Synthèse algébrique de lois de commande pour les systèmes à évènements discrets logiques

Yann Hietter

N° ENSC - 2009-155

THÈSE DE DOCTORAT
DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Présentée par

Monsieur Yann HIETTER

pour obtenir le grade de

DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Domaine :
ÉLECTRONIQUE-ÉLECTROTECHNIQUE-AUTOMATIQUE

Sujet de la thèse :

Synthèse algébrique de la loi de commande d'un système à évènements discrets logique.

Thèse présentée et soutenue à Cachan le 28 mai 2009 devant le jury composé de :

Jean-Louis FERRIER	Professeur des universités-Angers	Président
Janan ZAYTOON	Professeur des universités-Reims	Rapporteur
Jean-François PETIN	Maître de conférence HDR-UHP	Rapporteur
Étienne CRAYE	Professeur des universités-Centrale Lille	Examinateur
Jean-Jacques LESAGE	Professeur des universités-ENS Cachan	Directeur de thèse
Jean-Marc ROUSSEL	Maître de conférence-ENS Cachan	Encadrant

Laboratoire Universitaire de Recherche en Production Automatisée
(ENS CACHAN / Université Paris-Sud 11 / EA 1385)
61, avenue du Président Wilson, 94235 CACHAN CEDEX (France)

Remerciements

Le travail de recherche exposé dans ce mémoire de thèse a été réalisé au sein du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA - EA 1385) de l'École Normale Supérieure de Cachan.

Je tiens tout d'abord à remercier le Professeur Jean-Jacques Lesage pour avoir assumé la direction de mes travaux. Je remercie également Monsieur Jean-Marc Roussel pour avoir assuré l'encadrement scientifique de mon travail.

Je remercie le Professeur Janan Zaytoon et Monsieur Jean-François Petin qui ont accepté d'être rapporteurs de cette thèse. Je remercie également les Professeurs Étienne Craye et Jean-Louis Ferrier pour avoir participé au jury.

Merci à tous les membres du LURPA qui m'ont côtoyé et aidés durant ces trois années. Je remercie en particulier mes collègues de bureau Silvain, Labib et Antonio qui m'ont supporté plus que les autres, les chercheurs permanents comme Bruno, Olivier et Saïd pour leurs bonnes idées et leur bonne humeur, les non-chercheurs dont Marc, Françoise, Karina et Nathalie pour leur disponibilité et l'aide administrative et informatique qu'ils m'ont apportée et enfin les autres doctorants dont Zoulema, Vincent, Steve, Mathieu et Guillaume avec qui le travail en équipe a permis de nombreuses victoires surtout le samedi.

Je remercie enfin tous les membres de ma famille et surtout mes parents qui m'ont soutenu sans faille depuis 28 ans et qui ont contribué à faire de moi l'homme que je suis devenu.

Table des matières

Notations

1. Notations des algèbres de Boole

1.1. Cas d'une algèbre générale

- $\mathcal{B}$ est l'ensemble des éléments de l'algèbre.
- $+$, $\cdot$ et $\bar{}$ sont les trois opérations sur l'ensemble $\mathcal{B}$.
- 0 et 1 sont les deux éléments neutres de l'ensemble $\mathcal{B}$.
- $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ représente une algèbre de Boole.
- x, y, z, w représentent quatre éléments quelconques de $\mathcal{B}$.
- α_i représente un élément quelconque de l'ensemble $\mathcal{B}$ de l'algèbre.
- $C(\alpha_1, \ldots, \alpha_n)$ représente une composition des éléments $(\alpha_1, \ldots, \alpha_n)$.
- $x = y$ représente l'égalité entre deux éléments.
- $x \leq y$ représente la relation d'Inclusion entre deux éléments.
- $(\ )$; $\bar{}$; $\cdot$; $+$ est l'ordre de priorité entre les parenthèses et les opérations de l'algèbre.

1.2. Cas de l'algèbre des variables booléennes

- $\mathbb{B} = \{0, 1\}$ est l'ensemble des valeurs booléennes.
- $\vee$, $\wedge$ et $\neg$ sont les trois opérations sur les variables booléennes.
- 0 et 1 sont les deux éléments neutres : faux et vrai.
- $(\mathbb{B}, \vee, \wedge, \neg, 0, 1)$ est l'algèbre de Boole des variables booléennes.
- b_i (boolean variable) représente une variable booléenne quelconque.
- $(\ )$; $\neg$; $\wedge$; $\vee$ est l'ordre de priorité entre les parenthèses et les opérations sur les variables booléennes.

1.3. Cas de l'algèbre des fonctions booléennes

- $\Gamma = \{f : \mathbb{B}^n \to \mathbb{B}\}$ est l'ensemble des fonctions booléennes de dimension n.
- $+, \cdot$ et $\bar{}$ sont les trois opérations sur les fonctions booléennes.
- $\mathcal{F}$ (False) et $\mathcal{T}$ (True) sont les deux fonctions constantes toujours fausse et toujours vraie.
- $(\Gamma, +, \cdot, \bar{}, \mathcal{F}, \mathcal{T})$ est l'algèbre de Boole des fonctions booléennes.
- G_i représente une fonction booléenne quelconque.
- $G_1 = G_2$ est le symbole de la relation d'égalité entre deux fonctions booléennes.
- $G_1 \leq G_2$ est le symbole de la relation d'inclusion entre deux fonctions booléennes.
- $(\ ) \ ; \bar{} \ ; \cdot \ ; +$ est l'ordre de priorité entre les parenthèses et les opérations sur les fonctions booléennes.

2. Résolution d'équation dans une algèbre de Boole

- $C(x_1, \ldots, x_n) = 0$ est le type d'équation à résoudre.
- n est le nombre d'inconnues.
- ω_j représente un n-uplet d'éléments de $\{0, 1\}$ et représente en binaire l'indice j du n-uplet.
- Ω_n représente l'ensemble des ω_j.
- ω_j^l représente la $l^{\text{ème}}$ fonction du n-uplet ω_j.
- p_i (parameter) est un élément quelconque jouant le rôle de paramètre pour exprimer l'espace solution.

3. Notations propres aux systèmes logiques

- e_i représente une entrée du système. Il y a m entrées.
- s_i représente une sortie du système. Il y a n sorties.
- x_i représente une variable d'état interne du système. Il y a l variables d'état interne.
- $e_i[k], s_i[k], x_i[k]$ représentent les différentes variables à l'instant k.
- $e_i[k\text{-}1], s_i[k\text{-}1], x_i[k\text{-}1]$ représentent les différentes variables à l'instant (k-1).
- $e_i[0], s_i[0], x_i[0]$ représentent les différentes variables à l'instant initial.

Introduction

Ce mémoire présente un élément de réponse à l'une des questions essentielles auxquelles sont confrontés les concepteurs de systèmes de contrôle/commande : comment garantir que la commande qui est en cours de développement sera en tous points conforme aux exigences exprimées dans le cahier des charges ?

Notre étude porte exclusivement sur le contrôle/commande des Systèmes à Evénements Discrets (SED) dont les entrées/sorties sont des variables logiques. Pour ces systèmes, nous nous proposons d'obtenir la loi de commande recherchée en résolvant automatiquement et de manière littérale le système d'équations représentant les exigences données dans le cahier des charges.

Obtenir de manière formelle la loi de commande d'un SED logique à partir des exigences est un véritable défi scientifique porteur d'enjeux industriels importants. Les contrôleurs logiques sont en effet utilisés dans un grand nombre de systèmes critiques, comme les systèmes de transport ou les systèmes de production d'énergie. Garantir que la commande de ces systèmes est en adéquation avec le cahier des charges est une préoccupation majeure des concepteurs de ces commandes. Ils sont donc en attente de méthodes et d'outils leur permettant, soit de vérifier *a posteriori* que la commande qu'ils proposent satisfait le cahier des charges, soit d'obtenir directement cette commande à partir du cahier des charges. Les travaux présentés dans ce mémoire de thèse relèvent de cette seconde approche et proposent une méthode pour la synthèse automatique des lois de commande.

Au sein de la communauté des SED, nos travaux s'inscrivent dans la catégorie des approches de synthèse. La plus connue de ces méthodes est la Supervisory Control Theory (SCT) dont les concepts fondateurs ont été posés par Ramadge et Wonham. Comme la SCT, notre méthode de synthèse repose sur un cadre mathématique rigoureux. Le cadre mathématique que nous utilisons

est cependant totalement différent de celui utilisé pour la SCT et offre donc des possibilités différentes. Dans notre cas, nous travaillons avec l'algèbre de Boole des fonctions booléennes.

Nous avons retenu ce cadre mathématique pour sa très bonne adéquation à notre besoin car, comme nous allons l'établir dans le corps de ce mémoire :

- Dans le cas particulier des SED logiques, toute loi de commande peut être décrite à l'aide de fonctions booléennes.
- Les exigences exposées dans le cahier des charges peuvent être formalisées sous forme de relations entre des fonctions booléennes.
- Nous sommes aujourd'hui capables de déterminer automatiquement quelles sont les fonctions booléennes qui satisfont un ensemble d'exigences.

Pour mettre en place cette méthode de synthèse, il nous a fallu au préalable apporter une réponse théorique à un problème de mathématiques discrètes relatif aux algèbres de Boole : la résolution de systèmes d'équations à n variables dans une structure d'algèbre de Boole.

De manière à ce que ces travaux ne constituent pas uniquement une étude théorique de faisabilité, nous avons tenu à les valider par le traitement d'exemples nous conduisant au développement d'une maquette informatique nous permettant une résolution automatique de systèmes d'équations complexes. Ce travail nous a permis de montrer la praticabilité de nos résultats théoriques et nous a conduits à identifier par l'expérimentation un certain nombre de problèmes qu'une seule étude théorique n'aurait pas révélés.

Ce mémoire se compose de trois chapitres :

- Le premier chapitre est consacré à la présentation du problème et à son positionnement. Après une étude de cinq des méthodes de synthèse présentes dans la littérature, nous énonçons ensuite de manière synthétique notre démarche.

- Le chapitre 2 et les annexes A et B associées seront dédiés à la présentation des développements mathématiques que nous avons conduits pour être capables de résoudre aujourd'hui tout système d'équations entre des fonctions booléennes. Ces développements mathématiques sont relatifs aux structures d'algèbre de Boole, structure mathématique qui nous a permis de formaliser notre problème de synthèse et d'identifier une manière de le résoudre.

- Le chapitre 3 est consacré à notre méthode de synthèse. Après une présentation détaillée de chacune des étapes, nous montrerons comment un cahier des charges peut être formalisé sous la forme de relations entre des fonctions booléennes. Trois exemples de taille et de complexité croissantes seront traités pour permettre au lecteur de s'approprier cette méthode et d'évaluer ses possibilités.

Chapitre 1 :
Le problème et son positionnement

Ce premier chapitre, dédié à la présentation du problème et à son positionnement, comporte trois parties. La première partie est consacrée à la présentation des objectifs de nos travaux.

La section 2 est une étude bibliographique de cinq méthodes de synthèse proposées par la communauté scientifique pour obtenir automatiquement un modèle de la commande d'un SED. L'objectif de cette étude bibliographique est d'identifier quelles sont les données d'entrée, le mécanisme de synthèse et les résultats fournis pour chacune des méthodes. Nous listerons également les avantages et les limites de chacune de ces méthodes. Une fois cette étude présentée, il nous sera alors possible de positionner notre méthode par rapport aux autres travaux de la communauté.

La troisième et dernière partie de ce chapitre sera consacrée à la présentation générale de la méthode que nous proposons pour la synthèse de la loi de commande d'un SED logique. Nous expliquerons dans cette section pour quelles raisons le problème de synthèse de la loi de commande d'un SED logique peut toujours se ramener à un problème de synthèse de fonctions booléennes. Nous montrerons également en utilisant un exemple de taille élémentaire que le problème de la synthèse automatique de fonctions booléennes est néanmoins un problème complexe.

1. Objectifs des travaux

Les résultats présentés dans ce mémoire de thèse ont été obtenus dans le cadre de travaux portant sur la définition d'une méthode de synthèse automatique de la commande d'un Système à Événements Discrets (SED). L'objectif de cette méthode est d'obtenir la loi de commande à implanter dans le contrôleur industriel en charge de piloter un processus physique discret. Dans le cadre de ce mémoire de thèse, nous nous restreindrons aux SED pour lesquels les interactions avec la commande se font uniquement par l'intermédiaire de variables logiques (figure 1).

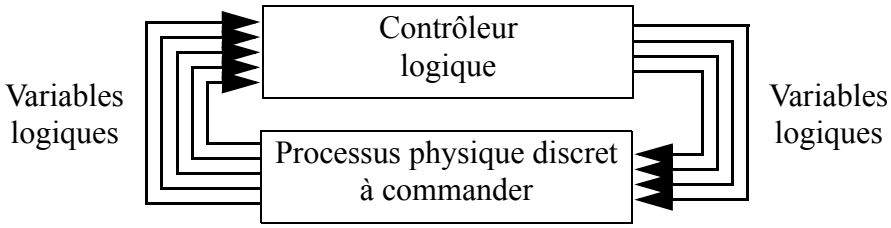


Figure 1 : Contrôle/commande d'un SED logique

L'originalité de cette méthode de synthèse réside dans la manière d'obtenir la loi de commande du SED à partir des attentes exprimées dans le cahier des charges. Cette loi de commande sera déterminée en résolvant de manière analytique un système d'équations représentant les caractéristiques attendues pour la loi de commande recherchée. Cette méthode originale repose principalement sur notre capacité à résoudre des systèmes d'équations entre des fonctions booléennes. Ce nouveau résultat mathématique a été obtenu dans le cadre de ce travail de thèse.

Être capable de résoudre un système d'équations entre des fonctions booléennes, quels que soient le nombre et la forme des équations, ne constitue pas seulement un nouveau résultat mathématique mais donne aux concepteurs de systèmes logiques un nouveau cadre mathématique opérationnel pour la synthèse. En utilisant ce cadre, tout concepteur d'un système logique pourra déterminer automatiquement quelles sont les fonctions booléennes qui satisfont le système d'équations qui caractérise son problème.

Cependant, il ne suffit pas de disposer d'un cadre mathématique performant pour être capable de synthétiser automatiquement un système logique à partir des attentes exprimées dans le cahier des charges : la manière d'exploiter ce cadre mathématique est aussi importante que le cadre lui-même. En complément de ce cadre mathématique, nous avons dû définir une nouvelle méthode de synthèse pour les systèmes logiques.

La rédaction de ce mémoire a été faite avec un double objectif :

- Le premier objectif est de présenter les résultats mathématiques que nous avons établis pour être en mesure de résoudre un système d'équations entre des fonctions booléennes quels que soient le nombre et la forme des équations.
- Le deuxième objectif est de donner les bases de la méthode de synthèse que nous proposons pour obtenir la loi de commande d'un SED logique par résolution d'un système d'équations entre des fonctions booléennes.

2. Positionnement bibliographique

Vouloir synthétiser automatiquement la commande d'un système logique à partir de son cahier des charges est un projet ambitieux qui correspond à une réelle attente industrielle. Les concepteurs sont aujourd'hui amenés à développer des commandes de plus en plus complexes pour lesquelles les exigences de sûreté sont de plus en plus fortes dans des délais de plus en plus courts. Ils sont donc demandeurs de méthodes leur permettant d'être plus performants.

La méthode proposée dans ce document ne se veut être qu'un des possibles éléments de réponse à ce problème de conception. Il serait prétentieux de prétendre apporter l'unique réponse à une problématique aussi complexe qui se heurte aux trois difficultés scientifiques suivantes :

- la représentation formelle d'un problème énoncé à partir de **spécifications informelles** potentiellement en contradiction,
- l'obtention de **l'ensemble des solutions** à un problème donné,
- **le choix d'une solution** particulière parmi l'ensemble possible de solutions.

Dans la suite de ce document, nous allons étudier quelques approches trouvées dans la littérature par rapport à ces trois problèmes scientifiques. Parmi les différentes méthodes de synthèse proposées dans la littérature pour obtenir la commande d'un système logique, cinq d'entre elles ont en effet retenu notre attention. Bâties sur des cadres mathématiques différents, développées avec des objectifs opérationnels également différents, ces cinq méthodes apportent donc des réponses différentes au problème de la synthèse d'un SED. Il est cependant possible de les étudier en analysant comment sont abordées les trois difficultés scientifiques que nous venons d'énoncer. Les cinq méthodes étudiées dans cette partie sont :

- La synthèse de contrôleurs logiques à l'aide de la « Supervisory Control Theory » (SCT) : cette méthode, qui fait l'objet de nombreux travaux dans la communauté scientifique, est aujourd'hui un élément de référence incontournable pour tout travail de synthèse de modèles pour les SED.
- La synthèse de contrôleurs logiques basée sur une modélisation polynomiale : cette méthode développée à la Division of Automatic Control du Department of Electrical Engineering de l'université de Linköping dans le cadre de la thèse de J. Gunnarson est entièrement algébrique. J. Gunnarson exploite les possibilités de résolution dans une structure mathématique particulière appelée « quotient ring » pour obtenir une loi de commande.
- La synthèse de Sequential Function Chart (SFC) à partir de spécifications données en langage naturel : cette méthode développée au Process Control Laboratory de la Technische Universität Dortmund dans le cadre de la thèse de S. Lohmann consiste à obtenir un modèle SFC à partir d'un ensemble de spécifications informelles.
- La synthèse de séquences de fonctionnement : cette méthode développée à l'IETR de Rennes dans le cadre de la thèse de T. Meftah repose sur une représentation originale du système physique à commander à l'aide d'« axes ». Les séquences sont ensuite obtenues à l'aide d'un outil de model-checking.
- La synthèse de contrôleurs logiques basée sur une algèbre spécifique : cette méthode développée ces dernières années au LURPA dans le cadre de la thèse d'A. Medina est également entièrement algébrique et utilise une algèbre dédiée à la manipulation de signaux logiques.

2.1. La théorie de supervision de Ramadge et Wonham

À la fin des années 80, une avancée majeure a été effectuée dans le domaine des systèmes à évènements discrets grâce à la théorie de supervision développée par P.J.G. Ramadge et W.M. Wonham [RAM87A], [RAM89]. Cette théorie permet de synthétiser un « superviseur » qui, une fois couplé à un système, impose à celui-ci de respecter des spécifications données.

2.1.1. Caractéristiques de l'approche proposée

Pour atteindre ce but, le système est considéré comme un générateur spontané d'évènements. Ces évènements (Σ) sont répartis en deux classes : les évènements contrôlables (Σ_c) et les évènements incontrôlables (Σ_u). Cette classification permet de déterminer sur quels évènements le superviseur peut agir. En effet, pour contraindre le comportement du système à respecter un

comportement spécifié, le superviseur ne peut interdire que la génération d'évènements contrôlables. Pour cela, il dispose de la connaissance de tous les évènements qui ont déjà été générés par le système. Le superviseur permet donc de restreindre le comportement d'un système pour que celui-ci respecte une spécification donnée (figure 2).

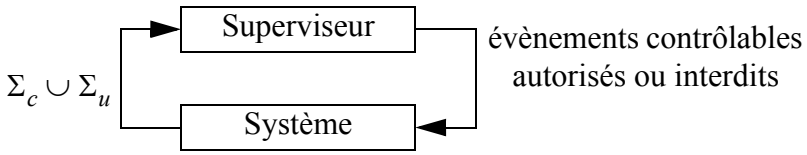


Figure 2 : Système vu par la théorie de la supervision

Pour générer un superviseur, il faut connaître l'ensemble des comportements possibles du système. Ces comportements sont décrits sous la forme d'un langage L. Un langage est constitué d'un ensemble de mots. Un mot est une séquence ordonnée d'évènements qui peut être générée par le système. La théorie de la supervision impose que le langage décrivant le système soit préfixe clos (le mot u est le préfixe du mot v s'il existe un mot w tel que $uv = w$). Soit $\bar{L}$ l'ensemble de tous les préfixes de tous les mots d'un langage L. L est un langage préfixe clos si $\bar{L} = L$.

Un langage préfixe clos peut également être décrit à l'aide d'un automate à états. Un automate est défini par un quadruplet (Q, Σ, δ, q_0). Q représente l'ensemble des états, Σ est l'ensemble des évènements, $\delta : Q \times \Sigma \to Q$ est la fonction transition et q_0 représente l'état initial. La fonction transition permet de définir quels évènements peuvent se produire à partir d'un état donné. Le langage reconnu par un automate représente l'ensemble des séquences d'évènements qui sont reconnues par cet automate. Lorsque l'automate comporte un nombre fini d'états, alors le langage associé est dit régulier [HOP79].

Par ailleurs, une extension de cette classe d'automates, indiquant que certains états sont marqués, est souvent utilisée. Un état marqué représente une situation « intéressante » à atteindre, comme la fin d'une tâche [RAM89]. À partir de ces états marqués, un nouveau langage peut être associé à un automate, c'est l'ensemble des séquences d'évènements qui aboutissent à un état marqué. Ce langage (le langage marqué) est en général différent du langage représentant tous les évènements pouvant être générés par le système (langage généré). Un automate est dit non bloquant si, à partir de n'importe lequel de ses états, il est possible d'atteindre un état marqué : on dit également que chacun de ses états est coaccessible.

Dans la théorie de la supervision, la spécification du système est également définie à l'aide d'un langage. Pour qu'un superviseur existe, il est nécessaire que ce langage soit préfixe clos et qu'il soit contrôlable. Un langage contrôlable K est un langage dont tout préfixe des mots qui le composent, suivi d'un évènement incontrôlable cohérent avec le comportement du système, est encore le préfixe d'un mot du langage, ce qui se traduit par $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$.

Lorsque toutes ces conditions sont réunies, plusieurs algorithmes ([RAM87B] et [KUM91]) ont été développés pour obtenir le superviseur le plus permissif, c'est-à-dire celui qui laisse le plus de liberté au système tout en lui imposant de respecter la spécification.

La théorie de la supervision a été l'une des premières techniques permettant la synthèse de la commande d'un système à évènements discrets. Elle a été à l'origine de nombreuses recherches dans le domaine des SED. Elle comporte plusieurs idées très intéressantes comme le fait de partir de l'ensemble des comportements possibles du procédé et de le restreindre ensuite pour obtenir le comportement autorisé. Une autre idée intéressante est la notion d'état marqué qui permet d'indiquer des objectifs à atteindre. L'obtention du superviseur se fait en s'assurant qu'il est toujours possible d'atteindre ces objectifs. Tout ceci est possible car certaines hypothèses sur les modèles ont été faites. Malheureusement, ces hypothèses limitent fortement les possibilités d'application de cette théorie dans le cadre d'applications industrielles.

Il a ainsi été choisi de considérer le système comme un générateur d'évènements. Cette modélisation est bien adaptée si l'on considère le système composé du processus couplé avec le contrôleur et que c'est bien un superviseur qui est recherché. Cette modélisation ne permet cependant pas d'étendre les possibilités de la théorie pour synthétiser un contrôleur. En effet, pour cela, le modèle de départ ne peut concerner que ce qui est connu, c'est-à-dire le processus. Or celui-ci est composé d'un ensemble de composants qui réagissent à des ordres tout en fournissant des informations sur leur état courant. De plus, les informations échangées sont le plus souvent des signaux qui sont persistants dans le temps, et non des évènements qui représentent une évolution fugace d'un paramètre.

Une autre limitation se situe au niveau de l'évolution de la complexité de recherche d'un superviseur en fonction de la taille du système étudié. Ramadge et Wonham [RAM89] ont montré que cette complexité évolue de manière polynomiale par rapport au nombre d'états d'un système. Malheureusement, ce nombre d'états peut évoluer de manière exponentielle par rapport au nombre de

composants du système [RAM89]. Ceci implique que cette technique est très sujette au problème d'explosion combinatoire.

Toutefois, plusieurs travaux ont tenté de repousser ces limites. Ainsi S. Balemi [BAL92] a redéfini le cadre formel utilisé pour permettre de prendre en compte le comportement d'entrées/sorties d'un système contrôlé. Il a ainsi créé la notion d'évènement forçable. Ceci a permis de se rapprocher de la notion de contrôleur, dont le rôle est d'agir directement sur le processus. R.J. Leduc [LED02] a travaillé sur le problème d'explosion combinatoire en modélisant le système de manière modulaire et hiérarchique. A. Gouin [GOU99] a proposé des automates temporisés afin de permettre la synthèse d'un superviseur pour les systèmes à évènement discrets temporisés. Bien d'autres auteurs ont contribué également à prolonger la théorie de la supervision sans toutefois réussir à repousser complètement ses limitations. Plusieurs exemples ont également été traités afin de montrer les possibilités d'utilisation de cette théorie. Dans [ROU05], les auteurs ont utilisé la SCT afin d'obtenir un **contrôleur** pour un portail automatique. Ils ont ainsi pu mettre en évidence les limites et les difficultés d'utilisation de la SCT pour la synthèse d'un contrôleur.

2.1.2. Exemple d'illustration

Afin d'illustrer la SCT, l'exemple de [ROU05] est repris dans cette partie. L'objectif est d'obtenir la loi de commande du contrôleur qui doit piloter un portail automatique. Pour cela, la première étape est de modéliser sous la forme d'automates le comportement possible du système. Dans le cas du portail, plusieurs automates à deux ou trois états sont construits pour représenter les états possibles des capteurs et du moteur qui actionne le portail (figure 3).

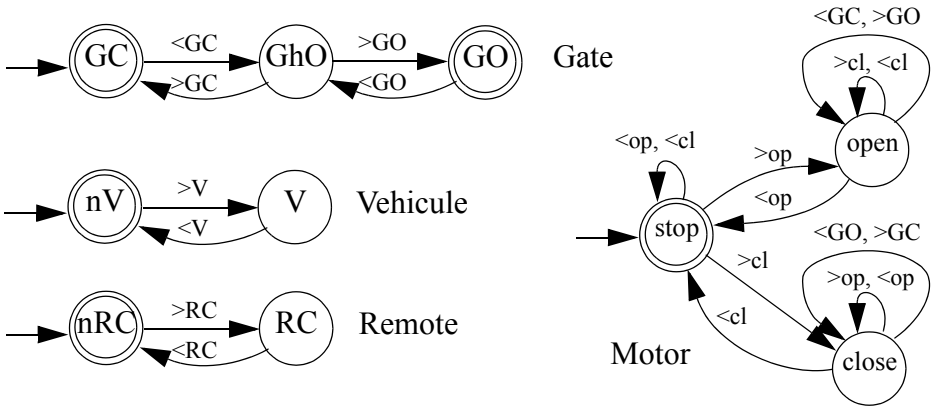


Figure 3 : Automates des éléments du portail automatique

Dans cette approche, les auteurs proposent de modéliser également le fonctionnement du contrôleur par un automate (figure 4).

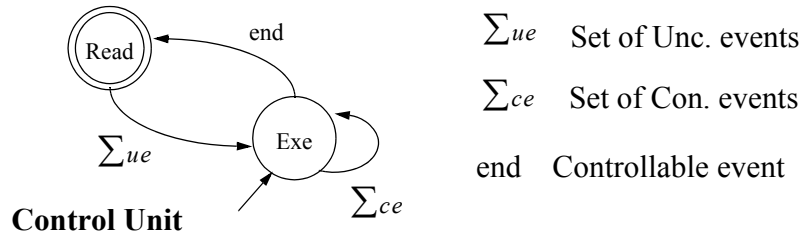


Figure 4 : Modélisation du comportement cyclique du contrôleur

L'étape suivante est l'expression des spécifications du comportement attendu à l'aide de plusieurs automates de spécification comme celui donné à la figure 5 qui spécifie l'arrêt de l'ouverture du portail lorsqu'il est déjà complètement ouvert.

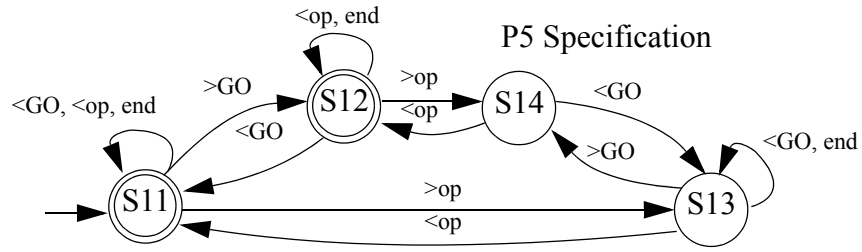


Figure 5 : Spécification sur l'ouverture du portail

Dans [CAN08], les auteurs donnent sur ce même exemple les dimensions des modèles traités. Le modèle du processus se compose de 11 automates et l'automate produit comporte 481 états et 1330 transitions. Les spécifications regroupent 11 automates conduisant à un automate produit de 276 états et 1215 transitions. Après minimisation, le superviseur calculé avec l'outil Supremica [AKE03] comporte 110 états et 154 transitions. Les auteurs ne retiennent qu'une partie de ce comportement pour le contrôleur (45 états et 70 transitions). Ils reconnaissent que pour établir une loi de commande, l'approche qu'ils proposent est très sensible à la taille des modèles établis pour le processus et les spécifications.

2.1.3. Analyse de l'approche

Notre démarche comporte plusieurs différences fondamentales avec la SCT. La première concerne la nature du système synthétisé. Ce n'est pas un **superviseur** qui est recherché mais bien un

contrôleur. En effet, le modèle recherché utilise les informations provenant du processus pour fournir des commandes qui permettent d'agir directement sur ce processus pour répondre aux besoins attendus. La seconde différence se situe au niveau des modèles employés. Dans le cas de la théorie de la supervision, le système est considéré comme étant déjà en boucle fermée avec un contrôleur, il évolue donc de lui-même sans actions extérieures. Dans notre approche, le contrôleur n'est pas connu et c'est lui qui est recherché, il est donc nécessaire de ne pouvoir considérer que le comportement de la partie opérative. Or celle-ci ne peut pas être autonome puisqu'elle reçoit des ordres d'un contrôleur. La dernière différence est le cadre employé. Notre approche utilise un cadre algébrique alors que la théorie de la supervision manipule des langages.

Comme indiqué dans l'introduction de cette partie, nous allons maintenant conclure quant à la capacité de la SCT à résoudre les trois problèmes scientifiques présentés précédemment :

- La représentation formelle du problème,
- L'obtention de l'ensemble des solutions,
- Le choix d'une solution particulière.

Concernant la **formalisation du problème**, elle est entièrement réalisée par un expert qui est supposé être capable d'exprimer toutes les spécifications sous la forme d'automates. Aucune technique ou méthode ne fut développée dans la théorie originelle. Plusieurs auteurs, avec plus ou moins de succès, ont ensuite mis au point des techniques pour faciliter cette démarche ([GOU04], [PET07],...). Elle reste toutefois encore aujourd'hui l'un des points délicats de la méthode.

Concernant l'**obtention des solutions** et le **choix d'une solution particulière**, le mécanisme de synthèse ne permet d'obtenir qu'une solution particulière, mais il s'agit de la solution « optimale » (au sens de « la plus permissive ») qui est celle permettant le plus de liberté. La recherche de l'ensemble des solutions n'a jamais, à notre connaissance, été faite.

2.2. La modélisation par polynômes de Gunnarson

2.2.1. Caractéristiques de l'approche proposée

Une approche de synthèse tout à fait différente de la précédente a été proposée par Gunnarson [GUN96]. La caractéristique essentielle de cette approche est l'utilisation d'une modélisation originale entièrement algébrique [GUN97]. Le cadre mathématique utilisé pour la modélisation est un « quotient ring » $\mathbb{R}_q[Z]$.

Cet anneau est dérivé d'un « polynomial ring » $\mathbb{F}_q[Z]$ qui regroupe l'ensemble des polynômes fonctions d'un ensemble de variables $Z = \{z_1, \ldots, z_n\}$ dont les coefficients appartiennent à un « field fini » de q éléments. Un field fini est un ensemble fini d'éléments disposant de deux opérations + et $\times$ qui sont internes, associatives, commutatives, disposant d'un élément neutre et d'un élément inverse. L'opération $\times$ est également distributive par rapport à l'opération +. Ces deux opérations sont les opérations arithmétiques classiques modulo le nombre total d'éléments sur l'ensemble. Un quotient ring est une sous-partie d'un « polynomial ring » de telle manière que chaque relation entre les variables de Z s'exprime par un unique polynôme de $\mathbb{R}_q[Z]$.

À partir de ce cadre mathématique, l'auteur modélise toutes ses relations à l'aide de polynômes sur $\mathbb{R}_q[Z]$ avec q correspondant au nombre maximal de valeurs différentes que peuvent prendre les différentes variables. Pour cela, l'auteur utilise le polynôme d'interpolation de Lagrange.

Un SED est considéré comme un système produisant et réagissant à des signaux. Les signaux d'entrée du système seront notés u, tandis que les signaux de sortie seront notés y. L'état du système, quant à lui, sera noté x (figure 6).

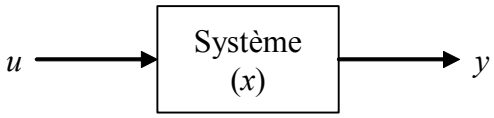


Figure 6 : Modèle d'un SED selon Gunnarson

Pour modéliser le comportement d'un système, l'auteur utilise une représentation habituelle qui consiste à formaliser séparément le changement d'état et les actions effectuées par le système. Pour le changement d'état, il utilise une relation de récurrence liant l'état futur du système x^+ à son état courant et aux informations disponibles (les entrées u). Les actions du système (c'est-à-dire ses sorties) sont, quant à elles, modélisées par une relation les liant à l'état courant et aux entrées du système. Dans le cas d'un système déterministe, la modélisation d'un système est donc réalisée à l'aide de deux fonctions :

$$\begin{cases} x^+ = f(x, u) \\ y = g(x, u) \end{cases}$$

Bien que cette modélisation à l'aide de polynômes soit le fer de lance des travaux de Gunnarson, celui-ci propose une méthode permettant de synthétiser la commande d'un système. La première

étape de la méthode de synthèse est la modélisation du comportement du système avec une ou plusieurs relations exprimées à l'aide de polynômes.

La seconde étape consiste à modéliser les attentes du système avec le même formalisme, c'est-à-dire à l'aide de polynômes. Ces attentes sont par exemple des comportements interdits ou des priorités entre différentes actions.

La dernière étape est la recherche de lois de commande pour le système. Pour cela, les polynômes sont exprimés de telle manière que les lois de commande recherchées correspondent aux racines des polynômes. Pour accomplir cette étape, les bases de Gröbner sont utilisées. Ces bases permettent, à partir d'éléments d'un quotient ring, de trouver les relations liant certaines variables aux autres. Il est donc possible de trouver les lois de commande du système en utilisant cet outil.

2.2.2. Exemple d'illustration

Pour illustrer cette méthode de synthèse, nous reprenons ici l'exemple utilisé par l'auteur. L'objectif de cet exemple est la recherche de la commande du niveau de remplissage d'un réservoir d'eau (figure 7). Celui-ci dispose d'une arrivée alimentée par une pompe (y1 et d pour signaler une défaillance) et de deux écoulements dont un est équipé d'un débitmètre (w). Plusieurs vannes sont placées sur l'alimentation (y2) et sur les évacuations du réservoir (y3 et y4). Les lois de commande recherchées correspondent aux relations liant les sorties de la commande (y1 à y4), c'est-à-dire l'état de la pompe et des différentes vannes, en fonction des entrées de la commande (w et d) et de son état interne, c'est-à-dire le niveau (x) dans le réservoir.

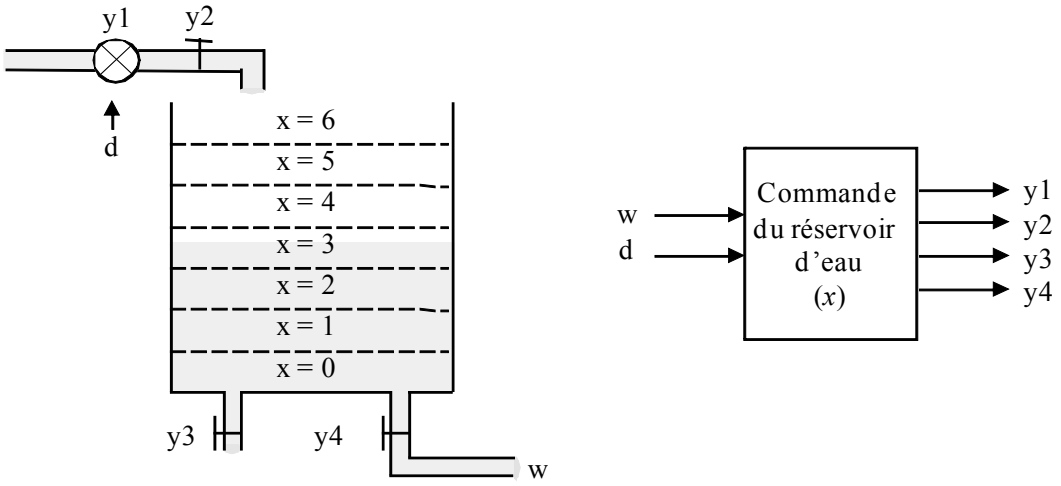


Figure 7 : Le réservoir d'eau

La première étape de la méthode de synthèse est la modélisation du comportement du système. Dans le cas du réservoir d'eau, la modélisation est réalisée à l'aide de deux fonctions :

$$\begin{cases} \phi(d, y1, y2, y3, y4, w) = 1 + 3dy_2 + \{9 \text{ termes}\} + d^2 y_1 y_2 y_3 y_4 w \\ x^+(\phi, x) = 6\phi + \{21 \text{ termes}\} + 6\phi^3 x^6 \end{cases}$$

Ces fonctions représentent les lois de comportement du processus. Pour cela, il a été choisi de passer par une variable d'état supplémentaire qui est le flux global (ϕ, qui est la différence entre le flux de remplissage et le flux de vidange) auquel est soumis le réservoir d'eau. La première fonction indique la valeur de ce flux d'eau en fonction de l'état des différentes vannes ($y2$ à $y4$), de la pompe alimentant l'arrivée d'eau ($y1$ et d) et de l'état de l'écoulement (w). Puis une seconde fonction est nécessaire pour déterminer le niveau dans le réservoir en fonction du flux d'eau (ϕ) et du niveau précédent (x).

La seconde étape consiste à modéliser les attentes du système. Dans le cas du réservoir d'eau, le cahier des charges contient deux spécifications. La première est que le réservoir ne doit jamais être complètement plein ou complètement vide. Cette spécification est représentée par le polynôme $p(x)$ qui s'annule pour tous les niveaux sauf 0 et 6. La seconde spécification est que le niveau dans le réservoir doit être de préférence au milieu ($x = 3$). Cette spécification est réalisée par le polynôme $J(x)$ qui introduit une priorité entre les différents niveaux atteignables. Cette priorité est représentée par une valeur qui est associée à chacun des niveaux. Cette valeur augmente à mesure que le niveau s'éloigne du niveau moyen ($x = 3$). Pour que le réservoir ait un niveau proche du milieu, les lois sont cherchées en minimisant le polynôme $J(x)$. Les deux polynômes mentionnés sont :

$$\begin{cases} p(x) = 1 + x + 6x^2 + x^3 + 6x^4 + x^5 + 5x^6 \\ J(x) = 3 + 3x + 6x^2 + x^3 + 2x^4 + 6x^5 + 2x^6 \end{cases}$$

La dernière étape est la recherche des lois de commande. Dans le cas du réservoir, le problème à résoudre est la recherche des racines de deux polynômes. Le premier polynôme permet de déduire le flux d'eau global en fonction du niveau courant. Le second polynôme permet de déterminer l'état des vannes et de la pompe pour atteindre le flux d'eau global nécessaire. Ces polynômes sont :

$$\begin{cases} p_1(x, \phi, m) = -3\phi + 3\phi^2 + \{10 \text{ termes}\} - 2\phi^5 \\ p_2(\phi, y_1, y_2, y_3, y_4, w, d) = 1 - \phi + \{10 \text{ termes}\} + d^2 y_1 y_2 y_3 y_4 w \end{cases}$$

En utilisant les bases de Gröbner, les lois de commande trouvées sont :

$$\begin{cases} y_1(\phi, w, d) = -d\phi - 3d^2\phi + \{4 \text{ termes}\} + d^2\phi^2 w \\ y_2(\phi, w, d) = 1 - 3d + \{8 \text{ termes}\} + 3d^2\phi^2 w \\ y_3(\phi, w, d) = 1 + 2\phi + \{3 \text{ termes}\} + 3\phi^2 w \\ y_4(\phi, w, d) = 1 - 3\phi + \{10 \text{ termes}\} - d^2\phi^2 w \end{cases}$$

2.2.3. Analyse de l'approche

Rappelons que ce travail vise avant tout la modélisation algébrique. La synthèse n'est abordée qu'à la fin de la thèse comme une des applications possibles du cadre mathématique de modélisation proposé. En ce sens, l'utilisation de ces travaux en synthèse est inspirée de la SCT par :

- l'utilisation d'une modélisation du processus qui sera ensuite contrôlé,
- la modélisation du résultat attendu par interdiction du comportement non souhaité (qui ne respecte pas les spécifications).

Un point fort de l'approche de Gunnarson est avant tout le cadre mathématique utilisé. En effet, l'utilisation d'un cadre algébrique manipulant des variables appartenant à des ensembles finis apporte un certain nombre d'avantages. Le premier est que la manipulation de relations algébriques permet d'éviter de devoir manipuler des espaces d'états comme c'est le cas par exemple dans la SCT. Le second avantage provient de la nature des variables manipulées. Celles-ci doivent appartenir à des ensembles finis. Ce cadre permet l'utilisation de variables entières qui apportent plus de possibilités que la manipulation de variables uniquement booléennes. Un autre point fort de cette approche est le résultat obtenu par la méthode de synthèse. En effet, bien que cette méthode s'inspire de la théorie de la supervision de Ramadge et Wonham, le résultat obtenu est bien un contrôleur donné sous la forme de lois de commande.

Toutefois, cette approche présente par ailleurs un certain nombre de limites qui sont intimement liées à ses points forts. La principale limite est justement le cadre mathématique employé. Celui-ci, bien que très performant pour la modélisation, n'est pas du tout habituel pour les concepteurs de systèmes automatisés. Cette méconnaissance de ce cadre entraîne de nombreuses difficultés pour appliquer la méthode présentée. En effet, les données d'entrée de cette méthode sont des relations entre des variables de ce cadre mathématique. La formalisation du cahier des charges sous

cette forme paraît difficile pour des ingénieurs automaticiens. De même, l'interprétation des résultats et la confiance qu'ils peuvent susciter en seront d'autant plus faibles.

Revenons maintenant aux trois problèmes scientifiques présentés en début de l'étude bibliographique qui sont :

- La représentation formelle du problème,
- L'obtention de l'ensemble des solutions,
- Le choix d'une solution particulière.

Concernant **la formalisation du problème**, plusieurs exemples sont traités et à chaque fois une modélisation est obtenue. Toutefois, aucune méthode ni démarche n'est donnée pour obtenir ces modèles. Pour **la recherche des solutions** du problème et **le choix d'une solution particulière**, la méthode permet bien de trouver toutes les solutions mais aucune précision n'est donnée sur le moyen d'en choisir une en particulier.

On a vu que cette méthode présentait de nombreux points intéressants qui n'étaient limités que par le fait que le cadre mathématique employé est trop novateur pour les concepteurs qui seraient amenés à l'utiliser pour la synthèse de la commande de leur système. C'est pour cela que notre approche se base sur une méthode de synthèse très proche mais avec un cadre mathématique plus proche de ce que les experts du domaine ont l'habitude de manipuler : des fonctions booléennes. Ce choix a pu être fait car nous n'avons considéré que les systèmes logiques.

2.3. Synthèse par couplage fonctionnel de Lohmann

2.3.1. Caractéristiques de l'approche proposée

La méthode de synthèse détaillée maintenant est celle mise au point par S. Lohmann dans [LOH06]. Cette méthode présente plusieurs particularités. La première concerne les données d'entrée et de sortie de la méthode. En effet, le type de données d'entrée est assez libre puisqu'il autorise l'utilisation de besoins à satisfaire exprimés en langage naturel. Au niveau des données produites par la méthode, elles ont la forme d'un Sequential Function Chart (SFC) qui est un langage de programmation utilisé pour les Automates Programmables Industriels (API). Les résultats de la méthode sont donc directement utilisables et implémentables dans un contrôleur sans aucune autre phase de conversion. La seconde particularité est la présence d'une phase de vérification di-

rectement dans la méthode de synthèse afin de s'assurer que les différents éléments du cahier des charges sont bien respectés.

Pour atteindre cet objectif, la méthode de S. Lohmann se décompose en trois grandes étapes (figure 8). La première étape (I) consiste à obtenir un SFC qui respecte les données d'entrée décrivant le comportement attendu. La seconde étape (II) a pour but de convertir le SFC précédemment obtenu en automate temporisé. Cette conversion permet dans une troisième étape (III) d'utiliser cet automate conjointement à un modèle du processus pour vérifier par model-checking que les éléments du cahier des charges sont bien vérifiés. Le cœur de la méthode de synthèse est la première étape (I), si bien que les autres ne seront pas détaillées. Si le lecteur désire plus d'informations, il pourra consulter [LOH06], qui détaille les deux autres étapes, ou [STU05], qui s'intéresse plus particulièrement à l'étape (II).

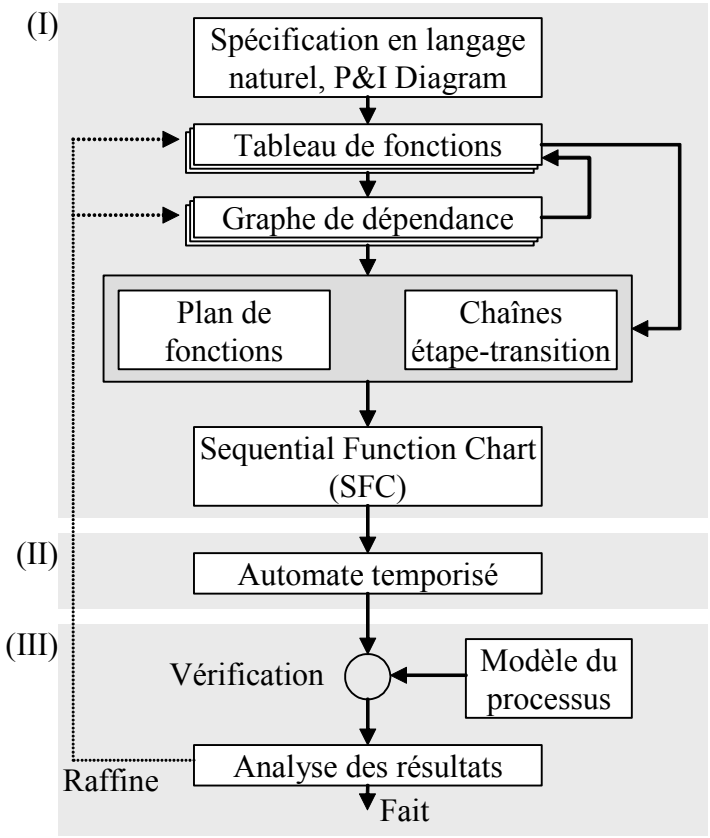


Figure 8 : Méthode de synthèse de S. Lohmann

L'étape (I) a pour but d'obtenir, à partir d'un ensemble de besoins informels, le modèle d'une commande sous la forme d'un SFC. Les données d'entrée de cette partie sont des spécifications en langage naturel et un « Piping and Instrumentation Diagram » (P&I Diagram). Les spécifications

en langage naturel décrivent les comportements attendus par le système, alors que le P&I Diagram précise les actionneurs et les capteurs présents au sein du système. Pour réaliser la commande, cinq étapes sont nécessaires. Les trois premières ont pour but de formaliser les spécifications informelles. Pour cela, deux modèles sont construits successivement et récursivement : un tableau de fonctions et un graphe de dépendance. La quatrième étape de la synthèse est la conversion des deux précédents modèles en fragments de SFC. Les graphes de dépendance sont convertis en des « plans de fonctions », tandis que les tableaux de fonctions sont convertis en « chaînes étape-transition ». Les plans de fonctions sont des traductions de l'organisation des fonctions provenant des graphes de dépendance. Les fonctions y sont représentées par des macro-étapes. Les chaînes étape-transition sont des séquences d'actions nécessaires pour accomplir chaque fonction. Ces chaînes sont ensuite substituées aux macro-étapes correspondantes durant le cinquième temps pour construire le SFC recherché. Seules ces deux dernières étapes sont actuellement réalisées de manière automatique. Pour bien comprendre la méthode de synthèse, il est nécessaire de détailler les deux modèles importants que sont les tableaux de fonctions et les graphes de dépendance.

Un tableau de fonctions décrit les différentes fonctions que doit exécuter le système. Une fonction est définie comme étant une succession d'actions permettant de répondre à un besoin. Le tableau de fonctions (tableau 1) est constitué de cinq colonnes :

Function		Precondition	Sensor	Operation	Actuator
Dose reactant into R1	P1 On	The desired liquid level for the solvent is reached.	$Lf1 = 1$	Dose reactant from A1 to R1 while stirring.	S-M1, S-V1, S-Temp_V1
	P1 Off	The upper bound of the reactant concentration is reached.	$QI\_up = 1$	Stop to dose reactant from A1 to R1, continue stirring.	R-V1
	Terminal	alternative 1: The desired product concentration is reached.	$QI3 = 1$	none (S1).	-
		alternative 2: The waste concentration in the reactor is too high.	$QI4 = 1$	none (S2).	-
		loop: the lower limit of reactant concentration is reached.	$QI\_down = 1$	loop.	-

Tableau 1 : Exemple de tableau de fonctions

- La première comprend le nom de la fonction considérée ainsi que les différentes actions qui permettent d'accomplir cette fonction.

- La seconde colonne contient, pour chacune des actions, les préconditions qui doivent être satisfaites pour pouvoir initialiser l'action. Ces préconditions sont formulées en langage naturel.
- La troisième colonne contient une déclaration logique qui spécifie la précondition donnée dans la colonne précédente. Cette déclaration contient habituellement les variables d'entrée de la commande correspondant aux capteurs du système.
- La quatrième colonne comporte une description de l'action considérée. Cette description est donnée en langage naturel.
- La dernière colonne décrit l'affectation des variables de sortie de la commande qui permettront de piloter les actionneurs du système. L'affectation est donnée sous la forme « qualificatif-nom de la variable/du programme ».

Le second modèle important est le graphe de dépendance (figure 9). Ce graphe, inspiré des diagrammes de Gantt, est nécessaire pour clarifier les interdépendances entre les différentes fonctions de manière graphique. Pour cela, chaque exécution d'une fonction est représentée par un rectangle sur une ligne qui lui est affectée. La taille du rectangle représente la durée d'exécution de la fonction. Toutefois celle-ci n'a pas besoin d'être très précise, en effet, seule la position relative de ses limites par rapport aux autres fonctions est importante. Cette position permet de caractériser les relations entre les exécutions de fonctions. Il y a trois différents types de relations :

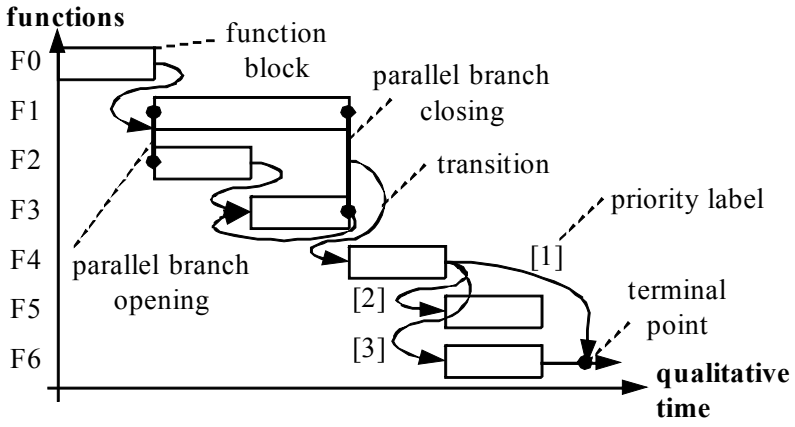


Figure 9 : Exemple de graphe de dépendance

- L'exécution séquentielle. Cette relation est représentée par une flèche qui part de la fin d'une fonction vers le début de la suivante.

- L'exécution parallèle. Deux fonctions peuvent avoir leur début et/ou leur fin d'exécution synchronisés. Ceci se représente par une ligne verticale dont les extrémités sont des points situés sur les extrémités des exécutions à synchroniser.
- L'exécution alternative. Cette relation se représente de la même manière que l'exécution séquentielle mais en multipliant les flèches. Une priorité entre les différents choix peut être indiquée par des chiffres au niveau des flèches.

La méthode de synthèse propose que la construction de ces deux modèles se fasse par raffinage itératif. Lors de la première itération, seule une modélisation globale est réalisée. Le raffinage se poursuit ensuite jusqu'à arriver à un niveau de détail suffisant pour permettre la génération d'un SFC, c'est-à-dire lorsque les préconditions et les actions ont pu être traduites en termes d'entrées/sorties de la commande (la troisième et la cinquième colonne des tableaux de fonctions).

2.3.2. Analyse de l'approche

Le point fort de cette méthode de synthèse, moins formelle que la précédente, réside dans la formalisation des spécifications de départ. Cette tâche, qui est laissée à un expert, est facilitée par la décomposition en deux problèmes moins complexes en s'occupant d'une part de la réalisation de plusieurs fonctions et d'autre part de l'organisation entre ces différentes fonctions. La formalisation des données de départ est également facilitée par la mise en place et l'utilisation de modèles qui permettent de faire le lien entre les données informelles et une modélisation formelle de celles-ci. De plus, la présence de cette formalisation va ensuite simplifier toute modification ultérieure de la commande. En effet, lors de la modification de la commande ou d'ajouts de nouveaux besoins à satisfaire, un cahier des charges précis et formel de ce qui a déjà été fait est disponible.

Toutefois, ce point est également une limite. En effet, bien que la formalisation soit facilitée par la décomposition en fonctions et l'organisation de celles-ci, elle demeure compliquée à réaliser. Les données d'entrée nécessaires à la génération d'une commande sous la forme d'un SFC demandent beaucoup de travail pour être créées. Cela est dû aux nombreuses informations qu'il est nécessaire de fournir sur chacune des fonctions.

De même que pour les autres méthodes, étudions maintenant comment sont traités les trois problèmes scientifiques présentés en début de l'étude bibliographique :

- En ce qui concerne **la formalisation du problème**, cette partie est laissée aux bons soins d'un expert. Toutefois, la méthode lui fournit des modèles particuliers afin de simplifier sa tâche.

- Le problème de **recherche des solutions** et du **choix d'une solution particulière** n'est jamais abordé. L'approche que nous venons de présenter aboutira systématiquement à un résultat. Aucun autre ne sera trouvé et celui qui est trouvé ne semble pas être systématiquement optimal pour un critère donné.

Par rapport à notre approche, l'idée que nous avons voulu conserver de cette méthode est la liberté laissée au concepteur pour spécifier le comportement souhaité de son système. Bien que cette liberté implique ensuite une difficulté de formalisation, c'est une idée qui nous est apparue comme séduisante et que nous avons décidé de garder. C'est pour cela que notre méthode accepte un grand nombre de formalismes différents en entrée afin que chaque concepteur puisse choisir celui qu'il préfère pour décrire chaque élément de spécification.

2.4. La synthèse par recherche de séquences de Meftah

2.4.1. Caractéristiques de l'approche proposée

Meftah [MEF05] a présenté une méthode de synthèse originale par contraintes pour les systèmes automatisés de production. Elle se caractérise par l'utilisation de deux notions particulières qui la rendent unique et intéressante. La première notion utilisée concerne la modélisation du système qui est réalisée sous la forme « d'axes ». La seconde particularité est l'utilisation d'un model-checker pour synthétiser une loi de commande. Du fait du type de la modélisation utilisée (les axes), cette méthode de synthèse n'est applicable que sur les systèmes mécaniques automatisés à axes de déplacement.

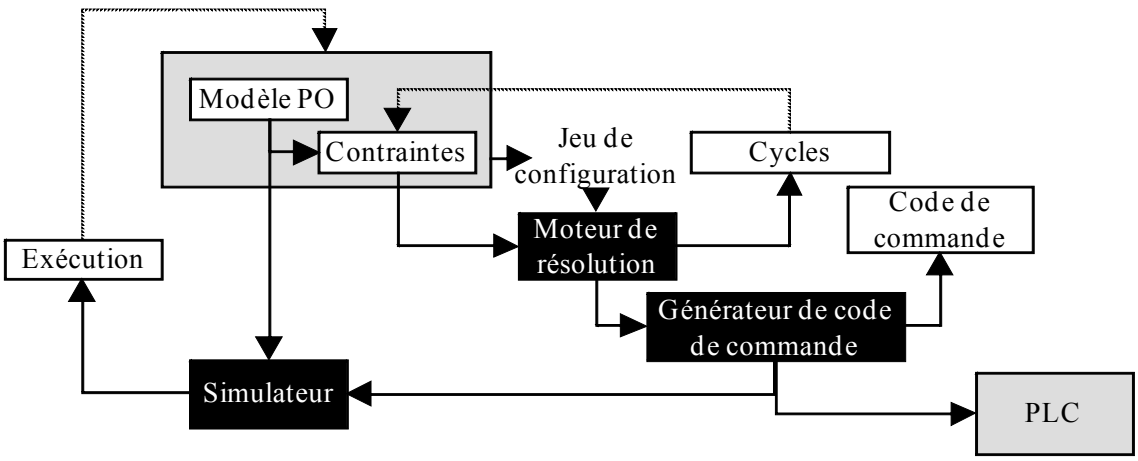


Figure 10 : Le processus de spécification et de réalisation de la commande

La méthode présentée se décompose en quatre étapes (figure 10). La première étape est la modélisation de la partie opérative du système et du comportement désiré. La modélisation de la partie opérative se fait par décomposition modulaire sous la forme d'axes tandis que la description du comportement attendu est donnée sous la forme d'un réseau de contraintes typées accompagné d'une situation de départ et d'une situation d'arrivée. La seconde étape de la méthode consiste à utiliser un moteur de résolution qui calcule la séquence optimale d'enchaînement des mouvements pour passer de la situation de départ à la situation d'arrivée. Le résultat est donné sous la forme de diagrammes de cycles. Ce moteur de résolution a la particularité comme nous le verrons plus tard d'utiliser le model-checker UPPAAL pour accomplir sa tâche. En rebouclant sur la première étape, à partir de la séquence obtenue, il est également possible d'obtenir les différentes reprises de cycle en modifiant la situation de départ. La troisième étape consiste à générer une grande partie du code de la commande à partir des différentes séquences obtenues lors de l'étape précédente. La dernière étape est la simulation de la commande obtenue pour la valider en utilisant le modèle de la partie opérative généré à la première étape.

La première originalité de cette méthode se situe au niveau de la modélisation employée pour le système. En effet, l'auteur a choisi d'utiliser une nouvelle notion qui est « l'axe ». Un axe est défini comme étant la représentation d'une grandeur physique mesurable appartenant au système à modéliser. Ce type de modélisation ne peut donc s'appliquer que pour un système où il est toujours possible d'associer une grandeur physique pour chaque sous-partie du système. De plus, chaque axe est composé de trois ensembles $\{P, Z, M\}$:

- L'ensemble des points P. Un point est une valeur particulière (intéressante pour la commande) de la grandeur représentée par l'axe. Il représente une position de l'axe.
- L'ensemble des zones Z. Une zone est un intervalle borné continu de valeurs que cette grandeur peut prendre.
- L'ensemble des mouvements M. Un mouvement représente le passage d'un point à un autre. Il est caractérisé par un point de départ, un point d'arrivée et une durée qui correspond au temps nécessaire au mouvement.

Il est aussi à noter, qu'un axe supplémentaire est introduit dans toute modélisation pour représenter la pièce. Dans ce cas, les points deviennent les différentes étapes du traitement que va subir la pièce au cours de son voyage au sein du système. Les mouvements sont les différentes opérations qui sont effectuées sur la pièce.

La description du comportement attendu se fait ensuite en utilisant cette modélisation. En effet, cette description doit être réalisée à l'aide de cinq types de contraintes typées. Ces types de contraintes sont :

- L'exclusion de points. Cette contrainte d'exclusion permet d'interdire certaines configurations, pour des raisons de sûreté, par exemple.
- L'exclusion de mouvements. Elle est utilisée pour éviter l'exécution simultanée de plusieurs mouvements, ce qui permet par exemple d'éviter les collisions.
- La précondition de mouvement. Cette contrainte permet de stipuler que certaines propriétés doivent être vérifiées pour qu'un mouvement puisse s'exécuter.
- La condition de mouvement. Cette condition représente les invariants qui doivent être vérifiés durant l'exécution d'un mouvement.
- L'équivalence de mouvement. Cette contrainte est particulière puisqu'elle permet de faire le lien entre l'axe pièce et les autres axes en précisant quel mouvement permet d'agir sur le produit.

La seconde originalité de la méthode réside au niveau du moteur de résolution dont le but est d'obtenir une séquence optimale. Pour cela, il utilise la fonction d'un model-checker qui, dans le cas de la non vérification d'une propriété, renvoie une trace d'évolution comme contre-exemple. Afin de procéder de cette manière, les « axes » de la partie opérative sont exprimés sous la forme d'automates temporisés. L'utilisation de ce type d'automate permet d'utiliser la durée des différents mouvements. Puis les cinq types de contraintes sont traduits sous la forme de gardes, d'invariants ou de synchronisations sur ces automates. La situation initiale est choisie conformément à la situation de départ indiquée. La situation d'arrivée est, quant à elle, exprimée sous la forme d'une propriété à vérifier. Cette propriété est que le système ne doit jamais atteindre cette situation d'arrivée. Lors de la vérification, si la situation est vraiment atteignable, le model-checker indique que la propriété n'est pas vérifiée. De plus, il renvoie le chemin optimal (selon un critère donné) qui, à partir de la situation de départ, aboutit à la situation d'arrivée. Le critère choisi est la durée du chemin.

2.4.2. Exemple d'illustration

Afin d'illustrer cette modélisation, l'auteur utilise le cas d'un portique (figure 11) dont le rôle est de déplacer des flacons d'une palette vers un tapis. La modélisation de ce système est réalisée en modélisant quatre axes qui sont la pince, l'élévateur, le translateur et le flacon. Par exemple, pour

la pince, les points sont {***ouvert***, ***fermé***} et les mouvements sont {**ouvrir**(***fermé***, ***ouvert***, 1), **fermer**(***ouvert***, ***fermé***, 1)}.

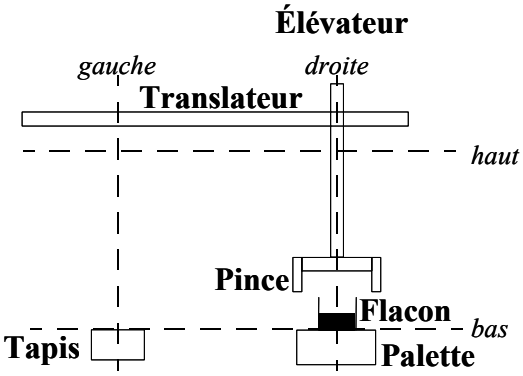


Figure 11 : L'exemple du portique

Pour décrire le comportement de ce système, 7 contraintes sont ajoutées au modèle de la partie opérative. Par exemple, une des contraintes de précondition de mouvement est : « Élévateur.**descendre** est autorisé si (Pince.***ouverte*** ET Translateur.***droite***) OU Translateur.***gauche*** ».

Lors de la recherche de chemins, le critère choisi est la minimisation de la durée séparant la situation initiale de la situation finale. Ce critère peut être utilisé car il correspond à une fonction particulière dans le logiciel UPPAAL. Ce model-checker renvoie donc la solution la plus rapide. La séquence d'opération obtenue est donnée figure 12. Ce cycle représente le mode nominal de fonctionnement.

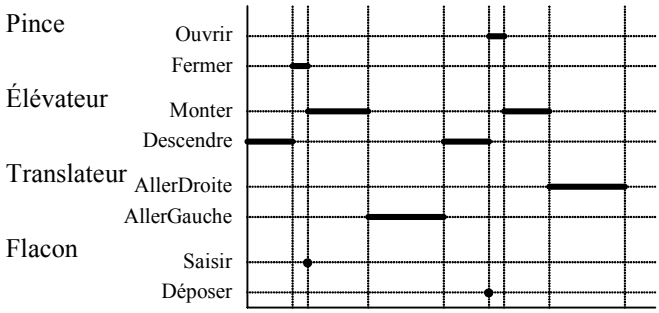


Figure 12 : Le cycle résultat pour le cas du portique

2.4.3. Analyse de l'approche

L'approche de Meftah est intéressante pour ces deux particularités qui sont le type de modélisation utilisé et la méthode employée pour synthétiser la commande d'un système. En effet, L'auteur présente une démarche de modélisation entièrement modulaire qui permet de construire facilement un modèle pour tout système. Elle se base sur la structure de la partie opérative du système. De plus, la description du comportement attendu se fait par l'ajout de contraintes qui permettent de caractériser les conditions pour qu'un mouvement puisse démarrer ainsi que les conditions à respecter pour qu'un mouvement puisse s'exécuter. Ce type de contrainte peut grandement simplifier l'expression des contraintes de sûreté.

Concernant le principe retenu pour synthétiser une commande, il est tout à fait original et présente plusieurs avantages. En effet, cette démarche permet d'obtenir toutes les séquences d'opérations qui ne font pas partie du mode nominal de fonctionnement, comme les reprises de cycle après une défaillance, les cycles de démarrage ou d'arrêt de production, ou les cycles de réinitialisation de la partie opérative.

Cette approche présente également quelques limites. La plus importante est que cette approche ne permet en aucun cas de déterminer directement la commande d'un système. En effet, la nécessité de définir une situation initiale et une situation finale interdit d'obtenir tout fonctionnement cyclique. Pour obtenir un cycle de production, il faudra donc passer par une ou plusieurs situations intermédiaires avant de revenir au point de départ. Cela pose des problèmes sur la recherche de chemin optimum. En effet, si on considère le critère de la durée la plus courte, le chemin le plus rapide entre deux situations en passant par une troisième n'a aucune raison d'être la réunion des deux chemins les plus rapides entre la situation initiale et la situation intermédiaire (choisie arbitrairement) et entre la situation intermédiaire et la situation finale.

Revenons maintenant sur les trois problèmes scientifiques présentés en début de partie :

- **La formalisation du problème** est grandement facilitée par l'introduction d'un type particulier de modélisation basé sur la partie opérative d'un système. De plus, plusieurs types de contraintes ont été prévus pour être ensuite facilement ajoutés et intégrés à la modélisation précédente.
- Concernant **la recherche des solutions** et le **choix d'une solution particulière**, ceci se réalise en même temps puisque la recherche de solution va se faire pour aboutir à la meilleure en optimisant un critère donné. Toutefois, les critères disponibles sont limités et dépendent fortement

du model-checker choisi. Il est aussi à noter qu'une seule solution est obtenue. Les autres solutions ne sont jamais calculées.

Par rapport à nos travaux, cette approche présente certaines caractéristiques que nous avons réutilisées. Premièrement, la modélisation modulaire est une idée intéressante, surtout si elle se base comme ici sur la structure même de la partie opérative. Deuxièmement, l'utilisation de contraintes données sous la forme d'invariants est compatible avec notre approche basée sur des relations algébriques. Les types de contraintes présentées sont de plus très utiles pour décrire de nombreuses spécifications de sûreté pour un système.

2.5. Synthèse de lois de commande à l'aide de l'algèbre des signaux binaires

Cette étude ne saurait être complète sans une analyse des travaux du LURPA antérieurs à cette thèse dans le domaine de la synthèse des lois de commande. Depuis plusieurs années, le LURPA se propose d'obtenir les lois de commande d'un SED logique de manière algébrique [ROU04] [ROU06] [MED07].

Dans cette partie, nous allons rapidement présenter les principales caractéristiques de ces travaux antérieurs et préciser quelles en sont les limites.

2.5.1. Caractéristiques de l'approche proposée

Cette méthode de synthèse a été spécifiquement développée pour obtenir les lois de commande d'un SED logique. Le cadre mathématique support de cette approche est une algèbre de Boole spécifique dont les éléments sont des signaux binaires. Cette algèbre conçue initialement pour formaliser la notion d'évènement [ROU93] utilisée en Grafcet a également été la base de travaux de vérification de programmes pour des contrôleurs logiques par Theorem Proving [ROU02].

Partant d'éléments de spécification de la loi de commande donnés sous la forme de relations entre signaux binaires, cette méthode permet d'obtenir une loi de commande par résolution successive de systèmes d'équations entre les signaux binaires (figure 13).

Sur le plan mathématique, la synthèse de la loi de commande à implanter dans un contrôleur est obtenue grâce à une résolution successive de systèmes d'équations. Ne disposant pas de moyens permettant de résoudre une équation à plusieurs inconnues, le problème global est scindé en *n* pro-

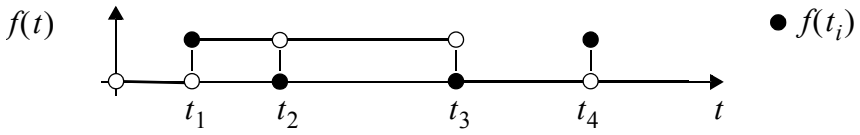


Figure 13 : Représentation graphique d'un signal binaire

blèmes locaux (un par inconnue) traités de manière séquentielle. Pour chaque problème local, la cohérence des éléments de spécification est analysée et une solution locale est déterminée. En résolvant chaque système d'équations selon le graphe de dépendance des inconnues proposées par le concepteur, il est possible d'obtenir une solution globale.

Les résultats présentés dans [ROU06] montrent que cette méthode peut s'appliquer sur des systèmes de taille significative et qu'une partie des éléments de spécification peut être obtenue automatiquement à partir des caractéristiques technologiques des composants du système physique à commander.

Le détail de cette méthode de synthèse et le traitement complet d'un exemple (commande d'un portail automatique) peuvent être trouvés dans [ROU06].

2.5.2. Analyse de l'approche

Dans le cas de la synthèse de lois de commande à l'aide de l'algèbre des signaux binaires, les points forts de cette approche sont également la source de ses limites actuelles.

La principale force de cette approche est de disposer d'un cadre mathématique permettant de s'intéresser à des aspects temporisés car cette algèbre dispose d'opérateurs de temporisation permettant d'exprimer des contraintes temporelles entre les signaux binaires. Il faut cependant signaler que les capacités actuelles de résolution sur cette algèbre ne permettent de traiter qu'un sous-ensemble réduit de contraintes temporelles. De plus, la spécificité du cadre mathématique (représentation continue du temps) ne permet pas d'intégrer des éléments de spécification donnés à l'aide d'une représentation discrète du temps. Pour pouvoir intégrer ce type de spécification, il faut donc changer d'algèbre. C'est ce que nous proposons de faire dans cette thèse.

Pour la dernière fois, positionnons cette technique par rapport aux trois problèmes scientifiques présentés en introduction de l'étude bibliographique :

- **La formalisation du problème**. Pour résoudre ce problème, les auteurs ont donné plusieurs exemples types de spécifications données sous la forme de phrases en langage naturel et ils ont

fourni pour chacune d'entre elles une formalisation. Bien que cette démarche ne permette pas de traiter de manière exhaustive toutes les spécifications, elle permet déjà d'en formaliser une grande partie.

- **La recherche de l'ensemble des solutions** et **le choix d'une solution particulière**. La technique de résolution se fait par le traitement de plusieurs problèmes locaux. Cette démarche de résolution a pour limite qu'une seule solution particulière est obtenue et que cette solution dépend entièrement de l'ordre de résolution choisi.

2.6. Bilan de l'étude bibliographique

Dans cette partie, plusieurs approches de synthèse ont été étudiées afin de voir si l'une d'entre elles correspondait à nos besoins. Malheureusement, aucune des approches étudiées ne correspond complètement à nos attentes, en résumé :

- La théorie de la supervision. Bien que cette théorie fut la première qui inspira toutes les autres, le point de vue considéré n'est pas le nôtre. Le système est vu comme déjà commandé, et c'est un superviseur qui est recherché.
- L'utilisation de polynômes. Bien que cette modélisation n'ait pas été créée pour la synthèse de commande, elle a su être utilisée dans ce sens. Toutefois, la modélisation n'est pas bien adaptée pour décrire la spécification d'un système, ce qui rend la formalisation du cahier des charges difficile. De plus, la technique de résolution n'est pas réellement développée.
- Le couplage fonctionnel. Cette approche est assez proche des objectifs que nous cherchons à atteindre. Toutefois, elle n'est pas suffisamment formalisée ni automatisable. Cette méthode comporte encore une importante partie manuelle qui nécessite le savoir d'un expert pour aboutir à la commande recherchée.
- La recherche de séquences. Bien que la recherche de séquences apporte de nombreux avantages, en particulier pour trouver des cycles de réinitialisation ou de reprise de cycle, la nécessité de fixer une situation initiale et finale peut poser des problèmes dans le cas de comportements peu séquentiels.

Bien qu'aucune méthode étudiée ne corresponde à nos attentes, elles ont toutes des caractéristiques très intéressantes que nous avons exploitées lors du développement de notre propre méthode de synthèse. Ces originalités sont :

- Pour la théorie de la supervision. Les caractéristiques intéressantes sont la notion même de synthèse d'une partie du système ainsi que la méthode retenue, qui consiste à partir d'un comportement complètement libre du processus pour le réduire, par interdiction d'évènements contrôlables, au comportement le plus permissif qui vérifie les spécifications.
- Pour l'utilisation de polynômes. Le point intéressant est le type de synthèse réalisée. L'approche développée est entièrement algébrique et ne nécessite donc pas la manipulation directe de tous les états possible du système. Ce type d'approche supprime complètement le problème d'explosion combinatoire du nombre d'états à manipuler.
- Pour le couplage fonctionnel. La particularité de cette méthode est la liberté laissée au spécificateur au niveau des données d'entrée de la méthode. C'est la méthode la moins contraignante à ce niveau-là. De plus, le résultat produit est directement utilisable avec la plupart des automates programmables industriels.
- Pour la recherche de séquences. Cette méthode présente une approche modulaire ingénieuse qui ne retient que les éléments importants et permet donc de traiter de plus gros système plus facilement.

3. Caractéristiques de notre technique de synthèse

La méthode de synthèse présentée dans ce mémoire de thèse a été développée pour obtenir la loi de commande à implanter dans le contrôleur industriel en charge de piloter un processus physique discret. Dans le cadre de cette thèse, nous nous limiterons aux SED pour lesquels les interactions avec la commande se font uniquement par l'intermédiaire de variables logiques (cf. figure 1). Nous ferons également l'hypothèse que la loi de commande recherchée est déterministe et non temporisée.

Pour décrire le comportement de cette famille de contrôleurs, différents modèles de représentation peuvent être envisagés comme les automates à états finis, les Réseaux de Petri ou le Grafcet. Dans le cas de notre approche, nous allons exploiter le caractère logique des entrées/sorties du contrôleur et décrire son comportement à l'aide de fonctions booléennes (cf. partie 3.1.).

En nous appuyant sur cette hypothèse de travail, synthétiser la loi de commande d'un SED logique consistera donc à établir les fonctions booléennes qui décrivent son comportement. Nous

montrerons dans la partie 3.2., au travers d'un exemple de taille élémentaire, que le problème de la synthèse automatique de fonctions booléennes est cependant un problème complexe.

Dans la dernière section sera décrite l'approche que nous proposons pour la synthèse de la loi de commande d'un SED logique.

3.1. Modèle retenu pour décrire le comportement d'un système logique

Le modèle mathématique que nous proposons d'utiliser pour décrire la loi de commande d'un SED logique exploite le caractère logique des entrées/sorties du contrôleur. Le contrôleur en fonctionnement sera vu comme un système logique à m entrées et n sorties (figure 14).

Figure 14 : Contrôleur logique à m entrées et n sorties

Que le comportement déterministe de ce contrôleur soit séquentiel ou combinatoire, il est toujours possible d'exprimer ce comportement à l'aide de fonctions logiques.

Pour éviter tout risque d'ambiguïté, nous allons maintenant préciser dans le détail comment nous allons exprimer les comportements combinatoires et séquentiels d'un système logique à l'aide de fonctions booléennes.

3.1.1. Cas des systèmes logiques combinatoires

Considérons le système logique à m entrées et n sorties présenté sur la figure 15.

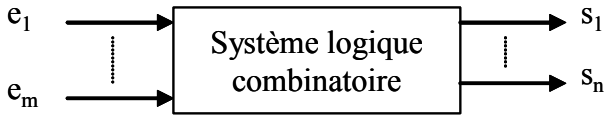


Figure 15 : Système logique combinatoire à m entrées et n sorties

Le comportement de ce système est dit combinatoire si, à chaque instant k, la valeur de chaque sortie s_i dépend uniquement des valeurs des entrées e_j à ce même instant. Nous noterons $e_j[k]$ et $s_i[k]$ la valeur à l'instant k des entrées e_j et s_i.

Réaliser la synthèse de ce système logique à partir de son cahier des charges consiste à déterminer, pour chacune des n sorties, la fonction qui définit la valeur de cette sortie pour chacune des combinaisons différentes des valeurs des entrées.

Dans le cas particulier des systèmes logiques combinatoires, comme les seules valeurs possibles de leurs entrées et de leurs sorties sont 0 ou 1, ces systèmes possèdent les caractéristiques suivantes :

- Pour un système à m entrées, le nombre de combinaisons différentes des valeurs des entrées est de 2^m.
- Pour chacune des 2^m combinaisons des entrées, comme la valeur d'une sortie peut être 0 ou 1, il est donc possible de définir $2^{(2^m)}$ fonctions de sortie différentes.

Le modèle mathématique le plus couramment utilisé pour décrire une fonction de sortie est celui des fonctions booléennes de dimension m à valeur booléenne :

$$\begin{array}{l} F: \quad \mathbb{B}^m \to \mathbb{B} \\ \quad (b_1, \ldots, b_m) \mapsto F(b_1, \ldots, b_m) \end{array} \quad \text{avec } \mathbb{B} = \{0, 1\}$$

Pour un système logique à m entrées et n sorties, il existe $(2^{(2^m)})^n$ comportements combinatoires différents. Réaliser la synthèse d'un tel système consiste à déterminer, pour chaque sortie s_i, quelle est la fonction booléenne $F_i(e_1[k], \ldots, e_m[k])$ qui permet de définir sa valeur $s_i[k]$ à l'instant k à partir des valeurs $e_1[k], \ldots, e_m[k]$ des entrées $e_1, \ldots, e_m$ à ce même instant k.

$$\begin{cases} s_1[k] = F_1(e_1[k], \ldots, e_m[k]) \\ \ldots \\ s_n[k] = F_n(e_1[k], \ldots, e_m[k]) \end{cases}$$

3.1.2. Cas des systèmes logiques séquentiels

Le comportement d'un système logique est dit séquentiel lorsque la valeur $s_i[k]$ d'au moins une de ses sorties s_i ne peut pas être déterminée uniquement à partir des seules valeurs $e_j[k]$ des entrées à ce même instant k car elle dépend également de la valeur antérieure de certaines entrées.

Si la connaissance de toute l'histoire de l'évolution des entrées depuis l'instant initial peut permettre le calcul de la valeur des sorties quel que soit l'instant k considéré, il est d'usage de faire référence à l'état interne du système. Dans le cas des systèmes logiques, cet état est généralement décrit à l'aide d'un ensemble de variables logiques dites internes.

Considérons le système logique séquentiel proposé sur la figure 16 comportant m entrées, n sorties et l variables internes notées x.

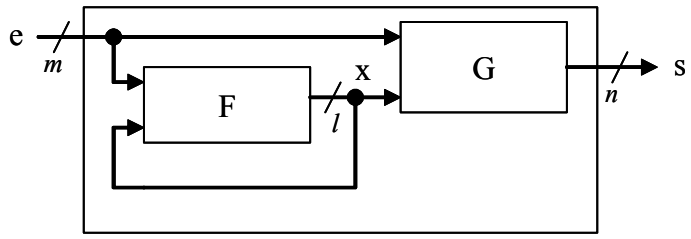


Figure 16 : Système logique séquentiel à m entrées, n sorties et l variables internes

Ce système est constitué de deux systèmes logiques combinatoires F et G permettant de décrire respectivement :

- l'état interne courant du système (la valeur des variables internes à l'instant k étant fonction de la valeur de ces variables à l'instant $(k\text{-}1)$ et des entrées à l'instant k, le système combinatoire F décrit par récurrence [ZAH87] le comportement de l'état interne du système) :

$$\begin{cases} x_1[k] = F_1(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ \ldots \\ x_l[k] = F_l(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \end{cases}$$

- la valeur courante des sorties :

$$\begin{cases} s_1[k] = G_1(e_1[k], \ldots, e_m[k], x_1[k], \ldots, x_l[k]) \\ \ldots \\ s_n[k] = G_n(e_1[k], \ldots, e_m[k], x_1[k], \ldots, x_l[k]) \end{cases}$$

La connaissance des systèmes combinatoires F et G, ainsi que de la valeur initiale de chaque variable interne (à l'instant $k = 0$) est suffisante pour déterminer le comportement du système en fonction des valeurs successives des entrées.

Le système séquentiel de la figure 16 peut donc être modélisé à l'aide de $(l + n)$ fonctions booléennes de dimension $(m + l)$:

$$\begin{cases} x_1[k] = F_1(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ \ldots \\ x_l[k] = F_l(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ s_1[k] = G_1(e_1[k], \ldots, e_m[k], x_1[k], \ldots, x_l[k]) \\ \ldots \\ s_n[k] = G_n(e_1[k], \ldots, e_m[k], x_1[k], \ldots, x_l[k]) \end{cases}$$

Si cette représentation a l'avantage d'être celle couramment retenue pour décrire le comportement d'un système logique séquentiel, elle a pour inconvénient de faire référence à deux types de fonctions booléennes : les fonctions des variables $(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1])$ et celles des variables $(e_1[k], \ldots, e_m[k], x_1[k], \ldots, x_l[k])$.

Pour décrire un comportement séquentiel à l'aide de fonctions booléennes, d'autres représentations équivalentes sont possibles. Dans le modèle de la figure 16, il avait été choisi de déterminer la valeur des n sorties à l'instant k à partir de la valeur des m entrées à l'instant k et de la valeur des l variables internes à l'instant k. Comme la valeur des variables internes est obtenue à partir de la valeur des m entrées à ce même instant k et de la valeur des l variables internes à l'instant $(k\text{-}1)$, il est également possible de choisir de déterminer directement la valeur des n sorties à l'instant k à partir de la valeur des m entrées à l'instant k et de la valeur des l variables internes à l'instant $(k\text{-}1)$.

C'est ce choix que nous allons retenir dans cette thèse afin de pouvoir travailler sur un seul type de fonctions booléennes exprimées à l'aide des variables $(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1])$.

Dans la représentation décrite à la figure 17, le comportement séquentiel est décrit à l'aide d'un seul système logique combinatoire bouclé H.

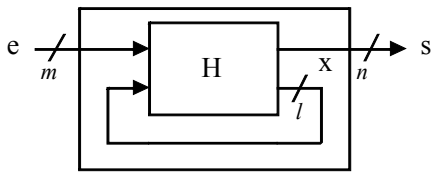


Figure 17 : Système logique séquentiel à m entrées, n sorties et l variables internes (représentation utilisée dans notre approche)

Ce système possède $(m + l)$ entrées et $(n + l)$ sorties. Il est parfaitement défini par la donnée des $(n + l)$ fonctions booléennes de dimension $(m + l)$ qui permettent de définir la valeur de chacune de ces sorties.

$$\begin{cases} s_1[k] = H_1(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ \ldots \\ s_n[k] = H_n(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ x_1[k] = H_{n+1}(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \\ \ldots \\ x_l[k] = H_{n+l}(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \end{cases}$$

Par construction, il est toujours possible de passer d'une modélisation comme celle proposée figure 16 à la modélisation donnée figure 17 par la transformation suivante :

- pour la détermination des variables internes $(i \leq l)$:

$$H_{n+i}(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) = F_i(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1])$$

- pour la détermination des fonctions de sorties $(i \leq n)$:

$$H_i(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) = G_i(e_1[k], \ldots, e_m[k], F_1(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]), \ldots, F_l(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]))$$

Dans le cas des systèmes séquentiels, il est possible d'obtenir ces fonctions booléennes récurrentes directement à partir d'un modèle à états. Dans le cas du Grafcet, ces fonctions booléennes récurrentes correspondent à l'interprétation algébrique du Grafcet [MAC06].

3.2. Difficultés à surmonter lors de la synthèse de fonctions booléennes

Dans la section précédente, nous avons montré que la synthèse de la loi de commande d'un SED logique peut se ramener à synthétiser les fonctions booléennes qui décrivent son comportement. Bien que les fonctions booléennes soient un modèle mathématique largement utilisé, les concepteurs de systèmes logiques ne disposent pas d'une méthode systématique permettant d'obtenir ces fonctions booléennes à partir des attentes données dans un cahier des charges. Nous allons montrer au travers du traitement d'un exemple de taille élémentaire pourquoi le problème de la synthèse automatique de fonctions booléennes est un problème complexe.

3.2.1. Exemple introductif

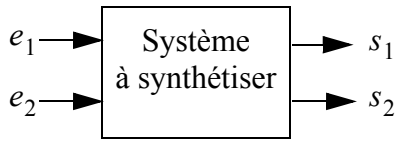
Cet exemple a été choisi pour sa simplicité. Ainsi, sa petite taille nous permet de le traiter manuellement. Nous nous sommes restreints à un problème purement combinatoire et le cahier des charges est volontairement très simple pour que son interprétation ne soit pas source d'ambiguïtés. Dans le cadre de ce mémoire, cet exemple sera utilisé à deux reprises :

- Le premier traitement (dans cette section) sera l'occasion de mettre en évidence les difficultés qui doivent être surmontées lors de la synthèse automatique de fonctions booléennes.
- Le second traitement sera présenté au chapitre 3. Il permettra de mettre en avant le réel potentiel de notre approche.

3.2.1.1. Présentation du problème

Considérons le système logique combinatoire présenté sur la figure 18a. Ce système comporte deux entrées et deux sorties pour lesquelles le comportement combinatoire attendu est décrit figure 18b.

a) Entrées/sorties du système



b) Cahier des charges

- P1 : Lorsque l'entrée e_1 est à 1, la sortie s_1 est à 1.
- P2 : Lorsque l'entrée e_2 est à 1, la sortie s_2 est à 0.
- P3 : Les sorties s_1 et s_2 ne sont jamais simultanément à 1.

Figure 18 : Système logique combinatoire à synthétiser

Réaliser la synthèse de ce système logique consiste à identifier le couple de fonctions booléennes de dimension 2 qui satisfait les trois propositions du cahier des charges. Chaque proposition correspond à une contrainte que doivent respecter les valeurs des sorties s_1 et s_2 en fonction des valeurs prises par les entrées e_1 et e_2. Bien que ces propositions soient données sous forme de relations entre des variables logiques, il est nécessaire de les considérer comme des contraintes sur les fonctions booléennes de dimension 2 recherchées car ces relations entre les variables logiques e_1, e_2, s_1 et s_2 doivent être vérifiées pour chaque combinaison distincte (e_1, e_2) des valeurs des entrées.

Pour ce système logique combinatoire à 2 entrées, il existe 16 (en fait 2^{2^2}) fonctions booléennes de dimension 2 différentes permettant de définir chacune des 2 sorties. Le tableau 2 regroupe les tables de vérité de chacune de ces 16 fonctions booléennes.

e_1	e_2	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Tableau 2 : Tables de vérité des fonctions booléennes de dimension 2

$F_0 = 0$ $F_1 = \neg e_1 \wedge \neg e_2$ $F_2 = \neg e_1 \wedge e_2$ $F_3 = \neg e_1$

$F_4 = e_1 \wedge \neg e_2$ $F_5 = \neg e_2$ $F_6 = \neg e_1 \wedge e_2 \vee e_1 \wedge \neg e_2$ $F_7 = \neg e_1 \vee \neg e_2$

$F_8 = e_1 \wedge e_2$ $F_9 = \neg e_1 \wedge \neg e_2 \vee e_1 \wedge e_2$ $F_{10} = e_2$ $F_{11} = \neg e_1 \vee e_2$

$F_{12} = e_1$ $F_{13} = e_1 \wedge \neg e_2$ $F_{14} = e_1 \vee e_2$ $F_{15} = 1$

Dans le cas du système logique présenté figure 18a, il existe donc 256 (16^2) couples de fonctions booléennes potentiellement solutions.

3.2.1.2. Approche classique

Classiquement, le choix d'une de ces 256 possibilités est réalisé manuellement. Le concepteur utilise toute son expertise pour proposer une solution. La durée de cette conception comme la qualité de la solution dépend entièrement de l'expertise du concepteur. Vue la large dépendance du résultat trouvé avec la compétence du concepteur, il est fréquent ensuite d'analyser la solution fournie afin de s'assurer qu'elle répond bien aux attentes du cahier des charges.

3.2.1.3. Recherche exhaustive des solutions

Dans notre cas, comme la dimension du problème est suffisamment réduite, il est encore possible de réaliser une étude exhaustive des 256 solutions potentielles pour ne retenir que les seuls couples de fonctions booléennes qui satisfont les trois propositions du cahier des charges.

Pour éviter de devoir analyser les trois propositions simultanément, nous allons procéder par élimination. Les 256 couples de fonctions booléennes seront considérés initialement comme solutions potentielles. Lors de l'étude de chaque proposition, seront éliminées les expressions qui ne satisfont pas la contrainte exprimée.

- Etude de la proposition P1 : Lorsque l'entrée e_1 est à 1 , la sortie s_1 est à 1 .
  La transcription de P1 par une table de vérité de la fonction booléenne S_1 est donnée tableau 3.

e_1	e_2	S_1
0	0	?
0	1	?
1	0	1
1	1	1

Tableau 3 : Traduction de la proposition P1

Parmi les 16 fonctions booléennes de dimension 2, seules les fonctions F_{12} , F_{13} , F_{14} et F_{15} ont une table de vérité qui satisfait cette contrainte.

- Etude de la proposition P2 : Lorsque l'entrée e_2 est à 1 , la sortie s_2 est à 0 .
  La transcription de P2 par une table de vérité de la fonction booléenne S_2 est donnée tableau 4.

e_1	e_2	S_2
0	0	?
0	1	0
1	0	?
1	1	0

Tableau 4 : Traduction de la proposition P2

Parmi les 16 fonctions combinatoires de dimension 2, seules les fonctions F_0 , F_1 , F_4 et F_5 ont une table de vérité qui satisfait cette contrainte.

- Etude de la proposition P3 : Les sorties s_1 et s_2 ne sont jamais simultanément à 1 .
  Cette proposition contraint les valeurs respectives des sorties s_1 et s_2 . Contrairement aux pro-

positions précédentes, les fonctions booléennes S_1 et S_2 ne peuvent pas être caractérisées indépendamment l'une de l'autre. Il est donc nécessaire d'étudier tous les couples de fonctions booléennes S_1 et S_2 solutions et d'analyser la valeur respective des fonctions pour chacune des combinaisons (e_1, e_2). Seuls les couples de fonctions booléennes S_1 et S_2 dont les tables de vérité ne présentent pas de 1 pour la même combinaison (e_1, e_2) sont conservés. Dans le tableau 5, sont pointés les 81 couples de fonctions booléennes S_1 et S_2 qui satisfont la proposition 3.

S_2 \ S_1	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
F_0	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
F_1	OK		OK		OK		OK		OK		OK		OK		OK	
F_2	OK	OK			OK	OK			OK	OK			OK	OK		
F_3	OK				OK				OK				OK			
F_4	OK	OK	OK	OK					OK	OK	OK	OK				
F_5	OK		OK						OK		OK					
F_6	OK	OK							OK	OK						
F_7	OK								OK							
F_8	OK	OK	OK	OK	OK	OK	OK	OK								
F_9	OK		OK		OK		OK									
F_{10}	OK	OK			OK	OK										
F_{11}	OK				OK											
F_{12}	OK	OK	OK	OK												
F_{13}	OK		OK													
F_{14}	OK	OK														
F_{15}	OK															

Tableau 5 : Couples de fonctions booléennes satisfaisant la proposition 3

- Etude de l'ensemble des propositions

  Dans le tableau 6, sont pointés les 6 couples de fonctions booléennes S_1 et S_2 qui satisfont l'ensemble des 3 propositions. Ce tableau a été obtenu à partir du tableau 5 en ne conservant

que les colonnes correspondant aux fonctions F_{12}, F_{13}, F_{14} et F_{15} (Proposition 1) et les lignes correspondant aux fonctions F_0, F_1, F_4 et F_5 (Proposition 2).

S_2 \ S_1	F_{12}	F_{13}	F_{14}	F_{15}
F_0	Ok	Ok	Ok	Ok
F_1	Ok		Ok	
F_4				
F_5				

Tableau 6 : Couples de fonctions booléennes satisfaisant l'ensemble des propositions

La définition de ces 6 couples de fonctions booléennes à l'aide d'expressions des variables booléennes e_1 et e_2 est donnée ci-dessous :

$$(F_{12}, F_0) \quad \begin{cases} S_1(e_1, e_2) = e_1 \\ S_2(e_1, e_2) = 0 \end{cases}$$

$$(F_{12}, F_1) \quad \begin{cases} S_1(e_1, e_2) = e_1 \\ S_2(e_1, e_2) = \neg e_1 \wedge \neg e_2 \end{cases}$$

$$(F_{13}, F_0) \quad \begin{cases} S_1(e_1, e_2) = e_1 \wedge \neg e_2 \\ S_2(e_1, e_2) = 0 \end{cases}$$

$$(F_{14}, F_0) \quad \begin{cases} S_1(e_1, e_2) = e_1 \vee e_2 \\ S_2(e_1, e_2) = 0 \end{cases}$$

$$(F_{14}, F_1) \quad \begin{cases} S_1(e_1, e_2) = e_1 \vee e_2 \\ S_2(e_1, e_2) = \neg e_1 \wedge \neg e_2 \end{cases}$$

$$(F_{15}, F_0) \quad \begin{cases} S_1(e_1, e_2) = 1 \\ S_2(e_1, e_2) = 0 \end{cases}$$

3.2.1.4. Choix d'une solution

L'analyse exhaustive qui vient d'être menée a montré que le problème étudié admettait six couples de fonctions booléennes solutions. En l'absence d'informations supplémentaires, le concepteur peut choisir arbitrairement l'un d'entre eux. Il est cependant possible d'optimiser ce choix en fonction de critères complémentaires comme par exemple :

- Maximiser le nombre de 1 dans la table de vérité de S_1, ce qui revient à choisir pour S_1 la solution la plus « grande » ou la plus « permissive » (au sens qui inclut le plus grand nombre de combinaisons d'entrée) :

$$\begin{cases} S_1(e_1, e_2) = 1 \\ S_2(e_1, e_2) = 0 \end{cases}$$

- Maximiser le nombre de 1 dans la table de vérité de S_2 :

$$\begin{cases} S_1(e_1, e_2) = e_1 \\ S_2(e_1, e_2) = \neg e_1 \wedge \neg e_2 \end{cases} \text{ ou } \begin{cases} S_1(e_1, e_2) = e_1 \vee e_2 \\ S_2(e_1, e_2) = \neg e_1 \wedge \neg e_2 \end{cases}$$

- Minimiser le nombre de 1 dans la table de vérité de S_1, c'est-à-dire choisir la solution la plus « petite » ou la moins « permissive » :

$$\begin{cases} S_1(e_1, e_2) = e_1 \\ S_2(e_1, e_2) = 0 \end{cases} \text{ ou } \begin{cases} S_1(e_1, e_2) = e_1 \\ S_2(e_1, e_2) = \neg e_1 \wedge \neg e_2 \end{cases}$$

Le choix de la solution à implanter parmi les différentes solutions possibles est un problème d'optimisation qui mérite une étude à part entière. Dans ce mémoire, le choix de la solution optimale parmi les différentes solutions possibles ne sera qu'abordé sans être complètement traité.

3.2.2. Difficultés à surmonter lors d'une synthèse de fonctions booléennes

Réaliser la synthèse d'un système logique combinatoire est un problème complexe qui repose encore aujourd'hui exclusivement sur l'expérience de l'expert qui traite le problème. Cette situation est principalement due à l'absence de méthodes pour trouver le résultat.

Lors du traitement de cet exemple introductif, une approche exhaustive a permis de déterminer quelles étaient les solutions du problème posé. Cette approche, lorsqu'elle nécessite d'énumérer les solutions potentielles, n'est cependant valide que pour des problèmes de petite taille. Pour s'en convaincre, il suffit de calculer le nombre de comportements combinatoires différents pour un système comportant seulement 4 entrées et 3 sorties :

$$2^{3 \times (2^4)} = 2^{3 \times 16} = 2^{48} \approx 2{,}8 \times 10^{14} \text{ comportements}$$

- La première difficulté rencontrée lors du traitement de cet exemple est **la transcription formelle des informations données dans le cahier des charges**. Pour traiter cet exemple, nous avons eu recours à des tables de vérité des fonctions booléennes recherchées pour exprimer les

contraintes stipulées. Lorsque la contrainte ne portait que sur une seule sortie, il a été possible de compléter partiellement la table de vérité de la fonction combinatoire associée à cette sortie en précisant la valeur attendue (0 ou 1) pour certaines combinaisons des entrées. Dans le cas de la proposition 3, cette approche ne fut malheureusement pas possible.

Pour dérouler une approche de synthèse d'un système logique combinatoire, il est nécessaire de pouvoir décrire avec rigueur chacune des contraintes que l'on peut rencontrer dans un cahier des charges. La technique de représentation de ces contraintes doit de plus permettre de les combiner entre elles.

- La deuxième difficulté à surmonter est **la recherche de l'ensemble des solutions**. La procédure à suivre lors de cette étape doit être parfaitement définie afin de permettre son automatisation. Lorsque le problème admet plusieurs solutions, la technique proposée ne doit pas se contenter de donner la première solution identifiée mais l'ensemble de ces solutions pour permettre au concepteur de choisir celle qui lui convient le mieux. Dans le cas où le problème posé n'admet pas de solutions, il est intéressant de connaître les raisons de cette impossibilité. Dans tous les cas, il est nécessaire de conduire une analyse exhaustive, sans avoir explicitement à faire l'inventaire des comportements combinatoires potentiellement solution.

- La troisième difficulté à surmonter est **le choix de la solution**. Présenter au concepteur les solutions du problème sous la forme d'une énumération ne lui permet pas de faire ce choix dans les meilleures conditions dès que le nombre de solutions est important car le concepteur n'a pas de moyens pour les comparer. Il sera donc intéressant de privilégier une description symbolique de celles-ci.

3.3. Proposition d'une méthode de synthèse

La méthode de synthèse que nous proposons a été spécifiquement développée pour obtenir la commande d'un SED logique. L'objectif de notre approche est de donner à un concepteur de tels systèmes les moyens d'obtenir de manière automatique les lois de commande à implanter dans le contrôleur logique à partir de tous les fragments de spécification qu'il a à sa disposition (figure 19).

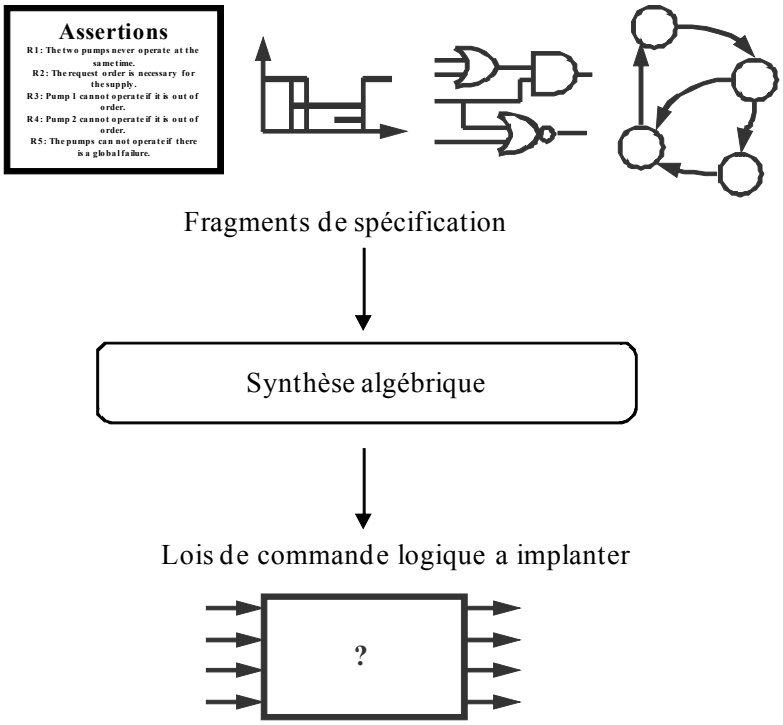


Figure 19 : Méthode de synthèse proposée

Obtenir automatiquement les lois de commande à implanter dans le contrôleur logique à partir d'un ensemble de fragments de spécification nécessite de disposer d'un cadre mathématique rigoureux permettant :

- D'exprimer de manière non ambiguë tous les fragments de spécification à la disposition du concepteur,
- De les analyser pour en détecter les éventuelles incohérences,
- D'établir la ou les solutions lorsqu'elles existent,
- De donner au concepteur les moyens de choisir parmi les différentes solutions possibles la solution particulière qu'il désirera implanter.

Pour la classe de SED étudiée, nous nous proposons d'obtenir la commande en résolvant de manière analytique un système d'équations entre des fonctions booléennes. Ce système d'équations, qui est la représentation de la spécification du comportement attendu de la commande, peut en fonction des cas :

- **Ne pas avoir de solution :** cette configuration est due à la présence de fragments de spécification incohérents entre eux. La condition d'incohérence retournée par notre méthode permet au concepteur d'identifier les fragments de spécification qui sont la source de cette incohérence.
- **N'admettre qu'une seule solution :** ce cas, qui dans la pratique s'avère assez rare, se rencontre lorsque les exigences du cahier des charges sont telles qu'il n'existe qu'une seule loi de

commande solution du problème. Notre méthode permet d'obtenir automatiquement cette loi de commande.

- **Admettre plusieurs solutions :** ce cas, qui est le plus courant, se rencontre lorsque toutes les exigences du cahier des charges peuvent être garanties non pas par une seule loi de commande mais par plusieurs. La méthode que nous proposons permet non seulement de déterminer toutes ces solutions mais également de toutes les exprimer à l'aide d'une seule forme paramétrique. Le concepteur peut alors choisir la solution qui lui convient le mieux en fixant la valeur de chacun des paramètres de la solution paramétrique. Il peut également ajouter de nouveaux éléments à son cahier des charges et procéder à une nouvelle synthèse.

3.3.1. Caractéristiques des données de départ

Pour qu'une méthode de synthèse automatique soit opérationnelle, il ne suffit pas de disposer d'un cadre mathématique performant. Il faut que le travail amont nécessaire à l'utilisation de ce cadre ne soit pas plus contraignant que le travail de synthèse qu'il est sensé remplacer. Conscients de cette exigence opérationnelle, nous avons tenu à ce que le concepteur puisse exprimer tous ses besoins avec le formalisme qui lui convient le mieux pour chacun des besoins. Dans la pratique, les fragments de spécification que le concepteur a à sa disposition peuvent être des exigences fonctionnelles, des contraintes de sûreté à respecter et éventuellement des éléments de solution. Pour les exprimer, il peut avoir recours à des modèles à états (comme proposé dans la Supervisory Control Theory), des équations logiques (comme proposé dans l'approche par signaux binaires) ou des diagrammes temporels d'enchaînement (comme proposé par Lohmann). Notre approche autorise l'expression des besoins dans des modèles différents à la condition que le comportement exprimé dans ces modèles puisse être reformulé de manière univoque à l'aide de relations entre des fonctions booléennes. Nous montrerons au travers des exemples présentés au chapitre 3, et plus particulièrement en traitant le troisième exemple, que le cadre mathématique que nous employons permet bien de représenter des fragments de spécification donnés originellement dans des formalismes différents.

Notre approche autorise le concepteur d'une loi de commande à spécifier la forme générale de la loi de commande qu'il recherche en injectant une solution partielle pour le problème à synthétiser. Dans un tel cas, il ne sera plus recherché la loi de commande en elle-même mais les éléments qui n'étaient pas définis dans cette solution. Nous reviendrons plus en détail sur cette possibilité au chapitre 3.

3.3.2. Les étapes de la démarche

L'approche que nous proposons comporte 4 étapes (cf. figure 20) :

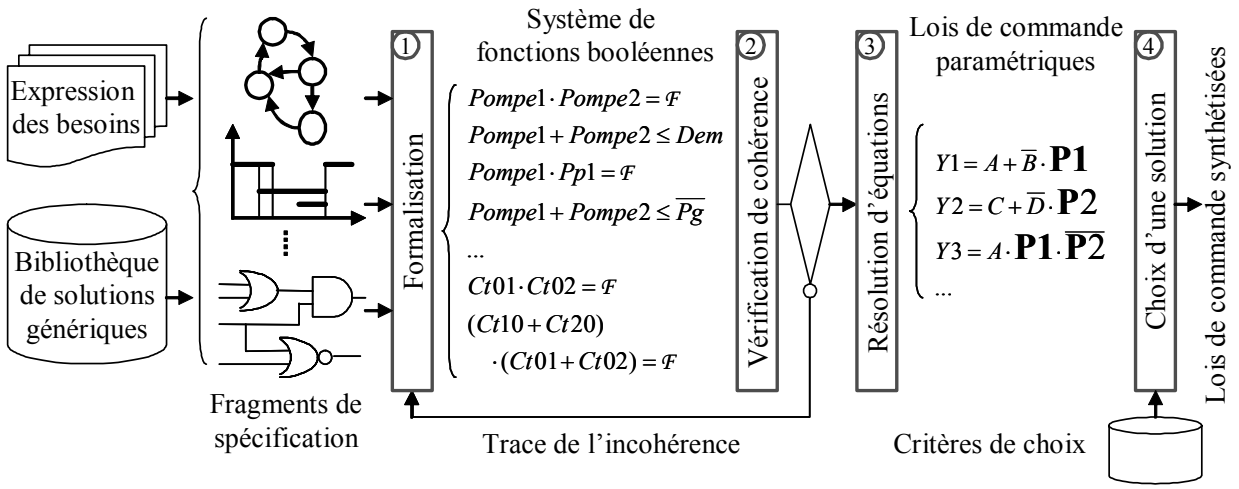


Figure 20 : Les quatre temps de la méthode

- **La formalisation des fragments de la spécification sous forme de relations entre des fonctions booléennes :** Cette étape est à la charge du concepteur. Nous souhaitons cependant lui apporter à terme le maximum d'assistance en lui permettant d'utiliser des bibliothèques de spécifications génériques ou lui traduire automatiquement sous forme de relations entre des fonctions booléennes les spécifications qu'il propose dans un autre formalisme. À l'issue de cette étape, l'ensemble des fragments de spécifications contenus dans le cahier des charges et des éléments de solutions qu'a souhaité donner le concepteur forme un système d'équations entre des fonctions booléennes.
- **La vérification de la cohérence des fragments de spécification :** Cette étape, qui sera réalisée automatiquement à partir du système d'équations entre des fonctions booléennes, permettra de s'assurer que le système d'équations admet des solutions. Dans le cas contraire, les informations retournées au concepteur lui permettront de déterminer quels sont les fragments de spécification qui présentent des incohérences mutuelles.
- **La résolution du système d'équations :** Cette étape est réalisée automatiquement à partir des résultats théoriques présentés au chapitre 2. Nous sommes capables de calculer analytiquement toutes les solutions d'un système d'équations entre des fonctions booléennes et d'exprimer ces solutions à l'aide d'une forme paramétrique.

- **Le choix d'une solution :** À de très rares exceptions près, la solution d'un système d'équations entre des fonctions booléennes n'est pas unique. Le dernier temps de notre méthode consiste à choisir parmi les différentes solutions proposées quelle sera la solution retenue. Dans le cadre de ce travail de thèse, cette étape sera laissée entièrement à la charge du concepteur. Elle consiste à fixer une valeur pour chacun des paramètres de la forme paramétrique obtenue précédemment. Il apparaît que la forme paramétrique que nous proposons permet bien de mettre en place des stratégies d'optimisation mais le problème de l'optimisation est un problème suffisamment complexe en soi pour mériter une étude à part entière.

4. Bilan

La démarche que nous proposons repose entièrement sur notre capacité à résoudre un système d'équations entre des fonctions booléennes. Cette possibilité a été obtenue en travaillant au sein d'une structure algébrique particulière que sont les algèbres de Boole. Les résultats mathématiques nécessaires à la résolution d'un système d'équations entre des fonctions booléennes sont présentés au chapitre 2. Le chapitre 3 sera consacré à l'utilisation de ces résultats pour la synthèse des lois de commande d'un SED logique.

Chapitre 2 :
Résolution de systèmes d'équations à *n* inconnues sur des algèbres de Boole

L'ensemble des résultats présentés dans ce chapitre est général à la résolution d'équations algébriques sur les algèbres de Boole. La première partie expose les différents travaux existant dans le domaine. La seconde partie permet de définir les algèbres de Boole et de donner les différentes propriétés communes à ces algèbres. La troisième partie de ce chapitre s'intéresse aux propriétés qui vont permettre la résolution d'un système d'équations en se ramenant à la résolution d'une seule équation. La dernière partie présente la résolution d'une équation à *n* inconnues sur une algèbre de Boole.

1. Travaux relatifs aux algèbres de Boole

« Résoudre des équations algébriques » est certainement l'un des plus vieux problèmes en mathématiques. Ce problème consiste à trouver quelles valeurs (des nombres, des fonctions, des ensembles,...) respectent une condition donnée sous la forme d'une équation. Ce besoin de résolution est d'ailleurs présent dès les tous premiers travaux relatifs à l'algèbre de Boole.

La notion d'algèbre de Boole a été initiée par les travaux de G. Boole [BOO1854]. Dès le début, l'auteur a posé le problème de résolution d'une équation avec une inconnue sans toutefois parvenir à le résoudre. D'autres auteurs ont ensuite prolongé ses travaux en développant la notion d'algèbre de Boole. C.E. Shannon [SHA40] a ainsi introduit la notion de fonction de variables booléennes et l'expansion de Shannon qui permet de décomposer une expression entre des variables booléennes

par rapport à une variable donnée. Cette propriété a servi de base pour les travaux postérieurs qui se sont intéressés au problème posé par Boole.

S.B. Akers [AKE59] s'intéressa aux fonctions de variables booléennes. Il apporta de nombreux résultats sur ces fonctions comme la notion de différence booléenne (qui est proche de la notion de dérivée pour les fonctions réelles). Concernant la résolution d'équations entre ces fonctions $F(x, y) = 0$, l'auteur présenta et démontra une condition nécessaire et suffisante pour qu'une telle équation admette au moins une solution : $F \cdot \overline{\Delta y F} = 0$. Malheureusement, aucun moyen de trouver ces solutions ne fut donné à l'époque.

R.M. Toms [TOM66] fut à notre connaissance le premier auteur à présenter une technique de résolution d'un système d'équations entre des variables booléennes. Cette technique est composée de deux étapes. La première est d'exprimer le problème sous la forme d'une égalité particulière. Il montra que tous les systèmes composés d'égalités (du type $f_i(x_n) = g_i(x_n)$) sont équivalents à une seule équation dont le second membre est 0 :

$$\sum_{i=1}^{m} (f_i(x_n) \cdot \overline{g_i(x_n)} + \overline{f_i(x_n)} \cdot g_i(x_n)) = 0$$

La seconde étape est de résoudre cette égalité. L'auteur a étudié et présenté le cas où une telle équation admet une unique solution.

Il y a une dizaine d'années, plusieurs auteurs se sont de nouveau intéressés à la résolution d'équations entre des variables booléennes ([WOO96], [RUD03] et [RUS04]). Certains auteurs, comme Woods et Rushdi, se sont intéressés à des systèmes d'équations. Pour les résoudre, ils se sont ramenés à une seule équation, comme Toms, $F(X) = 0$. Rudeanu s'est directement intéressé à résoudre une égalité dont le second membre est 0. Tous ces auteurs se sont ensuite évertués à trouver une condition d'existence de solution puis à obtenir l'ensemble des solutions lorsque celui-ci est non vide. Pour exprimer cet ensemble, ils ont utilisé un encadrement. Les bornes sont trouvées, soit de manière graphique pour Rushdi à l'aide de tableaux similaires à ceux de Karnaugh, soit par traitement algébrique pour Woods et Rudeanu.

Un autre auteur, V.S. Levchenkov, qui s'est également intéressé à la résolution d'équations entre des variables booléennes, utilise une représentation paramétrée pour exprimer l'ensemble des solutions : $y(x) = \alpha(x) \cdot y_1^0(x) + \overline{\alpha(x)} \cdot y_2^0(x)$. Il s'est intéressé au cas des équations à une seule

inconnue dans [LEV00A]. Il a ensuite généralisé ses résultats au cas d'équations à plusieurs inconnues dans [LEV00B].

Il est également à noter qu'une structure particulière nommée « système d'équations booléennes » a été définie par [MAD97]. Cette structure est constituée d'un ensemble de relations entre des variables booléennes en utilisant uniquement les lois de composition OU et ET. Plusieurs techniques de résolution, comme dans [MAT06], ont été développées pour ce type de problème. Pour le problème que nous cherchons à résoudre, les travaux sur les systèmes d'équations booléennes sont inutilisables en raison de l'absence de l'opérateur NON qui est exclu de la définition des systèmes d'équations booléennes telle que définie par [MAD97].

À notre connaissance, il n'existe pas de travaux permettant de résoudre des systèmes d'équations au sein de l'algèbre de Boole des fonctions booléennes (ce qui est nécessaire pour nos travaux). Toutefois, les nombreux travaux mentionnés précédemment sont très utiles pour résoudre ce problème. En effet, les différentes techniques de résolution, et les choix faits par les différents auteurs, sont généralisables. Le passage d'un système d'équations à une égalité particulière et l'expression de l'espace solution sous une forme paramétrique sont les deux caractéristiques qui ont été réutilisées. Il reste toutefois de nombreuses difficultés au niveau des démonstrations des résultats. En effet, les auteurs ([SHA40], [TOM66],...) ont utilisé des techniques énumératives pour démontrer leurs résultats. Cette technique est utilisable dans le cas des variables booléennes car chaque élément ne peut prendre que deux valeurs distinctes. Dans le cas de l'algèbre de Boole des fonctions booléennes, cette méthode n'est pas possible, il faut revenir à la définition même d'une algèbre de Boole. C'est pour cela que la prochaine partie revient sur la définition d'une algèbre de Boole ainsi que sur les différentes propriétés communes à ces algèbres. De plus, comme tous les résultats qui sont présentés utilisent uniquement cette définition, ils sont également valables pour toutes les autres algèbres de Boole.

2. Structure d'algèbre de Boole

En mathématiques, une algèbre de Boole est une structure algébrique particulière au même titre qu'un corps ou un anneau. Il existe dans la littérature plusieurs définitions pour cette structure mathématique. Pour éviter toute ambiguïté due à l'utilisation de définitions élaborées par des auteurs ayant des points de vue différents, les définitions de chaque concept mathématique utilisé dans ce

document sont toutes extraites d'un seul et même ouvrage. Nous avons retenu l'ouvrage de R. P. Grimaldi intitulé « Discrete and Combinatorial Mathematics: An Applied Introduction » [GRI04] qui consacre une partie du chapitre 15 « Boolean Algebra and Switching Functions » à cette structure algébrique.

2.1. Définition

La définition proposée ci-dessous est extraite de [GRI04] page 733.

Définition 1 : Soit $\mathcal{B}$ un ensemble non vide d'éléments contenant deux éléments particuliers 0 (élément Zéro) et 1 (élément Un) sur lequel sont définies deux lois de composition interne binaires notées $(+, \cdot)$ et une loi de composition interne unaire notée $(\bar{\ })$.
$(\mathcal{B}, +, \cdot, \bar{\ }, 0, 1)$ est une algèbre de Boole si les neuf axiomes suivants sont satisfaits pour tout élément x, y et z de $\mathcal{B}$:

$x + y = y + x$	[1]	$x \cdot y = y \cdot x$	[2]	Commutativité
$x + (y \cdot z) = (x + y) \cdot (x + z)$	[3]	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	[4]	Distributivité
$x + 0 = x$	[5]	$x \cdot 1 = x$	[6]	Élément neutre
$x + \bar{x} = 1$	[7]	$x \cdot \bar{x} = 0$	[8]	Élément complémentaire
$0 \neq 1$	[9]			

Par analogie à l'algèbre de Boole classique définie sur les variables booléennes, ces trois lois de composition interne $(+, \cdot, \bar{\ })$ seront respectivement nommées OU, ET, et NON dans la suite de ce document.

2.2. Exemples d'algèbres de Boole

Pour obtenir une algèbre de Boole, il suffit d'un ensemble d'éléments muni de trois lois de composition interne. Parmi les différents exemples d'algèbres de Boole proposés dans la littérature, nous en présentons trois, toutes extraites de [GRI04] :

- l'algèbre des parties d'un ensemble pour son aspect générique,
- l'algèbre de Boole des variables booléennes qui est à l'origine de cette structure algébrique,
- l'algèbre de Boole des fonctions booléennes qui est l'algèbre sur laquelle repose notre méthode de synthèse.

2.2.1. Algèbre des parties d'un ensemble

L'ensemble support de cette algèbre est l'ensemble $\mathcal{P}(U)$ des différents sous-ensembles d'un ensemble U donné. Les éléments particuliers de $\mathcal{P}(U)$ sont respectivement $\varnothing$ et U. Les trois lois de composition interne sont l'union ($\cup$), l'intersection ($\cap$) et le complément à U ($\neg$) d'un sous-ensemble donné.

En s'appuyant sur les définitions de ces trois lois de composition interne, il est possible de démontrer que $(\mathcal{P}(U), \cup, \cap, \neg, \varnothing, U)$ a une structure d'algèbre de Boole.

2.2.2. Algèbre de Boole des variables booléennes

Cette algèbre de Boole est celle définie originellement par G. Boole dans [BOO1854]. Une variable x est appelée variable booléenne si elle ne peut prendre comme valeurs que 0 ou 1. L'ensemble support de cette algèbre est l'ensemble $\mathbb{B} = \{0, 1\}$.

Les trois lois de composition interne de ces variables booléennes sont les suivantes :

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	$\neg x$
0	1
1	0

En s'appuyant sur les définitions de ces trois lois de composition interne, il est possible de démontrer que $(\mathbb{B}, \vee, \wedge, \neg, 0, 1)$ a bien une structure d'algèbre de Boole. Cette algèbre de Boole est celle utilisée en logique, en informatique et en automatique des systèmes logiques.

2.2.3. Algèbre de Boole des fonctions booléennes

Cette algèbre de Boole est l'algèbre sur laquelle repose notre méthode de synthèse. Contrairement aux deux exemples précédents, nous allons détailler avec précision les différentes parties qui la composent afin d'éliminer toute source d'ambiguïté.

Définition 2 : On appelle fonction booléenne d'ordre n toute fonction f de $\mathbb{B}^n$ dans $\mathbb{B}$ qui, à tout n-uplet de variables booléennes $(b_1, \ldots, b_n)$, associe la variable booléenne $f(b_1, \ldots, b_n)$.

$$\begin{aligned} f : \quad & \mathbb{B}^n \to \mathbb{B} \\ & (b_1, \ldots, b_n) \mapsto f(b_1, \ldots, b_n) \end{aligned}$$

Dans la suite de ce document, l'ensemble des fonctions booléennes d'ordre n sera noté Γ_n. Parmi les $2^{(2^n)}$ fonctions booléennes qui constituent Γ_n, deux d'entre elles ont un rôle prépondérant. Il s'agit des deux fonctions constantes qui associent systématiquement à chaque n-uplet de variables booléennes $(b_1, \ldots, b_n)$ la même variable booléenne 0 ou 1.

Définition 3 : On appellera $\mathcal{F}$ (respectivement $\mathcal{T}$) la fonction booléenne d'ordre n qui associe, à chaque n-uplet de variables booléennes $(b_1, \ldots, b_n)$, la variable booléenne 0 (resp. 1).

$$\begin{aligned} \mathcal{F}: \quad & \mathbb{B}^n \to \mathbb{B} \\ & (b_1, \ldots, b_n) \mapsto 0 \end{aligned} \qquad \begin{aligned} \mathcal{T}: \quad & \mathbb{B}^n \to \mathbb{B} \\ & (b_1, \ldots, b_n) \mapsto 1 \end{aligned}$$

Γ_n peut être muni de deux lois de composition interne binaires notées $(+, \cdot)$ et d'une loi de composition interne unaire notée $(\bar{\ })$.

Définition 4 : On appellera OU, ET et NON les trois lois de composition interne de Γ_n définies de la manière suivante :

$$\begin{aligned} \text{OU} : \quad & \Gamma_n \times \Gamma_n \to \Gamma_n \\ & (f, g) \mapsto f + g \end{aligned} \qquad \begin{aligned} \text{ET} : \quad & \Gamma_n \times \Gamma_n \to \Gamma_n \\ & (f, g) \mapsto f \cdot g \end{aligned} \qquad \begin{aligned} \text{NON} : \quad & \Gamma_n \to \Gamma_n \\ & f \mapsto \bar{f} \end{aligned}$$

où :

$$\begin{aligned} (f + g)(b_1, \ldots, b_n) &= (f(b_1, \ldots, b_n)) \vee (g(b_1, \ldots, b_n)) \\ (f \cdot g)(b_1, \ldots, b_n) &= (f(b_1, \ldots, b_n)) \wedge (g(b_1, \ldots, b_n)) \\ \bar{f}(b_1, \ldots, b_n) &= \neg(f(b_1, \ldots, b_n)) \end{aligned}$$

Théorème 1 : $(\Gamma_n, +, \cdot, \bar{\ }, \mathcal{F}, \mathcal{T})$ a une structure d'algèbre de Boole.

Pour démontrer ce théorème, il faut vérifier que la définition des trois lois proposées et les éléments de Γ_n retenus comme éléments particuliers permettent bien d'établir les neuf axiomes qui définissent cette structure. Le lecteur pourra trouver en annexe A la démonstration de toutes les

propriétés présentées dans la suite de ce document et, en annexe B, une démonstration complète de ce théorème.

2.3. Propriétés d'une algèbre de Boole

Cette section regroupe l'ensemble des propriétés d'une algèbre de Boole que nous avons pu identifier. Les 24 propriétés présentées découlent toutes des 9 axiomes qui définissent une structure d'algèbre de Boole. 15 de ces propriétés ([10] à [15], [20], [21], [23] à [29]) sont directement extraites de [GRI04]. Dans [GRI04], les propriétés [18], [19] et [22] ne sont données que sous la forme d'une implication. Nous les présentons ici à l'aide d'une équivalence puisque cette forme a pu être démontrée. Les propriétés [16], [17], [30], [31], [32] et [33] ont été rajoutées car elles correspondent à des résultats classiques dans le cas de l'algèbre de Boole des variables booléennes et sont généralisables à toutes les algèbres de Boole.

$x + x = x$	[10]	$x \cdot x = x$	[11]	Idempotence
$x + 1 = 1$	[12]	$x \cdot 0 = 0$	[13]	Loi de la dominance
$x + (x \cdot y) = x$	[14]	$x \cdot (x + y) = x$	[15]	Élément absorbant
$x + (\bar{x} \cdot y) = x + y$	[16]	$x \cdot (\bar{x} + y) = x \cdot y$	[17]	
$\begin{cases} x + y = x + z \\ \bar{x} + y = \bar{x} + z \end{cases} \Leftrightarrow y = z$	[18]	$\begin{cases} x \cdot y = x \cdot z \\ \bar{x} \cdot y = \bar{x} \cdot z \end{cases} \Leftrightarrow y = z$	[19]	Loi de l'élimination
$x + (y + z) = (x + y) + z$	[20]	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	[21]	Associativité
$\begin{cases} x + y = 1 \\ x \cdot y = 0 \end{cases} \Leftrightarrow y = \bar{x}$	[22]			Unicité de l'inverse
$\bar{\bar{x}} = x$	[23]			Théorème du double complément
$\overline{(x + y)} = \bar{x} \cdot \bar{y}$	[24]	$\overline{(x \cdot y)} = \bar{x} + \bar{y}$	[25]	Théorème de De Morgan
$\bar{0} = 1$	[26]	$\bar{1} = 0$	[27]	
$x + \bar{y} = 1 \Leftrightarrow x + y = x$	[28]	$x \cdot \bar{y} = 0 \Leftrightarrow x \cdot y = x$	[29]	
$((x \cdot y) + (\bar{x} \cdot z)) + (y \cdot z) = (x \cdot y) + (\bar{x} \cdot z)$	[30]			Théorème de la redondance
$x = y \Leftrightarrow (x \cdot \bar{y}) + (\bar{x} \cdot y) = 0$	[31]			
$\begin{cases} x = 0 \\ y = 0 \end{cases} \Leftrightarrow x + y = 0$	[32]			
$x = y \Leftrightarrow \bar{x} = \bar{y}$	[33]			

Le lecteur pourra trouver en annexe A une démonstration complète pour chacune de ces propriétés. Il convient de signaler que la démonstration de certaines d'entre elles (par exemple, le théorème de De Morgan) s'avère plus complexe que dans le cas de l'algèbre de Boole des variables

booléennes car il n'est plus possible d'exploiter des approches énumératives (la dimension de l'ensemble $\mathcal{B}$ n'est pas connue).

Convention de notation : Afin d'alléger les notations, les parenthèses seront dorénavant omises lorsqu'elles ne sont pas nécessaires. L'ordre de priorité retenu entre les lois est identique à celui de l'algèbre de Boole des variables booléennes : la loi NON est prioritaire sur la loi ET, qui est elle-même prioritaire sur la loi OU.

3. Notions complémentaires

Les résultats proposés dans cette partie ont été obtenus dans le cadre de ce travail de thèse. Ils permettent de reformuler tout ensemble de relations entre des éléments d'une algèbre de Boole sous la forme d'une unique équation dont la forme canonique permet sa résolution.

Le premier résultat présenté est la généralisation de la décomposition de Shannon aux compositions d'éléments d'une algèbre de Boole. Ce résultat permet d'établir que toute composition d'éléments d'une algèbre de Boole peut être exprimée sous une forme canonique.

Le second résultat concerne les relations qu'il est possible d'établir entre les éléments d'une algèbre de Boole et les différentes manières de les formuler.

3.1. Compositions d'éléments d'une algèbre de Boole

3.1.1. Introduction

Les expressions combinatoires que manipulent les automaticiens sont un exemple de compositions d'éléments d'une algèbre de Boole. Dans le cas des expressions combinatoires, il s'agit de compositions de variables booléennes à l'aide des lois de composition interne $\vee$, $\wedge$ et $\neg$. L'expansion de Shannon (ou la décomposition de Shannon) est le résultat mathématique qui permet de reformuler chaque expression combinatoire sous la forme d'une somme de mintermes. Cette forme canonique d'une expression booléenne est par ailleurs le fondement de la représentation par BDD (Binary Decision Diagrams). La généralisation de ce principe de décomposition à toute composition d'éléments d'une algèbre de Boole est possible car ce principe ne repose que sur les axiomes

qui définissent une structure d'algèbre de Boole. Une fois démontré, ce théorème permet de garantir que, pour toute algèbre de Boole, toute composition de ses éléments peut être exprimée sous une forme canonique.

Il convient de signaler que ce principe de décomposition, qui est crédité à C.E. Shannon [SHA40], avait été déjà proposé par G. Boole dans [BOO1854].

Pour éviter tout risque de confusion, le terme « expression combinatoire » ne sera utilisé que pour les compositions de variables booléennes, bien que sa généralisation aux compositions d'éléments d'une algèbre de Boole soit théoriquement possible.

3.1.2. Généralisation de l'expansion de Shannon sur une structure d'algèbre de Boole

Soient $(\alpha_1, \ldots, \alpha_n)$ n éléments d'une algèbre de Boole. Par définition des lois OU, ET et NON, toute composition des éléments $(\alpha_1, \ldots, \alpha_n)$ par ces trois lois est également un élément de cette algèbre. Par analogie aux expressions combinatoires, deux compositions $C_A(\alpha_1, \ldots, \alpha_n)$ et $C_B(\alpha_1, \ldots, \alpha_n)$ seront dites équivalentes si elles représentent le même élément.

Parmi les différentes compositions équivalentes à une composition $C(\alpha_1, \ldots, \alpha_n)$ donnée, certaines ont comme structure une somme de produits des éléments $(\alpha_1, \ldots, \alpha_n)$. C'est en appliquant le théorème 2 qu'il est toujours possible de ramener une composition donnée $C(\alpha_1, \ldots, \alpha_n)$ à cette forme canonique.

Théorème 2 : Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient $(\alpha_1, \ldots, \alpha_n)$ n éléments de $(\mathcal{B} - \{0, 1\})$. Toute composition $C(\alpha_1, \ldots, \alpha_n)$ peut se décomposer de la manière suivante :

$$C(\alpha_1, \alpha_2, \ldots, \alpha_n) = \overline{\alpha_1} \cdot C_0(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_1(\alpha_2, \ldots, \alpha_n)$$

où $C_0(\alpha_2, \ldots, \alpha_n)$ et $C_1(\alpha_2, \ldots, \alpha_n)$ sont deux compositions des seuls éléments $(\alpha_2, \ldots, \alpha_n)$.

Ces deux compositions $C_0(\alpha_2, \ldots, \alpha_n)$ et $C_1(\alpha_2, \ldots, \alpha_n)$ peuvent être directement obtenues à partir de $C(\alpha_1, \ldots, \alpha_n)$ en substituant α_1 par 0 et 1 :

$$C_0(\alpha_2, \ldots, \alpha_n) = C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_1 \leftarrow 0} = C(0, \alpha_2, \ldots, \alpha_n)$$

$$C_1(\alpha_2, \ldots, \alpha_n) = C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_1 \leftarrow 1} = C(1, \alpha_2, \ldots, \alpha_n)$$

Pour démontrer que ce théorème est valide pour toute algèbre de Boole, il n'est pas possible de reproduire les approches proposées par G. Boole et C.E. Shannon qui avaient exploité judicieuse-

ment le fait que $\mathbb{B}$ ne comportait que 2 éléments. La démonstration ne doit s'appuyer que sur les propriétés des lois de composition interne OU, ET et NON.

Démonstration :

Cette démonstration sera faite en deux temps :

- Dans un premier temps, il sera démontré que le théorème est vrai pour toutes les compositions élémentaires constituées d'un seul élément.
- Dans un deuxième temps, il sera démontré que, si deux compositions vérifient le théorème, alors toute composition de celles-ci à l'aide des trois lois de composition interne de l'algèbre vérifient également le théorème.

- **Démonstration du théorème pour les compositions élémentaires.** Les quatre compositions élémentaires sont : 0 , 1 , α_1 et α_j avec $\alpha_j \neq \alpha_1$:

- Cas où $C(\alpha_1, \alpha_2, \ldots, \alpha_n) = 0$:

  Nous avons :

$$0 = 1 \cdot 0 = (\overline{\alpha_1} + \alpha_1) \cdot 0 = \overline{\alpha_1} \cdot 0 + \alpha_1 \cdot 0$$

  Donc :

$$C_0(\alpha_2, \ldots, \alpha_n) = 0 = C(0, \alpha_2, \ldots, \alpha_n)$$

$$C_1(\alpha_2, \ldots, \alpha_n) = 0 = C(1, \alpha_2, \ldots, \alpha_n)$$

- Cas où $C(\alpha_1, \alpha_2, \ldots, \alpha_n) = 1$:

  Nous avons :

$$1 = 1 \cdot 1 = (\overline{\alpha_1} + \alpha_1) \cdot 1 = \overline{\alpha_1} \cdot 1 + \alpha_1 \cdot 1$$

  Donc :

$$C_0(\alpha_2, \ldots, \alpha_n) = 1 = C(0, \alpha_2, \ldots, \alpha_n)$$

$$C_1(\alpha_2, \ldots, \alpha_n) = 1 = C(1, \alpha_2, \ldots, \alpha_n)$$

- Cas où $C(\alpha_1, \alpha_2, \ldots, \alpha_n) = \alpha_1$:

Nous avons :

$$\alpha_1 = 0 + \alpha_1 \cdot 1 = \overline{\alpha_1} \cdot 0 + \alpha_1 \cdot 1$$

Donc :

$$C_0(\alpha_2, \ldots, \alpha_n) = 0 = C(0, \alpha_2, \ldots, \alpha_n)$$

$$C_1(\alpha_2, \ldots, \alpha_n) = 1 = C(1, \alpha_2, \ldots, \alpha_n)$$

- Cas où $C(\alpha_1, \alpha_2, \ldots, \alpha_n) = \alpha_j$ avec $\alpha_j \neq \alpha_1$:

  Nous avons :

$$\alpha_j = 1 \cdot \alpha_j = (\overline{\alpha_1} + \alpha_1) \cdot \alpha_j = \overline{\alpha_1} \cdot \alpha_j + \alpha_1 \cdot \alpha_j$$

  Donc :

$$C_0(\alpha_2, \ldots, \alpha_n) = \alpha_j = C(0, \alpha_2, \ldots, \alpha_n)$$

$$C_1(\alpha_2, \ldots, \alpha_n) = \alpha_j = C(1, \alpha_2, \ldots, \alpha_n)$$

- **Démonstration du théorème pour une composition par les lois OU, ET et NON.** La première partie a permis de démontrer que le théorème 2 est vrai pour toute composition élémentaire. Il nous faut maintenant mettre en place la démonstration par récurrence. Supposons $C_A(\alpha_1, \ldots, \alpha_n)$ et $C_B(\alpha_1, \ldots, \alpha_n)$ décomposables. Nous allons montrer que toute composition par les trois lois de composition interne est elle-même décomposable.

$$C_A(\alpha_1, \alpha_2, \ldots, \alpha_n) = \overline{\alpha_1} \cdot C_{A0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{A1}(\alpha_2, \ldots, \alpha_n)$$

$$C_B(\alpha_1, \alpha_2, \ldots, \alpha_n) = \overline{\alpha_1} \cdot C_{B0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{B1}(\alpha_2, \ldots, \alpha_n)$$

  où :

$$C_{A0}(\alpha_2, \ldots, \alpha_n) = C_A(0, \alpha_2, \ldots, \alpha_n)$$
$$C_{A1}(\alpha_2, \ldots, \alpha_n) = C_A(1, \alpha_2, \ldots, \alpha_n)$$

$$C_{B0}(\alpha_2, \ldots, \alpha_n) = C_B(0, \alpha_2, \ldots, \alpha_n)$$
$$C_{B1}(\alpha_2, \ldots, \alpha_n) = C_B(1, \alpha_2, \ldots, \alpha_n)$$

- Cas d'une composition par la loi de composition OU :

$$\begin{aligned} C(\alpha_1, \ldots, \alpha_n) &= C_A(\alpha_1, \ldots, \alpha_n) + C_B(\alpha_1, \ldots, \alpha_n) \\ &= \overline{\alpha_1} \cdot C_{A0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{A1}(\alpha_2, \ldots, \alpha_n) \\ &\quad + \overline{\alpha_1} \cdot C_{B0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{B1}(\alpha_2, \ldots, \alpha_n) \\ &= \overline{\alpha_1} \cdot (C_{A0}(\alpha_2, \ldots, \alpha_n) + C_{B0}(\alpha_2, \ldots, \alpha_n)) \\ &\quad + \alpha_1 \cdot (C_{A1}(\alpha_2, \ldots, \alpha_n) + C_{B1}(\alpha_2, \ldots, \alpha_n)) \end{aligned}$$

donc :

$$\begin{aligned} C_0(\alpha_2, \ldots, \alpha_n) &= C_{A0}(\alpha_2, \ldots, \alpha_n) + C_{B0}(\alpha_2, \ldots, \alpha_n) \\ &= C_A(0, \alpha_2, \ldots, \alpha_n) + C_B(0, \alpha_2, \ldots, \alpha_n) \\ &= C(0, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

$$\begin{aligned} C_1(\alpha_2, \ldots, \alpha_n) &= C_{A1}(\alpha_2, \ldots, \alpha_n) + C_{B1}(\alpha_2, \ldots, \alpha_n) \\ &= C_A(1, \alpha_2, \ldots, \alpha_n) + C_B(1, \alpha_2, \ldots, \alpha_n) \\ &= C(1, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

- Cas d'une composition par la loi de composition ET :

$$\begin{aligned} C(\alpha_1, \ldots, \alpha_n) &= C_A(\alpha_1, \ldots, \alpha_n) \cdot C_B(\alpha_1, \ldots, \alpha_n) \\ &= (\overline{\alpha_1} \cdot C_{A0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{A1}(\alpha_2, \ldots, \alpha_n)) \\ &\quad \cdot (\overline{\alpha_1} \cdot C_{B0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{B1}(\alpha_2, \ldots, \alpha_n)) \\ &= \overline{\alpha_1} \cdot (C_{A0}(\alpha_2, \ldots, \alpha_n) \cdot C_{B0}(\alpha_2, \ldots, \alpha_n)) \\ &\quad + \alpha_1 \cdot (C_{A1}(\alpha_2, \ldots, \alpha_n) \cdot C_{B1}(\alpha_2, \ldots, \alpha_n)) \end{aligned}$$

donc :

$$\begin{aligned} C_0(\alpha_2, \ldots, \alpha_n) &= C_{A0}(\alpha_2, \ldots, \alpha_n) \cdot C_{B0}(\alpha_2, \ldots, \alpha_n) \\ &= C_A(0, \alpha_2, \ldots, \alpha_n) \cdot C_B(0, \alpha_2, \ldots, \alpha_n) \\ &= C(0, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

$$\begin{aligned} C_1(\alpha_2, \ldots, \alpha_n) &= C_{A1}(\alpha_2, \ldots, \alpha_n) \cdot C_{B1}(\alpha_2, \ldots, \alpha_n) \\ &= C_A(1, \alpha_2, \ldots, \alpha_n) \cdot C_B(1, \alpha_2, \ldots, \alpha_n) \\ &= C(1, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

- Cas d'une composition par la loi de composition NON :

$$\begin{aligned} C(\alpha_1, \ldots, \alpha_n) &= \overline{C_A(\alpha_1, \ldots, \alpha_n)} \\ &= \overline{\overline{\alpha_1} \cdot C_{A0}(\alpha_2, \ldots, \alpha_n) + \alpha_1 \cdot C_{A1}(\alpha_2, \ldots, \alpha_n)} \\ &= (\alpha_1 + \overline{C_{A0}(\alpha_2, \ldots, \alpha_n)}) \cdot (\overline{\alpha_1} + \overline{C_{A1}(\alpha_2, \ldots, \alpha_n)}) \\ &= \alpha_1 \cdot \overline{C_{A1}(\alpha_2, \ldots, \alpha_n)} + \overline{\alpha_1} \cdot \overline{C_{A0}(\alpha_2, \ldots, \alpha_n)} \\ &\quad + \overline{C_{A0}(\alpha_2, \ldots, \alpha_n)} \cdot \overline{C_{A1}(\alpha_2, \ldots, \alpha_n)} \\ &= \overline{\alpha_1} \cdot \overline{C_{A0}(\alpha_2, \ldots, \alpha_n)} + \alpha_1 \cdot \overline{C_{A1}(\alpha_2, \ldots, \alpha_n)} \qquad \text{(théorème de la redondance)} \end{aligned}$$

Donc :

$$\begin{aligned} C_0(\alpha_2, \ldots, \alpha_n) &= \overline{C_{A0}(\alpha_2, \ldots, \alpha_n)} \\ &= \overline{C_A(0, \alpha_2, \ldots, \alpha_n)} \\ &= C(0, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

$$\begin{aligned} C_1(\alpha_2, \ldots, \alpha_n) &= \overline{C_{A1}(\alpha_2, \ldots, \alpha_n)} \\ &= \overline{C_A(1, \alpha_2, \ldots, \alpha_n)} \\ &= C(1, \alpha_2, \ldots, \alpha_n) \end{aligned}$$

□

La généralisation de l'expansion de Shannon étant démontrée, nous pouvons affirmer que toute composition d'éléments $(\alpha_1, \ldots, \alpha_n)$ d'une algèbre de Boole peut s'exprimer sous la forme d'une somme de produits dont les termes sont ces éléments ou leur complément.

3.1.3. Substitution d'éléments dans une composition

Théorème 3 : Soit $(\mathcal{B}, +, \cdot, \overline{\ }, 0, 1)$ une algèbre de Boole. Soient $(\alpha_1, \ldots, \alpha_n)$ n éléments de $\mathcal{B} - \{0, 1\}$. Dans toute composition $C(\alpha_1, \ldots, \alpha_n)$, la substitution de α_i par une composition D ne dépendant pas de α_i peut s'obtenir par :

$$C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow D} = \overline{D} \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 0} + D \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 1}$$

Démonstration :

- D'après le théorème 2, $C(\alpha_1, \ldots, \alpha_n)$ se décompose sous la forme :

$$C(\alpha_1, \ldots, \alpha_n) = \overline{\alpha_i} \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 0} + \alpha_i \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 1}$$

- Donc :

$$\begin{aligned} & C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow D} \\ & = \left(\overline{\alpha_i} \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 0} + \alpha_i \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 1} \right)\Big|_{\alpha_i \leftarrow D} \\ & = \overline{D} \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 0} + D \cdot C(\alpha_1, \ldots, \alpha_n)\big|_{\alpha_i \leftarrow 1} \end{aligned}$$

□

3.2. Relations entre les éléments d'une algèbre de Boole

La relation égalité n'est pas la seule relation possible entre des éléments d'une algèbre de Boole. Il est également possible de définir une relation d'ordre partiel entre ces éléments.

3.2.1. Relation d'ordre partiel dans une algèbre de Boole (définition et propriétés)

Sur toute algèbre de Boole peut être définie une relation d'ordre partiel en raison des propriétés des lois de composition OU et ET (Idempotence, Commutativité et Associativité). Parmi les différentes définitions de cette relation d'ordre partiel, nous avons retenu celle proposée par Grimaldi ([GRI04] page 737).

Définition 5 : Soit $x, y \in \mathcal{B}$, x et y satisfont $x \leq y$ si et seulement si $x \cdot y = x$:

$$x \leq y \Leftrightarrow x \cdot y = x$$

Dans les ouvrages mathématiques que nous avons consultés, cette relation ne possède pas de nom particulier. Nous avons choisi de retenir celui qu'elle porte au sein de la théorie des ensembles ($A \subseteq B \Leftrightarrow A \cap B = A$). Dans la suite de ce document, cette relation sera appelée « Inclusion ».

Théorème 4 : La relation « Inclusion » ($\leq$) est une relation d'ordre partiel.

Une relation sur un ensemble $\mathcal{B}$ est appelée relation d'ordre partiel si les trois propriétés suivantes sont vérifiées pour tout élément x, y et z de $\mathcal{B}$:

- Réflexivité : $x \leq x$,
- Antisymétrie : Si $x \leq y$ et $y \leq x$, alors $x = y$,
- Transitivité : Si $x \leq y$ et $y \leq z$, alors $x \leq z$.

La démonstration proposée par Grimaldi pour ce théorème est donnée en annexe B.

En raison des différentes propriétés de la structure d'algèbre de Boole, nous avons pu établir que la relation inclusion satisfait les propriétés suivantes pour tout $x, y, z, w \in \mathcal{B}$:

$$0 \leq x \quad [34] \qquad x \leq 1 \quad [35]$$

$$(x \cdot y) \leq y \quad [36] \qquad x \leq (x + y) \quad [37]$$

$$x \leq (y \cdot z) \Leftrightarrow \begin{cases} x \leq y \\ x \leq z \end{cases} \quad [38] \qquad (x + y) \leq z \Leftrightarrow \begin{cases} x \leq z \\ y \leq z \end{cases} \quad [39]$$

$$x \leq y \Rightarrow (x \cdot z) \leq (y \cdot z) \quad [40] \qquad x \leq y \Rightarrow (x + z) \leq (y + z) \quad [41]$$

$$\begin{cases} x \leq y \\ z \leq w \end{cases} \Rightarrow (x \cdot z) \leq (y \cdot w) \quad [42] \qquad \begin{cases} x \leq y \\ z \leq w \end{cases} \Rightarrow (x + z) \leq (y + w) \quad [43]$$

La démonstration de chacune de ces propriétés est donnée en annexe A.

3.2.2. Formes équivalentes d'une relation

Dans notre approche de synthèse, les relations ($=$ et $\leq$) sont utilisées pour exprimer des fragments du cahier des charges du système logique à synthétiser. Un même fragment peut être formalisé de plusieurs manières différentes car il existe plusieurs formes équivalentes pour exprimer la même relation entre deux éléments d'une algèbre. Pour éviter toute source de confusion, les principales formes équivalentes pour les deux relations $x = y$ et $x \leq y$ sont listées ci-dessous :

Théorème 5 : Les cinq formes suivantes pour l'égalité sont équivalentes pour tout $x, y \in \mathcal{B}$:

$$x = y \qquad x \cdot \bar{y} + \bar{x} \cdot y = 0 \qquad \begin{cases} x \cdot \bar{y} = 0 \\ \bar{x} \cdot y = 0 \end{cases}$$

$$\bar{x} = \bar{y} \qquad x \cdot y + \bar{x} \cdot \bar{y} = 1$$

Théorème 6 : Les huit formes suivantes pour l'inclusion sont équivalentes pour tout $x, y \in \mathcal{B}$:

$$x \leq y \qquad x \cdot y = x \qquad \bar{x} + \bar{y} = \bar{x} \qquad x \cdot \bar{y} = 0$$
$$\bar{y} \leq \bar{x} \qquad \bar{x} \cdot \bar{y} = \bar{y} \qquad x + y = y \qquad \bar{x} + y = 1$$

Ces deux théorèmes sont démontrés en annexe B. Certaines de ces équivalences seront exploitées pour reformuler un ensemble de relations à l'aide d'une relation unique.

3.2.3. Représentation d'un ensemble de relations

Dans notre approche de synthèse, nous sommes amenés à manipuler des ensembles de relations entre des éléments d'une algèbre de Boole puisque c'est sous cette forme qu'est formalisé le cahier des charges du système logique à synthétiser.

Grâce aux propriétés des lois OU, ET et NON et aux propriétés des relations $=$ et $\leq$, il est toujours possible de reformuler un ensemble de relations sous la forme d'une relation unique. La forme canonique que nous avons retenue est une égalité dont le second membre est l'élément neutre 0. Cette forme canonique a été choisie car, comme nous le montrerons dans la prochaine partie de ce chapitre, elle est bien adaptée à la résolution d'équations dans une algèbre de Boole.

Théorème 7 : Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient $(\alpha_1, \ldots, \alpha_n)$ n éléments de $\mathcal{B}$. Tout ensemble de relations entre les éléments $(\alpha_1, \ldots, \alpha_n)$ peut être reformulé sous la forme d'une unique relation de la forme :

$$C(\alpha_1, \ldots, \alpha_n) = 0$$

Démonstration :

Cette démonstration sera faite par l'exemple. L'exemple support de cette démonstration est un ensemble générique de relations entre les éléments $(\alpha_1, \ldots, \alpha_n)$ qui contient tous les types de relations. Soit l'ensemble de relations suivant :

$$\begin{cases} C_1(\alpha_1, \ldots, \alpha_n) \leq C_2(\alpha_1, \ldots, \alpha_n) \\ C_3(\alpha_1, \ldots, \alpha_n) = 0 \\ C_4(\alpha_1, \ldots, \alpha_n) = 1 \\ C_5(\alpha_1, \ldots, \alpha_n) = C_6(\alpha_1, \ldots, \alpha_n) \end{cases}$$

La démonstration se fera en trois temps :

- Dans un premier temps, toutes les inclusions seront reformulées sous la forme d'une égalité dont le second membre est 0 (grâce au théorème 6).
- Dans un deuxième temps, toutes les égalités seront reformulées sous la forme d'une égalité dont le second membre est 0 (grâce au théorème 5).
- Dans un troisième temps, l'ensemble des égalités sera reformulé sous la forme d'une seule égalité dont le second membre est 0 (grâce à la propriété [32]).

$$\begin{cases} C_1(\alpha_1, \ldots, \alpha_n) \leq C_2(\alpha_1, \ldots, \alpha_n) \\ C_3(\alpha_1, \ldots, \alpha_n) = 0 \\ C_4(\alpha_1, \ldots, \alpha_n) = 1 \\ C_5(\alpha_1, \ldots, \alpha_n) = C_6(\alpha_1, \ldots, \alpha_n) \end{cases}$$

$$\Leftrightarrow \begin{cases} C_1(\alpha_1, \ldots, \alpha_n) \cdot \overline{C_2(\alpha_1, \ldots, \alpha_n)} = 0 \\ C_3(\alpha_1, \ldots, \alpha_n) = 0 \\ C_4(\alpha_1, \ldots, \alpha_n) = 1 \\ C_5(\alpha_1, \ldots, \alpha_n) = C_6(\alpha_1, \ldots, \alpha_n) \end{cases}$$

$$\Leftrightarrow \begin{cases} C_1(\alpha_1, \ldots, \alpha_n) \cdot \overline{C_2(\alpha_1, \ldots, \alpha_n)} = 0 \\ C_3(\alpha_1, \ldots, \alpha_n) = 0 \\ \overline{C_4(\alpha_1, \ldots, \alpha_n)} = 0 \\ C_5(\alpha_1, \ldots, \alpha_n) \cdot \overline{C_6(\alpha_1, \ldots, \alpha_n)} + \overline{C_5(\alpha_1, \ldots, \alpha_n)} \cdot C_6(\alpha_1, \ldots, \alpha_n) = 0 \end{cases}$$

$$\Leftrightarrow C_1(\alpha_1, \ldots, \alpha_n) \cdot \overline{C_2(\alpha_1, \ldots, \alpha_n)} + C_3(\alpha_1, \ldots, \alpha_n) + \overline{C_4(\alpha_1, \ldots, \alpha_n)} + C_5(\alpha_1, \ldots, \alpha_n) \cdot \overline{C_6(\alpha_1, \ldots, \alpha_n)} + \overline{C_5(\alpha_1, \ldots, \alpha_n)} \cdot C_6(\alpha_1, \ldots, \alpha_n) = 0$$

Tout ensemble de relations entre les éléments $(\alpha_1, \ldots, \alpha_n)$ peut donc bien être reformulé sous la forme d'une unique égalité dont le second membre est 0.

□

La partie 2. de ce chapitre a été consacrée à la présentation de la définition et des propriétés d'une structure d'algèbre de Boole. Dans cette partie 3., nous avons présenté la relation d'ordre partiel qu'il est possible de définir entre les éléments d'une algèbre de Boole. Nous avons également démontré que tout ensemble de relations entre les éléments $(\alpha_1, \ldots, \alpha_n)$ d'une algèbre de Boole pouvait être reformulé sous la forme d'une unique égalité ayant pour caractéristiques :

- Le premier membre est une composition par les lois OU, ET et NON des éléments $(\alpha_1, \ldots, \alpha_n)$. Cette composition peut être exprimée sous la forme d'une somme de produits dont les termes sont les éléments $(\alpha_1, \ldots, \alpha_n)$ ou leur complément (théorème 2).
- Le second membre est l'élément particulier 0.

La résolution d'un système d'équations se ramène donc à la résolution d'une seule équation. Dans la prochaine partie, nous allons nous intéresser à la résolution d'une équation à plusieurs inconnues au sein d'une algèbre de Boole.

4. Résolution d'équations au sein d'une algèbre de Boole

Cette partie est au cœur scientifique de cette thèse. Nous allons démontrer que toute équation liant des éléments $(x_1, \ldots, x_n)$ considérés comme **inconnus** à des éléments $(\alpha_1, \ldots, \alpha_m)$ considérés comme **connus** peut être résolue et nous donnerons une forme paramétrique des solutions lorsque ces solutions existent. En couplant ce nouveau résultat à ceux démontrés dans la partie précédente, nous serons en mesure de résoudre tout système de relations liant des éléments d'une algèbre de Boole quelle que soit la forme de ces relations.

Contrairement aux équations dans le corps des nombres réels ou des nombres complexes, la résolution d'une équation à plusieurs inconnues dans une algèbre de Boole est analytiquement toujours possible. C'est sur cette caractéristique mathématique que repose notre méthode de synthèse des systèmes logiques.

Cette partie de ce chapitre comporte trois sous-parties :

- La première sous-partie sera consacrée à la résolution d'une équation à une seule inconnue. Nous proposerons et démontrerons le théorème permettant de résoudre toute équation à une seule inconnue. La forme paramétrique proposée pour décrire l'ensemble des solutions sera ensuite commentée.
- La deuxième sous-partie sera consacrée à la résolution d'une équation à deux inconnues. Nous montrerons que la résolution d'une équation à deux inconnues peut se faire en résolvant de manière imbriquée deux équations à une seule inconnue. C'est la généralisation de ce principe à l'ordre n qui permet de résoudre une équation à n inconnues. Nous montrerons que si l'ordre

de résolution a une influence sur la forme de la solution paramétrique proposée, cet ordre est sans influence sur les solutions de l'équation.

- La troisième sous-partie sera consacrée au cas général des équations à n inconnues. Après avoir présenté les notations nécessaires à sa formulation, nous donnerons le théorème qui permet de résoudre toute équation entre les éléments d'une algèbre de Boole. La démonstration de ce dernier théorème se fera par récurrence.

4.1. Résolution d'une équation à une seule inconnue

4.1.1. Obtention de la forme canonique

D'après le théorème 2, toute équation $C(x, \alpha_1, \ldots, \alpha_m) = 0$ peut être décomposée suivant x pour être ramenée à la forme suivante :

$$\bar{x} \cdot C(0, \alpha_1, \ldots, \alpha_m) + x \cdot C(1, \alpha_1, \ldots, \alpha_m) = 0$$

En considérant comme connus les éléments $(\alpha_1, \ldots, \alpha_m)$, les éléments de $\mathcal{B}$ auxquels correspondent les compositions $C(0, \alpha_1, \ldots, \alpha_m)$ et $C(1, \alpha_1, \ldots, \alpha_m)$ sont alors des éléments parfaitement définis. Soient a et b ces éléments. En conséquence, toute équation $C(x, \alpha_1, \ldots, \alpha_m) = 0$ peut toujours être reformulée sous la forme suivante :

$$a \cdot \bar{x} + b \cdot x = 0$$

où $a = C(0, \alpha_1, \ldots, \alpha_m)$ et $b = C(1, \alpha_1, \ldots, \alpha_m)$.

4.1.2. Théorème

Les résultats nécessaires pour résoudre dans une algèbre de Boole une équation à une seule inconnue sont donnés dans le théorème suivant :

Théorème 8 : Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient a, b et x trois éléments de cette algèbre. Considérons l'équation suivante où x est l'inconnue :

$$a \cdot \bar{x} + b \cdot x = 0$$

Cette équation admet des solutions si et seulement si :

$$a \cdot b = 0$$

Lorsque cette condition est respectée, cette équation admet une ou plusieurs solutions que la forme paramétrique suivante permet de décrire :

$$x = a + \bar{b} \cdot p \text{ où } p \text{ est un paramètre, élément de } \mathcal{B}.$$

L'ensemble des solutions de l'équation peut également être décrit à l'aide de l'une de ces deux formes équivalentes :

$$\begin{aligned} x &= \bar{b} \cdot (a + p) \\ &= a \cdot \bar{p} + \bar{b} \cdot p \end{aligned}$$

p joue le rôle de paramètre, cela peut être tout élément de $\mathcal{B}$. L'ensemble des valeurs possibles de ce paramètre permet de parcourir l'ensemble de l'espace solution.

Pour que ce théorème soit valide pour toute algèbre de Boole, il n'est pas possible de le démontrer en reproduisant l'approche proposée par N.P. Brusentsov [BRU98] qui avait exploité judicieusement le fait que $\mathbb{B}$ ne comportait que 2 éléments. La démonstration ne doit s'appuyer que sur les propriétés des lois de composition interne OU, ET et NON.

Démonstration :

Cette démonstration sera faite en quatre temps :

- Dans un premier temps, il sera démontré que la condition $a \cdot b = 0$ est une condition nécessaire et suffisante pour que l'équation admette des solutions.
- Dans un deuxième temps, il sera vérifié que la forme paramétrique est bien solution de l'équation $a \cdot \bar{x} + b \cdot x = 0$.

- Dans un troisième temps, il sera démontré que toute solution de l'équation $a \cdot \bar{x} + b \cdot x = 0$ peut être décrite avec la forme paramétrique proposée.
- Dans un quatrième temps, il sera vérifié que les trois formes paramétriques proposées sont bien équivalentes.

- **Condition nécessaire et suffisante pour avoir une ou plusieurs solutions.**
  - Condition nécessaire

Sachant que $a \cdot \bar{x} + b \cdot x = a \cdot \bar{x} + b \cdot x + a \cdot b$ (Théorème de la redondance), l'équation $a \cdot \bar{x} + b \cdot x = 0$ admet des solutions sur une algèbre de Boole si l'équation $a \cdot \bar{x} + b \cdot x + a \cdot b = 0$ ou une de ses formes équivalentes admet des solutions.

Sachant que :

$$a \cdot \bar{x} + b \cdot x + a \cdot b = 0 \Leftrightarrow \begin{cases} a \cdot \bar{x} + b \cdot x = 0 \\ a \cdot b = 0 \end{cases}$$

La condition $a \cdot b = 0$ est donc bien une condition nécessaire pour que $a \cdot \bar{x} + b \cdot x = 0$ admette des solutions.

  - Condition suffisante

Pour démontrer que la condition $a \cdot b = 0$ est une condition suffisante, il suffit de pouvoir donner une solution de cette équation en supposant que cette condition est respectée :

$$\begin{cases} a \cdot b = 0 \\ x = a \end{cases} \Rightarrow a \cdot \bar{x} + b \cdot x = 0$$

$$\begin{cases} a \cdot b = 0 \\ x = a \end{cases} \Rightarrow$$

$$\begin{aligned} & a \cdot \bar{x} + b \cdot x \\ & = a \cdot \bar{a} + b \cdot a \\ & = 0 + 0 \\ & = 0 \end{aligned}$$

La condition $a \cdot b = 0$ est bien une condition nécessaire et suffisante pour que $a \cdot \bar{x} + b \cdot x = 0$ admette des solutions.

- $x = a + \bar{b} \cdot p$ **(avec** p **un élément de** $\mathcal{B}$**) est une solution de l'équation** $a \cdot \bar{x} + b \cdot x = 0$**.**

$$\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \end{cases} \Rightarrow \quad a \cdot \bar{x} + b \cdot x = 0$$

$$\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \end{cases} \Rightarrow$$

$$\begin{aligned} & a \cdot \bar{x} + b \cdot x \\ & = a \cdot \overline{(a + \bar{b} \cdot p)} + b \cdot (a + \bar{b} \cdot p) \\ & = a \cdot \bar{a} \cdot (b + \bar{p}) + a \cdot b + b \cdot \bar{b} \cdot p \\ & = 0 + 0 + 0 \\ & = 0 \end{aligned}$$

Lorsque $a \cdot b = 0$, $x = a + \bar{b} \cdot p$ (avec p un élément de $\mathcal{B}$) est bien solution de $a \cdot \bar{x} + b \cdot x = 0$.

- **Toutes les solutions de** $a \cdot \bar{x} + b \cdot x = 0$ **peuvent s'exprimer sous la forme** $x = a + \bar{b} \cdot p$**.**

Pour démontrer ce point, il suffit de trouver un élément p de $\mathcal{B}$ qui satisfasse $x = a + \bar{b} \cdot p$ pour chaque solution x de $a \cdot \bar{x} + b \cdot x = 0$. Considérons p défini à partir de x de la manière suivante : $p = \bar{a} \cdot \bar{b} \cdot x$.

$$\begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ p = \bar{a} \cdot \bar{b} \cdot x \end{cases} \Rightarrow \quad x = a + \bar{b} \cdot p$$

$$\begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ p = \bar{a} \cdot \bar{b} \cdot x \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} = 0 \\ b \cdot x = 0 \\ p = \bar{a} \cdot \bar{b} \cdot x \end{cases} \Rightarrow$$

$$\begin{aligned} x &= 1 \cdot x \\ &= (a + b + \bar{a} \cdot \bar{b}) \cdot x \\ &= a \cdot x + b \cdot x + \bar{a} \cdot \bar{b} \cdot x \\ &= a \cdot x + 0 + \bar{b} \cdot (\bar{a} \cdot \bar{b} \cdot x) \\ &= a \cdot x + a \cdot \bar{x} + \bar{b} \cdot p \\ &= a \cdot (x + \bar{x}) + \bar{b} \cdot p \\ &= a + \bar{b} \cdot p \end{aligned}$$

Lorsque $a \cdot b = 0$, toutes les solutions de $a \cdot \bar{x} + b \cdot x = 0$ peuvent s'exprimer sous la forme proposée.

- **Les trois formes proposées sont équivalentes :**

$$\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ x = \bar{b} \cdot (a + p) \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ x = a \cdot \bar{p} + \bar{b} \cdot p \end{cases}$$

- $\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ x = \bar{b} \cdot (a + p) \end{cases}$

$$\begin{aligned} x &= a + \bar{b} \cdot p \\ &= a \cdot 1 + \bar{b} \cdot p \\ &= a \cdot (b + \bar{b}) + \bar{b} \cdot p \\ &= a \cdot b + a \cdot \bar{b} + \bar{b} \cdot p \\ &= 0 + \bar{b} \cdot (a + p) \\ &= \bar{b} \cdot (a + p) \end{aligned}$$

- $\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ x = a \cdot \bar{p} + \bar{b} \cdot p \end{cases}$

$$\begin{aligned} x &= a + \bar{b} \cdot p \\ &= a \cdot 1 + \bar{b} \cdot p \\ &= a \cdot (\bar{p} + p) + \bar{b} \cdot p \\ &= a \cdot \bar{p} + a \cdot p + \bar{b} \cdot p \\ &= a \cdot \bar{p} + (a + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + (a \cdot 1 + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + (a \cdot (b + \bar{b}) + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + (a \cdot b + a \cdot \bar{b} + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + (0 + a \cdot \bar{b} + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + (0 + \bar{b}) \cdot p \\ &= a \cdot \bar{p} + \bar{b} \cdot p \end{aligned}$$

Les trois formes proposées sont bien équivalentes.

Les quatre temps de la démonstration du théorème 8 étant maintenant menés à bien, ce théorème est donc prouvé pour toute algèbre de Boole.

□

4.1.3. Discussion

Dans la forme paramétrique proposée $x = a + \bar{b} \cdot p$, p peut être n'importe lequel des éléments de $\mathcal{B}$. Cependant, il n'est pas nécessaire que p parcoure tout l'ensemble $\mathcal{B}$ pour couvrir entièrement l'espace solution de l'équation $a \cdot \bar{x} + b \cdot x = 0$ car le nombre de solutions différentes pour cette équation est égal au nombre d'éléments de $\mathcal{B}$ qui satisfont :

$$a \leq x \leq \bar{b}$$

Cet encadrement de la valeur de x exprimé à l'aide de la relation inclusion s'obtient directement de l'équation $a \cdot \bar{x} + b \cdot x = 0$ de la façon suivante :

$$\begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} = 0 \\ b \cdot x = 0 \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \leq x \\ x \leq \bar{b} \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \leq x \leq \bar{b} \end{cases}$$

Il est cependant possible de créer une bijection entre l'ensemble des solutions de l'équation $a \cdot \bar{x} + b \cdot x = 0$ et l'ensemble des paramètres en restreignant ce dernier à la seule partie utile, c'est-à-dire : $p \leq \bar{a} \cdot \bar{b}$.

Remarque : Soit x une des solutions de l'équation $a \cdot \bar{x} + b \cdot x = 0$. Ces solutions, qui peuvent être obtenue par la forme paramétrique $a + \bar{b} \cdot p$, forment une bijection avec l'ensemble des paramètres p tel que $p \leq \bar{a} \cdot \bar{b}$. C'est-à-dire que :

$$\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \\ p \leq \bar{a} \cdot \bar{b} \end{cases} \Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ p = \bar{a} \cdot \bar{b} \cdot x \end{cases}$$

Démonstration :

$$\begin{cases} a \cdot b = 0 \\ x = a + \bar{b} \cdot p \\ p \leq \bar{a} \cdot \bar{b} \end{cases}$$

$$\Leftrightarrow \begin{cases} a \cdot b = 0 \\ \bar{x} \cdot (a + \bar{b} \cdot p) + x \cdot \overline{(a + \bar{b} \cdot p)} = 0 \\ p \cdot \overline{(\bar{a} \cdot \bar{b})} = 0 \end{cases}$$

$$\Leftrightarrow a \cdot b + \bar{x} \cdot (a + \bar{b} \cdot p) + x \cdot \overline{(a + \bar{b} \cdot p)} + p \cdot \overline{(\bar{a} \cdot \bar{b})} = 0$$

$$\Leftrightarrow a \cdot b + a \cdot \bar{x} + \bar{b} \cdot p \cdot \bar{x} + x \cdot (\bar{a} \cdot (b + \bar{p})) + p \cdot (a + b) = 0$$

$$\Leftrightarrow a \cdot b + a \cdot \bar{x} + \bar{b} \cdot p \cdot \bar{x} + \bar{a} \cdot x \cdot (b + \bar{b} \cdot \bar{p}) + p \cdot (a + b) = 0$$

$$\Leftrightarrow a \cdot b + a \cdot \bar{x} + \bar{a} \cdot b \cdot x + \bar{b} \cdot p \cdot \bar{x} + \bar{a} \cdot \bar{b} \cdot \bar{p} \cdot x + p \cdot (a + b) = 0$$

$$\Leftrightarrow \begin{cases} a \cdot b + a \cdot \bar{x} + \bar{a} \cdot b \cdot x = 0 \\ \bar{b} \cdot p \cdot \bar{x} + \bar{a} \cdot \bar{b} \cdot \bar{p} \cdot x + p \cdot (a + b) = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} a \cdot b + a \cdot \bar{x} + b \cdot x = 0 \\ \bar{a} \cdot \bar{b} \cdot \bar{p} \cdot x + p \cdot (a + b + \bar{b} \cdot \bar{x}) = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ \bar{p} \cdot (\bar{a} \cdot \bar{b} \cdot x) + p \cdot (a + b + \bar{x}) = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ \bar{p} \cdot (\bar{a} \cdot \bar{b} \cdot x) + p \cdot \overline{(\bar{a} \cdot \bar{b} \cdot x)} = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} a \cdot b = 0 \\ a \cdot \bar{x} + b \cdot x = 0 \\ p = \bar{a} \cdot \bar{b} \cdot x \end{cases}$$

□

Remarque : Si les éléments a et b sont tels que $a = \bar{b}$ (c'est-à-dire que $a + b = 1$ en plus de $a \cdot b = 0$), alors l'équation n'a qu'une seule solution qui est $x = a$ ($x = a + \bar{b} \cdot p = a + a \cdot p = a$).

Il est maintenant possible de résoudre toute équation comportant une inconnue. La suite de ce chapitre est consacrée à l'utilisation de ce résultat pour résoudre une équation à deux inconnues.

4.2. Résolution d'une équation à deux inconnues

Soit $C(x_1, x_2, \alpha_1, \ldots, \alpha_m) = 0$ une équation liant plusieurs éléments d'une algèbre de Boole. Les éléments x_1 et x_2 sont les deux éléments inconnus. $\alpha_1, \ldots, \alpha_m$ représentent m éléments connus. On cherche donc à exprimer les deux éléments inconnus x_1 et x_2 en fonction des éléments connus $\alpha_1, \ldots, \alpha_m$ pour que l'équation $C(x_1, x_2, \alpha_1, \ldots, \alpha_m) = 0$ soit respectée.

4.2.1. Obtention de la forme canonique

En appliquant le théorème 2 par rapport à x_1 puis x_2, toute équation $C(x_1, x_2, \alpha_1, \ldots, \alpha_m) = 0$ peut être ramenée à la forme suivante :

$$\overline{x_1} \cdot \overline{x_2} \cdot C(0, 0, \alpha_1, \ldots, \alpha_m) + \overline{x_1} \cdot x_2 \cdot C(0, 1, \alpha_1, \ldots, \alpha_m) + x_1 \cdot \overline{x_2} \cdot C(1, 0, \alpha_1, \ldots, \alpha_m) + x_1 \cdot x_2 \cdot C(1, 1, \alpha_1, \ldots, \alpha_m) = 0$$

En considérant comme connus les éléments $(\alpha_1, \ldots, \alpha_m)$, les éléments de $\mathcal{B}$ auxquels correspondent les compositions $C(0, 0, \alpha_1, \ldots, \alpha_m)$, $C(0, 1, \alpha_1, \ldots, \alpha_m)$, $C(1, 0, \alpha_1, \ldots, \alpha_m)$ et $C(1, 1, \alpha_1, \ldots, \alpha_m)$ sont alors des éléments parfaitement définis. Soient a, b, c et d ces éléments. En conséquence, toute équation à deux inconnues peut être reformulée sous la forme suivante :

$$a \cdot \overline{x_1} \cdot \overline{x_2} + b \cdot \overline{x_1} \cdot x_2 + c \cdot x_1 \cdot \overline{x_2} + d \cdot x_1 \cdot x_2 = 0 \qquad \text{[Eq. 1]}$$

4.2.2. Présentation du principe de résolution

Les couples (x_1, x_2) solution de l'équation [Eq. 1] peuvent être déterminés à partir des seuls résultats donnés dans le théorème 8 en résolvant de manière imbriquée deux équations à une seule inconnue de la forme suivante :

$$A_2 \cdot \overline{x_2} + B_2 \cdot x_2 = 0 \text{ avec } A_2 = a \cdot \overline{x_1} + c \cdot x_1 \text{ et } B_2 = b \cdot \overline{x_1} + d \cdot x_1 \qquad \text{[Eq. 2]}$$

$$A_1 \cdot \overline{x_1} + B_1 \cdot x_1 = 0 \text{ avec } A_1 = a \cdot b \text{ et } B_1 = c \cdot d \qquad \text{[Eq. 3]}$$

L'équation [Eq. 2] est une reformulation de l'équation [Eq. 1] faite en privilégiant l'inconnue x_2 par rapport à l'inconnue x_1. Dans cette équation, les termes A_2 et B_2 sont des compositions des éléments x_1, a, b, c et d. D'après le théorème 8, l'équation [Eq. 2] n'admet des solutions que si la condition $A_2 \cdot B_2 = 0$ est respectée.

L'équation [Eq. 3] est une reformulation de la condition $A_2 \cdot B_2 = 0$. Les termes A_1 et B_1 sont des compositions des seuls éléments a, b, c et d. D'après le théorème 8, l'équation [Eq. 3] n'admet des solutions que si la condition $A_1 \cdot B_1 = 0$ est respectée.

En résolvant l'équation [Eq. 3], il sera possible de déterminer l'ensemble des éléments x_1 de $\mathcal{B}$ pour lesquels l'équation [Eq. 2] admet des solutions. Une fois la forme paramétrique des solutions x_1 de l'équation [Eq. 3] reportée dans les termes A_2 et B_2, il sera alors possible de déterminer l'ensemble des éléments x_2 solution de l'équation [Eq. 2]. La forme paramétrique des solutions x_2 de l'équation [Eq. 2] fera référence à la forme paramétrique des solutions x_1 de l'équation [Eq. 3]. L'ensemble des couples (x_1, x_2) solution de l'équation [Eq. 1] sera décrit à l'aide de ces deux formes paramétriques.

Lorsque l'équation [Eq. 3] n'admet pas de solution (la condition $A_1 \cdot B_1 = 0$ n'est pas respectée), alors l'équation [Eq. 2] n'a pas de solution puisque la condition $A_2 \cdot B_2 = 0$ n'est pas respectée. Dans ce cas, il n'existe aucun couple (x_1, x_2) solution de l'équation [Eq. 1].

En utilisant les résultats du théorème 8, il est donc possible de déterminer l'ensemble des couples (x_1, x_2) solution de l'équation [Eq. 1] à l'aide d'une forme paramétrique comportant deux paramètres.

4.2.3. Théorème

Les résultats nécessaires pour résoudre dans une algèbre de Boole une équation à deux inconnues sont donnés dans le théorème suivant :

Théorème 9 : Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient x_1, x_2, a, b, c et d six éléments de cette algèbre. Considérons l'équation suivante où x_1 et x_2 sont des inconnues :

$$a \cdot \overline{x_1} \cdot \overline{x_2} + b \cdot \overline{x_1} \cdot x_2 + c \cdot x_1 \cdot \overline{x_2} + d \cdot x_1 \cdot x_2 = 0$$

Cette équation admet des solutions si et seulement si :

$$a \cdot b \cdot c \cdot d = 0$$

Lorsque cette condition est respectée, cette équation admet un ou plusieurs couples (x_1, x_2) solution donnés par la forme paramétrique suivante :

$$\begin{cases} x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\ x_2 = (a \cdot \overline{x_1} + c \cdot x_1) + \overline{(b \cdot \overline{x_1} + d \cdot x_1)} \cdot p_2 \end{cases}$$ où p_1 et p_2 sont des éléments quelconques de $\mathcal{B}$.

Soit, après substitution de x_1 par sa valeur :

$$\begin{cases} x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\ x_2 = (a \cdot c) + \overline{(b \cdot d)} \cdot (\bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2)) \end{cases}$$

où p_1 et p_2 sont des éléments quelconques de $\mathcal{B}$.

Démonstration :

La démonstration du théorème sera faite en suivant le principe de résolution proposé au paragraphe précédent.

L'équation à résoudre est :

$$a \cdot \overline{x_1} \cdot \overline{x_2} + b \cdot \overline{x_1} \cdot x_2 + c \cdot x_1 \cdot \overline{x_2} + d \cdot x_1 \cdot x_2 = 0 \qquad \text{[Eq. 4]}$$

Afin de pouvoir privilégier l'inconnue x_2 par rapport à l'inconnue x_1, il est nécessaire de reformuler l'équation [Eq. 4] ainsi :

$$(a \cdot \overline{x_1} + c \cdot x_1) \cdot \overline{x_2} + (b \cdot \overline{x_1} + d \cdot x_1) \cdot x_2 = 0 \qquad \text{[Eq. 5]}$$

D'après les résultats donnés dans le théorème 8, nous savons que :

- L'équation [Eq. 5] admet des solutions si et seulement si :

$$(a \cdot \overline{x_1} + c \cdot x_1) \cdot (b \cdot \overline{x_1} + d \cdot x_1) = 0$$

$$\Leftrightarrow (a \cdot b) \cdot \overline{x_1} + (a \cdot d) \cdot \overline{x_1} \cdot x_1 + (c \cdot b) \cdot \overline{x_1} \cdot x_1 + (c \cdot d) \cdot x_1 = 0$$

$$\Leftrightarrow (a \cdot b) \cdot \overline{x_1} + (c \cdot d) \cdot x_1 = 0 \qquad \text{[Eq. 6]}$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions :

$$x_2 = (a \cdot \overline{x_1} + c \cdot x_1) + \overline{(b \cdot \overline{x_1} + d \cdot x_1)} \cdot p_2 \text{ avec } p_2 \text{ un élément quelconque de } \mathcal{B}.$$

L'équation [Eq. 6] étant une équation de la seule inconnue x_1, d'après les résultats donnés dans le théorème 8, nous savons que :

- L'équation [Eq. 6] admet des solutions si et seulement si :

$$(a \cdot b) \cdot (c \cdot d) = 0$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions :

$$x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \text{ avec } p_1 \text{ un élément quelconque de } \mathcal{B}.$$

En conséquence, l'équation $a \cdot \overline{x_1} \cdot \overline{x_2} + b \cdot \overline{x_1} \cdot x_2 + c \cdot x_1 \cdot \overline{x_2} + d \cdot x_1 \cdot x_2 = 0$ admet des solutions si et seulement si l'équation [Eq. 6] admet des solutions, c'est-à-dire :

$$a \cdot b \cdot c \cdot d = 0$$

Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions :

$$\begin{cases} x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\ x_2 = (a \cdot \overline{x_1} + c \cdot x_1) + \overline{(b \cdot \overline{x_1} + d \cdot x_1)} \cdot p_2 \end{cases} \text{ avec } p_1 \text{ et } p_2 \text{ deux éléments quelconques de } \mathcal{B}.$$

Après substitution de x_1 par sa valeur, cette forme paramétrique est égale à :

$$\begin{cases} x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\ x_2 = (a \cdot c) + \overline{(b \cdot d)} \cdot (\bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2)) \end{cases}$$

où p_1 et p_2 sont des éléments quelconques de $\mathcal{B}$.

Cette forme est obtenue à partir de la précédente en effectuant les calculs suivants :

$$
\begin{aligned}
& a \cdot \overline{x_1} \\
&= a \cdot \overline{((a \cdot b) + \overline{(c \cdot d)} \cdot p_1)} \\
&= a \cdot ((\bar{a} + \bar{b}) \cdot ((c \cdot d) + \overline{p_1})) \\
&= a \cdot \bar{b} \cdot ((c \cdot d) + \overline{p_1}) \\
&= a \cdot \bar{b} \cdot c \cdot d + a \cdot \bar{b} \cdot \overline{p_1}
\end{aligned}
$$

$$
\begin{aligned}
& c \cdot x_1 \\
&= c \cdot ((a \cdot b) + \overline{(c \cdot d)} \cdot p_1) \\
&= a \cdot b \cdot c + c \cdot \bar{d} \cdot p_1
\end{aligned}
$$

Ce qui donne :

$$
\begin{aligned}
& a \cdot \overline{x_1} + c \cdot x_1 \\
&= a \cdot \bar{b} \cdot c \cdot d + a \cdot \bar{b} \cdot \overline{p_1} + a \cdot b \cdot c + c \cdot \bar{d} \cdot p_1 \\
&= a \cdot b \cdot c + a \cdot \bar{b} \cdot c \cdot d + (a \cdot \bar{b} \cdot \overline{p_1} + c \cdot \bar{d} \cdot p_1) \\
&= a \cdot b \cdot c + a \cdot \bar{b} \cdot c \cdot d + (a \cdot \bar{b} \cdot \overline{p_1} + c \cdot \bar{d} \cdot p_1 + a \cdot \bar{b} \cdot c \cdot \bar{d}) \\
&= a \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot \overline{p_1} + c \cdot \bar{d} \cdot p_1 \\
&= a \cdot c + a \cdot \bar{b} \cdot \overline{p_1} + c \cdot \bar{d} \cdot p_1
\end{aligned}
$$

De même :

$$
\begin{aligned}
& \overline{(b \cdot \overline{x_1} + d \cdot x_1)} \\
&= (\bar{b} + x_1) \cdot (\bar{d} + \overline{x_1}) \\
&= \bar{b} \cdot \bar{d} + \bar{d} \cdot x_1 + \bar{b} \cdot \overline{x_1} \\
&= \bar{b} \cdot \bar{d} + \bar{d} \cdot ((a \cdot b) + \overline{(c \cdot d)} \cdot p_1) + \bar{b} \cdot \overline{((a \cdot b) + \overline{(c \cdot d)} \cdot p_1)} \\
&= \bar{b} \cdot \bar{d} + a \cdot b \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot (\bar{a} + \bar{b}) \cdot ((c \cdot d) + \overline{p_1}) \\
&= \bar{b} \cdot \bar{d} + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot ((c \cdot d) + \overline{p_1}) \\
&= \bar{b} \cdot \bar{d} + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot c \cdot d + \bar{b} \cdot \overline{p_1} \\
&= \bar{b} \cdot \bar{d} + \bar{b} \cdot c \cdot d + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot \overline{p_1} \\
&= \bar{b} \cdot \bar{d} + \bar{b} \cdot c + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot \overline{p_1} \\
&= \bar{b} \cdot c + a \cdot \bar{d} + (\bar{b} \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot \overline{p_1}) \\
&= \bar{b} \cdot c + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot \overline{p_1}
\end{aligned}
$$

Finalement :

$$
\begin{aligned}
x_2 &= (a \cdot \overline{x_1} + c \cdot x_1) + \overline{(b \cdot \overline{x_1} + d \cdot x_1)} \cdot p_2 \\
&= (a \cdot c + a \cdot \bar{b} \cdot \overline{p_1} + c \cdot \bar{d} \cdot p_1) + (\bar{b} \cdot c + a \cdot \bar{d} + \bar{d} \cdot p_1 + \bar{b} \cdot \overline{p_1}) \cdot p_2 \\
&= a \cdot c + (a \cdot \bar{b} \cdot \overline{p_1} + \bar{b} \cdot c \cdot p_2 + \bar{b} \cdot \overline{p_1} \cdot p_2) + (c \cdot \bar{d} \cdot p_1 + a \cdot \bar{d} \cdot p_2 + \bar{d} \cdot p_1 \cdot p_2) \\
&= a \cdot c + \bar{b} \cdot (a \cdot \overline{p_1} + c \cdot p_2 + \overline{p_1} \cdot p_2) + \bar{d} \cdot (c \cdot p_1 + a \cdot p_2 + p_1 \cdot p_2) \\
&= a \cdot c + \bar{b} \cdot (a \cdot c + a \cdot \overline{p_1} + c \cdot p_2 + \overline{p_1} \cdot p_2) + \bar{d} \cdot (a \cdot c + c \cdot p_1 + a \cdot p_2 + p_1 \cdot p_2) \\
&= a \cdot c + \bar{b} \cdot ((a + p_2) \cdot (c + \overline{p_1})) + \bar{d} \cdot ((a + p_1) \cdot (c + p_2)) \\
&= a \cdot c + (\bar{b} + \bar{d}) \cdot \bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + (\bar{b} + \bar{d}) \cdot \bar{d} \cdot (a + p_1) \cdot (c + p_2) \\
&= a \cdot c + \overline{(b \cdot d)} \cdot (\bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2))
\end{aligned}
$$

□

4.2.4. Discussion

Le principe proposé pour résoudre une équation à deux inconnues repose sur la possibilité de ramener ce problème à la résolution successive de deux équations à une inconnue. Selon l'ordre choisi pour les inconnues, les formes paramétriques proposées pour les couples solution diffèrent :

- Ordre x_1 puis x_2 :

$$
\begin{cases}
x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\
x_2 = (a \cdot c) + \overline{(b \cdot d)} \cdot (\bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2))
\end{cases}
$$

- Ordre x_2 puis x_1 :

$$
\begin{cases}
x_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot (\bar{c} \cdot (a + q_1) \cdot (b + \overline{q_2}) + \bar{d} \cdot (a + q_2) \cdot (b + q_1)) \\
x_2 = (a \cdot c) + \overline{(b \cdot d)} \cdot q_2
\end{cases}
$$

Ces deux formes sont cependant bien équivalentes car il est possible de passer de l'une à l'autre via le changement de variables suivant :

$$
\begin{cases}
q_1 = p_1 \\
q_2 = \bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2)
\end{cases}
$$

Démonstration :

Au vu du changement de variables proposé, le passage d'une forme à l'autre pour la variable x_2 est évident. Nous allons montrer que c'est également le cas pour x_1 :

$$(a \cdot b) + \overline{(c \cdot d)} \cdot p_1 = (a \cdot b) + \overline{(c \cdot d)} \cdot (\bar{c} \cdot (a + q_1) \cdot (b + \overline{q_2}) + \bar{d} \cdot (a + q_2) \cdot (b + q_1))$$

$$\text{avec} \begin{cases} q_1 = p_1 \\ q_2 = \bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2) \end{cases}$$

En effet :

$$\begin{aligned}
&\bar{c} \cdot (a + q_1) \cdot (b + \overline{q_2}) \\
&= \bar{c} \cdot (a + p_1) \cdot (b + \overline{(\bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2))}) \\
&= \bar{c} \cdot (a + p_1) \cdot (b + (b + \bar{a} \cdot \overline{p_2} + \bar{c} \cdot p_1) \cdot (d + \bar{a} \cdot \overline{p_1} + \bar{c} \cdot \overline{p_2})) \\
&= \bar{c} \cdot (a + p_1) \cdot (b + (\bar{a} \cdot \overline{p_2} + \bar{c} \cdot p_1) \cdot (d + \bar{a} \cdot \overline{p_1} + \bar{c} \cdot \overline{p_2})) \\
&= (a + p_1) \cdot (b \cdot \bar{c} + \bar{c} \cdot (\bar{a} \cdot \overline{p_2} + \bar{c} \cdot p_1) \cdot (d + \bar{a} \cdot \overline{p_1} + \bar{c} \cdot \overline{p_2})) \\
&= (a + p_1) \cdot (b \cdot \bar{c} + \bar{c} \cdot (\bar{a} \cdot \overline{p_2} + p_1) \cdot (d + \bar{a} \cdot \overline{p_1} + \overline{p_2})) \\
&= (a + p_1) \cdot (b \cdot \bar{c} + \bar{c} \cdot (\bar{a} \cdot \overline{p_2} + d \cdot p_1 + p_1 \cdot \overline{p_2})) \\
&= a \cdot b \cdot \bar{c} + a \cdot \bar{c} \cdot d \cdot p_1 + a \cdot \bar{c} \cdot p_1 \cdot \overline{p_2} + b \cdot \bar{c} \cdot p_1 + \bar{a} \cdot \bar{c} \cdot p_1 \cdot \overline{p_2} + \bar{c} \cdot d \cdot p_1 + \bar{c} \cdot p_1 \cdot \overline{p_2} \\
&= a \cdot b \cdot \bar{c} + \bar{c} \cdot d \cdot p_1 + b \cdot \bar{c} \cdot p_1 + \bar{c} \cdot p_1 \cdot \overline{p_2} \\
&= \bar{c} \cdot (a \cdot b + (b + d + \overline{p_2}) \cdot p_1)
\end{aligned}$$

De même :

$$\begin{aligned}
&\bar{d} \cdot (a + q_2) \cdot (b + q_1) \\
&= \bar{d} \cdot (a + \bar{b} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2)) \cdot (b + p_1) \\
&= (a \cdot \bar{d} + \bar{b} \cdot \bar{d} \cdot (a + p_2) \cdot (c + \overline{p_1}) + \bar{d} \cdot (a + p_1) \cdot (c + p_2)) \cdot (b + p_1) \\
&= (a \cdot \bar{d} + \bar{b} \cdot \bar{d} \cdot p_2 \cdot (c + \overline{p_1}) + \bar{d} \cdot p_1 \cdot (c + p_2)) \cdot (b + p_1) \\
&= a \cdot b \cdot \bar{d} + b \cdot \bar{d} \cdot p_1 \cdot (c + p_2) + a \cdot \bar{d} \cdot p_1 + \bar{b} \cdot c \cdot \bar{d} \cdot p_1 \cdot p_2 + \bar{d} \cdot p_1 \cdot (c + p_2) \\
&= a \cdot b \cdot \bar{d} + b \cdot \bar{d} \cdot p_1 \cdot (c + p_2) + a \cdot \bar{d} \cdot p_1 + \bar{b} \cdot c \cdot \bar{d} \cdot p_1 \cdot p_2 + c \cdot \bar{d} \cdot p_1 + \bar{d} \cdot p_1 \cdot p_2 \\
&= a \cdot b \cdot \bar{d} + a \cdot \bar{d} \cdot p_1 + c \cdot \bar{d} \cdot p_1 + \bar{d} \cdot p_1 \cdot p_2 \\
&= \bar{d} \cdot (a \cdot b + (a + c + p_2) \cdot p_1)
\end{aligned}$$

Finalement :

$$
\begin{aligned}
&(a \cdot b) + \overline{(c \cdot d)} \cdot (\bar{c} \cdot (a + q_1) \cdot (b + \overline{q_2}) + \bar{d} \cdot (a + q_2) \cdot (b + q_1)) \\
&\quad = (a \cdot b) + \overline{(c \cdot d)} \cdot ((\bar{c} \cdot (a \cdot b + (b + d + \overline{p_2}) \cdot p_1)) + (\bar{d} \cdot (a \cdot b + (a + c + p_2) \cdot p_1))) \\
&\quad = (a \cdot b) + (\bar{c} \cdot (a \cdot b + (b + d + \overline{p_2}) \cdot p_1)) + (\bar{d} \cdot (a \cdot b + (a + c + p_2) \cdot p_1)) \\
&\quad = (a \cdot b) + \bar{c} \cdot p_1 \cdot (b + d + \overline{p_2}) + \bar{d} \cdot p_1 \cdot (a + c + p_2) \\
&\quad = (a \cdot b) + p_1 \cdot (b \cdot \bar{c} + \bar{c} \cdot d + \bar{c} \cdot \overline{p_2} + a \cdot \bar{d} + c \cdot \bar{d} + \bar{d} \cdot p_2) \\
&\quad = (a \cdot b) + p_1 \cdot (a \cdot \bar{d} + b \cdot \bar{c} + \bar{c} \cdot d + c \cdot \bar{d} + (\bar{c} \cdot \overline{p_2} + \bar{d} \cdot p_2)) \\
&\quad = (a \cdot b) + p_1 \cdot (a \cdot \bar{d} + b \cdot \bar{c} + \bar{c} \cdot d + c \cdot \bar{d} + (\bar{c} \cdot \bar{d} + \bar{c} \cdot \overline{p_2} + \bar{d} \cdot p_2)) \\
&\quad = (a \cdot b) + p_1 \cdot (a \cdot \bar{d} + b \cdot \bar{c} + \bar{c} + \bar{d} + \bar{c} \cdot \overline{p_2} + \bar{d} \cdot p_2) \\
&\quad = (a \cdot b) + p_1 \cdot (\bar{c} + \bar{d}) \\
&\quad = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1
\end{aligned}
$$

□

L'ordre de résolution n'a donc bien aucun impact sur l'espace solution mais seulement sur l'écriture de la forme paramétrique permettant de balayer cet espace solution.

Toute équation comportant une ou deux inconnues est résolvable. Dans la suite, le cas général des équations comportant n inconnues est étudié.

4.3. Résolution d'une équation à *n* inconnues

4.3.1. Obtention de la forme canonique

Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient $(\alpha_1, \ldots, \alpha_m)$ m éléments de $\mathcal{B}$ supposés connus. Soient $(x_1, \ldots, x_n)$ n éléments de $\mathcal{B}$ considérés comme inconnus. Soit l'équation suivante liant ces éléments :

$$C(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = 0$$

En appliquant le théorème 2 par rapport à chacune des inconnues x_i, cette équation peut être ramenée à la forme suivante :

$$\begin{aligned}&(\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot \overline{x_n}) \cdot C(0, \ldots, 0, 0, \alpha_1, \ldots, \alpha_m)\\ &+ (\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot x_n) \cdot C(0, \ldots, 0, 1, \alpha_1, \ldots, \alpha_m) + \ldots\\ &+ (x_1 \cdot \ldots \cdot x_{n-1} \cdot \overline{x_n}) \cdot C(1, \ldots, 1, 0, \alpha_1, \ldots, \alpha_m)\\ &+ (x_1 \cdot \ldots \cdot x_{n-1} \cdot x_n) \cdot C(1, \ldots, 1, 1, \alpha_1, \ldots, \alpha_m) = 0\end{aligned}$$

Cette somme comporte 2^n termes (autant de termes que de combinaisons différentes des inconnues x_i). Chacun de ces termes (par exemple, le deuxième terme $(\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot x_n) \cdot C(0, \ldots, 0, 1, \alpha_1, \ldots, \alpha_m)$) est le produit :

- d'une combinaison donnée des inconnues x_i : $(\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot x_n)$,
- de la composition correspondante des seuls éléments connus : $C(0, \ldots, 0, 1, \alpha_1, \ldots, \alpha_m)$.

Il est possible d'exprimer cette somme à l'aide d'une notation générique en s'appuyant sur le fait que chacun de ses termes peut être formulé à partir d'un n-uplet ω_j dont chaque composante a pour valeur 0 ou 1. Soit Ω_n l'ensemble de ces 2^n n-uplets ω_j.

$$\Omega_n = \{\omega_j = (\omega_j^1, \ldots, \omega_j^k, \ldots, \omega_j^n), \omega_j^k \in \{0, 1\}\}$$

Ainsi le terme $(\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot x_n) \cdot C(0, \ldots, 0, 1, \alpha_1, \ldots, \alpha_m)$ s'exprime en fonction de $\omega_j = (0, \ldots, 0, 1)$, ainsi :

- Pour la combinaison des inconnues x_i, nous avons :

$$(\overline{x_1} \cdot \ldots \cdot \overline{x_{n-1}} \cdot x_n) = \prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \text{ avec } \omega_j = (0, \ldots, 0, 1)$$

- Pour la composition des éléments connus, nous avons :

$$C(0, \ldots, 0, 1, \alpha_1, \ldots, \alpha_m) = C(\omega_j^1, \ldots, \omega_j^n, \alpha_1, \ldots, \alpha_m) \text{ avec } \omega_j = (0, \ldots, 0, 1)$$

En conséquence, pour la somme, nous avons :

$$C(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = \sum_{\omega_j \in \Omega_n} C(\omega_j^1, \ldots, \omega_j^n, \alpha_1, \ldots, \alpha_m) \cdot \left(\prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right)$$

C'est en utilisant cette notation que nous allons maintenant exprimer le théorème permettant la résolution d'une équation à n inconnues.

4.3.2. Théorème

Les résultats nécessaires pour résoudre dans une algèbre de Boole une équation à n inconnues sont donnés dans le théorème suivant :

Théorème 10 : Soit $(\mathcal{B}, +, \cdot, \bar{}, 0, 1)$ une algèbre de Boole. Soient $(x_1, \ldots, x_n)$ et $(\alpha_1, \ldots, \alpha_m)$ $(n+m)$ éléments de $\mathcal{B}$. Considérons l'équation suivante où les éléments x_i sont inconnus :

$$C_0(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = 0$$

Cette équation admet des solutions si et seulement si :

$$\prod_{\omega_j \in \Omega_n} C_0(\omega_j^1, \ldots, \omega_j^n, \alpha_1, \ldots, \alpha_m) = 0$$

où Ω_n est l'ensemble des 2^n n-uplets ω_j où chaque composante est l'un des éléments particuliers 0 et 1.

$$\Omega_n = \{\omega_j = (\omega_j^1, \ldots, \omega_j^k, \ldots, \omega_j^n), \omega_j^k \in \{0, 1\}\}$$

Lorsque cette condition est respectée, cette équation admet un ou plusieurs n-uplets solution $(S(x_1), \ldots, S(x_n))$ où chaque composante peut être décrite à l'aide de la forme paramétrique suivante :

$$S(x_i) = \prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 0}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m) + \overline{\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 1}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)} \cdot p_i$$

(p_i est un élément quelconque de $\mathcal{B}$)

où

$$C_i(x_{i+1}, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = C_{i-1}(x_i, x_{i+1}, \ldots, x_n, \alpha_1, \ldots, \alpha_m)\big|_{x_i \leftarrow S(x_i)}$$

Démonstration :

La démonstration de ce théorème se fera par récurrence et comportera deux temps :

- Dans un premier temps, nous montrerons que ce théorème est vrai à l'ordre 1 et à l'ordre 2 en comparant les résultats à ceux donnés dans les théorèmes 8 et 9.
- Dans un deuxième temps, nous le supposerons vrai à l'ordre n et nous montrerons qu'il est vrai à l'ordre $n+1$.

Une fois ces deux résultats démontrés, nous pourrons alors affirmer que ce théorème est démontré pour tout n, c'est-à-dire quel que soit le nombre d'inconnues.

- **Démonstration à l'ordre 1** (une seule inconnue).

$$C_0(x_1, \alpha_1, \ldots, \alpha_m) = \sum_{\omega_j \in \Omega_1} C_0(\omega_j^1, \alpha_1, \ldots, \alpha_m) \cdot (\overline{\omega_j^1} \cdot \overline{x_1} + \omega_j^1 \cdot x_1) = 0 \text{ avec}$$

$$\Omega_1 = \{(0), (1)\}.$$

$$\Leftrightarrow C_0(0, \alpha_1, \ldots, \alpha_m) \cdot \overline{x_1} + C_0(1, \alpha_1, \ldots, \alpha_m) \cdot x_1 = 0 \qquad \text{[Eq. 7]}$$

Les résultats donnés dans le théorème 8 permettent de résoudre cette équation une fois le changement de variables suivant effectué :

$$\begin{cases} C_0(0, \alpha_1, \ldots, \alpha_m) = a \\ C_0(1, \alpha_1, \ldots, \alpha_m) = b \end{cases}$$

D'après le théorème 8, nous savons que :

- L'équation [Eq. 7] admet des solutions si et seulement si :

$$a \cdot b = 0$$

$$\Leftrightarrow C_0(0, \alpha_1, \ldots, \alpha_m) \cdot C_0(1, \alpha_1, \ldots, \alpha_m) = 0$$

$$\Leftrightarrow \prod_{\omega_j \in \Omega_1} C_0(\omega_j^1, \alpha_1, \ldots, \alpha_m) = 0$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions (p est un élément quelconque de $\mathcal{B}$) :

$$S(x_1) = a + \bar{b} \cdot p$$

$$\Leftrightarrow S(x_1) = C_0(0, \alpha_1, \ldots, \alpha_m) + \overline{C_0(1, \alpha_1, \ldots, \alpha_m)} \cdot p$$

$$\Leftrightarrow S(x_1) = \prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 0}} C_0(\omega_j^1, \alpha_1, \ldots, \alpha_m) + \overline{\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 1}} C_0(\omega_j^1, \alpha_1, \ldots, \alpha_m)} \cdot p$$

Les résultats donnés dans le théorème 10 à l'ordre 1 étant identiques à ceux donnés dans le théorème 8, le théorème 10 est donc vrai à l'ordre 1.

- **Démonstration à l'ordre 2** (deux inconnues).

$$C_0(x_1, x_2, \alpha_1, \ldots, \alpha_m) = \sum_{\omega_j \in \Omega_2} C_0(\omega_j^1, \omega_j^2, \alpha_1, \ldots, \alpha_m) \cdot \left(\prod_{k=1}^{k=2} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right) = 0$$

$$\text{avec } \Omega_2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$$

$$\begin{aligned} &\Leftrightarrow C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{x_1} \cdot \overline{x_2} + C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot \overline{x_1} \cdot x_2 \\ &+ C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot x_1 \cdot \overline{x_2} + C_0(1, 1, \alpha_1, \ldots, \alpha_m) \cdot x_1 \cdot x_2 = 0 \end{aligned} \quad \text{[Eq. 8]}$$

Les résultats donnés dans le théorème 9 permettent de résoudre cette équation une fois le changement de variables suivant effectué :

$$\begin{cases} C_0(0, 0, \alpha_1, \ldots, \alpha_m) = a \\ C_0(0, 1, \alpha_1, \ldots, \alpha_m) = b \\ C_0(1, 0, \alpha_1, \ldots, \alpha_m) = c \\ C_0(1, 1, \alpha_1, \ldots, \alpha_m) = d \end{cases}$$

D'après le théorème 9, nous savons que :

- L'équation [Eq. 8] admet des solutions si et seulement si :

$$a \cdot b \cdot c \cdot d = 0$$

$$\Leftrightarrow C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(1, 1, \alpha_1, \ldots, \alpha_m) = 0$$

$$\Leftrightarrow \prod_{\omega_j \in \Omega_2} C_0(\omega_j^1, \omega_j^2, \alpha_1, \ldots, \alpha_m) = 0$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions (p_1 et p_2 sont des éléments quelconques de $\mathcal{B}$).

$$\begin{cases} S(x_1) = (a \cdot b) + \overline{(c \cdot d)} \cdot p_1 \\ S(x_2) = (a \cdot \overline{S(x_1)} + c \cdot S(x_1)) + \overline{(b \cdot \overline{S(x_1)} + d \cdot S(x_1))} \cdot p_2 \end{cases}$$

Concernant $S(x_1)$:

$$
\begin{aligned}
&S(x_1)\\
&= (a \cdot b) + \overline{(c \cdot d)} \cdot p_1\\
&= (C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(0, 1, \alpha_1, \ldots, \alpha_m))\\
&\quad + \overline{(C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(1, 1, \alpha_1, \ldots, \alpha_m))} \cdot p_1\\
&= \left(\prod_{\substack{\omega_j \in \Omega_2 \\ \omega_j^1 = 0}} C_0(\omega_j^1, \omega_j^2, \alpha_1, \ldots, \alpha_m) \right) + \overline{\left(\prod_{\substack{\omega_j \in \Omega_2 \\ \omega_j^1 = 1}} C_0(\omega_j^1, \omega_j^2, \alpha_1, \ldots, \alpha_m) \right)} \cdot p_1
\end{aligned}
$$

Concernant $S(x_2)$:

Par définition :

$$
\begin{aligned}
&C_1(x_2, \alpha_1, \ldots, \alpha_m)\\
&= C_0(x_1, x_2, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)}\\
&= C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} \cdot \overline{x_2} + C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} \cdot x_2\\
&\quad + C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot S(x_1) \cdot \overline{x_2} + C_0(1, 1, \alpha_1, \ldots, \alpha_m) \cdot S(x_1) \cdot x_2\\
&= (C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot S(x_1)) \cdot \overline{x_2}\\
&\quad + (C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 1, \alpha_1, \ldots, \alpha_m) \cdot S(x_1)) \cdot x_2
\end{aligned}
$$

Par définition, nous avons :

$$
\begin{cases}
C_1(0, \alpha_1, \ldots, \alpha_m) = C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot S(x_1)\\
C_1(1, \alpha_1, \ldots, \alpha_m) = C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 1, \alpha_1, \ldots, \alpha_m) \cdot S(x_1)
\end{cases}
$$

En conséquence :

$$
\begin{aligned}
&S(x_2)\\
&= (a \cdot \overline{S(x_1)} + c \cdot S(x_1)) + \overline{(b \cdot \overline{S(x_1)} + d \cdot S(x_1))} \cdot p_2\\
&= (C_0(0, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 0, \alpha_1, \ldots, \alpha_m) \cdot S(x_1))\\
&\quad + \overline{(C_0(0, 1, \alpha_1, \ldots, \alpha_m) \cdot \overline{S(x_1)} + C_0(1, 1, \alpha_1, \ldots, \alpha_m) \cdot S(x_1))} \cdot p_2\\
&= C_1(0, \alpha_1, \ldots, \alpha_m) + \overline{(C_1(1, \alpha_1, \ldots, \alpha_m))} \cdot p_2\\
&= \left(\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 0}} C_1(\omega_j^1, \alpha_1, \ldots, \alpha_m) \right) + \overline{\left(\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 1}} C_1(\omega_j^1, \alpha_1, \ldots, \alpha_m) \right)} \cdot p_2
\end{aligned}
$$

Les résultats donnés dans le théorème 10 à l'ordre 2 étant identiques à ceux donnés dans le théorème 9, le théorème 10 est donc vrai à l'ordre 2.

- **Démonstration de la récurrence.**

Supposons le théorème vrai à l'ordre n pour toute équation :

$$D_0(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = 0$$

Considérons l'équation suivante comportant $(n+1)$ inconnues :

$$C_0(x_1, \ldots, x_{n+1}, \alpha_1, \ldots, \alpha_m) = 0$$

Par application du théorème 2 par rapport à x_{n+1}, nous avons :

$$C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot \overline{x_{n+1}} + C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) \cdot x_{n+1} = 0 \qquad \text{[Eq. 9]}$$

D'après les résultats donnés par le théorème 8, nous savons que :

- L'équation [Eq. 9] admet des solutions si et seulement si :

$$C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) = 0 \qquad \text{[Eq. 10]}$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions (p_{n+1} est un élément quelconque de $\mathcal{B}$).

$$S(x_{n+1}) = C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) + \overline{C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)} \cdot p_{n+1}$$

L'équation [Eq. 10] est une équation à seulement n inconnues pour laquelle les résultats du théorème 10 à l'ordre n peuvent s'appliquer :

Par définition, nous avons :

$$C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) = \sum_{\omega_j \in \Omega_n} C_0(\omega_j^1, \ldots, \omega_j^n, 0, \alpha_1, \ldots, \alpha_m) \cdot \left(\prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right)$$

$$C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) = \sum_{\omega_j \in \Omega_n} C_0(\omega_j^1, \ldots, \omega_j^n, 1, \alpha_1, \ldots, \alpha_m) \cdot \left(\prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right)$$

En conséquence, nous avons :

$$C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)$$

$$= \sum_{\omega_j \in \Omega_n} C_0(\omega_j^1, \ldots, \omega_j^n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(\omega_j^1, \ldots, \omega_j^n, 1, \alpha_1, \ldots, \alpha_m) \cdot \left(\prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right)$$

$$\text{car } \forall \omega_i, \omega_j, \quad i \neq j, \quad \left(\prod_{k=1}^{k=n} (\overline{\omega_i^k} \cdot \overline{x_k} + \omega_i^k \cdot x_k) \right) \cdot \left(\prod_{k=1}^{k=n} (\overline{\omega_j^k} \cdot \overline{x_k} + \omega_j^k \cdot x_k) \right) = 0$$

Soit $\forall \omega_j \in \Omega_n$:

$$D_0(\omega_j^1, \ldots, \omega_j^n, \alpha_1, \ldots, \alpha_m) = C_0(\omega_j^1, \ldots, \omega_j^n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(\omega_j^1, \ldots, \omega_j^n, 1, \alpha_1, \ldots, \alpha_m)$$

D'après les résultats donnés dans le théorème 10 à l'ordre n, nous savons que :

- L'équation [Eq. 10] admet des solutions si et seulement si :

$$\prod_{\omega_j \in \Omega_n} D_0(\omega_j^1, \ldots, \omega_j^n, \alpha_1, \ldots, \alpha_m) = 0$$

$$\Leftrightarrow \prod_{\omega_j \in \Omega_n} C_0(\omega_j^1, \ldots, \omega_j^n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(\omega_j^1, \ldots, \omega_j^n, 1, \alpha_1, \ldots, \alpha_m) = 0$$

Sachant que :

$$\Omega_{n+1} = \left(\bigcup_{\omega_j \in \Omega_n} (\omega_j^1, \ldots, \omega_j^n, 0) \right) \cup \left(\bigcup_{\omega_j \in \Omega_n} (\omega_j^1, \ldots, \omega_j^n, 1) \right)$$

On a :

$$\prod_{\omega_j \in \Omega_{n+1}} C_0(\omega_j^1, \ldots, \omega_j^n, \omega_j^{n+1}, \alpha_1, \ldots, \alpha_m) = 0$$

- Lorsque cette condition est respectée, la forme paramétrique suivante permet de décrire l'ensemble des solutions (les p_i sont des éléments quelconques de $\mathcal{B}$).

$$S(x_i) = \prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 0}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m) + \overline{\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 1}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)} \cdot p_i$$

où

$$D_i(x_{i+1}, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = D_{i-1}(x_i, x_{i+1}, \ldots, x_n, \alpha_1, \ldots, \alpha_m)\big|_{x_i \leftarrow S(x_i)}$$

Posons :

$$C_i(x_{i+1}, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) = C_{i-1}(x_i, x_{i+1}, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m)\big|_{x_i \leftarrow S(x_i)}$$

$$C_i(x_{i+1}, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) = C_{i-1}(x_i, x_{i+1}, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)\big|_{x_i \leftarrow S(x_i)}$$

Les termes D_i peuvent être directement obtenus à partir des termes C_i car nous avons par construction :

- $D_0(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)$
- $D_1(x_2, \ldots, x_n, \alpha_1, \ldots, \alpha_m)$
  $= D_0(x_1, x_2, \ldots, x_n, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)}$
  $= (C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m))\big|_{x_1 \leftarrow S(x_1)}$
  $= C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)} \cdot C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)}$
  $= C_1(x_2, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_1(x_2, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)$

Par itération, nous avons : $\forall i \in [0, n-1]$

$$D_i(x_{i+1}, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = C_i(x_{i+1}, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) \cdot C_i(x_{i+1}, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)$$

En reportant ce résultat dans les termes qui constituent $S(x_i)$, nous avons :

- $$\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 0}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)$$

  $$= \prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 0}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, 0, \alpha_1, \ldots, \alpha_m) \cdot C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, 1, \alpha_1, \ldots, \alpha_m)$$

  $$= \prod_{\substack{\omega_j \in \Omega_{n+2-i} \\ \omega_j^1 = 0}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \omega_j^{n+2-i}, \alpha_1, \ldots, \alpha_m)$$

- $$\overline{\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 1}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)}$$

$$= \overline{\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 1}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, 0, \alpha_1, \ldots, \alpha_m) \cdot C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, 1, \alpha_1, \ldots, \alpha_m)}$$

$$= \overline{\prod_{\substack{\omega_j \in \Omega_{n+2-i} \\ \omega_j^1 = 1}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \omega_j^{n+2-i}, \alpha_1, \ldots, \alpha_m)}$$

En conséquence nous avons : $\forall i \in [1, n]$

- $S(x_i)$

$$= \prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 0}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)$$

$$+ \overline{\prod_{\substack{\omega_j \in \Omega_{n+1-i} \\ \omega_j^1 = 1}} D_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \alpha_1, \ldots, \alpha_m)} \cdot p_i$$

$$= \prod_{\substack{\omega_j \in \Omega_{n+2-i} \\ \omega_j^1 = 0}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \omega_j^{n+2-i}, \alpha_1, \ldots, \alpha_m)$$

$$+ \overline{\prod_{\substack{\omega_j \in \Omega_{n+2-i} \\ \omega_j^1 = 1}} C_{i-1}(\omega_j^1, \ldots, \omega_j^{n+1-i}, \omega_j^{n+2-i}, \alpha_1, \ldots, \alpha_m)} \cdot p_i$$

Le dernier point de la démonstration consiste à reformuler $S(x_{n+1})$ pour l'exprimer comme les solutions précédentes. Nous avons :

$$S(x_{n+1}) = C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) + \overline{C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)} \cdot p_{n+1}$$

où $(x_1, \ldots, x_n)$ sont les solutions de :

$$D_0(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = 0$$

Les compositions $C_0(x_1, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m)$ et $C_0(x_1, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)$ sont obtenues après remplacement de x_1 par $S(x_1)$, puis x_2 par $S(x_2)$... et enfin x_n par $S(x_n)$.

Nous avons par construction :

- $C_1(x_2, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) = C_0(x_1, x_2, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)}$

  $C_1(x_2, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) = C_0(x_1, x_2, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)\big|_{x_1 \leftarrow S(x_1)}$

- $C_2(x_3, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m) = C_1(x_2, x_3, \ldots, x_n, 0, \alpha_1, \ldots, \alpha_m)\big|_{x_2 \leftarrow S(x_2)}$

  $C_2(x_3, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m) = C_1(x_2, x_3, \ldots, x_n, 1, \alpha_1, \ldots, \alpha_m)\big|_{x_2 \leftarrow S(x_2)}$

- ...

- $C_n(0, \alpha_1, \ldots, \alpha_m) = C_{n-1}(x_n, 0, \alpha_1, \ldots, \alpha_m)\big|_{x_n \leftarrow S(x_n)}$

  $C_n(1, \alpha_1, \ldots, \alpha_m) = C_{n-1}(x_n, 1, \alpha_1, \ldots, \alpha_m)\big|_{x_n \leftarrow S(x_n)}$

Donc :

$$S(x_{n+1}) = C_n(0, \alpha_1, \ldots, \alpha_m) + \overline{C_n(1, \alpha_1, \ldots, \alpha_m)} \cdot p_{n+1}$$

$$\Leftrightarrow S(x_{n+1}) = \prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 0}} C_n(\omega_j^1, \alpha_1, \ldots, \alpha_m) + \overline{\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = 1}} C_n(\omega_j^1, \alpha_1, \ldots, \alpha_m)} \cdot p_{n+1}$$

Les résultats obtenus sont les mêmes que ceux du théorème 10 à l'ordre $(n+1)$. Le théorème 10 est donc vrai pour tout n, c'est-à-dire quel que soit le nombre d'inconnues.

□

5. Conclusion

Ce chapitre a été l'occasion de faire une synthèse des résultats existants et de ceux que nous avons obtenus sur les algèbres de Boole. Dans une première partie, la définition de ce type d'algèbre a été rappelée. Puis les propriétés connues pour les variables booléennes ont été étendues à l'en-

semble de ces algèbres. Cela va du théorème de De Morgan jusqu'à l'expansion de Shannon. La plupart de ces résultats n'avaient été démontrés que pour les variables booléennes et de manière énumérative. Elles le sont dorénavant quel que soit le nombre d'éléments et pour toutes les algèbres de Boole.

Dans une seconde partie, nous avons présenté les différentes relations qui peuvent être utilisées sur une algèbre de boole. Ces relations permettront, comme nous le verrons dans le chapitre 3, d'exprimer les informations du cahier des charges, c'est-à-dire les attentes que devra satisfaire le système logique que nous cherchons à commander.

Dans la dernière partie du chapitre, nous avons exposé et démontré des résultats qui permettent de résoudre un système d'équations en le transformant en une seule équation. Puis la résolution d'une équation à n inconnues a été présentée et prouvée.

Grâce à ces résultats, nous allons pouvoir, dans le chapitre suivant, présenter comment il est possible de synthétiser la commande d'un système logique. Pour ce faire, trois exemples de complexité croissante sont traités afin d'illustrer cette nouvelle méthode.

Chapitre 3 :
Synthèse de la loi de commande par résolution d'un système d'équations entre des fonctions booléennes

Dans ce chapitre, nous allons tout d'abord détailler les différentes étapes de notre méthode de synthèse. Nous illustrerons ensuite cette méthode au travers du traitement de trois études de cas de taille et de complexité croissante :

- La première étude de cas est la synthèse d'un problème purement combinatoire. Nous reprendrons l'exemple présenté au chapitre 1 pour le traiter selon notre approche. Cette étude de cas sera entièrement traitée de manière manuelle pour permettre au lecteur de s'approprier cette méthode de synthèse.
- La deuxième étude de cas est la synthèse d'un problème séquentiel. Nous nous proposons de synthétiser les conditions de franchissement dans un modèle à états. À travers cet exemple, nous montrerons que notre méthode de synthèse permet au concepteur de compléter les spécifications par des éléments de solutions lorsqu'il identifie des problèmes bien connus.
- La troisième étude de cas est également la synthèse d'un problème séquentiel. L'intérêt de cette étude réside dans sa taille et dans la nature des données de départ. Dans cette étude, nous travaillerons à partir de données fonctionnelles.

1. Le détail des étapes de la méthode de synthèse

La méthode de synthèse de la loi de commande d'un SED logique décrite dans ce mémoire de thèse est construite autour des résultats théoriques généraux présentés au chapitre précédent qui permettent la résolution analytique de systèmes d'équations dans une algèbre de Boole quelconque.

Dans ce chapitre, nous allons travailler sur l'algèbre de Boole des fonctions booléennes de dimension d donnée. Nous avons retenu cette algèbre de Boole pour les raisons suivantes :

- Nous avons montré au chapitre 1 que le comportement d'un système logique, qu'il soit combinatoire ou séquentiel, peut être entièrement défini par la donnée de chacune des fonctions booléennes qui permet de calculer la valeur de ses sorties ou des variables internes.
- Nous montrerons dans ce chapitre 3 que les fragments de spécification contenus dans un cahier des charges peuvent être exprimés sous la forme de relations entre des fonctions booléennes et ensuite assemblés sous la forme d'un système d'équations entre des fonctions booléennes.
- Les résultats établis au chapitre 2 nous permettent de déterminer analytiquement quelles sont les solutions d'un système d'équations donné entre des fonctions booléennes.

Nous disposons donc d'un cadre mathématique homogène permettant à la fois d'exprimer notre besoin et de déterminer quelles sont les solutions de notre problème.

Pour pouvoir utiliser de manière automatique les résultats théoriques donnés au chapitre précédent, un travail préalable de formalisation est nécessaire afin de constituer le système d'équations entre des fonctions booléennes à résoudre. C'est seulement à l'issue de ce travail de formalisation des informations données dans le cahier des charges que pourra se dérouler automatiquement le processus de résolution.

Dans la pratique, la méthode de synthèse de la loi de commande d'un système logique par résolution d'un système d'équations que nous proposons comporte quatre temps forts (figure 21) :

- la formalisation du problème, c'est-à-dire l'expression des fragments de la spécification sous forme de relations entre des fonctions booléennes,
- la vérification de la cohérence des fragments de spécification,
- la recherche des solutions possibles,
- le choix d'une solution.

Parmi ces quatre étapes, deux (« Vérification de la cohérence » et « Recherche de solutions ») peuvent être réalisées automatiquement sans l'aide du concepteur puisqu'elles reposent exclusive-

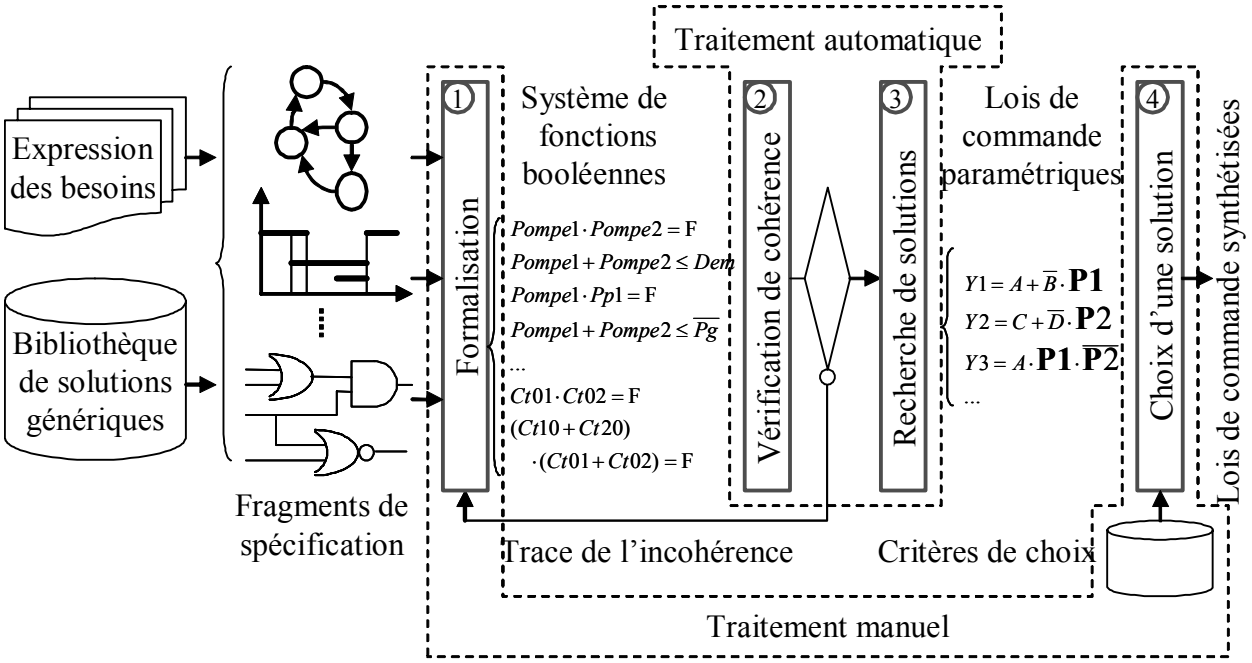


Figure 21 : Notre méthode de synthèse

ment sur les résultats établis dans le théorème 10. Pour les deux autres étapes de notre démarche, le concepteur est sollicité car le travail à réaliser peut nécessiter des prises de décision de sa part.

Nous allons maintenant détailler les différentes étapes de notre méthode et plus particulièrement la phase de formalisation.

1.1. Formalisation des fragments de spécification

Cette étape est la première de notre méthode. Elle consiste à exprimer chaque fragment de la spécification sous forme d'une ou de plusieurs relations entre des fonctions booléennes. Cette étape est à la charge du concepteur mais nous souhaitons cependant lui apporter à terme le maximum d'assistance en lui permettant d'utiliser des bibliothèques de spécifications génériques ou lui traduire automatiquement sous forme de relations entre des fonctions booléennes les spécifications qu'il propose dans un autre formalisme. À l'issue de cette étape, l'ensemble des fragments de la spécification contenus dans le cahier des charges comme les éléments de solutions qu'a souhaité donner le concepteur forme un système d'équations entre des fonctions booléennes d'une dimension *d* donnée.

Le choix de cette dimension est important car celle-ci influe directement sur la taille de l'espace solution. À ce jour, nous ne sommes pas capables de fixer automatiquement cette dimension car

elle est à la fois dépendante du nombre d'entrées du système de commande, mais également des informations données dans le cahier des charges. C'est seulement en connaissant ces informations qu'il est possible de lister quelles sont les variables logiques internes nécessaires à la résolution.

1.1.1. Définition des variables du problème

Pour un système logique à m entrées, n sorties et supposé faire référence à l variables logiques internes, nous avons montré dans la partie 3.1. du chapitre 1 que son comportement pouvait être entièrement décrit à l'aide de fonctions booléennes de dimension $(m + l)$ faisant référence aux variables booléennes $(e_1[k],..., e_m[k], x_1[k\text{-}1],..., x_l[k\text{-}1])$.

Pour ce système logique, nous associerons :

- À chaque entrée e_i $(1 \le i \le m)$, la fonction booléenne E_i considérée comme connue et définie de la manière suivante :

$$E_i : \mathbb{B}^{m+l} \to \mathbb{B} \quad \text{avec } \mathbb{B} = \{0, 1\}$$
$$(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \mapsto e_i[k]$$

Par construction, cette fonction booléenne E_i décrit le comportement de l'entrée e_i puisque son image est toujours égale à la valeur $e_i[k]$ de l'entrée e_i à l'instant k.

- À chaque sortie s_i $(1 \le i \le n)$, la fonction booléenne S_i qui permettra de définir la valeur $s_i[k]$ de la sortie s_i à l'instant k.

$$S_i : \mathbb{B}^{m+l} \to \mathbb{B} \quad \text{avec } \mathbb{B} = \{0, 1\}$$
$$(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \mapsto s_i[k]$$

- À chaque variable interne x_i, 2 fonctions booléennes X_i et Xp_i permettant de caractériser la valeur de cette variable interne x_i aux instants k et $(k\text{-}1)$.
  - La fonction booléenne X_i permettra de définir la valeur $x_i[k]$ de la variable interne x_i à l'instant k.

$$X_i[k] : \mathbb{B}^{m+l} \to \mathbb{B} \quad \text{avec } \mathbb{B} = \{0, 1\}$$
$$(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \mapsto x_i[k]$$

  - La fonction booléenne Xp_i considérée comme connue et définie par :

$$Xp_i[k\text{-}1] : \mathbb{B}^{m+l} \to \mathbb{B} \quad \text{avec } \mathbb{B} = \{0, 1\}$$
$$(e_1[k], \ldots, e_m[k], x_1[k\text{-}1], \ldots, x_l[k\text{-}1]) \mapsto x_i[k\text{-}1]$$

Par construction, cette fonction booléenne Xp_i représente le comportement de la variable interne x_i à l'instant $(k\text{-}1)$ puisqu'elle a comme image la variable booléenne $x_i[k\text{-}1]$.

Les $(m + l)$ fonctions booléennes E_i et Xp_i ont un rôle prépondérant dans cette algèbre de Boole car toutes les autres fonctions booléennes peuvent toujours être décrites comme une composition de E_i et Xp_i.

Les variables du problème étant définies, il est maintenant possible d'exprimer chaque fragment de spécification. Cette étape consiste à exprimer chaque information donnée dans le cahier des charges sous la forme de relations entre des compositions des fonctions booléennes E_i, S_i, X_i et Xp_i. À l'issue de cette phase, toutes ces relations seront regroupées et formeront le système d'équations à résoudre.

Exprimer chaque information donnée dans le cahier des charges sous la forme de relations entre des compositions des fonctions booléennes E_i, S_i, X_i et Xp_i est une tâche délicate qui demande une réelle expertise.

Dans le cadre de ce mémoire, nous donnerons quelques exemples de formalisation à partir de formalismes d'entrée différents. Nous aborderons :

- la formalisation de propositions données en langage naturel,
- la représentation de propositions données à l'aide d'un modèle logique (logigramme et Ladder Diagram),
- la représentation du comportement d'un modèle à état (Grafcet, par exemple).

1.1.2. Formalisation de propositions données en langage naturel

Cette activité est certainement la plus délicate à réaliser car le langage naturel est par nature informel. Il existe cependant des phrases type qu'il est très facile de formaliser à l'aide des relations « égalité » et « inclusion ». Le tableau 7 en présente quelques exemples. Nous nous appuyons prin-

cipalement sur la relation inclusion qui permet d'exprimer simplement une condition nécessaire comme une condition suffisante.

Type	Structure de phrase type en langage naturel	Formalisation
Condition suffisante	Si A est vrai alors B est également vrai.	$A \leq B$
	Lorsque A est présent, B est présent.	$A \leq B$
	Il suffit d'avoir A pour avoir B.	$A \leq B$
Condition nécessaire	Il est nécessaire d'avoir A pour avoir B.	$B \leq A$
	A est obligatoire pour avoir B.	$B \leq A$
	A autorise d'avoir B.	$B \leq A$
Situation interdite	Il est interdit d'avoir A et B simultanément.	$A \cdot B = \mathcal{F}$
	A ne peut pas être actif en même temps que B.	$A \cdot B = \mathcal{F}$
Invariant	Il faut toujours avoir A ou B.	$A + B = \mathcal{T}$

Tableau 7 : Formalisation de phrases en langage naturel

Il convient de souligner que lorsque 2 fonctions booléennes A et B satisfont $A \leq B$, nous avons :

$$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n, \text{ si } A(b_1, \ldots, b_n) = 1 \text{ alors } B(b_1, \ldots, b_n) = 1$$

1.1.3. Représentation de propositions données à l'aide d'un modèle logique

Dans notre cas, représenter une information donnée sous la forme d'un modèle logique à l'aide de relations entre des compositions des fonctions booléennes ne présente pas de difficultés car il ne s'agit le plus souvent que d'une simple transcription. Le tableau 8 regroupe trois exemples.

1.1.4. Représentation d'un modèle à états

La représentation du comportement d'un modèle à états consiste à décrire sous forme de relations entre des fonctions booléennes toutes les caractéristiques de sa fonction de transition et de sa fonction de sortie.

Nous allons exploiter le fait que le comportement d'un modèle à états comme le Grafcet peut être décrit sous forme algébrique à l'aide d'un jeu d'équations récurrentes. Dans [MAC06], les

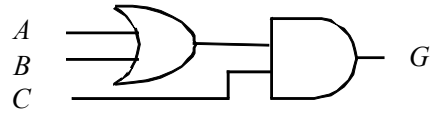
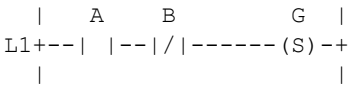
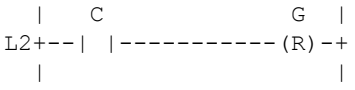
Structure du modèle logique	Formalisation
	$G = C \cdot (A + B)$
	$(A \cdot \overline{B}) \leq G$
	$C \cdot G = \mathcal{F}$

Tableau 8 : Représentation de propositions données à l'aide d'un modèle logique

auteurs rappellent comment ces équations peuvent être obtenues de manière systématique quelles que soient les structures utilisées.

Considérons par exemple le modèle Grafcet de la figure 22. À ce modèle est généralement associé le jeu d'équations suivant permettant de calculer à chaque instant k la valeur courante des états des étapes 10 et 11 ainsi que la valeur courante de la sortie « *produire* ».

$$\begin{cases} x10[k] = (x11[k\text{-}1] \wedge arrêt[k]) \vee (x10[k\text{-}1] \wedge (\overline{marche[k]} \vee arrêt[k])) \\ x11[k] = (x10[k\text{-}1] \wedge (marche[k] \wedge \overline{arrêt[k]})) \vee (x11[k\text{-}1] \wedge \overline{arrêt[k]}) \\ produire[k] = x11[k] \end{cases}$$

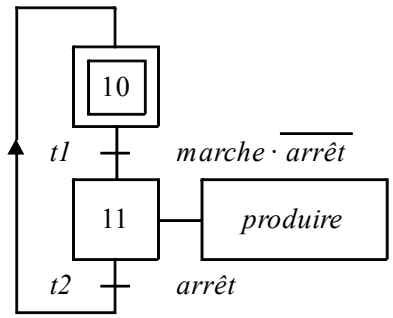


Figure 22 : Exemple de modèle Grafcet

Dans notre cas, nous considérons que la donnée de ce modèle à états consiste à définir que les 7 fonctions booléennes ($Marche$, $Arrêt$, $Produire$, $X10$, $X10p$, $X11$ et $X11p$) ne peuvent pas être indépendantes les unes des autres mais doivent respecter les exigences suivantes :

$$\begin{cases} X10 = (X11p \cdot Arrêt) + (X10p \cdot (\overline{Marche} + Arrêt)) \\ X11 = (X10p \cdot (Marche \cdot \overline{Arrêt})) + (X11p \cdot \overline{Arrêt}) \\ Produire = X11 \end{cases}$$

Cette formalisation sous forme de relations entre des fonctions booléennes permet d'intégrer des informations données à l'aide de modèles à états incomplets comme ceux où seule la structure du modèle à états est définie. Reprenons l'exemple précédent pour lequel les réceptivités associées aux transitions $t1$ et $t2$ ne sont pas connues. Soient $R1$ et $R2$ ces deux fonctions booléennes inconnues. La donnée de ce modèle partiel donne lieu aux relations suivantes :

$$\begin{cases} X10 = (X11p \cdot R2) + (X10p \cdot \overline{R1}) \\ X11 = (X10p \cdot R1) + (X11p \cdot \overline{R2}) \\ Produire = X11 \end{cases}$$

En complétant ces premières relations avec des relations complémentaires comme $Produire \leq \overline{Arrêt}$, il est possible d'obtenir automatiquement $R1$ et $R2$. Nous reviendrons en détail sur cette caractéristique lors du traitement de l'exemple 2.

1.1.5. Constitution du système d'équations et choix des inconnues

Cette étape a pour but d'établir le système d'équations à résoudre à partir des différentes relations obtenues lors de la formalisation des fragments du cahier des charges.

Les différentes fonctions booléennes sont à répartir en trois catégories :

- Les fonctions booléennes que nous considérons comme **connues** : il s'agit des fonctions booléennes E_i et Xp_i.
- Les fonctions booléennes que nous considérons comme **inconnues** : selon les cas, il s'agit des fonctions booléennes associées aux sorties et/ou associées aux variables internes ou des conditions d'évolution.
- Les autres fonctions booléennes, que nous considérons comme de simples **déclarations** : ces fonctions booléennes ont été introduites pour faciliter l'expression des fragments du cahier des

charges. Elles doivent correspondre à une composition des fonctions booléennes des deux autres catégories.

Les différentes relations booléennes sont également à répartir en trois catégories en fonction du rôle qu'elles ont dans le problème à résoudre :

- Les relations faisant uniquement référence à des fonctions booléennes que nous considérons comme connues. Ces relations seront utilisées comme **hypothèses** lors de la résolution du système d'équations.
- Les relations définissant les fonctions booléennes de type déclaration. Ces relations seront utilisées pour remplacer chaque fonction booléenne de type déclaration par la composition des fonctions booléennes des deux autres catégories qui la décrit.
- Les autres relations. Seules ces relations sont utilisées pour constituer le système d'équations à résoudre.

L'obtention du système d'équations à résoudre à partir des différentes relations obtenues lors de la formalisation des fragments du cahier des charges s'effectue en trois temps :

- Partitionnement des différentes relations suivant leur rôle,
- Remplacement de chaque fonction booléenne de type déclaration par la composition des fonctions booléennes des deux autres catégories qui la décrit,
- Constitution du système d'équations par réunion des relations de la troisième catégorie.

Cette opération sera illustrée dans l'exemple 2.

1.2. Vérification de la cohérence des fragments de la spécification

La vérification de la cohérence des fragments de la spécification est, dans notre méthode, réalisée automatiquement. Cette opération consiste à vérifier si le système d'équations proposé admet au moins une solution. Pour ce faire, le système d'équations est tout d'abord reformulé à l'aide d'une unique équation (théorème 7). Une fois sous cette forme, la condition d'existence de solution (première partie du théorème 10) est calculée à partir des coefficients de la forme canonique.

Cette opération sera réalisée manuellement lors du traitement de l'exemple 1. L'algorithme proposé figure 23 permet de la réaliser automatiquement.

Analyse_cohérence(système_équations, liste_inconnues)

(expression du système d'équations sous la forme d'une équation unique *)*
équation_unique := $\mathcal{F}$ *(* initialisation de l'équation unique *)*
pour équation **dans** système_équations **faire** *(* traitement de chaque équation *)*
  si équation est $A \leq B$ **alors** *(* détection d'une inclusion *)*
    équation_unique := équation_unique + $A \cdot \overline{B}$ *(* ajout de l'information à l'équation unique *)*
  sinon si équation est $A = B$ **alors** *(* détection d'une égalité *)*
    équation_unique := équation_unique + $A \cdot \overline{B} + \overline{A} \cdot B$ *(* ajout de l'information à l'équation unique *)*
  fin si
fin pour
(Détermination de l'existence de solutions *)*
cond_existence := $\mathcal{T}$ *(* initialisation de la condition d'existence de solution *)*
nombre_inconnues := longueur(liste_inconnues)
(Création de l'ensemble* Ω_n **)*
Ens_Oméga_n := Création_combinaisons(nombre_inconnues, $\{\mathcal{F}, \mathcal{T}\}$)
pour oméga_j **dans** Ens_Oméga_n **faire** *(* traitement de chaque équation *)*
  (Calcul du terme associé à* ω_j **)*
  C_oméga_j := Calcul_symbolique(équation_unique, liste_inconnues, oméga_j)
  (Construction progressive de la condition d'existence de solutions *)*
  cond_existence := cond_existence · C_oméga_j
fin pour
si cond_existence = $\mathcal{F}$ **alors** *(* détection du cas où il y a des solutions *)*
  Afficher(« Le problème admet au moins une solution »)
sinon *(* détection du cas où il n'y a pas de solution *)*
  Afficher(« Il n'y a pas de solution car il y a une incohérence lorsque » + cond_existence)
fin si

Figure 23 : Algorithme vérifiant la cohérence d'un système d'équations

1.3. Recherche des solutions

La recherche de solutions s'effectue en appliquant les résultats décrits dans le théorème 10. Elle n'est effectuée que dans le cas où la vérification de la cohérence a permis de conclure que le problème admet au moins une solution. La forme paramétrée obtenue permet de décrire tout l'espace solution du problème.

Cette opération sera réalisée manuellement lors du traitement de l'exemple 1. L'algorithme proposé figure 24 permet de la réaliser automatiquement.

Calcul_solutions(équation, liste_inconnues)

```
nombre_inconnues := longueur(liste_inconnues)
(* Calcul pour chaque inconnue d'une expression représentant l'espace solution *)
pour i = 1 à nombre_inconnues faire (* traitement de chaque inconnue *)
   (* nom de l'inconnue en cours *)
   inconnue_courante := prendre_élément_liste(liste_inconnues, 1)
   (* forme des solutions : Yi= Ai + /Bi · Pi *)
   Ai := 𝒯 , Bi := 𝒯  (* initialisation des deux expressions *)
   P_i := nouveau_paramètre() (* initialisation du paramètre *)
   (* Création de l'ensemble Ω_{n+1-i} *)
   Ens_Oméga_n+1-i := Création_combinaisons(nombre_inconnues+1-i, {ℱ,𝒯})
   pour omega_j dans Ens_Oméga_n+1-i faire (* traitement de chaque combinaison *)
      oméga_j^1 := prendre_élément_liste(oméga_j, 1) (* récupération de ω_j^1 *)
      (* Calcul du terme associé à ω_j *)
      C_oméga_j := Calcul_symbolique(équation, liste_inconnues, omega_j)
      si oméga_j^1 = ℱ alors (* détection que ω_j entre dans le calcul de Ai *)
         Ai := Ai · C_oméga_j (* Construction de Ai *)
      sinon (* détection que ω_j entre dans le calcul de Bi *)
         Bi := Bi · C_oméga_j (* Construction de Bi *)
      fin si
   fin pour
   Y_i := Ai + / Bi · P_i (* Calcul de Yi *)
   Afficher(inconnue_courante, « = »,Y_i)
   (* Remplacement de l'inconnue Yi par sa valeur dans l'équation *)
   équation := Calcul_symbolique(équation, inconnue_courante, Y_i)
   (* Suppression de l'inconnue Yi de la liste des inconnue *)
   liste_inconnues := enlever_élément_liste(liste_inconnues, inconnue_courante)
fin pour
```

Figure 24 : Algorithme de calcul de l'espace solution

1.4. Choix de la solution

À de très rares exceptions près, la solution d'un système d'équations entre des fonctions booléennes n'est pas unique. Dans le cadre de ce travail de thèse, il a été décidé de laisser à la charge du concepteur le choix de la solution à retenir parmi les différentes solutions proposées. Cette opération consiste à fixer une valeur pour chacun des paramètres de la forme paramétrique obtenue précédemment.

1.5. Assistance informatique pour le déroulement de la méthode

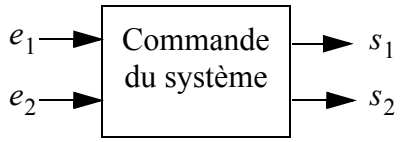
Parmi les quatre temps de notre méthode, deux d'entre eux (« Vérification de la cohérence » et « Recherche de solutions ») sont réalisés aujourd'hui automatiquement puisqu'ils ne nécessitent

aucune décision de la part du concepteur. Pour décharger le concepteur des calculs symboliques fastidieux à réaliser, un premier module de calcul symbolique fut réalisé sous Mathematica®. Malheureusement, il s'est très vite avéré que les calculs nécessaires au déroulement de notre méthode de synthèse demandaient de longues périodes de traitement informatique car Mathematica® n'est pas optimisé pour le calcul symbolique dans les algèbres de Boole. Nous utilisons aujourd'hui un outil de calcul symbolique développé au sein du laboratoire à l'aide du langage Python®. Développé spécifiquement pour la synthèse, ce module a permis de traiter efficacement les études de cas présentées dans ce chapitre.

2. Étude de cas n°1 : Synthèse d'un comportement combinatoire

Considérons le système logique combinatoire étudié au chapitre 1 dont la structure est rappelée figure 25a. Ce système comporte deux entrées et deux sorties pour lesquelles le comportement combinatoire attendu est décrit figure 25b.

a) Entrées/sorties du système



b) Cahier des charges

- P1 : Lorsque l'entrée e_1 est à 1, la sortie s_1 est à 1.
- P2 : Lorsque l'entrée e_2 est à 1, la sortie s_2 est à 0.
- P3 : Les sorties s_1 et s_2 ne sont jamais simultanément à 1.

Figure 25 : Système logique combinatoire à synthétiser

Les fonctions booléennes $E_1(e_1, e_2)$ et $E_2(e_1, e_2)$ associés aux entrées e_1 et e_2 seront notées E_1 et E_2. Elles sont considérées comme connues. Les fonctions booléennes $S_1(e_1, e_2)$ et $S_2(e_1, e_2)$ associées aux sorties s_1 et s_2 seront notées S_1 et S_2. Elles sont les inconnues de notre problème.

2.1. Formalisation des fragments de la spécification

Cette étape consiste à exprimer chacune des propositions relatives aux entrées e_1 et e_2 et aux sorties s_1 et s_2 sous forme de relations entre les fonctions booléennes E_1, E_2, S_1 et S_2.

- Formalisation de la proposition P1 : Lorsque l'entrée e_1 est à 1, la sortie s_1 est à 1.

La formalisation de cette proposition qui contraint la valeur de la sortie s_1 en fonction de la valeur de l'entrée e_1 conduit à la relation entre les fonctions booléennes E_1 et S_1 suivante :

$$E_1 \leq S_1$$

- Formalisation de la proposition P2 : Lorsque l'entrée e_2 est à 1, la sortie s_2 est à 0.

La formalisation de cette proposition qui contraint la valeur de la sortie s_2 en fonction de la valeur de l'entrée e_2 conduit à la relation entre les fonctions booléennes E_2 et S_2 suivante :

$$E_2 \leq \overline{S_2}$$

- Formalisation de la proposition P3 : Les sorties s_1 et s_2 ne sont jamais simultanément à 1.

La formalisation de cette proposition qui contraint la valeur des sorties s_1 et s_2 conduit à la relation entre les fonctions booléennes S_1 et S_2 suivante :

$$S_1 \cdot S_2 = \mathcal{F}$$

- Formalisation de l'ensemble des propositions

Une fois chacune des propositions formalisées, il est possible de les grouper sous la forme d'un système de relations :

$$\begin{cases} E_1 \leq S_1 \\ E_2 \leq \overline{S_2} \\ S_1 \cdot S_2 = \mathcal{F} \end{cases}$$

Ce système de relations peut être reformulé sous la forme d'une unique relation :

$$\begin{cases} E_1 \leq S_1 \\ E_2 \leq \overline{S_2} \\ S_1 \cdot S_2 = \mathcal{F} \end{cases} \Leftrightarrow \begin{cases} E_1 \cdot \overline{S_1} = \mathcal{F} \\ E_2 \cdot S_2 = \mathcal{F} \\ S_1 \cdot S_2 = \mathcal{F} \end{cases}$$

$$\Leftrightarrow E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 = \mathcal{F}$$

C'est cette équation qu'il nous faut maintenant résoudre.

2.2. Vérification de la cohérence des fragments de la spécification

L'équation $E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 = \mathcal{F}$ est issue de la formalisation du cahier des charges. Dans cette équation, les termes S_1 et S_2 représentent les inconnues du problème que nous cherchons à exprimer en fonction des termes E_1 et E_2.

Notons $C_0(S_1, S_2, E_1, E_2)$ cette composition des termes E_1, E_2, S_1 et S_2. Nous avons :

$$C_0(S_1, S_2, E_1, E_2) = E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 = \mathcal{F}$$

Par application de la décomposition de Shannon généralisée (théorème 2) par rapport aux termes S_1 et S_2, l'équation à résoudre peut être reformulée sous la forme canonique suivante :

$$\begin{aligned} &C_0(\mathcal{F}, \mathcal{F}, E_1, E_2) \cdot \overline{S_1} \cdot \overline{S_2} + C_0(\mathcal{F}, \mathcal{T}, E_1, E_2) \cdot \overline{S_1} \cdot S_2 \\ &+ C_0(\mathcal{T}, \mathcal{F}, E_1, E_2) \cdot S_1 \cdot \overline{S_2} + C_0(\mathcal{T}, \mathcal{T}, E_1, E_2) \cdot S_1 \cdot S_2 = \mathcal{F} \end{aligned}$$

Nous avons :

$$\begin{aligned} C_0(S_1, S_2, E_1, E_2) &= E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 \\ &= E_1 \cdot \overline{S_1} \cdot (\overline{S_2} + S_2) + E_2 \cdot (\overline{S_1} + S_1) \cdot S_2 + S_1 \cdot S_2 \\ &= E_1 \cdot \overline{S_1} \cdot \overline{S_2} + (E_1 + E_2) \cdot \overline{S_1} \cdot S_2 + S_1 \cdot S_2 \end{aligned}$$

Par association, nous avons :

$$\begin{cases} C_0(\mathcal{F}, \mathcal{F}, E_1, E_2) = E_1 \\ C_0(\mathcal{F}, \mathcal{T}, E_1, E_2) = E_1 + E_2 \\ C_0(\mathcal{T}, \mathcal{F}, E_1, E_2) = \mathcal{F} \\ C_0(\mathcal{T}, \mathcal{T}, E_1, E_2) = \mathcal{T} \end{cases}$$

Les quatre termes $C_0(\mathcal{F}, \mathcal{F}, E_1, E_2)$, $C_0(\mathcal{F}, \mathcal{T}, E_1, E_2)$, $C_0(\mathcal{T}, \mathcal{F}, E_1, E_2)$ et $C_0(\mathcal{T}, \mathcal{T}, E_1, E_2)$ sont les différents coefficients de notre équation à 2 inconnues. Rappelons que ces termes peuvent être obtenus directement à partir de $C_0(S_1, S_2, E_1, E_2)$ en faisant parcourir à $\omega_j = (\omega_j^1, \omega_j^2)$ l'ensemble $\Omega_2 = \{(\mathcal{F}, \mathcal{F}), (\mathcal{F}, \mathcal{T}), (\mathcal{T}, \mathcal{F}), (\mathcal{T}, \mathcal{T})\}$ et en procédant à la substitution suivante :

$$C_0(\omega_j^1, \omega_j^2, E_1, E_2) = (C_0(S_1, S_2, E_1, E_2)\big|_{S_1 \leftarrow \omega_j^1})\Big|_{S_2 \leftarrow \omega_j^2}$$

où $\Omega_2 = \{(\mathcal{F}, \mathcal{F}), (\mathcal{F}, \mathcal{T}), (\mathcal{T}, \mathcal{F}), (\mathcal{T}, \mathcal{T})\}$ est l'ensemble des couples ω_j utilisés pour décrire la forme canonique d'une équation à 2 inconnues.

En nous appuyant sur le théorème 10, nous savons que l'équation $C_0(S_1, S_2, E_1, E_2) = \mathcal{F}$ admet des solutions si et seulement si :

$$\prod_{\omega_i \in \Omega_2} C_0(\omega_j^1, \omega_j^2, E_1, E_2) = \mathcal{F} \text{ avec } \Omega_2 = \{(\mathcal{F}, \mathcal{F}), (\mathcal{F}, \mathcal{T}), (\mathcal{T}, \mathcal{F}), (\mathcal{T}, \mathcal{T})\}$$

Dans notre cas, nous avons :

$$\begin{aligned}
&\prod_{\omega_i \in \Omega_2} C_0(\omega_j^1, \omega_j^2, E_1, E_2) \\
&= C_0(\mathcal{F}, \mathcal{F}, E_1, E_2) \cdot C_0(\mathcal{F}, \mathcal{T}, E_1, E_2) \cdot C_0(\mathcal{T}, \mathcal{F}, E_1, E_2) \cdot C_0(\mathcal{T}, \mathcal{T}, E_1, E_2) \\
&= E_1 \cdot (E_1 + E_2) \cdot \mathcal{F} \cdot \mathcal{T} \\
&= \mathcal{F}
\end{aligned}$$

Cette condition est donc bien respectée. Le cahier des charges proposé ne comportait pas de spécifications contradictoires, donc incohérentes. Il existe au moins une solution pour S_1 et S_2.

2.3. Résolution détaillée

L'équation $C_0(S_1, S_2, E_1, E_2) = \mathcal{F}$ admet donc un ou plusieurs couples solution $(S(S_1), S(S_2))$ dont la forme paramétrique de chaque composante proposée par le théorème 10 est :

$$\begin{cases}
S(S_1) = \displaystyle\prod_{\substack{\omega_j \in \Omega_2 \\ \omega_j^1 = \mathcal{F}}} C_0(\omega_j^1, \omega_j^2, E_1, E_2) + \overline{\displaystyle\prod_{\substack{\omega_j \in \Omega_2 \\ \omega_j^1 = \mathcal{T}}} C_0(\omega_j^1, \omega_j^2, E_1, E_2)} \cdot P_1 \\
S(S_2) = \displaystyle\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = \mathcal{F}}} C_1(\omega_j^1, E_1, E_2) + \overline{\displaystyle\prod_{\substack{\omega_j \in \Omega_1 \\ \omega_j^1 = \mathcal{T}}} C_1(\omega_j^1, E_1, E_2)} \cdot P_2 \\
C_1(S_2, E_1, E_2) = C_0(S_1, S_2, E_1, E_2)\big|_{S_1 \leftarrow S(S_1)}
\end{cases}$$

où P_1 et P_2 sont des paramètres.

Par construction, P_1 et P_2 peuvent être chacune des fonctions booléennes de dimension 2.

Dans notre cas, nous avons :

- Pour $S(S_1)$:

$$
\begin{aligned}
&S(S_1)\\
&= \prod_{\substack{\omega_j \in \Omega_2 \\ \omega_i^1 = \mathcal{F}}} C_0(\omega_j^1, \omega_j^2, E_1, E_2) + \prod_{\substack{\omega_j \in \Omega_2 \\ \omega_i^1 = \mathcal{T}}} C_0(\omega_j^1, \omega_j^2, E_1, E_2) \cdot P_1\\
&= (C_0(\mathcal{F}, \mathcal{F}, E_1, E_2) \cdot C_0(\mathcal{F}, \mathcal{T}, E_1, E_2)) + \overline{(C_0(\mathcal{T}, \mathcal{F}, E_1, E_2) \cdot C_0(\mathcal{T}, \mathcal{T}, E_1, E_2))} \cdot P_1\\
&= (E_1 \cdot (E_1 + E_2)) + \overline{(\mathcal{F} \cdot \mathcal{T})} \cdot P_1\\
&= E_1 + P_1
\end{aligned}
$$

- Calcul de $C_1(S_2, E_1, E_2)$:

$$
\begin{aligned}
&C_1(S_2, E_1, E_2)\\
&= C_0(S_1, S_2, E_1, E_2)\big|_{S_1 \leftarrow S(S_1)}\\
&= (E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2)\big|_{S_1 \leftarrow E_1 + P_1}\\
&= E_1 \cdot \overline{(E_1 + P_1)} + E_2 \cdot S_2 + (E_1 + P_1) \cdot S_2\\
&= E_1 \cdot \overline{E_1} \cdot \overline{P_1} + E_2 \cdot S_2 + (E_1 + P_1) \cdot S_2\\
&= E_2 \cdot S_2 + (E_1 + P_1) \cdot S_2\\
&= (E_1 + E_2 + P_1) \cdot S_2
\end{aligned}
$$

En appliquant la décomposition de Shannon par rapport au terme S_2, le terme $C_1(S_2, E_1, E_2)$ peut être reformulé de la façon suivante :

$$
C_1(S_2, E_1, E_2) = C_1(\mathcal{F}, E_1, E_2) \cdot \overline{S_2} + C_1(\mathcal{T}, E_1, E_2) \cdot S_2 \text{ où } \begin{cases} C_1(\mathcal{F}, E_1, E_2) = \mathcal{F} \\ C_1(\mathcal{T}, E_1, E_2) = E_1 + E_2 + P_1 \end{cases}
$$

- Pour $S(S_2)$:

$$
\begin{aligned}
&S(S_2)\\
&= \prod_{\substack{\omega_j \in \Omega_1 \\ \omega_i^1 = \mathcal{F}}} C_1(\omega_j^1, E_1, E_2) + \prod_{\substack{\omega_j \in \Omega_1 \\ \omega_i^1 = \mathcal{T}}} C_1(\omega_j^1, E_1, E_2) \cdot P_2\\
&= C_1(\mathcal{F}, E_1, E_2) + \overline{C_1(\mathcal{T}, E_1, E_2)} \cdot P_2\\
&= \mathcal{F} + \overline{(E_1 + E_2 + P_1)} \cdot P_2\\
&= \overline{E_1} \cdot \overline{E_2} \cdot \overline{P_1} \cdot P_2
\end{aligned}
$$

En conclusion, l'équation $E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 = \mathcal{F}$ dans laquelle S_1 et S_2 sont inconnues admet un ou plusieurs couples solution $(S(S_1), S(S_2))$ que la forme paramétrique suivante permet de décrire :

$$\begin{cases} S(S_1) = E_1 + P_1 \\ S(S_2) = \overline{E_1} \cdot \overline{E_2} \cdot \overline{P_1} \cdot P_2 \end{cases}$$

où P_1 et P_2 sont des fonctions booléennes de dimension 2 quelconques permettant de couvrir l'espace solution.

2.4. Inventaire des solutions

Pour cette étude de cas, la forme paramétrique proposée pour décrire l'ensemble des solutions montre que le problème possède bien plusieurs solutions. En faisant parcourir à P_1 et à P_2 l'ensemble Γ_2 des fonctions booléennes de dimension 2, il est possible de lister toutes ces solutions (certaines de ces solutions peuvent être décrites par plusieurs valeurs différentes des paramètres). Dans notre cas, il n'existe que les 6 couples de solutions suivants :

- $\begin{cases} S(S_1) = E_1 \\ S(S_2) = \mathcal{F} \end{cases}$ (obtenu pour $P_1 = \mathcal{F}$ et $P_2 = \mathcal{F}$, par exemple)
- $\begin{cases} S(S_1) = E_1 \\ S(S_2) = \overline{E_1} \cdot \overline{E_2} \end{cases}$ (obtenu pour $P_1 = \mathcal{F}$ et $P_2 = \overline{E_1} \cdot \overline{E_2}$, par exemple)
- $\begin{cases} S(S_1) = E_1 + \overline{E_2} \\ S(S_2) = \mathcal{F} \end{cases}$ (obtenu pour $P_1 = \overline{E_1} \cdot \overline{E_2}$ quelle que soit la valeur de P_2)
- $\begin{cases} S(S_1) = E_1 + E_2 \\ S(S_2) = \mathcal{F} \end{cases}$ (obtenu pour $P_1 = \overline{E_1} \cdot E_2$ et $P_2 = \mathcal{F}$, par exemple)
- $\begin{cases} S(S_1) = E_1 + E_2 \\ S(S_2) = \overline{E_1} \cdot \overline{E_2} \end{cases}$ (obtenu pour $P_1 = \overline{E_1} \cdot E_2$ et $P_2 = \overline{E_1} \cdot \overline{E_2}$, par exemple)
- $\begin{cases} S(S_1) = \mathcal{T} \\ S(S_2) = \mathcal{F} \end{cases}$ (obtenu pour $P_1 = \mathcal{T}$ quelle que soit la valeur de P_2)

Ces six couples de solutions sont ceux obtenus au chapitre 1 lors du traitement manuel exhaustif du problème. Le concepteur a donc plusieurs solutions possibles pour son problème.

Parmi ces six solutions, quatre d'entre elles conduisent à fixer à $\mathcal{F}$ la fonction S_2. Ces solutions ne sont peut-être pas celles que retiendra finalement le concepteur.

2.5. Choix d'une solution

À partir de la forme paramétrique proposée, il est possible d'obtenir facilement des solutions particulières comme celles qui :

- Maximisent S_1 (au sens d'augmenter le nombre de combinaisons d'entrées pour lesquelles la valeur de S_1 est à 1). Il suffit de fixer P_1 à sa valeur maximale : $\mathcal{T}$.
- Minimisent S_1 : Il suffit de fixer P_1 à sa valeur minimale : $\mathcal{F}$.

Pour obtenir facilement les solutions particulières qui maximisent ou minimisent S_2, il suffit de privilégier S_2 pour l'ordre de résolution.

3. Étude de cas n°2 : Synthèse des conditions d'évolution dans un modèle à états

Cette deuxième étude de cas est relative à la synthèse d'un système séquentiel et plus particulièrement à la synthèse des conditions d'évolution entre des états définis par le concepteur. Nous montrerons à travers cette étude comment un concepteur peut imposer des éléments de solutions *a priori*.

3.1. Présentation du support de l'étude

L'objet de cette étude est le système de contrôle/commande d'un système de distribution d'eau fonctionnant avec deux pompes en redondance (figure 26).

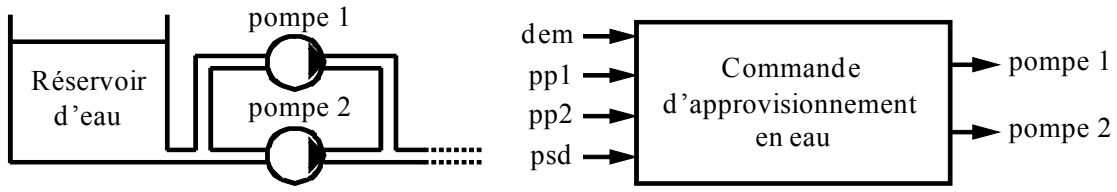


Figure 26 : Système de distribution d'eau

Le contrôleur logique qui commande ce système de distribution comporte 4 entrées (dem, pp1, pp2 et psd) et 2 sorties (pompe1, pompe2) :

- **dem :** Demande d'eau par les consommateurs.
- **pp1 :** Panne détectée sur la pompe 1.
- **pp2 :** Panne détectée sur la pompe 2.
- **psd :** Panne détectée sur le système de distribution.
- **pompe1 :** Entraîner la pompe 1.
- **pompe2 :** Entraîner la pompe 2.

Les principales attentes pour le système de distribution sont :

- Les deux pompes ne sont jamais utilisées en même temps.
- Une demande d'approvisionnement est nécessaire pour que l'une des pompes fonctionne.
- La pompe 1 ne peut pas être utilisée lorsqu'elle est en panne.
- La pompe 2 ne peut pas être utilisée lorsqu'elle est en panne.
- Aucune des pompes ne peut être utilisée lorsqu'une panne sur le système de distribution est détectée.
- En l'absence de pannes globales, une demande d'approvisionnement est suffisante pour qu'une des pompes fonctionne.
- Les permutations de pompes ne peuvent se faire que si l'une d'elles est en panne.

Un expert reconnaît dans ces informations un problème de contrôle de redondance. Il connaît plusieurs structures paramétriques qui sont des solutions de ce problème. L'une d'entre elles est donnée figure 27.

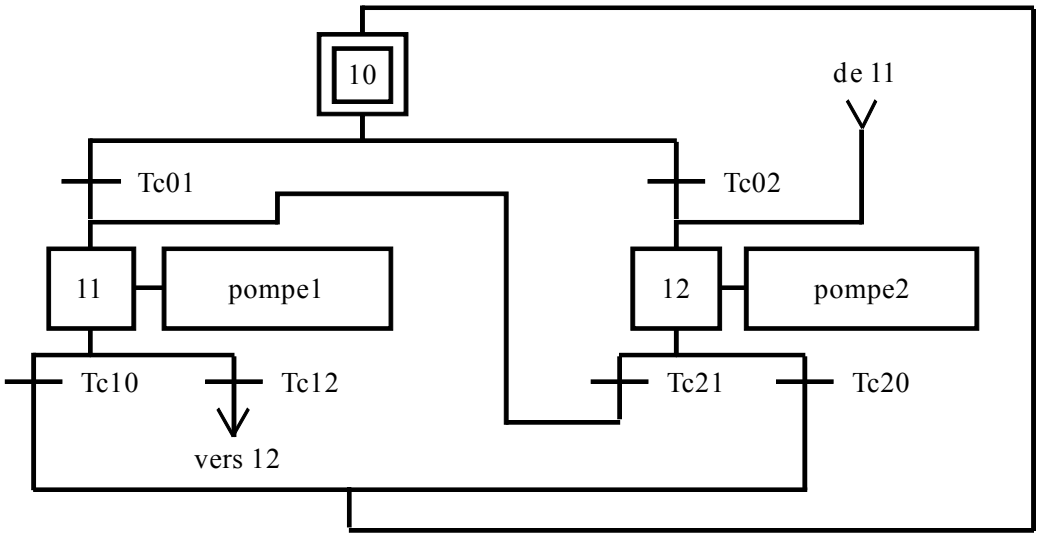


Figure 27 : Modèle de commande du système de distribution d'eau

La principale difficulté à laquelle est confronté le concepteur est de définir les conditions d'évolution entre états, c'est-à-dire les conditions pour le démarrage d'une seule pompe ($Tc01$ et $Tc02$), les conditions de passage d'une pompe à l'autre ($Tc12$ et $Tc21$) et les conditions pour arrêter l'installation ($Tc10$ et $Tc20$).

Sur le plan mathématique, les $Tcij$ sont des fonctions booléennes des entrées et des variables d'étapes du Grafcet. Ce sont ces $Tcij$ que nous proposons de déterminer en résolvant le système d'équations représentant le cahier des charges et les éléments de solution donnés par le concepteur.

3.2. Formalisation des fragments de spécification

3.2.1. Fonctions booléennes utilisées

La commande proposée pour le système de distribution d'eau est un système logique à 4 entrées, 2 sorties et faisant référence à 3 variables internes. Son comportement séquentiel peut donc être entièrement décrit à l'aide de fonctions booléennes de dimension 7 faisant référence aux variables booléennes ($dem[k]$, $pp1[k]$, $pp2[k]$, $psd[k]$, $x10[k\text{-}1]$, $x11[k\text{-}1]$ et $x12[k\text{-}1]$).

Les fonctions booléennes associées aux entrées dem, pp1, pp2 et psd seront notées Dem, $Pp1$, $Pp2$ et Psd. Les fonctions booléennes associées aux sorties pompe1 et pompe2 seront notées $Pompe1$ et $Pompe2$.

Les fonctions booléennes associées aux étapes du grafcet seront notées $X10$, $X11$, $X12$, $X10p$, $X11p$ et $X12p$. Les fonctions booléennes représentant les réceptivités associées aux transitions du grafcet seront notées $Tc01$, $Tc02$, $Tc10$, $Tc12$, $Tc20$ et $Tc21$.

3.2.2. Représentation de la solution proposée par le concepteur

Le grafcet proposé figure 27 précise que les 14 fonctions booléennes $X10$, $X11$, $X12$, $X10p$, $X11p$, $X12p$, $Tc01$, $Tc02$, $Tc10$, $Tc12$, $Tc20$, $Tc21$, $Pompe1$ et $Pompe2$ ne sont pas indépendantes les unes des autres car elles doivent respecter les relations suivantes :

$$\begin{cases} (FS1) & X10 = X11p \cdot Tc10 + X12p \cdot Tc20 + X10p \cdot \overline{Tc01} \cdot \overline{Tc02} \\ (FS2) & X11 = X10p \cdot Tc01 + X12p \cdot Tc21 + X11p \cdot \overline{Tc10} \cdot \overline{Tc12} \\ (FS3) & X12 = X10p \cdot Tc02 + X11p \cdot Tc12 + X12p \cdot \overline{Tc20} \cdot \overline{Tc21} \\ (FS4) & Pompe1 = X11 \\ (FS5) & Pompe2 = X12 \end{cases}$$

De plus, comme le grafcet proposé par le concepteur est prévu pour représenter un système à trois états, il est nécessaire que les fonctions booléennes $X10$, $X11$ et $X12$ satisfassent également :

$$\begin{cases} (FS6) & X10 + X11 + X12 = \mathcal{T} \\ (FS7) & X10 \leq \overline{X11} \cdot \overline{X12} \\ (FS8) & X11 \leq \overline{X10} \cdot \overline{X12} \\ (FS9) & X12 \leq \overline{X10} \cdot \overline{X11} \end{cases}$$

La relation $FS6$ précise qu'au moins une des étapes doit être active. Les trois autres relations sont introduites pour assurer l'unicité d'activité des étapes.

Il en est de même pour les fonctions booléennes $X10p$, $X11p$ et $X12p$ (ces quatre relations seront utilisées comme hypothèses) :

$$\begin{cases} (FS10) & X10p + X11p + X12p = \mathcal{T} \\ (FS11) & X10p \leq \overline{X11p} \cdot \overline{X12p} \\ (FS12) & X11p \leq \overline{X10p} \cdot \overline{X12p} \\ (FS13) & X12p \leq \overline{X10p} \cdot \overline{X11p} \end{cases}$$

3.2.3. Formalisation des éléments du cahier des charges donnés en langage naturel

Les éléments du cahier des charges donnés en langage naturel peuvent être formalisés de la façon suivante :

- Les deux pompes ne sont jamais utilisées en même temps.

$$(S1) \qquad Pompe1 \cdot Pompe2 = \mathcal{F}$$

- Une demande d'approvisionnement est nécessaire pour que l'une des pompes fonctionne.

  $(S2)\qquad Pompe1 + Pompe2 \leq Dem$

- La pompe 1 ne peut pas être utilisée lorsqu'elle est en panne.

  $(S3)\qquad Pompe1 \cdot Pp1 = \mathcal{F}$

- La pompe 2 ne peut pas être utilisée lorsqu'elle est en panne.

  $(S4)\qquad Pompe2 \cdot Pp2 = \mathcal{F}$

- Aucune des pompes ne peut être utilisée lorsqu'une panne sur le système de distribution est détectée.

  $(S5)\qquad (Pompe1 + Pompe2) \cdot Psd = \mathcal{F}$

- En l'absence de pannes globales, une demande d'approvisionnement est suffisante pour qu'une des pompes fonctionne.

  $(S6)\qquad \overline{(Psd + Pp1 \cdot Pp2)} \cdot Dem \leq Pompe1 + Pompe2$

- Les permutations de pompes ne peuvent se faire que si l'une d'elles est en panne.

  $(S7)\qquad X11 \cdot Tc12 + X12 \cdot Tc21 \leq Pp1 + Pp2$

3.2.4. Constitution du système d'équations et choix des inconnues

La formalisation du problème a conduit à établir 20 relations entre les 18 fonctions booléennes référencées. Ces 18 fonctions booléennes se répartissent dans notre cas de la façon suivante :

- les fonctions booléennes supposées connues : Dem, $Pp1$, $Pp2$, Psd, $X10p$, $X11p$ et $X12p$.
- les fonctions booléennes inconnues : $Tc01$, $Tc02$, $Tc10$, $Tc12$, $Tc20$ et $Tc21$.
- les simples déclarations introduites pour faciliter l'écriture des spécifications. Ici, il s'agit de $X10$, $X11$, $X12$, $Pompe1$ et $Pompe2$.

Parmi les 20 relations, nous avons :

- 4 hypothèses ($FS10$, $FS11$, $FS12$ et $FS13$) puisque ces relations ne font référence qu'à des fonctions booléennes supposées connues.
- 5 relations dites de déclaration ($FS1$, $FS2$, $FS3$, $FS4$ et $FS5$). Ces relations ont été introduites pour définir les fonctions booléennes $X10$, $X11$, $X12$, $Pompe1$ et $Pompe2$ à partir de Dem, $Pp1$, $Pp2$, Psd, $X10p$, $X11p$, $X12p$, $Tc01$, $Tc02$, $Tc10$, $Tc12$, $Tc20$ et $Tc21$.

Le système d'équations à résoudre sera obtenu à partir des 11 autres relations en remplaçant les fonctions booléennes $X10$, $X11$, $X12$, $Pompe1$ et $Pompe2$ par leur expression ($X10$ par

$X11p \cdot Tc10 + X12p \cdot Tc20 + X10p \cdot \overline{Tc01} \cdot \overline{Tc02}$). Les 4 hypothèses seront utilisées pour simplifier les calculs.

3.3. Vérification de la cohérence des fragments du cahier des charges

L'équation à résoudre comportant 6 inconnues ($Tc01$ à $Tc21$), la forme canonique comporte 64 (2^6) termes. La vérification de la cohérence des fragments du cahier des charges est faite automatiquement à l'aide du module logiciel développé dans le cadre de cette thèse. Le fichier d'entrée correspondant à cette étude est proposé figure 28.

3.4. Résolution du système d'équations

La solution paramétrique calculée par notre module logiciel est la suivante :

$$\begin{cases} Tc01 = \overline{X10p} \cdot P_{Tc01} + X10p \cdot Dem \cdot \overline{Pp1} \cdot \overline{Psd} \cdot (Pp2 + P_{Tc01}) \\ Tc02 = \overline{X10p} \cdot P_{Tc02} + X10p \cdot Dem \cdot \overline{Pp2} \cdot \overline{Psd} \cdot (Pp1 + \overline{P_{Tc01}}) \\ Tc10 = \overline{X11p} \cdot P_{Tc10} + X11p \cdot (\overline{Dem} + Psd + Pp1 \cdot Pp2) \\ Tc12 = \overline{X11p} \cdot P_{Tc12} + X11p \cdot Dem \cdot Pp1 \cdot \overline{Pp2} \cdot \overline{Psd} \\ Tc20 = \overline{X12p} \cdot P_{Tc20} + X12p \cdot (\overline{Dem} + Psd + Pp1 \cdot Pp2) \\ Tc21 = \overline{X12p} \cdot P_{Tc21} + X12p \cdot Dem \cdot \overline{Pp1} \cdot Pp2 \cdot \overline{Psd} \end{cases}$$

où P_{Tc01}, P_{Tc02}, P_{Tc10}, P_{Tc12}, P_{Tc20} et P_{Tc21} sont des éléments de Γ_7 introduits pour couvrir l'ensemble des solutions du problème.

Dans cette solution paramétrique calculée, chacune des réceptivités admet plusieurs solutions puisqu'elles dépendent toutes d'un paramètre différent. Cependant, en analysant plus finement cette solution paramétrique, nous pouvons noter que ce paramètre précise seulement que la réceptivité peut être quelconque lorsque l'étape qui valide la transition à laquelle elle est associée n'est pas active. Ce résultat est bien mathématiquement cohérent.

```
<PROBLEME>
<INFO> Auteur : Yann HIETTER (* Gestion de pompes redondantes *)</INFO>

<VARIABLES>
  tc01 : Inconnue (* Condition : Demarrage de la Pompe 1 *) ;
  tc02 : Inconnue (* Condition : Demarrage de la Pompe 2 *) ;
  tc10 : Inconnue (* Condition : Arret de la Pompe 1 *) ;
  tc20 : Inconnue (* Condition : Arret de la Pompe 2 *) ;
  tc12 : Inconnue (* Condition : Permutation Pompe 1 / Pompe 2 *) ;
  tc21 : Inconnue (* Condition : Permutation Pompe 2 / Pompe 1 *) ;
  Pp1 : Donnee (* Entree : Panne de la pompe 1 *) ;
  Pp2 : Donnee (* Entree : Panne de la pompe 2 *) ;
  Psd : Donnee (* Entree : Panne sur le circuit *) ;
  Dem : Donnee (* Entree : Demande de fonctionnement *) ;
  X10p : Donnee (* Etat precedent : Pas de distribution *) ;
  X11p : Donnee (* Etat precedent : Fonctionnement de la Pompe 1 *) ;
  X12p : Donnee (* Etat precedent : Fonctionnement de la Pompe 2 *) ;
  X10  : Declaration (* Etat attendu : Pas de distribution *) ;
  X11  : Declaration (* Etat attendu : Fonctionnement de la Pompe 1 *) ;
  X12  : Declaration (* Etat attendu : Fonctionnement de la Pompe 2 *) ;
  Pompe1 : Declaration (* Sortie : Commande de la Pompe 1 *) ;
  Pompe2 : Declaration (* Sortie : Commande de la Pompe 2 *) ;
</VARIABLES>

<DECLARATIONS>
  X10 = X11p.tc10 + X12p.tc20 + X10p./(tc01+tc02) (* Equation reccurente pour X10 *) ;
  X11 = X10p.tc01 + X12p.tc21 + X11p./(tc10+tc12) (* Equation reccurente pour X11 *) ;
  X12 = X10p.tc02 + X11p.tc12 + X12p./(tc20+tc21) (* Equation reccurente pour X12 *) ;
  Pompe1 = X11 (* Equation de sortie *) ;
  Pompe2 = X12 (* Equation de sortie *) ;
</DECLARATIONS>

<ASSERTIONS>
  f1 : Pompe1 . Pompe2 = 0    (* Pas de fonctionnement simultane *) ;
  f2 : (Pompe1 + Pompe2) =< Dem  (* Pas de fonctionnement sans une demande en cours *) ;
  f3 : Pompe1 . Pp1 = 0    (* Pas de fonctionnement de la pompe 1 lorsqu'elle est en
panne *) ;
  f4 : Pompe2 . Pp2 = 0    (* Pas de fonctionnement de la pompe 2 lorsqu'elle est en
panne *) ;
  f5 : (Pompe1 + Pompe2) . Psd = 0    (* Pas de fonctionnement en cas de panne sur le
circuit de distribution *) ;
  f6 : /(Psd + Pp1.Pp2) . Dem =< (Pompe1 + Pompe2)  (* En absence de panne globale, une
demande suffit avoir distribution *) ;
  f7 : X11p.tc12 + X12p.tc21 =< Pp1 + Pp2 (* Pas de permutation inutile *) ;

  A1 : X10 + X11 + X12 = 1  (* Au moins un etat actif *) ;
  A2a : X10 =< /X11./X12 (* Au plus un etat actif *) ;
  A2b : X11 =< /X10./X12 (* Au plus un etat actif *) ;
  A2c : X12 =< /X10./X11 (* Au plus un etat actif *) ;
</ASSERTIONS>

<HYPOTHESES>
  H1 : X10p + X11p + X12p = 1  (* Au moins un etat actif *) ;
  H2a : X10p =< /X11p./X12p (* Au plus un etat actif *) ;
  H2b : X11p =< /X10p./X12p (* Au plus un etat actif *) ;
  H2c : X12p =< /X10p./X11p (* Au plus un etat actif *) ;
</HYPOTHESES>

<ORDRE> (tc01,tc02,tc12,tc21,tc10,tc20)</ORDRE>
</PROBLEME>
```

Figure 28 : Fichier d'entrée du solveur pour la deuxième étude de cas

Une fois cette apparente liberté éliminée, nous avons :

$$\begin{cases} Tc01 = Dem \cdot \overline{Pp1} \cdot \overline{Psd} \cdot (Pp2 + P_{Tc01}) \\ Tc02 = Dem \cdot \overline{Pp2} \cdot \overline{Psd} \cdot (Pp1 + \overline{P_{Tc01}}) \\ Tc10 = \overline{Dem} + Psd + Pp1 \cdot Pp2 \\ Tc12 = Dem \cdot Pp1 \cdot \overline{Pp2} \cdot \overline{Psd} \\ Tc20 = \overline{Dem} + Psd + Pp1 \cdot Pp2 \\ Tc21 = Dem \cdot \overline{Pp1} \cdot Pp2 \cdot \overline{Psd} \end{cases}$$

3.5. Choix de la solution

La dernière étape de notre approche est le choix d'une solution spécifique. Seules les deux conditions de transition $Tc01$ et $Tc02$ dépendent d'un paramètre. Elles décrivent l'ensemble des possibilités pour activer une des pompes en fonctionnement normal (quand une demande d'eau est émise en l'absence de panne). Pour fixer ce paramètre, plusieurs stratégies sont possibles :

- La première solution est d'accorder la priorité à l'une des pompes. Or l'activation de la pompe 1 dépend de $Tc01$ et donc de P_{Tc01}, alors que la mise en route de la seconde pompe dépend de $Tc02$ et donc de $\overline{P_{Tc01}}$. Pour favoriser la pompe 1 il suffit donc que la fonction P_{Tc01} soit vraie dans le maximum de situations. Pour cela, il suffit de remplacer la fonction P_{Tc01} par la fonction toujours vraie $\mathcal{T}$. De même, pour favoriser la seconde pompe, P_{Tc01} sera remplacée par la fonction toujours fausse $\mathcal{F}$.
- Un autre objectif peut être d'équilibrer le temps de travail des deux pompes. Dans ce cas, une nouvelle entrée peut être ajoutée (par exemple « c », comme clock, qui change de valeur toutes les 12 heures). Quand $c = 1$, la pompe 1 doit être utilisée en fonctionnement normal (la pompe 2 doit être employée si $c = 0$). Dans ce cas, il est suffisant de choisir $P_{Tc01} = c$ pour traduire cette politique. La solution devient :

$$\begin{cases} Tc01 = Dem \cdot \overline{Psd} \cdot \overline{Pp1} \cdot (Pp2 + c) \\ Tc02 = Dem \cdot \overline{Psd} \cdot \overline{Pp2} \cdot (Pp1 + \overline{c}) \\ Tc10 = \overline{Dem} + Psd + Pp1 \cdot Pp2 \\ Tc12 = Dem \cdot \overline{Psd} \cdot Pp1 \cdot \overline{Pp2} \\ Tc20 = \overline{Dem} + Psd + Pp1 \cdot Pp2 \\ Tc21 = Dem \cdot \overline{Psd} \cdot \overline{Pp1} \cdot Pp2 \end{cases}$$

Bien d'autres solutions peuvent être choisies en considérant des critères différents. Toutefois, cela ne rentre plus dans le cadre de cette thèse.

3.6. Conclusions relatives à l'étude de cas

Au travers du traitement de cet exemple, nous avons montré qu'il était possible de déterminer automatiquement les conditions d'évolution d'un modèle à états à partir d'une description formelle des données du cahier des charges.

Pour cet exemple, nous avons utilisé conjointement des spécifications données en langage naturel et une solution générique proposée par le concepteur.

4. Étude de cas n°3 : Synthèse de la commande d'un système de production

4.1. Description du système

Le troisième et dernier exemple traité dans ce chapitre est un système de conditionnement de liquide en flacons. La partie opérative de ce système (figure 29) est composée d'un plateau rotatif autour duquel sont situés quatre postes distincts :

- Les flacons sont chargés sur le plateau rotatif au poste 1 à l'aide d'un robot manipulateur depuis une goulotte d'alimentation en flacons vides.
- Au poste 2, les flacons sont remplis.
- Au poste 3, les flacons sont fermés à l'aide d'un bouchon.
- Au poste 4, les flacons sont déchargés du plateau rotatif vers une goulotte d'évacuation à l'aide du même robot manipulateur.

Pour assurer le bon fonctionnement de ce système de production, il est nécessaire que les six tâches suivantes soient correctement synchronisées :

- tâche 1 : Positionner un Flacon sur le plateau rotatif au poste 1 ($t1PF$),
- tâche 2 : Remplir le Flacon au poste 2 ($t2RF$),
- tâche 3a : Saisir un Bouchon au poste 3 ($t3aSB$),
- tâche 3b : Placer le Bouchon sur le Flacon au poste 3 ($t3bPBF$),

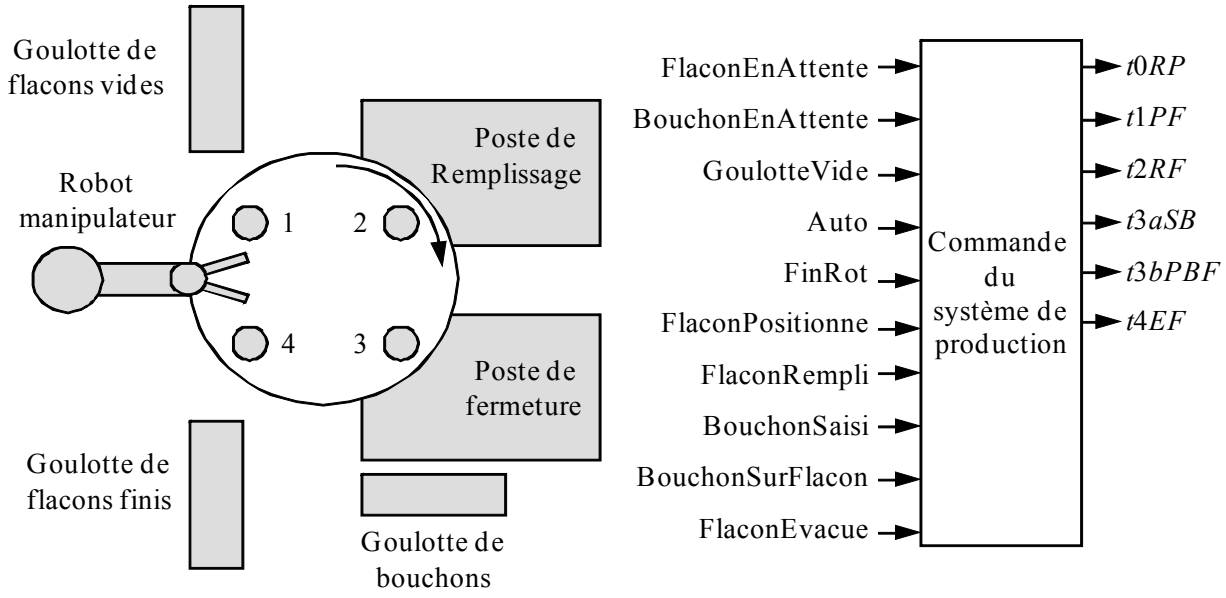


Figure 29 : Système de conditionnement de liquides en flacon

- tâche 4 : Évacuer le Flacon du plateau rotatif au poste 4 ($t4EF$),
- tâche 0 : entraîner en Rotation le Plateau rotatif d'un quart de tour ($t0RP$).

Le problème auquel est confronté le concepteur est de savoir comment ordonnancer les 6 tâches opératives du système en tenant compte des impératifs nécessaires à leur bon déroulement. Nous allons montrer comment cette coordination peut être obtenue à l'aide de notre approche en établissant les équations récurrentes de chacune des tâches à partir de leurs conditions externes de démarrage et de terminaison, des diagrammes d'enchaînement des tâches pour chacun des postes, et d'une règle d'exclusion due à l'utilisation de la ressource commune que représente le robot manipulateur.

4.2. Description du fonctionnement

4.2.1. Conditions externes de démarrage des tâches

Nous ferons l'hypothèse que le cahier des charges précise que le démarrage de certaines de ces tâches est contraint par des conditions nécessaires externes. Pour cette étude de cas, nous avons retenu les contraintes suivantes :

- Tâche $t1PF$: Pour démarrer cette tâche, il est nécessaire qu'un flacon soit en attente « $FlaconEnAttente$ »,
- Tâche $t3aSB$: Pour démarrer cette tâche, il est nécessaire qu'un bouchon soit en attente « $BouchonEnAttente$ »,

- Tâche $t4EF$: Pour démarrer cette tâche, il est nécessaire que la goulotte d'évacuation soit vide « $GoulotteVide$ »,
- Tâche $t0RP$: Pour démarrer cette tâche, il est nécessaire que le système soit en mode Automatique « $Auto$ ».

4.2.2. Conditions externes de terminaison des tâches

Pour cette étude de cas, nous considérons que les conditions nécessaires et suffisantes externes de terminaison de chacune de ces six tâches sont les suivantes :

- $FinRot$: Condition d'arrêt de la tâche $t0RP$ « Entraîner en rotation le plateau rotatif d'un quart de tour »,
- $FlaconPositionne$: Condition d'arrêt de la tâche $t1PF$ « Positionner un flacon sur le plateau rotatif au poste 1 »,
- $FlaconRempli$: Condition d'arrêt de la tâche $t2RF$ « Remplir le flacon au poste 2 »,
- $BouchonSaisi$: Condition d'arrêt de la tâche $t3aSB$ « Saisir un bouchon au poste 3 »,
- $BouchonSurFlacon$: Condition d'arrêt de la tâche $t3bPBF$ « Placer le bouchon sur le flacon au poste 3 »,
- $FlaconEvacue$: Condition d'arrêt de la tâche $t4EF$ « Évacuer le flacon du plateau rotatif au poste 4 ».

4.2.3. Enchaînement des tâches

Pour que le système de production fonctionne correctement, les tâches $t0RP$, $t1PF$, $t2RF$, $t3aSB$, $t3bPBF$, $t4EF$ doivent s'enchaîner dans un ordre précis. Pour cette étude de cas, nous avons retenu les contraintes suivantes :

- Au poste 3, les tâches $t3aSB$ et $t3bPBF$ s'enchaînent séquentiellement (figure 30).

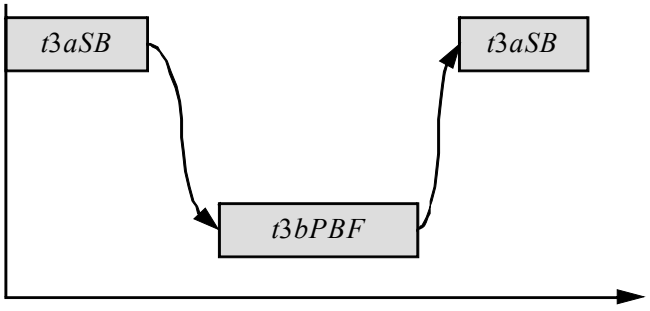


Figure 30 : Diagramme de séquence entre les tâches *t3aSB* et *t3bPBF*

- Au poste 1, les tâches $t0RP$ et $t1PF$ s'enchaînent séquentiellement.
- Au poste 2, les tâches $t0RP$ et $t2RF$ s'enchaînent séquentiellement.
- Au poste 3, les tâches $t0RP$ et $t3bPBF$ s'enchaînent séquentiellement.
- Au poste 4, les tâches $t0RP$ et $t4EF$ s'enchaînent séquentiellement.

4.2.4. Condition d'exclusion des tâches

Pour ce système, nous avons également une contrainte d'exclusion entre les tâches $t1PF$ et $t4EF$ en raison de l'utilisation de la ressource commune qui est le robot.

C'est à partir de ces données informelles que nous proposons d'obtenir les équations récurrentes du modèle à états de ce système.

4.3. Formalisation des éléments du cahier des charges

4.3.1. Représentation des tâches

À chacune des tâches $t0RP$, $t1PF$, $t2RF$, $t3aSB$, $t3bPBF$ et $t4EF$, nous associons 4 fonctions booléennes pour décrire ses principales caractéristiques. Par exemple, à la tâche $t0RP$ sont associées les fonctions booléennes :

- $T0RP$: Fonction booléenne représentant l'état courant de la tâche $t0RP$ (inconnue du problème),
- $T0RP\_p$: Fonction booléenne représentant l'état précédent de la tâche $t0RP$ (donnée du problème),
- Act_T0RP : Fonction booléenne représentant l'activation (le démarrage) de la tâche $t0RP$. Cette fonction booléenne s'obtient à partir des deux premières de la façon suivante :

$$Act\_T0RP = \overline{T0RP\_p} \cdot T0RP$$

- Des_T0RP : Fonction booléenne représentant la désactivation (l'arrêt) de la tâche $t0RP$. Cette fonction booléenne s'obtient également à partir des deux premières de la façon suivante :

$$Des\_T0RP = T0RP\_p \cdot \overline{T0RP}$$

Ces deux dernières fonctions ont été introduites pour faciliter l'expression des éléments du cahier des charges.

4.3.2. Représentation des conditions externes de terminaison des tâches

La formalisation des conditions de terminaison des tâches $t0RP$, $t1PF$, $t2RF$, $t3aSB$, $t3bPBF$ et $t4EF$ conduit à l'écriture de deux relations par tâche. Par exemple, pour la tâche $t0RP$, ces relations sont :

- Lorsque la tâche $t0RP$ est active, il suffit que la rotation soit finie ($FinRot$) pour que la tâche $t0RP$ soit désactivée :

$$T0RP\_p \cdot FinRot \leq Des\_T0RP$$

- Pour désactiver la tâche $t0RP$, il faut que la rotation soit finie ($FinRot$) :

$$Des\_T0RP \leq FinRot$$

En procédant de manière similaire, nous avons pour les cinq autres tâches :

$$\begin{cases} T1PF\_p \cdot FlaconPositionne \leq Des\_T1PF \\ Des\_T1PF \leq FlaconPositionne \\ T2RF\_p \cdot FlaconRempli \leq Des\_T2RF \\ Des\_T2RF \leq FlaconRempli \\ T3aSB\_p \cdot BouchonSaisi \leq Des\_T3aSB \\ Des\_T3aSB \leq BouchonSaisi \\ T3bPBF\_p \cdot BouchonSurFlacon \leq Des\_T3bPBF \\ Des\_T3bPBF \leq BouchonSurFlacon \\ T4EF\_p \cdot FlaconEvacue \leq Des\_T4EF \\ Des\_T4EF \leq FlaconEvacue \end{cases}$$

4.3.3. Représentation des conditions externes de démarrage des tâches

La formalisation des conditions d'activation des tâches $t0RP$, $t1PF$, $t3aSB$ et $t4EF$ conduit à l'écriture d'une relation par tâche. Par exemple, pour la tâche $t0RP$, cette relation est la suivante :

- Pour activer la tâche $t0RP$, il faut que le système soit en mode Automatique ($Auto$) :

$$Act\_T0RP \leq Auto$$

Pour les trois autres tâches, nous avons :

$$\begin{cases} Act\_T1PF \leq FlaconEnAttente \\ Act\_T3aSB \leq BouchonEnAttente \\ Act\_T4EF \leq GoulotteVide \end{cases}$$

4.3.4. Représentation de l'exclusion entre les tâches *t1PF* et *t4EF*

Les tâches $t1PF$ et $t4EF$ doivent être exclusives en raison de la ressource commune que représente le robot manipulateur. Dans notre approche, cette contrainte se représente de la façon suivante :

$$T1PF \cdot T4EF = \mathcal{F}$$

À cette relation liant les états courants des tâches $t1PF$ et $t4EF$, nous associons également une relation liant les états précédents des tâches $t1PF$ et $t4EF$ (cette relation sera utilisée en tant qu'hypothèse pour les calculs) :

$$T1PF\_p \cdot T4EF\_p = \mathcal{F}$$

4.3.5. Représentation d'un enchaînement séquentiel de tâches

Au poste 3, les tâches $t3aSB$ « Saisir un bouchon » et $t3bPBF$ « Placer le bouchon sur le flacon » doivent toujours s'enchaîner l'une après l'autre pour obtenir la valeur ajoutée pour laquelle le poste a été conçu. Le déroulement de ces deux tâches doit donc être conforme au diagramme de séquences proposé figure 30.

Sur ce diagramme de séquences, le déroulement des tâches $t3aSB$ et $t3bPBF$ est tel que :

- les tâches sont exclusives,
- l'occurrence d'un démarrage de la tâche $t3bPBF$ succède à l'occurrence d'un arrêt de la tâche $t3aSB$,
- l'occurrence d'un démarrage de la tâche $t3aSB$ succède à l'occurrence d'un arrêt de la tâche $t3bPBF$.

Sur le diagramme de séquences proposé figure 31, nous avons fait apparaître les temps d'attente entre les tâches de la séquence (entre les tâches $t3aSB$ et $t3bPBF$: $t3aSB\_t3bPBF$ et entre les tâches $t3bPBF$ et $t3aSB$: $t3bPBF\_t3aSB$). Sur ce diagramme de séquences, le déroulement des tâches $t3aSB$ et $t3bPBF$ et des temps d'attente est tel que :

- les quatre périodes sont exclusives,
- il y a toujours une période active,
- l'occurrence d'un démarrage d'une tâche est synchrone à l'arrêt d'un temps d'attente,
- l'occurrence d'un arrêt d'une tâche est synchrone au démarrage d'un temps d'attente.

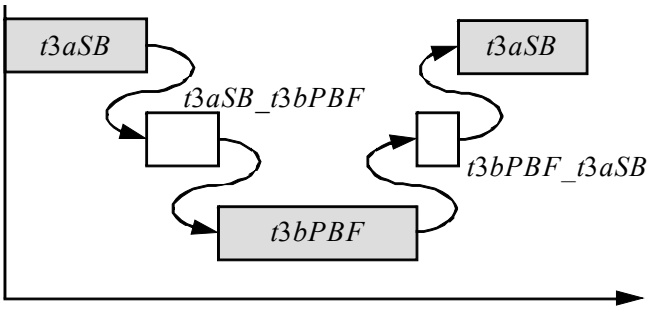


Figure 31 : Diagramme de séquence entre les tâches *t3aSB* et *t3bPBF*

Pour représenter ce comportement, nous associons à chaque temps d'attente 4 fonctions booléennes pour décrire ses principales caractéristiques. Par exemple, au temps d'attente $t3aSB\_t3bPBF$ sont associées les fonctions booléennes :

- $T3aSB\_T3bPBF$: Fonction booléenne représentant l'état courant du temps d'attente $t3aSB\_t3bPBF$ (inconnue du problème),
- $T3aSB\_T3bPBF\_p$: Fonction booléenne représentant l'état précédent du temps d'attente $t3aSB\_t3bPBF$ (donnée du problème),
- Act_T3aSB_T3bPBF : Fonction booléenne représentant l'activation (le démarrage) du temps d'attente. Cette fonction booléenne s'obtient à partir des deux premières de la façon suivante :

$$Act\_T3aSB\_T3bPBF = \overline{T3aSB\_T3bPBF\_p} \cdot T3aSB\_T3bPBF$$

- Des_T3aSB_T3bPBF : Fonction booléenne représentant la désactivation (l'arrêt) du temps d'attente. Cette fonction booléenne s'obtient également à partir des deux premières de la façon suivante :

$$Des\_T3aSB\_T3bPBF = T3aSB\_T3bPBF\_p \cdot \overline{T3aSB\_T3bPBF}$$

Les 16 fonctions booléennes ainsi définies sont liées entre elles par les 14 relations suivantes :

- Au moins un état courant parmi $t3aSB$, $t3bPBF$, $t3aSB\_t3bPBF$, $t3bPBF\_t3aSB$:

$$\begin{cases} T3aSB + T3bPBF + T3aSB\_T3bPBF + T3bPBF\_T3aSB = \mathcal{T} \\ T3aSB\_p + T3bPBF\_p + T3aSB\_T3bPBF\_p + T3bPBF\_T3aSB\_p = \mathcal{T} \end{cases}$$

- Au plus un état courant parmi $t3aSB$, $t3bPBF$, $t3aSB\_t3bPBF$, $t3bPBF\_t3aSB$:

$$\begin{cases} T3aSB \leq \overline{T3bPBF} \cdot \overline{T3aSB\_T3bPBF} \cdot \overline{T3bPBF\_T3aSB} \\ T3bPBF \leq \overline{T3aSB} \cdot \overline{T3aSB\_T3bPBF} \cdot \overline{T3bPBF\_T3aSB} \\ T3aSB\_T3bPBF \leq \overline{T3aSB} \cdot \overline{T3bPBF} \cdot \overline{T3bPBF\_T3aSB} \\ T3bPBF\_T3aSB \leq \overline{T3aSB} \cdot \overline{T3bPBF} \cdot \overline{T3aSB\_T3bPBF} \\ T3aSB\_p \leq \overline{T3bPBF\_p} \cdot \overline{T3aSB\_T3bPBF\_p} \cdot \overline{T3bPBF\_T3aSB\_p} \\ T3bPBF\_p \leq \overline{T3aSB\_p} \cdot \overline{T3aSB\_T3bPBF\_p} \cdot \overline{T3bPBF\_T3aSB\_p} \\ T3aSB\_T3bPBF\_p \leq \overline{T3aSB\_p} \cdot \overline{T3bPBF\_p} \cdot \overline{T3bPBF\_T3aSB\_p} \\ T3bPBF\_T3aSB\_p \leq \overline{T3aSB\_p} \cdot \overline{T3bPBF\_p} \cdot \overline{T3aSB\_T3bPBF\_p} \end{cases}$$

- La désactivation de la tâche $t3aSB$ est synchrone avec l'activation du temps d'attente $t3aSB\_t3bPBF$:

$$Des\_T3aSB = Act\_T3aSB\_T3bPBF$$

- L'activation de la tâche $t3bPBF$ est synchrone avec la désactivation du temps d'attente $t3aSB\_t3bPBF$:

$$Act\_T3bPBF = Des\_T3aSB\_T3bPBF$$

- La désactivation de la tâche $t3bPBF$ est synchrone avec l'activation du temps d'attente $t3bPBF\_t3aSB$:

$$Des\_T3bPBF = Act\_T3bPBF\_T3aSB$$

- L'activation de la tâche $t3aSB$ est synchrone avec la désactivation du temps d'attente $t3bPBF\_t3aSB$:

$$Act\_T3aSB = Des\_T3bPBF\_T3aSB$$

Les quatre autres enchaînements de tâches peuvent être formalisés de manière similaire.

4.4. Résolution du système d'équations

Pour cette étude de cas, nous avons finalement 16 inconnues (l'état courant des 6 tâches et 10 temps d'attente). Les fonctions booléennes à synthétiser sont de dimension 26 (6 conditions externes de terminaison, 4 conditions externes de lancement et les 16 états précédents des tâches et des temps d'attente).

La formalisation du cahier des charges donne lieu à 88 relations entre les 42 (16+26) fonctions booléennes principales du problème. Parmi ces 88 relations, 26 d'entre elles sont utilisées en tant qu'hypothèses.

La résolution du système d'équations a conduit à la solution paramétrique suivante :

$$\begin{cases} T0RP = T1PF\_T0RP\_p \cdot T2RF\_T0RP\_p \cdot T3bPBF\_T0RP\_p \cdot T4EF\_T0RP\_p \cdot Auto \cdot P_{T0RP} + T0RP\_p \cdot \overline{FinRot} \\ T1PF = T0RP\_T1PF\_p \cdot FlaconEnAttente \cdot P_{T1PF} \cdot (T0RP\_T4EF\_p + T4EF\_T0RP\_p + FlaconEvacue \cdot T4EF\_p) \\ \qquad + T1PF\_p \cdot \overline{FlaconPositionne} \\ T2RF = T0RP\_T2RF\_p \cdot P_{T2RF} + T2RF\_p \cdot \overline{FlaconRempli} \\ T3aSB = T3bPBF\_T3aSB\_p \cdot BouchonEnAttente \cdot P_{T3aSB} + T3aSB\_p \cdot \overline{BouchonSaisi} \\ T3bPBF = T0RP\_T3bPBF\_p \cdot T3aSB\_T3bPBF\_p \cdot P_{T3bPBF} + T3bPBF\_p \cdot \overline{BouchonSurFlacon} \\ T4EF = T0RP\_T4EF\_p \cdot GoulotteVide \cdot P_{T4EF} \\ \qquad \cdot (T1PF\_T0RP\_p + T1PF\_p \cdot FlaconPositionne + T0RP\_T1PF\_p \cdot (\overline{FlaconEnAttente} + \overline{P_{T1PF}})) \\ \qquad + T4EF\_p \cdot \overline{FlaconEvacue} \end{cases}$$

où P_{T0RP}, P_{T1PF}, P_{T2RF}, P_{T3aSB}, P_{T3bPBF} et P_{T4EF} sont des éléments de Γ_{26} introduits pour couvrir l'ensemble des solutions du problème.

Dans cette solution paramétrique calculée, chacune des tâches admet plusieurs solutions puisqu'elles dépendent toutes d'un paramètre différent. Cependant, en analysant plus finement cette solution paramétrique, on peut noter que ces paramètres ne conditionnent que le lancement des tâches. Cette caractéristique est due au fait que le cahier des charges donné ne faisait nullement référence au fait que ces tâches devaient être démarrées au plus tôt.

Pour éliminer cet indéterminisme dans le lancement des tâches, il faut rajouter à la formalisation du cahier des charges les contraintes suivantes :

$$\begin{cases} T1PF\_T0RP\_p \cdot T2RF\_T0RP\_p \cdot T3bPBF\_T0RP\_p \cdot T4EF\_T0RP\_p \cdot Auto \le Act\_T0RP \\ T0RP\_T2RF\_p \le Act\_T2RF \\ T3bPBF\_T3aSB\_p \cdot BouchonEnAttente \le Act\_T3aSB \\ T0RP\_T3bPBF\_p \cdot T3aSB\_T3bPBF\_p \le Act\_T3bPBF \\ T0RP\_T1PF\_p \cdot FlaconEnAttente \le Act\_T1PF + T4EF \\ T0RP\_T4EF\_p \cdot GoulotteVide \le Act\_T4EF + T1PF \end{cases}$$

Ces contraintes peuvent être obtenues automatiquement à partir du cahier des charges.

Une fois ces contraintes rajoutées, la résolution du système d'équations a conduit à la solution paramétrique suivante :

$$\begin{cases}
T0RP = T1PF_T0RP_p \cdot T2RF_T0RP_p \cdot T3bPBF_T0RP_p \cdot T4EF_T0RP_p \cdot Auto + T0RP_p \cdot \overline{FinRot} \\
T1PF = T0RP_T1PF_p \cdot FlaconEnAttente \\
\quad \cdot (T4EF_T0RP_p + T4EF_p \cdot FlaconEvacue + T0RP_T4EF_p \cdot (\overline{GoulotteVide} + P_{T1PF})) \\
\quad + T1PF_p \cdot \overline{FlaconPositionne} \\
T2RF = T0RP_T2RF_p + T2RF_p \cdot \overline{FlaconRempli} \\
T3aSB = T3bPBF_T3aSB_p \cdot BouchonEnAttente + T3aSB_p \cdot \overline{BouchonSaisi} \\
T3bPBF = T0RP_T3bPBF_p \cdot T3aSB_T3bPBF_p + T3bPBF_p \cdot \overline{BouchonSurFlacon} \\
T4EF = T0RP_T4EF_p \cdot \overline{T0RP_p} \cdot \overline{T4EF_T0RP_p} \cdot \overline{T4EF_p} \cdot GoulotteVide \\
\quad \cdot (T1PF_T0RP_p + T1PF_p \cdot FlaconPositionne + T0RP_T1PF_p \cdot (\overline{FlaconEnAttente} + \overline{P_{T1PF}})) \\
\quad + T4EF_p \cdot \overline{FlaconEvacue}
\end{cases}$$

Cette solution ne dépend plus que d'un unique paramètre P_{T1PF} (pour $T1PF$ et $T4EF$). Cette dépendance est cohérente avec le cahier des charges car il n'a jamais été précisé quelle est la tâche qui doit être démarrée en priorité lorsque toutes les conditions sont réunies pour permettre le lancement de $t1PF$ et de $t4EF$.

4.5. Conclusions relatives à l'étude de cas

Au travers de cette étude de cas, nous avons montré qu'il était possible d'établir les équations récurrentes de chacune de ces six tâches à l'aide de notre méthode de synthèse à partir des conditions externes de démarrage et de terminaison des tâches, des diagrammes d'enchaînement des tâches et de règles d'exclusion entre tâches.

En introduisant systématiquement un temps d'attente entre deux tâches qui s'enchaînent, nous sommes capables de synthétiser la loi de commande de systèmes dont certaines tâches sont communes à plusieurs enchaînements.

5. Conclusion

Dans ce chapitre, nous avons présenté dans le détail la méthode de synthèse de la loi de commande d'un SED que nous proposons. Cette loi de commande est obtenue par résolution d'un système d'équations entre des fonctions booléennes. Nous avons montré au début de ce chapitre

comment les relations entre des fonctions booléennes qui constituent ce système d'équations pouvaient être obtenues en formalisant les différents fragments du cahier des charges ou des éléments de solutions donnés par le concepteur.

Au travers des trois études de cas, nous avons montré comment notre méthode pouvait être utilisée pour la synthèse d'un système combinatoire, la recherche des conditions d'évolution dans un modèle à états ou la synthèse des équations récurrentes d'un système séquentiel.

Conclusions générales

Prévus initialement pour apporter une réponse au problème de la synthèse de la commande des systèmes logiques, les résultats obtenus dans le cadre de ce travail de thèse ont en définitive une portée plus importante que celle envisagée à leur lancement. Les résultats obtenus ne se limitent en effet pas au seul domaine des SED mais concernent plus généralement le domaine des mathématiques discrètes. En recherchant une réponse au problème de la synthèse de fonctions booléennes, nous avons eu l'opportunité de pouvoir établir un résultat beaucoup plus général puisque valable pour toutes les algèbres de Boole. Sur le plan mathématique, il nous apparaît que le théorème 10 démontré au chapitre 2 est certainement le résultat le plus important de ce travail de thèse.

Dans le domaine des SED, vouloir synthétiser automatiquement la commande d'un système à partir de son cahier des charges est un projet ambitieux qui correspond à de réels besoins industriels. La méthode présentée dans ce mémoire de thèse a été élaborée pour apporter une réponse originale et adaptée au problème spécifique de la synthèse des systèmes logiques.

Au travers du traitement des trois études de cas présentées au chapitre 3, nous avons montré que la méthode que nous avons mise au point permettait d'obtenir la loi de commande d'un SED logique en résolvant un système d'équations entre des fonctions booléennes.

Les points forts de notre approche sont au nombre de trois :

- À la différence des autres méthodes de synthèse, le résultat fourni par notre méthode est une représentation paramétrique de l'ensemble des solutions du problème. Nous considérons que

c'est un réel plus pour le concepteur. Donner accès au concepteur à l'ensemble des solutions du problème lui permet d'évaluer la qualité et la complétude du cahier des charges qu'il a formalisé. En ayant accès aux degrés de liberté de sa solution, le concepteur peut décider ou non d'affiner son travail de formalisation des besoins. Pour nous, il est en effet évident que la synthèse d'un système logique est une activité itérative dans laquelle l'expression des besoins se précise au cours de l'étude.

- À la différence des autres méthodes de synthèse, notre méthode propose également un résultat en cas d'incohérence des données de départ. Lorsque le système d'équations ne peut admettre de solution, nous retournons au concepteur la cause de cette incohérence. En analysant cette information, il lui est possible d'identifier quels sont les éléments du cahier des charges qui sont incohérents. Nous avons personnellement très largement exploité cette caractéristique lors de la mise au point des études de cas !

- À la différence des autres méthodes de synthèse, notre méthode permet au concepteur de proposer ses propres éléments de solutions. Il peut donc fixer la forme de la solution qu'il souhaite obtenir ou n'utiliser cette méthode pour synthétiser qu'une partie de son modèle. Cette possibilité a été présentée dans le détail lors de l'étude de cas n°2.

Notre méthode de synthèse possède également des limites. Certaines d'entre elles sont d'ailleurs communes avec toutes les méthodes de synthèse, d'autres lui sont plus spécifiques :

- Pour pouvoir synthétiser automatiquement la loi de commande à partir d'un cahier des charges, il est nécessaire de disposer d'une description formelle des informations qu'il contient. En conséquence, si les méthodes de synthèse dispensent bien le concepteur de trouver une solution du problème, elle l'oblige cependant à devoir spécifier avec précision son problème. Par expérience, pour des problèmes simples, il est plus difficile de spécifier proprement le problème que de trouver une solution. Cependant, lorsque les problèmes augmentent en taille ou en complexité, le concepteur a de plus en plus de mal à identifier une solution du problème mais il peut cependant continuer à spécifier proprement celui-ci. À terme, l'introduction des méthodes de synthèse automatique simplifiera le travail des concepteurs en les déchargeant des activités de recherche de solutions au bénéfice des activités de spécification qui sont, elles, plus large-

ment capitalisables d'un projet à l'autre. Pour rendre notre méthode plus performante, il est néanmoins nécessaire d'assister au maximum le concepteur lors de la formalisation de ses besoins. Nous avons proposé quelques pistes au début du chapitre 3 de ce mémoire mais il nous apparaît évident que de nombreux travaux restent à faire dans le domaine.

- Proposer au concepteur toutes les solutions pour son problème et non seulement une seule solution peut également être perçu comme une limite de l'approche car le choix d'une solution particulière peut s'avérer difficile. Dans ce mémoire de thèse, il a été décidé de laisser ce problème en suspens pour éviter de ne le traiter qu'approximativement. Il nous semble cependant que le cadre mathématique que nous proposons offre de nombreuses possibilités pour développer des méthodes de choix de paramètre grâce, notamment, à la relation d'ordre partiel qu'il contient (relation inclusion). Grâce à cette relation d'ordre, il est possible de comparer les solutions et certainement de pouvoir identifier automatiquement un jeu de paramètres permettant de définir une solution optimale pour un critère donné.

- Les résultats théoriques présentés dans ce mémoire démontrent que la résolution d'un système d'équations, quels que soient le nombre d'équations et le nombre d'inconnues, est possible. Les calculs nécessaires pour traiter ne serait-ce que la première étude de cas ont montré que la mise en œuvre d'une telle approche ne peut se faire sans une assistance informatique. Dans le cadre de cette thèse, différents développements logiciels ont été réalisés. Les derniers essais ont montré qu'une approche à base de calcul formel sur des BDD pouvait être très performante. Pour être capables de traiter des études de cas plus complexes, une analyse détaillée de l'algorithmie mise en œuvre doit maintenant être réalisée pour déterminer quelles sont les étapes qui peuvent être optimisées afin de rendre plus efficaces les différents calculs symboliques que nous sommes amenés à réaliser. Il serait également intéressant d'étudier la capacité de nos problèmes à être décomposables en sous-problèmes indépendants afin de faciliter leur résolution.

De nombreux travaux théoriques restent donc encore à mener pour développer les potentialités de cette approche. Ces travaux théoriques doivent cependant, de notre point de vue, être accompagnés d'expérimentations sur des cas réels afin de pouvoir juger par la pratique de la pertinence de cette approche novatrice.

Références bibliographiques

[AKE59] S.B. Akers Jr. On a theory of Boolean functions. Journal of the Society for Industrial and Applied Mathematics, 7(4): 487-498, 1959.

[AKE03] K. Akesson, M. Fabian, H.Flordal et A. Vahidi. Supremica: a tool for verification and synthesis of discrete event supervisors. 11th Mediterranean Conference on Control and Automation, 2003.

[BAL92] S. Balemi. Control of discrete event processes: theory and application. Thèse de doctorat, Swiss Federal Institute of Technology, Zurich, Suisse, 1992.

[BOO1854] G. Boole. An investigation into the laws of thought, on which are founded the mathematical theories of logic and probability. Macmillan, 1854.

[BRU98] N.P. Brusentsov et Y.S. Vladimirova. Solution of Boolean equations. Computational Mathematics and Modeling, 9(4): 287-295, 1998.

[CAN08] M. Cantarelli et J.-M. Roussel. Reactive control system design using the Supervisory Control Theory: evaluation of possibilities and limits. 9th International Workshop On Discrete Event Systems, WODES'08, pages 200-205, Göteborg (Sweden), 2008.

[GRI04] R.P. Grimaldi. Discrete and combinatorial mathematics: an applied introduction. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 5th edition, ISBN: 0-321-21103-0, 980 pages, 2004.

[GUN96] J. Gunnarsson et J. Plantin. Synthesis of a discrete system using algebraic methods. In Proceedings of International Workshop on Discrete Event Systems, Edinburgh, UK, pages 220-2252, 1996.

[GUN97] J. Gunnarsson. Symbolic methods and tools for discrete event dynamic systems. PhD thesis, Division of Automatic Control, Department of Electrical Engineering, Linköping University, 1997.

[GOU99] A. Gouin. Contribution à la commande de systèmes à événements discrets temporisés : synthèse de superviseur dans le cadre de modèle automate. Thèse de doctorat, ISTIA - Université d'Angers, 1999.

[GOU04] D. Gouyon, J. Petin et A. Gouin. Pragmatic approach for modular control synthesis and implementation. International Journal of Production Research, 42(14): 2839-2858, 2004.

[HIE08A] Y. Hietter, J.-M. Roussel et J.-J. Lesage. Algebraic synthesis of transition conditions of a state model. 9th International Workshop On Discrete Event Systems, WODES'08, pages 187-192, Göteborg (Sweden), 2008.

[HIE08B] Y. Hietter, J.-M. Roussel et J.-J. Lesage. Algebraic synthesis of dependable logic controllers. 17th IFAC World Congress, Seoul (Korea), pages 4132-4137, 2008.

[HIE08C] Y. Hietter, J.-M. Roussel et J.-J. Lesage. Calcul des conditions de transition d'un Réseau de Petri par synthèse algébrique. Conférence Internationale Francophone d'Automatique, CIFA 2008, Bucarest (Roumanie), CDRom papier N°83, 6p., 2008.

[HOP79] J.E. Hopcroft, R. Motwani et J.D. Ullman. Introduction to automata theory, languages, and computation. Addison-Wesley, 1979.

[KUM91] R. Kumar. Supervisory synthesis techniques for discrete event dynamical systems. PhD thesis, Texas University, 1991.

[LED02] R.J. Leduc. Hierarchical interface-based supervisory control. PhD thesis, University of Toronto, Department of Electrical and Computer Engineering, 2002.

[LEV00A] V.S. Levchenkov. Solution of equations in Boolean algebra. Computational Mathematics and Modeling, 11(2): 154-163, 2000.

[LEV00B] V.S. Levchenkov. Boolean equations with many unknowns. Computational Mathematics and Modeling, 11(2): 143-153, 2000.

[LOH06] S. Lohmann, L.A.D. Thi et O. Stursberg. Design of verified logic control programs. IEEE International Conference on Control Applications, pages 1855-1860, 2006.

[MAC06] J. Machado, B. Denis, J.-J. Lesage, J.-M. Faure, et J. Ferreira Da Silva. Logic controllers dependability verification using a plant model. 3th IFAC Workshop on Discrete-Event System Design. DESDes'06, Rydzyna (Poland), pages 37-42, Sep 2006.

[MAD97] A. Mader. Verification of modal properties using boolean equation systems. PhD thesis, Bertz Verlag, Berlin, versal 8 edition, 1997.

[MAT06] R. Mateescu. CAESAR SOLVE: A generic library for on-the-fly resolution of alternation-free Boolean equation systems. International Journal on Software Tools for Technology Transfer (STTT), 8(1): 37-56, 2006.

[MED07] A. Medina Rodriguez. Méthode de synthèse d'un contrôleur logique à partir de spécifications algébriques, Thèse de doctorat, École Normale Supérieure de Cachan, 2007.

[MEF05] T. Meftah, N. Bouteille, H. Gueguen et V. Boutin. Constraint specification of the control logic of automated manufacturing systems. 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2, 2005.

[PET07] J.-F. Petin, D. Gouyon et G. Morel. Supervisory synthesis for product-driven automation and its application to a flexible assembly cell. Control Engineering Practice , 15(5): 595-614, 2007.

[RAM87A] P.J.G. Ramadge et W.M. Wonham. Supervisory control of a class of discrete event processes. SIAM Journal on Control and Optimization, 25: 206-230, 1987.

[RAM87B] P.J.G. Ramadge et W.M. Wonham. On the supremal controllable sublanguage of a given language. SIAM Journal of Control and Optimization, 25(3): 637-659, 1987.

[RAM89] P.J.G. Ramadge et W.M. Wonham. The control of discrete event systems. Proceedings of the IEEE, 77(1): 81-98, 1989.

[ROU93] J.-M. Roussel, J.-J. Lesage. Une algèbre de Boole pour l'approche évènementielle des systèmes logiques, APII-AFCET/CNRS, Ed Hermes, 27(5): 541-560, 1993.

[ROU02] J.-M. Roussel, B. Denis. Safety properties verification of ladder diagram programs. Journal Européen des Systèmes Automatisés, 36(7): 905-917, 2002.

[ROU04] J.-M. Roussel, J.-M. Faure, J.-J. Lesage, A. Medina. Algebraic approach for dependable logic control systems design. International Journal of Production Research, 42(14): 2859-2876, 2004.

[ROU05] J.-M. Roussel et A. Guia. Designing dependable logic controllers using the supervisory control theory. 16th IFAC World Congress, CDROM paper n° 04427, Prague, 2005.

[ROU06] J.-M. Roussel, J.-M. Faure. Designing dependable controllers using algebraic specifications. Control Engineering Practice, 14(10): 1143-1155, 2006.

[RUD03] S. Rudeanu. Algebraic methods versus map methods of solving boolean equations. International Journal of Computer Mathematics, 80(7): 815-817, 2003.

[RUS04] A.M.A. Rushdi. Efficient Solution of Boolean Equations Using Variable-Entered Karnaugh Maps. Journal of king abdulaziz university, 15(1): 115-131, 2004.

[SHA40] C.E. Shannon. A symbolic analysis of relay and switching circuits. Master's thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, 1940.

[STU05] O. Stursberg, S. Lohmann et S. Engell. Improving Dependability of Logic Controllers by Algorithmic Verification. 16th IFAC World Congress, Prague, 2005.

[TOM66] R.M. Toms. Systems of Boolean Equations. The American Mathematical Monthly, 73(1): 29-35, 1966.

[WOO96] S. Woods et G. Casinovi. Efficient Solution of Systems of Boolean Equations. Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design, 542-546, 1996.

[ZAH87] J. Zahnd. Machines séquentielles. Presses Polytechniques et Universitaires Romandes (PPUR). 3 edition, ISBN : 2-88074-051-7, 265 pages, 1987.

Annexe A :
Démonstrations des propriétés

Cette annexe contient toutes les démonstrations des propriétés données au chapitre 2. Les démonstrations sont faites de manière directe en n'utilisant que les axiomes de la définition d'une algèbre de Boole (propriétés [1] à [9]) et les propriétés déjà démontrées. À chaque pas de progression est indiqué le nom de la propriété utilisée. Pour éviter d'alourdir inutilement les démonstrations, il a été choisi de ne pas introduire de pas de progression lorsque la propriété de Commutativité est utilisée.

Démonstration de la propriété [10] : $x + x = x$ (Idempotence)

$x = (x + 0)$	Élément neutre
$= x + (x \cdot \bar{x})$	Élément complémentaire
$= (x + x) \cdot (x + \bar{x})$	Distributivité
$= (x + x) \cdot 1$	Élément complémentaire
$= x + x$	Élément neutre

□

Démonstration de la propriété [11] : $x \cdot x = x$ (Idempotence)

$x = x \cdot 1$	Élément neutre
$= x \cdot (x + \bar{x})$	Élément complémentaire
$= (x \cdot x) + (x \cdot \bar{x})$	Distributivité

$= (x \cdot x) + 0$ Élément complémentaire
$= x \cdot x$ Élément neutre

□

Démonstration de la propriété [12] : $x + 1 = 1$ (Loi de la dominance)

$x + 1$
$= (x + 1) \cdot 1$ Élément neutre
$= (x + 1) \cdot (x + \bar{x})$ Élément complémentaire
$= x + (1 \cdot \bar{x})$ Distributivité
$= x + \bar{x}$ Élément neutre
$= 1$ Élément complémentaire

□

Démonstration de la propriété [13] : $x \cdot 0 = 0$ (Loi de la dominance)

$x \cdot 0$
$= (x \cdot 0) + 0$ Élément neutre
$= (x \cdot 0) + (x \cdot \bar{x})$ Élément complémentaire
$= x \cdot (0 + \bar{x})$ Distributivité
$= x \cdot \bar{x}$ Élément neutre
$= 0$ Élément complémentaire

□

Démonstration de la propriété [14] : $x + (x \cdot y) = x$ (Élément absorbant)

$x + (x \cdot y)$
$= (x \cdot 1) + (x \cdot y)$ Élément neutre
$= x \cdot (1 + y)$ Distributivité
$= x \cdot 1$ Loi de la dominance
$= x$ Élément neutre

□

Démonstration de la propriété [15] : $x \cdot (x + y) = x$ (Élément absorbant)

$x \cdot (x + y)$

$= (x + 0) \cdot (x + y)$	Élément neutre
$= x + (0 \cdot y)$	Distributivité
$= x + 0$	Loi de la dominance
$= x$	Élément neutre

□

Démonstration de propriété [16] : $x + (\bar{x} \cdot y) = x + y$ (Élément absorbant)

$x + (\bar{x} \cdot y)$

$= (x + \bar{x}) \cdot (x + y)$	Distributivité
$= 1 \cdot (x + y)$	Élément complémentaire
$= x + y$	Élément neutre

□

Démonstration de la propriété [17] : $x \cdot (\bar{x} + y) = x \cdot y$ (Élément absorbant)

$x \cdot (\bar{x} + y)$

$= (x \cdot \bar{x}) + (x \cdot y)$	Distributivité
$= 0 + (x \cdot y)$	Élément complémentaire
$= x \cdot y$	Élément neutre

□

Démonstration de la propriété [18] : $\begin{cases} x + y = x + z \\ \bar{x} + y = \bar{x} + z \end{cases} \Leftrightarrow y = z$ (Loi de l'élimination)

- $\begin{cases} x + y = x + z \\ \bar{x} + y = \bar{x} + z \end{cases} \Rightarrow y = z$

$y = y + 0$	Élément neutre
$= y + (x \cdot \bar{x})$	Élément complémentaire
$= (y + x) \cdot (y + \bar{x})$	Distributivité
$= (z + x) \cdot (z + \bar{x})$	Par hypothèse
$= z + (x \cdot \bar{x})$	Distributivité

$= z + 0$ Élément complémentaire
$= z$ Élément neutre

- $y = z \Rightarrow \begin{cases} x + y = x + z \\ \bar{x} + y = \bar{x} + z \end{cases}$

$x + y$
$= x + z$ Par hypothèse
$\bar{x} + y$
$= \bar{x} + z$ Par hypothèse

□

Démonstration de la propriété [19] : $\begin{cases} x \cdot y = x \cdot z \\ \bar{x} \cdot y = \bar{x} \cdot z \end{cases} \Leftrightarrow y = z$ (Loi de l'élimination)

- $\begin{cases} x \cdot y = x \cdot z \\ \bar{x} \cdot y = \bar{x} \cdot z \end{cases} \Rightarrow y = z$

$y = y \cdot 1$ Élément neutre
$= y \cdot (x + \bar{x})$ Élément complémentaire
$= (y \cdot x) + (y \cdot \bar{x})$ Distributivité
$= (z \cdot x) + (z \cdot \bar{x})$ Par hypothèse
$= z \cdot (x + \bar{x})$ Distributivité
$= z \cdot 1$ Élément complémentaire
$= z$ Élément neutre

- $y = z \Rightarrow \begin{cases} x \cdot y = x \cdot z \\ \bar{x} \cdot y = \bar{x} \cdot z \end{cases}$

$x \cdot y$
$= x \cdot z$ Par hypothèse

$\bar{x} \cdot y$
$= \bar{x} \cdot z$ Par hypothèse

□

Démonstration de la propriété [20] : $x + (y + z) = (x + y) + z$ (Associativité)

La démonstration se fait en utilisant la propriété « Loi de l'élimination » :

$$\begin{cases} X \cdot Y = X \cdot Z \\ \bar{X} \cdot Y = \bar{X} \cdot Z \end{cases} \Leftrightarrow Y = Z \text{ où } X = x,\ Y = x + (y + z) \text{ et } Z = (x + y) + z$$

$X \cdot Y$

$= x \cdot (x + (y + z))$	Par hypothèse
$= x$	Élément absorbant

$X \cdot Z$

$= x \cdot ((x + y) + z)$	Par hypothèse
$= (x \cdot (x + y)) + (x \cdot z)$	Distributivité
$= x + (x \cdot z)$	Élément absorbant
$= x$	Élément absorbant

$\bar{X} \cdot Y$

$= \bar{x} \cdot (x + (y + z))$	Par hypothèse
$= (\bar{x} \cdot x) + (\bar{x} \cdot (y + z))$	Distributivité
$= 0 + (\bar{x} \cdot (y + z))$	Élément complémentaire
$= \bar{x} \cdot (y + z)$	Élément neutre

$\bar{X} \cdot Z$

$= \bar{x} \cdot ((x + y) + z)$	Par hypothèse
$= (\bar{x} \cdot (x + y)) + (\bar{x} \cdot z)$	Distributivité
$= (\bar{x} \cdot y) + (\bar{x} \cdot z)$	Élément absorbant
$= \bar{x} \cdot (y + z)$	Distributivité

Comme $X \cdot Y = X \cdot Z$ et $\bar{X} \cdot Y = \bar{X} \cdot Z$, alors $x + (y + z) = Y = Z = (x + y) + z$

□

Démonstration de la propriété [21] : $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ (Associativité)

La démonstration se fait en utilisant la propriété « Loi de l'élimination » :

$$\begin{cases} X + Y = X + Z \\ \bar{X} + Y = \bar{X} + Z \end{cases} \Leftrightarrow Y = Z \text{ où } X = x,\ Y = x \cdot (y \cdot z) \text{ et } Z = (x \cdot y) \cdot z$$

$X + Y$

$= x + (x \cdot (y \cdot z))$	Par hypothèse
$= x$	Élément absorbant

$X + Z$

$= x + ((x \cdot y) \cdot z)$	Par hypothèse
$= (x + (x \cdot y)) \cdot (x + z)$	Distributivité
$= x \cdot (x + z)$	Élément absorbant
$= x$	Élément absorbant

$\bar{X} + Y$

$= \bar{x} + (x \cdot (y \cdot z))$	Par hypothèse
$= (\bar{x} + x) \cdot (\bar{x} + (y \cdot z))$	Distributivité
$= 1 \cdot (\bar{x} + (y \cdot z))$	Élément complémentaire
$= \bar{x} + (y \cdot z)$	Élément neutre

$\bar{X} + Z$

$= \bar{x} + ((x \cdot y) \cdot z)$	Par hypothèse
$= (\bar{x} + (x \cdot y)) \cdot (\bar{x} + z)$	Distributivité
$= ((\bar{x} + x) \cdot (\bar{x} + y)) \cdot (\bar{x} + z)$	Distributivité
$= (1 \cdot (\bar{x} + y)) \cdot (\bar{x} + z)$	Élément complémentaire
$= (\bar{x} + y) \cdot (\bar{x} + z)$	Élément neutre
$= \bar{x} + (y \cdot z)$	Distributivité

Comme $X + Y = X + Z$ et $\bar{X} + Y = \bar{X} + Z$, alors $x \cdot (y \cdot z) = Y = Z = (x \cdot y) \cdot z$

□

Démonstration de la propriété [22] : $\begin{cases} x + y = 1 \\ x \cdot y = 0 \end{cases} \Leftrightarrow y = \bar{x}$ (Unicité de l'inverse)

- $\begin{cases} x + y = 1 \\ x \cdot y = 0 \end{cases} \Rightarrow y = \bar{x}$

$y = y \cdot 1$	Élément neutre
$= y \cdot (x + \bar{x})$	Élément complémentaire
$= (y \cdot x) + (y \cdot \bar{x})$	Distributivité
$= 0 + (y \cdot \bar{x})$	Par hypothèse
$= (x \cdot \bar{x}) + (y \cdot \bar{x})$	Élément complémentaire
$= (x + y) \cdot \bar{x}$	Distributivité

$= 1 \cdot \bar{x}$ Par hypothèse

$= \bar{x}$ Élément neutre

- $y = \bar{x} \Rightarrow \begin{cases} x + y = 1 \\ x \cdot y = 0 \end{cases}$

$x + y$

$= x + \bar{x}$ Par hypothèse

$= 1$ Élément complémentaire

$x \cdot y$

$= x \cdot \bar{x}$ Par hypothèse

$= 0$ Élément complémentaire

□

Démonstration de la propriété [23] : $\bar{\bar{x}} = x$ (Théorème du double complément)

La démonstration se fait en utilisant la propriété « Unicité de l'inverse » :

$$\begin{cases} X + Y = 1 \\ X \cdot Y = 0 \end{cases} \Leftrightarrow Y = \bar{X} \text{ où } X = \bar{x} \text{ et } Y = x$$

$X + Y$

$= \bar{x} + x$ Par hypothèse

$= 1$ Élément complémentaire

$X \cdot Y$

$= \bar{x} \cdot x$ Par hypothèse

$= 0$ Élément complémentaire

Comme $X + Y = 1$ et $X \cdot Y = 0$, alors $x = Y = \bar{X} = \bar{\bar{x}}$

□

Démonstration de la propriété [24] : $\overline{(x + y)} = \bar{x} \cdot \bar{y}$ (Théorème de De Morgan)

La démonstration se fait en utilisant la propriété « Unicité de l'inverse » :

$$\begin{cases} X + Y = 1 \\ X \cdot Y = 0 \end{cases} \Leftrightarrow Y = \bar{X} \text{ où } X = x + y \text{ et } Y = \bar{x} \cdot \bar{y}$$

$X + Y$

$= (x + y) + (\bar{x} \cdot \bar{y})$	Par hypothèse
$= x + (y + (\bar{x} \cdot \bar{y}))$	Associativité
$= x + (y + \bar{x})$	Élément absorbant
$= (x + \bar{x}) + y$	Associativité
$= 1 + y$	Élément complémentaire
$= 1$	Loi de la dominance

$X \cdot Y$

$= (x + y) \cdot (\bar{x} \cdot \bar{y})$	Par hypothèse
$= (x \cdot (\bar{x} \cdot \bar{y})) + (y \cdot (\bar{x} \cdot \bar{y}))$	Distributivité
$= ((x \cdot \bar{x}) \cdot \bar{y}) + ((y \cdot \bar{y}) \cdot \bar{x})$	Associativité
$= (0 \cdot \bar{y}) + (0 \cdot \bar{x})$	Élément complémentaire
$= 0 + 0$	Loi de la dominance
$= 0$	Élément neutre

Comme $X + Y = 1$ et $X \cdot Y = 0$, alors $\bar{x} \cdot \bar{y} = Y = \bar{X} = \overline{(x + y)}$

□

Démonstration de la propriété [25] : $\overline{(x \cdot y)} = \bar{x} + \bar{y}$ (Théorème de De Morgan)

La démonstration se fait en utilisant la propriété « Unicité de l'inverse » :

$$\begin{cases} X + Y = 1 \\ X \cdot Y = 0 \end{cases} \Leftrightarrow Y = \bar{X} \text{ où } X = x \cdot y \text{ et } Y = \bar{x} + \bar{y}$$

$X + Y$

$= (x \cdot y) + (\bar{x} + \bar{y})$	Par hypothèse
$= ((x \cdot y) + \bar{x}) + \bar{y}$	Associativité
$= ((\bar{\bar{x}} \cdot y) + \bar{x}) + \bar{y}$	Théorème du double complément
$= (y + \bar{x}) + \bar{y}$	Élément absorbant
$= (y + \bar{y}) + \bar{x}$	Associativité
$= 1 + \bar{x}$	Élément complémentaire
$= 1$	Loi de la dominance

$X \cdot Y$

$= (x \cdot y) \cdot (\bar{x} + \bar{y})$	Par hypothèse

$= ((x \cdot y) \cdot \bar{x}) + ((x \cdot y) \cdot \bar{y})$ Distributivité

$= (y \cdot (x \cdot \bar{x})) + (x \cdot (y \cdot \bar{y}))$ Associativité

$= (y \cdot 0) + (x \cdot 0)$ Élément complémentaire

$= 0 + 0$ Loi de la dominance

$= 0$ Élément neutre

Comme $X + Y = 1$ et $X \cdot Y = 0$, alors $\bar{x} + \bar{y} = Y = \bar{X} = \overline{(x \cdot y)}$

□

Démonstration de la propriété [26] : $\bar{0} = 1$

La démonstration se fait en utilisant la propriété « Unicité de l'inverse » :

$$\begin{cases} X + Y = 1 \\ X \cdot Y = 0 \end{cases} \Leftrightarrow Y = \bar{X} \text{ où } X = 0 \text{ et } Y = 1$$

$X + Y$

$= 0 + 1$ Par hypothèse

$= 1$ Loi de la dominance

$X \cdot Y$

$= 0 \cdot 1$ Par hypothèse

$= 0$ Loi de la dominance

Comme $X + Y = 1$ et $X \cdot Y = 0$, alors $1 = Y = \bar{X} = \bar{0}$

□

Démonstration de la propriété [27] : $\bar{1} = 0$

La démonstration se fait en utilisant la propriété « Unicité de l'inverse » :

$$\begin{cases} X + Y = 1 \\ X \cdot Y = 0 \end{cases} \Leftrightarrow Y = \bar{X} \text{ où } X = 1 \text{ et } Y = 0$$

$X + Y$

$= 1 + 0$ Par hypothèse

$= 1$ Loi de la dominance

$X \cdot Y$

$= 1 \cdot 0$	Par hypothèse
$= 0$	Loi de la dominance

Comme $X + Y = 1$ et $X \cdot Y = 0$, alors $0 = Y = \overline{X} = \overline{1}$

□

Démonstration de la propriété [28] : $x + \overline{y} = 1 \Leftrightarrow x + y = x$

- $x + \overline{y} = 1 \Rightarrow x + y = x$

$x + y$

$= (x + y) \cdot 1$	Élément neutre
$= (x + y) \cdot (x + \overline{y})$	Par hypothèse
$= (x \cdot (x + \overline{y})) + (y \cdot (x + \overline{y}))$	Distributivité
$= x + (y \cdot (x + \overline{y}))$	Élément absorbant
$= x + (y \cdot x)$	Élément absorbant
$= x$	Élément absorbant

- $x + y = x \Rightarrow x + \overline{y} = 1$

$x + \overline{y}$

$= (x + y) + \overline{y}$	Par hypothèse
$= x + (y + \overline{y})$	Associativité
$= x + 1$	Élément complémentaire
$= 1$	Loi de la dominance

□

Démonstration de la propriété [29] : $x \cdot \overline{y} = 0 \Leftrightarrow x \cdot y = x$

- $x \cdot \overline{y} = 0 \Rightarrow x \cdot y = x$

$x \cdot y$

$= (x \cdot y) + 0$	Élément neutre
$= (x \cdot y) + (x \cdot \overline{y})$	Par hypothèse
$= (x + (x \cdot \overline{y})) \cdot (y + (x \cdot \overline{y}))$	Distributivité

$= x \cdot (y + (x \cdot \bar{y}))$ Élément absorbant
$= x \cdot (y + x)$ Élément absorbant
$= x$ Élément absorbant

- $x \cdot y = x \Rightarrow x \cdot \bar{y} = 0$

$x \cdot \bar{y}$
$= (x \cdot y) \cdot \bar{y}$ Par hypothèse
$= x \cdot (y \cdot \bar{y})$ Associativité
$= x \cdot 0$ Élément complémentaire
$= 0$ Loi de la dominance

□

Démonstration de la propriété [30] : $((x \cdot y) + (\bar{x} \cdot z)) + (y \cdot z) = (x \cdot y) + (\bar{x} \cdot z)$ (Théorème de la redondance)

$((x \cdot y) + (\bar{x} \cdot z)) + (y \cdot z)$
$= ((x \cdot y) + (\bar{x} \cdot z)) + (1 \cdot (y \cdot z))$ Élément neutre
$= ((x \cdot y) + (\bar{x} \cdot z)) + ((x + \bar{x}) \cdot (y \cdot z))$ Élément complémentaire
$= ((x \cdot y) + (\bar{x} \cdot z)) + ((x \cdot (y \cdot z)) + (\bar{x} \cdot (y \cdot z)))$ Distributivité
$= ((x \cdot y) + (\bar{x} \cdot z)) + (((x \cdot y) \cdot z) + ((\bar{x} \cdot z) \cdot y))$ Associativité
$= ((x \cdot y) + ((x \cdot y) \cdot z)) + ((\bar{x} \cdot z) + ((\bar{x} \cdot z) \cdot y))$ Associativité
$= (x \cdot y) + (\bar{x} \cdot z)$ Élément absorbant

□

Démonstration de la propriété [31] : $x = y \Leftrightarrow (x \cdot \bar{y}) + (\bar{x} \cdot y) = 0$

- $x = y \Rightarrow (x \cdot \bar{y}) + (\bar{x} \cdot y) = 0$

$(x \cdot \bar{y}) + (\bar{x} \cdot y)$
$= (x \cdot \bar{x}) + (\bar{x} \cdot x)$ Par hypothèse
$= 0 + 0$ Élément complémentaire
$= 0$ Élément neutre

- $(x \cdot \bar{y}) + (\bar{x} \cdot y) = 0 \Rightarrow x = y$

$x = x \cdot 1$	Élément neutre
$= x \cdot (y + \bar{y})$	Élément complémentaire
$= (x \cdot y) + (x \cdot \bar{y})$	Distributivité
$= ((x \cdot y) + (x \cdot \bar{y})) + 0$	Élément neutre
$= ((x \cdot y) + (x \cdot \bar{y})) + ((x \cdot \bar{y}) + (\bar{x} \cdot y))$	Par hypothèse
$= ((x \cdot y) + (\bar{x} \cdot y)) + ((x \cdot \bar{y}) + (x \cdot \bar{y}))$	Associativité
$= ((x \cdot y) + ((\bar{x} \cdot y) + (\bar{x} \cdot y))) + (x \cdot \bar{y})$	Idempotence
$= ((x \cdot y) + (\bar{x} \cdot y)) + ((x \cdot \bar{y}) + (\bar{x} \cdot y))$	Associativité
$= ((x \cdot y) + (\bar{x} \cdot y)) + 0$	Par hypothèse
$= (x \cdot y) + (\bar{x} \cdot y)$	Élément neutre
$= (x + \bar{x}) \cdot y$	Distributivité
$= 1 \cdot y$	Élément complémentaire
$= y$	Élément neutre

□

Démonstration de la propriété [32] : $\begin{cases} x = 0 \\ y = 0 \end{cases} \Leftrightarrow x + y = 0$

- $\begin{cases} x = 0 \\ y = 0 \end{cases} \Rightarrow x + y = 0$

$x + y$	
$= 0 + 0$	Par hypothèse
$= 0$	Élément neutre

- $x + y = 0 \Rightarrow \begin{cases} x = 0 \\ y = 0 \end{cases}$

$x = x + 0$	Élément neutre
$= x + (y \cdot \bar{y})$	Élément complémentaire
$= (x + y) \cdot (x + \bar{y})$	Distributivité
$= 0 \cdot (x + \bar{y})$	Par hypothèse
$= 0$	Loi de la dominance
$y = y + 0$	Élément neutre
$= y + (x \cdot \bar{x})$	Élément complémentaire

$= (y + x) \cdot (y + \bar{x})$ Distributivité

$= 0 \cdot (y + \bar{x})$ Par hypothèse

$= 0$ Loi de la dominance

□

Démonstration de la propriété [33] : $x = y \Leftrightarrow \bar{x} = \bar{y}$

- $x = y \Rightarrow \bar{x} = \bar{y}$

$\bar{x} = \bar{y}$ Par hypothèse

- $\bar{x} = \bar{y} \Rightarrow x = y$

$x = \overline{(\bar{x})}$ Théorème du double complément

$= \overline{(\bar{y})}$ Par hypothèse

$= y$ Théorème du double complément

□

Démonstration de la propriété [34] : $0 \leq x$

Sachant que :

$0 \cdot x = 0$ Loi de la dominance

Nous avons bien $0 \leq x$.

□

Démonstration de la propriété [35] : $x \leq 1$

Sachant que :

$x \cdot 1 = 1$ Élément neutre

Nous avons bien $x \leq 1$.

□

Démonstration de la propriété [36] : $(x \cdot y) \leq y$

Sachant que :

$(x \cdot y) \cdot y$

$= x \cdot (y \cdot y)$ Associativité

$= x \cdot y$ Idempotence

Comme $(x \cdot y) \cdot y = x \cdot y$, nous avons bien $(x \cdot y) \leq y$.

□

Démonstration de la propriété [37] : $x \leq (x + y)$

Sachant que :

$x \cdot (x + y) = x$ Élément absorbant

Nous avons bien $x \leq (x + y)$.

□

Démonstration de la propriété [38] : $x \leq (y \cdot z) \Leftrightarrow \begin{cases} x \leq y \\ x \leq z \end{cases}$

Par définition :

$$x \leq (y \cdot z) \Leftrightarrow x \cdot (y \cdot z) = x$$

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$x \leq z \Leftrightarrow x \cdot z = x$$

- $x \cdot (y \cdot z) = x \Rightarrow \begin{cases} x \cdot y = x \\ x \cdot z = x \end{cases}$

$x \cdot y$

$= (x \cdot y) \cdot 1$ Élément neutre

$= (x \cdot y) \cdot (z + \bar{z})$ Élément complémentaire

$= ((x \cdot y) \cdot z) + ((x \cdot y) \cdot \bar{z})$ Distributivité

$= (x \cdot (y \cdot z)) + (x \cdot (y \cdot \bar{z}))$ Associativité

$= x + (x \cdot (y \cdot \bar{z}))$ Par hypothèse

$= x$ Élément absorbant

$x \cdot z$

$= (x \cdot z) \cdot 1$ Élément neutre

$= (x \cdot z) \cdot (y + \bar{y})$ Élément complémentaire

$= ((x \cdot z) \cdot y) + ((x \cdot z) \cdot \bar{y})$ Distributivité

$= (x \cdot (y \cdot z)) + (x \cdot (z \cdot \bar{y}))$ Associativité

$= x + (x \cdot (z \cdot \bar{y}))$ Par hypothèse

$= x$ Élément absorbant

- $\begin{cases} x \cdot y = x \\ x \cdot z = x \end{cases} \Rightarrow x \cdot (y \cdot z) = x$

$x \cdot (y \cdot z)$

$= (x \cdot y) \cdot z$ Associativité

$= x \cdot z$ Par hypothèse

$= x$ Par hypothèse

□

Démonstration de la propriété [39] : $(x + y) \leq z \Leftrightarrow \begin{cases} x \leq z \\ y \leq z \end{cases}$

Par définition :

$$(x + y) \leq z \Leftrightarrow (x + y) \cdot z = x + y$$

$$x \leq z \Leftrightarrow x \cdot z = x$$

$$y \leq z \Leftrightarrow y \cdot z = y$$

- $(x + y) \cdot z = x + y \Rightarrow \begin{cases} x \cdot z = x \\ y \cdot z = y \end{cases}$

$x \cdot z$

$= (x + 0) \cdot z$ Élément neutre

$= (x + (y \cdot \bar{y})) \cdot z$ Élément complémentaire

$= ((x + y) \cdot (x + \bar{y})) \cdot z$ Distributivité

$= ((x + y) \cdot z) \cdot (x + \bar{y})$ Associativité

$= (x + y) \cdot (x + \bar{y})$ Par hypothèse

$= x + (y \cdot \bar{y})$ Distributivité

$= x + 0$ Élément complémentaire

$= x$ Élément neutre

$y \cdot z$

$= (y + 0) \cdot z$	Élément neutre
$= (y + (x \cdot \bar{x})) \cdot z$	Élément complémentaire
$= ((y + x) \cdot (y + \bar{x})) \cdot z$	Distributivité
$= ((y + x) \cdot z) \cdot (y + \bar{x})$	Associativité
$= (y + x) \cdot (y + \bar{x})$	Par hypothèse
$= y + (x \cdot \bar{x})$	Distributivité
$= y + 0$	Élément complémentaire
$= y$	Élément neutre

- $\begin{cases} x \cdot z = x \\ y \cdot z = y \end{cases} \Rightarrow (x + y) \cdot z = x + y$

$(x + y) \cdot z$

$= (x \cdot z) + (y \cdot z)$	Distributivité
$= x + y$	Par hypothèse

□

Démonstration de la propriété [40] : $x \leq y \Rightarrow (x \cdot z) \leq (y \cdot z)$

Par définition :

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$(x \cdot z) \leq (y \cdot z) \Leftrightarrow (x \cdot z) \cdot (y \cdot z) = x \cdot z$$

- $x \cdot y = x \Rightarrow (x \cdot z) \cdot (y \cdot z) = x \cdot z$

$(x \cdot z) \cdot (y \cdot z)$

$= (x \cdot y) \cdot (z \cdot z)$	Distributivité
$= x \cdot (z \cdot z)$	Par hypothèse
$= x \cdot z$	Idempotence

□

Démonstration de la propriété [41] : $x \leq y \Rightarrow (x + z) \leq (y + z)$

Par définition :

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$(x + z) \leq (y + z) \Leftrightarrow (x + z) \cdot (y + z) = x + z$$

- $x \cdot y = x \Rightarrow (x + z) \cdot (y + z) = x + z$

$(x + z) \cdot (y + z)$

$= ((x + z) \cdot y) + ((x + z) \cdot z)$	Distributivité
$= ((x + z) \cdot y) + z$	Élément absorbant
$= ((x \cdot y) + (z \cdot y)) + z$	Distributivité
$= (x \cdot y) + ((z \cdot y) + z)$	Associativité
$= x + ((z \cdot y) + z)$	Par hypothèse
$= x + z$	Élément absorbant

□

Démonstration de la propriété [42] : $\begin{cases} x \leq y \\ z \leq w \end{cases} \Rightarrow (x \cdot z) \leq (y \cdot w)$

Par définition :

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$z \leq w \Leftrightarrow z \cdot w = z$$

$$(x \cdot z) \leq (y \cdot w) \Leftrightarrow (x \cdot z) \cdot (y \cdot w) = x \cdot z$$

- $\begin{cases} x \cdot y = x \\ z \cdot w = z \end{cases} \Rightarrow (x \cdot z) \cdot (y \cdot w) = x \cdot z$

$(x \cdot z) \cdot (y \cdot w)$

$= (x \cdot y) \cdot (z \cdot w)$	Associativité
$= x \cdot z$	Par hypothèse

□

Démonstration de la propriété [43] : $\begin{cases} x \le y \\ z \le w \end{cases} \Rightarrow (x + z) \le (y + w)$

Par définition :

$$x \le y \Leftrightarrow x \cdot y = x$$

$$z \le w \Leftrightarrow z \cdot w = z$$

$$(x + z) \le (y + w) \Leftrightarrow (x + z) \cdot (y + w) = x + z$$

- $\begin{cases} x \cdot y = x \\ z \cdot w = z \end{cases} \Rightarrow (x + z) \cdot (y + w) = x + z$

$(x + z) \cdot (y + w)$

$= (x \cdot (y + w)) + (z \cdot (y + w))$	Distributivité
$= ((x \cdot y) + (x \cdot w)) + ((z \cdot y) + (z \cdot w))$	Distributivité
$= (x + (x \cdot w)) + ((z \cdot y) + z)$	Par hypothèse
$= x + z$	Élément absorbant

□

Annexe B :
Démonstrations des théorèmes

Les démonstrations des théorèmes fondamentaux sont données dans le corps du texte. Cette annexe contient seulement la démonstration de théorèmes annexes donnés au chapitre 2.

Démonstration du théorème 1 : $(\Gamma_n, +, \cdot, \bar{}, \mathcal{F}, \mathcal{T})$ a une structure d'algèbre de Boole.
Pour démontrer que $(\Gamma_n, +, \cdot, \bar{}, \mathcal{F}, \mathcal{T})$ a une structure d'algèbre de Boole, il faut vérifier que la définition des trois lois proposées et les éléments de Γ_n retenus comme éléments particuliers permettent bien d'établir les neuf axiomes qui définissent cette structure.

Pour cette démonstration, nous allons exploiter le fait que $(\mathbb{B}, \vee, \wedge, \neg, 0, 1)$ est une algèbre de Boole. Les huit égalités à établir entre fonctions booléennes seront obtenues en montrant que ces fonctions ont la même valeur pour chacun des n-uplets $(b_1, \ldots, b_n)$. Soit $f, g, h \in \Gamma_n$.

- $f + g = g + f$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f + g)(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \vee g(b_1, \ldots, b_n)$
$= g(b_1, \ldots, b_n) \vee f(b_1, \ldots, b_n)$
$= (g + f)(b_1, \ldots, b_n)$

- $f \cdot g = g \cdot f$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f \cdot g)(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \wedge g(b_1, \ldots, b_n)$
$= g(b_1, \ldots, b_n) \wedge f(b_1, \ldots, b_n)$
$= (g \cdot f)(b_1, \ldots, b_n)$

- $f + (g \cdot h) = (f + g) \cdot (f + h)$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f + (g \cdot h))(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \vee (g \cdot h)(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \vee (g(b_1, \ldots, b_n) \wedge h(b_1, \ldots, b_n))$
$= (f(b_1, \ldots, b_n) \vee g(b_1, \ldots, b_n)) \wedge (f(b_1, \ldots, b_n) \vee h(b_1, \ldots, b_n))$
$= (f + g)(b_1, \ldots, b_n) \wedge (f + h)(b_1, \ldots, b_n)$
$= ((f + g) \cdot (f + h))(b_1, \ldots, b_n)$

- $f \cdot (g + h) = (f \cdot g) + (f \cdot h)$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f \cdot (g + h))(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \wedge (g + h)(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \wedge (g(b_1, \ldots, b_n) \vee h(b_1, \ldots, b_n))$
$= (f(b_1, \ldots, b_n) \wedge g(b_1, \ldots, b_n)) \vee (f(b_1, \ldots, b_n) \wedge h(b_1, \ldots, b_n))$
$= (f \cdot g)(b_1, \ldots, b_n) \vee (f \cdot h)(b_1, \ldots, b_n)$
$= ((f \cdot g) + (f \cdot h))(b_1, \ldots, b_n)$

- $f + \mathcal{F} = f$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f + \mathcal{F})(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \vee \mathcal{F}(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \vee 0$
$= f(b_1, \ldots, b_n)$

- $f \cdot \mathcal{T} = f$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,
$(f \cdot \mathcal{T})(b_1, \ldots, b_n)$
$= f(b_1, \ldots, b_n) \wedge \mathcal{T}(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \wedge 1$

$= f(b_1, \ldots, b_n)$

- $f + \bar{f} = \mathcal{T}$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,

$(f + \bar{f})(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \vee (\bar{f})(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \vee \neg(f(b_1, \ldots, b_n))$

$= 1$

$= \mathcal{T}(b_1, \ldots, b_n)$

- $f \cdot \bar{f} = \mathcal{F}$

$\forall (b_1, \ldots, b_n) \in \mathbb{B}^n$,

$(f \cdot \bar{f})(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \wedge (\bar{f})(b_1, \ldots, b_n)$

$= f(b_1, \ldots, b_n) \wedge \neg(f(b_1, \ldots, b_n))$

$= 0$

$= \mathcal{F}(b_1, \ldots, b_n)$

- $\mathcal{F} \neq \mathcal{T}$

Par définition

Les neuf axiomes étant vérifiés, $(\Gamma_n, +, \cdot, \bar{}, \mathcal{F}, \mathcal{T})$ a bien une structure d'algèbre de Boole.

□

Démonstration théorème 4 : La relation « Inclusion » ($\leq$) est une relation d'ordre partiel. Pour démontrer ce théorème, il suffit de prouver que les trois propriétés suivantes sont vérifiées pour tout élément x, y et z de $\mathcal{B}$:

- Réflexivité : $x \leq x$,
- Antisymétrie : Si $x \leq y$ et $y \leq x$, alors $x = y$,
- Transitivité : Si $x \leq y$ et $y \leq z$, alors $x \leq z$.

- **Réflexivité :** $x \leq x$ ou $x \cdot x = x$

Sachant que :

$x \cdot x = x$ Idempotence

Nous avons bien $x \leq x$.

- **Antisymétrie :** $\begin{cases} x \leq y \\ y \leq x \end{cases} \Rightarrow x = y$

Par définition :

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$y \leq x \Leftrightarrow y \cdot x = y$$

- $\begin{cases} x \cdot y = x \\ y \cdot x = y \end{cases} \Rightarrow x = y$

$x = x \cdot y$ Par hypothèse
$= y \cdot x$ Commutativité
$= y$ Par hypothèse

- **Transitivité :** $\begin{cases} x \leq y \\ y \leq z \end{cases} \Rightarrow x \leq z$

Par définition :

$$x \leq y \Leftrightarrow x \cdot y = x$$

$$y \leq z \Leftrightarrow y \cdot z = y$$

$$x \leq z \Leftrightarrow x \cdot z = x$$

- $\begin{cases} x \cdot y = x \\ y \cdot z = y \end{cases} \Rightarrow x \cdot z = x$

$x \cdot z$
$= (x \cdot y) \cdot z$ Par hypothèse
$= x \cdot (y \cdot z)$ Associativité
$= x \cdot y$ Par hypothèse
$= x$ Par hypothèse

La relation « Inclusion » $\leq$ étant une relation réflexible, antisymétrique et transitive, cette relation est bien une relation d'ordre partiel.

□

Démonstration du théorème 5 : Les cinq formes suivantes pour l'égalité sont équivalentes pour tout $x, y \in \mathcal{B}$:

$$\begin{array}{lll} x = y & x \cdot \bar{y} + \bar{x} \cdot y = 0 & \begin{cases} x \cdot \bar{y} = 0 \\ \bar{x} \cdot y = 0 \end{cases} \\ \bar{x} = \bar{y} & x \cdot y + \bar{x} \cdot \bar{y} = 1 & \end{array}$$

Pour démontrer l'équivalence entre ces cinq formes, il suffit de montrer l'équivalence entre la première forme et chacune des autres.

- $x = y \Leftrightarrow x \cdot \bar{y} + \bar{x} \cdot y = 0$

$x = y$

$\Leftrightarrow x \cdot \bar{y} + \bar{x} \cdot y = 0$ propriété [31]

- $x = y \Leftrightarrow \begin{cases} x \cdot \bar{y} = 0 \\ \bar{x} \cdot y = 0 \end{cases}$

$x = y$

$\Leftrightarrow x \cdot \bar{y} + \bar{x} \cdot y = 0$ propriété [31]

$\Leftrightarrow \begin{cases} x \cdot \bar{y} = 0 \\ \bar{x} \cdot y = 0 \end{cases}$ propriété [32]

- $x = y \Leftrightarrow \bar{x} = \bar{y}$

$x = y$

$\Leftrightarrow \bar{x} = \bar{y}$ propriété [33]

- $x = y \Leftrightarrow x \cdot y + \bar{x} \cdot \bar{y} = 1$

$x = y$

$\Leftrightarrow x \cdot \bar{y} + \bar{x} \cdot y = 0$ propriété [31]

$\Leftrightarrow \overline{(x \cdot \bar{y} + \bar{x} \cdot y)} = \bar{0}$ propriété [33]

$\Leftrightarrow \overline{(x \cdot \bar{y})} \cdot \overline{(\bar{x} \cdot y)} = 1$ Théorème de De Morgan et propriété [26]

$\Leftrightarrow (\bar{x} + \overline{(\bar{y})}) \cdot (\overline{(\bar{x})} + \bar{y}) = 1$ Théorème de De Morgan

$\Leftrightarrow (\bar{x} + y) \cdot (x + \bar{y}) = 1$ Théorème du double complément
$\Leftrightarrow \bar{x} \cdot x + \bar{x} \cdot \bar{y} + y \cdot x + y \cdot \bar{y} = 1$ Distributivité
$\Leftrightarrow \bar{x} \cdot \bar{y} + y \cdot x = 1$ Élément complémentaire et Élément neutre

□

Démonstration du théorème 6 : Les huit formes suivantes pour l'inclusion sont équivalentes pour tout $x, y \in \mathcal{B}$:

$$x \leq y \qquad x \cdot y = x \qquad \bar{x} + \bar{y} = \bar{x} \qquad x \cdot \bar{y} = 0$$
$$\bar{y} \leq \bar{x} \qquad \bar{x} \cdot \bar{y} = \bar{y} \qquad x + y = y \qquad \bar{x} + y = 1$$

Pour démontrer l'équivalence entre ces huit formes, il suffit de montrer l'équivalence entre la première forme et chacune des autres.

- $x \leq y \Leftrightarrow x \cdot y = x$

$x \leq y$
$\Leftrightarrow x \cdot y = x$ Par définition

- $x \leq y \Leftrightarrow \bar{x} + \bar{y} = \bar{x}$

$x \leq y$
$\Leftrightarrow x \cdot y = x$ Par définition
$\Leftrightarrow \overline{(x \cdot y)} = \bar{x}$ propriété [33]
$\Leftrightarrow \bar{x} + \bar{y} = \bar{x}$ Théorème de De Morgan

- $x \leq y \Leftrightarrow x \cdot \bar{y} = 0$

$x \leq y$
$\Leftrightarrow x \cdot y = x$ Par définition
$\Leftrightarrow x \cdot \bar{y} = 0$ propriété [29]

- $x \leq y \Leftrightarrow \bar{y} \leq \bar{x}$

$x \leq y$
$\Leftrightarrow x \cdot \bar{y} = 0$ Démontré précédemment
$\Leftrightarrow \bar{y} \cdot \overline{(\bar{x})} = 0$ Théorème du double complément
$\Leftrightarrow \bar{y} \leq \bar{x}$ Démontré précédemment

- $x \leq y \Leftrightarrow \bar{x} \cdot \bar{y} = \bar{y}$

$x \leq y$

$\Leftrightarrow \bar{y} \leq \bar{x}$ Démontré précédemment

$\Leftrightarrow \bar{x} \cdot \bar{y} = \bar{y}$ Par définition

- $x \leq y \Leftrightarrow x + y = y$

$x \leq y$

$\Leftrightarrow \bar{x} \cdot \bar{y} = \bar{y}$ Démontré précédemment

$\Leftrightarrow \overline{(\bar{x} \cdot \bar{y})} = \overline{(\bar{y})}$ propriété [33]

$\Leftrightarrow \bar{\bar{x}} + \bar{\bar{y}} = \bar{\bar{y}}$ Théorème de De Morgan

$\Leftrightarrow x + y = y$ Théorème du double complément

- $x \leq y \Leftrightarrow \bar{x} + y = 1$

$x \leq y$

$\Leftrightarrow x \cdot \bar{y} = 0$ Démontré précédemment

$\Leftrightarrow \overline{(x \cdot \bar{y})} = \bar{0}$ propriété [33]

$\Leftrightarrow \bar{x} + \bar{\bar{y}} = \bar{0}$ Théorème de De Morgan

$\Leftrightarrow \bar{x} + y = \bar{0}$ Théorème du double complément

$\Leftrightarrow \bar{x} + y = 1$ propriété [26]

□

Résumé : Les travaux présentés dans ce mémoire sont relatifs à l'élaboration formelle de la commande d'un Système à Evènements Discrets (SED) logique à partir des exigences exprimées dans le cahier des charges. La méthode proposée est basée sur la résolution de manière littérale d'un système d'équations représentant ces exigences.
Le cadre mathématique, support de ces travaux, est l'algèbre de Boole des fonctions booléennes. Ce cadre mathématique a été retenu pour les raisons suivantes :
- Dans le cas particulier des SED logiques non temporisés, toute loi de commande peut être décrite à l'aide de fonctions booléennes.
- Les exigences exposées dans un cahier des charges peuvent être formalisées sous forme de relations entre des fonctions booléennes.
- Les résultats obtenus dans le cadre de cette thèse nous permettent de déterminer automatiquement quelles sont les fonctions booléennes qui satisfont le système d'équations entre fonctions booléennes représentant ces exigences.
La méthode proposée permet au concepteur d'exprimer les exigences dans des formalismes différents. Il a également la possibilité de fixer la forme de la solution qu'il souhaite obtenir ou de ne réaliser la synthèse que sur une partie du modèle.
Le chapitre 2 de ce mémoire est consacré à la présentation des résultats mathématiques que nous avons établis pour pouvoir résoudre un système d'équations à n inconnues dans toute structure d'algèbre de Boole.
L'approche de synthèse est détaillée au chapitre 3 au travers du traitement de 3 exemples de taille et de complexité croissantes. Nous montrons comment les exigences exprimées dans un cahier des charges peuvent être formalisées sous forme de relations entre des fonctions booléennes. La résolution du système d'équations est réalisée automatiquement grâce à une maquette informatique développée au LURPA.

Mots-clés : synthèse algébrique, système à évènements discrets logique, résolution d'équations sur une algèbre de Boole

Abstract : The work presented in this memory is relating to the formal development of the control of a logical Discrete Event System (DES) starting from the requirements expressed in the specifications. The method suggested is based on the literal solving of a system of equations representing these requirements.
The mathematical framework, support of this work, is the Boolean algebra of the Boolean functions. This mathematical framework was retained for the following reasons:
- In the particular case of the non temporal logical SED, any control law can be described using Boolean functions.
- The requirements exposed in specifications can be formalized in the form of relations between Boolean functions.
- The results obtained in this thesis enable us to determine automatically which are the Boolean functions which satisfy the system of equations between Boolean functions representing these requirements.
The method suggested allows the designer to express the requirements in different formalisms. It also has the possibility of fixing the form of the solution which he wishes to obtain or making the synthesis only on a part of the model.
Chapter 2 of this memory is about the presentation of the mathematical results which we established to be able to solve a system of equations with n unknowns in any structure of Boolean algebra.
The synthesis approach is detailed in chapter 3 through the treatment of 3 examples of increasing size and complexity. We show how the requirements expressed in specifications can be formalized in relations between Boolean functions. The solving of the system of equations is automatically realized with a experimental module developed in the LURPA.

Keywords : algebraic synthesis, logical discrete event system, equations solving on a Boolean algebra