire**

former les faits Boucle_Causale (n°, X, Y, A, B)

fait

{signes et gains associés aux boucles}

pour toute Boucle_Causale (n°, X, Y, PA, PR) **faire**

si X $\in$ $\alpha'(.)$ **alors** commencer PA par un parcours "effort"

sinon commencer PA par un parcours "flux"

Alterner le sens des parcours effort ou flux en accord avec le sens de départ des "cce" formant les "ccd" PA et PR. Et appliquer à chacun de leurs "cce" l'algorithme suivant :

NombreInversion :=0 {initialisation du compteur des inversions des demi-flèches}

gain := 1 {initialisation du gain de la boucle}

pour tout "cce" **n** appartenant à PA et PR **faire**

si le parcours est de type "effort" **alors**

si existence du fait demi_flèche_inversée(n, 1) **alors**

NombreInversion := NombreInversion + 1

fsi

sinon {parcours de type "flux"}

si existence du fait demi_flèche_inversée(n, 0) **alors**

NombreInversion := NombreInversion + 1

fsi

fsi

si existence du fait tf_exposant(n, TF_i, k) **alors** gain := gain x $(m_i)^{2k}$ **fsi**

si existence du fait tf_exposant(n, GY_j, h) **alors** gain := gain x $(r_j)^{2h}$ **fsi**

fait

si NombreInversion est pair **alors** signe positif **sinon** signe négatif **fsi**

gain := Gain x les impédences des éléments X et Y (extrémités de la boucle).

Insérer dans la base de données le fait : Gain_Boucle(n°, signe, gain)

V. Exemple d'application et implantation sur ARCHER

Au cours de cette section, nous allons appliquer les différents algorithmes précédents au modèle de la figure 2.36. Nous montrerons également toutes les étapes suivies par Archer, ainsi que les données qu'il génère.

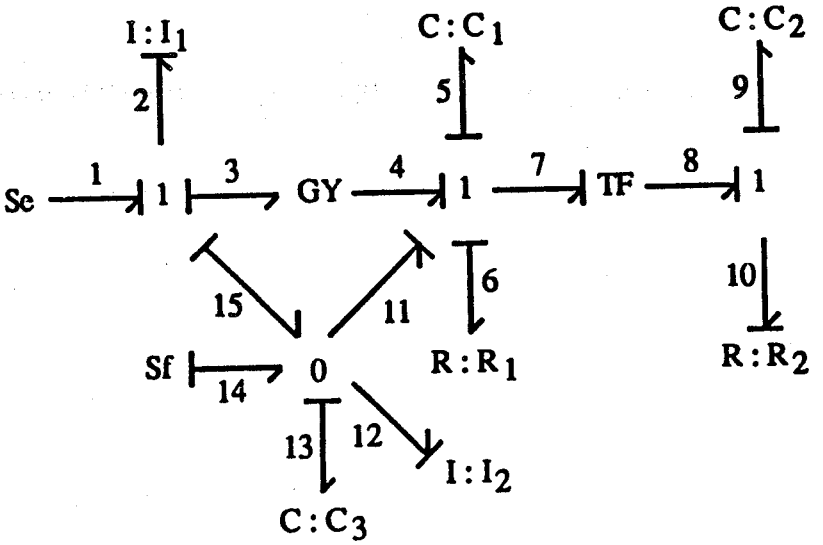


figure 2.40 : exemple de modèle bond-graph

Comme nous l'avons signalé au chapitre 1, la première étape d'Archer consiste à saisir la description d'un modèle physique à partir d'un langage naturel ou d'une représentation graphique. Ce modèle sera compilé pour engendrer un modèle bond-graph représenté formellement par les données de la première colonne du tableau suivant :

Faits représentant le modèle bond-graph simplifié	**Faits caratérisant la causalité et le numéro de chaque lien**
Structure :	NumEffort(1,Se_1,1_1)
	NumEffort(2,1_1,I_1)
Lien(1_1,GY_1) ; lien(GY_1,1_2)	NumEffort(3,GY_1,1_1)
Lien(1_2,TF_1) ; Lien(TF_1,1_3)	NumEffort(4,GY_1,1_2)
Lien(0_1,1_2) ; Lien(1_1,0_1)	NumEffort(5, C_1, 1_2)
	NumEffort(6,Se_1,1_1)
éléments :	NumEffort(7,R_1,TF_1)
	NumEffort(8,TF_1,1_3)
Jonction(1_1,[Se_1,I_1],0)	NumEffort(9,C_2,1_3)
Jonction(GY_1,[ ],0)	NumEffort(10,1_3,R_2)
Jonction(1_2,[Se_1,I_1],0)	NumEffort(11,0_1, 1_2)
Jonction(1_1,[C_1,R_1],0)	NumEffort(12,0_1,I_2)
Jonction(TF_1,[ ],0)	NumEffort(13,C_3,0_1)
Jonction(1_3,[C_2,R_2],0)	NumEffort(14,0_1,1_1,)
Jonction(0_1,[S_{f1}, I_2, C_3],0)	NumEffort(15,0_1, 1_1)

tableau 2.3 : représentation formelle du modèle bond-graph causal de la figure 2.

L'affectation de la causalité correspond à la deuxième étape et génère l'information de la colonne 2 du tableau précédent. A ce niveau Archer peut éditer graphiquement, sur un périphérique de sortie, le bond-graph associé aux données du tableau 2 : 3 et dessiner le modèle de la figure 2. (R. Bouayad [1991]).

Toujours, à partir des données du tableau 2.4, Archer détermine α, β et γ par l'interprétation de la figure suivante :

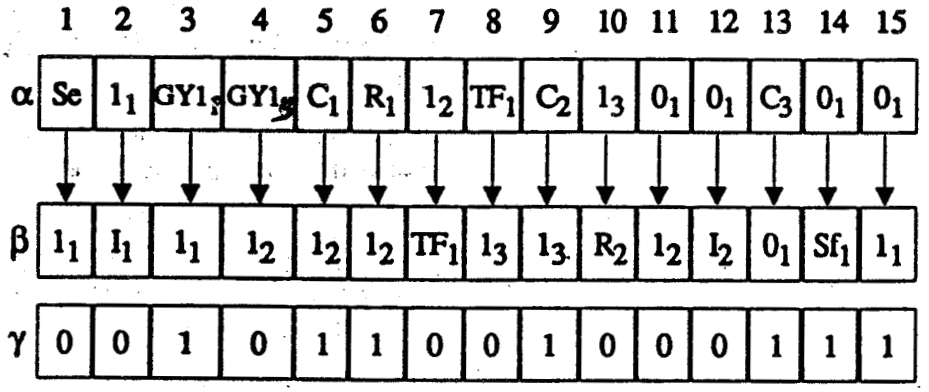


figure 2.37 : Valeurs des composantes des vecteurs α, β et γ

De ces derniers il extrait les sous-ensembles α' et β' :

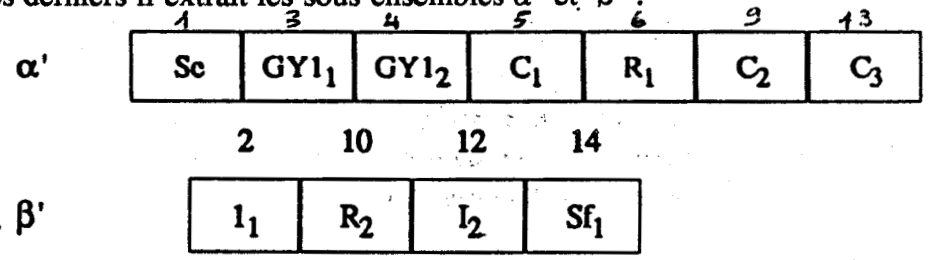


figure 2.38 : sous-ensembles α' et β'

A partir de ces vecteurs et de ces sous-ensembles, Il construit le graphe orienté suivant:

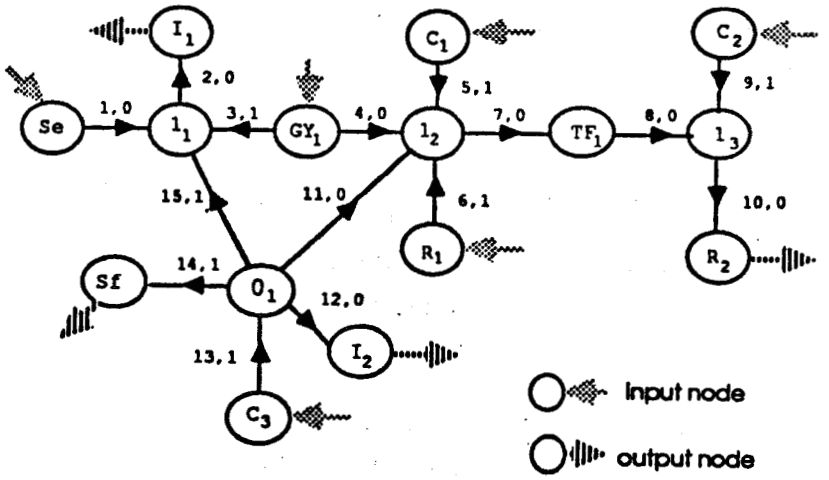


figure 2. 39 : graphe orienté associé au modèle de la figure 2.

Ce graphe se représente, noeud par noeud, par la fait "Sous-graph(α(u), [(u, β(u), γ(u))...]". Celui de la figure 2.39 est représenté, entre autres , par :

Sous-graphe (Se, [(1, I_1, 0)]) ; Sous-graphe (I_1, [(2, I_1, 0)]) ;
Sous-graphe ((GY_1)$_p$, [(3, I_1, 1)]) ; Sous-graphe ((GY_1)$_s$, [(4, I_2, 0)]) ;
Sous-graphe(I_1, [ ]) ...

Archer explore alors le graphe arbre par arbre. Il en déduit les "cce". Ainsi pour les données précédentes nous avons :

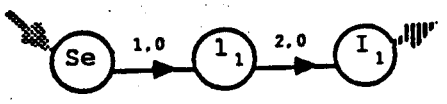


figure 2. 40 : sous-arbre de racine S_e et de feuille I_1.

Il commence par le "Sous-graphe (Se, [(1, 1_1, 0)])" qui possède un fils "Sous-graphe (1_1, [(2, I_1, 0)])" qui a également un fils "Sous-graphe(I_1, [ ])". Ce dernier, à cause de sa liste vide, est un noeud d'arrivée, Archer déduit alors le fait suivant :

"cce (1, S_e, I_1, [(1, Se, 1_1, 0), (2, 1_1, I_1, 0)])" sans information particulière pas de demi-flèches d'orientation invrerses ni de "exposant_tf(.)" ni de "exposant_gy(.)".

La représentation des "cce" peut paraître redondante. Malheureusement, on ne peut échapper à cet état de fait. Car, en fonction des traitements, on aura besoin uniquement des trois premiers parmètres (par exemple la détermination des boucles), ou de toute la trace du chemin, c'est-à-dire de son quatrième paramètre car le sens du parcours est primordial (par exemple deux chemins sont non disjoints s'ils ont au moins un lien parcouru dans le même sens).

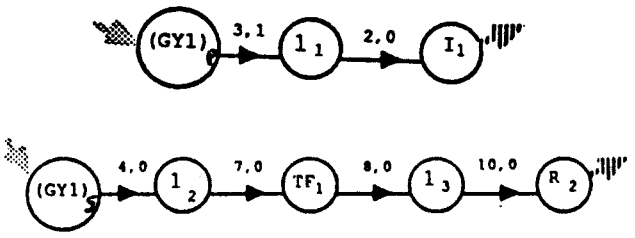


figure 2.41 : exemple de sous-arbres de racine GY_i

De la même manière Archer déduit les faits suivant des sous-arbres de la figure 2.41 :

"cce(2, $(GY_1)_p$, I_1, [(3,GY_1, 1_1,1), (2, 1_1, I_1, 0)])"
"demi_flèche_inversée (2, 1)"et "exposant_gy(2, 1, 1)"

"cce(3, $(GY_1)_s$, R_2, [(4, GY_1, 1_2, 0), (7, 1_2, TF_1, 1_3, 0), (8, TF_1, 1_3, 0), (10, 1_3, R_2, 0])" et "exposant_tf(3, 1, 1)"

Archer entame alors l'étape de la détermination des "ccd" à partir de "cce", et donnera par exemple, à partir d'une combinaison du "cce(2, $(GY_1)_p$, I_1, _)" et du "cce(3, $(GY_1)_s$, R_2, _)", le fait : "ccd(1, I_1, R_2, [2, 3])". Tous les autres "cce" se convertissent directement en un "ccd", nous avons par exemple :

"cce (1, S_e, I_1, _)" => "ccd(2, S_e, I_1, [1])"

"cce (4, C_1, R_2, _)" => "ccd(3, C_1, R_2, [4])"

...

A partir des "cce" dont les extrémités sont de type {I, C, R}, Archer va déterminer toutes les boucles causales. A noter que dans ce modèle bond-graph nous n'avons pas plus d'un "ccd" entre deux éléments quelconques. Donc les parcours "aller" et "retour" d'une boucle vont correspondre à ceux du "ccd" qui lui est associé. Nous avons par exemple :

"ccd(1, I_1, R_2, [2, 3])" => Boucle_Causale(1, I_1, R_2, 1, 1)

"ccd(3, C_1, R_2, [4])" => Boucle_Causale(1, C_1, R_2,1, 1)

...

Pour déterminer le signe et le gain d'une boucle, nous prenons pour exemple la boucle causale numéro 1. Notons son signe S et son gain G.

Archer commence par chercher les "cce" associés au "ccd" numéro 1 (parcours aller), soit [2, 3]. Il concatène cette liste avec l'inverse de celle associée au "ccd" numéro 1 (parcours retour), on a alors la liste suivante [2, 3, 3, 2]. Comme $I_1 \in \beta'$, on commence l'exploration de la liste par un parcours "flux" (avec n_0 , n_1 initialisés à 0) de la manière suivante :

"cce" n°2 avec un parcours "flux" => n_0 inchangé car pas de demi_flèche_inversée(n°2, 0)
G=G x $(r_1)^1$ car exposant_gy(2, 1, 1) existe

"cce" n°3 avec un parcours "effort" => n_1 inchangé car pas de demi_flèche_inversée(n°3, 1)
G=G x $(m_1)^1$ car exposant_tf(3, 1, 1) existe

"cce" n°3 avec un parcours "flux" => n_0 inchangé car pas de demi_flèche_inversée(n°3, 0)
G=G x $(m_1)^1$ car exposant_gy(3, 1, 1) existe

"cce" n°2 avec un parcours "effort" => $n_1 := n_1+1$ car demi_flèche_inversée(n°2, 1) existe
G=G x $(r_1)^1$ car exposant_gy(2, 1, 1) existe

Finalement :
S est négatif car $(-1)^{n0} + n1 = -1$
$G = (r_1)^2 \times (m_1)^2 \times (1/I_1p) \times (1/R_2)$

Remarque 2.16

Dorénavant lorsque nous parlerons de gain de chemin ou de boucle, nous sous entendons signe et gain.

La détermination des chemins indirects n'est pas systématique, Archer fait appel à ce module lorsqu'il ne trouve pas d'information sur une liaison causale directe entre deux éléments bond-graph. Connaissant ces derniers, il va former tous les "cci" qui les relient.

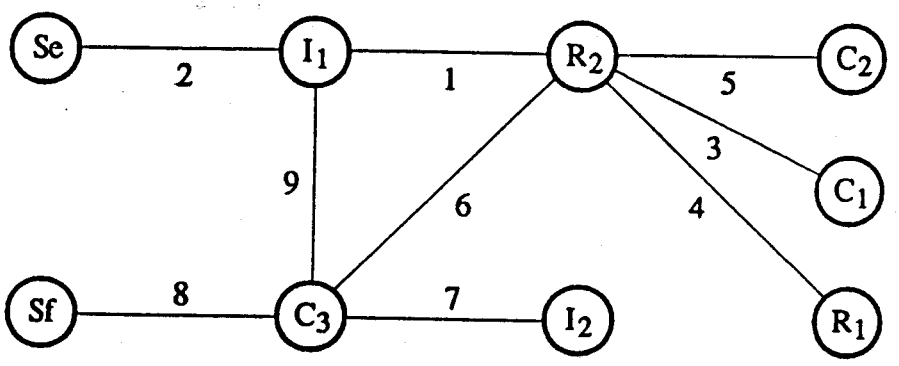


figure 2 .42 : graphe non orienté formé à partir des "ccd" du modèle de la figure 2.

La figure 2.42 présente le graphe non-orienté, associé à notre exemple, qui permet de déterminer tous les chemins causaux indirects. Cela donnera une idée sur le nombre des chemins possibles ainsi que sur la compléxité algorithmique de son exploration.

A titre d'application, nous déterminons les "cci" entre S_e et I_2 :

"cci(S_e , I_2 , [2, 9, 7], [I_1 , C_3])" et "cci(S_e , I_2 , [2, 1, 6, 7], [I_1 , R_2 ,C_3])"

VI.Conclusion

Tout au long de cette partie nous avons :
-recensé les différentes informations causales,
-donné des méthodes algorithmiques pour leur détermination et indiqué comment

résoudre la difficulté relative à leur implantation sur calculateur.
-illustré sur un exemple l'application des différentes algorithmes, et montré comment Archer représente et génère ces informations.

Les graphes sont largement utilisée en informatique car c'est la structure qui représente le mieux le monde réel. Leur littérature propose un nombre important d'outils et de techniques. Nous en avons utilisées quelques unes afin d'explorer un bond-graph comme un graphe, et d'exploiter toutes les informations causales qu'il contient.

Contrairement aux problèmes classiques, recontrés dans la théorie des graphe comme "le commis voyageur", "plus court chemin", etc... où à chaque fois on garde la meilleure exploration du graphe, ici on doit souvent garder toutes les explorations possibles car aucune n'est à négliger dans les phase de modélisation et d'analyse, ce qui accroît la difficulté d'une implantation sur calculateur. Celle-ci doit être précédée d'une étude sérieuse de la complexité algorithmique et de la technique d'adaptation au langage de programmation utilisé. Les techniques I.A. dont nous avons rappelé les 3 axes principaux se prêtent bien à ce type de problèmes(*).

Conclusion générale

Dans ce chapitre nous avons présenté dans une première partie une étude sur l'affectation de la causalité à un modèle bond-graph. Nous avons exposé les problèmes qu'on risque de rencontrer lorqu'une causalité mixte (intégrale et dérivée) est obligatoire. Plusieurs raisons nous incitent à privilégier une affectation automatique par rapport à une affectation directe :

- ne pas restreindre l'utilisation des bond-graphs aux seuls bond-graphistes.
-les méthodes d'analyse et de réduction des modèles (C. Sueur [1990]), s'appuyent sur diverses étapes de la dualisation de la causalité des éléments. Elles nécessitent donc une méthode automatique souple et rapide.

Ceci nous montre que la phase de l'affectation de la causalité ne peut pas se faire sans permettre à l'utilisateur d'une part, d'imposer certaines causalités et de participer, d'autre part, à la résolution d'un conflit causal. Ainsi on peut avoir pour un modèle plusieurs états causaux. Ce qui permettra de proposer plusieurs solutions au niveau de la

* une partie du chapitre 4 est consacrée au différentes techniques I.A. que nous avons utilisé.

modélisation. Ces états pourront être comparés soit directement, soit à partir d'une méthode d'analyse structurelle. Toutes ces manipulations ne seront pas sans influence sur la modification et la correction du modèle dynamique étudié.

Dans cette optique, nous avons proposé une méthode de l'affectation automatique de la causalité qui prend en compte à la fois le souhait de l'utilisateur et la nature physique des éléments. Elle s'appuie, dans le cas où une partie du modèle traité ne possède pas d'information prèalable, sur une expertise que nous avons dégagée au cours de nos recherches à partir de la topologie d'un bond-graph.

Nous l'avons doté, dans le cadre d'Archer, d'un interface utilisateur convivial permettant dans le cas d'échange de dialogues avec l'utilisateur de faire appel, lorsque ce dernier n'arrive pas à se décider, à des règles déterminées à partir de problèmes rencontrés dans les modèles mathématiques ou dans l'analyse. On peut augmenter le nombre de celles-ci par l'intermédiaire d'un module d'Archer qui gère ce type de règles.

La seconde partie montre comment déterminer toutes les informations causales. Elles constituent la base de connaissances nécessaires à toutes les phases de modélisation (Matrice de transfert, Equation d'état ...) et d'analyse (Commandabilité et Observabilité structurelles, Stabilité asymptotique...). Elles sont notamment utiles pour la construction de la matrice de structure et du bond-graph réciproque, pour la réduction des modèles etc... (*)

* L'annexe A va présenter les algorithmes associés à la détermination de certaines de ces notions.

Chapitre 3

Proposition d'une méthode permettant de déterminer l'équation d'état sous forme d'expression formelle pour les systèmes linéaires et non-linéaires

I. Introduction

Dans ce chapitre, nous allons présenter une synthèse des méthodes permettant la détermination de l'équation d'état déduite du modèle bond-graph de systèmes linéaires et non linéaires.

Dans un premier temps, nous allons exposer et discuter les deux méthodes existantes. La première, connue sous le nom de ***méthode de Rosenberg*** produit une équation d'état sous forme numérique. La seconde, nommée ***méthode de chemins*** fournit une équation d'état paramétrique.

Puis nous proposerons, suite à nos travaux, une nouvelle méthode permettant d'obtenir une équation d'état sous forme d'expressions formelles. Cette méthode présente l'avantage d'être pratique, instructive et facilement programmable.

A la fin du chapitre, nous étendrons nos résultats à certaines classes de systèmes non linéaires.

II. Problématique

La modélisation des systèmes dynamiques dans divers domaines physiques est confrontée au défit suivant :

> développer des méthodes systématiques (numériques ou pratiques) permettant de transformer un modèle descriptif du système en l'une des deux formes suivantes :

$$\dot{X} = AX + BU \qquad (3.1)$$

$$\dot{X} = \Phi(X,U) \qquad (3.2)$$

avec Φ fonction non linéaire du vecteur d'état et du vecteur d'entrée.

Cette présentation, exprimée par R.C. Rosenberg [1971], introduit bien l'esprit du problème de la détermination de l'équation d'état d'un système. Pendant la réalisation de ARCHER, nous sommes confrontés à la résolution de ce problème à partir d'un modèle bond-graph. Ainsi pour

arriver aux équations (3.1) et (3.2), il faut tenir compte de facteurs comme :

- le type des variables de stockage d'énergie (indépendants et/ou dépendants),
- le type de certaines liaisons causales (existence de boucles algébriques entre des éléments dissipatifs)
- la nature du système (linéaire ou non linéaire).

Tous ces cas conduisent à des équations d'état soit du type (3.1), (3.2), soit à d'autres formes qui, moyennant quelques manipulations algébriques, se rapprochent assez souvent de l'équation recherchée.

Dans ce qui va suivre, nous donnerons l'essentiel de l'étude présentée par D. Karnopp et R.C Rosenberg [1983 partie 4, chapitre 14] et qui reflète les formes générales de l'équation d'état générée à partir d'un bond-graph en fonction de sa nature.

Un système ne contenant que des éléments dynamiques associés à des éléments bond-graphs en causalité intégrale, ne présente aucune difficulté quant à la détermination de son équation d'état. On obtient alors, dans le cas d'un système linéaire, une équation de type (3.1). La présence d'éventuels éléments non linéaires aboutira à une équation de type (3.2) qui se dérive facilement à partir de la démarche suivie pour déterminer l'équation (3.1).

Le problème se complique avec l'existence d'éléments en causalité dérivée (présence dans le système de variables d'énergie dépendantes. L'équation d'état prend alors la représentation suivante :

$$\begin{cases} \dot{X}^i = \Phi^i\left(X^i, \dot{X}^d, U\right) & \text{(3.3.a)} \\ X^d = \Phi^d\left(X^i, U\right) & \text{(3.3.b)} \end{cases}$$

avec X^i = vecteur des variables indépendantes
X^d = vecteur des variables dépendantes (*)

* Dorénavant l'exposant 'i' correspond à une causalité intégrale, quant à la causalité dérivée

Des transformations différentielles et algébriques des équations (3.3), vont nous conduire dans un premier temps à :

$$\dot{X}^d = \Phi^{dd}\left(X^i, \dot{X}^i, U, \dot{U}\right) \qquad (3.4)$$

avec Φ^{dd} une fonction dérivée à partir de Φ^d,et dans un deuxième temps à :

$$\dot{X}^i = \Phi^{ii}\left(X^i, \dot{X}^i, U, \dot{U}\right) \qquad (3.5)$$

avec Φ^{ii} une nouvelle fonction formée à partir de Φ^i et Φ^{dd}.

Notons l'apparition de $\dot{X}^i$ part et d'autre de l'équation (3.5). Et dans le cas d'un système non linéaire(*), le faire passer du côté gauche ne constitue pas toujours une tâche facile. Cette opération devient quasi-impossible dans le cadre d'un formalisme très général utilisant une forme non explicite (cf. fin de ce chapitre : utilisation de Φ pour les fonctions non-linéaires associées aux composantes du système). Cependant une solution numérique peut apporter une satisfaction à ce problème. Ainsi l'équation (3.5) va se définir de la manière suivante :

$$\dot{X}^i = \Phi\left(X, U, \dot{U}\right) \qquad (3.6)$$

L'équation (3.6) se ramène dans le cas d'un système linéaire à :

$$\dot{X} = AX + BU + E\dot{U} \qquad (3.7)$$

Signalons que, d'après D. Karnopp & R.C Rosenberg [1983], l'apparition de $\dot{U}$ dans l'équation d'état est un signe d'éventuelles erreurs de modélisation.

La phase de l'affectation de la causalité peut entraîner des boucles causales algébriques entre des éléments dissipatifs (Système possédant des éléments résistifs couplés). Et la formulation de l'équation d'état demandera, là aussi, certaines manipulations algébriques qui, dans le cas linéaire, se ramènent à des opérations matricielles (somme, produit et

---

il sera indiquée par l'exposant 'd'.

* Si le système est linéaire, faire passer $\dot{X}^i$ du côté gauche va induire une inversion matricielle

inversion) et dans le cas non linéaire, à une cascade de compositions de fonctions non linéaires qui s'avèrent parfois difficiles à expliciter et nécessitent le recours à des méthodes numériques.

Nous présentons ci-dessous, une procédure générale de la détermination de l'équation d'état formée à partir des différentes procédures proposées dans D. Karnopp et R.C Rosenberg [1983]

Procédure à suivre pour obtenir l'équation d'état

1°) Identifier les vecteurs $\dot{X}^i$. X^d . U et H

avec H : vecteur formé à partir de variables effort ou flux sortants des ports des éléments R couplés.

2°) formuler les équations des vecteurs

$$\dot{X}^i = \Phi_a\left(X^i, \dot{X}^d, U; H\right) \quad (a)$$

$$H = \Phi_b\left(X^i, U; H\right) \quad (b)$$

$$X^d = \Phi_c\left(X^i, U\right) \quad (c)$$

3°) Réduire si c'est possible (b) pour aboutir à

$$H = \Phi_d\left(X^i, U\right) \quad (d)$$

4°) Substituer (d) dans (a)

$$\dot{X}^i = \Phi_e\left(X^i, X^d, U\right) \quad (e)$$

5°) Calculer à partir de (c) et substituer le résultat dans (e)

$$\dot{X}^i = \Phi_f\left(X^i, \dot{X}^i, \dot{U}, U\right) \quad (f)$$

6°) Si c'est possible faire passer $\dot{X}^i$ du côté gauche

$$\dot{X}^i = \Phi_g\left(X^i, U, \dot{U}\right) \quad (g)$$

Nous terminons cette section par quelques remarques :

1°) Très souvent peu d'éléments X^i apparaissent dans (b) et (c) et aucun élément de U n'apparait.

2°) Un programme comme ENPORT [R.C Rosenberg 1974] produit directement pour un système linéaire, l'équation de type (g) à partir d'un modèle bond-graph.

3°) Si des causalités dérivées et des boucles algébriques entre des éléments résistifs existent simultanément, il faut anticiper l'élimination de certaines variables dépendantes. On peut utiliser, par exemple, la notion d'éléments équivalents (cf. D. Karnopp et R.C Rosenberg [1983])

III. Méthodes existantes (cas linéaire)

Cette section sera consacrée à l'étude et l'analyse des deux méthodes, rencontrées dans la littérature bond-graph, permettant de déterminer l'équation d'état.

III.1. Méthode de Rosenberg (méthode numérique)
ROSENBERG [1971]

La méthode de Rosenberg permet de générer les équations d'état de manière systématique, en opérant sur la représentation structurelle du bond-graph et sur la manipulation des relations liant les éléments. Des transformations algébriques permetteront d'éliminer les variables intermédiaires et d'aboutir à une équation d'état fonction des variables d'énergie.

Cette méthode se traduit par des procédures numériques utilisées dans plusieurs programmes comme ENPORT [R.C ROSENBERG 1974], CAMAS [J.F. Broenink et al. 1985] et MS-BOND [M. Delgado et al. 1988].

III.1.1 Présentation de la méthode

Nous présentons ici un bref résumé de la méthode, et sans chercher à entrer dans les détails et les différents cas de figures, nous énonçons directement l'équation d'état d'un système linéaire dans le cas le plus général.

La figure 3.1 montre le bloc diagramme de la structure générale d'un modèle bond-graph avec les différents champs et les vecteurs correspondants. La table 3.1 donne les caractéristiques et les relations entre les différents vecteurs.

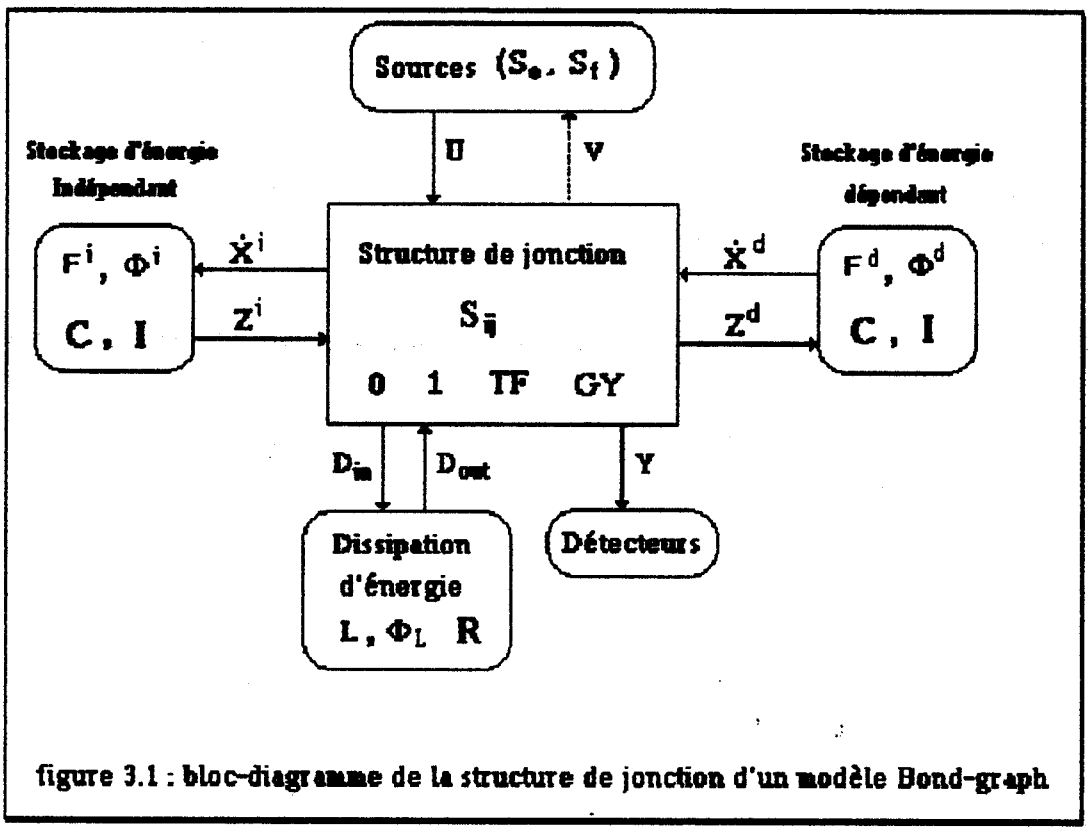


figure 3.1 : bloc-diagramme de la structure de jonction d'un modèle Bond-graph

Elément Bond-graph	Type de causalité	Relation E/ S des variables au port		Dimensions des vecteurs	Relations entre les vecteurs
R	R ⟵⊣	$D_{in} = f$	$D_{out}=e$	$D_{in} \in \Re^r$	$D_{out} = LD_{in}$
	R ⊢⟵	$D_{in} = e$	$D_{out}=f$	$D_{out} \in \Re^r$	
C	C ⟵⊣	$X^i = f$	$Z^i = e$	$X^i, Z^i \in \Re^n$	$Z^i = F^i X^i$
	C ⊢⟵	$Z^d = e$	$X^d = f$		
I	I ⊢⟵	$X^i = e$	$Z^i = f$	$X^d, Z^d \in \Re^d$	$X^d = (F^d)^{-1} Z^d$
	I ⟵⊣	$Z^d = f$	$X^d = e$		
S_e ; S_f	S_e —e→ f	$V = f$	$U = e$	$V \in \Re^p$	
	S_f —e→ f	$V = e$	$U = f$	$U \in \Re^p$	

Table 3.1

caractéristiques des vecteurs associés aux éléments bond-graphs.

L, F^i, F^d sont des matrices carrées, diagonales lorsque les multi-ports sont décomposés en ports simples. Le vecteur de sortie est représenté par $Y \in \Re^q$.

Au bloc diagramme de la figure 3.1, on associe l'équation suivante :

$$\begin{bmatrix} \dot{X}^i \\ Z^d \\ D_{in} \\ Y \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \begin{bmatrix} Z^i \\ \dot{X}^d \\ D_{out} \\ U \end{bmatrix} \qquad (3.8)$$

Pour des raisons logiques et physiques, certaines sous-matrices S_{ij} de la matrice de structure S, sont nulles. Ainsi $S_{22} = 0$, car deux variables dépendantes (éléments bond-graph I et C en causalité dérivée) ne doivent pas être directement causalement liées. Si pourtant cette situation intervient, il suffit de dualiser le chemin causal direct liant deux variables dépendantes pour récupérer deux variables indépendantes.

R.C. Rosenberg [1971] a posé aussi $S_{23} = 0$ pour les mêmes raisons, c'est-à-dire le fait de dualiser le chemin direct liant un élément résistif à un élément en causalité dérivée permettra de récupérer une variable indépendante.

Et finalement on a $S_{42} = 0$, car le vecteur de sortie ne doit pas s'exprimer explicitement en fonction des variables associées aux éléments en causalité dérivée (hypothèse simplificatrice généralement vérifiée).

D'autres sous-matrices de S vérifient les propriétés suivantes :

$$\begin{cases} S_{13} = -S_{31}^t \\ S_{11} = -S_{11}^t \\ S_{33} = -S_{33}^t \\ S_{21} = -S_{12}^t \end{cases} \qquad (3.9)$$

La manipulation algébrique de l'équation (3.8), s'appuyant sur les relations de (3.9) et sur la table 3.1, va engendrer l'équation d'état suivante :

$$\dot{X}^i = AX^i + BU + E\dot{U} \qquad (3.10)\ (*)$$

avec

$$A = \left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1}\left[S_{11} + S_{13} L\left(I - S_{33} L\right)^{-1} S_{31}\right] F^i$$

$$-S_{13} L\left(I - S_{33} L\right)^{-1} S_{32}\left(F^d\right)^{-1} S_{21} F^i$$

$$B = \left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1}\left[S_{14} + S_{13} L\left(I - S_{33} L\right)^{-1} S_{34}\right]$$

$$E = S_{13} L\left(I - S_{33} L\right)^{-1} S_{32}\left(F^d\right)^{-1} S_{24} + S_{12}\left(F^d\right)^{-1} S_{24}$$

III.1.2 Discussion autour de la méthode de Rosenberg

La méthode de Rosenberg est conçue pour générer une équation d'état sous forme numérique. Une tentative de paramétrage basée sur les expressions précédentes des matrices d'état s'avère difficile et complexe, voire même impossible. La cause en est le calcul matriciel formel, qui nécessite, surtout dans notre cas (matrices de taille importante), l'utilisation d'un grand système informatique pour toute automatisation et implantation sur calculateur. Ajoutons que les matrices mises en jeu (F^i, F^d, S_{ij} et L) possèdent, la plupart du temps, plusieurs composantes nulles, entraînant forcément une mauvaise utilisation de la mémoire et un accroissement non négligeable du temps d'exécution.

La théorie des bond-graphs implique une dualité entre les variables effort et flux. D'où toute relation causale induit implicitement l'existence de son duale en inversant le rôle des variables effort et flux. Il s'avère alors que si un élément en causalité dérivée est directement causalement lié à un élément dissipatif R $\left(S_{32}\right)_{kh} \neq 0$, alors inévitablement l'élément résistif

---

*R.C. Rosenberg ne présente pas la partie de l'équation d'état associée à la sortie $Y = CX + DU + F\dot{U}$ mais la démarche est la même et on arrive facilement à exprimer les matrices C, D et F.

est directement causalement lié à l'élément en causalité dérivée. Ce qui entraine que $(S_{23})_{hk} \neq 0$.

On a vu que Rosenberg pose $S_{23}=0$ car il suffit de dualiser le chemin direct liant un élément résistif à un élément en causalité dérivée pour récupérer une variable indépendante.

Pour les même raisons et à cause de la dualité des relations du bond-nous posons $S_{32}=0$.

Sur cette base nous simplifions les expressions précédentes des matrices d'état, ce qui du point de vue informatique diminuera largement la complexité des programmes.
On arrive à :

$$A = \left(I - S_{12}\left(F^{d}\right)^{-1} S_{21} F^{i}\right)^{-1}\left[S_{11} + S_{13} L\left(I - S_{33} L\right)^{-1} S_{31}\right] F^{i} \qquad (3.11)$$

$$B = \left(I - S_{12}\left(F^{d}\right)^{-1} S_{21} F^{i}\right)^{-1}\left[S_{14} + S_{13} L\left(I - S_{33} L\right)^{-1} S_{34}\right] \qquad (3.12)$$

$$E = S_{12}\left(F^{d}\right)^{-1} S_{24} \qquad (3.13)$$

III.2. Méthode des chemins (méthode paramétrique)
M. DELGADO [1991]

Cette méthode propose des résultats sous forme paramétrique, et d'autre part elle se rapproche un peu de la philosophie de la méthode que nous proposons dans le cadre de ce travail. Nous l'avons donc analysée pleinement afin de faire une comparaison objective avec celle que nous proposons. Cette analyse s'est portée à la fois sur sa capacité à traiter les différents cas de figures (vus précédemment) sensés être présents dans un modèle bond-graph, et sur son implantation informatique.

Dans un premier temps nous allons exposer les points clefs de la méthode et dans un deuxième temps nous discuterons son champ de validité.

III.2.1 Présentation de la méthode

Partant d'un élément indépendant (causalité intégrale) dans le sens de la variable effort ou flux sortant (suivant le type de l'élément), la méthode va parcourir un modèle bond-graph à la recherche des chemins liant cet élément aux autres éléments bond-graph de type $\{I, C, S_e, S_f\}$. Elle s'appuie sur l'utilisation des lois élémentaires associées aux éléments et sur celle des lois structurelles associées aux jonctions. Ces dernières seront interprétées sous forme de noeuds appelés croisement. Ainsi pour une jonction -0 (resp. 1) : $\Sigma f_i = 0$ (resp. $\Sigma e_i = 0$) vont correspondre à un croisement "*" qui a le même sens qu'un "et" logique et qui signifie que toutes ses branches descendantes doivent être explorées.

En revanche lorsque la loi structurelle est telle que $f_1 = ... = f_n$ ou $e_1 = ... = e_n$, la correspondance est alors un croisement "+" ("**ou**" logique) qui indique que seulement une de ses branches est à prendre en compte, et le passage à la suivante n'est possible que si la dernière ne peut atteindre un état terminal.

III.2.2 Discussion autour de la méthode des chemins :

Le traitement des différents types de causalité que peut avoir un champ constitue le point positif de cette méthode. Il permet de ne pas négliger un cas particulier des boucles algébriques, certes assez simple à déceler, mais qui demande néanmoins un traitement particulier. Ce type de boucles est induit par l'affectation d'une causalité mixte (intégrale et dérivée) autour d'un champ.

Le principe, basé sur l'écriture des lois structurelles et élémentaires associées à un bond-graph, est intéressant dans le cas de bond-graphs simples, mais devient très vite difficile à mettre en oeuvre et à implanter informatiquement dans le cas de problèmes plus complexes.

Ainsi L'application manuelle de ce principe, devient longue et fastidieuse dès que le modèle bond-graph possède au moins deux boucles algébriques entre des éléments dissipatifs (R-couplés), plusieurs éléments en causalité dérivée, et/ou des boucles causales algébriques entre des jonctions (nommées aussi boucles de causalité ou chemin causal fermé).

L'approche proposée par M. Delgado conduit aux remarques suivantes :

- Les restrictions citées plus haut n'ont pas été traitées, ce qui peut entraîner des problèmes de non décidabilité (le programme boucle indéfiniment). Donc problème de terminaison.
- Le parcours du bond-graph (ou de l'arbre des chemins) se base sur le principe des arbres binaires, limitant ainsi le traitement aux seules jonctions-0 et 1 liées au plus à 2 autres jonctions.
- Le noeud correspondant à un croisement "+" représente une situation non déterministe. L'absence des *heuristiques* se traduira par plusieurs traitement inutiles, loin d'être sans influence sur l'accroissement de la complexité du programme (consommation gratuite de l'espace mémoire alloué et augmentation sensible du temps d'exécution) (*). Donc risque d'une grande complexité.

-Le nombre important d'informations transparentes (utiles pour le traitement seulement) résidant dans la mémoire vive risque de saturer celle-ci, ainsi que le choix de la structure de données qui aggrave l'implantation informatique. Il est préférable, comme c'est le cas habituellement pour une exploration par profondeur, d'opter pour une pile qui optimise l'utilisation de la mémoire et permet aussi une sorte de mise en facteur des parties des chemins non disjoints pour qu'elles ne soient pas dédoublées comme cela a été fait.

III.3. Conclusion :

Nous venons d'exposer deux méthodes complètement différentes pour déterminer l'équation d'état. La première, connue sous le nom de méthode de Rosenberg, aboutit à une équation d'état sous forme numérique. Son implantation informatique requiert une mémoire très importante à cause du nombre élevé de calculs matriciels qu'elle nécessite ; et de la taille, généralement grande, de ses matrices. En outre, ces dernières possèdent généralement plusieurs composantes nulles entraînant des calculs inutiles et une mauvaise utilisation de la mémoire. Néanmoins, elle est la plus plausible théoriquement, d'une très grande efficacité et a fait ses preuves (implantée dans plusieurs programmes).

---

*En informatique et surtout en programmation logique, on connaît très bien les problèmes engendrés par une liaison avec un "ou" logique.

Nous l'utiliserons par la suite, pour prouver la validité de la méthode pratique, que nous proposons dans le cadre de ce travail.

La deuxième méthode (cf. M. Delgado [1991]) fournit une équation d'état sous forme paramétrique. Il s'est avéré qu'elle ne traite pas tous les cas de figure susceptibles d'exister dans un modèle bond-graph (cf. § II), et présente quelques faiblesses au niveau de sa validité algorithmique.

IV.Proposition d'une nouvelle méthode pour déterminer l'équation d'état à partir d'un bond-graph

(Méthode des Boucles et Chemins Causaux : MBCC)

Nous présentons ici une méthode permettant d'obtenir l'équation d'état d'un système linéaire sous forme d'expression formelle. Elle présente un certain nombre d'avantages ; elle est :

-**Complète** : elle ne néglige aucun cas de figure susceptible d'exister dans un modèle bond-graph (boucles algébriques, causalité dérivée).
-**Pratique** : elle ne se base que sur le parcours et sur l'analyse des chemins causaux et des boucles causales.
-**Facile à programmer** : elle ne nécessite aucun calcul matriciel et ne génère aucune situation non déterministe.
-**Rapide** : par rapport aux méthodes existantes et même sur un micro-ordinateur. Elle l'est également en utilisation manuelle.

Avant d'entrer dans les détails de la méthode, nous allons nous placer dans un certain nombre d'hypothèses. Ces dernières ne vont rien changer au champ de validité de la méthode. Elles constituent seulement une sorte de restriction voulue pour plus de clarté (simplification des notations) et pour faciliter les notations et la compréhension de notre démarche.

Hypothèse 3.1 : on ne s'intéresse qu'aux élément 1-port. Pour généraliser aux élément multi-ports, il suffit de les décomposer en élément simples.

Hypothèse 3.2 : aucun élément en causalité dérivée ni directement causalement lié avec les entrées (S_{24}=0), les sorties (S_{42}=0) et les éléments dissipatifs (cf. plus haut S_{23}=S_{32}=0).

IV.1 Rappel de la règle des boucles
[règle de Mason]

Les graphes sont utilisés depuis longtemps comme des outils de représentation des connaissances pour plusieurs disciplines scientifiques. Ils permettent assez souvent de dégager toutes les informations possibles pour analyser et représenter une situation ou un état quelconque du système étudié.

Dans le cadre de la modélisation des systèmes dynamiques, on distingue trois types de graphes :

- les graphes à flux ou graphe linéaire
- les graphes de fluence
- les bond-graphs

Un de leurs intérêts réside dans le fait qu'ils nous permettent de trouver des équations ou des relations mathématiques caractérisant l'effet d'une ou plusieurs entrées sur une ou plusieurs sorties. Le premier point commun de ces trois graphes est qu'ils peuvent tous être associés à un système d'équations linéaires caractérisant la relation d'entrée-sortie.

Plusieurs méthodes mathématiques (Gauss, pivot, Chovsky, Jordan, Cramer, etc...) peuvent résoudre ce système et permettent de déterminer le rapport entrée-sortie. Chacune de ces méthodes nécessitent soit une inversion matricielle, soit un nombre important d'opérations arithmétiques et surtout des divisions. D'où une grande *complexité* dans l'implantation de ces méthodes sur calculateurs et surtout la difficulté, voire même l'impossibilité de résolution dans le cas d'un système de grande dimension représenté formellement.

L'analyse des expressions du déterminant de la matrice à inverser et de sa matrice cofacteur ainsi que les méthodes de résolution précitées, et particulièrement la méthode de Cramer, ont permis d'aboutir aux résultats escomptés à partir des chemins et des boucles constituant le graphe.

C'est A. Tutsin [1952] qui a montré que le déterminant d'un système linéaire associé à un graphe à flux représentant un modèle électrique, est égal à l'unité plus la somme de tous les produits possibles, des gains des boucles

prises une à une, puis deux à deux, trois à trois, etc... En excluant les produits des boucles qui coïncident même partiellement et en alternant le signe algébrique.

La généralisation de ce résultat a été donnée par S. J. Mason [1956] qui a montré que le rapport entrée-sortie peut être déterminé par l'expression suivante, appelée règle des boucles et connue surtout sous le nom de règle de Mason :

$$G = \frac{\sum_k G_k \Delta_k}{\Delta} \qquad (3.14)$$

avec :

$$\Delta = 1 - \sum P_{m_1} + \sum P_{m_2} - \sum P_{m_3} + \cdots$$

P_{m_r} = Produit des gains de la $m^{ième}$ combinaison possible de r boucles disjointes.

G_k = gain du $k^{ième}$ chemin liant l'entrée à la sortie.

Δ_k = la valeur de Δ appliquée à la partie du graphe disjointe du $k^{ième}$ chemin.

C. S. Lorens [1964] a fait un rapprochement entre diverses méthodes de résolution d'un système d'équations et la règle des boucles ; il a montré également son application dans les domaines électrique, mécanique et stochastique. Il a démontré en outre que l'application de la règle de Cramer aux graphes à flux aboutit à la règle des boucles (On trouve notamment dans Robichaud. et al. [1962] une étude complète de cette approche).

Dans le cas des modèles bond-graphs, Paynter [1961], on a cherché à trouver à partir des règles simples, les équations modélisant un système dynamique. F. T. Brown [1972] a proposé donc une méthode permettant de retrouver et d'appliquer la règle de Mason sur un bond-graph à partir d'une analyse des chemins causaux et des boucles causales ; et ainsi, déterminer la fonction de transfert dans le cas d'un système linéaire en utilisant l'expression (3.14) avec la notion de causalité. Nous notons qu'on peut arriver au résultat de (3.14) par une méthode mathématique utilisant

l'équation d'état (3.15).

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX + DU \end{cases} \qquad (3.15)$$

avec :

$A \in \Re^n \times \Re^n$ $\quad n, p, q \in \aleph^+$

$B \in \Re^n \times \Re^p$ $\quad n$ = dim du système ou du vecteur X

$C \in \Re^q \times \Re^n$ $\quad p$ = dim du vecteur entrée U

$D \in \Re^q \times \Re^p$ $\quad q$ = dim du vecteur sortie Y

De cette équation on peut en déduire le rapport entrée/sortie (E/S) par

$$\frac{Y}{U} = C(sI - A)^{-1} B + D \qquad (3.16)$$

L'expression (3.16) comporte à la fois des produits et une inversion de matrices. Son calcul nécessite, dans le cas de systèmes complexes de grande dimension, un calculateur performant (gérant une mémoire vive importante) ainsi qu'un bon logiciel de calcul formel. Même dans ces conditions, les résultats ne sont pas toujours probants à cause du temps d'exécution ou d'une saturation de mémoire. D'où l'intérêt d'utiliser une méthode de type règle de Mason lorsque cela est possible.

Nous rappelons ici que notre but n'est pas de trouver la fonction de transfert, mais de proposer une méthode à la fois simple et rapide permettant de trouver l'équation d'état sous forme d'expressions formelles.

Pour atteindre notre objectif, nous allons nous appuyer dans notre démonstration sur la relation implicite qui existe entre (3.14) et (3.16) et sur les expressions mathématiques (3.17) de l'équation d'état d'un système représenté par un bond-graph déduites de la méthode de Rosenberg.

$$A = \left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1} \left[S_{11} + S_{13} L (I - S_{33} L)^{-1} S_{31}\right] F^i \qquad (3.17.a)$$

$$B = \left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1} \left[S_{14} + S_{13} L (I - S_{33} L)^{-1} S_{34}\right] \qquad (3.17.b)$$

$$C = \left[S_{41} + S_{43} L (I - S_{33} L)^{-1} S_{31} \right] F^{i} \qquad (3.17.c)$$

$$D = \left[S_{44} + S_{43} L (I - S_{33} L)^{-1} S_{34} \right] F^{i} \qquad (3.17.d)$$

L'idée de base est de substituer à chaque expression du type (3.17) une pseudo règle de Mason type (3.14).

Du point de vue pratique, chaque composante de la matrice d'état caractérise une relation entre deux noeuds, l'un jouant le rôle provisoire d'une sortie et l'autre celui d'une entrée. Chacune de ces relations correspond à la détermination des chemins et des boucles dans un graphe réduit par rapport au graphe initial (on ne s'intéresse chaque fois qu'à une partie du système : relation entre X et X, U et X, X et Y, U et Y).

Remarque :
Dorénavant, nous allons utiliser les expressions (3.17) pour les matrices d'état A, B, C et D.

IV.2. Méthode proposée dans le cadre de ce travail

On va chercher l'équation d'état associée à un système dynamique représenté par un bond-graph à partir de l'analyse des chemins causaux et des boucles causales.

L'idée est d'appliquer la règle des boucles, explicitée précédemment, aux expressions des matrices d'état A, B, C et D. Chaque expression mathématique de ces matrices met en évidence l'existence d'une relation entre deux composantes de deux vecteurs. Ainsi la détermination des chemins liant les éléments du vecteur d'état X^{i} à ceux de $\dot{X}^{i}$ permet de former la matrice A. Ceux liant les vecteurs U à $\dot{X}^{i}$ vont constituer la matrice B etc...

Ces chemins causaux peuvent être élémentaires, directs ou indirects (passant par un ou plusieurs éléments résistifs ou passant par des éléments I et C en causalité dérivée) [AZMANI.A. et al. 1990].

Traitons en exemple l'expression (3.17.a) de la matrice d'état A :

Chaque composante a_{ij} de la matrice correspond à une relation entre deux éléments dynamiques : l'un représente une composante x^i_j du vecteur d'état X^i et l'autre une composante $\dot{X}^i_i$ de son vecteur dérivé $\dot{X}^i$.

la valeur de a_{ij}, pour $i \neq j$, est égale au gain du chemin causal direct liant ces deux éléments : $(S_{11})_{ij}$ multiplié par le gain de l'élément bond-graph associé à la composante x^i_j du vecteur x^i : $(F^i)_{jj}$. L'existence d'éventuelles boucles algébriques entre des éléments dissipatifs va induire un gain supplémentaire :

$$\left(\left[S_{13}L\left(I - S_{33}L\right)^{-1}S_{31}\right]F^i\right)_{ij}$$

Si i=j, a_{ij} représente la relation liant $\dot{X}^i_j$ à X^i_j. C'est aussi le gain du chemin causal indirect liant l'élément bond-graph, associé à la fois à X^i_j et à X^i_j, en passant par un ou plusieurs éléments R : $\left(\left[S_{13}L\left(I - S_{33}L\right)^{-1}S_{31}\right]F^i\right)_{jj}$; multiplié par le module de l'élément bond-graph associé à la composante X^i_j du vecteur X^i : $(F^i)_{jj}$.

L'existence d'une relation, entre une composante x^i_j du vecteur d'état X^i et une composante x^d_j de X^d (élément I ou C en causalité dérivée), va engendrer un multiplicateur supplémentaire, correspondant au rapport du gain du chemin causal direct liant ces deux éléments (ou plus exactement les éléments bond-graph qui leur sont associés) par un scalaire caractérisant une relation algébrique, formée à partir des différentes liaisons causales existant entre les composantes de X^i et X^d :

$$\left[I - S_{12}\left(F^d\right)^{-1}S_{21}F^i\right]^{-1}_j$$

En conclusion de cette section nous avons :

$$a_{ij} = \left(\left[I - S_{12} \left(F^d \right)^{-1} S_{21} F^i \right]^{-1} \right)_{ii} \left[S_{11} F^i + S_{13} L \left(I - S_{33} L \right)^{-1} S_{31} F^i \right]_{ij} \quad (i \neq j) \tag{3.19.a}$$

$$a_{jj} = \left(\left[I - S_{12} \left(F^d \right)^{-1} S_{21} F^i \right]^{-1} \right)_{jj} \left[S_{13} L \left(I - S_{33} L \right)^{-1} S_{31} F^i \right]_{ji} \tag{3.19.b}$$

Dans ce qui va suivre, nous allons analyser les expressions des matrices d'état cas par cas et nous allons montrer comment on y retrouve la règle de Mason. Nous proposerons alors des règles et des méthodes simples et facilement programmables, basées sur la connaissance des chemins causaux et boucles causales. (*)

VI.2.1. MATRICE D'ETAT A

1° étape : Pas de causalité dérivée

Intéressons nous dans un premier temps, au terme

$$S_{13} L \left(I - S_{33} L \right)^{-1} S_{31} F^i$$

-la sous-matrice S_{33} de la matrice de structure S caractérise l'existence d'une liaison causale directe entre deux éléments de type dissipatif d'énergie (éléments résistifs).

-L est une matrice diagonale (avec l'hypothèse 3.1 : éléments de 1-port type R) telle que L_{hh} = **gain de** R_h . avec h indice indiquant le rang de l'élément R dans les vecteurs D_{in} et D_{out}.

$L_{hh} = R_h$ si R_h a une causalité effort sortant R_h

$L_{hh} = 1/R_h$ si R_h a une causalité effort entrant R_h

Le produit matriciel $S_{33}L$ correspond à une matrice de mêmes dimensions que S_{33} , et dont les composantes sont de la forme suivante :

$$\left(S_{33} L \right)_{kh} = \left(S_{33} \right)_{kh} \left(L \right)_{hh}$$

---

* Le chapitre 4 comprend tous les algorithmes associés à cette méthode.

Ce qui s'interprète graphiquement par : le produit du gain du chemin causal liant R_k à R_h par le module de R_h .

La matrice $(I - S_{33}L)^{-1}$ ne fait intervenir donc que des éléments de type R (dissipatifs), elle rappelle également le terme $(sI - A)^{-1}$ de l'expression (3.16), avec $S_{33}L$ matrice qui joue le rôle de A. Cependant on remarque l'absence de l'opérateur de Laplace "s" dans la première matrice. Cette absence est liée au fait que A est la matrice d'état qui caractérise la relation entre les éléments bond-graph I et C, dont l'impédance dépend de l'opérateur "s". Alors que $S_{33}L$ ne spécifie que la relation entre des éléments R, dont l'impédance est indépendante de s. (*)

D'après la pseudo équivalence entre (3.14) et (3.16), on peut conclure que $(I - S_{33}L)^{-1}$ peut être déterminée par l'application de la règle de Mason à un bond-graph réduit ne contenant que les relations causales entre les éléments de type R (en éliminant les éléments I, C, S_e et S_f). Dorénavant cette partie du bond-graph s'appellera la *partie résistive*.

Nous pouvons écrire que :

$$\boxed{(I - S_{33}L)^{-1} = \frac{\Gamma^R}{\Delta^R}} \qquad (3.20)(**)$$

Avec :

$\Delta^R = 1 - \sum$ des gains des boucles entre deux éléments de type R
$+ \sum$ des produits des gains de 2 boucles disjointes entre des éléments de type R
$- \sum$ des produits des gains de 3 boucles disjointes entre des éléments de type R
$+ \ldots$

---

* l'impédance d'un élément 1-port |------ Z est R, I s ou 1/C s
l'impédance d'un élément 1-port ------| Z est 1/R, 1/I s ou C s

** l'égalité (3.20) est toujours vraie du point de vue mathématique, ce qui change c'est la spécificité graphique de Δ^R et de Γ^R.

Γ^R est la matrice *adjointe* de $(I - s_{33}L)$ dont une composante quelconque Γ^a_{kh} est un mineur d'ordre inférieur à celui de Δ^R. Pour déterminer Γ^a_{kh}, la méthode mathématique consiste à éliminer de $(I - s_{33}L)$ la ligne h et la colonne k et à effectuer le calcul du déterminant de la matrice restante. Ce qui, du point de vue pratique, revient à l'éliminer de Δ^R tout terme possédant une occurrence de R_k et R_h.

Nous concluons alors que Γ^a_{kh} est le cofacteur, par rapport à R_k et R_h, de Δ^R. Nous verrons plus loin que la détermination pratique de Γ^a_{kh} diffère selon qu'on cherche les élements de la diagonale ou les éléments hors-diagonaux.

De même la sous-matrice S_{13} caractérise une relation causale entre un élément de type R_k et un élément de $\dot{X}^i$. Ainsi la forme des composantes de la matrice produit $S_{13}L$ est la suivante :

$$(S_{13}L)_{jk} = (S_{13})_{jk} (L)_{kk}$$

Elle s'interpréte graphiquement par : le produit du gain du chemin liant un élément R_k à l'élément bond-graph I ou C associé à la composante $\dot{X}^i_j$ de $\dot{X}^i$, par le module (ou gain) de R_k.

La sous-matrice S_{31} de S caractérise une relation entre Z^i et D_{in}. Ses composantes non nulles reflètent l'existence d'une relation causale entre un élément de type I ou C (en causalité intégrale) et un élément de type R.

F^i, matrice diagonale (cf. hypothèse 3.1), met en évidence le module v d'un élément dynamique de type I ou C tel que $(F^i)_{jj} = 1/v$. Or d'après la table (3.1) $Z^i = F^i X^i$. Ainsi une composante $(S_{31}F^i)_{hj}$ correspond au gain de la liaison causale entre v_j et R_h multiplié par le module de v_j.

Poursuivons cette approche graphique pour le terme $S_{11}F^i$ de l'équation (3.19.a).

La sous-matrice S_{11} de S caractérise les relations causales entre Z^i et $\dot{X}^i$. Ses composantes indiquent l'existence d'un chemin causal direct entre deux éléments bond-graph de type I ou C en causalité intégrale. Aussi, une composante $(S_{11}F^i)_{ij}$ représente-t-elle le module de l'élément bond-

graph associé à x_j^i ($[Fi]_{jj}$ = 1/I ou 1/C) multiplié par le gain ($[S_{11}]_{ij}$) du chemin causal direct liant deux éléments bond-graph I ou C en causalité intégrale. Le premier est associé à la composante x_j^i de Xi de même type et de même causalité, le second est associé à $\dot{x}_i^i$.

Après cette phase d'approche graphique des différent termes matriciels (avec la condition de l'étape1) de (3.19.a) pris séparément, nous généralisons la démarche à l'ensemble de l'expression de a_{ij}.

Soit :

$$a_{ij} = \left[S_{13}L(I - S_{33}L)^{-1}S_{31}F^i\right]_{ij} + \left[S_{11}F^i\right]_{ij}$$

qu'on peut écrire aussi sous la forme :

$$a_{ij} = \left[(S_{13}L)\frac{\Gamma^R}{\Delta^R}(S_{31}F^i)\right]_j + \left[S_{11}F^i\right]_{ij}$$

avec $i,j = 1, \ldots, n$ $\quad n=\dim(X^i)$

$k,h = 1, \ldots, r$ $\quad r=\dim(D_{out})=\dim(Din)$

En effectuant les produits matriciels, on a :

$$a_{ij} = \frac{1}{\Delta^R}\sum_{h=1}^{r}\left[\sum_{k=1}^{r}\left(\left[S_{13}L\right]_{ik}\Gamma^R_{kh}\left[S_{31}F^i\right]_{hj}\right)\right] + \left[S_{11}F^i\right]_{ij}$$

Ce qui donne

$$a_{ij} = \frac{1}{\Delta^R}\sum_{h}\sum_{k}\left[S_{13}L\right]_{ik}\left[S_{31}F^i\right]_{hj}\ \Gamma^R_{kh} + \left[S_{11}F^i\right]_{ij} \qquad (3.22)$$

Qui peut s'écrire également sous la forme :

$$a_{ij} = \frac{1}{\Delta^R}\begin{bmatrix} (S_{13}L)_{i1}(S_{31}F^j)_{1j}\Gamma^R_{11} + \ldots + \ldots + (S_{13}L)_{i1}(S_{31}F^j)_{rj}\Gamma^R_{r1} + \\ (S_{13}L)_{i2}(S_{31}F^j)_{1j}\Gamma^R_{12} + \ldots + \ldots + (S_{13}L)_{i2}(S_{31}F^j)_{rj}\Delta^R_{r2} + \\ + \ldots + \quad + \ldots + \ldots \quad + \ldots + \\ (S_{13}L)_{ir}(S_{31}F^j)_{1j}\Gamma^R_{1r} + \ldots + \ldots + (S_{13}L)_{ir}(S_{31}F^j)_{rj}\Gamma^R_{rr} \end{bmatrix} + \left[S_{11}F^j\right]_{ij}$$

(3.23)

L'architecture d'un bond-graph entraine que les sous matrices S_{13} et le transposé de sa matrice anti-symétrique S_{31} contiennent plusieurs composantes nulles. Et seules celles qui correspondent aux gains d'un chemin causal sont différentes de zéro. On conclut que l'équation (3.23) peut être représentée par une expression qui ne fait apparaître que les termes non nuls, c'est-à-dire ceux qui correspondent à des gains de chemins. Posons alors, l'équation (3.23) égale à

$$a_{ij} = \frac{\left(\sum_{k,h} G_{kh}\ \Gamma^R_{kh}\right)_{ij}}{\Delta^R} \qquad (3.24)$$

et interprétons les termes G_{kh} et Γ_{kh}.

Nous avons signalé que le calcul de Γ^R_{kh} est différent pour les termes diagonaux et les termes hors-diagonaux, cette différence existe également pour G_{kh}. D'où la nécessité de traiter séparément le cas de la diagonale de celui des autres.

a_{jj} est une composante de la diagonale qui caractérise une relation causale entre x_j et $\dot{X}_j$. Ces derniers sont représentés sur un bond-graph par le même élément I ou C en causalité intégrale. Leur liaison, lorsqu'elle existe, est matérialisée par un chemin causal indirect passant par un et un seul élément résistif. Il est clair (du fait que le même élément bond-graph représente les deux composantes du vecteur d'état et de son vecteur dérivé) que le même élément R va lier ces deux composantes. Ce qui nous permet, en tenant compte aussi de (3.19.b), d'adapter les équations (3.23) et (3.24) aux termes de la diagonale :

$$a_{ij} = \frac{1}{\Delta^R}\left[(S_{13}L)_{j1}(S_{31}F)_{1j}\Gamma^R_{11} + (S_{13}L)_{j2}(S_{31}F)_{2j}\Gamma^R_{22} + \ldots + (S_{13}L)_{jr}(S_{31}F)_{rj}\Gamma^R_{rr}\right] \quad (3.25)$$

soit, à cause aussi de la présence de plusieurs termes nuls dans (3.25) :

$$a_{ij} = \frac{\left(\sum_{kk} G_{kk}\ \Gamma^R_{kk}\right)_{jj}}{\Delta^R} \quad (3.26)$$

avec G_{kk} gain du chemin causal indirect, passant par un élément résistif R_k, liant $\dot{X}^i_j$ à X^i_j multiplié par le module de l'élément bond-graph associé à X^i_j. Ici il est évident que l'existence d'un chemin direct, liant un élément E de type I ou C en causalité intégrale à un élément dissipatif R_k, va induire l'existence d'un chemin indirect passant par R_k et liant les composantes de X^i et $\dot{X}^i$ associées à E.

Ainsi G_{kk}, que nous notons tout simplement G^R_k, correspond aussi au gain du kième chemin liant X^i_j à $\dot{X}^i_j$ et passant par R_k.

Γ^R_{kk} se calcule de la même manière que Δ^R en éliminant la $k^{ième}$ ligne et la $k^{ième}$ colonne. Ce qui revient à éliminer de Δ^R tout terme possédant une occurrence de l'élément dissipatif R intervient dans G^R_k. Or on a vu que Δ^R n'utilise que la partie du modèle bond-graph caractérisant les relations causales entre des éléments dissipatifs (de type R), que nous appellons la "**partie résistive**". D'où l'élimination de l'élément R intervenant dans G^R_k, revient à supprimer de la partie résistive toute intersection avec le chemin associé à G^R_k.

Nous sommes en mesure de noter que Γ^R_{kk}, qu'on peut représenter également par Δ^R_k, se calcule de la même manière que Δ^R sur la partie résistive du modèle bond-graph disjointe avec le chemin affilié à G^R_k.

Nous pouvons réécrire maintenant l'équation (3.26) sous la forme de la règle de Mason, soit :

$$a_{ij} = \frac{\left(\sum_k G_k^R \, \Delta_k^R \right)_{ij}}{\Delta^R} \qquad (3.27)$$

Nous n'allons pas donner ici les définitions des termes de (3.27) (*). Bien entendu un rappel sera donné à la fin de l'étape1.

La détermination des termes hors-diagonaux a_{ij} va aboutir au même résultat, mais la démarche à suivre est plus complexe. Ce qui rend moins évident la justification d'une approche graphique des équations (3.23) et (3.24). Cette difficulté est liée surtoût aux éléments hors-diagonaux de Γ^R qui ne s'expriment pas graphiquement de la même manière que Δ^R.

Pour éviter de compliquer à ce niveau la compréhension de notre démarche nous allons annoncer et utiliser, sans le prouver, un certain nombre de résultats. Nous tenons à rappeler que cela n'affecte la validité ni la généralité de nos résultats, . Signalons, toutefois, que l'annexe 3.1 et l'exemple qui va suivre, apporteront plus d'éclaircissement pour une démonstration détaillée.

Ainsi Γ_{kk}^R avec $k \neq h$ représente la somme de tous les chemins (directs et indirects) liant R_k à R_h multiplié par un mineur formé par application de Δ^R à la partie résistive, disjointe avec le chemin précédent. Il est clair que si cette liaison n'existe pas $\Gamma_{kk}^R = 0$.

Nous anonçons alors le résultats suivant : $\Gamma_{kk}^R = \left(\sum_{l=1}^{nr} g_l^R \, \delta_l \right)_{kh}$

avec

- nr : nombre de chemins liant R_k à R_h
- g_l^R : gain du $l^{ième}$ chemin liant R_k à R_h.
- δ_l : Δ^R appliquée sur la partie résistive disjointe avec le $l^{ième}$ chemin précédent.

L'insertion de cette définition dans (3.23) va ajouter une somme

---

*on signale que les spécificité de la règle de Mason restent valables et qu'il faut pour le moment se reporter à ce qui précède

supplémentaire qui du point de vu calcul numérique complique encore l'expression. Mais qui, dans une interprétation graphique, va correspondre tout simplement au gain d'un chemin causal.

Réécrivons l'équation (3.34) avec ces nouveaux paramètres, soit :

$$a_{ij} = \frac{1}{\Delta^R} \begin{bmatrix} (S_{13}L)_{i1}(S_{31}F^i)_{1j}\left(\sum_{l=1}^{nr} g_l^R \delta_l\right)_{11} & + \ldots + & (S_{13}L)_{i1}(S_{31}F^i)_{rj}\left(\sum_{l=1}^{nr} g_l^R \delta_l\right)_{r1} & + \\ \hline \ldots & + \ldots + & \ldots + & \\ \hline (S_{13}L)_{ir}(S_{31}F^i)_{1j}\left(\sum_{l=1}^{nr} g_l^R \delta_l\right)_{1r} & & (S_{13}L)_{ir}(S_{31}F^i)_{rj}\left(\sum_{l=1}^{nr} g_l^R \delta_l\right)_{rr} & \end{bmatrix}$$

$$+ (S_{11}F^i)_{ij}$$

Qu'on peut simplifier, en éliminant les termes nuls et en se rendant compte que ceux qui sont différents de zéro se caractérisent par le gain d'un chemin causal direct, sous la forme :

$$a_{ij} = \frac{1}{\Delta^R} \sum_{l=1}^{nc} \left(g^{\dot{X}_i R_k} g^{R_h R_k} g^{R_h X_j}\right)_l \delta_l^{R_k R_h} + \left(S_{11}F^i\right)_{ij} \qquad (3.28)$$

avec

nc =nombre des chemins causaux comportant une succession de liaisons : de X_j^i à un élément résistif R_h, de R_h à un élément de même type R_k et de R_k à $\dot{X}_i^i$.

$g^{R_h X_j}$=gain du chemin direct liant X_j^i à R_h multiplié par le module de l'élément bond-graph associé à X_j^i.

$g^{R_h R_k}$ = gain du chemin direct ou indirect liant R_h à R_k multiplié par le module de R_h.

$g^{\dot{X}_i R_k}$ = gain du chemin direct liant R_k à $\dot{X}_i^i$ multiplié par le gain de R_k.

$\delta_l^{R_k R_h} = \Delta^R$ appliqué sur la partie résistive du modèle bond-graph, dans laquelle on élimine le chemin causal associé à $g^{R_h R_k}$.

A ce niveau on ne peut pas dire que le premier terme de (3.28)

correspond à la définition de la règle de Mason, bien qu'il en ait la forme. La raison en est $g^{\dot{X}_i R_k} g^{R_h R_k} g^{R_h \dot{X}_j}$ dont le chemin associé est toujours disjoint de la partie resistive adjointe à $\delta_l^{R_k R_h}$. Or le chemin précédent contient nécessairement celui liant X_j^i à $\dot{X}_i^i$. D'où $g^{\dot{X}_i R_k} g^{R_h R_k} g^{R_h \dot{X}_j}$ est égal au gain du chemin causale directe liant X_j^i à $\dot{X}_i^i$ multiplié par le gain de la boucle causale formée par R_k et R_h. La mise en facteur de ce produit permet de réécrire (3.28) sous la forme :

$$a_{ij} = \frac{1}{\Delta^R} \sum_{l=1}^{nc} \left(g^{\dot{X}_i \dot{X}_j} G^{R_k R_h} \right)_l \delta_l^{R_k R_h} + G^{\dot{X}_i \dot{X}_j} \qquad (3.29)$$

avec

$G^{\dot{X}_i \dot{X}_j} = - g^{\dot{X}_i \dot{X}_j}$ = gain du chemin causal liant X_j^i à $\dot{X}_i^i$ multiplié par le module de l'élément bond-graph associé à X_j^i.

$G^{R_k R_h}$ = gain du chemin liant R_h à R_k multiplié par les modules de R_h et R_k, c'est aussi à un signe prés le gain de la boucle causale formée par ces deux éléments.

$\delta_l^{R_k R_h}$ garde la définition antérieure.

Le produit de $G^{R_k R_h}$ par $\delta_l^{R_k R_h}$, du fait que ces deux termes sont déterminés sur deux parties disjointes du modèle bond-graph, forme la partie de Δ^R qui contient tous les termes possédant une occurrence de R_k et de R_h. Il est facile de vérifier que ce produit correspond à la partie de Δ^R qui fait intérvenir les boucles formées par R_h et Rk et que nous notons $\Delta^{R_k R_h}$.
Nous pouvons réécrire alors l'équation (3.29) en réduisant au même dénominateur ses deux termes :

$$a_{ij} = \frac{1}{\Delta^R} \left[\sum_{l}^{nc} \left(-G^{\dot{X}_i \dot{X}_j} \Delta^{R_k R_h} \right)_l + G^{\dot{X}_i \dot{X}_j} \Delta^R \right]$$

qui peut s'écrire, sachant que la somme est associé cette fois-ci au nombre de chemin réels existant entre X_j^i et $\dot{X}_i^i$, comme suit :

$$a_{ij} = \frac{1}{\Delta^R} \sum_{l}^{nc} G_l^{\dot{X}_i \dot{X}_j} \left(\Delta^R - \Delta^{R_k R_h} \right)_l$$

Il devient évident de constater que la seule partie résistive non disjointe avec le chemin liant x_j^i à $\dot{X}_i^i$ est celle qui correspond à $\Delta^{R_k R_h}$. D'où, celle caractérisée par $(\Delta^R - \Delta^{R_k R_h})$ est nécessairement disjointe avec le chemin dont le gain est représenté par $G^{\dot{X}_i x_j}$. Ce qui s'accorde totalement avec la définition de la règle de Mason. Et nous arrivons ainsi au résultat escompté, à savoir que la règle donnée précédemment (pour déterminer les termes de la diagonale de la matrice d'état A) reste valabe pour les termes hors-diagonaux.

Finalement nous proposons, pour déterminer la marice d'état A d'un modèle bond-graph ne possédant pas d'élément en causalité dérivée, la règle suivante :

$$a_{ij} = \frac{\left(\sum_k G_k \, \Delta_k^R \right)_{ij}}{\Delta^R} \qquad (3.30)$$

avec :

Δ^R égal au déterminant de la ègle de Mason appliqué à la partie résistive du modèle bond-graph (cf. plus haut).

$(G_k)_{ij}$ = module de l'élément bond-graph I ou C associé à x_j^i multiplié par le gain du $k^{ième}$ chemin causal direct ou indirect (passant par des éléments dissipatifs R), et liant les éléments I ou C représentant x_j^i et $\dot{X}_i^i$.

Δ_k^R = Δ^R calculé sur la partie résistive du module bond-graph disjointe avec le $k^{ième}$ chemin associé à $(G_k)_{ij}$.

Exemple d'application

Nous allons déterminer de deux façons la matrice A de l'équation d'état du modèle bond-graph de la figure 3.2. Dans la première on va calculer l'expression (3.17.a) (méthode de Rosenberg) et dans la seconde on va utilisée la MBCC. Cette exemple simple, mais comportant des boucles

algébriques entre des éléments dissipatifs, permettra à la fois d'appliquer une partie de la méthode proposée, et donnera aussi un aperçu sur ses avantages.

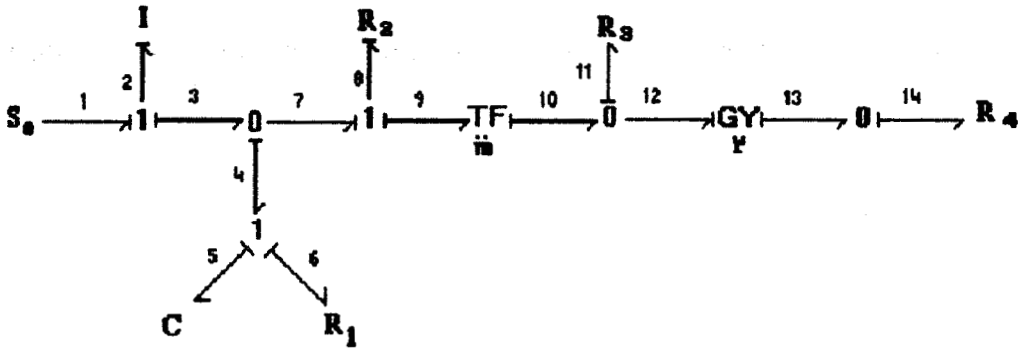


figure 3.2 : exemple de modèle bond-graph possédant des boucles algébrique entre des éléments R

L'équation du bloc diagramme (cf. figure 3.1) représentant le modèle bond-graph de la figure 3.2, est la suivante :

$$\begin{bmatrix} e_2 \\ f_5 \\ f_6 \\ e_8 \\ f_{11} \\ f_{14} \end{bmatrix} = \left(\begin{array}{cc|cccc|c} 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & -m & 0 & 0 \\ 0 & 0 & 0 & m & 0 & -r & 0 \\ 0 & 0 & 0 & 0 & r & 0 & 0 \end{array}\right) \begin{pmatrix} f_2 \\ e_5 \\ e_6 \\ f_8 \\ e_{11} \\ e_{14} \\ e_1 \end{pmatrix}$$

L'absence des éléments en causalité dérivée entraîne celle des sous-matrices S_{2j} et S_{i2} avec i,j=1,...,4. Mais pour ne pas modifier les notations on considère comme si ces sous matrices existent virtuellement.

Avant d'écrire la matrice d'état A sous sa forme finale, nous allons commencer par exprimer chaque terme de l'expression (3.17.a).

-Relations causales directes entre les éléments indépendants de stockage d'énergie :

$$S_{11}F^{i} = \begin{pmatrix} 0 & -1/C \\ 1/I & 0 \end{pmatrix}$$

-Relations causales directes entre les éléments dissipatifs et les éléments indépendants de stockage :

$$S_{13}L = \begin{pmatrix} -R_1 & 0 & 0 & 0 \\ 0 & -1/R_2 & 0 & 0 \end{pmatrix} \qquad S_{31}F^{i} = \begin{pmatrix} 1/I & 0 \\ 0 & 1/C \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

- Relations causales entre des éléments déssipatifs (boucles algébriques entre des R)

$$(I - S_{33}L) = \begin{bmatrix} 1 & 1/R_2 & 0 & 0 \\ -R_1 & 1 & mR_3 & 0 \\ 0 & -m/R_2 & 1 & rR_4 \\ 0 & 0 & -rR_3 & 1 \end{bmatrix}$$

Déterminons l'inverse de cette matrice, en rappelant que :

$$(I - S_{33}L)^{-1} = \frac{\text{adjoint}(I - S_{33}L)}{\text{determinant}(I - S_{33}L)}$$

Commençons par le calcul du déterminant, on a :

$$\det(I - S_{33}L) = 1 + \frac{R_1}{R_2} + r^2R_3R_4 + m^2\frac{R_3}{R_2} + \frac{R_1}{R_2}r^2R_3R_4$$

Comme on va le voir plus bas, ce déterminant est égal à Δ^R.

En calculant la comatrice ou la matrice cofacteur de (I - S_{33}L) et en la transposant (adjoint(M)= [cofacteur(M)]^t), on arrive à :

$$\text{adj}(I - S_{33}L) = \begin{bmatrix} 1+r^2R_3R_4 & \frac{1}{R_2}(1+r^2R_3R_4) & \frac{1}{R_2}mR_3 & \frac{1}{R_2}mR_3rR_4 \\ R_1(1+r^2R_3R_4) & 1+r^2R_3R_4 & -mR_3 & mR_3rR_4 \\ \frac{mR_1}{R_2} & \frac{m}{R_2} & 1+\frac{R_1}{R_3} & -rR_4\left(1+\frac{R_1}{R_2}\right) \\ R_1\frac{m}{R_2}rR_3 & \frac{m}{R_2}rR_3 & rR_3\left(1+\frac{R_1}{R_2}\right) & 1+\frac{mR_3}{R_2}+\frac{R_1}{R_2} \end{bmatrix}$$

En réalisant les produits des termes intervenant dans (3.17.a) à savoir S_{13} L adj(I - S_{33} L) S_{31} F^i on arrive à :

$$\begin{pmatrix} -\frac{R_1}{I}\left(1+r^2R_3R_4+m\frac{R_3}{R_2}\right) & -R_1\frac{1}{C}\frac{-1}{R_2}\left(1+r^2R_3R_4\right) \\ -\frac{1}{R_2}\frac{R_1}{I}\left(1+r^2R_3R_4\right) & -\frac{1}{R_2}\frac{1}{C}\left(1+r^2R_3R_4\right) \end{pmatrix}$$

En additionnant avec $S_{11}F^i$ on trouve :

$$A=\begin{bmatrix} \frac{-R_1}{I}\frac{1+r^2R_3R_4+m\frac{R_3}{R_2}}{\Delta^R} & \frac{-1}{C}+\frac{1}{C}\frac{R_1}{R_2}\frac{\left(1+r^2R_3R_4\right)}{\Delta^R} \\ \frac{1}{I}+\frac{1}{I}\frac{R_1}{R_2}\frac{\left(1+r^2R_3R_4\right)}{\Delta^R} & \frac{-1}{R_2C}\frac{\left(1+r^2R_3R_4\right)}{\Delta^R} \end{bmatrix}$$

Et finalement en rendant au même dénominateur Δ^R et en effectéuant les opréartions arithmétiques on arrive à la forme finale de la matrice A :

$$A=\frac{1}{\Delta^R}\begin{bmatrix} \frac{-R_1}{I}\left(1+r^2R_3R_4+m\frac{R_3}{R_2}\right) & \frac{-1}{C}\left(1+r^2R_3R_4+m\frac{R_3}{R_2}\right) \\ \frac{1}{I}\left(1+r^2R_3R_4+m\frac{R_3}{R_2}\right) & \frac{-1}{R_2C}\left(1+r^2R_3R_4\right) \end{bmatrix}$$

La table 3.2 résume toutes les étapes de l'application de la MBCC sur l'exemple de la figure 3.2.

On peut constater qu'on arrive au même résultat plus rapidement et sans faire aucun calcul matriciel.

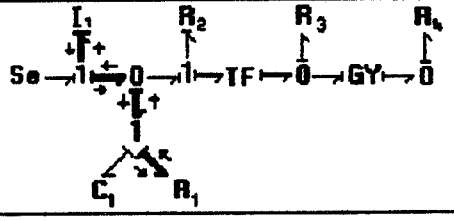
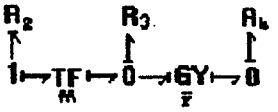
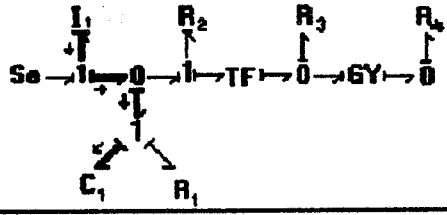
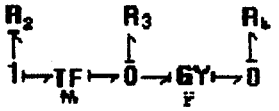
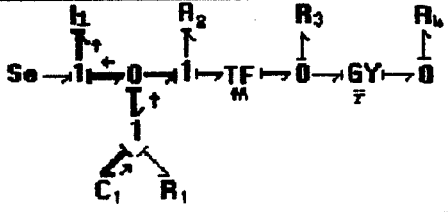
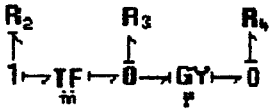
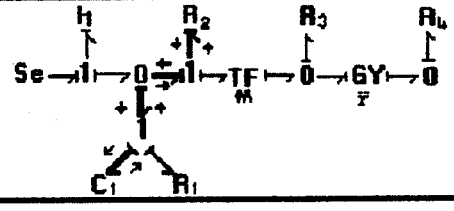
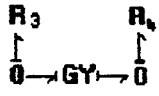
Relation entre X_j et $\dot{X}_i$	Chemin causal liant X_j à $\dot{X}_i$	Module bond-graph réduit du chemin $X_j \longrightarrow \dot{X}_i$	G^R_{ij}	Δ^R_{ij}
X_1 à $\dot{X}_1$			$\frac{-R_1}{I_1}$	$1+\frac{R_3}{m^2R_2}r^2R_3R_4$
X_1 à $\dot{X}_2$			$\frac{-1}{C_1}$	$1+\frac{R_3}{m^2R_2}r^2R_3R_4$
X_2 à $\dot{X}_1$			$\frac{1}{I_1}$	$1+\frac{R_3}{m^2R_2}r^2R_3R_4$
X_2 à $\dot{X}_2$			$\frac{-1}{R_2C_2}$	$1+r^2R_3R_4$

Table 3.2 : application de la MBCC sur l'exemple de la figure 3.2

2° étape : Existence d'éléments I ou C en causalité dérivée

On a vu plus haut que l'existence d'éléments en causalité dérivée induit un gain supplémentaire égal à :

$$\left(I - S_{12}\left(F^{d}\right)^{-1} S_{21} F^{i}\right)^{-1} \qquad (3.31)$$

Pour commencer faisons une approche graphique de l'équation (3.31)

S_{12} symbolise la relation causale qui peut exister entre deux éléments bond-graph de type I ou C. Le premier est en causalité intégrale et représente une composante du vecteur $\dot{X}^{i}$, quant au second il est en causalité dérivée et représente une composante du vecteur $\dot{X}^{d}$.

Dans le cadre de cette analyse (voir hypothèse 3.1) les éléments I, C et R sont 1-port, le matrices F^{i} et F^{d} sont alors diagonales. Il en est de même pour $(F^{d})^{-1}$ dont chaque composante $(F^{d})_{jj}$ est représentée par le module de l'élément I ou C en causalité dérivée (associé au $j^{ième}$ élément de $\dot{X}^{d}$).

La relation entre X^{d} et X^{i} se déduit de la table 3.1, avec l'hypothèse 3.2 (S_{24}=0 : absence de relation entre l'entrée et les éléments dépendant de stockage d'énergie).

On a $Z^{d} = S_{21} Z^{i} = S_{21} F^{i} X^{i}$, et on a également $Z^{d} = F^{d} X^{d}$.
Ce qui entraine : $X^{d} = (F^{d})^{-1} S_{21} F^{i} X^{i}$ (3.32)

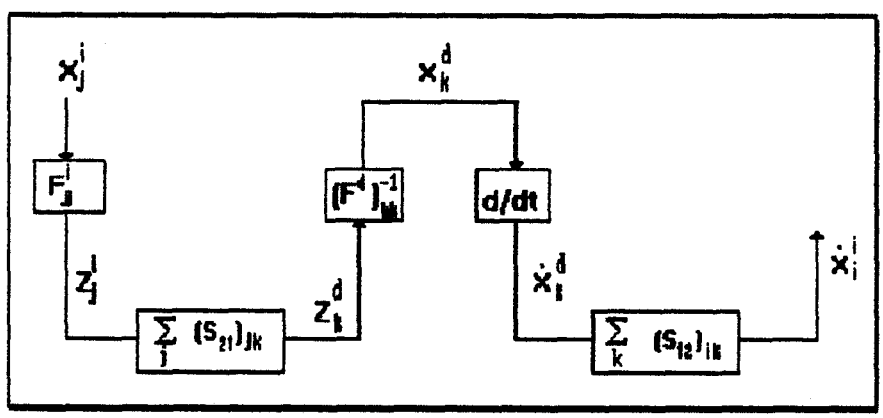


Figure 3.3.a : relation entre X_j^i et $\dot{X}_i^i$ avec $i \neq j$ en passant par X_k^d et $\dot{X}_k^d$

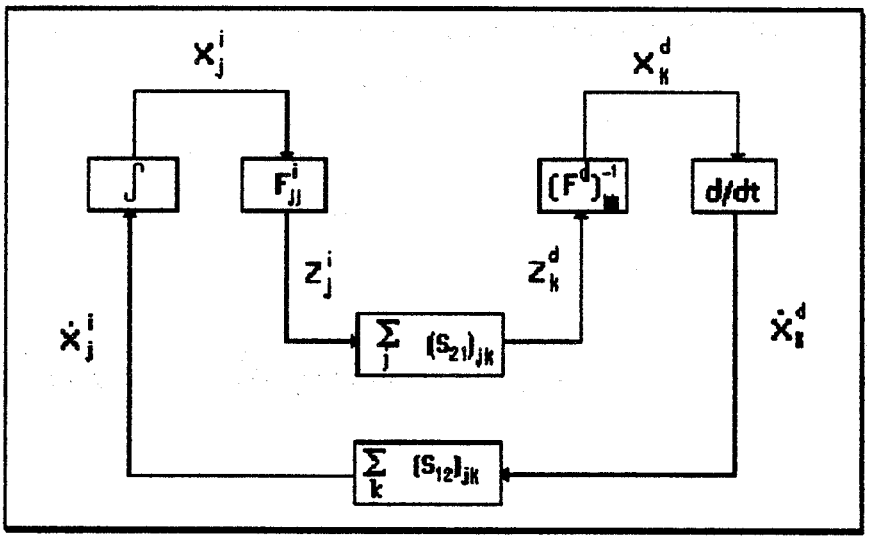


Figure 3.3.b : relation entre X^i_j et $\dot{X}_j$ en passant par X^d_k et $\dot{X}^d_k$

Les figures 3.3.a et 3.3.b schématisent, d'une manière générale, les différents cas de la relation (3.32). On a cherché à distinguer le cas où les composantes du vecteur d'état et de son vecteur dérivé ont le même indice (figure 3.3.b), de celui où les indices sont différents (figure 3.3.a).

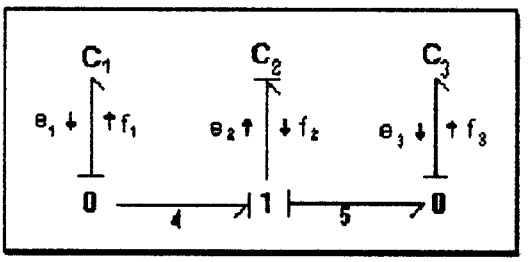


figure 3.4 : exemple d'une partie d'un modèle bond-graph possédant des élément C en causalité mixte.

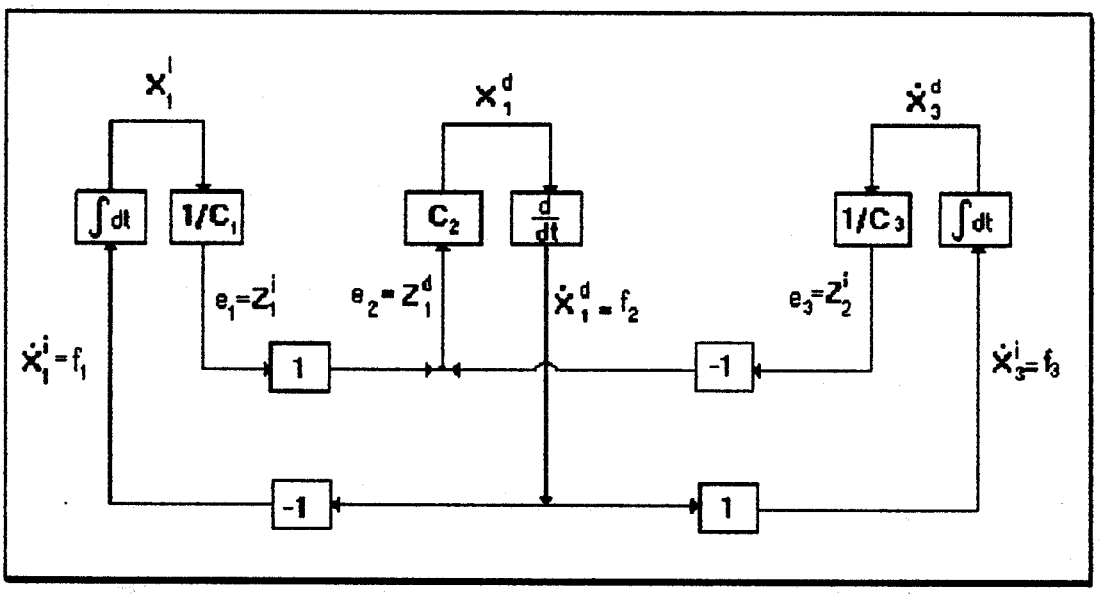


figure 3.5 : schéma reflétant les diverses relations causales de la figure3.4

La figure 3.5 quant à elle, présente une application de la relation (3.32), basée sur le modèle bond-graph de la figure 3.4. On y retrouve aussi une superposition des cas de la figure 3.4.

A partir des figures 3.3.a 3.3.b et 3.5, nous déduison les règles suivantes pour la matrice $(S_{12}\,(F^d)^{-1}\,S_{21}\,F^i)$:

Les termes diagonaux $(S_{12}\,(F^d)^{-1}\,S_{21}\,F^i)_{jj}$ correspond au gain de la boucle, formée par l'élément bond-graph (associé à X_j^i) et par un autre élément bond-graph I ou C en causalité dérivée.

Les termes diagonaux $(S_{12}\,(F^d)^{-1}\,S_{21}\,F^i)_{ij}$ représente le gain du chemin causal *indirect* liant X_j^i à $\dot{X}_i^i$ et passant par un élément I ou C en causalité dérivée.

Le tableau suivant retrace l'application de la règle précédente appliquée à la figure 3.5 qui est associée au modèle bond-graph de la figure 3.4

Valeur de i et j	valeur de $(S_{12}\,(F^d)^{-1}\,S_{21}\,F^i)_{ij}$
i = j = 1	$\frac{-C_2 s}{C_1 s} = \frac{-C_2}{C_1}$
i = j =2	$\frac{-C_2 s}{C_3 s} = \frac{-C_2}{C_3}$
i=1 ; j=2	$\frac{1}{C_3 s} \times (-1) \times C_2 s \times (-1) = \frac{C_2}{C_3}$
i=2 ; j=1	$\frac{1}{C_1 s} \times 1 \times C_2 s \times 1 = \frac{C_2}{C_1}$

Remarque :

La boucle formée, par l'élément bond-graph (associé à X_j^i) et par un autre élément bond-graph de type I ou C en causalité dérivée, peut aussi être symbolisée par un chemin causal indirect. Ce dernier va lier X_j^i à $\dot{x}_i^i$ passant par l'élément bond-graph en causalité dérivée.

La matrice de (3.31) rappelle la matrice $(sI - A)^{-1}$ de (3.14) avec la matrice $(S_{12}\,(F^d)^{-1}\,S_{21}\,F^i)$ qui se substitue à A. L'opérateur de Laplace "s" est simplifié à cause du produit de l'impédance de l'élément bond-

graph en causalité intégrale (engendre un facteur "1/s") associé à une composante de X^i par celle de l'élément bond-graph en causalité dérivée (engendre un facteur "s") associé à une composante X^d.

Utilisant de nouveau la pseudo-équivalence entre (3.14) et (3.16), la matrice de (3.31) peut être déterminée par l'application de la règle de Mason aux seules boucles entre des éléments de type I ou C en causalité inverse. On propose alors l'égalité suivante :

$$\left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1} = \frac{\left(\Gamma^d\right)}{\Delta^d} \qquad (3.33)$$

avec :

Δ^d = 1 - $\sum$ gain des boucles entre deux éléments, de type I ou C, dont l'un a une causalité intégrale et l'autre a une causalité dérivée.
+ $\sum$ produit des gains de 2 boucles disjointes de type précédent.
- $\sum$ produit des gains de 3 boucles disjointes de même type que précédemment.
+ ...

Γ^d représente une matrice carrée d'ordre **nxn** (*) dont chaque composante Γ^d_{ij} désigne un élément de la matrice *adjointe* de $(I - S_{12} [F^d]^{-1} S_{21} F^i)$. Ce dernier est un mineur d'ordre inférieur à celui de Δ^d, et se calcule de la même manière que Δ^d mais en éliminant dans ce dernier la $i^{ième}$ colonne et la $j^{ième}$ ligne. On a vu que le terme $(S_{12} [F^d]^{-1} S_{21} F^i)_{ij}$ peut être exprimé graphiquement soit par une boucle causale, soit par un chemin causal indirect.

Nous proposons finalement les règles suivantes pour calculer (Γ^d_{ij}) :

Γ^d_{ij} =déterminant de la règle de Mason calculé sur la partie du modèle bond-graph (formée à partir de relations causales entre des éléments de type I ou C de causalité différente) disjointe avec le chemin causal, passant par au moins un élément en causalité dérivée, liant X^i_j à $\dot{X}^i_j$.

(*) S_{12} : matrice d'ordre **nxd**, F^d : matrice d'ordre **dxd**, $(S_{12})^t$: matrice d'ordre **dxn** et F^i : matrice d'ordre **nxn** => $S_{12} (F^d)^{-1} S_{21} F^i$ est une matrice d'ordre **nxn**.

ou encore :

Γ^d_{ij} =Δ^d duquel on élimine tous les termes possédant une occurrence de l'élément bond-graph associé à la fois à x^i_j et à $\dot{x}^i_j$.

Γ^d_{ij} = module de l'élément bond-graph associé à X^i_j : F^i_{jj} multiplié par le gain du chemin causal, passant par un ou plusieurs éléments I ou C en causalité dérivée, liant X^i_j à $\dot{X}^i_i$.

Exemple d'application

Le but de cette exemple est de vérifier l'égalité (3.33) sur le modèle bond-graph de la figure (3.6).

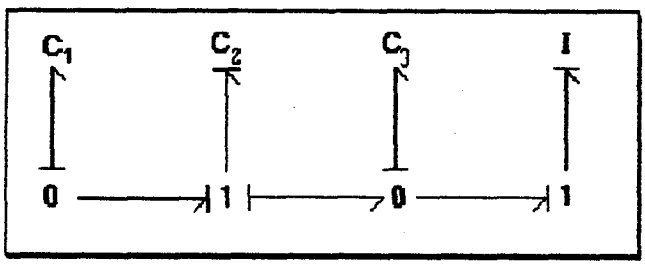


figure 3.6 : exemple d'une partie d'un modèle bond-graph possédant 3 éléments indépendants et un élément dépendant.

Exprimons les valeurs des matrices intervenant dans le premier terme de (3.33) et présentons, sans détailler les calculs, le résultat de l'expression mathématique associée à ce terme, soit :

$$S_{12} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} \quad ; \quad -S_{12}^t = S_{21} = (0 \quad 1 \quad -1) \quad ; \quad \left(F^d\right)^{-1} = C_2 \quad ; \quad F^i = \begin{pmatrix} 1/I & 0 & 0 \\ 0 & 1/C_1 & 0 \\ 0 & 0 & 1/C_2 \end{pmatrix}$$

$$\left(I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right)^{-1} = \frac{1}{1 + \frac{C_2}{C_1} + \frac{C_2}{C_3}} \begin{pmatrix} 1 + \frac{C_2}{C_1} + \frac{C_2}{C_3} & 0 & 0 \\ 0 & 1 + \frac{C_2}{C_3} & \frac{C_2}{C_3} \\ 0 & \frac{C_2}{C_1} & 1 + \frac{C_2}{C_1} \end{pmatrix}$$

Appliquons maintenant les règles explicitées précédemment pour calculer le deuxième terme de (3.33).

1°) calcul de Δ^d (se vérifie aisément).

$$\Delta^d = 1 + \frac{C_2}{C_1} + \frac{C_2}{C_3}$$

2°) détermination de la diagonale de Γ^d

Γ^d_{11} = Δ^d duquel on a enlevé les termes possédant une occurrence de I

Γ^d_{11} = Δ^d (car I n'apparait pas dans Δ^d)

Γ^d_{22}=Δ^d dans lequel on a éliminé les termes possédant une occurrence de C_1.

Soit

$$\Gamma^d_{22} = 1 + \frac{C_2}{C_3}$$

Γ^d_{33} =Δd dans lequel on a éliminé les termes possédant une occurrence de C_1

Soit

$$\Gamma^d_{33} = 1 + \frac{C_2}{C_1}$$

3°) détermination des termes hors-diagonaux :

Γ^d_{12} =gain de la liaison indirecte entre x^i_2 et $\dot{X}^i_1$. C'est aussi le gain du chemin causal, passant par C_2, de C_1 à I, multiplié par le module de C_1 ($F^i_{22} = 1/C_1$).

Soit·

Γ^d_{12} = 0, car il n'existe pas de chemin causal indirect liant C_1 à I et passant par C_2 (seul élément en causalité dérivée).

$\Gamma^d_{13} = 0$, car la liaison entre X^i_3 (représenté par C_3) et $\dot{X}^i_1$ (représenté par I) est directe.

$\Gamma^d_{21} = 0$ (même raisonnement que pour Γ^d_{13})

Γ^d_{23}=gain de la liaison indirecte entre X^i_3 et $\dot{X}^i_2$. C'est aussi le gain du chemin causal, passant par C_2, liant C_3 à C_1, multiplié par le module de C_3 ($F^i_{33} = 1/C_3$).

soit

$$\Gamma^d_{23} = \frac{C_2}{C_3}$$

$\Gamma^d_{31} = 0$ (même raisonnement que pour Γ^d_{13})

Γ^d_{32}=gain de la liaison indirecte entre X^i_2 et $\dot{X}^i_3$. C'est aussi le gain du chemin causal, passant par C_2, liant C_1 à C_3, multiplié par le module de C_1 ($F^i_{22} = 1/C_1$).

soit :

$$\Gamma^d_{32} = \frac{C_2}{C_1}$$

L'égalité a été vérifiée. Nous verrons plus loin une application de la méthode sur un modèle bond-graph plus important.

$\Gamma^d_{13} = 0$, car la liaison entre x^i_3 (représenté par C_3) et $\dot{x}^i_1$ (représenté par I) est directe.

$\Gamma^d_{21} = 0$ (même raisonnement que pour Γ^d_{13})

Γ^d_{23}=gain de la liaison indirecte entre x^i_3 et $\dot{x}^i_2$. C'est aussi le gain du chemin causal, passant par C_2, liant C_3 à C_1, multiplié par le module de C_3 ($F^i_{33} = 1/C_3$).

soit

$$\Gamma^d_{23} = \frac{C_2}{C_3}$$

$\Gamma^d_{31} = 0$ (même raisonnement que pour Γ^d_{13})

Γ^d_{32}=gain de la liaison indirecte entre x^i_2 et $\dot{x}^i_3$. C'est aussi le gain du chemin causal, passant par C_2, liant C_1 à C_3, multiplié par le module de C_1 ($F^i_{22} = 1/C_1$).

soit :

$$\Gamma^d_{32} = \frac{C_2}{C_1}$$

L'égalité a été vérifiée. Nous verrons plus loin une application de la méthode sur un modèle bond-graph plus important.

En conclusion, la matrice d'état A peut être déterminée par une méthode numérique à partir de l'expression :

$$\left[I - S_{12}\left(F^d\right)^{-1} S_{21} F^i\right]^{-1}\left[S_{11} + S_{13} L\left(I - S_{33} L\right)^{-1} S_{31}\right] F^i$$

dont nous avons retracé les limites ; mais aussi par la méthode que nous proposons ici et qui se base sur une analyse des informations causales d'un bond-graph, soit :

$$A = \frac{1}{\Delta^d \Delta^R} \Gamma^d \left[\left(\sum_k G^R_{k_A} \Delta^R_{k_A} \right)_{ij} \right] \qquad i, j = 1, n$$

avec

$$A_{ij} = \frac{1}{\Delta^d \Delta^R} \sum_{p=1}^{n} \Gamma^d_{ip} \left(\sum_k G^R_{k_A} \Delta^R_{k_A} \right)_{pj}$$

qu'on peut écrire également sous la forme :

$$A_{ij} = \frac{1}{\Delta^d \Delta^R} \sum_{p=1}^{n} \sum_k \Gamma^d_{ip} \left(G^R_{k_A} \Delta^R_{k_A} \right)_{pj}$$

où

Δ_R = 1 - Σ gain des boucles formées par deux éléments de type R
+Σ produit des gains de deux boucles disjointes entre des éléments de type R
- Σ produit des gains de trois boucles disjointes entre des éléments de type R
...

Δ^d = 1 - Σ gain des boucles entre deux éléments de type I ou C de causalité différente.
+Σ produit des gains de deux boucles disjointes de type précédent.
-Σ produit des gains de trois boucles disjointes de même type que précédemment.

$\left(G^R_{k_A}\right)_{ij}$ = gain du kième chemin causal liant x^i_j à $\dot{x}^i_i$.
ce chemin peut être direct ou indirect passant par un élément R_k.

$\Delta^R_{k_A}$ = Δ^R dans lequel on supprime les membres possédant une occurrence de l'élément R intervenant dans $G^R_{k_A}$.

Γ^d_{ij} = Δ^d dans lequel on supprime les termes ayant une occurrence de l'élément I ou C associé à x^i_j et à $\dot{x}^i_j$.

Γ^d_{ij} = (F^i_{ij} = 1/I ou 1/C) multiplié par le gain du chemin indirect passant par un ou plusieurs éléments de type I ou C en causalité dérivée et liant x^i_j à $\dot{x}^i_i$ (ou plus exactement les éléments bond-graphs qui leur sont associés).

IV2.2 Matrices d'état B, C et D

Comme on a fait pour la matrice d'état A, nous allons proposer les règles permettant de déterminer graphiquement les matrices d'état B, C, et D. Le raisonnement suivi reste le même, il va s'appuyer sur les expressions (3.17.).

matrice d'état B

1° étape : "pas d'éléments I et C en causalité dérivée"

On a :

$$B = S_{14} + S_{13} L (I - S_{33} L)^{-1} S_{34}$$

avec : S_{14} symbolisant une liaison causale directe entre U et $\dot{X}^i$

$S_{13} L (I - S_{33} L)^{-1} S_{34}$ représentant une liaison causale indirecte passant par un ou plusieurs éléments de type R, et liant deux composantes de U et de $\dot{X}^i$.

S_{34} étant la sous-matrice qui caractérise la relation causale entre U et les éléments R représentés par D_{in}.

En procédant de la même manière que pour A, on arrive à :

$$B = \frac{1}{\Delta^R} \left[\left(\sum_k G^R_{k_B} \Delta^R_{k_B} \right)_{ij} \right] \qquad i = 1, n \; ; \; j = 1, p$$

avec Δ^R et $\Delta^R_{k_B}$ définis comme pour A

$G^R_{k_B}$ =gain du k^{ième} chemin causal (direct ou indirect liant une composante U_j de U (représentée par les éléments bond-graphs S_e ou S_f) et une composante $\dot{X}^i_i$ de $\dot{X}^i$ (représentée par les éléments bond-graphs I ou C en causalité intégrale).

2°étape : "Existence d'éléments I ou C en causalité dérivée"

Avec ces hypothèses le gain engendré est le même que celui rencontré dans A, dans (3.33) et que nous avons détaillé et représenté sous la forme :

$$\left(I - S_{12}\left(F^{d}\right)^{-1} S_{21} F^{i}\right)^{-1} = \frac{\Gamma^{d}}{\Delta^{d}}$$

Nous proposons de déterminer la matrice B par application de la règle suivante :

$$B = \frac{1}{\Delta^{d}\Delta^{R}} \Gamma^{d} \left[\left(\sum_{k} G^{R}_{k_B} \Delta^{R}_{k_B} \right)_{ij} \right] \qquad i = 1, n \; ; \; j = 1, p$$

avec

$$B_{ij} = \frac{1}{\Delta^{R}\Delta^{d}} \sum_{p=1}^{R} \sum_{k} \Gamma^{d}_{ip} \left(G^{R}_{k_B} \Gamma^{R}_{k_B} \right)_{pj}$$

les définitions données précédemment pour Δ^R, Γ^d, Γ^d_{ip}, $\Gamma^R_{k_B}$ et $G^R_{k_B}$ restent inchangées.

matrice d'état C

La méthode numérique utilise, pour déterminer la matrice d'état C, l'expression suivante :

$$C = \left[S_{41} + S_{43} L \left(I - S_{33} L \right)^{-1} S_{31} \right] F^{i}$$

Cette expression matricielle rappelle celle déjà rencontrée à l'étape 1 de la détermination de A, avec le vecteur Y qui remplace le vecteur $\dot{X}^{i}$. La procédure suivie dans cette étape reste valable et nous permet (sans recommencer tout en détails) de conclure que :

$$C = \frac{1}{\Delta^{R}} \left[\left(\sum_{k} G^{R}_{k_C} \Delta^{R}_{k_C} \right)_{ij} \right] \qquad i = 1, q \; ; \; j = 1, n$$

avec

$\left(G^{R}_{k_c}\right)_{ij}$ = (F^{i}_{ij} = 1/I ou 1/C avec I ou C associé à X^{i}_{j}) multiplié par le gain du chemin causal direct ou indirect, passant par un ou plusieurs

éléments de type R, liant la composante x_j^i de x^i (représentée graphiquement par un élément bond-graph I ou C en causalité intégrale) à la composante Y_i de Y (symbolisée par un élément bond-graph I, C ou R).

Δ^R et $\Delta^R_{k_C}$ gardent les mêmes définitions vues précédemment.

matrice d'état D :

De la même manière que la matrice B (avec l'hypothèse de l'étape1) la matrice d'état D, qui se détermine numériquement par

$$D = S_{44} + S_{43} L (I - S_{33} L)^{-1} S_{34}$$

qui peut être symbolisée formellement par :

$$D = \frac{1}{\Delta^R} \left[\left(\sum_k G^R_{k_D} \Delta^R_{k_D} \right)_{ij} \right] \quad i = 1, q ; j = 1, p$$

avec

$\left(G^R_{k_D}\right)_{ij}$ = gain du chemin causal direct ou indirect liant une composante U_j de U (représentée sur un bond-graph par S_e ou S_f) à une composante Y_i de Y :
représentée sur un bond-graph par I, C ou R.

Là aussi Δ^R et $\Delta^R_{k_D}$ gardent les mêmes définitions.

V. Récapitulatif de la méthode proposée et exemple d'application

Cette section résume l'essentiel de la MBCC que nous proposons pour déterminer l'équation d'état sous forme d'une expression formelle à partir de l'analyse des liaisons causales dans un bond-graph :

Ainsi pour déterminer les matrices d'état, nous proposons pour chacune d'elles les formules suivantes :

Matrice d'état	Une composante de la matrice d'état
$A = \frac{1}{\Delta^d \Delta^R}\left(\Gamma^d\left(\sum_k G^R_{k_A}\Delta^R_{k_A}\right)_{ij}\right)$	$A_{ij} = \frac{1}{\Delta^d \Delta^R}\sum_{p=1}^{n}\Gamma^d_{ip}\left(\sum_k G^R_{k_A}\Delta^R_{k_A}\right)_{pj}$
$B = \frac{1}{\Delta^d \Delta^R}\left(\Gamma^d\left(\sum_k G^R_{k_B}\Delta^R_{k_B}\right)_{ij}\right)$	$B_{ij} = \frac{1}{\Delta^d \Delta^R}\sum_{p=1}^{n}\Gamma^d_{ip}\left(\sum_k G^R_{k_B}\Delta^R_{k_B}\right)_{pj}$
$C = \frac{1}{\Delta^R}\left[\left(\sum_k G^R_{k_C}\Delta^R_{k_C}\right)_{ij}\right]$	$C_{ij} = \frac{1}{\Delta^R}\sum_k \left(G^R_{k_C}\Delta^R_{k_C}\right)_{ij}$
$D = \frac{1}{\Delta^R}\left[\left(\sum_k G^R_{k_D}\Delta^R_{k_D}\right)_{ij}\right]$	$D_{ij} = \frac{1}{\Delta^R}\sum_k \left(G^R_{k_D}\Delta^R_{k_D}\right)_{ij}$

Table 3.3 : récapitulatif des règles pratique pour la détermination des matrices d'état.

avec

Δ^R = déterminant de la règle de Mason appliqué seulement sur la partie résistive (*) du modèle bond-graph.

$\Delta^R = 1 - \Sigma$ gain des boucles formées par deux éléments de type R.
$+\Sigma$ produit des gains de deux boucles disjointes entre des éléments de type R.
$-\Sigma$ produit des gains de trois boucles disjointes entre des éléments de type R.
etc ...

$\Delta^d = 1 - \Sigma$ gain de boucles entre deux éléments de type I ou C de causalité différente.
$+\Sigma$ produit des gains de deux boucles disjointes de type précédent.
$-\Sigma$ produit des gains de trois boucles disjointes de même type que précédemment...

* ne garder que les relation causales entre des élément R.

Γ^{d}_{ij} = Δd dans lequel on supprime les termes possédant une occurrence de l'élément I ou C associé à x^{i}_{j} et à $\dot{x}^{i}_{j}$.

Γ^{d}_{ij} = module de l'élément bond-graph affilié à x^{i}_{j} multiplié par le gain du chemin indirect passant par un ou plusieurs éléments de type I ou C en causalité dérivée et liant les éléments bond-graphs associés à x^{i}_{j} et à $\dot{x}^{i}_{i}$.

$\left[G^{R}_{k_A}\right]_{ij}$ = module de l'élément bond-graph affilié à x^{i}_{j} multiplié par le gain du $k^{ième}$ chemin causal liant les deux éléments bond-graphs de type I ou C, en causalité intégrale, associés respectivement à x^{i}_{j} et à $\dot{x}^{i}_{i}$.
Ce chemin peut être direct ou indirect passant par un élément R.

$\left[G^{R}_{k_B}\right]_{ij}$ = gain du $k^{ième}$ chemin causal (direct ou indirect liant une composante U_j de U (représentée par les éléments bond-graphs S_e ou S_f) à une composante $\dot{x}^{i}_{i}$ de $\dot{x}^{i}$ (représentée par les éléments bond-graphs I ou C en causalité intégrale).

$\left[G^{R}_{k_C}\right]_{ij}$ =module de l'élément bond-graph affilié à x^{i}_{j} multiplié par le gain du chemin causal direct ou indirect, passant par un ou plusieurs éléments de type R, liant la composante x^{i}_{j} de x^{i} à la composante Y_i de Y (symbolisée par capteur).

$\left[G^{R}_{k_D}\right]_{ij}$ = gain du chemin causal direct ou indirect liant une composante U_j de U (représentée sur un bond-graph par S_e ou S_f) à une composante Y_i de Y .

$\Delta^{R}_{k_M}$ = Δ^{R} calculé sur la partie résistive disjointe avec le chemin causale représenté par $G^{R}_{k_M}$, avec M= A, B, C, ou D.

Exemple d'application

L'exemple de la figure (3.7) présente diverses situations susceptibles d'exister dans un modèle bond-graph (différents types de chemins causaux et de boucles causales, existence de boucles algébriques etc...).

Nous indiquons que ce bond-graph ne possède pas forcément un sens physique et qu'il a été choisi pour montrer une application de notre méthode et pour rendre compte de ses points positifs (utilisation manuelle pratique et rapide etc...).

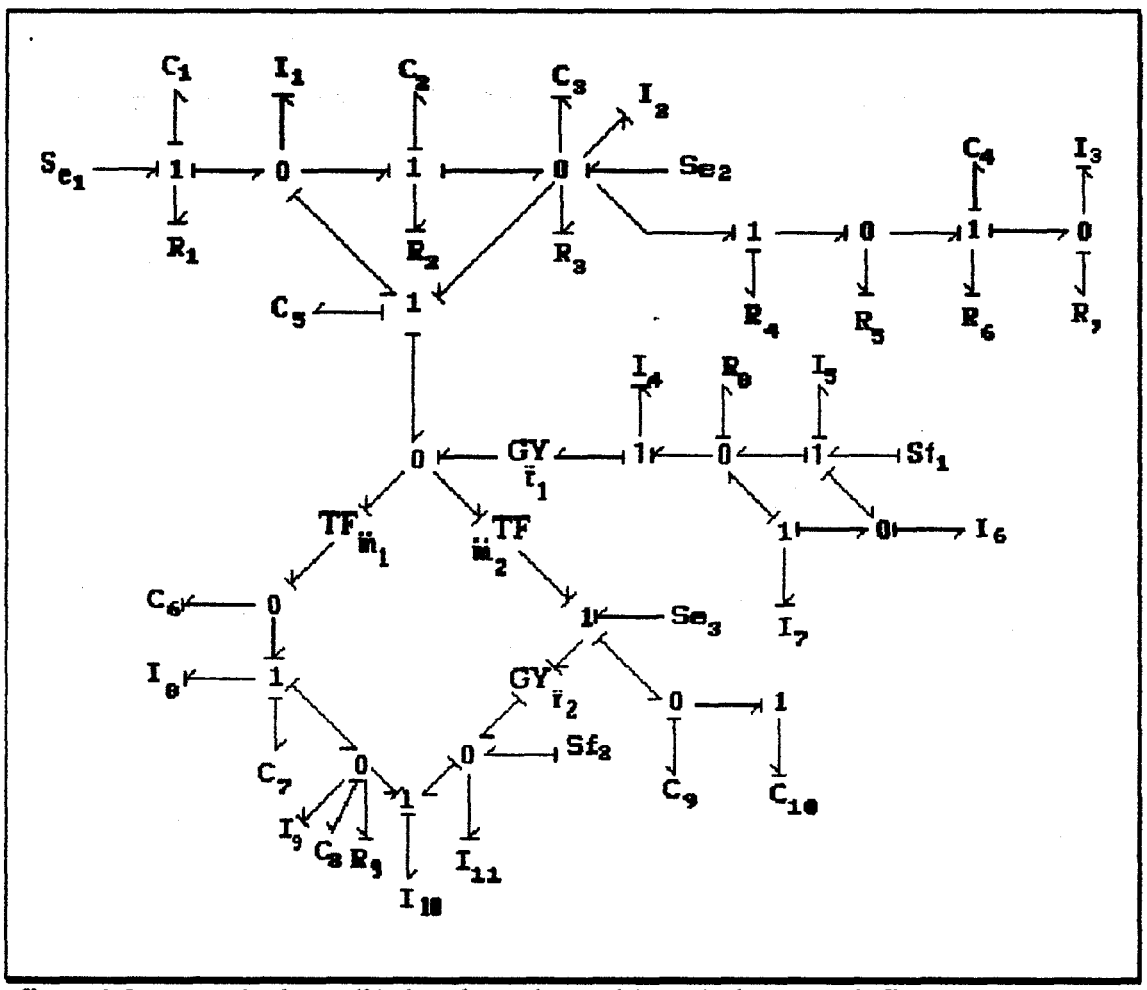


figure 3.7 : exemple de modèle bond-graph possédant plusieurs cas de figure permettant de montrer l'intérêt de l'utilisation de la MBCC

Nous allons nous intéresser seulement à l'application de la MBCC pour la détermination de la matrice d'état A qui représente le cas le plus général. Il suffit d'adapter le même raisonnement en utilisant les règles de la table 3.3 pour retrouver les matrices B, C, et D.

Commençons par extraire quelques informations du modèle bond-graph de la figure 3.7.

Eléments bond-graphs	Type des éléments	Module	Impédance	Dimension des vecteurs
Se_1 , Se_2 , Se_3 Sf_1 , Sf_2	Sources d'effort Sources de flux	/ /	/ /	dim(U)=5
R_1,R_2,R_3, R_5, R_6, R_9	éléments dissipatifs	$1/R_k$ K=1, 2, 3, 5, 6, 9	$1/R_k$	$dim(D_{out}) = dim(D_{in}) =10$
R_4, R_7, R_8	,,	R_h h=1, 2, 3	R_h	
I_1 , I_2 , I_3 , I_4 , I_7 , I_8 , I_9 , I_{11}	éléments de stockage d'énergie (indépendants)	$1/I_i$ i=1, 2, 3, 4, 7, 8, 9, 11	$1/I_i s$	$dim(X^i)$ = 15
C_1 , C_2 ,C_4 , C_5 C_7 , C_8 , C_9	,,	$1/C_j$ j=1, 2, 4, 5,7, 8, 9,	$1/C_j s$	
I_5 , I_6 , I_{10}	éléments de stockage d'énergie (dépendants)	I_i i=5, 6, 10	$I_i s$	$dim(X^d)$ = 6
C_3 , C_6 , C_{11}	,,	C_j j=3, 6, 11	$C_j s$	

Ce tableau donne une idée sur la taille des différentes matrices qui interviennent dans les expressions (3.17.a), (3.17.b), (3.17.c) et (3.17.d), pour calculer les matrices d'état. Et à part les produits matriciels, il faut inverser une matrice d'ordre 9 x 9 et une autre d'ordre 15 x 15 à savoir respectivement $(I - S_{33}L)$ et $(I - S_{12}(F^d)^{-1}S_{21}F^i)$.

Ajoutons que ces matrices ont en moyenne 70% de composantes nulles. Toutes ces difficultés vont rendre la tâche de détermination à partir des expressions précitées impossible manuellement et extrêment délicate et excessivement longue sur un calculateur (même sur un grand système).

Parmi les intérêts d'une méthode graphique on trouve l'économie des traitements inutiles, car on ne prend en compte que des chemins existant c'est-à-dire qu'aux termes non nuls de la matrice de structure

1) Calcul de Δ^R

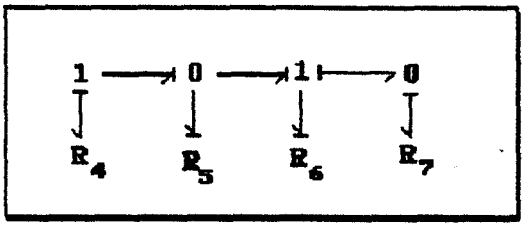


figure 3. 8: partie résistive du modèle bond-graph de la figure (3.7)

Constatons que les éléments dissipatifs R_1, R_2, R_3, R_8 et R_9 ne sont pas directement causalement liés à un autre élément déssipatif. Ils ne font donc pas intégrés à la partie résitive. Cette dernière peut aussi être formée par des sous-parties résistives disjointes.

A partir de la figure 3.8 nous déduisons pour Δ^R la valeur suivante :

$$\Delta^R = 1 + R_4 / R_5 + R_4 / R_6 + R_7 / R_6 + R_4 R_7 / R_5 R_6$$

On peut remarquer qu'on arrive plus rapidement et sans opération inutile par cette méthode que par le calcul du déterminant d'une matrice d'ordre 9x9.

2) Calcul de Δ^d

Le tableau suivant présente toutes les boucles, et leur gain, formées par un élément en causalité dérivée et un autre en causalité intégrale.

Relation causale entre deux éléments de stockage de causalités inverses	Gain de la boucle correspondante à cette relation
I_4 et I_{10}	$- I_{10} / I_4$
I_4 et C_6	$- C_6 / I_4$
I_7 et I_6	$- I_6 / I_7$
C_7 et C_6	$- C_6 / C_7$
I_{11} et I_{10}	$- I_{10} / I_{11}$
C_9 et I_{10}	$- I_{10} / C_9$
C_9 et C_{10}	$- C_{10} / C_9$

Le tableau qui va suivre donne toutes les informations sur les boucles disjointes formées à partir des boucles du tableau précédent.

	Boucles causales disjointes formées à partir des boucles du tableau précédent	Gain correspondant
Boucles disjointes d'ordre 2	(I_4 , I_{10}) avec (I_7 , I_6)	$[I_{10} / I_4] \times [I_6 / I_7]$
	(I_4 , I_{10}) avec (C_7 , C_6)	$[I_{10} / I_4] \times [C_6 / C_7]$
	(I_4 , I_{10}) avec (C_9 , C_{10})	$[I_{10} / I_4] \times [C_{10} / C_9]$
	(I_4 , C_6) avec (I_7 , I_6)	$[C_6 / I_4] \times [I_6 / I_7]$
	(I_4 , C_6) avec (C_9 , C_{10})	$[C_6 / I_4] \times [C_{10} / C_9]$
	(I_7 , I_6) avec (C_9 , C_{10})	$[I_6 / I_7] \times [C_{10} / C_9]$
	(I_7 , I_6) avec (I_{11} , I_{10})	$[I_6 / I_7] \times [I_{10} / I_{11}]$
	(I_7 , I_6) avec (C_7 , C_6)	$[I_6 / I_7] \times [C_6 / C_7]$
	(I_{11} , I_{10}) avec (C_7 , C_6)	$[I_{10} / I_{11}] \times [C_6 / C_7]$
	(I_{11} , I_{10}) avec (C_9 , C_{10})	$[I_{10} / I_{11}] \times [C_{10} / C_9]$
	(C_7 , C_6) avec (C_9 , C_{10})	$[C_6 / C_7] \times [C_{10} / C_9]$
Boucles disjointes d'ordre 3	(I_4 , I_{10}) ; (I_7 , I_6) ; (C_7 , C_6)	$[I_{10} / I_4] \times [I_6 / I_7] \times [C_6 / C_7]$
	(I_4 , I_{10}) ; (I_7 , I_6) ; (C_9 , C_{10})	$[I_{10} / I_4] \times [I_6 / I_7] \times [C_{10} / C_9]$
	(I_4 , I_{10}) ; (I_{11} , I_{10}) ; (C_9 , C_{10})	$[I_{10} / I_4] \times [I_{10} / I_{11}] \times [C_{10} / C_9]$
	(I_4 , C_6) ; (I_7 , I_6) ; (I_{11} , I_{10})	$[C_6 / I_4] \times [I_6 / I_7] \times [I_{10} / I_{11}]$
	(I_4 , C_6) ; (I_7 , I_6) ; (C_9 , C_{10})	$[C_6 / I_4] \times [I_6 / I_7] \times [C_{10} / C_9]$
	(I_4 , C_6) ; (I_{11} , I_{10}) ; (C_9 , C_{10})	$[C_6 / I_4] \times [C_6 / C_7] \times [C_{10} / C_9]$
	(I_{11} , I_{10}) ; (I_7 , I_6) ; (C_7 , C_6)	$[I_{10} / I_{11}] \times [I_6 / I_7] \times [C_6 / C_7]$
	(I_{11} , I_{10}) ; (I_7 , I_6) ; (C_9 , C_{10})	$[I_{10} / I_{11}] \times [I_6 / I_7] \times [C_{10} / C_9]$
	(I_{11} , I_{10}) ; (C_7 , C_6) ; (C_9 , C_{10})	$[I_{10} / I_{11}] \times [C_6 / C_7] \times [C_{10} / C_9]$
	(I_7 , I_6) ; (C_7 , C_6) ; (C_9 , C_{10})	$[I_7 / I_6] \times [C_6 / C_7] \times [C_{10} / C_9]$
Boucles disjointes d'ordre 4	(I_4 , I_{10}) ; (I_7 , I_6) ; (C_7 , C_6) ; (C_9 , C_{10})	$[I_{10} / I_4] \times [I_6 / I_7] \times [C_6 / C_7] \times [C_{10} / C_9]$
	(I_4 , C_6) ; (I_7 , I_6) ; (I_{11} , I_{10}) ; (C_9 , C_{10})	$[C_6 / I_4] \times [I_6 / I_7] \times [I_{10} / I_{11}] \times [C_{10} / C_9]$
	(I_{11} , I_{10}) ; (I_7 , I_6) ; (C_7 , C_6) ; (C_9 , C_{10})	$[I_{10} / I_{11}] \times [I_6 / I_7] \times [C_6 / C_7] \times [C_{10} / C_9]$

Calcul de Δ^d

$$\Delta^d = 1 + I_{10}\left(\frac{1}{I_4} + \frac{1}{I_{11}} + \frac{1}{C_9}\right) + C_6\left(\frac{1}{I_4} + \frac{1}{C_7}\right) + \frac{I_6}{I_7} + \frac{C_{10}}{C_9}$$

$$+ \frac{I_{10}}{I_4}\left(\frac{I_6}{I_7} + \frac{C_6}{C_7} + \frac{C_{10}}{C_9}\right) + \frac{I_6}{I_7}\left(\frac{C_6}{I_4} + \frac{C_{10}}{C_9} + \frac{I_{10}}{I_{11}} + \frac{C_6}{C_7}\right)$$

$$+ \frac{C_{10}}{C_9}\left(\frac{C_6}{I_4} + \frac{I_{10}}{I_{11}} + \frac{C_6}{C_7}\right) + \frac{I_{10}}{I_{11}}\frac{C_6}{C_7} + \frac{I_{10}}{I_4}\frac{I_6}{I_7}\left(\frac{C_6}{C_7} + \frac{C_{10}}{C_9}\right)$$

$$+ \frac{C_6}{I_4}\frac{I_6}{I_7}\left(\frac{I_{10}}{I_4} + \frac{I_{10}}{I_{11}}\right) + \frac{I_{10}}{I_4}\frac{I_6}{I_7}\left(\frac{C_6}{C_7} + \frac{C_{10}}{C_9}\right)$$

$$+ \frac{I_6}{I_7}\frac{C_{10}}{C_9}\left(\frac{I_{10}}{I_4} + \frac{C_6}{C_7}\right) + \frac{C_6}{C_7}\frac{C_{10}}{C_9}\left(\frac{I_{10}}{I_{11}} + \frac{C_6}{I_4}\right)$$

$$+ \frac{I_6}{I_7}\frac{C_{10}}{C_9}\left(\frac{C_6}{C_7}\left(\frac{I_{10}}{I_4} + \frac{I_{10}}{I_{11}}\right) + \frac{C_6}{I_4}\frac{I_{10}}{I_{11}}\right)$$

Calcul des $\Gamma_{i,j}$

$\Gamma_{1,1} = \Gamma_{2,2} = \Gamma_{3,3} = \Gamma_{6,6} = \Gamma_{7,7} = \Gamma_{9,9} = \Gamma_{10,10} = \Gamma_{11,11} = \Gamma_{12,12} = \Gamma_{14,14} = \Delta^d$ car les représentants de X_i et de $\mathring{X}_i$ avec i = 1,2,3,6,7,9,10,11,12,14 n'apparaissant pas dans Δ^d.

Quant aux autres on a par exemple, pour $\Gamma_{4,4}$ dont I_4 représenté par X_4 et $\mathring{X}_4$ à la valeur suivante :

$$\Gamma_{4,4} = 1 + I_{10}\left(\frac{1}{I_{11}} + \frac{1}{C_9}\right) + \frac{C_6}{C_7} + \frac{I_6}{I_7} + \frac{C_{10}}{C_9} + \frac{I_6}{I_7}\left(\frac{C_{10}}{C_9} + \frac{I_{10}}{I_{11}} + \frac{C_6}{C_7}\right) +$$

$$\frac{C_{10}}{C_9}\left(\frac{I_{10}}{I_{11}} + \frac{C_6}{C_7}\right) + \frac{I_{10}}{I_{11}}\frac{C_6}{C_7}\frac{I_{10}}{I_{11}}\frac{I_6}{I_7}\left(\frac{C_6}{C_7} + \frac{C_{10}}{C_9}\right) +$$

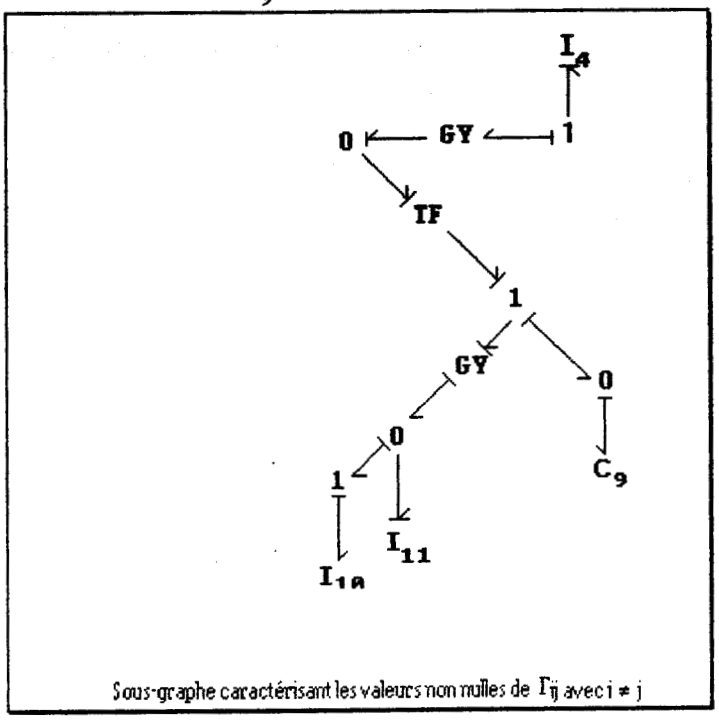
$$\frac{C_6}{C_7}\frac{C_{10}}{C_9}\left(\frac{I_6}{I_7} + \frac{I_{10}}{I_{11}} + \frac{C_6}{I_4}\right) + \frac{I_6}{I_7}\frac{C_{10}}{C_9}\frac{C_6}{C_7}\frac{I_{10}}{I_{11}}$$

$\Gamma_{5,5}$ (dont X_5 et $\mathring{X}_5$ sont représentés par I_7) se détermine comme $\Gamma_{4,4}$ en éliminant de Δ^d les occurences de I_7.

$\Gamma_{8,8}$ (dont X_8 et $\dot{X}_8$ sont représentés par I_8) se détermine comme $\Gamma_{4,4}$ en éliminant de Δ^d les occurences de I_8 .

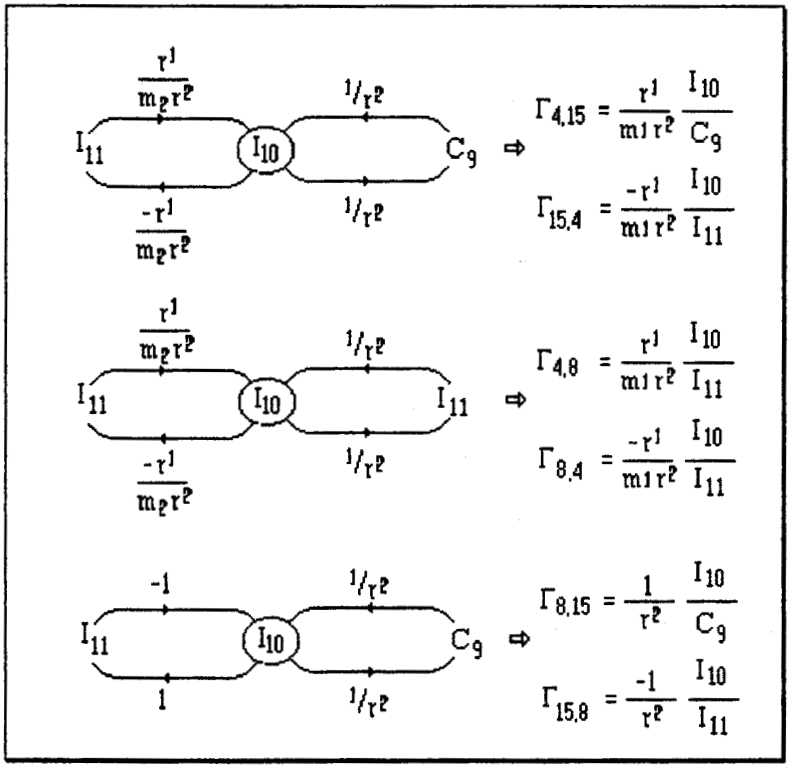
$\Gamma_{13,13}$ (dont X_{13} et $\dot{X}_{13}$ sont représentés par C_7) se détermine comme $\Gamma_{4,4}$ en éliminant de Δ^d les occurences de C_7 .

$\Gamma_{15,15}$ (dont X_{15} et $\dot{X}_{15}$ sont représentés par C_9) se détermine comme $\Gamma_{4,4}$ en éliminant de Δ^d les occurences de C_9 .



Sous-graphe caractérisant les valeurs non nulles de Γ_{ij} avec $i \neq j$

Les valeurs non nulles de Γ_{ij} correspondent à :



$$\Gamma_{4,15} = \frac{r^1}{m_1 r^2}\frac{I_{10}}{C_9} \qquad \Gamma_{15,4} = \frac{-r^1}{m_1 r^2}\frac{I_{10}}{I_{11}}$$

$$\Gamma_{4,8} = \frac{r^1}{m_1 r^2}\frac{I_{10}}{I_{11}} \qquad \Gamma_{8,4} = \frac{-r^1}{m_1 r^2}\frac{I_{10}}{I_{11}}$$

$$\Gamma_{8,15} = \frac{1}{r^2}\frac{I_{10}}{C_9} \qquad \Gamma_{15,8} = \frac{-1}{r^2}\frac{I_{10}}{I_{11}}$$

Tous les autres Γ_{ij} avec $i \neq j$ sont nuls soit parceque X_j est lié directement à $\dot{X}_i$ soit parcequ'il n'existe pas de chemin indirect liant leurs deux représentants passant par un élément en causalité dérivée.

Par exemple $\Gamma_{12} = 0$ car pas de chemin indirect passant par un élément en causalité dérivée entre I_2 et I_1.
$\Gamma_{46} = 0$ car chemin direct entre I_4 et I_8.

Détermintion des composantes des matrices d'état

Pour calculer G^{R}_{ij} et Δ^{R}_{ij} nous allons nous restreindre au calcul de quelques composantes de A. Pour les autres la démarche est la même, elle se base sur l'application directe de la table 3.3.

$$- a_{11} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{1h} \sum_{k}^{Nbc} (G^{R}_{kA} \Delta^{R}_{kA})_{h1}$$

seul Γ_{11} est différent de 0 d'où : $a_{11} = \frac{1}{\Delta^R} \sum_{k}^{Nbc} (G^{R}_{kA} \Delta^{R}_{kA})_{11}$

Nbc = 0 car il n'existe pas de chemin liant X_1 à $\dot{X}_1$ passant par des R, donc $a_{11} = 0$.

$$- a_{33} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{3h} \sum_{k}^{Nbc} (G^{R}_{kA} \Delta^{R}_{kA})_{h3}$$

seul Γ_{33} est différent de 0 : $a_{33} = \frac{1}{\Delta^d} \sum_{k}^{Nbc} (G^{R}_{kA} \Delta^{R}_{kA})_{33}$

Nbc = 1 car un seul chemin entre X_3 à $\dot{X}_3$ donc :

$$(G^{R}_{1A})_{33} = - \frac{R_7}{I_3} \qquad (\Delta^{R}_{1A})_{33} = 1 + \frac{R_4}{R_5} + \frac{R_4}{R_6}$$

$$- a_{41} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{4h} \sum_{k}^{Nbc} (G^{R}_{kA} \Delta^{R}_{kA})_{h1}$$

seul $\Gamma_{4,4}$, $\Gamma_{4,8}$, $\Gamma_{4,15}$ sont différents de 0, donc :

$$a_{41} = \frac{1}{\Delta^d \Delta^R} \left(\Gamma_{4,4} \sum_{k}^{Nbc1} (G_{kA} \Delta_{kA})_{41} + \Gamma_{4,8} \sum_{k}^{Nbc2} (G_{kA} \Delta_{kA})_{8} \right.$$

$$+ \quad \Gamma_{4,15} \sum_{k}^{Nbc3} (G_{kA} \ \Delta_{kA})_{15}$$

Les valeurs de $\Gamma_{4,4}$, $\Gamma_{4,8}$, $\Gamma_{4,15}$ on les a déjà calculé précédemment.

$Nbc1 = 1 \quad G_{1A} = \frac{1}{r1 \ I_1} \quad \Delta_{1\,A=} \ \Delta^R$

$$- a_{41} = \frac{1}{\Delta^d \ \Delta^R} \sum_{h=1}^{15} \Gamma^{\,d}_{4h} \sum_{k}^{Nbc} (G^{\,R}_{kA} \ \Delta^{\,R}_{kA})_{hl}$$

seul $\Gamma_{4,4}$, $\Gamma_{4,8}$, $\Gamma_{4,15}$ sont différents de 0

$$- a_{41} = \frac{1}{\Delta^d \ \Delta^R} \ \Gamma_{4,4} \sum_{k}^{Nbc1} (G_{kA} \ \Delta_{kA})_{41} + \Gamma_{4,8} \sum_{k}^{Nbc2} (G_{kA} \ \Delta_{kA})_{81} + \Gamma_{4,15}$$

$$\sum_{k}^{Nbc3} (G_{kA} \ \Delta_{kA})_{151}$$

Les valeurs de $\Gamma_{4,4}$, $\Gamma_{4,8}$, $\Gamma_{4,15}$ on les a déjà calculé précédemment.

$Nbc1 = 1 \quad G_{k1} = \frac{1}{r1 \ I_1} \quad \Delta_{k1} = \Delta^R$

Nbc2 = 0 car pas de chemin causal direct ou indirect passant par des éléments R liant I_1 à I_{11}.

Nbc3 = 0 car pas de chemin causal direct ou indirect passant par des éléments R liant I_1 à C_9.

$a_{41} = \frac{1}{\Delta^d} \ \Gamma_{4,4} \ \frac{-1}{r1 \ I_1}$

pour Δ^d et $\Gamma_{4,4}$ voir précédemment

De la même manière on détermine les autres termes.

Pour la matrice B nous calculons les valeurs formelles de quelques unes de ces composantes.

$$- b_{11} = \frac{1}{\Delta^d \ \Delta^R} \sum_{h=1}^{15} \Gamma^{\,d}_{1h} \sum_{k}^{Nbc} (G^{\,R}_{kB} \ \Delta^{\,R}_{kB})_{hl}$$

Comme on a vu plus haut seul Γ_{11} est différent de 0

$$- b_{11} = \frac{\sum^{Nbc} (G^{\,R}_{kB} \ \Delta^{\,R}_{kB})_{hl}}{\Delta^R}$$

Nbc = 0 car pas de chemin direct ou passant par un élément R liant S_{e1} à I_1

$$- b_{11} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{1h} \sum_{k}^{Nbc} (G^{R}_{kB} \Delta^{R}_{kB})_{h1}$$

$$- b_{41} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{4h} \sum_{k}^{Nbc} (G^{R}_{kB} \Delta^{R}_{kB})_{h1}$$

soit

$$- b_{41} = \frac{1}{\Delta^d \Delta^R} \Gamma_{4,4} \sum_{k}^{Nbc1} (G_{kB} \Delta_{kB})_{41} + \Gamma_{4,8} \sum_{k}^{Nbc2} (G_{kB} \Delta_{kB})_{81} + \Gamma_{4,15} \sum_{k}^{Nbc3} (G_{kB} \Delta_{kB})_{151}$$

$Se_1 \longrightarrow R_1 \longrightarrow GY_1 \longrightarrow I_1 \Rightarrow$

$$G^{R}_{kB} = \frac{1}{\Delta^d \Delta^R} \text{ -> } \Delta^{R}_{kB} = \Delta^R$$

Nbc2 = Nbc3 = 0 pas de chemin direct ou passant par un R liant S_{e_1} à I_4 et à C_9

$$S_{22} = \frac{1}{\Delta^d \Delta^R} \sum_{h=1}^{15} \Gamma^{d}_{2h} \sum_{k}^{Nbc} (G_{kB} \Delta_{kB})_{h2}$$

Seul Γ_{22} est différent de 0 d'où

$$b_{22} = \frac{1}{\Delta^d \Delta^R} \Gamma^{d}_{2,2} \sum_{k}^{Nbc} (G_{kB} \Delta_{kB})_{22}$$

Nbc = 1 chemin direct entre Se_2 et I_2 d'où $G_{1B} = 1 \Rightarrow \Delta^R = \Delta$

et comme $\Gamma_{22} = \Delta^d$ on a $b_{22} = 1$

De la même manière on détermine les autres composantes de la matrice B. Quant aux matrices C et D connaissant les variables associées aux sorties on applique comme pour A et pour B les règles de la table 3.3.

VI. Etude de la détermination de l'équation d'état d'un système non linéaire

Dans cette section nous allons présenter une étude générale de l'équation d'état d'un système non linéaire obtenue à partir d'un bond-graph. Cette étude représente une extension de la méthode de Rosenberg. C'est un développement assez général, qui se base, tout au moins au début, sur une formulation de l'équation (3.1) dans le cas non linéaire représentant un bloc diagramme de la figure 3.1. Cependant on ne peut pas expliciter ni paramètriquement, ni mathématiquement (expressions mathématiques du type 3.17) l'équation d'état. On reste dans un formalisme général du même type que celui exposé dans la section I. Une méthode numérique s'avère alors plus que nécessaire.

Toutefois nous essayerons de proposer dans certaines classes de système non linéaire, une méthode représentant une forme générale de l'équation d'état paramètrée par des fonctions Φ non linéaires liées aux éléments 1-port (I, C et R) et aux modules des éléments 2-ports (TF et GY).

Ces données permetteront à tout logiciel de simulation d'effectuer sans difficultés les différents calculs connaissant la nature des fonctions Φ.

La table 3.2 résume les définitions des éléments I et C dans le cas non linéaire.

Type de causalité	Relation E/ S des variables au port		Relation	Relations entre les vecteurs
R ⟵	$D_{in} = f$	$D_{out} = e$	$e = \Phi_R(f)$ $f = [\Phi_R]^{-1}(e)$	$D_{out} = \Phi_L(D_{in})$
R ⟵	$D_{in} = e$	$D_{out} = f$		
C ⟵	$X^i = f$	$Z^i = e$	$Z^i = [\Phi_C]^{-1}(X^i)$ $X^d = \Phi_C(Z^d)$	$Z^i = \Phi_{Fi}(X^i)$
C ⟵	$Z^d = e$	$X^d = f$		
I ⟵	$X^i = e$	$Z^i = f$	$Z^i = [\Phi_I]^{-1}(X^i)$ $X^d = \Phi_I(Z^d)$	$X^d = \Phi_{Fd}(Z^d)$
I ⟵	$Z^d = f$	$X^d = e$		

Table 3.4 : relations entre les variables d'entrées - sorties des élément 1-port dont le module est non linéaire

VI. 1. Systèmes possédant une équation de type $\dot{X} = \Phi(X, U)$
ROSENBERG [1971]

hypothèse VI.1.1 : pas d'éléments en causalité dérivée

hypothèse VI.1.2 : on ne s'intéresse qu'aux éléments 1-ports (I, C et R) et aux éléments 2-port(TF et GY).

hypothèse VI.1.3 : L'équation représentant le bloc-diagramme de la figure 3.1 s'écrit comme suit

$$\begin{cases} \dot{X} = S_{11}(.)\, Z + S_{13}(.)\, D_{out} + S_{14}(.)\, U & (3.34) \\ D_{in} = S_{31}(.)\, Z + S_{33}(.)\, D_{out} + S_{34}(.)\, U & (3.35) \end{cases}$$

avec

$$D_{out} = \Phi_L(D_{in}) \qquad (3.36)$$
$$Z = \Phi_F(X) \qquad (3.37)$$

$S_{ij}(.)$ dépendant de X, du temps ou de paramètres extérieurs.

En remplaçant D_{out} dans (3.34) par sa valeur de (3.36) on arrive à :

$$D_{in} - S_{33}(\cdot)\Phi_L(D_{in}) = S_{31}(\cdot)\Phi_F(\cdot) + S_{34}(\cdot)U \qquad (3.38)$$

Lorsque l'équation (3.38) peut s'écrire

$$D_{in} = \Phi_D(X, U) \qquad (3.39)$$

Alors l'équation (3.36) s'exprime sous la forme :

$$D_{out} = \Phi_L[\Phi_D(X, U)] \qquad (3.40)$$

Et en insérant (3.37), (3.40) dans (3.34) on arrive à :

$$\dot{X} = S_{11}(\cdot)\Phi_F(X) + S_{13}(\cdot)\Phi_L[\Phi_D(X, U)] + S_{14}(\cdot)U \qquad (3.41)$$

VI.2. Systèmes possédant une équation du type $\Phi(\dot{X}, X, U) = 0$

ROSENBERG [1971]

Les hypothèses VI.1.2 et VI.1.3 restent valables ici, et on va traiter le cas de l'existence d'une causalité mixte (intégrale et dérivée), c'est-à-dire le système possède les variables de stockage d'énergie indépendantes et dépendantes.

Ainsi le vecteur d'énergie se représente par : $X = \begin{bmatrix} X^i \\ X^d \end{bmatrix}$

et celui de co-énergie par : $Z = \begin{bmatrix} Z^i \\ Z^d \end{bmatrix}$

Les relations non linéaires entre les composantes de ces vecteurs sont :

$$Z^i = \Phi_{F^i}(X^i) \qquad (3.42)$$

$$X^d = \Phi_{F^d}(Z^d) \qquad (3.43)$$

On rappelle que l'équation (3.36) reste valable dans ce cas.

Et l'équation représentant dans le cas non linéaire le bloc-diagramme de la figure 3.1 s'exprime par :

$$\begin{bmatrix} \dot{X}^i \\ Z^d \\ D_{in} \\ Y \end{bmatrix} = \begin{bmatrix} S_{11}(.) & S_{12}(.) & S_{13}(.) & S_{14}(.) \\ S_{22}(.) & 0 & 0 & S_{24}(.) \\ S_{31}(.) & 0 & S_{33}(.) & S_{34}(.) \\ S_{41}(.) & 0 & S_{43}(.) & S_{44}(.) \end{bmatrix} \begin{bmatrix} Z^i \\ \dot{X}^d \\ D_{out} \\ U \end{bmatrix} \qquad (3.44)$$

Notons qu'on s'est placé dans le cas déjà débattu dans III.1.2, c'est-à-dire $S_{23} = S_{32} = 0$.

Il arrive qu'à ce niveau on ne puisse pas expliciter clairement l'équation donnant $\dot{X}$ en fonction du vecteur d'état X et de commande U.

Seules des manipulations numériques sous forme algébro- différentielles de matrices peuvent aboutir à une équation d'état. On ne retrace pas ici

les différentes procédures, rencontrées dans la section II, qui proposent implicitement dans un formalisme très général et abstrait l'équation d'état à partir de l'équation (3.44).

VI.3 Discussions autour de la méthode de ROSENBERG dans le cas d'un système non linéaire

Les expressions mathématiques, explicitées précédemment, permettent d'aboutir à une équation d'état représentant un système non linéaire par le biais de méthodes numériques. Toute tentative pour atteindre le même but à partir d'une méthode graphique (ou manuelle) se révèle très complexe voire même impossible dans le cas d'un bond-graph totalement non linéaire (élément et structure non linéaires). Nous pensons que la détermination de l'équation d'état d'un système non linéaire, ne peut, dans le cas de l'existence dans le modèle bond-graph d'une causalité mixte, être réalisée à partir d'une procédure automatique. Et l'utilisation des programmes de résolutions numériques, reste pour le moment la seule et unique solution à quelques exceptions près pour certaines classes de systèmes non linéaires.

VI.4. Extension de la MBCC à certaines classes de non linéarité

Nous étudierons dans ce qui va suivre les différentes classes possibles de systèmes non linéaires susceptibles d'exister dans un modèle bond-graph. Nous indiquerons pour certaines d'entres elles une méthode pratique et pour les autres nous concevrons des expressions mathématiques s'appuyant le plus possible sur une analyse graphique.

Les hypothèses VI.2.2 et VI.2.3 restent valables là aussi. Nous pensons qu'une généralisation de l'hypothèse restrictive VI.2.2, se fera sans difficulté. Il suffit de considérer les composantes d'un éléments n-port (champs : I, C, et R) comme des éléments à part entière.

VI.4.1 Cas où seule la structure des jonctions est non linéaire

On rappelle que cette non linéarité est caractérisée par les jonctions non linéaires représentant le module des jonctions TF et GY. Cette classe ne

représente pas de difficulté majeure quant à la détermination de l'équation d'état. Et la méthode que nous avons proposée dans le cadre linéaire, s'applique littéralement.
Passons directement au cas le plus général et représenté par l'équation (3.44) associée au schéma Bloc-diagramme de la figure (3.1) qu'on peut réécrire d'une manière simplifiée :

$$\left(\dot{X}, Z^d, D_{in}, Y\right)^t = S(\cdot)\left(Z^i, \dot{X}^d, D_{out}, U\right)^t \qquad (3.45)$$

avec $S(.)$ matrice de structure dont les éléments sont des fonctions non linéaires du vecteur d'état. (*)

Les relations entre les différents vecteurs mise en jeu dans (3.45) sont celles d'un système linéaire, à savoir :

$$Z^i = F^i X^i \qquad Z^d = F^d X^d \qquad D_{out} = L D_{in} \qquad (3.46)$$

Les transformations algébriques (**) formulées à partir de (3.44) et (3.46) aboutissent à l'équation d'état suivante :

$$\begin{cases} \dot{X}^i = A(\cdot) + B(\cdot)\, U + E(\cdot)\, \dot{U} \\ Y = C(\cdot) + D(\cdot)\, U + F(\cdot)\, \dot{U} \end{cases} \qquad (3.47)$$

avec

$$A(\cdot) = \left(I - S_{12}(\cdot)\left(F^d\right)^{-1} S_{21}(\cdot) F^i\right) \times$$

$$\left[S_{11}(\cdot) F^i + S_{13}(\cdot) L \left(I - S_{33}(\cdot) L\right)^{-1} S_{31}(\cdot) F^i\right]$$

$$B(\cdot) = \left(I - S_{12}(\cdot)\left(F^d\right)^{-1} S_{21}(\cdot) F^i\right)\left[S_{14}(\cdot) + S_{13}(\cdot) L \left(I - S_{33}(\cdot) L\right)^{-1} S_{34}(\cdot)\right]$$

$$C(\cdot) = \left[S_{41}(\cdot) + S_{13}(\cdot) L \left(I - S_{33}(\cdot) L\right)^{-1} S_{31}(\cdot)\right] F^i$$

$$D(\cdot) = \left[S_{44}(\cdot) + S_{13}(\cdot) L \left(I - S_{33}(\cdot) L\right)^{-1} S_{34}(\cdot)\right]$$

(*) Les composantes de la matrice de structure, ne sont pas tous forcément non linéaires.

(**) Ces transformations sont de même type que celles proposées par Rosenberg [1971] et aboutissent sans difficultés à l'équation recherchées.

$$E(\cdot) = S_{12}(\cdot)\left(F^{d}\right)^{-1} S_{24}(\cdot)$$

$$F(\cdot) = S_{42}(\cdot)\left(F^{d}\right)^{-1} S_{24}(\cdot)$$

En prenant compte le résultat de la section IV, il devient évident que la méthode graphique que nous avons proposée s'applique sans aucune difficulté (*).

VI.4.2 Cas où seuls les éléments sont non linéaires :

Contrairement au cas précédent, dans cette partie on va traiter toutes les éventualités possibles qu'on est susceptible de rencontrer dans un modèle bond-graph. Mais d'une manière générale la matrice de structure est celle correspondante au cas linéaire, et les relations entre les vecteurs, sont celles données dans IV.2, à savoir :

$$D_{out} = \Phi_{L}\left(D_{in}\right);\ Z^{i} = \Phi_{F^{i}}\left(X^{i}\right);\ Z^{d} = \Phi_{F^{d}}\left(X^{d}\right)$$

VI.4.2.1 "Pas de causalité dérivée"

Les conditions sont celles présentées dans la méthode de Rosenberg aboutissant à une équation de type $\dot{X} = \Phi(X, U)$ (cf. hypothèses VI.1).

Les équations représentant le bloc-diagramme de la figure 3.1 (sans le vecteur de sortie Y), se notent par :

$$\dot{X} = S_{11}Z + S_{13}D_{out} + S_{14}U \qquad (3.48)$$

$$D_{in} = S_{31}Z + S_{33}D_{out} + S_{34}U \qquad (3.49)$$

1* cas

Si le modèle bond-graph ne contient pas de boucles algébriques entre des éléments R (dissipatifs) nous nous plaçons dans un sous-cas qui favorise une détermination graphique de l'équation d'état. Dans cette condition la

---

* Risque de problèmes pour résoudre numériquement si les matrices sont non inversibles

sous-matrice S_{33} devient nulle. Nous pouvons réécrire alors l'équation (3.49) sous la forme :

$$D_{in} = S_{31}Z + S_{34}U = S_{31}\Phi_F(X) + S_{34}U \qquad (3.50)$$

on a alors :

$$D_{out} = \Phi_L\left(S_{31}\Phi_F(X) + S_{34}U\right)$$

En remplaçant cette valeur de D_{out} dans (3.48) on arrive à :

$$\dot{X} = S_{11}\Phi_F(X) + S_{13}\Phi_L\left(S_{31}\Phi_F(X) + S_{34}U\right) + S_{14}U \qquad (3.51)$$

Procédons à une analyse des termes intervenant dans l'équation (3.51). $S_{11}\Phi_F(X)$ caractérise le produit d'une matrice carrée par une fonction non linéaire du vecteur X. Or d'après l'hypothèse VI.2 nous pouvons écrire pour chaque composante X_j de X :

$$\Phi_F(X) = \begin{bmatrix} \Phi_F(X_1) \\ \cdots \\ \Phi_F(X_j) \\ \cdots \end{bmatrix} = \begin{bmatrix} \Phi_{F_{X_1}}(X_1) \\ \cdots \\ \Phi_{F_{X_j}}(X_j) \\ \cdots \end{bmatrix}$$

on a

$$\left(S_{11}\Phi_F(X)\right)_{ij} = \left(S_{11}\right)_{ij}\Phi_{F_{xj}}(X)$$

dont l'interprétation est la suivante :

$(S_{11})_{ij}$ =gain du chemin liant l'élément bond-graph associé à la composante X_j de X, à celui qui représente la composante $\dot{X}_i$ de $\dot{X}$.

$\Phi_{F_{xj}}(X_j)$ =Fonction non linéaire du vecteur d'état qui correspond à la loi caractéristique de l'élément bond-graph représentant la composante Xj de X.

L'exemple de la figure 3.7 présente une simple application du calcul de $s_{11}\Phi_F(X)$.

$$x = \begin{bmatrix} p_I \\ q_C \end{bmatrix} \qquad \Phi_F(x) = \begin{pmatrix} \Phi_{F_{x_1}}(x_1) \\ \Phi_{F_{x_2}}(x_2) \end{pmatrix} = \begin{pmatrix} \Phi_I(x_1) \\ \Phi_C(x_2) \end{pmatrix}$$

$$S_{11}\Phi_F(X) = \begin{pmatrix} 0 & \Phi_C(X) \\ -\Phi_I(X) & 0 \end{pmatrix}$$

figure 3.7

Cet exemple montre le calcul de la partie de l'équation d'état qui reflète une relation directe entre les éléments de stockage de loi non linéaire.

La signification du dernier terme de (3.51) rejoint totalement celle donnée dans le cas linéaire.
Le terme $(S_{14}U)_{ik} = (S_{14})_{ik} U_k$ correspond au gain du chemin liant l'élément bond-graph de type S_e ou S_f (associé à la composante U_k du vecteur d'entrée U) à l'élément bond-graph représentant la composante $\dot{X}_i$ de $\dot{X}$.

Le terme $s_{31}\Phi_F(X) + s_{34}U$ représente le vecteur D_{in} dont l'une des composantes est :

$$(D_{in})_i = \sum_{k=1}^{n} (s_{31})_{ik} \Phi_{F_{x_k}}(x_k) + \sum_{k=1}^{m} (s_{34})_{ih} U_h$$

avec n = dim(X) et m = dim(U).

L'interprétation graphique de $(D_{in})_i$ évoque l'addition de deux sommes de produits. La première est la somme de tous les produits des gains du chemin causal liant l'élément bond-graph dissipatif (associé à la composante R_i de D_{out}) à l'élément bond-graph de type I ou C (associé à

la composante X_k de X) ; multipliée par la fonction non linéaire représentant le module de l'élément physique correspondant à l'élément bond-graph associé à X_k. La seconde somme représente le gain du chemin causal liant R_i à la source (S_e ou S_f), associée à la composante U_h de U.

L'expression $S_{13}\Phi_L(S_{31}\Phi_r(X) + S_{34}U)$ représente le produit d'une matrice par une fonction de vecteur. Soit $(S_{13})_{kh}$, une composante de S_{13}, elle caractérise l'existence d'une relation causale entre un élément dissipatif (de type R, de rang h par rapport au vecteur D_{out}) et un élément de stockage d'énergie (associé à la composante $\dot{X}_k$ de $\dot{X}$).

Remarque :

Pour permettre une bonne compréhension de $S_{13}\Phi_L(D_{in})$, nous considérons Φ_L comme un simple gain et non comme une fonction. Ainsi pour chaque composante R_h de Dout : $(\Phi_L)_h = \Phi_{R_h}$, cette fonction caractérise le module non linéaire associé à l'élément physique (amortisseurs, résistances, etc...) symbolisé par la composante R_h.
Nous notons que le processeur Archer se base sur cette supposition, il produit le résultat final sous forme d'expression formelle en manipulant des chaînes de caractères. Nous utilisons pour cela une pile pour comptabiliser le nombre de parenthèse fermantes et une fonction de concaténation de chaîne, en plaçant les parenthèses à leur place après chaque détermination par une analyse graphique de différents termes intervenant dans l'expression.

Après cette remarque chaque composante de $S_{13}\Phi_L(D_{in})$ (associé à $\dot{X}_i$) peut s'écrire :

$$(S_{13}\Phi_L)_i = \sum_{l=1}^{r}(S_{13})_{il}\,\Phi_{R_l}$$

avec $r = \dim(D_{out}) = \dim(D_{in})$

Ce qui correspond à la somme des produits de gain du chemin causal liant une composante $\dot{X}_i$ (représentée graphiquement par un élément I ou C en causalité intégrale) à un élément dissipatif (représenté

graphiquement par un élément R), par le module non linéaire associé à ce dernier.
Soit d'une manière générale :

$$\left(S_{13}\Phi_L(D_{in})\right)_i = \sum_{l=1}^{r}\left(S_{13}\right)_{il}\Phi_{R_l}\left(\sum_{k=1}^{n}\left(S_{31}\right)_{jk}\Phi_{F_{x_k}}(X_k)+\sum_{h=1}^{m}\left(S_{34}\right)_{ih}U_h\right)$$

En tenant compte de ce qui précède on a :

$$\dot{X}_i = \sum_{j=1}^{n}\left(S_{11}\right)_{ij}\Phi_{F_{x_j}}(X) \qquad (a)$$

$$+\sum_{l=1}^{r}\left(S_{13}\right)_{il}\Phi_{R_l}\left(\sum_{k=1}^{n}\left(S_{31}\right)_{lk}\Phi_{F_{x_k}}(X)+\sum_{h=1}^{m}\left(S_{34}\right)_{ih}U_k\right) \qquad (b)$$

$$+\sum_{p=1}^{m}\left(S_{14}\right)_{ip}U_p \qquad (c)$$

Ce qui se traduit par l'analyse graphique suivante :

(a) = Somme de tous les produits du gain du chemin causal (lorsqu'il existe) liant une composante X_j de X (représentée par un élément bond-graph I ou C en causalité intégrale) à une composante $\dot{X}_i$ de $\dot{X}$ (représentée par le même type d'éléments), par la fonction non linéaire associée à l'élément physique caractérisé par l'élément bond-graph associé à X_j.

(b) = Somme de tous les produits du gain du chemin causal existant entre un élément R_l (représenté par un élément dissipatif de type R) et une composante $\dot{X}_i$ de $\dot{X}$ par une fonction non linéaire (associée à R_l) d'un terme composé de la somme de tous les produits du gain du chemin causal liant une composante X_k de X à R_l, multiplié par la fonction non linéaire de X (associé à X_k), et de la somme des gains des chemins causaux existant entre les sources associées à U et l'élément dissipatif associé à R_l.

(c) = La somme des gains des chemins causaux liant les composantes U (représentées par les sources effort ou flux du modèle bond-graph) et la composante $\dot{x}_i$ de $\dot{x}$ (représentée par un élément I ou C en causalité intégrale).

Exemple d'application :

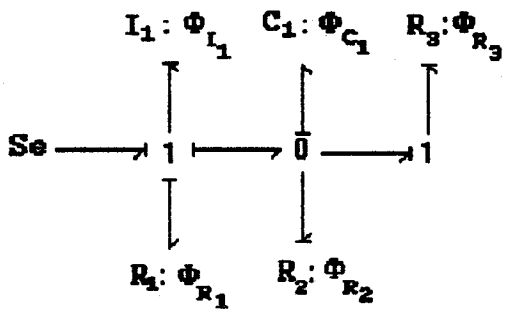


figure 3.8 Modèle bond-graph dont les éléments sont caractérisés par des fonctions non linéaires .

Nous allons déterminer l'équation d'état associée au modèle bond-graph de la figure 3.8. Ce modèle possède les hypothèses du cas traité précédemment à savoir : " pas de causalité dérivée et pas de boucles algébriques entre des éléments dissipatifs". L'application manuelle directe du résultat annoncée plus haut va aboutir à :

$$\dot{x}_1 = -\Phi_{C_1}(x) + \Phi_{R_1}\left(1.\,\Phi_{I_1}(x) + 0.\,\Phi_{C_1}(x) + 0.U\right) + 1.U$$

soit

$$\dot{x}_1 = -\Phi_{C_1}(x) + \Phi_{R_1}\left(\Phi_{I_1}(x)\right) + U$$

$$\dot{x}_2 = -\Phi_{I_1}(x) + \Phi_{R_2}\left(0.\,\Phi_{I_1}(x) + -1\,\Phi_{C_1}(x) + 0.U\right) + \Phi_{R_3}\left(0\,\Phi_{I_1}(x) + -1\,\Phi_{C_1}(x) + 0.U\right) + 0.U$$

soit

$$\dot{x}_2 = -\Phi_{I_1}(x) + \left(\Phi_{R_2} + \Phi_{R_3}\right) \mathbf{O} \left(-\Phi_{C_1}\right)(x) \qquad (*)$$

* Cette factorisation n'est pas toujours vraie.

2°cas "existence des boucles algébriques entre des éléments dissipatifs"

L'existence de boucles algébriques entre des éléments dissipatifs (R-couplés) va compliquer l'expression du vecteur D_{in} , et comme $S_{33} \neq 0$ nous avons :

$D_{in} - S_{33}D_{out} = S_{31}Z + S_{34}U$ qu'on peut écrire en utilisant les relations inter-vecteurs sous la forme :

$$D_{in} - S_{33}\Phi_L(D_{in}) = S_{31}\Phi_F(X) + S_{34}U \qquad (3.52)$$

Notons "Id" la fonction identité on a $Id(D_{in})=Din$, nous pouvons écrire que :

$$Id(D_{in}) - S_{33}\Phi_L(Id(D_{in})) = Id(D_{in}) - S_{33}\Phi_L \circ Id(D_{in})$$

Qui se ramène, sachant que $(f+g)(x)=f(x)+g(x)$, à :

$$(Id - S_{33}\Phi_L \circ Id)(D_{in})$$

En remplaçant dans (3.52) et en multipliant par la fonction inverse de $(Id - S_{33}\Phi_L \circ Id)$ nous arrivons à :

$$D_{in} = inv(Id - S_{33}\Phi_L \circ Id)[S_{31}\Phi_F(X) + S_{34}U] \quad (*)$$

qui se ramène, à cause de [f o Id(x)=f(x)], à :

$$D_{in} = inv(Id - S_{33}\Phi_L)[S_{31}\Phi_F(X) + S_{34}U]$$

Nous arrivons finalement à :

$$\dot{X} = S_{11}\Phi_F(X) + S_{13}\cdot\Phi_L \circ inv(Id - S_{33}\cdot\Phi_L)(S_{31}\Phi_F(X) + S_{34}U) + S_{14}U$$

---

* ici $inv(Id - S_{33}\Phi_L \circ Id)$ n'est pas une inversion de matrice, mais l'inverse d'une fonction vectorielle.

Une composante $\dot{x}_i$ de $\dot{x}$ peut être représentée par :

$$\dot{x}_i^i = \sum_{j=1}^{n} (s_{11})_{ij}\, \Phi_{F_{X_j}}(x_j) \qquad \text{(a)}$$

$$+ \sum_{l=1} (s_{33})_{il}\, \Phi_{h_l} \circ \mathrm{inv}(Id - s_{33}\Phi_L)\left(\sum_{k=1}^{n} (s_{31})_{lk}\, \Phi_{F_{Xk}}(x_k) + \sum_{h=1}^{m} (s_{34})_{lh}\, U_h \right) \qquad \text{(b)}$$

$$+ \sum_{r=1}^{m} (S_{14})_{ir}\, U_r \qquad \text{(c)}$$

A partir d'ici une interprétation graphique devient très compliquée, les termes (a) et (c) gardent les définitions données précédemment. Le terme (b) reste quasiment inexploitable graphiquement, seule une méthode numérique permettra de donner $\dot{x}_i^i$ sous sa forme finie. Cette complexité est liée à la fonction vectorielle $(Id - s_{33}\Phi_L)$ qu'il faut inverser. On peut toutefois produire une forme utilisant le plus possible une analyse graphique et réduisant les calculs(*).

VI.4.2 Existence de causalité dérivée

Les équations représentant le schéma bloc sont les suivantes :

$$\begin{aligned} \dot{X}^i &= S_{11}Z + S_{12}\dot{X}^d + S_{13}D_{out} + S_{14}U \\ Z^d &= S_{21}Z + S_{24}U \\ D_{in} &= S_{31}Z + S_{33}D_{out} + S_{34}U \end{aligned}$$

Sans chercher à rentrer dans les détails, nous présentons la forme générale de $\dot{X}^i$ soit :

$$\dot{X}^i = S_{11}\Phi_{F^i}(X) \qquad \text{(a)}$$

$$+ S_{12}\Phi_{F^d}^{-1}\left(S_{21}\frac{d}{dt}\Phi_{F^i}\left(X^i\right) + S_{24}\dot{U} \right) \qquad \text{(b)}$$

$$+ S_{13}\Phi_L \circ \mathrm{inv}\left(Id - S_{33}\Phi_L\right)\left(S_{31}\Phi_{F^i}\left(X^i\right) + S_{34}U\right) \qquad \text{(c)}$$

---

*Signalons que le programme Archer ne représente en mémoire que les composantes non nulles dans la sous matrice S_{ij}. On réduit ainsi le temps de calcul nécessaire pour déterminer, dans la phase de *simulation*, les valeurs correspondant à chaque vecteur.

$+ S_{14}U$ (d)

Les termes (a), (c) et (d) gardent les définitions rencontrées précédemment. La non présence de boucles algébriques entre des éléments dissipatifs entraîne que $S_{33}=0$, et simplifie la détermination graphique de (c). Le terme (b) ajoute une difficulté supplémentaire pour toute détermination graphique ou manuelle. Une méthode numérique s'avère alors nécessaire, mais plutôt que de laisser les termes dans leur forme générale, Archer produira des expressions qui maximise l'utilisation de l'analyse graphique.

VI.4.3 Cas où les éléments et la structure des jonctions sont non-linéaires

A ce niveau, toute tentative d'une méthode graphique devient quasiment impossible. Seul le cas d'inexistence de causalité dérivée et de boucles algébriques entre des éléments dissipatifs, reste possible graphiquement. Et même une écriture générale du vecteur $\dot{X}^i$ devient irréalisable. Nous pouvons seulement exprimer les équations du schéma bloc-diagramme de la figure 3.1, qui cède la place à un programme de simulation numérique. Ainsi le modèle sera symbolisé par les équations (3.53), qui représente une extension de la méthode de Rosenberg (cf. 3.V.2).

$$\begin{cases} \dot{X}^i = S_{11}(.)\,\Phi_{F^i}(X) + S_{12}(.)\,\Phi_F^{-1}{}_d \dfrac{d}{dt}\left(S_{21}(.)\,\Phi_{F^i}(X) + S_{24}(.)\,U\right) \\ \quad + S_{13}(.)\,\Phi_L(D_{in}) + S_{14}(.)U \\ \\ D_{in} - S_{33}(.)\,\Phi_L(D_{in}) = S_{31}(.)\,\Phi_{F^i}(X) + S_{34}(.)\,U \end{cases} \qquad (3.53)$$

Nous notons que Archer va proposer pour la phase de *simulation* cette forme générale accompagnée des différentes valeurs associées aux sous-matrices S_{ij} et éventuellement celles des fonctions Φ. Dans le cas particulier où il n'y a ni boucles algébriques entre des éléments dissipatifs, ni causalité dérivée, l'équation d'état sera donnée sous forme d'expression formelle en se basant sur la même analyse graphique proposée dans V.2.1.

VI.4.5 Conclusion

La détermination de l'équation d'état dans le cas d'un système non linéaire, présente, en fonction de la nature du modèle bond-graph représentant ce dernier, une difficulté plus ou moins croissante.

Rosenberg a proposé [1971] deux méthodes (selon qu'on a une causalité dérivée ou non) permettant de représenter dans un formalisme général le modèle d'état. Nous avons contourné les méthodes de Rosenberg afin de déceler les différents cas possibles liés à la nature de l'affectation causale du modèle bond-graph, pouvant maximiser l'utilisation graphique ou pratique.

Les limites de la méthode proposée pour arriver à l'équation d'état, simplement et sans utilisation de méthodes numériques, sont apparues dans le cas non-linéaire.

Pour certaines classes de non-linéarité nous avons proposé des méthodes s'appuyant sur une analyse graphique. Pour d'autres et en fonction de la nature du modèle bond-graph, nous avons essayé de simplifier les expressions du modèle d'état. Pour atteindre cette simplification nous avons utilisé une analyse pratique pour certains termes et nous avons restreint l'application numérique à d'autres. Cette cohabitation va permettre d'optimiser le temps pour calculer l'équation d'état à partir de méthode numérique.

VII Conclusion générale

Au cours de ce chapitre, nous avons exposé une synthèse de la détermination de l'équation d'état à partir d'un modèle bond-graph symbolisant un système dynamique. Ainsi une présentation générale du problème et des différents cas de figure qui peuvent surgir, en fonction de la nature du modèle du bond-graph, ont été développés sur la base des travaux de D. Karnopp & Rosenberg [1987].

Les points forts et les points faibles des différentes méthodes existantes, ont été discutés dans le cas de système linéaire et non-linéaire. La première de ces méthodes, connue sous le nom de la méthode de Rosenberg, est une méthode numérique qui s'applique à la fois à un système linéaire et à un système non linéaire. Cette méthode est d'une grande fiabilité, et a été implémentée dans plusieurs programmes utilisant la méthodologie bond-graph dont ENPORT, CAMAS, MS-BOND pour ne citer qu'eux.

La deuxième méthode est la seule, à notre connaissance, qui permet d'exprimer l'équation d'état sous forme paramétrique. Nous avons expliqué et montrer ses limites à la fois du point de vue théorie des bond-graphs et du point de vue implantation informatique.

Par la suite nous avons développé la méthode que nous proposons dans le cadre de ce travail et prouver son champ de validité à partir d'une analyse graphique des expressions des matrices d'état de la méthode de Rosenberg.

Une tentative de sa généralisation dans le cadre d'un système non linéaire, s'est révélée possible pour certaines classes de systèmes non linéaire. Et pour les autres nous avons poussé l'analyse graphique le plus loin possible afin de faciliter la phase d'application des méthodes numériques (ou toute phase de simulation).

Notre méthode qui s'intitule "Méthodes des boucles et chemins causaux" (MBCC), présente, dans le cas d'un système linéaire, un certains nombre d'avantages dont nous rappelons l'essentiel :

- Méthode graphique et manuelle (intérêt instructif et pédagogique).
-Rapide, même manuellement (l'exemple de la section VI donne une idée à la fois de sa rapidité et de son efficacité).
-Facile à programmer : son implémentation ne demande pas de structure de données compliquées, le calcul du déterminant revient assez souvent ce qui pour l'utilisation d'un langage orienté objet induira une économie non négligeable de codes sources. (*)

La MBCC s'applique également à des systèmes partiellement non linéaires.

---

* Les différents algorithmes liés à cette méthode, seront présentés dans l'Annexe A.

Chapitre 4

Spécificités Informatiques du Projet ARCHER

Introduction

Ce chapitre sera consacré essentiellement à l'analyse de la partie informatique de notre travail. Nous commencerons par présenter les techniques I.A. et voir leur application aux algorithmes d'Archer. Nous verrons alors quels sont les problèmes que nous avons affrontés et quelles solutions nous avons adaptées.

Ensuite suivra une partie situant Archer par rapport aux systèmes experts. Pour cela nous allons parcourir toutes les étapes d'Archer afin de montrer le niveau de formalisation et de modélisation de la connaissance qui a servi de base à ses programmes.

Puis, nous présenterons les diverses techniques utilisées par le projet Archer, à savoir :

- Comment a-t-il été conçu ?
- Comment répond-il à certains concepts de la programmation objet ?
- Comment gére-t-il ses bases des données ?
- Comment a-t-il évolué et quels sont ses outils de programmation ?

Ainsi que d'autres caractéristiques et particularités d'Archer :
- Traitement de langage naturel
- Calcul formel
- Fichiers d'aide, de configuration...

Et enfin nous ferons l'inventaire d'Archer et de ses perspectives, et nous présenterons une idée, qui est encore à l'état embryonnaire, concernant l'application des méthodologies bond-graphs à certains domaines de l'I.A, à partir de la physique qualitative.

I. Archer et l'Intelligence Artificielle (I.A.)

I.1. Situation d'une discipline

Depuis plus d'un quart de siècle l'I.A. fait l'objet d'espoirs, de critiques, de controverses et de confusions.

Actuellement, nous sommes loin de tous les pronostics faits au milieu de ce siècle. Et même les personnes qui, jadis, étaient de farouches adversaires de cette discipline ont fini par l'accepter, l'utiliser, voire participer à l'élaboration de ses projets.

L'idée, qui angoissait la plupart de ces personnes, telle que : *"les informaticiens cherchent a construire un modele complet (éventuellement amélioré) du cerveau humain"*, était à la fois très optimiste et même naïve. Car d'une part, cela concerne plutôt le domaine de la neurologie ; et d'autre part, il a été démontré que la réalisation d'une telle idée est impossible (voir dans J.P. Delahaye [1987.a] les neufs objections de Turing. Ils comportent le théorème de complétude de Gödel).

En revanche, l'idée selon laquelle l'informaticien ne se substitue nullement aux experts d'un domaine précis a calmé les esprits. Son rôle est de créer des algorithmes et des programmes permettant d'aider les experts à automatiser leurs tâches et à modéliser leur savoir-faire.

Cette frontière entre l'expert et l'informaticien fait l'objet d'un désaccord au sein même de la communauté des spécialistes de l'I.A et a donné naissance à deux thèses.

La première thèse défend l'idée selon laquelle l'I.A. est une science cognitive comme la psychologie, la linguistique, l'épistimologie, la philosophie et la neurologie. Pour elle l'ordinateur est un outil permettant de travailler sur des modèles etc...

Quant à la deuxième, elle situe l'I.A comme une branche de l'informatique dont le but est de rendre l'ordinateur plus habile. Or l'habilité aux niveaux vitesse, précision et calcul numérique complexe est la préoccupation d'autres branches. Le raisonnement symbolique n'est plus considéré comme faisant partie de l'I.A.(*)

---

* Cependant l'I.A. a contribué à la banalisation du raisonnement symbolique.

La définition de l'I.A. comme branche de l'informatique s'est précisée vers les années 80 (Lindsay et al. [1980]).

> *L'I.A. cherche à concevoir des programmes et des machines en mesure de traiter des problèmes pour lesquels on ne connaît pas de méthodes de résolution directes et assurées.*

Nous avons cherché à donner un aperçu des deux tendances pour pouvoir nous situer en tant que consommateurs des techniques de cette discipline. Débattre et discuter les deux thèses précédentes ne fait pas partie de nos objectifs. Ceci a été traité par H. Farreny et M. Ghallab [1987] qui ont montré l'opposition et la convergence des deux thèses et donné les références bibliographiques pour chacune.

En ce qui nous concerne, l'I.A. est une branche de l'informatique. Elle nous fournit un certain nombre de méthodes et de techniques pour réaliser des programmes difficiles pouvant engendrer une grande complexité.

I.2. Position d' Archer

Comme nous l'avons présenté au premier chapitre, Archer est un processus d'aide à la modélisation et à l'analyse des systèmes dynamiques utilisant la méthodologie bond-graph. Aussi les informaticiens, qui y travaillent, essayent-ils d'écrire des programmes basés sur l'expertise à la fois des bond-graphistes et des physiciens. Leur rôle consiste à chercher les meilleurs moyens pour représenter, organiser, et traiter les connaissances spécifiques aux objectifs d'Archer. Ils s'occupe également de le munir, pour améliorer le rapport Homme-Archer, de toutes les qualités que possèdent les logiciels du marché actuel (interface utilisateur convivial, temps de traitement réel...).

Le projet Archer caractérise un lieu où des spécialistes de domaines différents échangent leurs connaissances et leur savoir-faire qui s'avèrent parfois difficiles à implanter sur calculateur. Et pour y arriver correctement, nous avons eu recours aux outils et aux techniques de l'I.A.

I.3. Qu'entendons-nous par technique I.A. (TIA) ?

I.3.1 Historique

De nos jours, les frontières entre les techniques de la programmation avancée et celles de l'I.A. ne sont pas évidentes. Et le fait de se consacrer à des problèmes réputés pour leur difficulté ne constitue pas non plus un critère d'appartenance à cette discipline.

Cependant n'oublions pas que c'est grâce aux premiers chercheurs de l'I.A. et à cause de leur confrontation à des problèmes non déterministes qu'on a pu voir apparaître des langages tenant compte de la récursivité. Celle-ci est en relation directe avec la récurrence qui permet d'aborder la résolution de plusieurs problèmes.

Ainsi on a pu exploiter des structures récursives de données comme les arbres ou les graphes qui constituent les abstractions sur lesquelles nous représentons le monde réel. L'exploitation de la logique informatique a été incitée par le traitement et la représentation symbolique. Ceci a permis de développer des outils (J.P. Delahaye [1987.a]) qui ont conduit à la construction des démonstrateurs automatiques et à l'apparition des moteurs d'inférence.

L'I.A. a ajouté au style de programmation des méthodes particulières de résolution telles que les heuristiques, la marche arrière (backtracking) etc...

On a vu apparaître alors des centaines de langages spécialisés pour l'I.A. (M. Griffiths [1986]). Deux d'entre eux se sont distingués. Le premier est le LISP créé par J. McCarthy (1960 et 1970), il a permis à l'informatique de faire un grand pas. Le deuxième est PROLOG créé par A. Colmerauer et P. Roussel (1975), il est écrit selon la logique des prédicats et des propositions (logique d'ordre 1) et a été consacré dans les années 80 comme le langage par excellence de l'I.A.

I.3.2 Représentation, Stratégie et Organisation

Ces trois notions, qui constituent le noyau des techniques algorithmiques de l'I.A., sont étroitement liées. Toutefois, pour montrer leur application dans Archer nous les avons définies séparément. Elles sont largement développées dans la littérature spécialisée et notamment dans H. Farreny et M.Ghallab [1987], The Handbook of A.I. [1983, 1985, 1986, 1989].

La représentation caractérise la détermination d'une structure qui va symboliser le traitement du problème étudié.

On qualifie d' "étude de la stratégie" la partie qui s'occupe de vérifier la validité d'un algorithme avant son implantation.

Quant à l'organisation, elle caractérise l'ordre dans lequel seront traitées les alternatives qui seront gérées par une pile. Ainsi on distingue trois façons d'organiser une pile :

* LIFO ou DEPS (Dernier Entrant Premier Sortant), associé à une recherche en profondeur (simple à mettre en oeuvre).

* FIFO ou PEPS (Premier Entrant Premier Sortant), associé à une recherche en largeur (moins efficace en complexité).

* Recherche ordonné par rapport à un critère, par exemple le coût, la distance etc... (se prête bien au traitement heuristique)

I.3.3 Validité d'un algorithme

En I.A. la notion de résolution des problèmes est étroitement associée aux techniques de recherche d'heuristiques dans les graphes.

Ainsi, lors de l'écriture d'un algorithme de recherche valide, on se doit de s'intéresser particulièrement au non-déterminisme en guidant sa résolution et en gérant le backtraking pour éviter l'explosion combinatoire.

D'une manière générale la validité d'un bon algorithme est fonction de sa "terminaison", de son "admissibilité" et de sa "complexité" (P. Genthon [1989]), dont les définitions sont :
- La **terminaison** : elle assure l'arrêt de l'algorithme si une solution existe. Ceci correspond à la définition de la semi-décidabilité.
- L'**admissibilité** : elle consiste à fournir une solution qui tend vers l'optimum.
- La **complexité** : elle permet de connaître la taille mémoire disponible et le temps nécessaire pour avoir une solution.(*)

Remarque 5.1
La réduction de la complexité dépend surtout de l'amélioration des algorithmes et non de l'utilisation de compilateurs ou de matériaux performants.

---

* Les deux dernières sont souvent contradictoires.

I.4. Application des TIA à Archer

Comme nous l'avons signalé précédemment, l'I.A. s'attaque à des problèmes dont les données sont complexes, souvent représentées par des graphes ou par leurs dérivés (forêt, arbre, arborescence, graphe d'état etc...). La théorie des graphes (M. Minoux et M. Gourdon [1979], F. Buckely et F. Harary [1990]) nous donne des outils de représentation sous forme de matrices d'incidence, d'adjacence et autres.

Dans le cas du bond-graph, on doit manipuler ces structures de sorte qu'elles contiennent d'autres informations que la simple existence d'une liaison entre deux noeuds. Et pour pouvoir les traiter, on a intérêt à décomposer, au maximum, une représentation des données et à créer des relations entre ses différents niveaux.

Nous avons appliqué ce principe lors de la détermination des informations causales. En effet, on peut remarquer que l'utilisation des informations véhiculées par un chemin causal élémentaire "cce" (*) représente le niveau le plus bas d'une information causale. Il sera suivi des chemins causaux directs "ccd", puis des chemins causaux indirects "cci" et des boucles causales.

Cette manière, généralement adoptée par les systèmes de gestion des bases de données, se prête bien à la Programmation orientée objet car le principe d'hiérarchisation est favorisé.
En I.A., le type de traitement ressemble souvent à la forme des données traitées. Donc après une décomposition hiérarchique des données, nous nous intéressons à des structures qui vont aider à leur exploitation.

En ce qui nous concerne, les données sont souvent représentées sous forme de listes. Cette structure de Turbo Prolog (l'un des langages utilisés pour développer Archer), permet des traitement récursifs et peut être utilisée sous plusieurs aspects (tableau, ensemble, pile, arbre, graphe...).

Nous avons montré (*) que pour traiter un bond-graph on est souvent amené à le considérer comme un graphe classique. Dans ce cas, les boucles entre des jonctions et les chemins causaux fermés correspondent à des circuits. Ces derniers engendrent souvent des situations non déterministes nécessitant une stratégie de résolution.

Pour s'en rendre compte, parcourons quelques programmes d'Archer :

---

* Voir chapitre 2

La première phase d'Archer consiste à construire un modèle bond-graph à partir de la description formelle d'un système dynamique. Cette opération s'effectue par l'intermédiaire d'un analyseur syntaxique, d'un graphe d'état et d'un analyseur sémantique.

Bien que le graphe d'état possède une situation non-déterministe, son traitement se fait sans difficulté apparente. A ce niveau, la stratégie consiste à lire deux caractères quand cette situation est identifiée.

La phase de l'affectation de la causalité, quant à elle, est confrontée pendant le parcours du bond-graph à des problèmes de complexité acceptable, simples à résoudre. Les stratégies, qui lui sont associées, permettent d'identifier des éléments déjà traités et de ne pas s'engager dans des circuits. Cette phase est confrontée surtout à des problèmes décisionnels. Ceux-ci seront traités dans la section suivante.

Pour la détermination des informations causales, nous traitons seulement les programmes associés aux "cce" et aux "cci". Les autres ne présentent aucune difficulté.

Nous avons vu que ces deux programmes se basent sur une représentation graphique, et que -selon certains critères ou heuristiques- on ne s'engage pas dans l'exploration de certaines branches. Ces deux programmes risquent une lourde combinatoire que nos heuristiques ne permettent pas d'éviter. L'objectif de chacun d'eux, nécessite une exploration totale et la détermination de tous les chemins -de même famille- existant dans le bond-graph.

Nous ne cherchons donc pas un chemin répondant à un critère donné (coût ou autres) mais tous les chemins, car ils interviennent tous dans la détermination des modèles mathématiques.

Nous avons alors choisi pour le premier une forme permettant d'explorer le graphe, symbolisant un bond-graph causal, arbre par arbre. Le temps du traitement est certes augmenté ; par contre la mémoire n'est pas saturée et la terminaison est assurée.

Quant au deuxième, une explosion combinatoire est inévitable dans le cas où plusieurs "ccd" existent.

Pour se rendre compte de sa complexité, il faut essayer de déterminer, dans un graphe non orienté, dont les noeuds sont à la fois des noeuds de départ et d'arrivée, tous les

chemins de longueur supérieure ou égale à 2. Si ce graphe a une grande densité et s'il possède des circuits, la complexité est alors exponentielle (Problème NP).

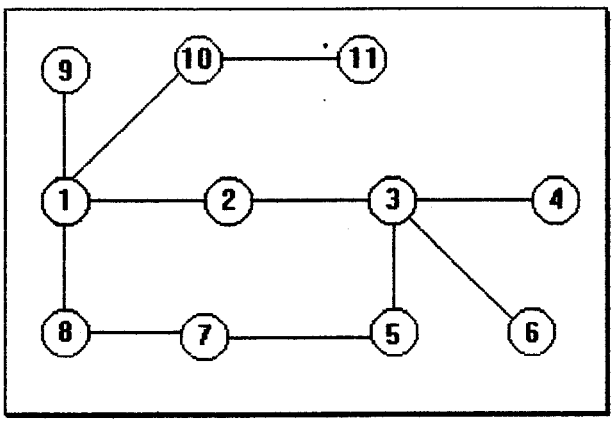


figure 5.1 : exemple de graphe non orienté formé à partir des différents chemins causaux direct "ccd" permettant de déterminer les "cci"

Les quelques heuristiques que nous avons pues dégager ne permettent pas de palier ce problème. Pour y remédier, nous avons préféré déterminer ce type de données uniquement quand deux éléments ne sont pas directement liés causalement.

La complexité est alors réduite par rapport au problème initial, mais l'admissibilité est loin d'être optimale, car la probabilité de faire des traitements inutiles atteint son maximum. Pour s'en rendre compte, considérons l'exemple de la figure 5.1. Son but est de déterminer tous les chemins liant le noeud 1 au noeud 7. Comme on peut le remarquer, il existe deux chemins [1,2,3,5,7] et [1,8,7], mais on est bien obligé de traiter tous les autres chemins.

La pile de parcours va avoir les états suivants:

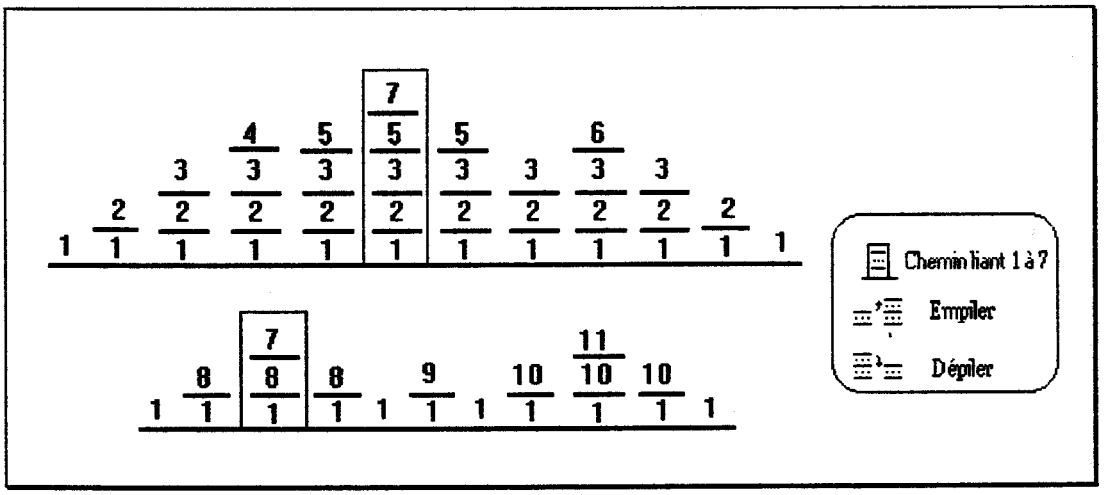


figure 5.2 : etats de la pile de parcours du graphe de la figure 5.1

Nous avons essayé de traiter le problème par dichotomie en utilisant deux piles. L'une part du noeud de départ, l'autre de celui d'arrivée. Cette méthode s'avère intéressante uniquement pour les graphes possèdant une parfaite symétrie.

L'absence d'heuristiques nous interdisant de nous engager dans telle ou telle branche du graphe, nous contraint à utiliser, faute de mieux, un algorithme, de complexité importante, qui assure néanmoins une solution.

Résultent alors deux inconvénients : le temps de traitement (pas très excessif) et le risque d'une saturation de mémoire vive, puisque la détermination des "cci" liant deux noeuds particuliers est toujours sollicitée par d'autres programmes en cours d'exécution. Ceci nous a incités à bien gérer la mémoire vive (nous reviendrons sur ce point ultérieurement).

Le programme, qui a pour rôle de vérifier si une boucle est disjointe avec une autre ou avec un chemin causal, atteint parfois une complexité excessive (à partir de 30 boucles causales). Nous avons signalé au chapitre 4 une méthode permettant de résoudre complétement ce genre de problème.

En général, les autres algorithmes se terminent normalement, possèdent une admissibilité acceptable et une complexité tolérable.

II. Archer et les systèmes experts

Tout au long de cette partie nous essayerons à travers des questions, d'apporter des éléments de réponses par rapport aux systèmes experts en général et de situer Archer vis-à-vis de cette technique en particulier. La littérature dans ce domaine est prospère(*).

II.1. Qu'est-ce qu'un Système Expert (SE)?

Un système expert est un programme informatique qui se base sur un ensemble de connaissances ainsi que sur le savoir-faire d'un ou plusieurs experts d'un domaine donné. Il utilise des techniques d'inférence pour résoudre des problèmes non

---

* Entre autres, "The handbook of artificial intelligence" Vol. I [1983], J. P. Delahaye [1987.b], H. Farreny et M. Ghallab [1987], C. Townsend [1987], P. Frot [1987] (ces deux derniers sont consacrés à la réalisation des systèmes experts sur des micros.

structurés pour lesquels il est difficile, voire impossible, de définir une résolution par procédures.

En revanche, si le processus intellectuel d'un expert faisant autorité dans son domaine est entièrement modélisé, le logiciel qui en découle n'est pas forcément considéré comme un système expert.

Pour comprendre cette définition, il faut considérer un SE comme un outil qui exploite les connaissances de l'expert et les transmet à l'utilisateur qui s'en servira pour résoudre les problèmes qu'il est susceptible de rencontrer. Donc un système expert, contrairement aux programmes informatiques classiques ou procéduraux, est à la fois un instrument de mise en oeuvre et de transfert d'un savoir-faire.

Remarque 5.2 :

Deux pièges sont à éviter :

-croire qu'un système expert remplacera les meilleurs spécialistes.
-croire qu'un SE dispensera de tout travail d'analyse approfondi.

D'un autre côté, ce ne sont nullement des gadgets gratuits que de bons algorithmes suffiraient à remplacer, car leur technicité est très riche et d'une grande efficacité.

II.1.1 Comment développer un SE ?

La réalisation d'un système expert se fait en deux phases :

La collecte et la formalisation de l'information relative aux activités du domaine étudié, ce que nous pourrons nommer "phase cognitive". Celle-ci détermine le fondement d'un SE, parce que transformer les connaissances d'un expert en un ensemble d'informations décrites dans un langage accessible à l'ordinateur est une opération délicate.

Les experts disposent d'un savoir plus ou moins inductif, voire expérimental. Cette connaissance, dite de "surface", permet de résoudre plusieurs problèmes de fonctionnement mais s'avère parfois insuffisante et incertaine.

La formalisation de la connaissance et du savoir-faire a donné naissance à une nouvelle profession : "ingénierie de la connaissance" ou "ingénieurs cogniticiens".

La deuxième phase caractérise le travail informatique, à savoir :

-L'écriture de la base de connaissances dans un langage de représentation (règles de déduction, objets, frames...) en laissant à l'expert la possibilité de définir son propre vocabulaire. Elle peut être construite progressivement (plusieurs experts pourraient participer à son élaboration), elle est généralement scindée en une base de faits et une base de règles.

-Le développement d'un générateur de SE formé par un programme de résolution ou moteur d'inférence et par un programme d'aide à la formalisation des connaissances.

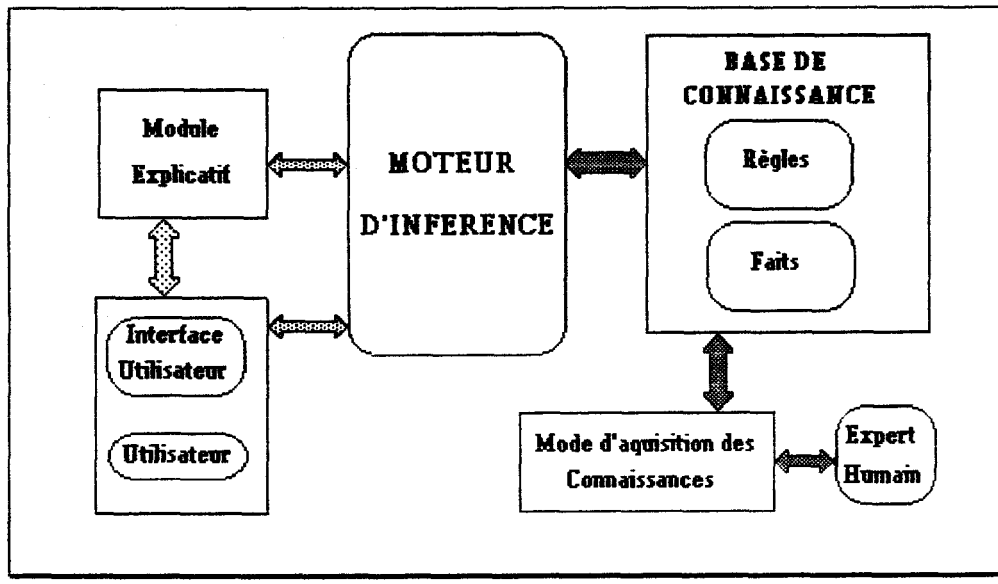


figure 5.3 : Architecture d'un SE

II.1.2 Qu'est-ce qu'un Moteur d'Inférence (MI) ?

C'est un programme dont le rôle est de simuler la "réflexion" de l'expert en appliquant le savoir-faire formalisé -sous forme de règles- aux faits d'un problème en cours de traitement.

Son souci est de déclencher les règles adéquates. Pour cela, son cycle de fonctionnement suit trois étapes :

-Le **filtrage** ou la constitution d'un ensemble de conflits à partir de règles dont les prémisses appartiennent à la base de faits.

-La **résolution de conflits** à partir d'un critère de priorité ou par établissement de méta-règle. Etape qui pourrait engendrer un problème décisionnel et risquerait, sans recherche préalable d'heuristiques, une explosion combinatoire.

-L'**exécution** qui peut être sous forme de l'ajout de nouveaux faits à partir de la "partie condition" des règles, de la suppression d'un fait, du lancement d'un affichage graphique des résultats ou éventuellement de la réalisation d'une opération. Par exemple, donner un ordre à un robot.

Le cycle 

est répété par le MI jusqu'à la satisfaction d'une condition d'arrêt ou la saturation de la base de connaissances.

II.1.3 Comment fonctionne un M I ? (*)

On distingue deux modes de déclenchement des règles.

-**chaînage avant** ou déduction de nouveaux faits à partir de faits et règles actifs. Il s'agit d'un raisonnement depuis les données (faits communs) vers les buts (faits à établir). Ce raisonnement est connu sous le nom de "modus ponens".

-**chaînage arrière** ou vérification d'une hypothèse. Dans ce cas le système va essayer d'atteindre soit le **but** soit la conclusion d'une règle (par vérification de toutes ses conditions). Chaque condition non vérifiée deviendra le but à atteindre, donc un traitement favorablement récursif. Ce raisonnement est connu sous le nom de "modus tolens".

-**chaînage mixte** : plusieurs systèmes combinent les deux modes, on parle alors de **chaînage mixte**. Ce dernier dispose à la fois d'un raisonnement déductif et inductif. Par conséquent, il est plus fidèle aux stratégies utilisées par de nombreux experts.

II.1.4 Qu'est-ce qu'un générateur de SE ?

---

*Pour plus de détails sur les moteurs d'inférence et sur leurs différentes logiques formelles ainsi que sur la monotonie d'une base de connaissances, voir J.P Delahaye [1987.a] et H. Farreny et M. Ghallab [1987].

Cette catégorie de progiciels se nomme aussi "système essentiel", il s'agit d'un "noyau vide" de SE, c'est-à-dire sans connaissance particulière dans un domaine ou un autre. Il suffit alors d'établir une expertise et de la lui associer pour avoir un SE. Ceci est rendu possible par la séparation entre la base de connaissances et le raisonnement du moteur (*).

II.2. Archer est-il un système expert ?

Le projet Archer, comme le montre la figure 1.1 du chapitre 1, est constitué de modules basés sur diverses expertises. Ses programmes peuvent être scindés en deux familles.

La première famille caractérise les programmes qui utilisent une connaissance entièrement formalisée à partir de diverses méthodes rencontrées dans la littérature bond-graph. La plupart de ses algorithmes sont bien définis et ne présentent pas de situations abstraites, ni incertaines, ni imprécises. Donc ils peuvent être implantés efficacement à partir de langages procéduraux.

On y trouve les programmes de détermination des modèles mathématiques (**) qui ont été conçus, à partir de l'exploration de l'architecture d'un bond-graph.

Prenons comme exemple la méthode de détermination de l'équation d'état que nous avons proposée dans le cadre de ce travail (***). Son but est de déterminer la formule permettant de donner les composantes non nulles sous forme d'expressions formelles. Elle se distingue des méthodes existantes, qu'elles soient numériques ou paramétriques, par son aisance et sa rapidité. D'autre part, elle prend en compte toutes les éventualités susceptibles d'exister dans un modèle bond-graph.

Le programme qui lui est associé n'est pas pour autant un système expert : bien qu'il applique les règles définies par cette méthode, il ne rencontre jamais de situation non déterministe ni de problème de choix ou de prise de décision. Le savoir-faire à ce niveau est entièrement défini et ne présente pas de raisonnement flou ni expérimental.

---

* Parmi les produits connus, on trouve EMycin (Empty Mycin), qui a été extrait du célèbre SE de diagnostic médical MYCIN ; SPT (Système Propositionnel Type) Delahaye [1987].

** Voir annexe A.

*** Voir chapitre 3

De la même manière, le programme de détermination de la matrice de transfert pour un système linéaire est construit à partir de méthodes bien définies (**).

Remarquons que les programmes de détermination des informations causales font également partie de cette famille.

La deuxième famille est constituée de deux programmes qui ne sont pas totalement indépendants : L'**affectation de la causalité** et l'**analyse structurelle.**

Le premier, qui a fait l'objet d'une partie du chapitre 2, se compose de trois étapes : Causalité obligatoire "co" - Causalité préférentielle "cp" - Causalité décisionnelle "cd".

Le but de chacune est de déterminer une liste de "représentants causaux" (*). Cette liste, lorsqu'elle existe, sera traitée par un module, nommé "déduction causale". Il s'occupe d'affecter la causalité en s'appuyant sur 5 règles appelées "règles de déduction" (**). Celles-ci permettent de déduire d'autres représentants causaux. D'où une sorte de récurrence aboutissant à un processus récursif.

L'automate d'états finis de la figure 5.4 schématise toutes les transitions possibles pour affecter la causalité à un bond-graph.

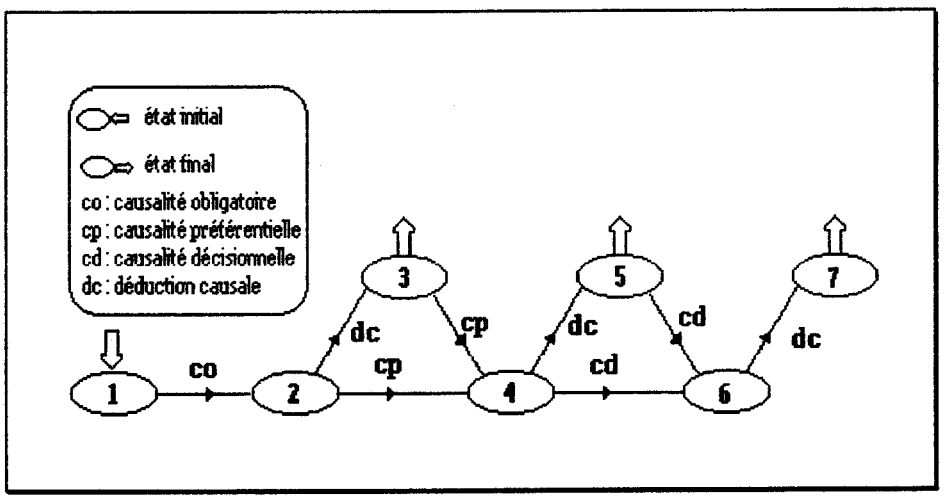


figure 5.4 : automate à états shématisant le processus de la procédure de l'affectation de la causalité

---

* Voir la définition d'un représentant causal au chapitre 2, page 34

** Voir page ...

Les mots syntaxiquement corrects, formés à partir de l'alphabet {"co", "cp", "cd", "dc"}, correspondent aux chaînes liant l'état initial aux états terminaux. On trouve ainsi les 6 mots suivants :

mot syntaxiquement correct	sémantique associée
codc	cas où les représentants causaux de l'étape "causalité obligatoire" vont déduire la causalité de tous les liens du modèle bond-graph.
codccpdc	ici ce sont les représentants causaux des étapes "causalité obligatoire" et "causalité préférentielle" qui se chargent de déduire la causalité de tous les liens.
codccpcddc	-affectation de la causalité à partir de l'existence de causalité obligatoire, -pas de causalité préférentielle, -affectation de la causalité à partir d'un choix décisionnel.
cocpdccddc	-pas de causalité obligatoire -existence de causalité préférentielle -affectation de la causalité à partir d'un choix décisionnel.
cocpcddc	-pas de causalité obligatoire ou préférentielle -affectation de la causalité à partir d'un choix décisionnel.
codccpdccddc	toutes les étapes ont lieu.

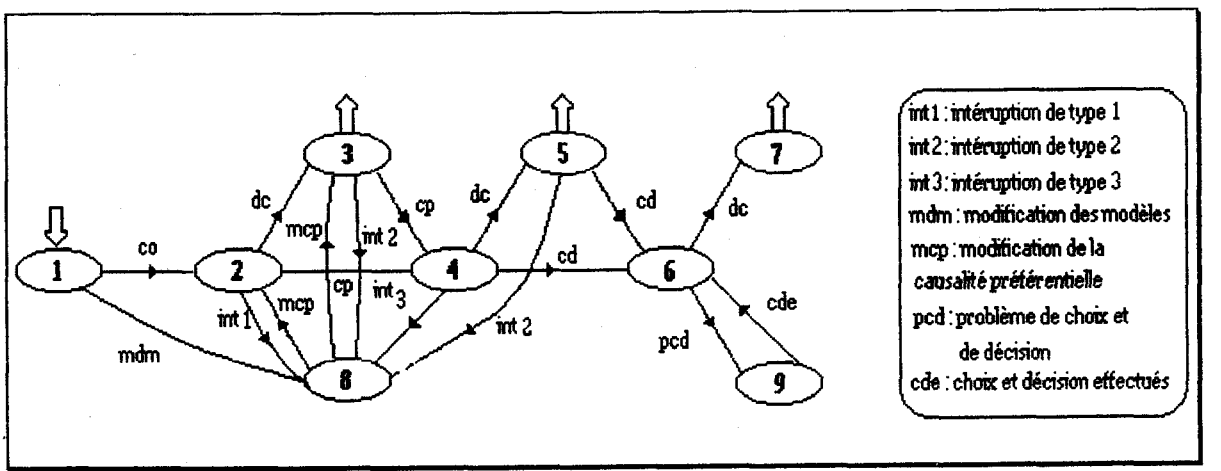


figure 5.5 : automate symbolisant une session complète de l'affectation de la causalité

L'automate de la figure 5.5 montre toutes les phases du processus de l'affectation causale. On y constate trois types d'interruption nécessitant à la fois un dialogue avec

l'utilisateur et un "retour arrière" dans le déroulement, progressif ou normal, de l'action de l'affectation automatique de la causalité.

Ces interruptions sont le résultat d'une contradiction dans le modèle physique étudié. Ce type d'erreurs sémantiques est indétectable pendant la phase de construction du modèle bond-graph. Aussi peut-on dire qu'à ce niveau le programme d'affectation de la causalité est un prolongement de l'analyseur sémantique associé au modèle traité.

La première interruption est détectée au niveau de l'état 2 par une fonction possédant plus d'un représentant causal. Cette situation, qui correspond à un "conflit induit"(*), nécessite absolument une intervention pour modifier la description (texte et/ou graphique) du modèle physique ou des lois associées à ses éléments.

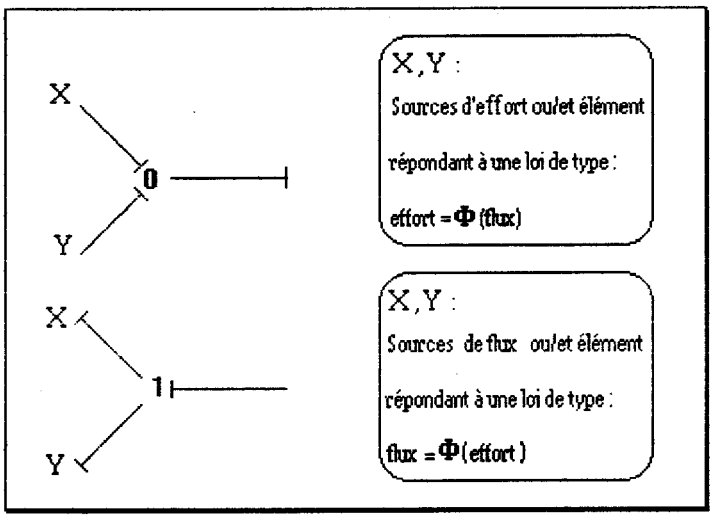


figure 5.6 : type de "conflit induit"qui provoque une interruption de type 1

Comme le montre la figure 5.6, on peut être confronté à un problème de prise de décision et de choix entre plusieurs possibilités :

-suppression d'un ou plusieurs éléments (lesquels ?)
-modification d'une ou plusieurs lois physiques associées aux éléments concernés
-déplacement d'une partie du système.

L'interruption de type 2 peut se produire au niveau des états 3 et 5. Elle correspond à un conflit induit à partir d'au moins deux déductions causales. Elle va nécessiter, à une exception près (**), l'adjonction d'un ou plusieurs éléments. Là aussi, nous allons

* Voir chapitre 2.

** pour certains types de systèmes électroniques (A. Castelin et al [1991]).

être confrontés à un problème que seuls les spécialistes du système physique étudié peuvent résoudre. Donc problèmes de décision et de choix qui dépendent de la nature du système étudié, de ses particularités, etc.

Notons que cette phase correspond également à une façon d'analyser sémantiquement le système physique étudié.

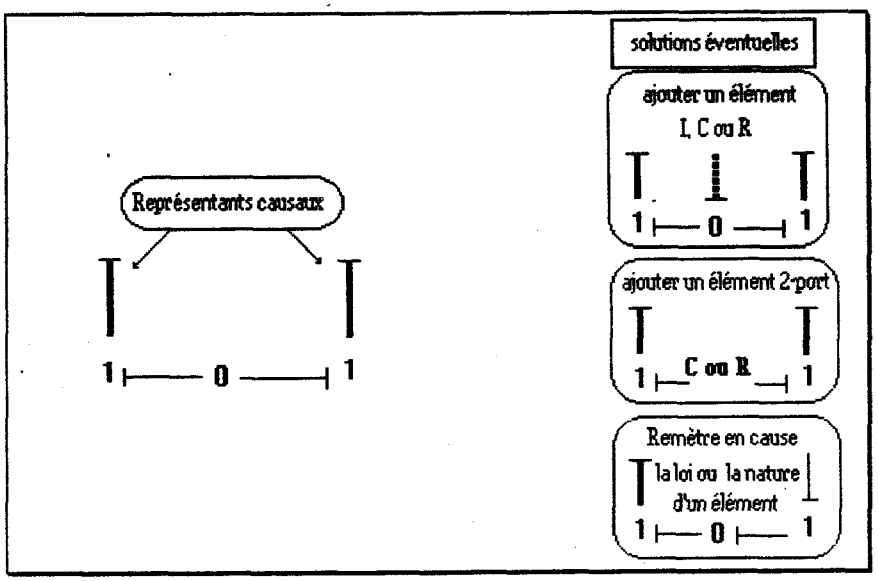


fig 5.7 : exemple d'une situation provoquent une interruption de type 2 et solutions éventuelles

La figure 5.7 montre quelques exemples de ce problème ainsi que des solutions possibles. Il ne faut cependant pas oublier d'établir des règles particulières pour déterminer la nature de l'élément (I, C ou R) qu'il faut ajouter et le nombre de ports qu'il doit contenir.

Lorsqu'il s'agit d'un élément 1-port, il faut décider s'il doit être dépendant ou indépendant, car cela a un rapport direct avec l'ordre du système étudié.

L'interruption 3 peut être engendrée, pendant l'étape de la causalité préférentielle, à partir de situations de même type que celles qui provoquent les intéruptions 1 et 2. Cependant on peut éviter ces situations en éliminant les éléments, des listes proposées à l'utilisateur, directement affectés par un choix préférentiel.

Quels sont les problèmes qui peuvent en découler ?

Premièrement, toute modification (suppression, adjonction, déplacement) intervenant sur les composants du système dynamique provoquera celle du modèle bond-graph et

de sa causalité. A ce niveau on peut, soit interrompre d'une manière classique tout processus par retour au système d'exploitation ; soit prendre en charge toutes les opérations en faisant des propositions et en se basant sur un certain nombre de règles.

Le cahier des charges d'Archer ne prévoyait pas une modification des modèles physique et bond-graph à partir de l'affectation de la causalité. Lorsque ce problème s'est posé, nous avons opté pour la stratégie de prise en charge des opérations. Ce qui a nécessité la collecte de connaissances relatives à ce sujet pour pouvoir développer un générateur de règles.

L'interruption de type 3 est liée à un mauvais choix de la causalité que l'utilisateur a donnée à certains éléments.

En ce qui concerne Archer, l'étape de la causalité préférentielle est entièrement basée sur un interface utilisateur dont les propositions se font sous forme de menus déroulants. Aucun choix n'est définitif. Il va l'être lorsque l'utilisateur désire quitter cette application. Et tant qu'il se trouve dans cet interface, il est maître d'affecter la causalité, de modifier ou de remettre en cause ses choix. Par conséquent, pour éviter des traitements inutiles, dès qu'un choix est imposé, tous les éléments qui seront remis en cause ne seront plus proposés. L'annulation de ce choix entraînera automatiquement la réapparition de ces éléments.

Poursuivons l'analyse de l'automate associé à la phase de l'affectation causale. Nous rappelons(*) que l'étape de la causalité décisionnelle est basée sur le principe suivant :

> *Provoquer des conflits - Constituer une liste des compétitions causales - Choisir, en fonction des possibilités, un ou plusieurs représentants causaux.*

Cette étape, qui constitue le noyau de toute affectation causale, a pour but de maximiser la causalité -intégrale ou dérivée- imposée à un modèle bond-graph. Ainsi, nous avons relevé un certain nombre de points :

-problème de décision (quelle est la meilleure possibilité pour la résolution d'une liste de compétitions ?)

-problème de choix (quels éléments choisir ?)

-problèmes de prédiction, comment :

---

* voir chapitre 2.

. éviter les boucles algébriques (d'ordre 1+) ?
. éviter les boucles associées à un chemin causal fermé ?
. réduire les boucles et la taille des chemins directs ?
. favoriser l'analyse structurelle (observabilité, commandabilité...) ?

-problème de maniabilité (comment garder le même traitement quand on modifie le type de la causalité imposée au bond-graph ?).

Nous avons pu constater que les méthodes existantes, comme SCAP (Séquencial Causality Assignement Program) de Karnopp et Rosenberg, sont intéressantes pour un système de rang maximal, mais elles ne tiennent pas compte des réflexions précédentes. Toutefois, on pourrait se demander pourquoi compliquer une procédure simple et quelle est l'importance de définir de manière précise la causalité des éléments, lorsqu'une causalité mixte est essentielle.

Puisque la causalité est un moyen primordial pour la modélisation et l'analyse d'un système dynamique, la méthodologie bond-graph n'est pas l'apanage des seuls connaisseurs en la matière. D'autre part, pendant l'affectation de la causalité, on pourrait résoudre d'éventuels problèmes de boucles algébriques (non décidables) et vérifier les propriétés structurelles du modèle.

Après cette présentation, il devient clair que le problème tel que nous l'avons posé ne peut être résolu simplement par des algorithmes procéduraux. Pour sa résolution, la conception d'un système expert devient nécessaire.

Dans un premier temps nous allons distinguer la partie bien formalisée de celle qui ne peut être modélisée de manière définitive.

En nous référant aux conventions(*), notées "i" ou "d", nous provoquons un type particulier de conflit (noeud de convention 'd'). Cette technique permettra de localiser un ensemble d'éléments ou de listes d'éléments qui entrent en compétition pour résoudre un conflit de causalité.

En se basant sur les règles statiques, qui jouent le rôle d'heuristiques (**), on peut localiser et constituer une liste de compétitions causales.

---

* voir chapitre 2, page 48

** voir chapitre 2, page 67

Ces règles sont bien spécifiées et peuvent être placées dans un ordre qui ne permet d'en déclencher qu'une seule à la fois. Dans Archer, elles correspondent à des clauses de Turbo Prolog dont le processus d'unification se fait par chaînage arrière. Ceci dit, tout autre langage procédural aurait convenu, mais Turbo Prolog présente l'avantage d'être rapide lorsqu'il s'agit de déclencher la bonne règle.

D'autre part, se posera un problème de décision et de choix lorsque la résolution, par l'intermédiaire de la liste de compétitions, d'un vrai conflit causal sera confrontée à plusieurs possibilités. A ce niveau, on fait appel à une connaissance liée à la nature des éléments et à la structure du système étudié pour guider et conseiller l'utilisateur. Ce savoir-faire correspond aux règles dynamiques définies au chapitre 2.

Par ailleurs, l'analyse structurelle d'un modèle bond-graph constitue la partie d'Archer qui nécessite un gestionnaire de connaissances : au chapitre 4, nous avons signalé des méthodes d'analyse relatives à la commandabilité et l'observabilité (en état et en sortie) d'une part ; la non stabilité asymptotique et la réduction des modèles (voir C. Sueur [1990.b]) d'autre part. Ce sont des méthodes pratiques, basées sur la manipulation de la causalité des éléments. Elles consistent à imposer la causalité dérivée au bond-graph et à dualiser la causalité des éléments représentant les entrées (les sources) et les sorties (détecteurs).

Du point de vue pratique, ces méthodes paraissent simples. Mais pendant leur mise en oeuvre automatique, on pourrait se heurter à un raisonnement non déterministe. En effet, lorsqu'on impose la causalité dérivée à un même modèle bond-graph, plusieurs versions peuvent exister.

D'après C. Sueur [1990.b], une seule version permet de minimiser le nombre de commandes et de détecteurs que nécessite le système étudié. D'ailleurs cette information indique s'il faut ajouter ou supprimer des sources, des détecteurs ou des éléments de stockage d'énergie (non observables et non commandables).

Cependant les emplacements des commandes et des détecteurs doivent être choisis de manière judicieuse. Ceci ne peut se faire correctement qu'à partir de règles spécifiques au domaine étudié.

Là encore, le manque d'un savoir-faire pourrait entraîner un traitement incertain. D'où la nécessité de prévoir un gestionnaire de connaissances.

A la question Archer est-il un système expert ? Nous pouvons répondre :

A la question Archer est-il un système expert ? Nous pouvons répondre :

Archer est un projet qui a plusieurs objectifs. Ses programmes, se basent sur une connaissance diversifiée tirée des savoir-faires de bond-graphistes, d'automaticiens et de physiciens appartenant à plusieurs domaines.

Une bonne partie de ce savoir-faire est entièrement modélisée et ne présente aucune abstraction. En revanche, certains de ses programmes se basent sur des méthodes pouvant entraîner des situations non déterministes, nécessitant la recherche d'heuristiques pour les débloquer. Par exemple, l'affectation de la causalité engendre souvent des problèmes de choix et de décision.

La connaissance est parfois floue, par conséquent difficile à implanter. Elle réclame une étude permettant de la formaliser sous forme de règles explicites. Par exemple, l'expression "judicieusement placé", rencontrée dans les méthodes d'analyse, est difficilement interprétable. Elle peut même aboutir à un problème décisionnel, car son sens peut varier d'un domaine physique à un autre.

Archer n'est donc pas un simple SE, cependant il possède des programmes qui peuvent être considérés comme tel.

III. Archer et la Programmation Orientée Objet (POO)

Archer est un projet informatique conçu selon une technologie orientée objet. On peut le partager en deux classes importantes, le traitement symbolique et l'échange avec l'extérieur (interface utilisateur, communication avec les périphériques).

La première est développée en Turbo Prolog, qui possède certaines caractéristiques des Langages Orientés Objet (**LOO**) ("les données sont exprimées sur le problème en termes de faits et de relations entre les faits" C. Towsend [1988]). Bien que nous ne partagions pas entièrement cette idée, nous pensons que Turbo Prolog possède certains concepts de la POO.

Cette dernière, en tant que technique, pourrait s'avérer d'une grande richesse lors de la conception d'un projet. Néanmoins l'utilisation d'un LOO est préférable puisqu'il permet d'appliquer tous les concepts d'une bonne POO.

La deuxième classe est écrite, quant à elle, en Turbo Pascal Windows. Ce logiciel, qui est en réalité un langage objétisé, est considéré comme un LOO.

III.1. Qu'entendons-nous par Conception Orientée Objet (COO) ?

La COO est une manière d'étudier un projet de développement de logiciel avant et pendant sa réalisation informatique. Son but est de faciliter les corrections et les améliorations sans ébranler l'ensemble du projet.

Pour cela, elle doit s'orienter suivant trois axes :

- Définir et maximiser la modulation. C'est-à-dire mettre en évidence toutes les articulations du projet pour permettre de classifier les tâches des intervenants, de déterminer les données qu'elles vont produire ou s'échanger.

- Rendre indépendant chaque module qui s'occupera de ses propres données ainsi que de leur évolution. Aucune modification extérieure des données ne doit survenir.

- Représenter les données de manière abstraite. Autrement dit, elles doivent être indépendantes de la structure qui les contient. En règle générale, aucune modification du module ne doit altérer son utilisation.

Un module qui remplit ses spécifications (A. Gourdin [1991]) doit être :

1. compatible ou composable
2. suffisamment général pour s'adapter à d'autre réalisations, c'est-à-dire être réutilisable
3. adaptable à de nouvelles spécifications, c'est-à-dire être flexible et extensible.

Avant de nous consacrer à la COO d'Archer, nous voulons faire quelques rappels essentiels de la POO.

III.2. But et concepts de la POO

Le but principal de la POO est de favoriser la réutilisabilité de l'existant. Donc la POO est avant tout une réponse à un problème économique de taille : il correspond à la réduction du coût du développement d'un logiciel et principalement celui de sa maintenance (40 à 75 % du coût A. Gourdin [1991]).

Elle est aussi une technique qui permet de programmer de moins en moins lors de la réalisation de nouvelles versions d'un produit.

Cette technique a pris deux directions qui convergent (H. Farreny et M. Ghallab [1990]). La première est exploitée par les branches de l'I.A. et a donné naissance à la notion des Frames et d'acteurs (G. Masini et al. [1989]). On trouve actuellement des générateurs de systèmes experts à base de Frames comme EMycin.

La deuxième est liée au développement des LOO. Le pionnier dans ce domaine est Simula (1965), suivi de Smalltalk (1976, 1980) qui est un interpréteur répondant vraiment à tous les concepts de la POO. D'autres comme C++, Turbo Pascal 6 et Turbo Pascal Windows sont apparus en 1990-1991. Les deux derniers ont ajouté une dimension d'objétisation à leur environnement classique.

"La POO est surtout un ensemble de techniques et on peut programmer plus ou moins par objet selon qu'on utilise partiellement ou totalement celle-ci" (B. Meyer [1990]). Dans cette optique, A. Gourdin [1991] a présenté les huit marches pour une bonne objétisation :

- les modules construits autour de structure de données
- les données modélisées par des types abstraits
- la généricité
- le module= type ou classe d'objets
- la gestion dynamique des objets
- l'héritage simple
- le polymorphisme
- l'héritage multiple
- la composition d'objets

III.3. Conception d'Archer

Archer répond dans sa globalité aux caractéristiques d'une COO : toute modification, intervenant à n'importe quel niveau de son processus, ne met pas en cause l'ensemble du projet.

Pour nous en rendre compte, nous allons nous servir des modules d'Archer, conçus selon une orientation objet (leur amélioration ou leur adaptation à un environnement particulier pourra se faire sans difficulté par un vrai LOO).(*)

---

* C'est ce que nous faisons actuellement pour la réalisation d'une version, sous windows, d'Archer.

Ces modules pourront être assimilés à des classes d'objets :

- **Interface utilisateur** : correspond à la gestion des objets "barre de menu", "menu déroulant", et "boîte de dialogue". Cette classe, initialement écrite en T.Prolog, est en cours de réalisation, pour des raisons d'esthétique, de convivialité et de standardisation, à partir de Turbo Pascal Windows.

- **Système physique** : correspond à un domaine appartenant au système dynamique étudié. Son module comporte toutes les opérations qui génèrent les modèles physique et bond-graph. Il est complètement réutilisable pour de nouveaux systèmes(*).

- **Affectation de la causalité**, formée par :
."conflit causal" : permet la création de pseudo-conflits et de localisation des compétitions.
."Liste de compétitions" : s'occupe de la résolution des compétitions causales, de l'application des règles dynamiques et de la détermination des représentants causaux.
. "Représentant causal" : se charge de la déduction de la causalité aux liens.

- **Liaison causale** : a pour rôle la détermination de l'existence d'un chemin causal et de ses informations.

- **Expression mathématique** : gère la manipulation des représentations symboliques des expressions mathématiques et le calcul formel des équations représentant le modèle.(*)

IV. Autres spécificités d'Archer

Dans cette section nous montrons certains aspects du projet Archer, ainsi que la résolution de quelques problèmes rencontrés pendant son développement.

IV.1. Base de données, mémoire et langages

Pendant la réalisation d'un projet, on doit analyser sérieusement les problèmes de la représentation et du stockage de l'information.

---

* voir chapitre 1.

** voir infra : calcul formel dans Archer, page.203

Tout au long des chapitres précédents, nous avons illustré quelques représentations des données d'Archer (*). Celles-ci sont formées à partir de structures récursives et favorisent la hiérarchisation par une sorte de mise en facteur des parties communes (voir par exemple au chapitre 2 la représentation des "cce", "ccd" et "cci"). Leur relation est réalisée à partir d'expressions clés.

Les données principales d'Archer sont écrites sous forme de prédicats de T.Prolog. Ce qui permet d'avoir des bases de données dites "intelligentes" ("logiques" ou "déductives" J. Kouloumdjian [1990]) qui combinent les informations dans leur état brut (les données) et celles représentées sous une forme travaillée (la connaissance).

T.Prolog associe à chaque programme une base de données dynamique résidant en mémoire vive (RAM) dont les avantages sont, entre autres : la possibilité d'avoir des champs de longueur variable, des clés d'accès sans limitation, une rapidité de recherche, ...

Mais Sa présence dans la RAM pendant son traitement constitue son inconvénient principal, donc, limite son utilisation à des programmes ne gérant pas de flux de données importants. Par exemple, son utilisation pour traiter un problème de gestion (compte bancaire, Achat-vente-stock, ...) est déconseillée.

T.Prolog permet dans ce cas une alternative qui consiste à créer, à partir d'outils prédéfinis, une base dynamique sur disque (mémoire de stockage). On peut ainsi définir des prédicats sur mesure selon les besoins du problème à traiter. D'une manière générale, il est préférable d'avoir ses propres outils de gestion de fichiers que d'utiliser les prédéfinis.

Cette alternative s'avère parfois fastidieuse à cause des accès directs au disque, d'autant plus que l'absence d'un fichier d'index entraînera, à chaque recherche, un balayage systématique du fichier de données.

Les fichiers d'index correspondent à une technique utilisée dans les systèmes de gestion de bases de données (SGBD) classiques, comme Dbase III+. Leur efficacité dans le cas de T.Prolog dépend de la présence dans le système d'une extension de mémoire (EMS).

Nous avons omis l'utilisation de la database dynamique sur disque, pour ne pas

---

* voir aussi annexe A

imposer une contrainte matérielle par rapport à une configuration standard d'un micro-ordinateur. Et puisque le nombre de données générées par Archer n'est jamais excessif, nous avons utilisé directement la database de T.Prolog. Celle-ci ne sera jamais saturée quelque soit la taille des modèles physiques, réels que l'on pourrait traiter sur micro-ordinateur.

Remarque :
Les procédures d'appel, d'insertion ou de suppression de prédicats sont définies, d'une manière simple, dans un module à part. Ainsi elles seront faciles à modifier et ne mettront pas en cause les programmes qui les utilisent.

Par exemple les fonctions prédéfinies "asserta", "assert", "assertz" de T.Prolog sont remplacées par :

Insérerd(X) :- asserta(X)	"insertion au début des données de type X"
Insérer (X) :- assert (X).	"insertion au milieu des données de type X"
Insérerf(X) :- assertz(X).	" insertion à la fin des données de type X"

La saturation de la RAM constitue un obstacle non négligeable pendant le traitement d'un système de taille relativement importante. Donc au lieu de chercher à grignoter les quelques centaines d'octets utilisés par la base dynamique active, nous avons opté pour la réduction de la complexité de nos programmes et pour la création de plusieurs points d'articulation dans le projet.

Cette articulation, conçue pour répondre à une COO, permet une bonne décomposition de la base de données. En effet, chaque module gère ses propres données et ne fait appel que rarement à des informations extérieures.

L'organisation d'Archer en modules indépendants peut aussi pallier l'inconvénient de T.Prolog qui consiste, à cause du manque de directives de compilation, à produire des fichiers exécutables de taille importante. De cette manière les fichiers exécutables dépasseront rarement les 100 Ko (80 Ko en moyenne). Ils permettront ainsi d'optimiser la place réservée aux traitements et aux générations des codes.

Le problème de gestion de la RAM est commun à tous les logiciels développés sous DOS. Et on est loin de la période où l'on ne savait quoi faire des 64 Ko disponibles sur les premiers micros. Actuellement on dispose de micros pouvant gérer entre 1 Mo et 1 Go en mode réel (manipulation directe des adresses) et entre 16 Mo et 4 Go en mode protégé (accès par une table d'adresses).

Cependant la fameuse barrière des 640 Ko n'est pas entièrement franchie. Ainsi, dès que les programmes réclament une taille importante de la RAM, pour tourner correctement, il devient nécessaire de gérer l'extension de mémoire.

Nous étions confronté à ce problème lors du développement, à partir T.Prolog, des interfaces utilisateurs en mode graphique(*). De plus deux programmes résident dans la RAM pendant une session d'Archer :

- Le DOS qui occupe un peu plus 100 Ko, problème résolu par les récentes versions de ce système (DOS 5 et Dr DOS).

- Le programme principal d'Archer qui s'occupe de la gestion de son environnement. Sa taille en mode graphique est deux fois plus grande que celle en mode texte.

L'arrivée sur le marché de windows 3 de Microsoft, ainsi que la décision prise par les grands constructeurs, à commencer par IBM, de faire de l'interface G.D.I (graphique Device Interface) (*) le standard de tous les futures machines, nous ont décidé à nous aligner dans cette optique. D'autant plus que Windows fournit un langage permettant de gérer les extensions de mémoire et la communication avec les périphériques quelques soient leur nature et leur type.

Nos premières tentatives de programmation sous cet environnement ont été réalisées à partir du langage C (Version C6) et SDK (Soft développement Kit). Cependant en constatant, A la sortie de la première version Turbo Pascal Windows, les possibilités offertes pour POO et l'indépendance par rapport à SDK (**) nous l'avons choisi pour développer Archer sous Windows.

A ce niveau, abordons une question qui se pose assez souvent en cours de réalisation d'un projet.

Doit-on forcément réécrire une partie d'un projet a l'arrivée d'un produit répondant a une technologie neuve ?

---

* Bien que cette opération soit totalement réalisable(voir affichage graphique de tous les résultats d'Archer par R. Bouayad [1991]) les fichiers générés sont de taille importante.

* Windows est écrit sous G.D.I.

** plus une certaine familiarisation avec les caractéristiques du Pascal.

En règle générale la réponse est non, d'ailleurs l'un des objectifs de la POO est la réutilisabilité de l'existant (et vue la prospérité annuelle du marché du logiciel, le problème de réalisation d'un projet risquerait de devenir non décidable). Cependant pour les trois critères suivants l'adaptation à un nouveau produit répondant au besoin et à la spécificité du projet nous paraissent judicieux :

- Standardisation : l'utilisation du logiciel doit suivre une norme spécifique à sa catégorie. Du point de vue commercial (car un logiciel est développé pour être utilisé) le coût de la formation doit être pris en compte.

- Résolution d'anomalies : possibilité que le produit apporte des solutions à des problèmes rencontrés par exemple au niveau de la communication avec les périphériques.

- Extensibilité rapide : le fameux principe "reculer pour mieux avancer". Donc réécrire une partie du projet pour pouvoir atteindre l'objectif fixé plus vite.

Tous ces critères ont été satisfaits pour le choix de Windows comme environnement principal d'Archer. On a évoqué plus haut le problème de standardisation, ainsi que les anomalies rencontrées vis-à-vis de la mémoire.

L'extensibilité concerne l'écriture de modèle physique en mode graphique à partir d'une gestion de "BitMap" (graphe par points) avec la possibilité d'un mélange d'éléments physiques et de blocs d'éléments bond-graphs. Ajouter à cela les sorties et l'utilisation sans effort de Windows ainsi que la possibilité de créer des versions multiples par rapport au même modèle physique (plusieurs versions de bond-graph causal, diverses entrées sorties donc différentes équations mathématiques et différentes analyses) ce qui permet de procéder à des comparaisons de performance d'une manière locale. Ceci est rendu possible par l'environnement MDI (Multiple Device Interface) de Windows.

Les autres points intéressants de Windows (*), sont :

- convivialité : tout est géré d'une manière graphique ;
- indépendance du matériel de l'application elle-même rendu possible par le GDI. Ce point nous permet de se consacrer plus à l'application qu'à la gestion de

* Windows est écrit en C selon une orientation objet totale. Pour plus de détails voir Microsoft [1990], M. Desmadril [1990], Et pour TPW voir Borland [1991].

l'environnement (souris, imprimante, clavier, écran, configuration de l'environnement: couleurs, police de caractères, taille ...) ;
- transfert dynamique des données entre les applications au moyen du D.D.E (Dynamic Data Exchange) ;
- simulation d'un système multitache : en effet Windows travaille sur un seul microprocesseur ;
- gestion de la mémoire

IV.2. Traitement en langage naturel

Archer fait du traitement en langage naturel pour communiquer avec l'utilisateur dans ses propres termes. Cela veut dire qu'il interprète les données de l'utilisateur (en occurrence la description du modèle physique) sous une forme symbolique représentant un bond-graph ; et inversement il s'adressera à lui ou affichera les résultats dans son propre langage.

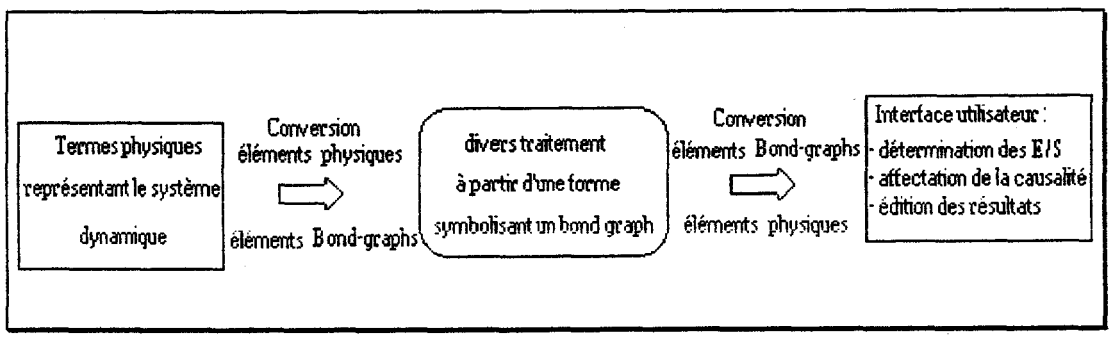


figure 5.8 : Principe de la communication entre Archer et l'utilisateur

La première conversion est basée sur une analyse par graphe d'état (voir chapitre 1). Dans la version Windows d'Archer deux modes sont possibles (texte et graphique) pour décrire le modèle. Dans l'éditeur graphique l'utilisateur ne dessine pas réellement son système, il se contente de donner des instructions à partir d'une bande qui contient une panoplie d'outils. Ceux-ci proposent des menus déroulants (pour passer d'un système à un autre), des boutons personnalisés (associés au symbole graphique des éléments physiques), des boutons alternés (pour série et parallèle), des Icones représentant la forme des points de connexions ainsi qu'un petit éditeur pour saisir le nombre de branches partant des points de connections, et un menu déroulant gardant la trace de tous ces points pour toute opération d'insertion, de suppression ...

Cette technique permettra de prendre en compte l'analyse syntaxique sans pour autant le programmer. Le fait que nous connaissons très bien à partir de l'automate à état fini le caractère général des systèmes physiques (ensemble de composants placés en série ou en parallèle les unes par rapport aux autres), nous dispense d'un traitement

supplémentaire.

Les autres avantages sont la rapidité de la description graphique, la possibilité de dessiner en même temps le modèle bond-graph (rendu possible par l'interface MDI de Windows), la nomination et numérotation automatique des éléments.

La deuxième conversion utilise une table qui sera créée par l'analyseur sémantique interprétant un modèle physique en un modèle bond-graph. Cette table fait correspondre à chaque élément physique son représentant bond-graph.

Le traitement d'un pseudo-langage naturel permet à Archer de ne pas s'adresser seulement aux bond-graphistes.

IV.3. Calcul formel

Archer est muni d'un petit générateur de calcul formel qui effectue pour le moment des opérations simples : addition, soustraction, multiplication, mise en facteur et simplification des expressions rationnelles constantes ou plynomialles.

Elle se base sur une représentation qui peut paraître compliquées, cependant elle favorise la récursivité, et facilite les manipulation des expressions mathématiques formelles. On peut ainsi déterminer les équations mathématiques du modèle étudié et simplifier leur affichage graphique sous une forme classique.

Son principe consiste à représenter un élément (bond-graph ou physique) par un triplet "(élément, son indice, son exposant)" de les trois paramètres sont de type chaîne de caractères. Exemple : pour une inductance d'ordre 4 "L_4 " on a ("L", "4", "1").

Une fraction est représentée par :

(signe, multiplicateur, [liste de triplets de type précédents associés au numérateur], [Idem pour un numérateur])

Exemple :

$2\,\frac{R_1}{I_1}\,\frac{1}{C_1}$ correspondra à ("+", 2, [(R,1,1)], [(I,1,1), (C,1,1)])

Une liste de fractions est représentée par une liste des quadruplets précédents.

En ce qui nous concerne les matrices d'état et les fonctions de transferts ont la forme suivantes :

(M, i, j, [liste de fractions représentant le numérateur], [Idem pour le dénominateur])

Avec pour l'équation d'état M="a", "b", "c", ou "d", et i et j sont les indices -ligne et colonne- de la composante de la matrice considérée.

Pour les fonctions de transfert M="a" pour indiquer un coefficient du dénominateur, et M="b" pour celui d'un numérateur. L'indice "i" représente le numéro de la fonction de transfert (facultatif pour "a" car le dénominateur est le même pour toutes les fonctions) et "j" indique l'ordre par rapport à l'opérateur de la place "s".

exemple : $b_2 s^{-2} = \frac{R_1}{I_1 s} \frac{1}{C_1 s}$ correspondra à

("b", _,2, [("+", 2, [(R,1,1)], [(I,1,1), (C,1,1)]], [ ])

Cette dernière représentation permettra de traiter des polynômes.

IV.5. Fichiers d'aide et configuration d'Archer

Archer possède un fichier d'aide pour faciliter l'utilisation de son environnement et apporter un nombre d'explication par rapport à ces programmes ; un didacticiel bond-graph rappelant leur principe et donnant diverses définitions ; un fichier d'analyse enregistrant tous les problèmes rencontrés et les traces de certains résolution et un fichier de configuration que la figure 5.9 retrace ces particularité.

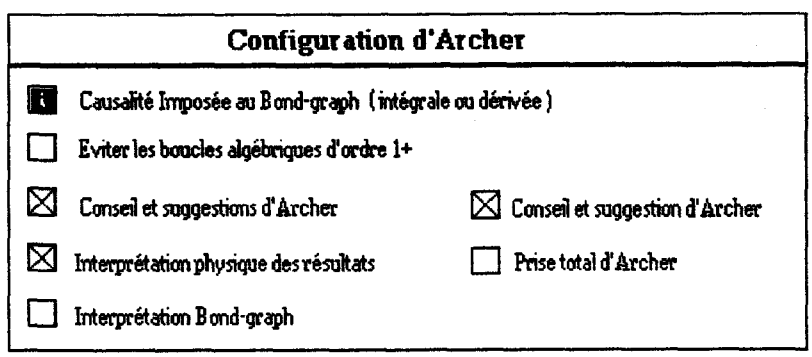


figure 5.9 : configuration par défaut dans Archer.

Nous entendons par configuration celle qui correspond au choix de l'utilisateur pour la manière de l'affectation de la causalité, de l'affichage des résultats, et de l'analyse.

V. Inventaire et perspectives d'Archer

A l'heure actuelle, on peut estimer qu'Archer a atteint une certaine maturité(*), ses bases sont devenues solides à la fois du point de vue informatique et bond-graph. Il est entré dans une phase de tests et de suggestions dans le but d'enrichir sa connaissance générale à partir de celle des spécialistes des systèmes physiques .

Sa conception, dans sa dernière version -celle que nous avons retracée tout au long de ce mémoire- lui permet d'être un produit informatique de son époque. Celle-ci a été soigneusement étudiée pour lui permettre d'une part d'être en harmonie avec la méthodologie sur laquelle il s'est basé pour atteindre ses objectifs et d'autre part, de répondre aux normes actuelles des produits de la micro-informatique.

Nous allons mentionner l'état d'Archer, à savoir les parties existantes, les programmes en cours de développement et les extensions prévues.

Le travail déjà réalisé a été axé sur :

La description du modèle physique en mode texte.
Les analyseurs syntaxique et sémantique.
La simplification des modèles.
L'affectation de la causalité avec ses interfaces utilisateur.
La détermination des informations causales.
La détermination des entrées-sorties à partir d'un interface utilisateur.
La détermination de la matrice de structure.
La détermination de la matrice de transfert pour les systèmes linéaires.
La détermination de l'équation d'état pour les systèmes linéaires ou partiellement non linéaires.
La vérification de certaines propriétés structurelles.
L'affichage graphique des bond-graphs et des équations.

Quant aux programmes de gestion des interfaces utilisateur, sous son nouveau environnement Windows, ils sont presque terminés, il s'agit de :

La description du modèle en mode graphique sous Windows .
L'édition des données (dessins du bond-graph, équations) sur les périphériques de sortie.

---

* Les modules de base de la première version (conversion d'un modèle physique en modèle bond-graph et l'affectation de la causalité) L. Coutereel et al. [1988], L. Coutereel [1991] ne répondaient pas à toutes les spécificités d'Archer et ont été mis en cause par sa nouvelle orientation.

Les interfaces utilisateur des programmes de la détermination des entrées-sorties et d'affectation de la causalité. Cependant on peut utiliser sans problème ceux développés avec Turbo Prolog en mode texte.

D'autre part, pour compléter Archer, nous avons entamé le développement de certaines parties concernant :

La conversion des données formelles en un langage accessible à chacun des logiciels de simulation suivants : ACSL, TUTSIM.

Le couplage entre d'une part, les bond-graphs et d'autre part, le modèle mathématique (fonction de transfert, équation de second ordre), le modèle physique ou le modèle bond-graph afin de pouvoir décrire et traiter des modèles dynamiques très complexes dont on connaît pas de manière précise la nature de certains composants.

Le développement d'un générateur de moteur d'inférence pour toutes les parties qui fonctionnent comme un système expert. Nous n'avons pas donné la priorité à cette phase parce que d'abord, nous avions des problèmes capitaux à résoudre et en plus, nous étions limité par le nombre de règle en notre possession.

Nous commençons cependant à cerner le problème et nous savons à l'heure actuelle comment cette connaissance peut être représentée(*). Aussi avons-nous commencer le développement d'un gestionnaire de connaissances sous forme d'un noyau vide (près à accepter d'autres règles).

D'une manière générale, on peut dire que la partie modélisation est terminée et que la partie analyse, qui est l'une des originalités d'Archer, est en plein expansion.

En outre, nous prévoyons l'extension d'Acher à travers :

La détermination du modèle mathématique sous forme de système d'équations différentielles d'ordre 2.

L'implantation des méthodes de réduction de modèles, notemment par les perturbations singulières (C. Sieur et G. Dauphin-Tanguy [1991.b])

---

*Ceci est rendu possible par les recherches effectués au sein du laboratoire pour formaliser la connaissance, à savoir les méthodes de l'analyse et de couplage entre divers domaines et diverses présentations du modèle physique.

La détermination des informations concernant les domaines de variation des dynamiques du système et les variables correspondants.

L'adaptation des données à d'autres logiciels de simulation comme BASIL, MATLAB...

La description des modèles physique par blocs et la gestion d'une bibliothèque de modèles, surtout pour les éléments physiques dont la décomposition exacte est difficile malgré qu'on ait déterminé leur bond-graph associé.

L'augmentation du nombre de domaines pouvant former un système dynamique.

VI. Réflexion sur une synergie entre l'I.A. et la théorie du Bond-Graph

Cette section n'est qu'une introduction succinte pour exprimer le pont qui commence à s'établir entre la méthodologie bond-graph et un domaine de l'I.A consacrée à l'application des principes de la physique qualitative au raisonnement du même nom(*). Cette idée introduite dernièrement par J.L Top et al.[1991], montre comment appliquer l'I.A au bond-graph et vice-versa.

Ainsi les techniques I.A en général et le raisonnement qualitatif en particulier peuvent aider aux traitements et à l'implantation sur calculateur des méthodes bond-graph (essentiellement pour l'analyse de la base des connaissances associée au système physique). Inversement utiliser les bond-graphs comme une forme de représentation de la connaissance pour raisonner sur l'aspect dynamique d'un problème permet d'obtenir des informations qualitatives d'un système.

D'une manière générale il existe des similitudes entres les tâches de la physique qualitative et la méthodologie bond-graph. Ainsi la première commence par identifier les variables et les paramètres d'un domaine (analogie avec la phase des analyses syntaxique et sémantique pour arriver à exprimer le bond-graph),puis repéré les

---

* Le raisonnement qualitatif est devenu ces derniers années l'un des champs les plus actifs l'I.A. Les chercheurs qui y travaillent se sont tournés vers la physique qualitative et ont pu introduire des notions comme le calcul différentiel qualitatif (dérivation par rapport au temps et équations portant sur les signes des variables [Chatain J.N. 1990].

relations causales entre les variables et paramètres (affectation de la causalité),enfin décrit le comportement du système sous forme de caractéristiques qualitatives (analyse structurelle du bond-graph). La dernière tâche consiste à prévoir l'amplitude des variations du système (utilisation des équations mathématiques sans avoir recours à la simulation numérique).

On trouve dans Iwasaki Y. [1989] une étude détaillée sur le raisonnement qualitatif et sur son rapport avec la physique qualitative. Une bonne partie est consacrée à la notion de causalité. Cette information qualitative constitue le noyau idéal pour la synergie des deux disciplines I.A et Bond-Graph.

Dans un rapport de l'Observatoire Français des Techniques Avancées (OFTA [1989]) concernant le développement des systèmes experts pour la conduite d'un processus de fabrication, on suggère d'associer au système expert un "graphe causal". Ce dernier caractérise la modélisation qualitative du comportement du système traité et permet de faire de l'analyse prédictive. La définition donnée du graphe causal correspond à celle plus générale du bond-graph.

Il est clair que ce sujet ne peut être développé ici. Nous pensons orienter nos travaux futures dans cette voie.

CONCLUSION

Nous pouvons qualifier ce chapitre de cahier des charges associées à l'analyse des problèmes algorithmiques et techniques susceptibles d'exister lors de l'élaboration d'un projet, et à l'étude des méthodes de résolution apportées par le projet Archer.

Archer est un produit informatique qui possède le profil et les caractéristiques des logiciel modernes : convivialité, réponse en temps réel, traitement en langage naturel...

C'est aussi un gestionnaire de connaissances, car il se base sur le savoir-faire des bonds graphistes et sur celui des physiciens. La connaissance peut se symboliser sous forme de règles et de faits dont une bonne partie est entièrement modélisée. l'autre, elle peut engendrer des situations non déterministes et des problèmes décisionnels. Pour remédier à cela nous avons fait appel aux techniques I.A. L'intérêt de cette approche réside dans l'étude de la représentation des connaissances en entités symbolisant le

modèle réel des problèmes traités. Elle détermine également une stratégie de recherche pour arriver à des programmes admissibles, décidables et ayant une complexité acceptable.

D'autre part, le choix de la Conception Orientée Objet permet à la modulation de se faire sans complication. Ce qui facilite l'extensibilité d'Archer.

Archer entame actuellement une phase de collaboration avec des experts des sciences physiques afin d'augmenter ses connaissance générales l'aidant à prendre des bonnes résolutions.

Conclusion générale

Conclusion générale

Les bond-graphs constituent un outil de modélisation des systèmes dynamiques plus déterminant que les méthodes classiques (graphiques, mathématiques) puisqu'ils représentent à la fois la structure d'un système et les relations de cause à effet entre ses objets.

Afin de pouvoir déterminer toutes les équations mathématiques modélisant le système dynamique étudié et d'analyser sa structure, nous avons procédé à des manipulations topologiques que nous pouvons qualifier d'analyses qualitatives. En effet, celles-ci nous ont aidé dans un premier temps de trouver une méthode d'affectation de la causalité provocant, en fonction de l'architecture du bond-graph, des conflits causaux qui correspondent à des situations non déterministes. Leur résolution se fait par les techniques de l'intelligence artificielle.

La détermination des informations causales (Azmani A. et al. [1991]) se fait à partir d'une représentation symbolique sous forme d'arbres construits par l'intermédiaire de vecteurs d'incidence extraits du modèle bond-graph. L'exploration et l'analyse de ces arbres conduisent à la détermination de la base de connaissances nécessaire à toutes les méthodes basées sur la théorie des bond-graphs.

D'autre part, la méthode que nous avons proposée pour déterminer l'équation d'état sous forme d'une expression formelle est entièrement basée sur l'analyse qualitative des caractéristiques causales entre les éléments d'un bond-graph. C'est une méthode graphique, elle traite séparément les connexions causales des éléments dissipatifs d'énergie, celles des éléments de stockage d'énergie indépendants et dépendants ainsi que la partie associée à la liaison entre les éléments dynamiques.

L'implantation de ces méthodes sur calculateur se fait sans difficultés grâce aux techniques de l'I.A. Elle constitue l'une des originalités d'Archer (Azmani A. et G. Dauphin-Tanguy [1992]).

La conception, la mise en place et la conduite d'un projet comme Archer demande la collaboration de plusieurs personnes. En ce qui nous concerne, nous nous sommes occupés principalement de la conception et de la réalisation des fondements de ce logiciel, l'automate de simulation du processus suivi par Archer lors d'une session (*) donnera une idée des tâches qui ont été notres. Tout au long de ce mémoire, nous

* Voir chapitre 1

avons montré les difficultés que nous avons rencontrées et les méthodes de résolution que nous avons adoptées.

La conception orientée objet d'Archer facilite son extensibilité et plusieurs méthodes peuvent être implantées par simples manipulation des données produites par les modules existants d'autre part (*).

Notamment les modèles mathématiques déduits du modèle bond-graph sont des expressions formelles qui peuvent, sans grandes difficulté, être adaptées pour tout logiciel de simulation existant.

Avant de clore ce mémoire, nous voudrions exprimer l'intérêt qu'a suscité chez nous la synergie entre l'I.A. et les bond-graphs. Cette voie nous ouvre bien des perspectives...

** Voir inventaire et perspectives d'Archer, chapitre 4.

REFERENCES BIBLIOGRAPHIQUES

Références bibliographiques

AZMANI A., DAUPHIN-TANGUY G. [A PARAITRE EN 1992]
"ARCHER : a processor for computer aided modelling"
accépté pour publication dans le volume spécial IMACS Transactions
Elsevier

AZMANI A.,BOUAYAD R., DAUPHIN-TANGUY G. [1991]
"Artificial Intelligence approach for the causal analysis of bond-graph models"
Mathematical and intelligent models in system simulation
R. Hanus, P. Kool, S. Tzafestas (editors)
J.C. Baltzer AG, Scientific Publishing Co. IMACS, [1991]
pp. 319-324

BORNE P. DAUPHIN-TANGUY G. ROTTELA F. ZAMBETTAKIS I. [1991-92]
"Modélisation et identification des processus, tome 1, tome 2"
Editions Technip

Bouayad R. [1991]
"Conception et réalisation d'un processeur de tracé automatique du modèle bond-graph par le logiciel Archer"
Thèse, Université des Sciences et Techniques de Lille (FRANCE)

BOUAYAD R. AZMANI A. DAUPHIN-TANGUY G. (1991)
"Algorithm for the computer aided drawing of bond-graph models"
Mathematical and intelligent models in system simulation
R. Hanus, P. Kool, S. Tzafestas (editors)
J.C. Baltzer AG, Scientific Publishing Co. IMACS
pp. 201-205

BREEDVELD P. C. [1984]
"Essantial Gyrators and Equivalence Rules for 3-ports Junctions Structures"
Journal of Franklin Institut, vol. 318, N°2, pp. 77-89

BREEDVELD P. C., ROSENBERG R. C., ZHOU T. [1991]
"Bond Graph Bibliography"
Journal of the Franklin Institute vol. 328, N°5/6, pp. 1067-1109
Pergamon Press

BROENINK J. F. [1990]
"Computer-aided physical-systems modelling and simulation : a bond graph approach"
Ph.D. Thesis, Electrical Engineering, University of Twente, Enschede (Pays-Bas)

BROWN F. T. [1972]
"Direct Application Loop Rule to Bond-graph"
Jounal of Dynamic Syst., Measurement and Control, pp.253-261, Sept. 1972.

BUCKELY F., HARARY F. [1990]
"Distance in Graphs"
Addison-Wesley

CASTELAIN A., DUCREUX J.P., DAUPHIN-TANGUY G., ROMBAUT C. [1990]
"Modelling and Analysis of Power Electronic Networks by Bond Graph"
Symposium IMACS, Nancy, 19-21 sept. 90 (FRANCE)

CHATAIN J-N. [1990]
"Le raisonnement approximatif dans les systèmes experts en maintenance."
Modèles logiques et systèmes d'intlligence artificielle, chap. 17, pp. 273-290
Coordonnateurs : L. Iturrioz et A. Duchaussoy.
Hermès

COUTEREEL L. BOUAYAD R. DAUPHIN-TANGUY G. [1989]
"Artificial Intelligence and Bond-Graph methodology for the modelling of dynamical systems"
Modelling and simulation of systems
P.C. Breedveld et al. (editors)
J.C. Baltzer AG, Scientific Publishing Co. IMACS pp. 21-23.

DAUPHIN-TANGUY G., LEBRUN M., BORNE P. [1983]
"Interprétation de la notion de système réciproque par les bond graphs pour processus multi-échelle de temps"
Proc. AESTED AI'83, pp. 137-144. (FRANCE)

DAUPHIN-TANGUY G., LEBRUN M., BORNE P. [1985]
"Order reduction of multi-time scale systems using bond graphs, reciprocal system and singular perturbation method"
Journal of the Franklin Institute, Vol. 319, N°1/2, pp. 157-171

DAUPHIN-TANGUY G. SUEUR C. COUTEREEL L. [1987]
"Presentation of a processor for the computer aided modelling of dynamicals systems through a Bond-Graph approach"
IMACS-Intenational Symposium on "Artificial Intelligence, Expert Systems and Language in Modelling and Simulation" Barcelone (Espagne)
pp. 197-202

DELAHAYE J.P. [1987]
"Outils logiques pour l'Intelligence Artificielle"
Eyrolles

DELAHAYE J.P. [1987]
"SYSTEMES EXPERTS : organisation et programmation des bases de connaissance en calcul propositionnel"
Eyrolles

DESMADRIL M. [1990]
"La programmation sous Windows"
Eyrolles

DELGADO DE NIETO M.[1991]
"Description et simulation de systèmes linéaires représentés par le bond-graph"
Thèse de Doctorat, Université de Rennes (France)

FARRENY H., GHALLAB M. [1987]
"Eléments d'intelligence artificielle"
Hermès

FROT P. [1987]
"Trois systèmes expert en Tubo Pascal"
Sybex

GENTHON P. [1989]
"Dictionnaire de l'intelligence artificielle"
Hermès.

GOURDIN A. [1991]
"Programmation Objet en C"
Informatique et langages

GRANDA J. J. [1982]
"Computer aided modelling program (CAMP) : a bond graph preprocessor for computer-aided design and simulation of physical systems using digital simulation languages"
Ph.D. Thesis, Mechanical Enginneering, University of California

GRIFFITHS M. [1986]
"Techniques algorithmiques pour l'intelligence artificielle"
Hermès.

Hood S. J., Palmer E. R., Dantzig P. M. [1989]
"A fast, complete method for automatically assigning causality to bond graph "
J. Franklin Inst., Vol.326, No. 1, pp. 83-92, 1989.

Iwasaki Y. [1989]
"Qualitative Physics"
The HandBook of Artificial Intelligence, Vol. IV, Chap. XXI, pp. 323-413

Joseph B.J., Martens H.R. [1974]
"The Method of Relaxed Causality in The Bond Graph Analysis of NonLinear Systems"
Trans. ASME, J. Dynamic Syst.Measure.Control, Vol. 96, N° 1, pp. 95-99, 1974.

KARNOP D.C. ROSENBERG R.C. [1975]
"Systems dynamics : an unified approach"
Wiley and Sons, New-York

KARNOPP D., ROSENBERG R.C. [1983]
"Introduction to Physical System Dynamics"
McGraw-Hill

KAYSER D. [1990]
"La représentation des connaissance"
Modèles logiques et systèmes d'intlligence artificielle, chap. 2, pp. 59-74
Coordonnateurs : L. Iturrioz et A. Duchaussoy.
Hermès.

KOULOUMDJIAN J. [1990]
"Bases de données déductives"
Modèles logiques et systèmes d'intlligence artificielle, chap. 15, pp. 314-321
Coordonnateurs : L. Iturrioz et A. Duchaussoy.
Hermès.

LINDSAY R., BUCHANAN B., FEIGENBAUM E., LEDERBERG J. [1980]
"The Dendral Project"
McGraw Hill

LORENS C.S. [1964]
"For the Modeling and Analysis of Linear Systems"
McGraw-Hill New York

MASINI G., NAPOLI A., COLNET D., LEONARD D., TOMBRE K. [1989]
"Les langages à objet, langages de classe, langages de frames, langages d'acteurs"
Interéditions.

MASON S.J. [1956]
"Feedback Theoty - Further Properties of Signal"
Proceeding I.R.E. Vol 44, N°7, July 1956, pp 920-926

MEYER B. [1990]
"Conception et programmation par objet"
Interéditions.

MINOUX M., GOURDON M. [1979]
"Graphes et algorithmes"
Collection de la Direction des Etudes et Recherches d'Eléctricité de France
Eyrolles

MONTBRUN-DI FILLIPO J., DELGADO M., BRIE C., PAYNTER H. M. [1991]
"A Survey of Bond Graphs : Theory, Applications and Programs"
Journal of the Franklin Institute

OBSERVTOIRE FRANÇAIS DES TECHNIQUES AVANCEES [1989]
"ARAGO 8 : Systèmes Experts et Conduite de Processus"
Masson Paris

ORT J.R. MARTENS M.R. [1974]
"A topological procedure for converting a bond-graph to a linear graph"
J. Dynamical Systems, Measurement and Control, Sept. 74, pp. 307-314

PAYNTER, M.M. [1961]
"Analysis and Design of Engineering Systems"
M.I.T. Press, Cambridge (Mass.)

PERELSON A. OSTER G.F. [1976]
"Bond Graphs and Linear Graphs"
J. Franklin Institute, vol. 302, n°2, pp. 159-185

REMY P. [1990]
"Elaboration d'un module de couplage entre les modèles bond-graphs pour le processeur ARCHER"
DEA de Productique, Université des sciences et techniques de Lille Flandres Artois

Ringot D. [1990]
"Analyseur syntaxique et sémantique d'un langage utilisateur pour la construction des bond-graph"
mémoire de D.E.A Université des Sciences et Techniques de Lille

Robichaud L.P.A., Boisvert M., Robert J. [1962]
"Signal Flow Graph and Applications"
Prentice-Hall, Englewood Cliffs N.J.

Rosenberg R. C. [1971]
"State Space Formulation of Bond Graph Models of Multiport Systems"
Trans. ASME J. Dyn. Syst. Meas. Control, Vol. 93, N°. 1, pp. 35-40

Rosenberg R. C. [1974]
"User's Guide to ENPORT-4"
Wiley, New York

Rosenberg R. C., Andry A.N. [1979]
"Solvability of Bond Graph Junction Structures with Loops"
IEEE Trans. on Circuits and Systems, Vol. CAS-26, n° 2, pp. 130-137.

Rosenberg R. C. [1987]
"Exploiting Bond Graph Causality in Physical System Models"
Trans. of the ASME, Journ. of Dyn. Syst., Meas. and Contr., vol. 109, pp. 378-383.

Suda N., Hatanaka T. [1986]
"Structural Properties of SystemsRepresented by Bond-Graph"IMACS, Complex and Dist. Syst. Analysis, Sim. and Cont., pp. 73-80.

Sueur C., et Dauphin-Tanguy G. [1989]
"Structural Contrallability/Observability of Linear Systems Represented by Bond-Graph"
Journal of the Franklin Institute, vol. 326, n° 6, pp. 869-883.

Sueur C. [1990]
"Contribution à la modélisation et à l'analyse des systèmes dynamiques par une approche bond-graph : application aux systèmes polyarticulés, plan à segments flexibles"
Thèse, Université des Sciences et Techniques de Lille (FRANCE)

Sueur C., et Dauphin-Tanguy G. [1991]
"Bond-Graph Approach for Structural Analysis of MIMO Linear Systems"
Journal of the Franklin Institute, vol. 328, N°1, pp. 55-70

Sueur C., et Dauphin-Tanguy G. [1991]
"Bond Graph Approach to Multi-time Scale Systems Analysis"
Journal of the Franklin Institute vol. 328, N°5/6, pp. 1005-1026
Pergamon Press

The Handbook of A.I. [1983, 1985, 1986, 1989]
Volumes I, II, III et IV.
Addison-Wesley

Top J.L., Akkermans J.M., Breedveld P.C. [1991]
"Qualitative Reasoning about Physical Systems : an Artificiel Intelligence Perspective"
Journal of the Franklin Institute vol. 328, N°5/6, pp. 1047-1065
Pergamon Press

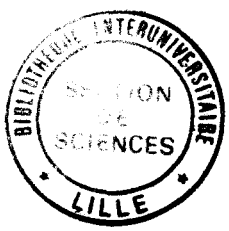
Towsend C. [1987]
"Mastering Expert Systems with Turbo Prolog"
Howard W. Sams & Company

Towsend C. [1988]
"Turbo Prolog applications"
Sybex

Tutsin A. [1952]
"Direct Current Machines for Control Systems"
The MacMillan Company, New York, pp. 45-46, 1952

Van Dijk J., Breedveld [1991]
"Simulations of System Models Containing Zero Order Paths
I. Classification of Zero-Order Causal Path"
Journal of the Franklin Institute vol. 328, N°5/6, pp. 959-979
Pergamon Press

Wlasowski M., Lorenz F. [1991]
"How to Determine the Solvability of Bond Graph Linear Junction Structures"
Journal of the Franklin Institute vol. 328, N°5/6, pp. 855-870
Pergamon Press



UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

THESE DE : *DOCTORAT (nouvelle Thèse)*
DISCIPLINE : *PRODUCTIQUE*

N° D'ORDRE :

NOM DU CANDIDAT : *Abdellah AZMANI*

JURY

Président :	P. BORNE	*Professeur (EC LILLE - Villeneuve d'Ascq)*
Rapporteurs :	J.P. DELAHAYE	*Professeur (UFR d'I.E.E.A. - Villeneuve d'Ascq)*
	S. SCAVARDA	*Professeur (I.N.S.A. - Villeurbanne)*
Examinateurs :	M. BENREJEB	*Professeur (E.N.I.T. - Tunis / Tunisie)*
	J.P. CASSAR	*Maître de Conférences (I.U.T. A - Villeneuve d'Ascq)*
	G. DAUPHIN-TANGUY	*Professeur (EC LILLE - Villeneuve d'Ascq)*

TITRE DE LA THESE :

Analyse qualitative d'un Bond-Graph par les techniques de l'Intelligence Artificielle.
Contribution à la conception et à la réalisation du logiciel d'aide à la modélisation ARCHER.

RESUME

Ce mémoire propose diverses méthodes originales basées sur l'analyse qualitative d'un bond-graph.

La première est consacrée à l'affectation automatique de la causalité. Elle présente une grande convivialité et une souplesse d'utilisation, et prend en compte les difficultés classiquement rencontrées dans un bond-graph (causalités dérivées, boucles algébriques). Elle se prête particulièrement à l'analyse des propriétés structurelles du modèle. Des algorithmes sont proposés dans le but de déterminer les informations causales nécessaires à l'exploitation de la méthodologie bond-graph.

Une autre méthode concerne la détermination de l'équation d'état sous forme d'expressions formelles. Elle est intitulée "Méthode des Boucles et Chemins Causaux" (MBCC). Elle se fonde uniquement sur le parcours et l'analyse des chemins causaux et des boucles causales. La MBCC présente l'originalité de traiter à la fois les boucles algébriques entre des éléments dissipatifs et la causalité dérivée. Cette méthode est applicable pour certaines classes de systèmes non linéaires et pour des éléments multiports.

Ces différentes méthodes ont contribué à la réalisation d'un logiciel d'aide à la modélisation des systèmes dynamiques : ARCHER. Sa philosophie ainsi que sa spécificité sont exposées et détaillées. ARCHER est réalisé selon les concepts de la programmation orientée objet et utilise les techniques de l'Intelligence Artificielle.

Soutenance prévue le : **19 Décembre 1991** à **10 heures 30**

Bâtiment B - ECOLE CENTRALE DE LILLE **Amphithéâtre BODA**