Analyse de Grafcets par Génération Logique de l'Automate Équivalent

Jean-Marc Roussel

DOCTORAT DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

SPÉCIALITÉ : AUTOMATIQUE

THÈSE

PRÉSENTÉE PAR

Jean-Marc ROUSSEL

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Analyse de Grafcets par Génération Logique de l'Automate Équivalent

---

Soutenue le 16 décembre 1994 devant le jury composé de :

J.P. FRACHET	Président
F. PRUNET	Rapporteur
P. LHOSTE	Rapporteur
P. BOURDET	Directeur de thèse
J.J. LESAGE	Directeur des travaux
O. DOUCHIN	Examinateur

Laboratoire Universitaire de Recherche en Production Automatisée

École Normale Supérieure de CACHAN

61, Avenue du Président Wilson - 94235 CACHAN Cedex

à Isabelle
à notre enfant qu'elle porte
à nos parents, frères et sœur

Avant-propos

Le travail présenté dans ce mémoire a été effectué au sein du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA - EA 1385) de l'École Normale Supérieure de Cachan. Il a été dirigé par Monsieur Jean-Jacques LESAGE, Professeur à l'ENS de Cachan, sous la responsabilité scientifique de Monsieur Pierre BOURDET, Professeur à l'ENS de Cachan.

Que le Professeur Pierre BOURDET, trouve ici toute ma reconnaissance pour m'avoir accueilli dans le laboratoire qu'il dirige, et pour m'avoir accordé sa confiance.

Je tiens à remercier tout spécialement Monsieur Jean-Jacques LESAGE à la fois pour ses conseils avisés, sa disponibilité, et son soutien dans ce travail, mais également pour son enthousiasme à la recherche qu'il a su me communiquer.

Je suis particulièrement reconnaissant à Monsieur Jean-Paul FRACHET, Professeur de l'Institut Supérieur des Matériaux et de la Construction Mécanique (ISMCM) de TOULON, de me faire l'honneur de présider le jury.

Les rapports auprès de la formation doctorale ont été établis par Monsieur François PRUNET, Professeur à l'Université de Montpellier II et Monsieur Pascal LHOSTE, Maître de conférences - Habilité à diriger les recherches de l'Université de Nancy I. Je les remercie vivement d'avoir accepté cette tâche et leur sais gré pour les conseils qu'il m'ont apportés pour la rédaction de ce mémoire.

Je remercie tout particulièrement Monsieur Olivier DOUCHIN, Docteur de l'Université de Nancy I - directeur technique de la société FAMIC - EURILOR. Par sa

participation à ce jury, il témoigne de l'intérêt porté par le milieu industriel à nos travaux.

Que tous les membres du groupe GRAFCET de l'AFCET soient assurés de ma profonde gratitude pour m'avoir aidé, par leurs questions et surtout par les réponses à mes questions, à mener à bien ces travaux.

Je ne pourrais terminer sans remercier tous les membres du laboratoire pour leur soutien ainsi que leur contribution directe ou indirecte à ce mémoire. Je pense plus particulièrement à Bruno, Guy, Christophe(s), Olivier, Emmanuel et Loïc mais je n'oublie pas les autres.

Table des matières

Chapitre 2

Compléments théoriques

concernant le modèle GRAFCET

Chapitre 3

Construction d'une algèbre de Boole pour l'approche événementielle en GRAFCET

Chapitre 4

Prise en compte des entrées et calcul symbolique dans $\mathbb{I}$

Chapitre 5

La génération de l'automate équivalent

Chapitre 6

Validation de grafcets

par analyse de l'automate équivalent

Chapitre 7

Exemples de validation de grafcets

Annexe A

Illustration à l'aide de chronogrammes
des 14 propriétés établies au chapitre 3

Annexe B

Calcul symbolique sur $\mathbb{I}$
avec contention des entrées

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

Chapitre 7

Annexe A

Liste des tableaux

Chapitre 4

Chapitre 5

Chapitre 7

Annexe B

Liste des définitions, propriétés & théorèmes

Chapitre 4

Chapitre 5

Chapitre 6

Introduction

Ce mémoire présente quelques éléments de réponse à l'une des questions essentielles auxquelles est confronté le génie automatique : *comment s'assurer que le système automatisé qu'on est en train de concevoir sera en tout point conforme aux exigences exposées dans le cahier des charges ?*

Notre étude porte exclusivement sur la validation des modélisations écrites en GRAFCET[1] [IEC 848], et plus particulièrement sur la validation de modèles de spécification[2]. Nous nous sommes intéressés au GRAFCET car il s'agit certainement du modèle le plus usité industriellement pour la spécification de la dynamique des systèmes à événements discrets, mais paradoxalement celui pour lequel il a été effectué le moins de travaux scientifiques relatifs à la *validation des modèles*. Les travaux dont les résultats sont exposés dans ce mémoire ont donc pour objectif de *procurer un cadre formel et des techniques opérationnelles pour la validation d'un grafcet de spécification*.

Pour valider tout grafcet, nous utilisons sa représentation par *graphe des situations accessibles* (qui est un automate à états) dans laquelle toutes les possibilités d'évolutions, structurelles ou interprétées, ont été identifiées et rendues explicites.

---

1. Dans ce mémoire, le mot GRAFCET sera écrit en majuscules lorsque nous parlerons de la technique de modélisation et en minuscules lorsque nous parlerons du résultat d'une modélisation.

2. Sont donc exclus de cette étude les modèles de conception détaillée ou d'implantation en Automate Programmable Industriel, écrits dans divers langages inspirés du GRAFCET.

Notre approche repose sur le fait que le GRAFCET est une machine d'état pour laquelle il existe différentes représentations : l'une d'entre elles est la donnée de l'automate à états qui est intrinsèquement contenu dans le modèle. Nous proposons donc d'extraire du grafcet l'automate à états pour l'utiliser lors de la validation.

Pour mettre en place cette technique, il nous a fallu au préalable renforcer les fondements théoriques du GRAFCET (qui sur certains points sont insuffisamment formalisés), identifier un modèle de comportement des entrées qui soit conforme aux hypothèses du GRAFCET et définir un langage rigoureux pour l'expression des propriétés à valider sur les grafcets.

De manière à ce que ces travaux ne constituent pas seulement une étude théorique de faisabilité, nous avons tenu à les valider par le développement d'une maquette informatique expérimentale intitulée «AGGLAÉ» (Analyse de Grafcets par Génération Logique de l'Automate Équivalent). Ce maquettage nous a permis de montrer la praticabilité de nos résultats théoriques et nous a conduit à identifier par l'expérimentation un certain nombre de problèmes qu'une seule étude théorique n'aurait pas su révéler.

Ce mémoire se compose de sept chapitres :

- Le premier chapitre propose un état de l'art de la validation de spécifications écrites en GRAFCET et une analyse des besoins des utilisateurs. Nous énonçons ensuite de manière synthétique notre démarche.
- Le second chapitre présente succinctement les travaux qui ont conduit à introduire, au sein du modèle GRAFCET, la notion de double échelle de temps et précise la notion de forçage de situations. Nous exposons également quelques ambiguïtés qui subsistent dans le modèle pour lesquelles nous proposons des solutions.
- Le troisième chapitre expose une formalisation de la notion d'événements par la définition d'une algèbre de Boole étendue dans laquelle les fronts sont définis par des opérateurs unaires. Quatorze propriétés relatives au développe-

ment des opérateurs fronts par rapport au opérateurs ET, OU, NON sont ensuite démontrées.

- Le quatrième chapitre présente comment nous prenons en compte les entrées du modèle ainsi qu'un module de calcul symbolique sur des expressions décrivant des ensembles de variations des entrées.
- Le cinquième chapitre expose la technique de génération automatique de l'automate équivalent à tout grafcet en tenant compte de l'historique des entrées.
- Le sixième chapitre présente les différentes possibilités d'analyse de l'automate équivalent dans le but de valider une description GRAFCET.
- Le septième chapitre traite de deux exemples illustratifs de notre démarche. Il s'agit du «problème des philosophes» - problème classique de l'informatique - et d'un exemple plus industriel relatif à la «gestion d'un système de pompage».

Chapitre 1
Le problème et son positionnement

1.1. Validation de spécifications écrites en GRAFCET : état de l'art

1.1.1. Un bref historique

Depuis la création du GRAFCET en 1977, la validation des modèles a fait l'objet de relativement peu de travaux universitaires ou industriels. En 1979 déjà, Michel Blanchard, animateur du groupe qui créa le GRAFCET, consacra pourtant le septième chapitre de son livre, au titre évocateur «comprendre, maîtriser et appliquer le grafcet» [Blanchard 79], à l'analyse et à la validation des modèles. Dans ce chapitre, il présente une méthode pour la validation d'un cahier des charges basée sur une exploitation du **graphe des situations accessibles** d'un grafcet. Toute la puissance de modélisation, mais aussi toute la complexité du GRAFCET en tant que machine d'états, sont très bien décrites dans ce chapitre. L'auteur y développe les fondements de la validation de grafcets en définissant les notions de **situation atteinte** et de **situation accessible**. Dans l'impossibilité de connaître avec certitude les situations atteintes par un grafcet (faute de connaître le comportement du processus commandé), c'est bien en analysant l'ensemble de ses situations accessibles que l'on peut valider un modèle. L'apport essentiel de nos travaux, qui s'inscrivent parfaitement dans la continuité de ceux de M. Blanchard, est d'avoir **systématisé** et **formalisé** la détermination des situations accessibles d'un grafcet.

C'est tout probablement pour contourner cette difficulté majeure qu'est la recherche des situations accessibles, que les recherches ayant trait à la validation de grafcets ont ensuite essentiellement porté sur la **traduction** des grafcets dans des langages ou formalismes dotés de capacité à établir des preuves. Voici une chronologie des quelques travaux significatifs dans ce domaine, un développement de chacune de ces approches sera proposé dans la section suivante de manière à positionner plus précisément l'apport de nos travaux.

En juillet 1981, Mohamed Moalla proposa dans sa thèse d'état [Moalla 81], une méthode dans laquelle le GRAFCET est utilisé comme outil de spécification du cahier des charges, et les Réseaux de Petri Interprétés (RdPI) comme outil de conception. La validation des spécifications du cahier des charges se faisant par la vérification de propriétés sur les RdPI.

Lors du congrès GRAFCET'92, de nouveaux travaux relatifs à la validation de grafcets sont présentés. Dans [Aygalinc 92], les auteurs proposent de vérifier si le modèle traduit fidèlement les spécifications fonctionnelles du cahier des charges en étudiant le RdP autonome traduisant un grafcet. Dans [André 92], il est présenté une analyse comparative entre le GRAFCET et les langages synchrones. Les auteurs proposent d'utiliser les techniques de validation sur les automates, après une traduction d'un grafcet en langage synchrone ESTEREL.

En janvier 1994, Philippe Le Parc consacre un chapitre de son doctorat [Le Parc 94] à la validation des grafcets. Il propose deux techniques de preuve qui nécessitent la traduction du grafcet en systèmes de transitions pour la première et en équations SIGNAL pour la seconde. De manière similaire, dans [Roux 94] les auteurs proposent de traduire le GRAFCET en ELECTRE, un langage asynchrone cette fois, pour permettre sa validation.

Quant aux travaux industriels dans ce domaine, ils ont essentiellement porté sur la validation des grafcets par simulation. Bien que moins rigoureuse, puisqu'imposant à l'analyste de se placer dans les conditions de fonctionnement du processus commandé, cette approche est plus directement opérationnelle et peut donc facilement être implantée dans tout environnement orienté GRAFCET.

Nous allons maintenant revenir dans le détail sur chacune des approches que nous venons de citer.

1.1.2. Approche par simulation

Dans de nombreuses méthodes de spécification ou de conception utilisant le GRAFCET, il est préconisé d'utiliser la simulation pour la validation des modèles [Deneux 83] [Prunet 87] [Muller 89]. Les logiciels supports de ces méthodes proposent des modules de simulation dans lesquels des procédures ont été développées pour faciliter le dépouillement des résultats (simulation pas à pas, utilisation de fichier trace, possibilités de retour en arrière, ...) [Panetto 91] [Giaccone 91] [Chapurlat 93] [3IP 88].

L'apport de la simulation pour la validation de grafcets est certain. En effet, le simulateur dispensant l'analyste de tous les calculs d'expressions combinatoires et des calculs d'évolution, celui-ci peut concentrer son attention sur la conséquence de la variation des entrées qu'il a proposée sur l'évolution du grafcet simulé. Cependant, nous considérons que la simulation ne permet pas réellement de valider un modèle grafcet. Pour que le comportement d'un modèle puisse être éprouvé par simulation, il est en effet nécessaire de simuler l'ensemble des séquences possibles de variations des entrées. Une telle simulation «exhaustive» n'est bien entendu pas possible car pour un grafcet de n entrées et de p étapes, il y a en théorie jusqu'à $n \times 2^{n+p}$ simulations à effectuer ![1]

Ne pouvant être exhaustive, la qualité d'une validation par simulation dépend donc exclusivement de la pertinence des scénarios simulés par l'analyste. Cependant, comme l'analyste n'est pas assisté pour la détermination des scénarios, ce sont souvent ceux qu'il avait à l'esprit lors de l'élaboration des grafcets, qui sont simulés. Le résultat d'un telle simulation est donc plus la vérification de la fidélité du modèle à la pensée de son auteur que la vérification de la fidélité du modèle au cahier des charges. De plus, comme il est rappelé dans [Corbier 87], pour être performante du point de vue de la validation, toute simulation devrait être effectuée par une personne différente de l'auteur des grafcets car lorsqu'elle est faite avec un seul point de vue l'analyste a plutôt tendance à tester ce qu'il a fait et non ce que le système doit faire.

---

1. En effet, pour n entrées il existe $n.2^n$ évolutions possibles pour les entrées (depuis chacune des 2^n valeurs de l'ensemble des entrées, seuls n évolutions sont possibles puisque deux entrées ne peuvent changer d'état simultanément). Comme chaque évolution de l'ensemble des entrées doit être envisagée pour chaque situation possible du grafcet (au maximum 2^p situations), le nombre maximum de simulations à effectuer est bien de $n.2^{n+p}$.

En conclusion, nous retiendrons avec M. Blanchard que *«cette approche est très difficile à mettre en œuvre et qu'on peut douter de la valeur de la validation qui en découle»*. (Extrait de [Blanchard 79] page 134).

1.1.3. Approche par traduction des grafcets en RdP [Moalla 81] [Aygalinc 92]

En raison des différences[1] notables de capacité de modélisation entre le GRAFCET et les RdPI, la traduction directe d'un grafcet quelconque en un RdPI n'est pas possible. Pour éviter que cette approche ne permette que la validation d'une certaine classe de grafcets (celle regroupant les grafcets traduisibles en RdPI), M. Moalla a établi que *«tout fonctionnement décrit par un grafcet peut avoir une représentation grafcet équivalente dans laquelle :*

- *une étape n'est activée qu'au plus une fois à chaque franchissement de transitions et n'est jamais réactivée tant qu'elle est active,*
- *une étape active ne participe qu'au franchissement d'une seule transition à la fois,*
- *aucun test à 1 ou à 0 n'est exprimé à l'aide des variables internes.»*

C'est ce grafcet équivalent qui est ensuite traduit en RdPI dont les propriétés sont vérifiées par les techniques classiques d'analyse des RdP.

Pour obtenir la représentation équivalente, il propose ([Moalla 81] Chapitre 7 page 11) une méthode dont le passage obligé est la génération du graphe des situations accessibles du grafcet initial. Or, ce graphe des situations accessibles est lui même porteur de tous les comportements et de toutes les propriétés du grafcet originel. Nous pensons donc que la traduction en RdPI est une opération inutile pour la validation dans la mesure où nous nous rendons capables de prouver les propriétés sur le graphe des situations accessibles lui même. D'autre part, la systématisation de la génération du graphe des situations accessibles est sans aucun doute la phase la plus délicate de cette méthode, pour laquelle aucune technique n'est proposée dans ces travaux.

---

1. Ces différences sont présentées dans [Moalla 81]. Elles concernent principalement le franchissement des transitions, la nature du marquage et les possibilités de tester les variables internes.

Dans [Aygalinc 92], les auteurs proposent quant à eux d'analyser un grafcet à partir de sa traduction en un RdP autonome en s'affranchissant de tous les problèmes induits par le franchissement simultané, puisqu'ils rejettent les possibilités de parallélisme interprété du GRAFCET. Après une traduction dans laquelle le contenu des réceptivités ne peut pas être pris en compte (le degré de finesse des RdP autonomes étant insuffisant), ils proposent de vérifier que le RdP construit est sauf et vivace. Notons que si le RdP ne s'avère pas sauf, aucun résultat établi sur celui-ci ne peut être pris en compte pour la validation car il existe une différence de comportement notable entre le franchissement d'une transition d'un grafcet et le tir d'une transition RdP lorsque le marquage n'est pas unitaire (cf. figure 1).

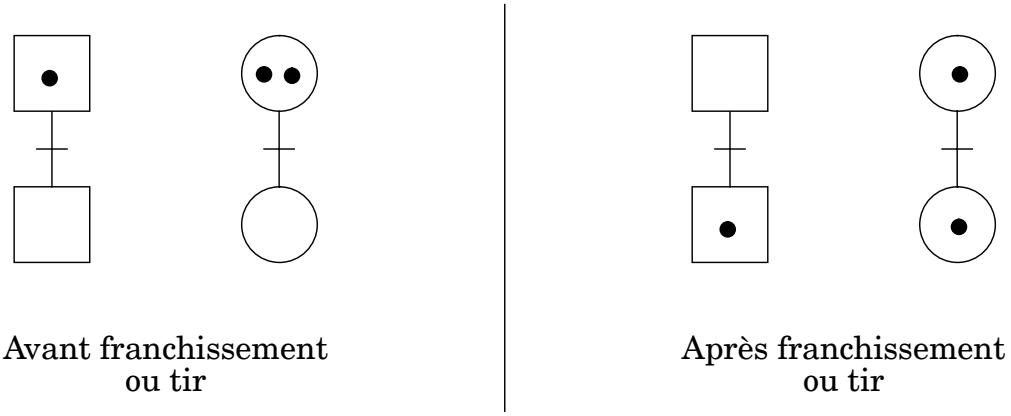


figure 1. Différence de comportement sur franchissement entre un grafcet et un RdP non sauf

D'autre part, la vivacité du RdP autonome ne permet pas de conclure sur l'absence de blocage pour le grafcet. En effet, pour un grafcet le franchissement d'une transition ne dépend pas seulement de sa validation mais également de la véracité de la réceptivité associée.

Compte tenu des hypothèses restrictives sous-tendant ces travaux, il est évident qu'ils ne sauraient apporter des solutions satisfaisantes à la validation de grafcets complexes.

1.1.4. Approche par traduction de grafcets en systèmes de transitions [Le Parc 94]

Une technique de validation de grafcets présentée dans [Le Parc 94] est basée sur l'utilisation des systèmes de transitions, mis au point par A. Arnold et M. Nivat

[Arnold 92a]. P. Le Parc propose pour cela d'établir le graphe d'état équivalent au grafcet (donc le graphe des situations accessibles) par construction d'un système de transitions (ST). Pour ce faire, il traduit chaque structure élémentaire d'un grafcet (une transition avec ses étapes amont et aval) en un «système de transition atomique» (une transition GRAFCET est alors représentée par un ST à deux transitions). A la suite de cette traduction, l'auteur construit un système de transitions complet en effectuant de manière itérative le produit de synchronisation entre les ST traduisant les transitions validées du grafcet étudié (cf. figure 2).

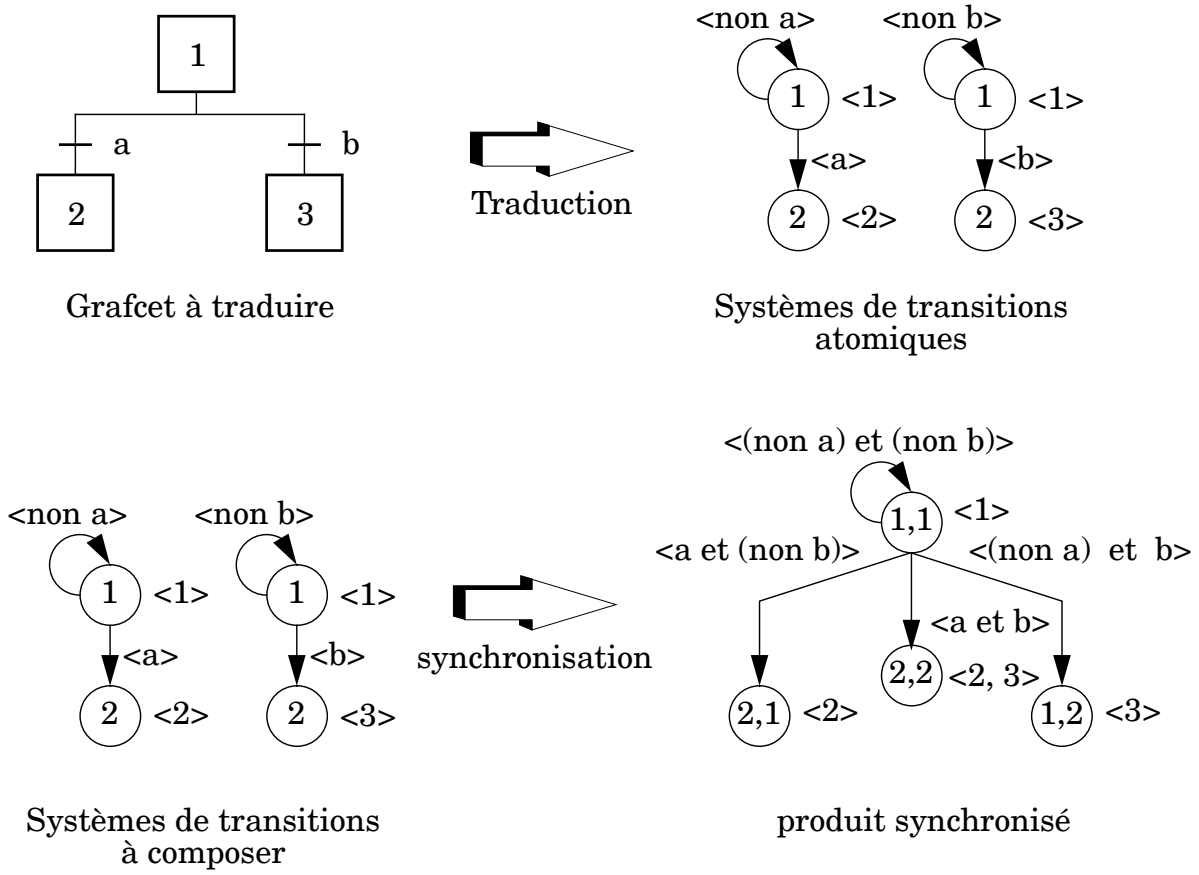


figure 2. Approche par traduction en systèmes de transitions [Le Parc 94]

Lors de cette traduction, P. Le Parc est obligé de représenter la réceptivité associée à la transition d'un grafcet par l'étiquette associée à une transition d'un ST. Cependant, d'un point de vue formel cette association n'est pas cohérente car il ne s'agit pas d'éléments de même nature : une réceptivité est une condition logique exprimée dans l'Algèbre de Boole tandis qu'une étiquette est une lettre d'un alphabet représentant une action ou un événement [Arnold 90] et ignorant les propriétés de

l'Algèbre de Boole. Si une lettre d'un alphabet peut être toujours représentée par une condition logique, la réciproque n'est pas vraie puisqu'une condition logique peut représenter plusieurs lettres.

Pour pallier l'absence du concept de réceptivité dans les systèmes de transitions, l'auteur est alors obligé de reproduire cette notion essentielle du GRAFCET lors des produits de synchronisation en choisissant quels sont les ST composants à synchroniser et en fusionnant les étiquettes associées. La faisabilité de l'approche repose essentiellement sur la redéfinition habile du produit de synchronisation.

De cette approche, nous retiendrons les idées présentées pour l'analyse des graphes construits et les nombreuses possibilités offertes par un outil comme MEC [Crubillé 89] conçu pour la vérification des systèmes de transitions (ces possibilités seront du reste largement exploitées dans le chapitre 6).

1.1.5. Analyse de l'approche par traduction en langages synchrones [André 92] [André 94a] [André 94b] [Le Parc 94]

Une seconde technique de validation de grafcets présentée dans [Le Parc 94] passe par une traduction du grafcet en un jeu d'équations SIGNAL proche de l'interprétation algébrique proposée dans [Afcet 83] [Toulotte 81] [Lhoste 94]. A ces équations, qui traduisent le grafcet à étudier, il est possible d'associer un nouvel ensemble d'équations qui traduit les propriétés que l'on souhaite vérifier. Ces équations, encodées sur le corps des entiers relatifs modulo 3 (Z/3Z), sont ensuite analysées par le système de calcul SIGALI [Dutertre 93].

Dans cette méthode, le «graphe des situations accessibles» est calculé en même temps que la propriété est vérifiée. Pour la validation d'un grafcet, cette simultanéité s'avère très pénalisante car elle oblige l'analyste à identifier préalablement toutes les propriétés qu'il souhaite vérifier. Dans ce cas, tout affinage de propriétés est impossible sans nécessiter un nouveau calcul.

C. André propose quant à lui d'utiliser une traduction en langage ESTEREL pour l'analyse de grafcets. Le code ESTEREL est ensuite compilé pour obtenir un automate au format OC [André 93] qui sera analysé. Pour ce faire, il associe à chaque

étape d'un grafcet un module ESTEREL interagissant avec son milieu extérieur par trois signaux d'entrées et quatre signaux de sorties. L'automate à états fini qui caractérise ce module se compose quant à lui de 3 états et 11 transitions [André 94a]. Une fois que chaque étape est traduite, l'ensemble est ensuite confié à un compilateur qui rejette le programme si le grafcet présente des boucles d'instabilité. Actuellement des travaux sont en cours pour compiler directement les grafcets en un automate au format OC afin d'éviter la traduction intermédiaire en ESTEREL.

Il convient de souligner que dans le cadre de ces travaux, les auteurs ne retiennent pas le modèle GRAFCET normalisé [CEI 88] [UTE 93] mais définissent leur propre interprétation des règles d'évolution [André 94b].

Notre approche ayant au contraire comme objectif la validation de tout modèle conforme à la norme (et à ses extensions), nous ne reviendrons plus sur cette approche dans la suite de ce mémoire.

1.1.6. Approche par traduction de grafcets en langage asynchrone [Roux 94]

O. Roux et V. Rusu proposent de déterminer des propriétés d'un grafcet en analysant sa traduction en ELECTRE [Roux 92]. Pour cela, ils proposent de représenter chaque étape par une structure de contrôle à deux «états» (actif et inactif) et chaque transition par une structure de contrôle à trois «états» (non_validé, validé, franchir). Les différentes structures de contrôle sont synchronisées entre elles à l'aide d'événements ELECTRE émis par les différents «états». Cependant, le respect des règles d'évolution [CEI 88] du grafcet pour tous les cas de figure n'est pas assuré.

La règle concernant l'évolution des étapes actives (dite règle 3) n'est pas respectée car la modélisation proposée pour une transition prévoit l'émission de deux événements temporellement distincts, le premier provoquant la désactivation des étapes en amont et le second l'activation des étapes en aval.

La règle concernant les franchissements simultanés de transitions (dite règle 4) n'est pas toujours respectée car la modélisation proposée pour une transition permet que les structures de contrôle de deux transitions simultanément franchissables ne soient pas dans le même état au moment de ce franchissement. Pour le grafcet proposé figure 3, à l'activation de l'étape 12, la structure de contrôle modélisant la transition t_1 est dans l'état «validé» alors que celle modélisant t_2 est encore dans l'état

«non_validé», elles ne pourront donc pas passer dans l'état «franchir» simultanément.

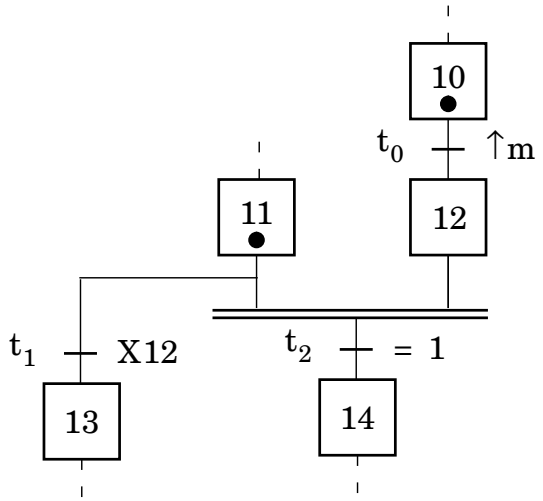


figure 3. Grafcet possédant deux transitions simultanément franchissables (t_1 et t_2) mais non franchies simultanément après traduction en ELECTRE.

La règle concernant l'activation et la désactivation simultanée d'une étape (dite règle 5) n'est pas respectée car en cas d'activation et de désactivation simultanées, la structure de contrôle représentant l'étape reçoit successivement les événements provoquant sa désactivation, puis son activation.

Nous pensons que ces problèmes de traduction ne pourront malheureusement pas être résolus rigoureusement car dans cette approche les auteurs cherchent à modéliser des éléments au comportement *synchrone* (les transitions peuvent être simultanément franchissables, les activations et désactivations d'étapes sont simultanés) à l'aide d'un langage *asynchrone* qui rejette la simultanéité de deux événements.

1.1.7. Approche par génération du graphe des situations accessibles [Blanchard 79]

La méthode présentée par M. Blanchard pour la validation d'un grafcet s'articule autour du graphe des situations accessibles. Pour cela, il propose la méthode suivante :

- établissement du grafcet autonome,
- recherche de l'ensemble des situations accessibles pour le grafcet autonome,
- validation sur le graphe des situations accessibles.

Rappelons que le **grafcet autonome** est obtenu en remplaçant la partie de chaque réceptivité portant sur les variables externes par le symbole ∅ (signifiant indifféremment 0 ou 1). A la suite de cette substitution, le grafcet autonome ne contient plus que des informations relatives aux variables internes et constitue le «noyau du grafcet» indépendant de son interprétation externe. M. blanchard introduit également la notion de **transition sensibilisée** (transition validée pour laquelle la partie de la réceptivité portant sur les variables internes est vraie).

Pour déterminer l'ensemble des situations accessibles pour le grafcet autonome, l'auteur propose une méthode informelle de construction «de proche en proche». Conscient de l'explosion combinatoire de cette approche, M. Blanchard indique page 135 : *«L'accroissement du nombre de situations accessibles est généralement très important si on ne possède pas de renseignements sur l'interprétation externe des transitions solidaires. Les renseignements sur l'interprétation externe des transitions concurrentes permettent souvent de supprimer certaines possibilités d'évolution entre situations sans en diminuer le nombre ; seule une interprétation externe des transitions solidaires permet de réduire l'ensemble des situations accessibles».*
Tout en précisant que *«les renseignements apportés vont réduire l'ensemble des situations accessibles. Tout renseignement inexact va donc exclure de l'analyse des situations qui en réalité peuvent être atteinte. C'est pourquoi il est dangereux de renseigner le grafcet autonome avec des informations incertaines telle que l'ordre chronologique des occurrences des événements externes, les incompatibilités physiques,... En général, on se limitera à renseigner le grafcet autonome sur les relations logiques entre réceptivités externes qu'associe l'interprétation externe aux transitions simultanément sensibilisées».*

Nous retiendrons essentiellement trois enseignements de l'approche de M. Blanchard :

- *le graphe des situations accessibles est porteur de toutes les propriétés d'un grafcet* car il comprend toutes les évolutions possibles, quelque soit le comportement du processus commandé,
- la génération de ce graphe est difficile (voire impossible manuellement) et se heurte rapidement à une *explosion combinatoire* du nombre d'évolutions à prendre en compte,

- l'une des clés essentielles à la génération rigoureuse (et habile !) du graphe des situations accessibles est la *prise en compte de l'histoire des entrées* du modèle.

Après ce rapide tour d'horizon des (trop) rares travaux scientifiques ayant porté sur la validation de grafcets, nous concluons que dans ce domaine beaucoup reste à faire et qu'une voie prometteuse est sans aucun doute l'analyse directe du graphe des situations accessibles d'un grafcet.

Avant de présenter notre contribution à la résolution des problèmes soulevés, nous allons maintenant examiner quelques attentes et pratiques courantes des utilisateurs finaux, telles que nous avons pu les percevoir dans la littérature, mais aussi - et surtout - dans les contacts que nous avons établis avec de nombreux praticiens du GRAFCET.

1.2. Les besoins des utilisateurs

Lorsqu'un analyste achève l'élaboration d'un grafcet de spécification, il se trouve à chaque fois confronté au problème de sa validation. Pour effectuer ce travail, il doit se poser deux questions fondamentales :

- ce grafcet, satisfait-il aux *recommandations normalisées du modèle GRAFCET* ?
- *les spécifications* définies dans le cahier des charges, sont-elles toutes (et seulement toutes) *correctement traduites* ?

S'il est bien évident que de telles questions ne doivent jamais rester sans réponse, nous venons de montrer que les analystes ne disposent aujourd'hui que de peu de moyens leur permettant d'y répondre à chaque fois d'une manière sûre. Si l'on excepte la simulation, il n'existe en effet pas d'outil qui puisse les assister pour la validation de grafcets. Cela conduit souvent les analystes à renforcer les précautions qu'ils prennent lors de l'élaboration des modèles. Afin d'éviter de produire des grafcets erronés, ils ont alors tendance à limiter l'emploi de certaines possibilités offertes par le GRAFCET jugées «dangereuses», ou à effectuer de la «sur-spécification». Nous

allons maintenant étudier, par quelques exemples significatifs, les conséquences de telles pratiques.

Dès l'apparition du GRAFCET dans l'industrie, de nombreux services de conception d'automatismes se sont aperçus que sa puissance de modélisation avait comme corollaire la difficulté de maîtriser les modèles réalisés, notamment dans la représentation des parallélismes. C'est pourquoi certaines sociétés ont imposé à leurs analystes de suivre des méthodes très strictes pour l'élaboration de grafcets, basées sur d'importantes restrictions concernant l'emploi des structures offertes par le GRAFCET. Dans [Citroën 88] par exemple, les règles suivantes sont définies :

- il est interdit d'utiliser les possibilités du parallélisme interprété offertes par le GRAFCET ; les conditions des transitions concurrentes doivent être mutuellement exclusives,
- deux actions exécutées simultanément seront contrôlées séparément,
- toute description de parallélisme se fera par parallélisme structural ou par graphes séparés[1],
- la mémorisation des actions est interdite[2],
- pas de saut de séquences dans une structure de parallélisme.

Dans le cadre de leur approche scientifique, certains universitaires ont également proposé de restreindre les capacités de modélisation du GRAFCET. Dans [David 89] par exemple, forts de leur expérience sur les RdP les auteurs présentent un certain nombre de recommandations pour l'élaboration de grafcets comme :

- *éviter tout «conflit» sur un grafcet (parallélisme interprété),*
- *éviter d'utiliser l'état interne (variables* X_i*),*
- *ne pas utiliser d'événements internes tels que* $\uparrow X_i$*,*
- *ne pas construire des grafcets pouvant avoir des cycles instables,*

1. A noter la contradiction avec la première règle décrite plus haut, le parallélisme par graphes séparés étant naturellement interprété.

2. Autre forme de parallélisme interprété.

- *manier avec prudence le cas où deux grafcets agissent globalement sur un troisième par forçage.*

Si certaines de ces règles ou recommandations sont naturelles (ne pas introduire sciemment de boucles d'instabilité totale dans un modèle ou manier avec prudence les forçages multiples sur un grafcet partiel), d'autres sont purement arbitraires, restrictives, voire contraires à des impératifs méthodologiques. Par exemple, l'interdiction d'utiliser des événements internes est en conflit avec la nécessité de découper un grafcet lorsque celui-ci est trop important, en réalisant la synchronisation des différents graphes par des événements internes [Bouteille 92]. Ajoutons que dans le cas général, ces recommandations amenuisent bien évidemment les capacités de modélisation du GRAFCET.

De plus, comme limiter l'emploi de certaines structures offertes par le GRAFCET ne suffit pas pour assurer que le modèle conçu est correct, les analystes ont également tendance à «sur-spécifier» afin de se protéger des erreurs qui peuvent s'être glissées dans leur modèle. Par exemple, pour être certain que deux actions mécaniquement incompatibles ne seront jamais émises simultanément, les analystes renforcent les conditions associées à chacune des actions en introduisant systématiquement des sécurités supplémentaires. Comme dans la majorité des cas ces conditions supplémentaires ne sont pas nécessaires au fonctionnement du système, les analystes introduisent ainsi des «sur-spécifications» dont les conséquences ne sont pas toujours mesurées. Ainsi sur la figure 4, si «indexage» et «rotation» sont deux actions mécaniquement incompatibles, les conditions associées (respectivement $\overline{X20}$ et $\overline{X10}$) garantissent que ces actions ne sont jamais émises en même temps, même si les étapes 10 et 20 sont actives simultanément. Cette «sur-spécification» est bien un aveu de l'analyste de son incapacité à savoir s'il existe des situations atteignables pour son modèle comprenant simultanément les étapes 10 et 20 actives.

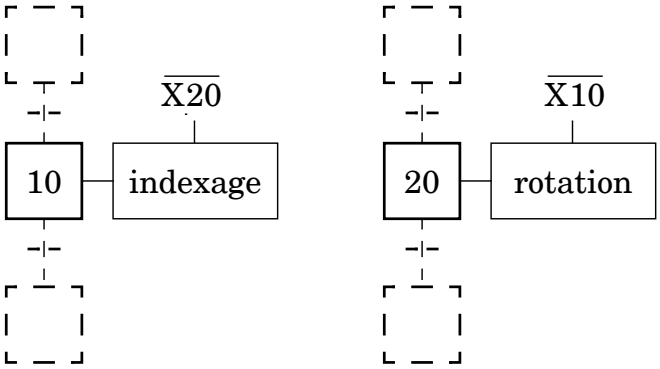


figure 4. Cas de sur-spécification introduite afin d'être certain que les deux actions ne seront jamais émises simultanément

Si une telle technique permet généralement d'accroître la sûreté de fonctionnement (la non émission d'ordres erronés est une composante essentielle de la sûreté de fonctionnement [Sourisse 85]), elle a tendance également à diminuer la disponibilité lorsque ces précautions supplémentaires sont réalisées par l'emploi des sécurités câblées de premier niveau [Pruvost 85]. De plus, cette technique de sur-spécification ne permet pas de corriger les erreurs de modélisation mais permet seulement d'en réduire les conséquences. Conceptuellement, cette pratique revient donc à admettre qu'une modélisation peut être erronée mais cependant jugée acceptable si on restreint les conséquences de ces erreurs !

Actuellement, en l'absence de toute technique de validation, ces deux pratiques (sous-utilisation du GRAFCET et sur-spécification) sont certainement les plus sûrs moyens opérationnels pour construire des modèles fiables. Il ne s'agit cependant que de mesures palliatives qui n'ont que vocation à limiter les risques d'erreurs de conception ou les conséquences de ces erreurs, mais qui en contre-partie n'exploitent pas pleinement la puissance de modélisation du GRAFCET et accroissent les contraintes non fonctionnelles du cahier des charges.

Le problème de fond qui reste posé est donc celui de la validation de tout modèle grafcet, sans considération restrictive sur le GRAFCET ni complément ou modification du cahier des charges.

1.3. Notre apport

Nous l'avons rappelé au début de ce chapitre (section 1.1.1. page 31) : le GRAFCET est une puissance machine d'état permettant notamment de représenter de manière ergonomique des parallélismes massifs. Cette puissance de modélisation se révèle pourtant être le principal obstacle à la validation de modèles. Cependant, comme l'a montré M. Blanchard [Blanchard 79] tout grafcet possède un grafcet d'état équivalent construit à partir de son graphe des situations accessibles.

Ce graphe des situations accessibles n'est en fait qu'une machine d'état rudimentaire sur laquelle de nombreuses techniques de validation éprouvées depuis plusieurs années sont applicables. Le graphe des situations accessibles fait de plus partie de la «culture GRAFCET», et de nombreux analystes ont recours à lui pour disséquer certaines structures délicates de grafcets.

C'est pour ces raisons que nous avons choisi de valider les modèles grafcet en nous appuyant sur leur graphe des situations accessibles.

Les deux problèmes fondamentaux auxquels nous nous sommes efforcés d'apporter des solutions sont :

- la **génération automatique** du graphe des situations accessibles, et ce pour toutes les possibilités de modélisation offertes par le GRAFCET,
- la **vérification automatique** des propriétés du graphe des situations accessibles spécifiées par l'analyste.

L'automatisation des procédures de génération du graphe des situations accessibles et de preuve de propriétés n'est en effet pas seulement un confort pour l'analyste mais bien une condition sine qua non de praticabilité de notre démarche puisque le problème majeur auquel elle se heurte est celui de l'explosion combinatoire du nombre des situations atteignables par un grafcet.

Dans la section suivante, nous présentons dans ses grandes lignes notre démarche pour la validation formelle de grafcets par l'analyse du graphe des situations accessibles. Cette présentation synthétique est destinée à identifier les principaux problèmes théoriques qui doivent être résolus avant de développer les détails de notre approche.

1.4. Présentation synthétique de notre démarche

La technique de validation que nous proposons se compose (cf. figure 5) d'une phase de *calcul des comportements* possibles du grafcet suivie d'une phase *d'analyse de ces comportements*.

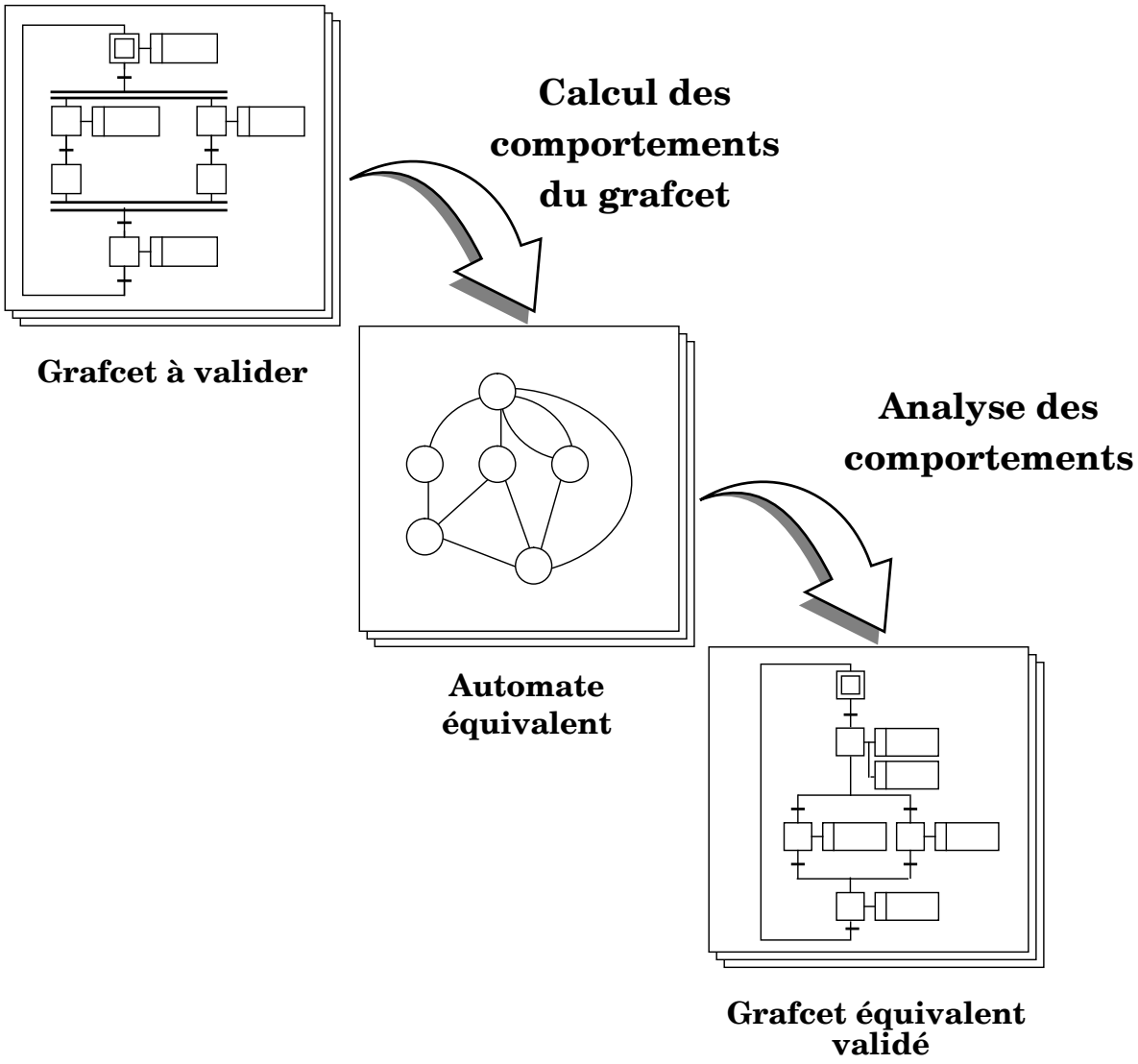


figure 5. Présentation de notre démarche pour la validation d'un grafcet

Nous avons tenu à distinguer ces deux phases car elles sont de nature et d'ampleur totalement différentes. En effet, le calcul de l'ensemble des comportements possibles d'un modèle relève d'une procédure systématique qui ne nécessite aucune intervention de l'analyste, alors que l'analyse des comportements demande quant à elle, moins de calculs mais qui sont tous à l'initiative de l'analyste.

Dans la pratique, cette distinction entre la phase de calcul des comportements et la phase d'analyse de ces comportements s'est avérée très pertinente car elle nous a permis d'adopter une stratégie d'analyse de grafcets dans laquelle les temps d'attente de l'analyste sont minimisés, la majorité des calculs étant effectués préalablement à l'analyse proprement dite.

1.4.1. Le calcul des comportements d'un modèle grafcet

D'une manière générale, cette phase consiste à dresser la liste exhaustive des situations accessibles par le grafcet et des possibilités d'évolution entre ces situations, c'est-à-dire le graphe des situations accessibles (GSA).

Pour permettre la validation d'un grafcet par rapport à son cahier des charges, la notion de situations accessibles, telle que l'a défini M. Blanchard, a été complétée pour prendre en compte les actions associées aux étapes. L'ensemble constitué du GSA et des actions associées est un **automate à états**. En effet, comme nous le verrons au chapitre 6, ce modèle est sur le plan théorique une Machine de Mealy complètement spécifiée dont la machine séquentielle est uniforme (d'après les définitions proposées dans [Zahnd 87]).

La démarche de génération du GSA que nous proposons comporte trois phases (cf. figure 6).

La première phase consiste *à générer le GSA à l'échelle de temps interne au grafcet* [UTE 93] *depuis chaque situation stable*. Pour cette génération, nous considérons que toutes les variations d'entrées peuvent se produire lorsque le grafcet est dans la situation stable considérée. Pour déterminer les différentes évolutions, il est successivement recherché : les transitions sensibilisées [Blanchard 79], les ensembles de transitions qui peuvent être simultanément franchies et enfin les variations d'entrées qui provoquent le franchissement simultané des transitions simultanément franchissables.

Le GSA ainsi calculé comporte toutes les situations stables ou instables accessibles depuis une situation stable donnée sur occurrence de tout événement extérieur.

La seconde étape consiste à *identifier le comportement du grafcet à l'échelle de temps externe* en construisant son automate à états équivalent. Lors de cette construc-

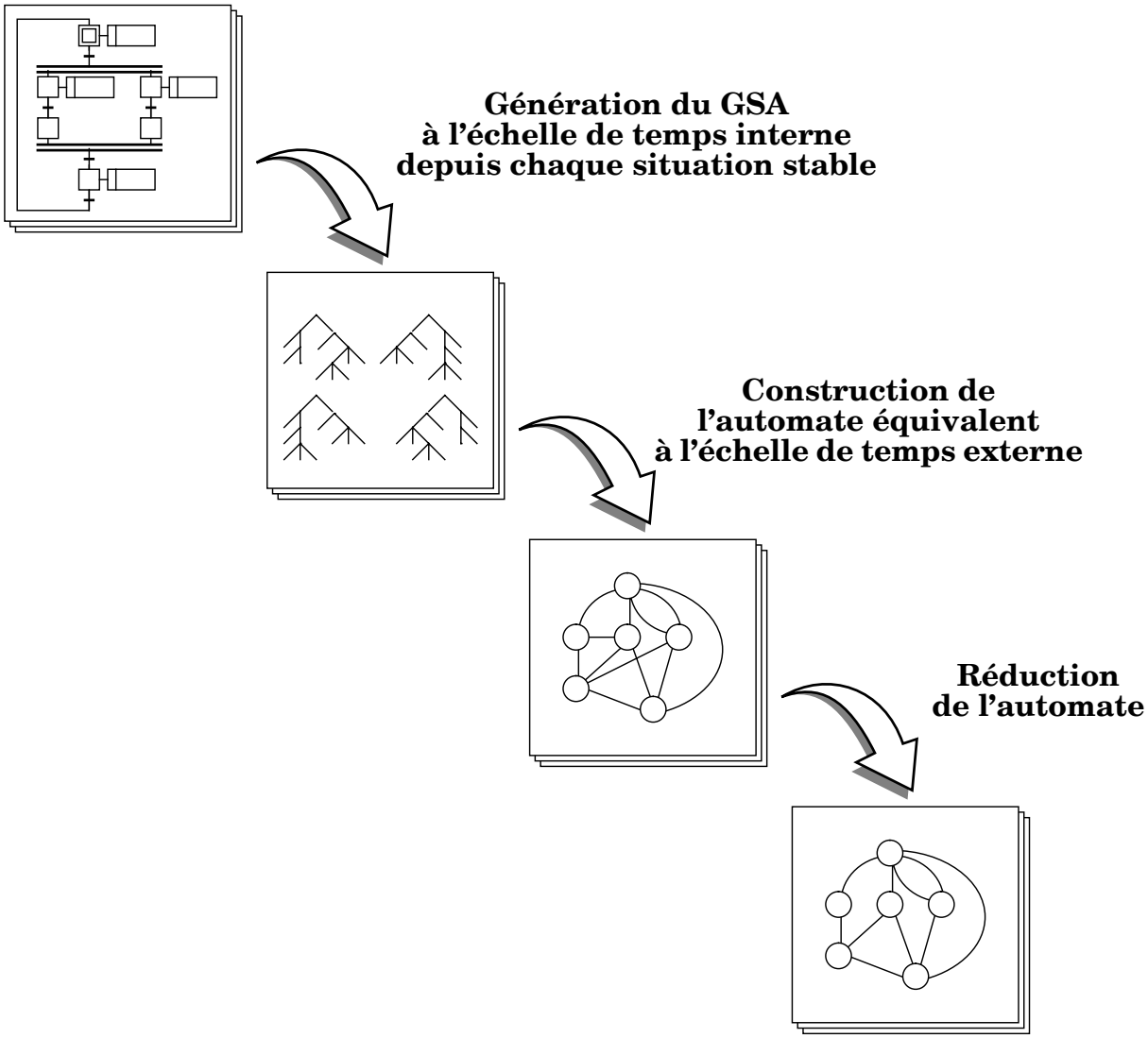


figure 6. Calcul des comportements d'un modèle grafcet

tion, qui s'appuie essentiellement sur les résultats de la phase précédente, il est d'abord déterminé le graphe de toutes les situations stables. Cette description est ensuite complétée en intégrant les actions associées aux étapes. Le résultat est présenté sous la forme d'un automate où chaque état représente une situation stable, chaque transition une évolution entre ces situations, et chaque action de l'automate une action émise depuis une situation stable.

La troisième phase de la démarche de génération est une *réduction de cet automate* en tenant compte de l'hypothèse de non simultanéité de deux occurrences d'événements externes non corrélés. Cette hypothèse, fondamentale en GRAFCET, a pour principale conséquence de limiter les variations d'entrées possibles depuis chaque situation stable.

1.4.2. L'analyse des comportements d'un modèle grafcet

Cette analyse qui se pratique sur l'automate à états équivalent au grafcet ne peut être entièrement automatisée puisque seul l'analyste est capable d'identifier les propriétés qu'il aimerait voir vérifier par son modèle. Cette opération doit cependant présenter un haut degré d'assistance informatique car la vérification manuelle de propriétés ne peut être envisagée en raison de la taille de l'automate.

Pour ce faire, nous avons mis en place un cadre formel pour l'expression des propriétés à vérifier. Cette formalisation repose sur une définition ensembliste de l'automate à états, adaptée autant aux besoins d'analyse qu'aux possibilités de traitement. Ces derniers s'appuient sur les travaux existants pour l'analyse des systèmes de transitions [Crubillé 89].

Après cette présentation succincte de la démarche que nous proposons pour la validation de grafcets, nous allons maintenant préciser quels sont les problèmes théoriques qu'il nous faut résoudre avant de développer la génération du GSA proprement dite.

1.5. Problèmes théoriques à résoudre

Le calcul des comportements d'un grafcet (et plus particulièrement le calcul des évolutions entre situations) suppose (cf paragraphe 1.4.1.) que soient clairement définies, les notions de **double échelle de temps** (temps interne, temps externe) de **stabilité** et de **forçage de situations**. Durant ces dernières années, le groupe GRAFCET de l'AFCET a été le cadre de nombreux travaux scientifiques sur ces sujets [Colombari 90], [Bouteille 92]. Une partie de ces travaux a d'ailleurs été normalisée en juin 1993, à l'occasion de la publication par l'UTE du complément [UTE 93] à la norme française NFC 03-190 [UTE 82] normalisant ainsi les extensions des concepts de base du GRAFCET. Cependant, quelques imprécisions ou ambiguïtés nous paraissent devoir être levées ; **le chapitre 2** de ce mémoire y sera consacré.

De la même manière, l'automatisation complète du calcul des conditions d'évolution entre les situations nécessite que ce calcul soit purement symbolique, y compris pour les **événements externes** (occurrence de changement d'état d'une entrée d'un

grafcet) ou **internes** (occurrence de changement d'état d'une variable d'étape d'un grafcet). La notion de front, bien que pratiquée en grafcet depuis 1977 [Afcet 77], n'a jamais fait l'objet d'une définition formelle qui permette le calcul symbolique sur les événements. Dans **le chapitre 3** de ce mémoire, nous proposons de construire une algèbre de Boole intégrant des opérateurs unaires «fronts» sur laquelle de tels calculs pourront être développés.

Parmi les problèmes théoriques qui doivent également être résolus pour établir le graphe des situations accessibles, le plus important est certainement celui de la maîtrise de l'explosion combinatoire que ce calcul peut entraîner. S'il est évident qu'il existe des grafcets dont la taille et la complexité sont telles que le calcul de ce graphe soit matériellement irréaliste, les progrès de l'informatique nous permettent aujourd'hui d'envisager cette opération pour un grand nombre de grafcets à condition de maîtriser au mieux l'explosion combinatoire qui peut se présenter. Lors de la génération du graphe des situations accessibles, cette maîtrise est un facteur déterminant de réussite : se masquer le problème, ou le juger seulement comme un problème de développement logiciel revient à considérer que la validation de grafcets par analyse du graphe des situations accessibles est une solution utopique qui ne peut fonctionner que sur des «cas d'école». Pour notre part, nous avons considéré ce problème si crucial qu'il est l'un des principaux critères de choix pour la manière de prendre en compte les entrées. Nous exposerons cette technique au **chapitre 4** de ce mémoire.

Une fois ces différentes bases théoriques exposées, nous décrivons dans les **chapitres 5 et 6** de ce mémoire la méthode de génération du GSA et d'analyse du comportement d'un grafcet.

Deux exemples illustrant la démarche sont ensuite exposés dans le **chapitre 7**.

Chapitre 2
Compléments théoriques concernant le modèle GRAFCET

Durant ces dernières années, de nombreux travaux sont menés au sein du groupe GRAFCET de l'AFCET pour renforcer les fondements théoriques du modèle. Ceux-ci portent essentiellement sur l'interprétation temporelle du GRAFCET et sur le forçage de situations. Dans ce chapitre, nous en reprenons les principaux résultats que nous complétons lorsque cela est nécessaire aux développements de nos travaux.

2.1. La double échelle de temps

2.1.1. Historique

En 1990, dans [Colombari 90], il est proposé un cadre temporel pour le GRAFCET composé d'une double échelle de temps afin de satisfaire les deux finalités suivantes :

- tous les événements externes doivent être pris en compte dès leur occurrence et ceci pour toutes leurs incidences,
- toutes les évolutions consécutives à l'occurrence d'un événement externe doivent toujours être effectuées avant l'apparition de l'occurrence d'un nouvel événement externe.

Ce cadre temporel, repris dans [Bouteille 92], fut normalisé à l'occasion de la publication par l'UTE du complément [UTE 93] à la norme française NFC 03-190 [UTE 82].

2.1.2. Définitions (d'après UTE C 03-191)

L'isolement d'un système décrit par GRAFCET établit une frontière de description qui définit une partition de l'univers en un interne et un externe au modèle. Cette frontière d'isolement correspond également à la frontière temporelle entre une échelle de temps interne et une échelle de temps externe au modèle. Ces deux échelles de temps sont sans commune mesure.

A l'échelle du temps externe, tout changement d'état des entrées est pris en compte par le modèle, dès son apparition. La totalité des conséquences de cet événement sur le modèle est déterminé à temps nul. Depuis l'extérieur du modèle, les événements d'entrée et les états des sorties qui en résultent sont vues à la même date.

A l'échelle de temps interne, la durée séparant l'instant où une transition est franchissable de l'instant où elle est franchie (appelée aussi durée d'évolution) est aussi petite qu'il est nécessaire, mais non nulle. En conséquence la durée minimale de l'activité d'une étape ne sera jamais nulle.

Ces définitions sont illustrées figure 7.

2.1.3. Conséquences sur le modèle

Grâce à l'introduction de la double échelle de temps, le modèle GRAFCET possède maintenant les caractéristiques suivantes :

- seules les sorties associées aux étapes appartenant à des situations stables sont émises ;
- seules les situations stables sont sensibles aux changements d'état des entrées ;
- seules les situations instables sont sensibles aux activations/désactivations d'étapes.[1]

Il est cependant nécessaire de définir clairement la notion de **stabilité d'une situation**.

Dans [Bouteille 92], il est précisé qu'une situation est stable si, après application des ordres de forçage et franchissement de toutes les transitions franchissables, une

1. Il n'y a donc jamais simultanéité entre en événement interne et un événement externe.

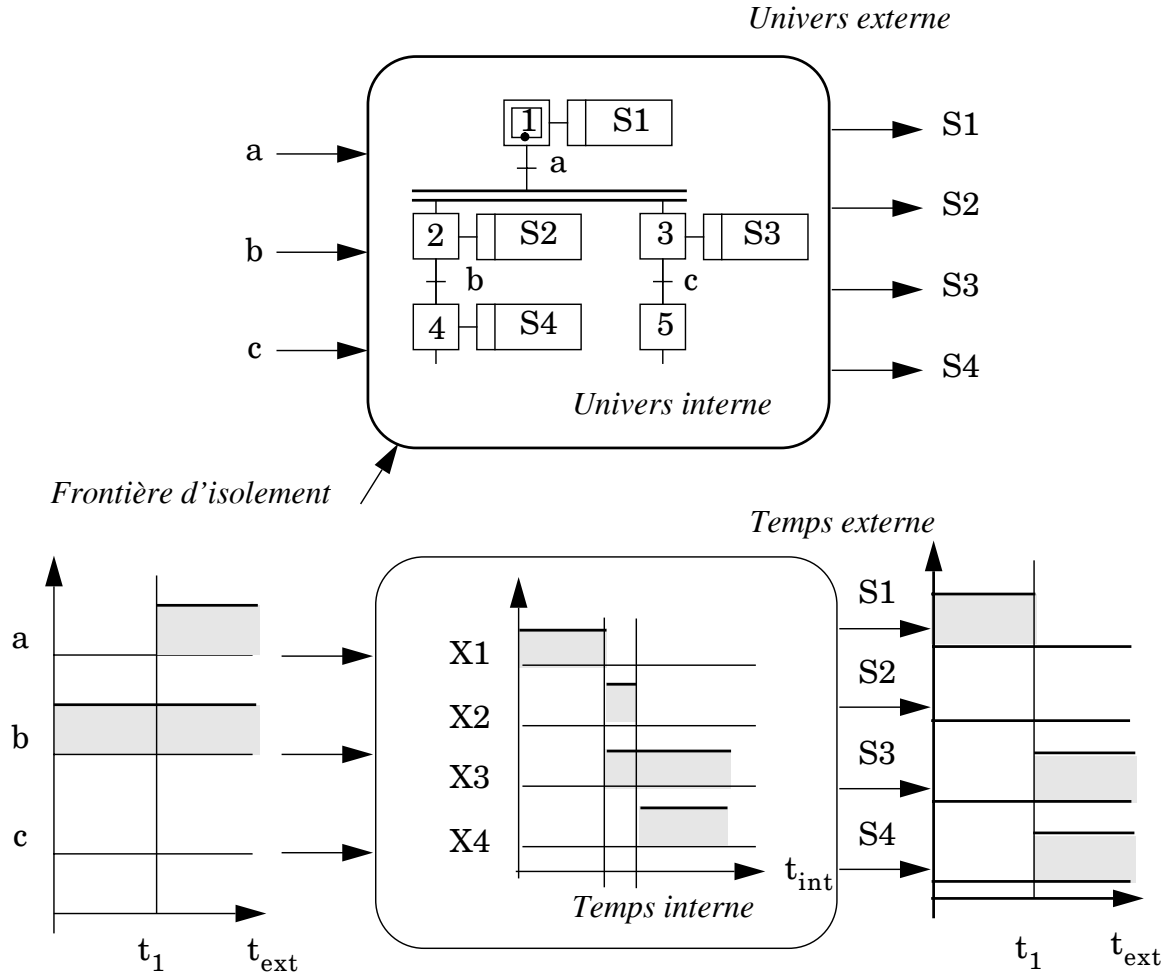


figure 7. Frontière d'isolement et échelle de temps

nouvelle situation ne peut être atteinte que sur occurrence d'un événement externe. Dans [Afcet 83], la stabilité d'une situation est définie comme suit : «Quand *un GRAFCET arrive dans une situation, celle-ci est dite instable si une au moins des transitions devient franchissable, sans occurrence d'événement externe. Si aucune des transitions n'est franchissable la situation est dite stable.*»

Bien qu'exprimées dans des termes différents, ces deux définitions sont sensiblement équivalentes et ramènent la notion de stabilité d'une situation à la nécessité d'occurrence d'événement externe pour franchir une des transitions validées.

Remarque :

La stabilité d'une situation ne dépend donc pas seulement de cette situation mais également de la variation d'entrées qui a permis de l'atteindre. Ainsi, pour le grafcet proposé figure 8 où la situation courante est {10}, la situation {11} atteinte sur occurrence de l'événement $\uparrow m$ sera stable si $\bar{a} \cdot \bar{b} = 1$ et instable si $\uparrow m$ survient lorsque $a + b = 1$.

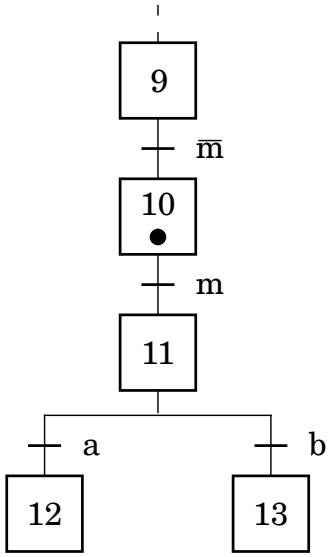


figure 8. Grafcet montrant que la stabilité d'une situation ne dépend pas que de cette situation mais également de la variation d'entrées qui a permis de l'atteindre.

Cette notion de stabilité doit maintenant être complétée, car l'une des propriétés essentielles que tout analyste exige généralement de son modèle est qu'il soit exempt de «cycles» [Blanchard 79][1], c'est-à-dire qu'il ne comporte pas de situations «stationnaires» [Afcet 83][2] ou «totalement instables»[3]. N'ayant pas trouvé dans la littérature de critère d'instabilité totale suffisamment rigoureux, nous allons maintenant exposer celui que nous avons construit.

2.2. La détection des situations totalement instables

2.2.1. Etude de l'existant

Que cela soit lors d'une simulation ou lors d'un calcul du graphe des situations accessibles, le problème de la détection des situations totalement instables pose le

1. [Blanchard 79], page 142 : «*Un grafcet peut, dans une certaine configuration des variables d'entrée, non pas évoluer vers une situation stable mais être constamment en évolution et repasser périodiquement par la même succession de situations. On dit qu'il évolue suivant un cycle*».

2. [Afcet 83] : «*On dira qu'un grafcet possède une situation stationnaire s'il existe un cycle de situations instables, pour une configuration des entrées*».

3. Dans la pratique le vocabulaire de «situation totalement instable» est souvent préféré aux deux précédents car plus imagé.

problème de choix d'un critère d'instabilité. A notre connaissance, les rares outils informatiques offrant une interprétation avec recherche de stabilité (comme [Le Parc 94], [Pagnol 93]) ont retenu comme critère d'instabilité totale l'un des deux suivants :

- Critère n°1 : L'instabilité totale est prononcée dès que le nombre d'évolutions entre deux événements externes est supérieur au nombre d'étapes contenues dans le grafcet global.
- Critère n°2 : L'instabilité totale est prononcée dès que le grafcet atteint deux fois une même situation instable entre deux événements externes.

Nous pouvons affirmer que ces deux critères, même s'ils donnent la plupart du temps de bons résultats, ne sont pas totalement fiables car ils conduisent à détecter de faux cas d'instabilité totale. Pour étayer cette affirmation, nous nous appuierons sur les deux exemples présentés sur les figures 9 et 10.

Le critère n°1 est mis en défaut par le grafcet présenté figure 9 qui se compose de seulement 7 étapes mais qui nécessite 12 évolutions pour atteindre la situation stable {10, 20} à partir de la situation initiale sur occurrence de l'événement ↑m .

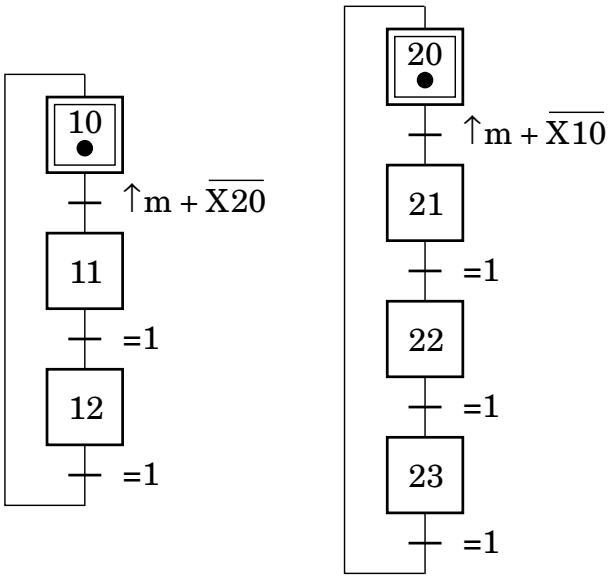


figure 9. Grafcet de 7 étapes nécessitant 12 évolutions pour atteindre une situation stable

Le critère n°2 est mis en défaut par le grafcet présenté figure 10 qui, depuis de la situation initiale, sur occurrence de l'événement ↑m passe 2 fois par la même situation instable {10, 21} pour atteindre la situation stable {10, 20}.

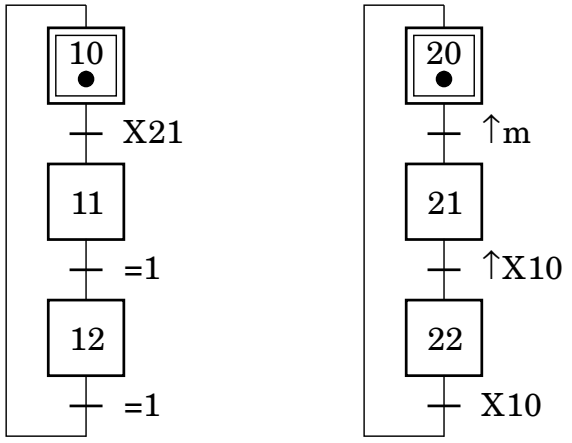


figure 10. Grafcet passant 2 fois par la même situation instable {10, 21} pour atteindre une situation stable

2.2.2. Critère d'instabilité totale retenu

Pour pallier les carences que nous venons d'exposer, nous avons défini le critère d'instabilité totale suivant :

Théorème 1 :

Si, sur occurrence d'un événement externe, un grafcet effectue à deux reprises la même évolution, alors il y a instabilité totale.

Par évolution, nous entendons passage d'une situation à une autre situation.

Illustration :

Depuis la situation initiale du grafcet proposé figure 11(a), l'événement ↑a conduit à une situation totalement instable. Le graphe des situations accessibles de la figure 11(b) met en évidence cette instabilité totale par l'existence de deux évolution identiques ({11} vers {12}). L'événement ↑b ne provoque quant à lui pas d'instabilité totale, car lorsque la situation S_9 est atteinte aucune transition n'est franchissable. S_9 est donc une situation stable.

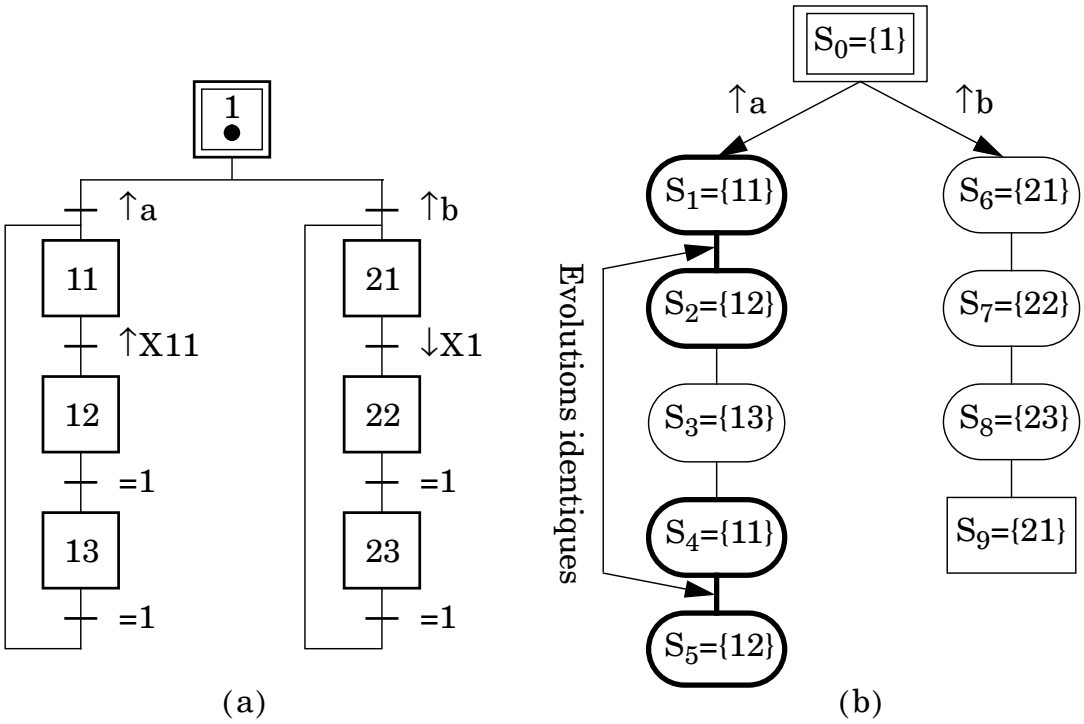


figure 11. Grafcet dont une seule branche conduit à une situation totalement instable

Démonstration :

Nous noterons Ev_i la $i^{ième}$ évolution d'un grafcet à l'échelle de temps interne, depuis l'occurrence d'un événement externe (cf. figure 12).

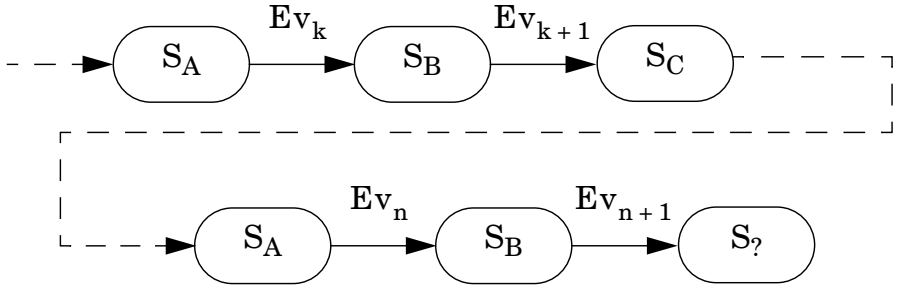


figure 12. Support de la démonstration du critère d'instabilité totale

Montrons que si $Ev_n = Ev_k$ $(k < n)$ alors nécessairement $Ev_{n+1} = Ev_{k+1}$.

Pour que Ev_{n+1} soit différente de Ev_{k+1} lorsque Ev_n est égale à Ev_k il est nécessaire que l'ensemble des transitions franchissables à l'issue de Ev_n (c'est-à-dire depuis la situation S_B) soit différent de l'ensemble des transitions franchissables à l'issue de Ev_k (depuis la même situation S_B). Or, il s'agit des mêmes transitions vali-

dées et la valeur de leur réceptivité associée est identique dans un cas comme dans l'autre puisque la valeur des entrées est inchangée et les mêmes événements internes sont générés. Nécessairement lorsque $Ev_n = Ev_k$, Ev_{n+1} est identique à Ev_{k+1}.

Toute évolution postérieure Ev_m (m>n) sera donc identique à une des évolutions antérieure Ev_j $(k \leq j < n)$. Comme aucune de ces évolutions n'a permis d'atteindre une situation stable, aucune situation stable ne pourra jamais être atteinte.

C.Q.F.D.

Ce critère d'instabilité totale sera utilisé lors de la génération du graphe des situations accessibles à l'échelle de temps interne depuis chaque situation stable.

2.3. Le forçage de situations

Le concept de forçage fut introduit dans le GRAFCET pour permettre d'imposer la situation d'un grafcet partiel donné, à partir d'un autre grafcet partiel [Afcet 87]. En utilisant le forçage, l'analyste spécifie que le grafcet forcé se retrouve, dès que l'ordre de forçage est émis dans la situation imposée par le grafcet forçant.

En 1990, à partir de l'exemple présenté figure 13, le groupe GRAFCET montra que le modèle dynamique retenu pour le forçage pouvait être, dans certains cas, en désaccord avec ce que l'on en attendait sur le plan fonctionnel. En effet, sur occurrence de $\uparrow a$, l'interprétation qui était proposée à cette époque du grafcet de la figure 13 conduisait à une activation à priori non désirée par son auteur de l'étape 7. Pour renforcer l'aspect fonctionnel du forçage, le groupe proposa que l'ordre de forçage soit prioritaire sur l'application des règles d'évolution afin que le grafcet forcé se retrouve, dès que l'ordre de forçage est émis dans la situation imposée par le grafcet forçant.

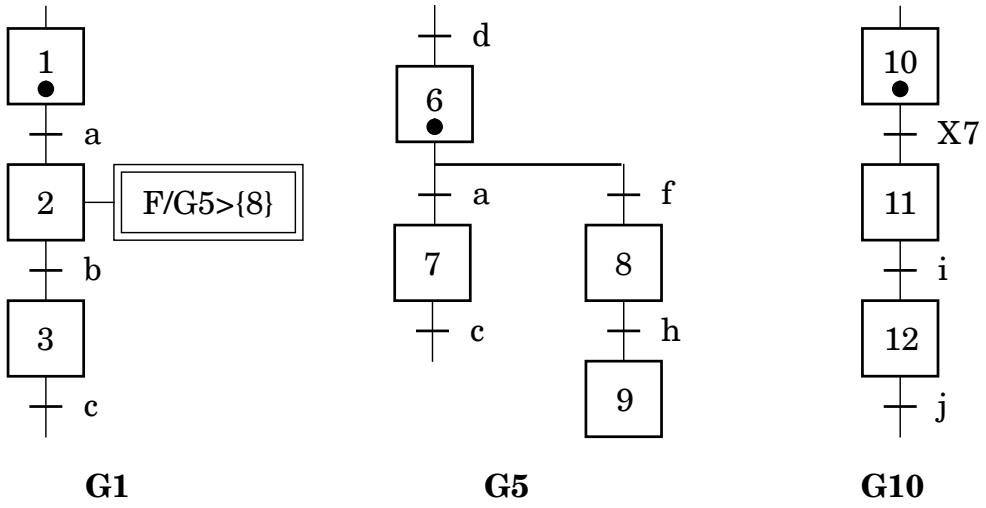


figure 13. Grafcet présentant un cas de conflit entre forçage et évolutions par franchissement de transitions d'après [Colombari 90]

Cette proposition fut celle retenue lors de la normalisation du concept du forçage puisqu'il est stipulé dans [UTE 93], que *«le forçage est un ordre interne dont l'exécution est prioritaire sur l'application des règles d'évolution. Le grafcet forcé ne peut évoluer tant que l'ordre de forçage est présent»*.

Cependant, le concept de forçage présente encore aujourd'hui quelques imprécisions dans son interprétation pour des cas très spécifiques (cf. figure 14). Nous avons dû faire des choix «arbitraires», mais dictés par le bon sens fonctionnel, pour lever ces ambiguïtés.

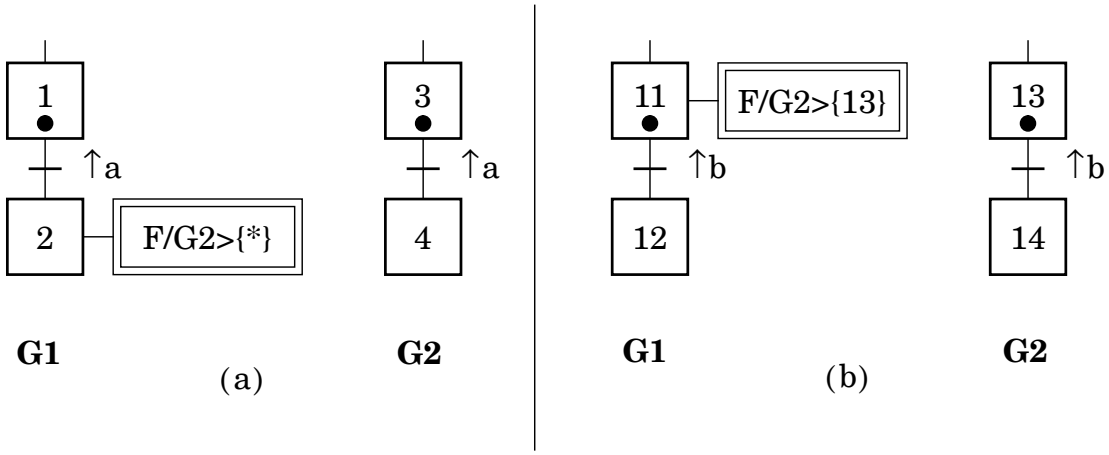


figure 14. Grafcets dont l'interprétation n'est pas clairement définie

Sur l'exemple figure 14(a), deux choix d'évolution sont possibles : on peut considérer qu'à l'apparition de l'entrée a , le grafcet atteint soit la situation {2, 3} soit la situation {2, 4}. Nous avons opté pour la situation {2, 4} car nous considérons que lors d'un forçage en situation courante, la situation imposée par le grafcet forçant est la situation prise par le grafcet forcé à l'instant où l'ordre est émis (c'est-à-dire à l'activation de {2}).

Sur l'exemple figure 14(b), deux choix d'évolutions sont également possibles : on peut considérer qu'à l'apparition de l'entrée b le grafcet atteint la situation {12, 13} ou à la situation {12, 14}. Nous avons opté pour la situation {12, 13} car nous considérons que lorsque b apparaît, le grafcet G2 n'est pas libre d'évoluer puisque l'ordre de forçage est toujours émis.

Pour générer le graphe des situations accessibles en tenant compte du forçage, il est également nécessaire de prendre position sur le comportement attendu en cas de forçages simultanés. Pour notre part, nous avons pris la position la plus catégorique qui consiste à considérer que le cas de forçages simultanés, compatibles ou non, issus de grafcets de niveaux hiérarchiques différents ou non, correspond à une erreur de modélisation.

Dans ce chapitre, nous venons de lever les ambiguïtés que contenaient le modèle GRAFCET concernant sa dynamique. Dans le chapitre suivant, nous nous intéressons exclusivement aux problèmes de la prise en compte des événements dans l'interprétation des modèles.

Chapitre 3
Construction d'une algèbre de Boole pour l'approche événementielle en GRAFCET

3.1. Problématique

En GRAFCET, la notion de front est un concept utilisé de manière courante depuis 1977 [Afcet 77], bien qu'elle n'ait jamais fait l'objet d'une définition formelle. Sur le plan théorique, ce concept correspond à une information de spectre temporel nul (cf. figure 15), traduisant le changement d'état supposé instantané, d'une variable logique ou d'une fonction de variables logiques [Grepa 85], [CEI 88].

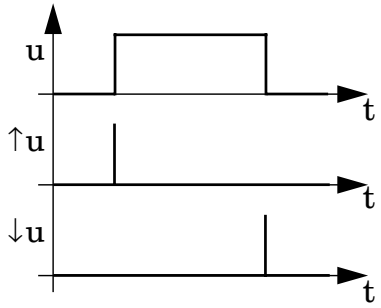


figure 15. Front montant et front descendant d'une variable booléenne

Dans la pratique cependant, l'emploi des fronts est souvent limité aux seules variables logiques élémentaires car l'évaluation d'expressions comprenant des fronts de fonctions de variables logiques - comme l'équation 1 - n'est pas maîtrisée.

$$E_1 = \uparrow(a \cdot \bar{c} + b) \cdot \downarrow(a \cdot b + c) \qquad \text{(Eq. 1)}$$

En effet, seuls R. David et H. Alla donnent - sous les hypothèses restrictives de non-simultanéité d'événements, d'indépendance totale entre les variables a, b et après avoir recommandé de ne pas utiliser d'événements internes dans les réceptivités d'un grafcet - les deux égalités suivantes [David 89] :

$$\uparrow(a \cdot b) = \uparrow a \cdot b + a \cdot \uparrow b$$
$$\uparrow(a + b) = \uparrow a \cdot \bar{b} + \bar{a} \cdot \uparrow b$$

Ces deux propriétés trouvent donc leur limite lorsque des événements se produisent simultanément, ce qui est le cas pour des variables corrélées et plus généralement pour les variables internes d'un grafcet.

Il suffit d'examiner une expression telle que (Eq. 1) pour identifier la cause des difficultés à évaluer de telles équations logiques. En effet, dans cette expression "+" et "." représentent les deux lois de composition de l'Algèbre de Boole et "$\bar{}$" la loi unaire de complément, elles sont parfaitement définies par leur table de vérité. Par contre "$\uparrow$" n'est qu'une simple notation indiquant que le concepteur de cette expression s'intéresse au changement d'état de la fonction $(a \cdot \bar{c} + b)$ et non à son niveau logique "1".

Pour être à même de développer et d'évaluer de telles expressions, quel que soit le nombre de variables composant les fonctions logiques et en prenant en compte la possibilité d'événements distincts simultanés, nous avons construit une algèbre de Boole "étendue" comprenant deux lois unaires événementielles : le "front montant" et le "front descendant" [Roussel 93]. Ainsi munis d'une définition algébrique des fronts, nous établirons un ensemble de propriétés permettant le développement et l'évaluation d'expressions telles que (Eq. 1).

3.2. Construction d'une algèbre de Boole «étendue»

3.2.1. Ensemble de définition

L'ensemble de définition de l'algèbre recherchée doit :

- représenter fidèlement les entrées et sorties de tout système logique,
- permettre la prise en compte "temporelle" des événements,

- comprendre au moins deux éléments,
- être stable pour toutes les lois qui y seront définies (c'est-à-dire, être tel que toute composition possible entre les éléments de cet ensemble soit un élément de cet ensemble).

En tenant compte de ces quatre critères nous avons retenu la définition suivante :

Définition 1 :

Nous appelons $\mathbb{I}$, l'ensemble des fonctions définies sur $\mathbb{R}^{+*}$, à valeurs booléennes, qui vérifient la propriété suivante :

$$\mathbb{I} = \{ u : \mathbb{R}^{+*} \to \mathbb{B} \mid \forall t \in \mathbb{R}^{+*} : \left(\exists \varepsilon_t > 0 : \left(\forall (\varepsilon_1, \varepsilon_2) \in]0, \varepsilon_t[^2, u(t-\varepsilon_1) = u(t-\varepsilon_2) \right) \right) \}$$

$$\text{avec } \mathbb{B} = \{0, 1\}$$

Par définition, toutes les fonctions u de $\mathbb{I}$ sont donc continues par morceaux et admettent en certains points une double discontinuité. La forme générale d'une fonction de l'ensemble $\mathbb{I}$ est représentée sur la figure 16.

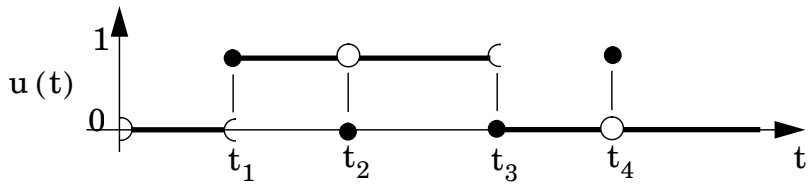


figure 16. Exemple de fonction élément de l'ensemble $\mathbb{I}$

Aux points de discontinuité, se pose le problème de la valeur de la fonction. La fonction peut en effet être considérée continue à droite, ou continue à gauche. Nous retiendrons avec J.P. Frachet la continuité à droite, plus naturelle pour le physicien puisque causale [Frachet 93]. A la date d'occurrence d'un événement, nous considérerons donc que la fonction a déjà changé de valeur.

Nous admettons l'existence de points présentant un phénomène de double discontinuité ($u(t_2) = 0$; $u(t_4) = 1$) de manière à assurer la stabilité de $\mathbb{I}$ relativement aux lois fronts.

Il convient de remarquer qu'une telle définition des éléments de $\mathbb{I}$ est parfaitement conforme à la pratique des automaticiens qui représentent souvent ainsi, sous

forme de chronogrammes l'évolution temporelle des variables booléennes des systèmes étudiés.

3.2.2. Convention de notation

Pour faciliter la lecture de ce chapitre et afin d'éviter toute confusion entre les opérations sur les éléments de $\mathbb{I}$ (fonction $u : \mathbb{R}^{+*} \to \mathbb{B}$) et les booléens (valeurs prises par ces fonctions à un instant donné), nous noterons dans toute la suite de ce chapitre "$\wedge$" l'opérateur ET logique entre deux booléens, "$\vee$" l'opérateur OU logique entre deux booléens et "$\neg$" l'opérateur NON sur un booléen. Les notations ".", "+", et "$\bar{\ }$" seront quant-à-elles réservées aux opérations sur $\mathbb{I}$.

De plus, nous avons pris soin de toujours distinguer la fonction du booléen, valeur prise à un instant donné par cette fonction. Par exemple, u, v, w sont trois fonctions élément de $\mathbb{I}$ tandis que u(t), v(t), w(t) sont trois booléens.

3.2.3. Définition des opérations sur $\mathbb{I}$

Après avoir explicité notre ensemble de définition et précisé les notations, nous pouvons définir sur $\mathbb{I}$ des lois de composition.

Définition 2 :

L'ensemble $\mathbb{I}$ peut être muni des trois lois suivantes :

- la loi **ET**

$$\begin{array}{c} \mathbb{I}^2 \to \mathbb{I} \\ (u, v) \to (u \cdot v) \end{array} \quad \text{avec } \forall t \in \mathbb{R}^{+*},\ (u \cdot v)(t) = u(t) \wedge v(t)$$

- la loi **OU**

$$\begin{array}{c} \mathbb{I}^2 \to \mathbb{I} \\ (u, v) \to (u + v) \end{array} \quad \text{avec } \forall t \in \mathbb{R}^{+*},\ (u + v)(t) = u(t) \vee v(t)$$

- la loi **NON**

$$\begin{array}{c} \mathbb{I} \to \mathbb{I} \\ u \to \bar{u} \end{array} \quad \text{avec } \forall t \in \mathbb{R}^{+*},\ \bar{u}(t) = \neg u(t)$$

Par définition toutes ces lois sont internes.

3.2.4. Structure d'algèbre de Boole sur $\mathbb{I}$

Nantis de ces définitions, nous allons maintenant démontrer que l'ensemble $\mathbb{I}$ muni des lois définies précédemment possède une structure d'algèbre de Boole, c'est-à-dire [Permingeat 91] [Marchand 89] :

- Les deux lois de composition doivent être commutatives :

$$\forall (u, v) \in \mathbb{I}^2, u \cdot v = v \cdot u$$

$$\forall (u, v) \in \mathbb{I}^2, u + v = v + u$$

- Les deux lois de composition doivent être associatives :

$$\forall (u, v, w) \in \mathbb{I}^3, (u \cdot v) \cdot w = u \cdot (v \cdot w)$$

$$\forall (u, v, w) \in \mathbb{I}^3, (u + v) + w = u + (v + w)$$

- La loi ET doit posséder un élément neutre :
  Il s'agit de la fonction constante notée 1^* définie par : $\forall t \in \mathbb{R}^{+*}, 1^*(t) = 1$.

- La loi OU doit posséder un élément neutre :
  Il s'agit de la fonction constante notée 0^* définie par : $\forall t \in \mathbb{R}^{+*}, 0^*(t) = 0$.

- Chaque loi de composition est distributive par rapport à l'autre :

$$\forall (u, v, w) \in \mathbb{I}^3, u \cdot (v + w) = (u \cdot v) + (u \cdot w)$$

$$\forall (u, v, w) \in \mathbb{I}^3, u + (v \cdot w) = (u + v) \cdot (u + w)$$

- La somme de tout élément de $\mathbb{I}$ et de son complément est égale à 1^*. Le produit de tout élément de $\mathbb{I}$ et de son complément est égal à 0^* :

$$\forall u \in \mathbb{I}, u + \bar{u} = 1^*$$

$$\forall u \in \mathbb{I}, u \cdot \bar{u} = 0^*$$

Démonstration de la propriété de commutativité :

$$\forall (u, v) \in \mathbb{I}^2, \forall t \in \mathbb{R}^{+*}, (u \cdot v)(t) = u(t) \wedge v(t) = v(t) \wedge u(t) = (v \cdot u)(t)$$

$$\forall (u, v) \in \mathbb{I}^2, \forall t \in \mathbb{R}^{+*}, (u + v)(t) = u(t) \vee v(t) = v(t) \vee u(t) = (v + u)(t)$$

C.Q.F.D.

Démonstration de la propriété d'associativité :

$$\forall (u, v, w) \in \mathbb{I}^3, \forall t \in \mathbb{R}^{+*},$$
$$((u \cdot v) \cdot w)(t) = (u(t) \wedge v(t)) \wedge w(t) = u(t) \wedge v(t) \wedge w(t)$$
$$= u(t) \wedge (v(t) \wedge w(t)) = (u \cdot (v \cdot w))(t)$$

$$\forall (u, v, w) \in \mathbb{I}^3, \forall t \in \mathbb{R}^{+*},$$
$$((u + v) + w)(t) = (u(t) \vee v(t)) \vee w(t) = u(t) \vee v(t) \vee w(t)$$
$$= u(t) \vee (v(t) \vee w(t)) = (u + (v + w))(t)$$

C.Q.F.D.

Démonstration de la propriété des éléments neutres :

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, \left(u \cdot 1^*\right)(t) = u(t) \wedge 1 = u(t)$$

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, \left(u + 0^*\right)(t) = u(t) \vee 0 = u(t)$$

C.Q.F.D.

Démonstration de la propriété de distributivité :

$$\forall (u, v, w) \in \mathbb{I}^3, \forall t \in \mathbb{R}^{+*},$$
$$(u \cdot (v + w))(t) = u(t) \wedge (v(t) \vee w(t)) = (u(t) \wedge v(t)) \vee (u(t) \wedge w(t))$$
$$= ((u \cdot v) + (u \cdot w))(t)$$

$$\forall (u, v, w) \in \mathbb{I}^3, \forall t \in \mathbb{R}^{+*},$$
$$(u + (v \cdot w))(t) = u(t) \vee (v(t) \wedge w(t)) = (u(t) \vee v(t)) \wedge (u(t) \vee w(t))$$
$$= ((u + v) \cdot (u + w))(t)$$

C.Q.F.D.

Démonstration de la propriété des complémentaires :

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, (u + \bar{u})(t) = u(t) \vee \neg u(t) = 1^*(t)$$

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, (u \cdot \bar{u})(t) = u(t) \wedge \neg u(t) = 0^*(t)$$

C.Q.F.D.

De plus, nous pouvons vérifier les propriétés fondamentales classiques des algèbres de Boole :

- La loi NON est involutive :

$$\forall u \in \mathbb{I}, \bar{\bar{u}} = u$$

- Les lois de composition sont idempotentes :

$$\forall u \in \mathbb{I}, u \cdot u = u$$

$$\forall u \in \mathbb{I}\!\mathbb{I}, u + u = u$$

- Les théorèmes de De Morgan s'appliquent :

$$\overline{\sum_{i=1}^{n} u_i} = \prod_{i=1}^{n} \overline{u_i} \qquad \overline{\prod_{i=1}^{n} u_i} = \sum_{i=1}^{n} \overline{u_i}$$

Démonstration de la propriété d'involution :

$$\forall u \in \mathbb{I}\!\mathbb{I}, \forall t \in \mathbb{R}^{+*}, \bar{\bar{u}}(t) = \neg(\bar{u}(t)) = \neg(\neg u(t)) = u(t)$$

C.Q.F.D.

Démonstration de la propriété d'idempotence ;

$$\forall u \in \mathbb{I}\!\mathbb{I}, \forall t \in \mathbb{R}^{+*}, (u \cdot u)(t) = u(t) \wedge u(t) = u(t)$$

$$\forall u \in \mathbb{I}\!\mathbb{I}, \forall t \in \mathbb{R}^{+*}, (u + u)(t) = u(t) \vee u(t) = u(t)$$

C.Q.F.D.

Démonstration des théorèmes de De Morgan : (démonstration par récurrence)

$$\overline{\sum_{i=1}^{n} u_i} = \prod_{i=1}^{n} \overline{u_i}$$

Au rang 2 : $\overline{u + v} = \bar{u} \cdot \bar{v}$

$$\forall (u, v) \in \mathbb{I}\!\mathbb{I}^2, \forall t \in \mathbb{R}^{+*}, \overline{(u+v)}(t) = \neg(u+v)(t) = \neg(u(t) \vee v(t)) = \neg u(t) \wedge \neg v(t) = \bar{u}(t) \wedge \bar{v}(t) = (\bar{u} \cdot \bar{v})(t)$$

La propriété est donc vraie au rang 2. Admettons maintenant qu'elle soit vraie au rang n et démontrons qu'elle est alors vraie au rang n+1.

$$\overline{\sum_{i=1}^{n+1} u_i} = \overline{\sum_{i=1}^{n} u_i + u_{n+1}} = \overline{\sum_{i=1}^{n} u_i} \cdot \overline{u_{n+1}} = \prod_{i=1}^{n} \overline{u_i} \cdot \overline{u_{n+1}} = \prod_{i=1}^{n+1} \overline{u_i}$$

Le prédicat "vrai au rang n implique vrai au rang n+1" pour la propriété venant d'être vérifié, nous pouvons maintenant conclure que la propriété est vraie quel que soit n.

C.Q.F.D.

$$\overline{\prod_{i=1}^{n} u_i} = \sum_{i=1}^{n} \overline{u_i}$$

Au rang 2 : $\overline{u \cdot v} = \bar{u} + \bar{v}$

$$\forall (u, v) \in \mathbb{I}^2, \forall t \in \mathbb{R}^{+*}, \overline{(u \cdot v)}(t) = \neg (u \cdot v)(t) = \neg (u(t) \wedge v(t))$$
$$= \neg u(t) \vee \neg v(t) = \bar{u}(t) \vee \bar{v}(t) = (\bar{u} + \bar{v})(t)$$

La propriété est donc vraie au rang 2. Admettons maintenant qu'elle soit vraie au rang n et démontrons qu'elle est alors vraie au rang n+1.

$$\overline{\prod_{i=1}^{n+1} u_i} = \overline{\prod_{i=1}^{n} u_i \cdot u_{n+1}} = \overline{\prod_{i=1}^{n} u_i} + \overline{u_{n+1}} = \sum_{i=1}^{n} \overline{u_i} + \overline{u_{n+1}} = \sum_{i=1}^{n+1} \overline{u_i}$$

Le prédicat "vrai au rang n implique vrai au rang n+1" pour la propriété venant d'être vérifié, nous pouvons maintenant conclure que la propriété est vraie quel que soit n.

C.Q.F.D.

3.3. Prise en compte des événements dans cette algèbre

3.3.1. Définition des lois fronts

Tout l'intérêt de cette algèbre réside dans le fait que nous pouvons maintenant définir rigoureusement deux lois unaires supplémentaires pour exprimer formellement la notion d'événement.

Définition 3 :

L'algèbre définie sur $\mathbb{I}$ peut être complétée par les deux lois suivantes :

- la loi **FM** (front montant)

$$\mathbb{I} \rightarrow \mathbb{I}$$
$$u \rightarrow \uparrow u$$

avec $\forall t \in \mathbb{R}^{+*}, \uparrow u(t) = u(t) \wedge (\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[, u(t - \varepsilon) = 0)$

- la loi **FD** (front descendant)

$$\mathbb{I} \rightarrow \mathbb{I}$$
$$u \rightarrow \downarrow u$$

avec $\forall t \in \mathbb{R}^{+*}, \downarrow u(t) = \bar{u}(t) \wedge (\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[, u(t - \varepsilon) = 1)$

Les images de la fonction $\uparrow u$ (respectivement $\downarrow u$) sont donc déterminées, à chaque instant, comme le ET logique de deux booléens. Le premier booléen est la valeur (respectivement le complément de la valeur) à cet instant de la fonction u, tandis que le second booléen est la valeur, à ce même instant, d'un prédicat. La véracité de ce prédicat dépend de la valeur prise par la fonction u sur l'intervalle $]t - \varepsilon_0, t[$.

Pour la fonction représentée sur la figure 16, par exemple :

- $\uparrow u(t) = 1$ si $t \in [t_1, t_2[\cup]t_2, t_3[\cup \{t_4\}$ et si $t \in]0, t_1] \cup]t_3, t_4] \cup]t_4, \infty[$ soit $t = t_1$ ou $t = t_4$.
- $\downarrow u(t) = 1$ si $t \in]0, t_1[\cup \{t_2\} \cup [t_3, t_4[\cup]t_4, \infty[$ et si $t \in]t_1, t_2] \cup]t_2, t_3]$ soit $t = t_2$ ou $t = t_3$.

Par définition, ces deux lois sont internes puisque les fonctions $\uparrow u$ et $\downarrow u$ sont définies sur $\mathbb{R}^{+*}$, à valeurs booléennes et vérifient la propriété des éléments de $\mathbb{I}$ développées au paragraphe 3.2.1.

ε_0 et ε étant toujours considérés strictement positifs, dans la suite de ce chapitre nous allégerons la notation en notant seulement "$\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0$" à la place de "$\exists \varepsilon_0 > 0 : \forall \varepsilon \in]0, \varepsilon_0[$".

3.3.2. Propriétés des lois front montant et front descendant

Nous allons démontrer dans cette section 14 propriétés concernant les lois front (dans l'annexe A, ces propriétés sont illustrées à l'aide de chronogrammes).

3.3.2.1. Composition des lois ET, OU, NON avec la loi FM

Propriété 1 :

$$u + \uparrow u = u$$

Démonstration :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*},\ (u + \uparrow u)(t) &= u(t) \vee \uparrow u(t) \\ &= u(t) \vee u(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t - \varepsilon) = 0) \\ &= u(t) \end{aligned}$$

C.Q.F.D.

Propriété 2 :

$$u \cdot \uparrow u = \uparrow u$$

Démonstration :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*},\ (u \cdot \uparrow u)(t) &= u(t) \wedge \uparrow u(t) \\ &= u(t) \wedge u(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 0) \\ &= \uparrow u(t) \end{aligned}$$

C.Q.F.D.

Propriété 3 :

$$\uparrow \bar{u} = \downarrow u$$

Démonstration :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*},\ \uparrow \bar{u}(t) &= \bar{u}(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ \bar{u}(t-\varepsilon) = 0) \\ &= \bar{u}(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ \neg u(t-\varepsilon) = 0) \\ &= \bar{u}(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 1) \\ &= \downarrow u(t) \end{aligned}$$

C.Q.F.D.

Propriété 4 :

$$\uparrow \left(\prod_{i=1}^{n} u_i \right) = \sum_{i=1}^{n} \left(\uparrow u_i \cdot \prod_{(j=1),\ (j \neq i)}^{n} u_j \right)$$

Démonstration :

Cette démonstration se fera par récurrence, mais il est nécessaire de démontrer au préalable le lemme suivant :

$$\forall t \in \mathbb{R}^{+*},\ u(t) \wedge \neg(\uparrow u(t)) = u(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 1)$$

Démonstration du lemme :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*},\ u(t) \wedge \neg(\uparrow u(t)) &= u(t) \wedge \neg(u(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 0)) \\ &= u(t) \wedge (\neg u(t) \vee \neg(\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 0)) \quad \text{(Eq. 2)} \\ &= u(t) \wedge \neg(\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u(t-\varepsilon) = 0) \end{aligned}$$

Or :

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, \neg(\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) = (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1)$$

En effet, toutes les fonctions de $\mathbb{I}$, étant continues par morceaux, vérifient les deux équations suivantes :

$$\forall t \in \mathbb{R}^{+*}, \begin{cases} (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) \vee (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1) = 1 \\ (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1) = 0 \end{cases}$$

Compte tenu des propriétés de la loi NON de l'Algèbre de Boole, les deux termes de ces équations sont nécessairement complémentaires.

On a donc :

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, \neg(\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) = (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1) \quad \text{(Eq. 3)}$$

En injectant ce résultat dans (Eq. 2), nous obtenons :

$$\forall t \in \mathbb{R}^{+*}, u(t) \wedge \neg(\uparrow u(t)) = u(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1)$$

Le lemme étant démontré, nous allons maintenant démontrer la propriété 4. Nous débuterons par la démonstration au rang 2, c'est-à-dire :

$$\uparrow(u \cdot v) = \uparrow u \cdot v + u \cdot \uparrow v$$

On a :

$$\begin{aligned} \forall t \in \mathbb{R}^{+*}, \uparrow(u \cdot v)(t) &= (u \cdot v)(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, (u \cdot v)(t-\varepsilon)=0) \\ &= u(t) \wedge v(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) \\ &\quad \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, v(t-\varepsilon)=0) \\ &\quad \vee u(t) \wedge v(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=1) \\ &\quad \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, v(t-\varepsilon)=0) \\ &\quad \vee u(t) \wedge v(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t-\varepsilon)=0) \\ &\quad \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, v(t-\varepsilon)=1) \end{aligned}$$

soit en utilisant le lemme que nous venons de démontrer :

$$\begin{aligned} \uparrow(u \cdot v)(t) = &(\uparrow u(t) \wedge \uparrow v(t)) \vee (u(t) \wedge \neg(\uparrow u(t)) \wedge \uparrow v(t)) \\ &\vee (\uparrow u(t) \wedge v(t) \wedge \neg(\uparrow v(t))) \end{aligned}$$

En remarquant que

$$\uparrow u(t) \wedge \uparrow v(t) = (\uparrow u(t) \wedge \uparrow v(t) \wedge v(t)) \vee (\uparrow u(t) \wedge u(t) \wedge \uparrow v(t))$$

l'expression devient :

$$\begin{aligned}\uparrow(u \cdot v)(t) = {} & (\uparrow u(t) \wedge \uparrow v(t) \wedge v(t)) \\ & \vee (\uparrow u(t) \wedge v(t) \wedge \neg(\uparrow v(t))) \\ & \vee (\uparrow u(t) \wedge u(t) \wedge \uparrow v(t)) \\ & \vee (u(t) \wedge \neg(\uparrow u(t)) \wedge \uparrow v(t))\end{aligned}$$

Soit après mise en facteur et simplification :

$$\begin{aligned}\uparrow(u \cdot v)(t) &= (\uparrow u(t) \wedge v(t)) \vee (u(t) \wedge \uparrow v(t)) \\ &= ((\uparrow u \cdot v) + (u \cdot \uparrow v))(t)\end{aligned}$$

La propriété 4 est donc vraie au rang 2. Admettons maintenant qu'elle soit vraie au rang n et démontrons qu'elle est alors vraie au rang n+1.

$$\begin{aligned}\uparrow\left(\prod_{i=1}^{n+1} u_i\right) &= \uparrow\left(\prod_{i=1}^{n} u_i \cdot u_{n+1}\right) = \uparrow\left(\prod_{i=1}^{n} u_i\right) \cdot u_{n+1} + \left(\prod_{i=1}^{n} u_i\right) \cdot \uparrow u_{n+1} \\ &= \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} u_j\right) \cdot u_{n+1} + \left(\prod_{i=1}^{n} u_i\right) \cdot \uparrow u_{n+1} \\ &= \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n+1} u_j\right) + \uparrow u_{n+1} \cdot \prod_{i=1}^{n} u_i \\ &= \sum_{i=1}^{n+1}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n+1} u_j\right)\end{aligned}$$

Le prédicat "vrai au rang n implique vrai au rang n+1" pour la propriété 4 venant d'être vérifié, nous pouvons maintenant conclure que la propriété 4 est vraie quel que soit n.

C.Q.F.D.

Propriété 5 :

$$\begin{aligned}\uparrow\left(\sum_{i=1}^{n} u_i\right) &= \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \uparrow u_j + \left(\overline{u_j} \cdot \overline{\downarrow u_j}\right)\right) \\ &= \prod_{i=1}^{n}\left(\uparrow u_i + \left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)}\end{aligned}$$

Nous proposons deux formes développées en raison de la complémentarité de leur utilisation. La première forme s'utilise lors d'un «développement manuel», chaque terme s'expliquant naturellement, tandis que la seconde forme est employée lors d'un développement automatique en raison de la compacité de son écriture.

Démonstration :

Cette démonstration se fera par récurrence, mais il est nécessaire de démontrer au préalable le lemme suivant :

$$\forall t \in \mathbb{R}^{+*},\ (\bar{u} \cdot \overline{\downarrow u})\ (t) \ = \ \neg u\,(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 0)$$

Démonstration du lemme :

Dans cette démonstration, seront utilisés les résultats de l'équation 3.

$$\begin{aligned}
\forall t \in \mathbb{R}^{+*},\ (\bar{u} \cdot \overline{\downarrow u})\ (t) \ &= \ \bar{u}\,(t) \wedge \neg (\bar{u}\,(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 1))\\
&= \ \neg u\,(t) \wedge (\neg \bar{u}\,(t) \vee \neg (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 1))\\
&= \ \neg u\,(t) \wedge (u\,(t) \vee (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 0))\\
&= \ \neg u\,(t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 0)
\end{aligned}$$

Le lemme démontré, nous allons maintenant démontrer la proposition suivante :

$$\uparrow (u+v) \ = \ (\uparrow u \cdot \bar{v} \cdot \overline{\downarrow v}) + (\uparrow v \cdot \bar{u} \cdot \overline{\downarrow u}) + \uparrow u \cdot \uparrow v \qquad \text{(Eq. 4)}$$

Démonstration :

$$\begin{aligned}
\forall t \in \mathbb{R}^{+*}, \uparrow (u+v)\ (t) \ &= \ (u+v)\ (t) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ (u+v)\ (t-\varepsilon) = 0)\\
&= \ (u\,(t) \wedge \neg v\,(t) \vee \neg u\,(t) \wedge v\,(t) \vee u\,(t) \wedge v\,(t))\\
&\quad \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ u\,(t-\varepsilon) = 0) \wedge (\exists \varepsilon_0,\ \forall \varepsilon < \varepsilon_0,\ v\,(t-\varepsilon) = 0)
\end{aligned}$$

En utilisant le lemme que nous venons de démontrer, nous obtenons :

$$\begin{aligned}
\uparrow (u+v)\ (t) \ &= \ (\uparrow u \cdot \bar{v} \cdot \overline{\downarrow v})\ (t) \vee (\uparrow v \cdot \bar{u} \cdot \overline{\downarrow u})\ (t) \vee (\uparrow u \cdot \uparrow v)\ (t)\\
&= \ ((\uparrow u \cdot \bar{v} \cdot \overline{\downarrow v}) + (\uparrow v \cdot \bar{u} \cdot \overline{\downarrow u}) + \uparrow u \cdot \uparrow v)\ (t)
\end{aligned}$$

Nous allons maintenant démontrer la propriété 5. Nous débuterons par la démonstration au rang 2 pour les deux formes.

Nous avons d'une part :

$$\begin{aligned}
\uparrow\left(\sum_{i=1}^{2} u_i\right) = \uparrow (u_1 + u_2) \ &= \ \left(\uparrow u_1 \cdot \overline{u_2} \cdot \overline{\downarrow u_2}\right) + \left(\uparrow u_2 \cdot \overline{u_1} \cdot \overline{\downarrow u_1}\right) + \uparrow u_1 \cdot \uparrow u_2\\
&= \ \uparrow u_1 \cdot \left(\uparrow u_2 + \left(\overline{u_2} \cdot \overline{\downarrow u_2}\right)\right) + \uparrow u_2 \cdot \left(\uparrow u_1 + \left(\overline{u_1} \cdot \overline{\downarrow u_1}\right)\right)\\
&= \ \sum_{i=1}^{2} \left(\uparrow u_i \cdot \prod_{(j=1),\ (j \neq i)}^{2} \uparrow u_j + \left(\overline{u_j} \cdot \overline{\downarrow u_j}\right)\right)
\end{aligned}$$

et d'autre part :

$$\uparrow\left(\sum_{i=1}^{2} u_i\right) = \uparrow(u_1+u_2) = \left(\uparrow u_1 \cdot \overline{u_2} \cdot \overline{\downarrow u_2}\right) + \left(\uparrow u_2 \cdot \overline{u_1} \cdot \overline{\downarrow u_1}\right) + \uparrow u_1 \cdot \uparrow u_2$$

$$= \left(\uparrow u_1 \cdot \overline{u_2} \cdot \overline{\downarrow u_2} + \uparrow u_2 \cdot \overline{u_1} \cdot \overline{\downarrow u_1} + \uparrow u_1 \cdot \uparrow u_2\right) \cdot \left(\prod_{i=1}^{2} \overline{u_i} \cdot \overline{\downarrow u_i} + \overline{\prod_{i=1}^{2} \overline{u_i} \cdot \overline{\downarrow u_i}}\right)$$

$$+ \left(\overline{u_1} \cdot \overline{\downarrow u_1}\right) \cdot \left(\overline{u_2} \cdot \overline{\downarrow u_2}\right) \cdot \overline{\prod_{i=1}^{2} \overline{u_i} \cdot \overline{\downarrow u_i}}$$

$$= \left(\left(\uparrow u_1 \cdot \overline{u_2} \cdot \overline{\downarrow u_2}\right) + \left(\uparrow u_2 \cdot \overline{u_1} \cdot \overline{\downarrow u_1}\right) + \uparrow u_1 \cdot \uparrow u_2\right) \cdot \left(\overline{u_1} \cdot \overline{\downarrow u_1}\right) \cdot \left(\overline{u_2} \cdot \overline{\downarrow u_2}\right)$$

$$+ \left(\left(\left(\uparrow u_1 \cdot \overline{u_2} \cdot \overline{\downarrow u_2} + \uparrow u_2 \cdot \overline{u_1} \cdot \overline{\downarrow u_1} + \uparrow u_1 \cdot \uparrow u_2\right) + \left(\overline{u_1} \cdot \overline{\downarrow u_1}\right) \cdot \left(\overline{u_2} \cdot \overline{\downarrow u_2}\right)\right)\right.$$

$$\left. \cdot \overline{\prod_{i=1}^{2} \overline{u_i} \cdot \overline{\downarrow u_i}}\right)$$

$$= 0 + \left(\left(\uparrow u_1 + \left(\overline{u_1} \cdot \overline{\downarrow u_1}\right)\right) \cdot \left(\uparrow u_2 + \left(\overline{u_2} \cdot \overline{\downarrow u_2}\right)\right)\right) \cdot \overline{\prod_{i=1}^{2} \overline{u_i} \cdot \overline{\downarrow u_i}}$$

$$= \prod_{i=1}^{2} \left(\uparrow u_i + \left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{2} \left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)}$$

La propriété 5 est donc vraie au rang 2 pour ces deux formes. Admettons maintenant qu'elles soient vraies au rang n et démontrons qu'elles sont alors vraies au rang n+1.

Nous débuterons par la simplification suivante :

$$\overline{\sum_{i=1}^{n} u_i} \cdot \overline{\downarrow\left(\sum_{i=1}^{n} u_i\right)} = \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\uparrow\left(\overline{\sum_{i=1}^{n} u_i}\right)} = \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\uparrow\left(\prod_{i=1}^{n} \overline{u_i}\right)}$$

$$= \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\sum_{i=1}^{n} \left(\uparrow \overline{u_i} \cdot \prod_{(j=1),\,(j\neq i)}^{n} \overline{u_j}\right)}$$

$$= \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\sum_{i=1}^{n} \left(\uparrow \overline{u_i} \cdot \prod_{j=1}^{n} \overline{u_j}\right)} = \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\sum_{i=1}^{n} \uparrow \overline{u_i} \cdot \prod_{i=1}^{n} \overline{u_i}}$$

$$= \prod_{i=1}^{n} \overline{u_i} \cdot \left(\prod_{i=1}^{n} \overline{\uparrow \overline{u_i}} + \overline{\prod_{i=1}^{n} \overline{u_i}}\right) = \prod_{i=1}^{n} \overline{u_i} \cdot \prod_{i=1}^{n} \overline{\downarrow u_i}$$

Nous avons, d'une part :

$$\uparrow\left(\sum_{i=1}^{n+1} u_i\right) = \uparrow\left(\left(\sum_{i=1}^{n} u_i\right) + u_{n+1}\right)$$

$$= \uparrow\left(\sum_{i=1}^{n} u_i\right) \cdot \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}} + \uparrow u_{n+1} \cdot \overline{\sum_{i=1}^{n} u_i} \cdot \downarrow \overline{\sum_{i=1}^{n} u_i} + \uparrow\left(\sum_{i=1}^{n} u_i\right) \cdot \uparrow u_{n+1}$$

$$= \uparrow\left(\sum_{i=1}^{n} u_i\right) \cdot \left(\uparrow u_{n+1} + \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}}\right) + \uparrow u_{n+1} \cdot \prod_{i=1}^{n} \overline{u_i} \cdot \prod_{i=1}^{n} \overline{\downarrow u_i}$$

$$= \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n+1} \uparrow u_j + \left(\overline{u_j} \cdot \overline{\downarrow u_j}\right)\right) + \uparrow u_{n+1} \cdot \prod_{i=1}^{n} \overline{u_i} \cdot \prod_{i=1}^{n} \overline{\downarrow u_i}$$

$$= \sum_{i=1}^{n+1}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n+1} \uparrow u_j + \left(\overline{u_j} \cdot \overline{\downarrow u_j}\right)\right)$$

et d'autre part :

$$\uparrow\left(\sum_{i=1}^{n+1} u_i\right) = \uparrow\left(\left(\sum_{i=1}^{n} u_i\right) + u_{n+1}\right)$$

$$= \left(\uparrow\left(\sum_{i=1}^{n} u_i\right) + \overline{\sum_{i=1}^{n} u_i} \cdot \downarrow \overline{\sum_{i=1}^{n} u_i}\right) \cdot \left(\uparrow u_{n+1} + \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}}\right)$$

$$\cdot \overline{\left(\overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}} \cdot \overline{\sum_{i=1}^{n} u_i} \cdot \downarrow \overline{\sum_{i=1}^{n} u_i}\right)}$$

$$= \left(\prod_{i=1}^{n}\left(\uparrow u_i + \overline{u_i} \cdot \overline{\downarrow u_i}\right) \cdot \overline{\prod_{i=1}^{n} \overline{u_i} \cdot \overline{\downarrow u_i}} + \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\downarrow u_i}\right) \cdot \left(\uparrow u_{n+1} + \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}}\right)$$

$$\cdot \overline{\prod_{i=1}^{n+1} \overline{u_i} \cdot \overline{\downarrow u_i}}$$

$$= \left(\prod_{i=1}^{n}\left(\uparrow u_i + \overline{u_i} \cdot \overline{\downarrow u_i}\right) + \prod_{i=1}^{n} \overline{u_i} \cdot \overline{\downarrow u_i}\right) \cdot \left(\uparrow u_{n+1} + \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}}\right) \cdot \overline{\prod_{i=1}^{n+1} \overline{u_i} \cdot \overline{\downarrow u_i}}$$

$$= \left(\prod_{i=1}^{n}\left(\uparrow u_i + \overline{u_i} \cdot \overline{\downarrow u_i}\right)\right) \cdot \left(\uparrow u_{n+1} + \overline{u_{n+1}} \cdot \overline{\downarrow u_{n+1}}\right) \cdot \overline{\prod_{i=1}^{n+1} \overline{u_i} \cdot \overline{\downarrow u_i}}$$

$$= \prod_{i=1}^{n+1}\left(\uparrow u_i + \left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{n+1}\left(\overline{u_i} \cdot \overline{\downarrow u_i}\right)}$$

Le prédicat “vrai au rang n implique vrai au rang n+1” pour les deux formes de la propriété 5 venant d’être vérifié, nous pouvons maintenant conclure que la propriété 5 est vraie quel que soit n.

C.Q.F.D.

3.3.2.2. Composition des lois ET, OU, NON avec la loi FD

Propriété 6 :

$$\bar{u} + \downarrow u = \bar{u}$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3 et la propriété 1.

$$\bar{u} + \downarrow u = \bar{u} + \uparrow \bar{u} = \bar{u}$$

C.Q.F.D.

Propriété 7 :

$$\bar{u} \cdot \downarrow u = \downarrow u$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3 et la propriété 2.

$$\bar{u} \cdot \downarrow u = \bar{u} \cdot \uparrow \bar{u} = \uparrow \bar{u} = \downarrow u$$

C.Q.F.D.

Propriété 8 :

$$\downarrow \bar{u} = \uparrow u$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3 et la propriété d’involution de la loi NON.

$$\downarrow \bar{u} = \uparrow \bar{\bar{u}} = \uparrow u$$

C.Q.F.D.

Propriété 9 :

$$\downarrow\left(\prod_{i=1}^{n} u_i\right) = \sum_{i=1}^{n}\left(\downarrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \downarrow u_j + \left(u_j \cdot \overline{\uparrow u_j}\right)\right)$$

$$= \prod_{i=1}^{n}\left(\downarrow u_i + \left(u_i \cdot \overline{\uparrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(u_i \cdot \overline{\uparrow u_i}\right)}$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3, la propriété 5 et la propriété d'involution de la loi NON.

Nous avons d'une part :

$$\downarrow\left(\prod_{i=1}^{n} u_i\right) = \downarrow\left(\overline{\overline{\prod_{i=1}^{n} u_i}}\right) = \uparrow\left(\sum_{i=1}^{n} \overline{u_i}\right) = \sum_{i=1}^{n}\left(\uparrow\overline{u_i} \cdot \prod_{(j=1),\,(j\neq i)}^{n} \uparrow\overline{u_j} + \left(\overline{\overline{u_j}} \cdot \overline{\downarrow\overline{u_j}}\right)\right)$$

$$= \sum_{i=1}^{n}\left(\downarrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \downarrow u_j + \left(u_j \cdot \overline{\uparrow u_j}\right)\right)$$

et d'autre part :

$$\downarrow\left(\prod_{i=1}^{n} u_i\right) = \downarrow\left(\overline{\overline{\prod_{i=1}^{n} u_i}}\right) = \uparrow\left(\sum_{i=1}^{n} \overline{u_i}\right) = \prod_{i=1}^{n}\left(\uparrow\overline{u_i} + \left(\overline{\overline{u_i}} \cdot \overline{\downarrow\overline{u_i}}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(\overline{\overline{u_i}} \cdot \overline{\downarrow\overline{u_i}}\right)}$$

$$= \prod_{i=1}^{n}\left(\downarrow u_i + \left(u_i \cdot \overline{\uparrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(u_i \cdot \overline{\uparrow u_i}\right)}$$

C.Q.F.D.

Propriété 10 :

$$\downarrow\left(\sum_{i=1}^{n} u_i\right) = \sum_{i=1}^{n}\left(\downarrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \overline{u_j}\right)$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3, la propriété 4 et la propriété d'involution de la loi NON.

$$\downarrow\left(\sum_{i=1}^{n} u_i\right) = \downarrow\overline{\overline{\left(\sum_{i=1}^{n} u_i\right)}} = \uparrow\left(\prod_{i=1}^{n} \overline{u_i}\right) = \sum_{i=1}^{n}\left(\uparrow\overline{u_i} \cdot \prod_{(j=1),\,(j\neq i)}^{n} \overline{u_j}\right)$$

$$= \sum_{i=1}^{n}\left(\downarrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \overline{u_j}\right)$$

C.Q.F.D.

3.3.2.3. Composition des lois fronts

Propriété 11 :

$$\uparrow(\uparrow u) = \uparrow u$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 2 et la propriété 4.

$$\uparrow(\uparrow u) = \uparrow(u \cdot (\uparrow u)) = (\uparrow u \cdot \uparrow u) + (u \cdot \uparrow(\uparrow u))$$
$$= \uparrow u + u \cdot \uparrow(\uparrow u) \cdot \uparrow u = \uparrow u \cdot (1 + \uparrow(\uparrow u)) = \uparrow u$$

C.Q.F.D.

Propriété 12 :

$$\uparrow(\downarrow u) = \downarrow u$$

Démonstration :

Cette démonstration se fera en utilisant successivement la propriété 3 et la propriété 11.

$$\uparrow(\downarrow u) = \uparrow(\uparrow\overline{u}) = \uparrow\overline{u} = \downarrow u$$

C.Q.F.D.

Propriété 13 :

$$\downarrow(\uparrow u) = 0^{*}$$

Démonstration :

Pour démontrer cette proposition, il est nécessaire de démontrer au préalable la proposition suivante :

$$\forall u \in \mathbb{I},\ (\forall t \in [t_1, t_2], \uparrow u(t) = 1) \Rightarrow (t_1 = t_2)$$

Démonstration de la proposition :

Par la définition de la loi FM, nous avons d'une part,

$$(\forall t \in [t_1, t_2], \uparrow u(t) = 1) \Rightarrow (\forall t \in [t_1, t_2], u(t) = 1)$$

et d'autre part,

$$(\uparrow u(t_2) = 1) \Rightarrow (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, u(t_2 - \varepsilon) = 0)$$

Ces deux implications ne sont conjointement possibles que si et seulement si $t_1 = t_2$.

Grâce à cette proposition nous pouvons maintenant écrire que :

$$\forall u \in \mathbb{I}, \forall t \in \mathbb{R}^{+*}, (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, \uparrow u(t - \varepsilon) = 0)$$

Donc

$$\forall t \in \mathbb{R}^{+*}, (\downarrow(\uparrow u))(t) = \overline{\uparrow u}(t) \wedge (\exists \varepsilon_0, \forall \varepsilon < \varepsilon_0, \uparrow u(t - \varepsilon) = 1) = 0$$

C.Q.F.D.

Propriété 14 :

$$\downarrow(\downarrow u) = 0^*$$

Démonstration :

Cette démonstration se fera en utilisant conjointement la propriété 3 et la propriété 13.

$$\downarrow(\downarrow u) = \downarrow(\uparrow \bar{u}) = 0^*$$

C.Q.F.D.

Grâce à cette algèbre, nous avons pu définir formellement la notion de fronts et identifier quatorze propriétés relatives aux opérateurs fronts et leur combinaisons. Nous disposons maintenant d'un ensemble de cinq opérateurs et des propriétés suffisantes pour être capable de calculer symboliquement toute expression élaborée à l'aide de ces cinq opérateurs.

Dans les chapitres 2 et 3 nous avons renforcé les fondements théoriques du GRAFCET en proposant une solution aux ambiguïtés qui subsistaient dans le modèle

et en formalisant la notion d'événements. Dans les trois prochains chapitres, nous nous intéressons à la génération de l'automate équivalent et à son utilisation pour la validation de grafcets. Dans le chapitre 4, nous exposons la manière dont nous allons prendre en compte les entrées ainsi que le module de calcul symbolique sur $\mathbb{I}$ qui nous permettra de générer l'automate équivalent.

Chapitre 4
Prise en compte des entrées et calcul symbolique dans Ⅱ

Dans ce chapitre, nous exposons comment nous prenons en compte les variations des entrées d'un grafcet. Nous présentons également un module de calcul symbolique sur Ⅱ, qui nous permettra d'appliquer cette prise en compte des entrées lors de la génération du graphe des situations accessibles.

4.1. La prise en compte des entrées

L'explosion combinatoire des évolutions possibles d'un grafcet est l'une des difficultés majeures auxquelles se heurte la recherche du graphe des situations accessibles. Si cette explosion combinatoire est inévitable pour certains grafcets, il est fondamental de pouvoir en reculer les limites de praticabilité. C'est pour une large part, sur la manière de prendre en compte les entrées du modèle que repose la capacité de notre méthode à traiter des grafcets de taille «raisonnable», nous paraissant ainsi porteuse d'avenir pour une utilisation industrielle.

C'est cette manière de prendre en compte les entrées que nous allons développer dans cette section.

4.1.1. Les contraintes à intégrer

Première contrainte : la nature des entrées

Les entrées et sorties de grafcets ont jusqu'à présent toujours été définies comme des variables booléennes, nous venons de montrer dans le chapitre 3 qu'il est néces-

saire de les modéliser par des **fonctions du temps continues par morceaux à valeurs booléennes** pour que les fronts puissent être définis formellement.

Cette modélisation est d'ailleurs conforme à l'usage puisque les évolutions des entrées du modèle sont le plus souvent représentées à l'aide de chronogrammes. Il convient également de souligner que les entrées d'un grafcet ne sont pas des éléments quelconque de $\mathbb{B}$ puisque le modèle suppose que deux entrées ne peuvent varier simultanément que si elles sont corrélées.

Deuxième contrainte : la dualité événement/condition présente dans les réceptivités

Le modèle GRAFCET est «événementiel» au sens où toute évolution depuis une situation stable ne peut se faire que sur occurrence d'un événement externe, événement qui est nécessairement exprimé par l'intermédiaire d'un changement de valeur d'au moins une entrée. Cependant, pour déterminer l'évolution du grafcet qui est induite par cet événement externe, il ne suffit pas de prendre en compte ce seul changement de valeurs mais il est nécessaire de s'intéresser également aux autres entrées en raison de la nature des réceptivités.

Pour le grafcet figure 17, bien que la situation stable {10} ne soit sensible qu'à l'événement $\uparrow$m, il est nécessaire de connaître de la valeur d'autres entrées (ici a et b) pour déterminer l'évolution du grafcet. Pour cette évolution, la valeur de l'entrée c n'intervient pas.

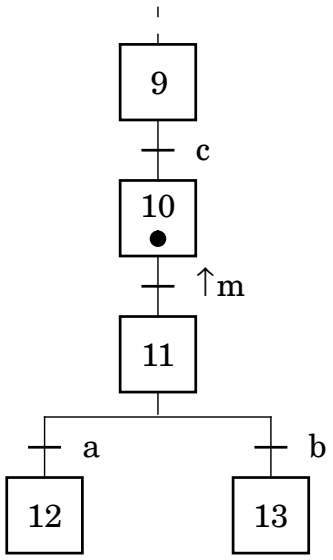


figure 17. Grafcet illustrant la difficulté de la prise en compte des entrées

Ce simple exemple illustre parfaitement la difficulté de la prise en compte des entrées pour déterminer le graphe des situations accessibles : il faut pouvoir prendre en considération la valeur d'une entrée ou son changement de valeur pour tous les éléments d'un sous-ensemble de l'ensemble des entrées, sous-ensemble qui n'est pas défini à l'avance.

Troisième contrainte : la maîtrise de l'explosion combinatoire

La troisième contrainte est de nature opérationnelle. La prise en compte des entrées doit permettre un traitement qui ne nécessite pas, en raison de leur nombre, d'inventorier toutes les combinaisons possibles pour les entrées ni ne demande de manipuler des ensembles de combinaisons.

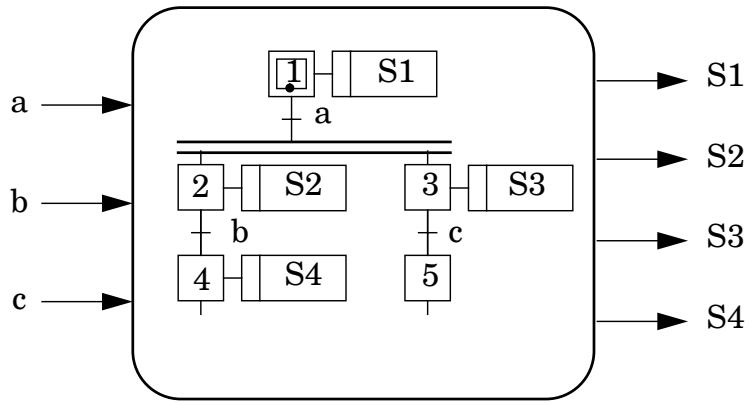
4.1.2. Notre solution

Pour générer le GSA, nous avons considéré l'**ensemble des entrées d'un grafcet** comme une **seule entité** dont la valeur varie en fonction des événements externes.

Comme toute évolution d'un grafcet est la conséquence d'un changement de valeur de cette entité «ensemble des entrées» **et** qu'il est nécessaire de connaître le changement de valeur de cette entité pour déterminer certaines évolutions, nous nous intéressons uniquement à des **ensembles de variations de l'«ensemble des entrées»** (cf. figure 18).

Il convient de préciser dès à présent (nous le montrons dans la suite de cette section) que cette manière de prendre en compte les entrées peut être réalisée sans nécessiter l'identification préalable des variations des entrées ni demander de manipulations directes d'ensembles de variations des entrées.

La pertinence de cette approche repose entièrement sur l'existence d'une représentation commune pour décrire une variation des entrées, pour définir la règle d'appartenance d'un tel élément à un ensemble et pour réaliser des opérations sur ces ensembles.



Entité «ensemble des entrées» : {a, b, c}

Exemples de variations de l'entité «ensemble des entrées» :

$((a = 1), (b = 1), (c \leftarrow 1))$ $((a = 0), (b \leftarrow 1), (c = 1))$

$((a \leftarrow 1), (b = 1), (c = 0))$ $((a = 0), (b = 0), (c \leftarrow 1))$

Exemples d'ensembles de variations de l'entité «ensemble des entrées» :

$\{ ((a = 1), (b = 1), (c \leftarrow 1)), ((a = 0), (b = 0), (c \leftarrow 1)) \}$

$\{ ((a \leftarrow 1), (b = 1), (c = 0)), ((a = 0), (b = 0), (c \leftarrow 1)) \}$

figure 18. entrées, variations des entrées, ensembles de variations des entrées

4.1.2.1. Représentation d'une variation d'entrées

Postulat 1 :

Tout changement de valeur de l'«ensemble des entrées» peut être représentée de manière unique par une expression définie sur $\mathbb{I}$ établie à l'aide de toutes les variables d'entrées et des quatre opérateurs ET, NON, FM et FD.

L'expression définie sur $\mathbb{I}$ qui caractérise un changement de valeur de l'«ensemble des entrées» est tout simplement un produit de termes correspondant chacun à une des entrées. Ce terme précise si l'entrée a évolué (il est alors égal à " $\uparrow$e " ou " $\downarrow$e ") ou si l'entrée est restée constante (il est alors égal à "e" ou "$\bar{e}$").

Ainsi, si $\{a_1, \ldots, a_n\}$ sont les n entrées d'un grafcet, toutes les expressions correspondant au diagramme syntaxique décrit figure 19 dans laquelle seul un des opérateurs fronts est employé et seulement une seule fois, représente de manière unique un changement de valeur de ces entrées (pour la figure 19, nous avons utilisé la représentation graphique des diagrammes syntaxiques employés en informatique pour représenter des grammaires algébriques [Marchand 89]).

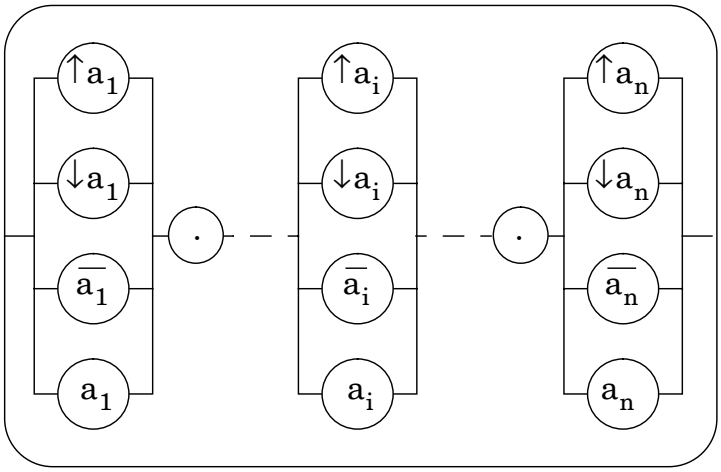


figure 19. Diagramme de syntaxe des expressions qui caractérisent une variation des entrées {a_1, ... , a_n}

Remarque :

Dans la suite du document, nous nous permettrons l'abus suivant : nous ne ferons plus la distinction entre la variation des entrées et l'expression qui la représente.

La forme de ces expressions est une «extension» des intersections canoniques des fonctions logiques [Girard 73] [Naslin 70].

Exemple :

Soient {a, b, c} les trois entrées d'un grafcet. Les 24[1] changements de valeur de l'ensemble des entrées sont les suivants :

↑a · b̄ · c̄	ā · ↑b · c̄	ā · b̄ · ↑c	↓a · b̄ · c̄	ā · ↓b · c̄	ā · b̄ · ↓c
↑a · b · c̄	a · ↑b · c̄	a · b̄ · ↑c	↓a · b · c̄	a · ↓b · c̄	a · b̄ · ↓c
↑a · b · c	a · ↑b · c	a · b · ↑c	↓a · b · c	a · ↓b · c	a · b · ↓c
↑a · b̄ · c	ā · ↑b · c	ā · b · ↑c	↓a · b̄ · c	ā · ↓b · c	ā · b · ↓c

1. En effet, pour n entrées il existe $n.2^n$ évolutions possibles pour les entrées (depuis chacune des 2^n valeurs de l'ensemble des entrées, seuls n évolutions sont possibles puisque deux entrées ne peuvent changer d'état simultanément).

4.1.2.2. Représentation en compréhension d'un ensemble de variations des entrées

Dans [Marchand 89], l'auteur rappelle qu'un ensemble d'éléments peut être décrit en extension (liste exhaustive des éléments de l'ensemble) mais aussi en compréhension (*soit p(x) un prédicat en la variable x, l'ensemble A de tous les objets x rendant vraie la proposition p(x) s'écrit, en compréhension* $A = \{x \mid p(x)\}$. *La proposition p(x) exprime une condition nécessaire et suffisante pour que l'objet x appartienne à l'ensemble A)*

Postulat 2 :

Tout ensemble de variations de l'entité «ensemble des entrées» peut être décrit en compréhension à l'aide d'une expression définie sur $\mathbb{I}$.

Dans le cas où A est un ensemble de variations des entrées défini en extension, il est toujours possible de définir en compréhension cet ensemble. Le prédicat en la variable x qui permet une notation de cet ensemble en compréhension est élaboré à partir d'une expression sur $\mathbb{I}$, établie à l'aide des variables d'entrées et des cinq opérateurs.

Prenons par exemple l'ensemble $A = \{x_1, x_2, \ldots, x_n\}$ ou chaque élément est une variation d'entrées. Le prédicat $p(x)$ qui permet de décrire cet ensemble en compréhension est :

$$p(x) : \bar{x} + P = 1^* \quad \text{avec } P = \sum_{i=1}^{n} x_i$$

Exemple :

Soit E l'ensemble de tous les changements de valeurs des trois entrées {a, b, c} d'un grafcet (cf. page précédente).

soit $A = \{\uparrow c \cdot a \cdot b, \uparrow c \cdot \bar{a} \cdot b, \uparrow b \cdot \bar{a} \cdot c, \uparrow b \cdot a \cdot c\}$ un sous-ensemble de E.

A peut être défini en compréhension de la manière suivante :

$$A = \{x \in E \mid \bar{x} + (\uparrow c \cdot b + \uparrow b \cdot c) = 1^*\}$$

$$\text{nota : } \uparrow c \cdot a \cdot b + \uparrow c \cdot \bar{a} \cdot b + \uparrow b \cdot \bar{a} \cdot c + \uparrow b \cdot a \cdot c = \uparrow c \cdot b + \uparrow b \cdot c$$

En effet, pour tous les éléments de A l'expression $\bar{x} + (\uparrow c \cdot b + \uparrow b \cdot c)$ est égale à 1^*. Pour tous les éléments de $E - A$ l'expression est différente de 1^*.

4.1.2.3. Opérations ensemblistes réalisées sur des ensembles de variations des entrées

Soient $A = \{x \mid p(x)\}$ et $B = \{x \mid q(x)\}$ deux ensembles de variations des entrées. Soient P et Q les deux expressions définies sur $\mathbb{I}$ à partir desquelles les deux prédicats $p(x)$ et $q(x)$ sont définis.

Tous les ensembles qui résultent d'une opération ensembliste entre A et B sont également définissables en compréhension car il existe toujours une expression définie sur $\mathbb{I}$, écrite à l'aide des expressions P et Q, à partir de laquelle il est possible d'établir le prédicat nécessaire à la définition en compréhension. Ces prédicats s'obtiennent ainsi :

$$A \cap B = \{x \mid (p(x) \wedge q(x))\} = \{x \mid p_1(x)\} \qquad p_1(x) : \bar{x} + (P \cdot Q) = 1^*$$

$$A \cup B = \{x \mid (p(x) \vee q(x))\} = \{x \mid p_2(x)\} \qquad p_2(x) : \bar{x} + (P + Q) = 1^*$$

$$A - B = \{x \mid (p(x) \wedge \neg q(x))\} = \{x \mid p_3(x)\} \qquad p_3(x) : \bar{x} + (P \cdot \bar{Q}) = 1^*$$

Il est en outre possible de vérifier qu'un ensemble est vide ou qu'il est inclus dans un autre en analysant les expressions sur $\mathbb{I}$ qui les définissent :

$$\left(A = \varnothing\right) \equiv \left(P = 0^*\right) \qquad \left(A \subset B\right) \equiv \left(\bar{P} + Q = 1^*\right)$$

4.1.3. L'intérêt de cette solution

L'intérêt majeur de cette forme de prise en compte les entrées est de permettre l'utilisation de la théorie des ensembles pour déterminer les conditions de franchissement des transitions simultanément franchissables. Pour étayer cette idée, nous nous appuierons sur le grafcet de la figure 20.

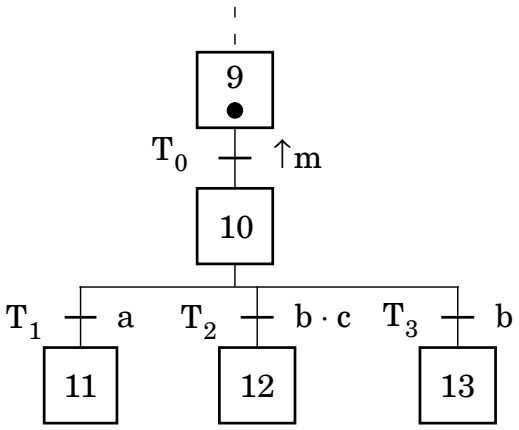


figure 20. Grafcet montrant la nature ensembliste du problème de la détermination des transitions simultanément franchissables

Sur la figure 21 sont représentés les ensembles de variations qui correspondent à l'analyse des possibilités d'évolution depuis la situation {10}, situation qui a été atteinte depuis la situation stable {9}.

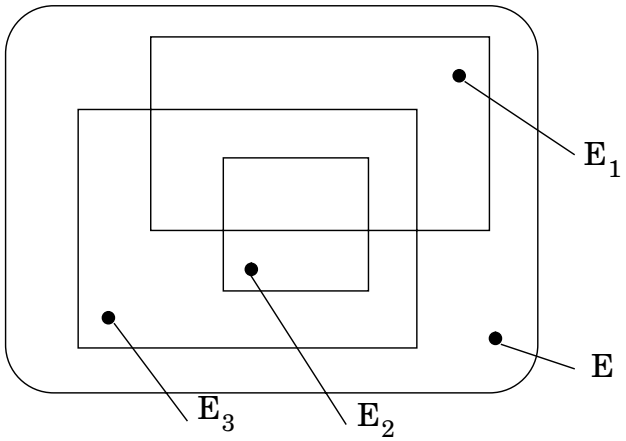


figure 21. Illustration de la nature ensembliste du problème de la détermination des transitions simultanément franchissables

Les quatre ensembles de variations des entrées sont définis de la manière suivante :

- E : Ensemble des variations des entrées qui permettent d'atteindre la situation {10} depuis la situation stable {9},

$$E = \{x \mid \bar{x} + \uparrow m = 1^*\}$$

- E_1 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_1 depuis la situation {10},

$$E_1 = \{x \mid \bar{x} + \uparrow m \cdot a = 1^*\}$$

- E_2 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_2 depuis la situation {10},

$$E_2 = \{x \mid \bar{x} + \uparrow m \cdot b \cdot c = 1^*\}$$

- E_3 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_3 depuis la situation {10}.

$$E_3 = \{x \mid \bar{x} + \uparrow m \cdot b = 1^*\}$$

En utilisant les lois de composition des ensembles, il est facile d'identifier les ensembles de variations des entrées pour lesquelles :

- la situation {10} est stable :

$$E - (E_1 \cup E_2 \cup E_3) = E - (E_1 \cup E_3) = \{x \mid \bar{x} + \uparrow m \cdot \bar{a} \cdot \bar{b} = 1^*\}$$

- seule la transition T_1 est franchie :

$$E_1 - (E_2 \cup E_3) = E_1 - E_3 = \{x \mid \bar{x} + \uparrow m \cdot a \cdot \bar{b} = 1^*\}$$

- seule la transition T_3 est franchie :

$$E_3 - (E_1 \cup E_2) = \{x \mid \bar{x} + \uparrow m \cdot \bar{a} \cdot b \cdot \bar{c} = 1^*\}$$

- seules les transitions T_1 et T_3 sont franchies :

$$(E_1 \cap E_3) - E_2 = \{x \mid \bar{x} + \uparrow m \cdot a \cdot b \cdot \bar{c} = 1^*\}$$

- seules les transitions T_2 et T_3 sont franchies :

$$(E_3 \cap E_2) - E_1 = \{x \mid \bar{x} + \uparrow m \cdot \bar{a} \cdot b \cdot c = 1^*\}$$

- les trois transitions sont franchies :

$$E_1 \cap E_2 \cap E_3 = \{x \mid \bar{x} + \uparrow m \cdot a \cdot b \cdot c = 1^*\}$$

Tout l'intérêt de notre solution pour la prise en compte des entrées réside dans le fait que nous allons bénéficier de l'utilisation de la théorie des ensembles (pour déterminer les conditions de franchissement des transitions simultanément franchissables) sans en avoir les inconvénients (l'identification en extension des ensembles et la manipulation directe des ensembles de variations des entrées).

Nous venons de montrer que les opérations de composition d'ensembles pouvaient être réalisées à partir des expressions établies sur $\mathbb{I}$ qui définissent les ensembles en compréhension. Nous montrerons au prochain chapitre, comment l'expression qui caractérise un ensemble peut être obtenue à partir des réceptivités d'un grafcet. Si nous sommes capables d'effectuer du calcul symbolique pour les expressions définies sur $\mathbb{I}$, nous pourrons alors utiliser la théorie des ensembles pour déterminer le graphe des situations accessibles.

Un module opérationnel de calcul symbolique pour les expressions définies sur $\mathbb{I}$ étant nécessaire pour développer et simplifier toutes les expressions que nous serons amenés à établir à partir des réceptivités d'un grafcet, les travaux relatifs à son élaboration font l'objet de la deuxième section de ce chapitre.

4.2. Module de calcul symbolique des expressions combinatoires sur $\mathbb{I}$

Au chapitre 3 nous avons décrit un cadre algébrique permettant la définition rigoureuse des opérateurs fronts. Dans cette section, nous présentons un module de calcul formel[1] mis au point pour simplifier toute expression utilisée en GRAFCET définie sur $\mathbb{I}$. Pour ce faire, ce module s'appuie sur des propriétés communes à toute algèbre de Boole (telle que $a + \bar{a} = 1^*$), mais également sur les propriétés établies au chapitre précédent (telle que $\uparrow\bar{a} = \downarrow a$) et prend également en compte des propriétés propres au modèle GRAFCET (telles que $\uparrow a \cdot \uparrow X1 = 0^{*2}$, $\uparrow a \cdot \uparrow b = 0^{*3}$).

Nous attendons de ce module qu'il soit capable d'établir automatiquement la forme simplifiée de toute expression écrite avec un enchaînement quelconque des cinq opérateurs définis sur $\mathbb{I}$.

En raison des liens étroits qui lient l'algèbre de Boole «étendue» que nous proposons au chapitre 3 et l'Algèbre de Boole classique, nous débuterons cette section par un bref état de l'art des méthodes de simplification en logique combinatoire pour identifier parmi leurs concepts ceux qui pourront être utilisés.

4.2.1. Bref état de l'art des méthodes de simplification en logique combinatoire

L'analyse d'expressions construites à l'aide de variables propositionnelles et de connecteurs interpropositionnels tels que ET, OU, NON n'est pas une préoccupation nouvelle puisque les bases mathématiques datent du XIX$^{\text{ième}}$ siècle tandis que certains concepts sont même dus à Aristote. Pour P. Gochet et P. Gribomont, les notions fondamentales de la logique ne peuvent être séparées des théories philosophiques qui les ont vues naître. Ils consacrent ainsi deux chapitres de leur ouvrage «Logique : méthodes pour l'informatique fondamentale» [Gochet 91], aux liens étroits qui lient philosophie et logique.

---

1. Le terme de «calcul formel» est celui retenu en informatique [Davenport 87].

2. L'occurrence d'un événement externe n'est jamais simultanée à l'occurrence d'un événement interne (cf. chapitre 2).

3. l'occurrence de deux événements externes est rejetée par le modèle.

Parmi les fondateurs de la logique moderne, le mathématicien G. Boole (1815-1864) auteur de «The mathematical Analysis of Logic» est le plus célèbre. L'algèbre qu'il proposa permit d'exprimer des raisonnements philosophiques comme les quatre propositions fondamentales d'Aristote au moyen d'équations et il montra que la validité ou la non-validité des syllogismes pouvaient être établies par un calcul algébrique sur ces équations.

Le système de Boole, comportant certains défauts, fut complété par ces successeurs. La version de l'Algèbre de Boole que nous connaissons actuellement est due à Schröder [Gochet 91]. Aujourd'hui, les travaux de Boole ne sont pas seulement la base de la logique ou de l'analyse combinatoire mais également celle de la théorie des ensembles.

Comme la logique des propositions permettait de valider des raisonnements, elle fut la source de nombreuses études qui débouchèrent sur différentes méthodes déductives pour le calcul des propositions. Dans le seul domaine de la logique, P. Gochet et P. Gribomont dans [Gochet 91] ne recensent pas moins de huit méthodes déductives pour le calcul des propositions. Il s'agit de :

- la méthode des tableaux matriciels (Peirce, Wittgenstein),
- la méthode axiomatique (Fredge, Witehead, et Russel),
- la méthode de déduction naturelle (Gentzen, Jaskowski),
- la méthode des séquents (Gentzen),
- la méthode des tableaux sémantiques (Beth, Hintikka, Smullmyan),
- la méthode des connections et des variantes (Bibel, Wallen),
- la méthode de réduction à la forme normale (Hilbert et Ackermann),
- la méthode de résolution (Robinson).

Pour leurs parts, N. Permingeat et D. Claude évoquent dans [Permingeat 91], six méthodes employées en mathématiques ou en analyse combinatoire :

- les diagrammes de Venn,
- les diagrammes de Veitch,
- les cartes de Karnaugh,
- la méthode de Quine,
- la méthode des consensus,
- la grille de Mac Cluskey.

Il faut cependant reconnaître que ces parmi ces 14 méthodes certaines sont très voisines.

Ces méthodes peuvent être classées en deux catégories en fonction de leur objectif. Dans la première catégorie, sont regroupées toutes les méthodes qui se limitent à évaluer une proposition pour la classer parmi les tautologies, les contradictions ou les formes non définies et dans la seconde catégorie sont placées les méthodes qui recherchent la forme indéfinie minimale équivalente. Les méthodes utilisant les tableaux matriciels, les séquents, les tableaux sémantiques et les diagrammes de Venn ainsi que la méthode des connections, de réduction à la forme normale ou de résolution font toutes partie de la première catégorie.

Si l'objectif principal du module de calcul formel que nous recherchons est de prouver qu'une expression est toujours nulle, nous attendons également qu'il établisse automatiquement la forme minimale équivalente en cas d'échec. Nous avons besoin de cette forme simplifiée car sa compacité améliore la lisibilité de l'expression lors d'une exploitation manuelle des résultats et offre aussi un gain de performance lors d'un traitement automatique ultérieur. Pour cette raison, nous ne pouvons nous satisfaire des techniques de la première catégorie.

Parmi les méthodes de la seconde catégorie, certaines nécessitent de dresser la liste exhaustive des combinaisons que l'on peut rencontrer pour les variables. C'est le cas des diagrammes de Veitch, des tableaux de Karnaugh, ou de la grille de Mac Cluskey. Comme nous nous refusons à dresser toute liste exhaustive des combinaisons pour échapper à l'explosion combinatoire qui en résulte, nous ne pouvons pas utiliser ces méthodes.

En ce qui concerne la méthode axiomatique et la méthode de déduction naturelle, nous n'en retenons que la philosophie car elles ont été élaborées pour démontrer des raisonnements. Leur domaine d'application privilégié étant trop distant du notre, elles s'avèrent inefficaces face à notre besoin.

Des quatorze méthodes, il ne reste plus que celle de Quine ou celle des consensus, nous devons également les rejeter car elles imposent de partir de la décom-

position disjonctive (nous recherchons une méthode de simplification qui puissent s'appliquer à toutes les formes d'expressions). Nous reprenons cependant leurs concepts et les règles de simplification sur lesquelles elles reposent.

4.2.2. Démarche opératoire

Le module de calcul symbolique dont nous avons besoin, doit être capable d'établir automatiquement la forme simplifiée de toutes les expressions écrites avec un enchaînement quelconque des cinq opérateurs définis sur $\mathbb{I}$ en tenant compte de la nature des variables de l'expression (variable d'entrées ou variable d'étapes).

La démarche opératoire que nous proposons pour simplifier des expressions combinatoires sur $\mathbb{I}$ est une adaptation de celle utilisée pour simplifier manuellement les fonctions combinatoires de l'Algèbre de Boole classique.

Bien qu'en pratique seul un petit nombre de structures d'expressions sont employées de manière usuelle en GRAFCET, il en existe théoriquement une infinité de formes différentes en raison des multiples possibilités d'enchaîner les cinq opérateurs (les diagrammes de syntaxe présentés sur la figure 22 décrivent de manière récursive, l'ensemble des structures des expressions construites à l'aide des 5 opérateurs).

Pour éviter les difficultés induites par cette diversité des formes sans avoir à restreindre l'emploi de structures, il suffit que tout calcul formel d'expressions se fasse en deux temps : une phase de mise en forme de l'expression suivie de la phase de simplification. Le but de la première phase est de transformer l'expression pour qu'elle soit exprimée sous une forme qu'il est possible de simplifier. Cependant, pour que cette démarche soit valide, il est nécessaire de disposer d'une forme générique dans laquelle toute expression écrite à l'aide des cinq opérateurs doit pouvoir être ramenée. Il est également nécessaire que toutes les règles de simplification puissent toutes pouvoir s'exprimer sur cette forme générique.

Par analogie avec les méthodes de simplification en logique combinatoire, la forme générique dont nous avons besoin est la forme développée dite «somme de produits» (la figure 23 en présente les diagrammes de syntaxe).

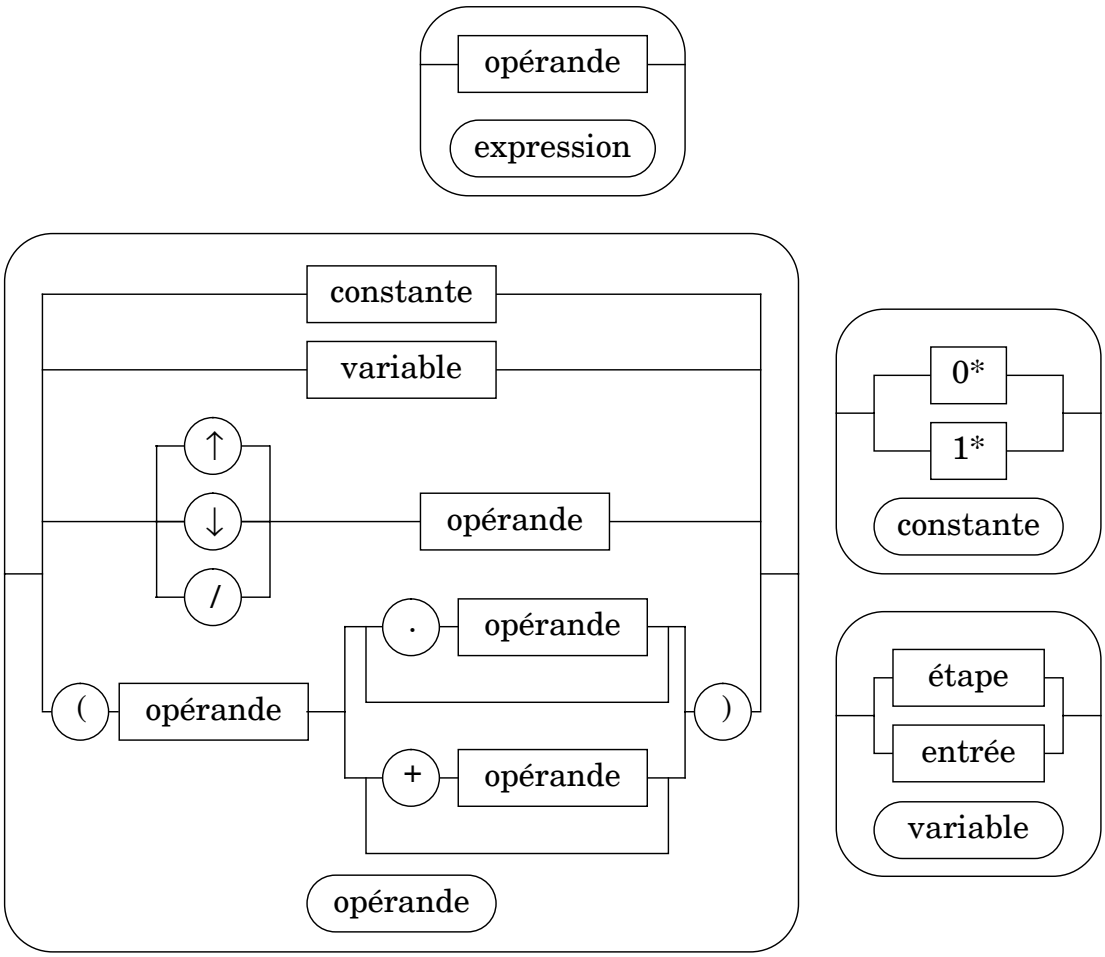


figure 22. Diagrammes de syntaxe des expressions utilisées en GRAFCET

Nous allons maintenant présenter les deux étapes du calcul formel : la phase de mise en forme qui permet de ramener toute expression dans la forme générique «somme de produits» puis la phase de simplification d'une «somme de produits».

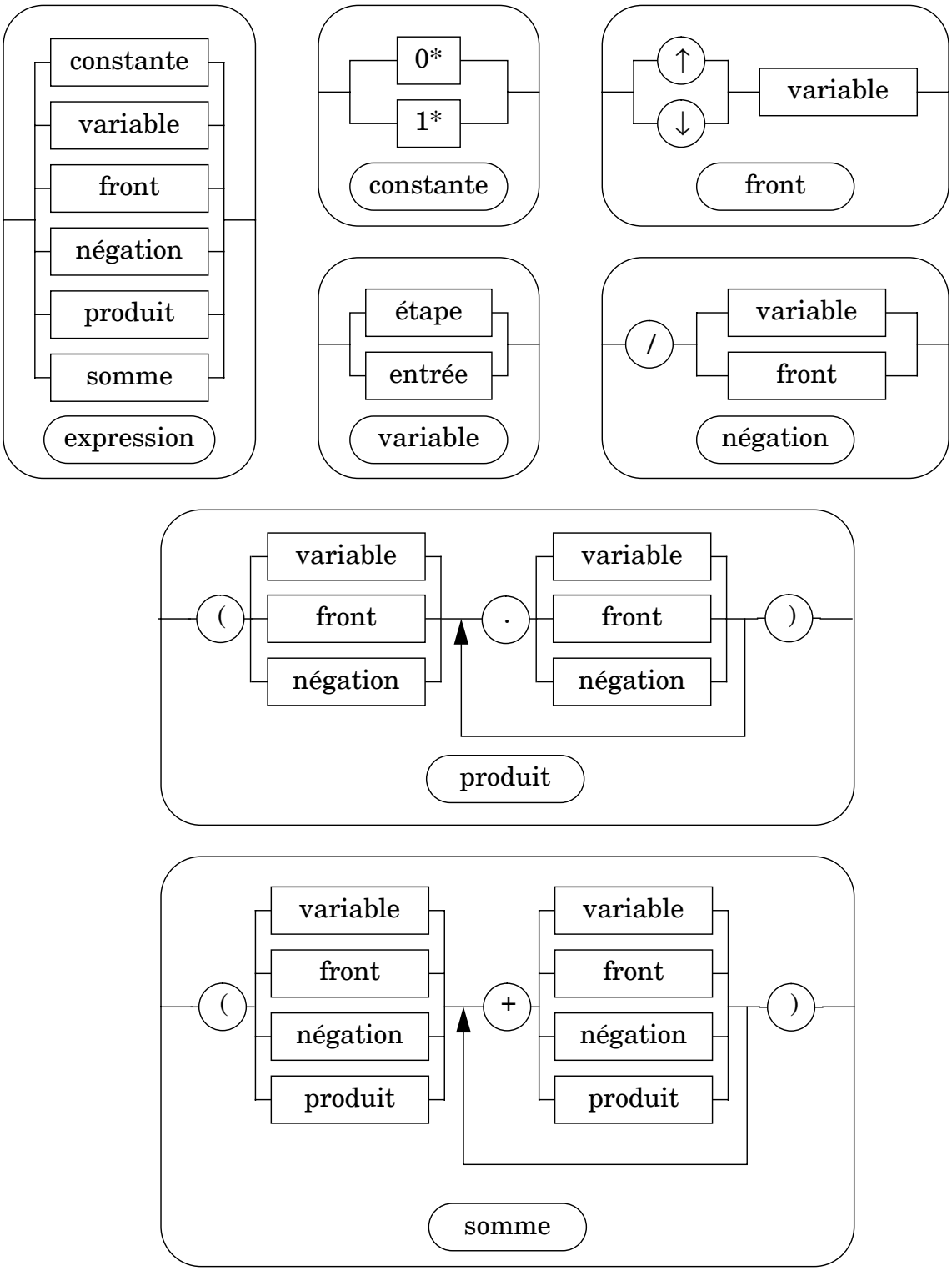


figure 23. Diagrammes de syntaxe d'une expression simplifiable

4.2.3. Mise en forme d'une expression combinatoire

Cette phase de mise en forme (cf. figure 24) consiste à transformer progressivement l'expression à analyser pour la ramener à la forme générique simplifiable. Cette phase se compose des opérations suivantes : développement des fronts, suppression des constantes, développement des négations, distributivité de l'opérateur ET par rapport à l'opérateur OU. Nous illustrerons chacune de ces étapes par le diagramme de syntaxe de la forme atteinte à l'issue de cette opération.

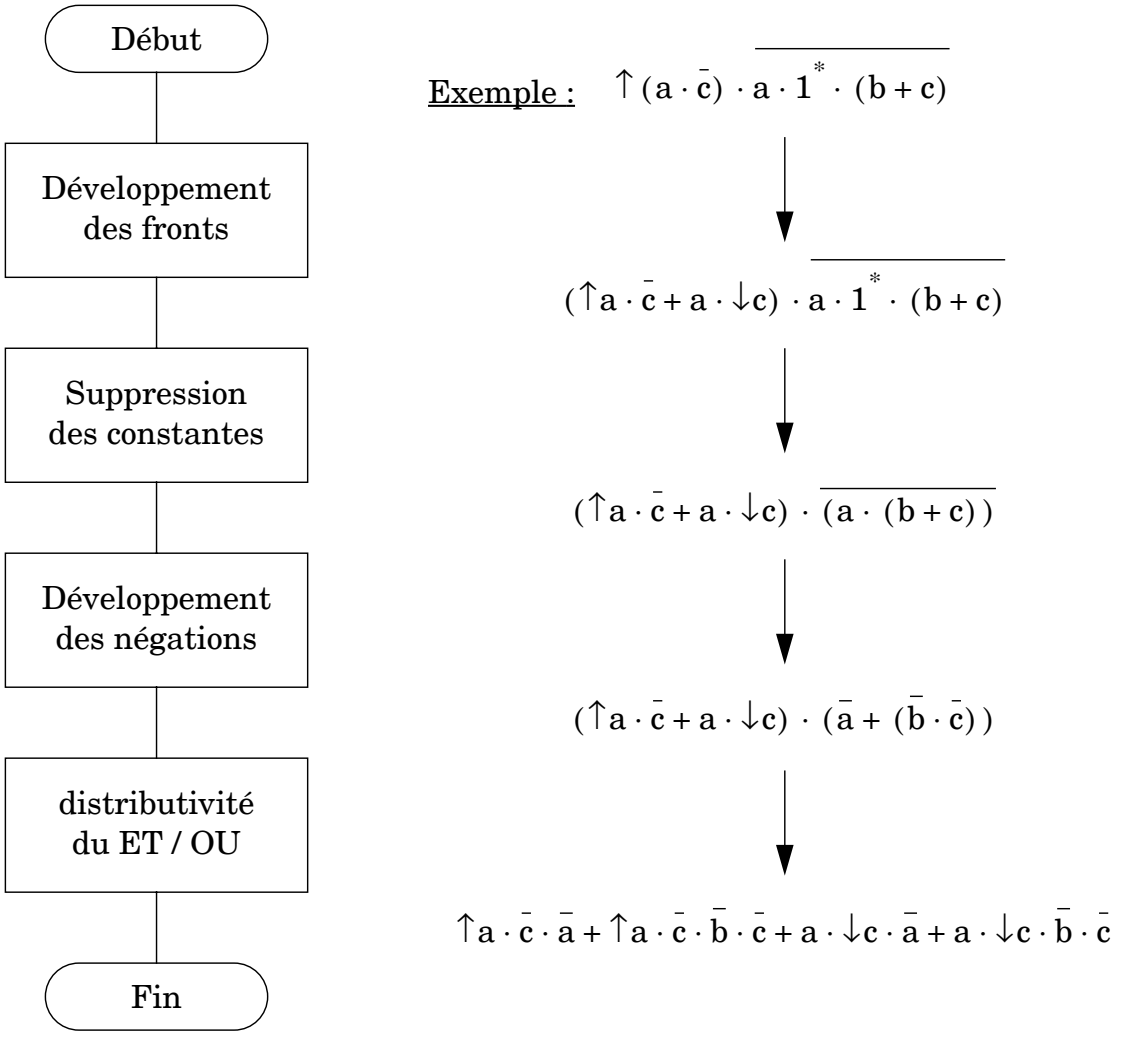


figure 24. Organigramme et exemple de la mise en forme d'une expression

Chaque opération de cette démarche systématique s'effectue de manière récursive. Une opération sera dite descendante si le traitement de l'expression débute depuis le niveau le plus haut (la racine de l'expression) tandis qu'une opération sera dite ascendante si elle débute par les niveaux les plus bas (les feuilles de

l'expression) ; une expression de Π pouvant toujours être représentée par un arbre (cf. figure 25).

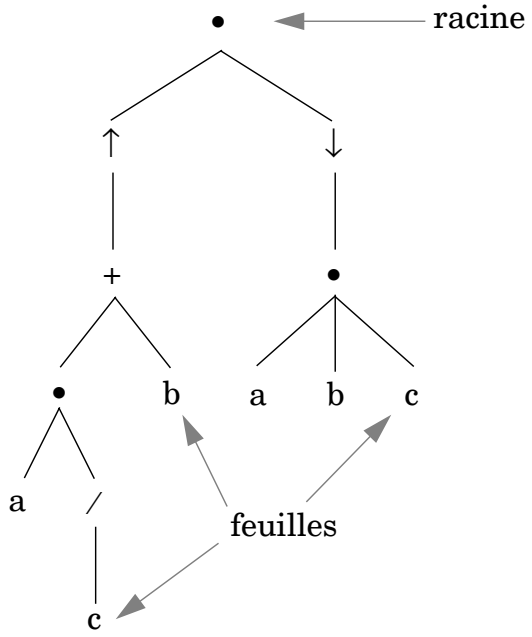


figure 25. Représentation par arbre de l'expression $\uparrow(a \cdot \bar{c} + b) \cdot \downarrow(a \cdot b \cdot c)$

4.2.3.1. Développement des fronts

Cette opération est descendante (elle débute par la racine de l'expression). Elle consiste à rechercher les niveaux de l'expression pour lesquels l'opérateur est de type front et l'opérande n'est pas une simple variable. Ce niveau de l'expression est alors remplacé par sa forme développée.

Pour cette opération, les règles employées sont les suivantes :

$$\uparrow\left(\prod_{i=1}^{n} u_i\right) = \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} u_j\right)$$

$$\uparrow\left(\sum_{i=1}^{n} u_i\right) = \prod_{i=1}^{n}\left(\uparrow u_i + \left(\bar{u_i} \cdot \overline{\downarrow u_i}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(\bar{u_i} \cdot \overline{\downarrow u_i}\right)}$$

$$\downarrow\left(\prod_{i=1}^{n} u_i\right) = \prod_{i=1}^{n}\left(\downarrow u_i + \left(u_j \cdot \overline{\uparrow u_j}\right)\right) \cdot \overline{\prod_{i=1}^{n}\left(u_j \cdot \overline{\uparrow u_j}\right)}$$

$$\downarrow\left(\sum_{i=1}^{n} u_i\right) = \sum_{i=1}^{n}\left(\downarrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \bar{u_j}\right)$$

$$\uparrow\bar{u} = \downarrow u \qquad \downarrow\bar{u} = \uparrow u$$
$$\uparrow(\uparrow u) = \uparrow u \qquad \uparrow(\downarrow u) = \downarrow u$$
$$\downarrow(\uparrow u) = 0^{*} \qquad \downarrow(\downarrow u) = 0^{*}$$
$$\uparrow 0^{*} = 0^{*} \qquad \uparrow 1^{*} = 0^{*}$$
$$\downarrow 0^{*} = 0^{*} \qquad \downarrow 1^{*} = 0^{*}$$

La figure 26 contient les diagrammes de syntaxe des expressions à l'issue de cette opération.

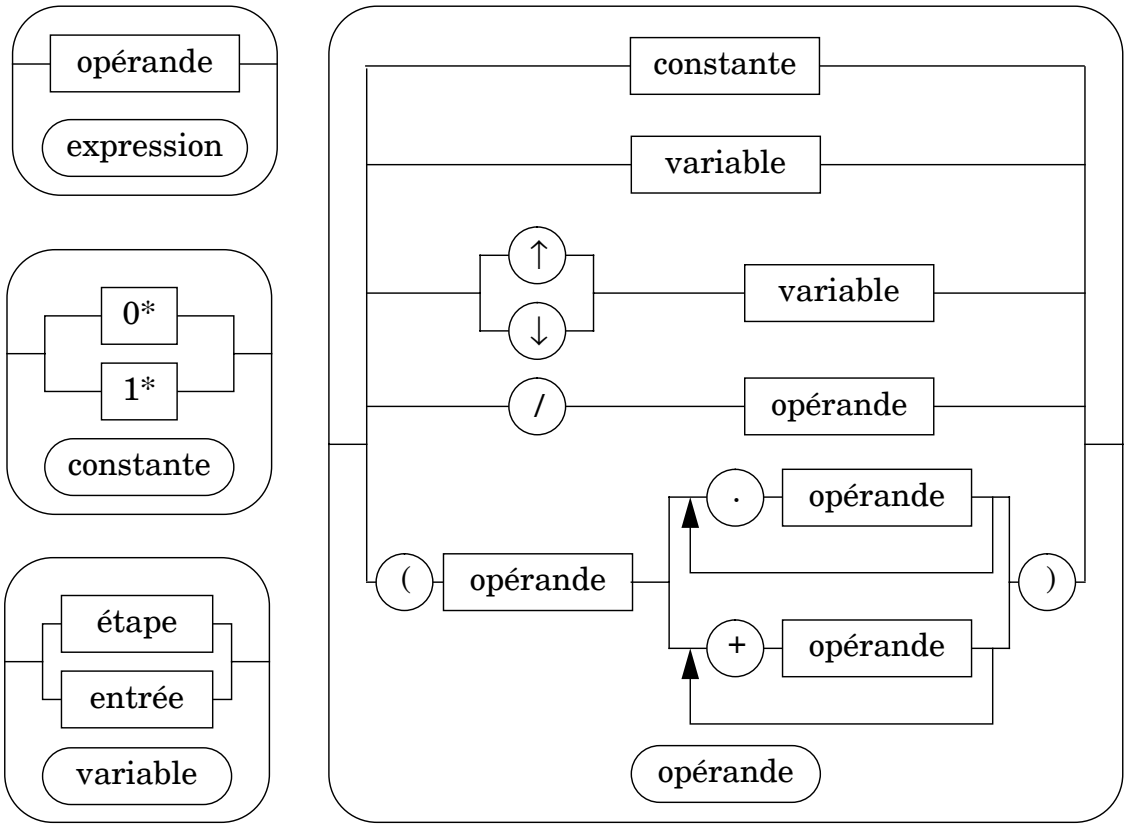


figure 26. Diagrammes de syntaxe d'une expression une fois les fronts développés

4.2.3.2. Suppression des constantes

Cette opération est ascendante (elle débute par les feuilles de l'expression). Elle consiste à rechercher dans chaque niveau de l'expression la présence d'opérandes sous forme de constantes ou réductibles à des constantes et à traiter leurs incidences

sur ce niveau. Ces opérandes sont directement supprimés de ce niveau (Eq. 5) ou permettent de réduire tout ce niveau à une constante (Eq. 6).

$$u + 0^{*} + v = u + v \qquad \text{(Eq. 5)}$$

$$u + 1^{*} + v = 1^{*} \qquad \text{(Eq. 6)}$$

Pour cette opération, les règles employées sont les suivantes :

$$0^{*} + u + v = u + v \qquad 1^{*} + u + v = 1^{*}$$

$$0^{*} \cdot u \cdot v = 0^{*} \qquad 1^{*} \cdot u \cdot v = u \cdot v$$

$$\overline{0^{*}} = 1^{*} \qquad \overline{1^{*}} = 0$$

La figure 27 contient les diagrammes de syntaxe des expressions combinatoires à l'issue de cette opération.

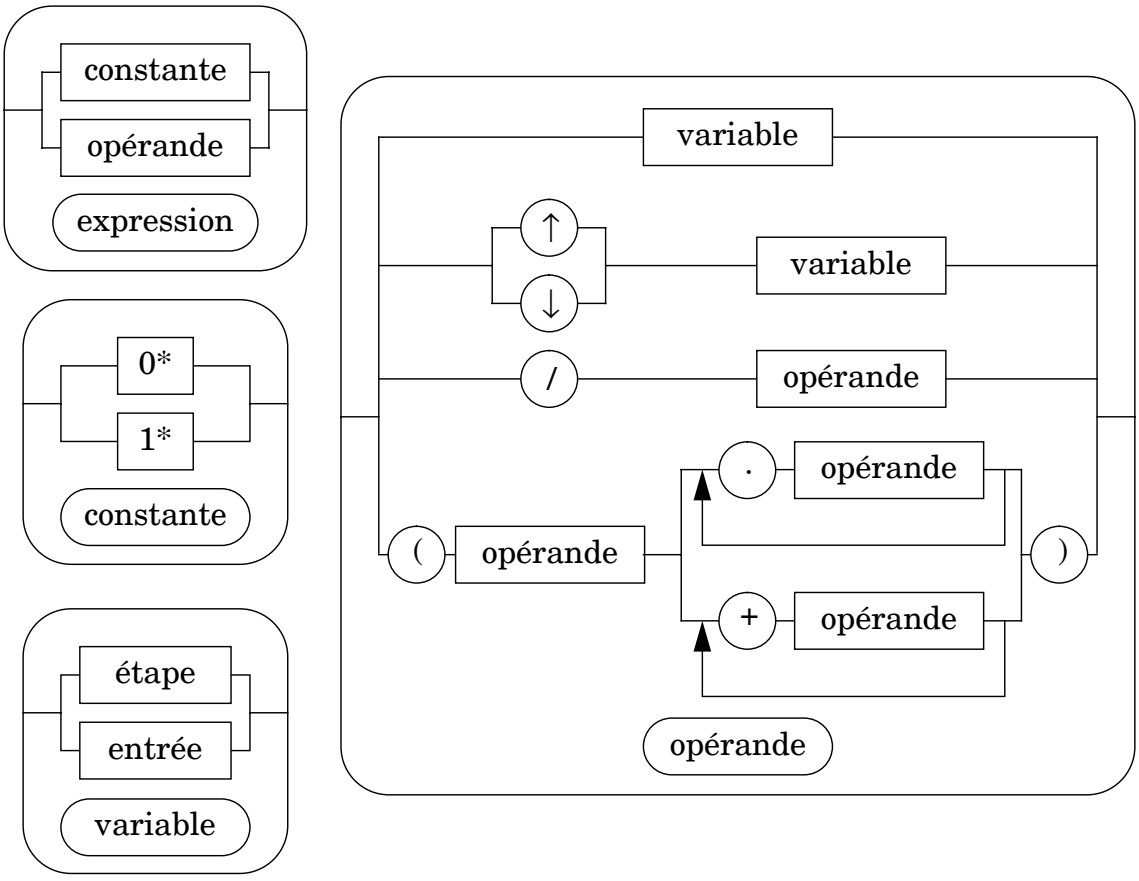


figure 27. Diagrammes de syntaxe d'une expression une fois les constantes supprimées

4.2.3.3. Développement des négations

Cette opération est descendante (elle débute par la racine de l'expression). Elle consiste à rechercher les niveaux de l'expression pour lesquels l'opérateur est NON et l'opérande n'est pas une simple variable ou le front d'une variable. Ce niveau de l'expression est alors remplacé par sa forme développée selon les règles de De Morgan :

$$\overline{\prod_{i=1}^{n} u_i} = \sum_{i=1}^{n} \overline{u_i} \qquad \overline{\sum_{i=1}^{n} u_i} = \prod_{i=1}^{n} \overline{u_i}$$

La figure 28 contient les diagrammes de syntaxe des expressions à l'issue de cette opération.

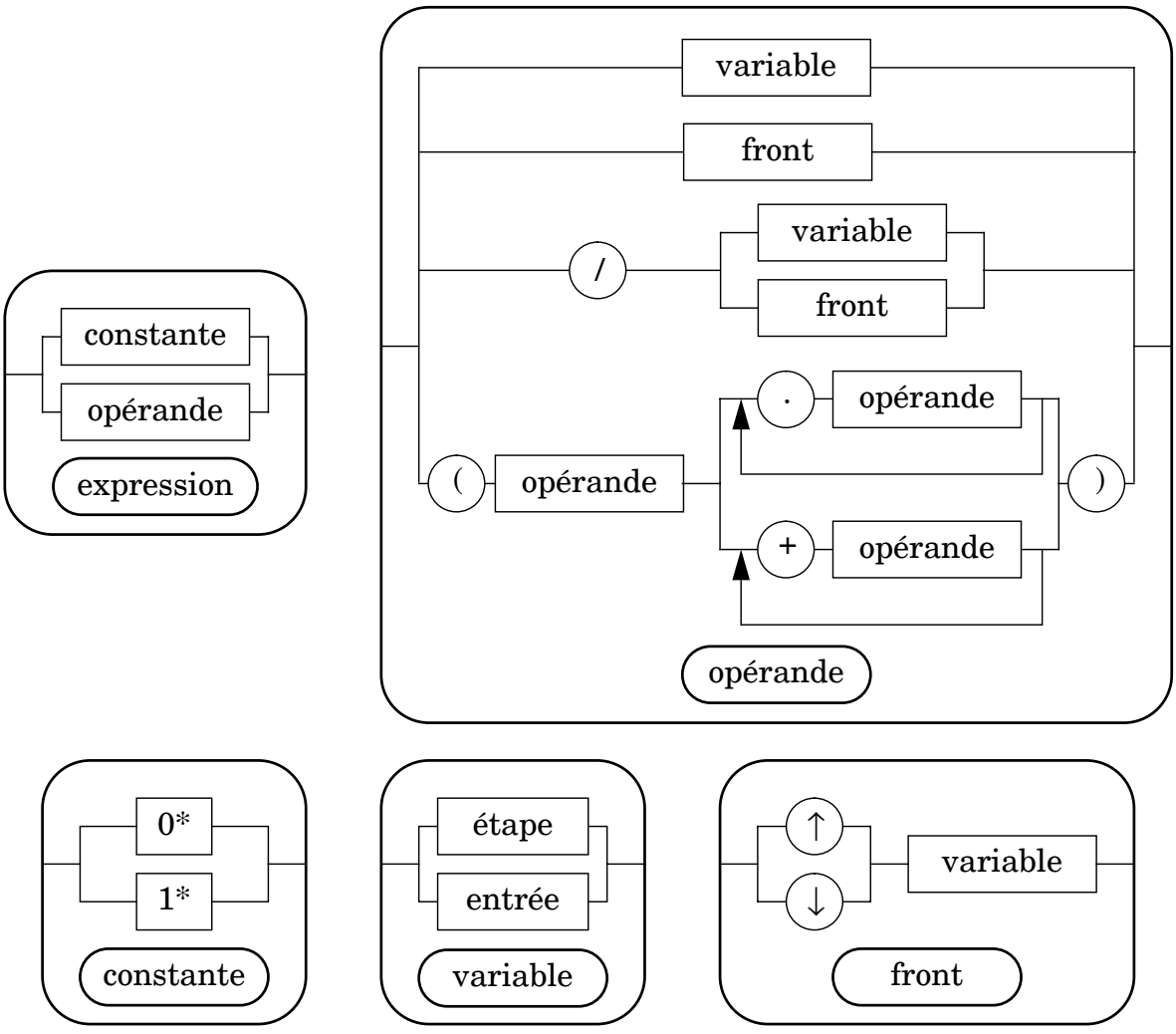


figure 28. Diagrammes de syntaxe décrivant une expression combinatoire à la suite de l'application des théorèmes de De Morgan

4.2.3.4. Distributivité de l'opérateur ET par rapport à l'opérateur OU

Cette opération est ascendante (elle débute par les feuilles de l'expression). Elle consiste à rechercher les niveaux de l'expression correspondant à un produit dont l'un de ses opérandes est une somme. Ce niveau de l'expression est alors remplacé par sa forme développée. La figure 23, page 96 contient les diagrammes de syntaxe des expressions à l'issue de cette opération.

A l'issue de cette phase de mise en forme, toutes les expressions présentent maintenant une structure en «somme de produits». C'est sur cette forme générique d'expressions que s'effectuera la simplification.

4.2.4. Simplification de l'expression

La phase de simplification est également faite en deux temps, elle débute par la simplification de chaque produit puis celle de la somme de ces produits.

4.2.4.1. Simplification d'un produit

La simplification de chaque produit demande d'analyser deux à deux tous ses opérandes pour identifier s'il existe des possibilités de réduire tout le produit à la constante nulle (Eq. 7) ou de faire disparaître un des opérandes (Eq. 8).

$$u \cdot \bar{u} = 0^{*} \qquad \text{(Eq. 7)}$$

$$\uparrow u \cdot u = \uparrow u \qquad \text{(Eq. 8)}$$

Afin de n'oublier aucune possibilité de simplification, il est nécessaire de dresser la liste exhaustive de toutes les combinaisons qui peuvent être rencontrées. D'un point de vue structurel, chaque opérande d'un produit peut avoir six structures différentes qui sont : u, $\bar{u}$, $\uparrow u$, $\downarrow u$, $\overline{\uparrow u}$, $\overline{\downarrow u}$ (cf. figure 23). Du point de vue de la variable traitée, il existe pour chaque couple d'opérandes 5 cas :

- cas n°1 : les deux opérandes portent sur la même entrée ($a \cdot \uparrow a$),
- cas n°2 : les deux opérandes portent sur la même étape ($X1 \cdot \uparrow X1$),
- cas n°3 : les deux opérandes portent sur deux entrées distinctes ($a \cdot \uparrow b$),
- cas n°4 : les deux opérandes portent sur deux étapes distinctes ($X1 \cdot \uparrow X2$),
- cas n°5 : un opérande porte sur une entrée et l'autre sur une étape ($a \cdot \uparrow X1$).

Le nombre total des combinaisons différentes s'élèvent à 120 (21 pour chacun des quatre premiers cas et 36 pour le cinquième cas). Avant d'envisager une à une toutes les combinaisons différentes, nous allons en réduire le nombre grâce aux trois remarques qui suivent ;

- comme il n'existe pas de règles de simplification concernant l'état ou le changement d'état de deux étapes, aucune des 21 combinaisons du cas n°4 ne peut présenter de possibilité de simplification ;
- lorsque les opérandes sont relatifs à une même variable, que cette variable soit une entrée ou une étape importe peu puisque les règles de calculs de distinguent pas la nature de la variable, les cas n°1 et 2 peuvent être traités simultanément ;
- comme les règles de calcul s'expriment de façon identique ($\uparrow a \cdot \uparrow X1 = 0^*$, $\uparrow a \cdot \uparrow b = 0^*$), il n'est pas nécessaire de distinguer le cas n°3 du cas n°5.

Grâce à ces trois remarques, il est possible de restreindre à 57 le nombre de combinaisons à envisager, initialement de 120.

Dans le tableau 1 se trouvent les 21 combinaisons différentes possibles lorsque la variable est commune aux deux opérandes.

ET	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
u	u	0^*	$\uparrow u$	0^*	-	u
$\bar{u}$	0^*	$\bar{u}$	0^*	$\downarrow u$	$\bar{u}$	-
$\uparrow u$	$\uparrow u$	0^*	$\uparrow u$	0^*	0^*	$\uparrow u$
$\downarrow u$	0^*	$\downarrow u$	0^*	$\downarrow u$	$\downarrow u$	0^*
$\overline{\uparrow u}$	-	$\bar{u}$	0^*	$\downarrow u$	$\overline{\uparrow u}$	-
$\overline{\downarrow u}$	u	-	$\uparrow u$	0^*	-	$\overline{\downarrow u}$

tableau 1 Possibilités de simplification pour le produit de deux opérandes lorsqu'ils sont relatifs à la même variable.

Dans le tableau 2 sont présentées les 36 combinaisons qu'il est possible de rencontrer lorsque la nature des opérandes est distincte ou lorsqu'il s'agit de deux entrées.

ET	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
v	-	-	-	-	-	-
$\bar{v}$	-	-	-	-	-	-
$\uparrow v$	-	-	0^*	0^*	$\uparrow v$	$\uparrow v$
$\downarrow v$	-	-	0^*	0^*	$\downarrow v$	$\downarrow v$
$\overline{\uparrow v}$	-	-	$\uparrow u$	$\downarrow u$	-	-
$\overline{\downarrow v}$	-	-	$\uparrow u$	$\downarrow u$	-	-

tableau 2 Possibilités de simplification pour le produit de deux opérandes lorsqu'ils sont relatifs à deux entrées distinctes ou relatifs à une entrée et une étape.

4.2.4.2. Simplification d'une somme

La simplification d'une somme consiste à analyser deux à deux tous ses opérandes pour identifier s'il existe des possibilités de réduire toute la somme à la constante vraie (Eq. 9), de supprimer (Eq. 10) (Eq. 11) ou de modifier (Eq. 12) un des opérandes.

$$u + \bar{u} = 1^* \qquad \text{(Eq. 9)}$$

$$\uparrow u + u = u \qquad \text{(Eq. 10)}$$

$$u + u \cdot v = u \qquad \text{(Eq. 11)}$$

$$u + \bar{u} \cdot v = u + v \qquad \text{(Eq. 12)}$$

Cependant, cette opération est beaucoup plus complexe que dans le cas du produit car les opérandes d'une somme peuvent être des produits d'opérandes. D'une façon générale, trois éventualités sont à envisager :

- aucun opérande n'est un produit (éventualité n°1),
- un seul opérande est un produit (éventualité n°2),
- les deux opérandes sont des produits (éventualité n°3).

Si aucun opérande n'est un produit (éventualité n°1), nous retrouvons les 120 combinaisons possibles dont l'étude théorique, comme précédemment peut être limitée aux 57 combinaisons pertinentes.

Dans le tableau 3 se trouvent les 21 combinaisons différentes possibles lorsque la variable est commune aux deux opérandes.

OU	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
u	u	1^*	u	-	1^*	$\overline{\downarrow u}$
$\bar{u}$	1^*	$\bar{u}$	-	$\bar{u}$	$\overline{\uparrow u}$	1^*
$\uparrow u$	u	-	$\uparrow u$	-	1^*	$\overline{\downarrow u}$
$\downarrow u$	-	$\bar{u}$	-	$\downarrow u$	$\overline{\uparrow u}$	1^*
$\overline{\uparrow u}$	1^*	$\overline{\uparrow u}$	1^*	$\overline{\uparrow u}$	$\overline{\uparrow u}$	1^*
$\overline{\downarrow u}$	$\overline{\downarrow u}$	1^*	$\overline{\downarrow u}$	1^*	1^*	$\overline{\downarrow u}$

tableau 3 Possibilités de simplification pour la somme de deux opérandes lorsqu'ils sont relatifs à la même variable.

Dans le tableau 4 sont présentées les 36 combinaisons qu'il est possible de rencontrer lorsque la nature des opérandes est distincte ou lorsqu'il s'agit de deux entrées.

OU	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
v	-	-	-	-	-	-
$\bar{v}$	-	-	-	-	-	-
$\uparrow v$	-	-	-	-	$\overline{\uparrow u}$	$\overline{\downarrow u}$
$\downarrow v$	-	-	-	-	$\overline{\uparrow u}$	$\overline{\downarrow u}$
$\overline{\uparrow v}$	-	-	$\overline{\uparrow v}$	$\overline{\uparrow v}$	1^*	1^*
$\overline{\downarrow v}$	-	-	$\overline{\downarrow v}$	$\overline{\downarrow v}$	1^*	1^*

tableau 4 Possibilités de simplification pour la somme de deux opérandes lorsqu'ils sont relatifs à deux entrées distinctes ou relatifs à une entrée et une étape.

Si seul un opérande est un produit (éventualité n°2), le deuxième opérande doit être comparé à tous les opérandes du produit pour vérifier si le produit peut être supprimé (Eq. 11) ou modifié par suppression de l'opérande analysé (Eq. 12). Pour chaque comparaison, 120 combinaisons sont possibles. En raison de la rupture de symétrie de l'expression $(f(u) \cdot P + g(v))$, l'étude théorique ne peut être limitée qu'à l'analyse de 72 combinaisons pertinentes.

Dans le tableau 5, sont regroupées les 36 combinaisons qui peuvent être rencontrées pour une expression de la forme $f(u) \cdot P + g(u)$.

OU	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
$u \cdot P$	u	$\bar{u} + P$	-	-	-	$\overline{\downarrow u}$
$\bar{u} \cdot P$	$u + P$	$\bar{u}$	-	-	$\overline{\uparrow u}$	-
$\uparrow u \cdot P$	u	-	$\uparrow u$	-	$\overline{\uparrow u} + P$	$\overline{\downarrow u}$
$\downarrow u \cdot P$	-	$\bar{u}$	-	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u} + P$
$\overline{\uparrow u} \cdot P$	-	-	$\uparrow u + P$	-	$\overline{\uparrow u}$	$\overline{\downarrow u} + P$
$\overline{\downarrow u} \cdot P$	-	-	-	$\downarrow u + P$	$\overline{\uparrow u} + P$	$\overline{\downarrow u}$

tableau 5 Possibilités de simplification de $f(u) \cdot P + g(u)$.

Dans le tableau 6, sont regroupées les 36 combinaisons qui peuvent être rencontrées pour une expression de la forme $f(v) \cdot P + g(u)$ lorsque la nature des opérandes est distincte ou lorsqu'il s'agit de deux entrées.

OU	u	$\bar{u}$	$\uparrow u$	$\downarrow u$	$\overline{\uparrow u}$	$\overline{\downarrow u}$
$v \cdot P$	-	-	-	-	-	-
$\bar{v} \cdot P$	-	-	-	-	-	-
$\uparrow v \cdot P$	-	-	-	-	-	-
$\downarrow v \cdot P$	-	-	-	-	-	-
$\overline{\uparrow v} \cdot P$	-	-	-	-	$\overline{\uparrow u} + P$	$\overline{\downarrow u} + P$
$\overline{\downarrow v} \cdot P$	-	-	-	-	$\overline{\uparrow u} + P$	$\overline{\downarrow u} + P$

tableau 6 Possibilités de simplification de $f(v) \cdot P + g(u)$ *lorsque u et v sont deux entrées distinctes ou une entrée et une étape.*

Lorsque nous sommes en présence de deux produits (éventualité n°3), il n'est pas possible de simplifier directement la somme de ces deux produits. Une mise en facteur

des opérandes communs aux deux produits doit être effectuée au préalable. En fonction de cette factorisation, trois possibilités se présentent :

- Si tous les opérandes du premier produit sont inclus dans le second produit, le second produit est supprimé (Eq. 13).

$$u \cdot v + u \cdot v \cdot w = u \cdot v \qquad \text{(Eq. 13)}$$

- Si seul un opérande du premier produit est différent des opérandes du second produit (Eq. 14), cet opérande sera alors comparé, comme pour l'éventualité n°2, à tous les opérandes du second produit qui n'ont pas été factorisés.

$$u \cdot v + u \cdot \bar{v} \cdot w = u \cdot (v + \bar{v} \cdot w) = u \cdot v + u \cdot w \qquad \text{(Eq. 14)}$$

- Pour chacun des produits, si au moins deux opérandes sont différents, aucune simplification n'est réalisée.

La modification d'un opérande étant parfois la source de nouvelles simplifications, l'analyse des opérandes d'une somme deux à deux doit donc être recommencée tant que des opérandes sont modifiés. Prenons comme exemple l'équation 15, la suppression du premier terme n'est possible qu'après la modification du second. Celle-ci est réalisée lors de l'analyse des deux derniers termes.

$$v \cdot w + \bar{u} \cdot v + u = u + v \qquad \text{(Eq. 15)}$$

4.2.5. Quelques remarques sur les résultats obtenus

4.2.5.1. Performances du module

Pour certaines expressions, cette méthode de calcul symbolique ne permet pas d'atteindre la forme simplifiée minimale. En effet, des expressions comme celle de l'équation 16 (dite théorème de redondance) ne donc sont pas simplifiables par cette méthode car la simplification n'est identifiable que par une analyse simultanée trois opérandes.

$$\bar{u} \cdot v + u \cdot w + v \cdot w = \bar{u} \cdot v + u \cdot w \qquad \text{(Eq. 16)}$$

Nous n'avons pas retenue cette option car elle se révèle trop pénalisante en temps de calcul.

Nous avons validé cette méthode par le développement d'une maquette informatique écrite en langage C.

Dans un premier temps, nous avons testé la qualité de la simplification. Cette étape a consisté à appliquer la méthode sur des expressions présentant soit une structure complexe, soit des possibilités particulières de simplification. A l'exception des rares cas nécessitant l'analyse simultanée de trois opérandes, nous n'avons pas été capable de découvrir une forme équivalente plus performante que celle proposée par la maquette.

Dans un second temps, nous nous sommes intéressés à la performance opérationnelle en analysant le temps de calcul et les besoins en taille mémoire qui sont tous deux des facteurs décisifs en calcul formel [Davenport 87]. La démarche en deux temps, consistant à mettre en forme puis seulement après, à simplifier l'expression s'est avérée pénalisante à grande échelle[1]. Nous lui avons préféré une démarche mixte dans laquelle la phase de simplification commence lors de la phase de mise en forme.

Le tableau 7 donne un aperçu des possibilités de calculs de notre module de calcul symbolique. Ces essais ont été réalisés sur station de travail SUN (modèle SPARCstation LX). Nous avons mesuré le temps mis par la machine pour lire le fichier contenant les expressions à analyser, pour effectuer les simplifications et écrire les résultats sur disque. Pour obtenir des chiffres significatifs, le fichier de tests contenait 100 fois la même expression à analyser.

Expression à analyser	Expression simplifiée	temps en ms
$(a+b)\cdot(b+c)\cdot(c+a)$	$a\cdot b+b\cdot c+c\cdot a$	8
$(b+\bar{c})\cdot(\bar{b}+c)+\overline{(\bar{a}+b+\bar{c})}$	$b\cdot c+\bar{b}\cdot\bar{c}+a\cdot\bar{b}$	9
$\uparrow(a\cdot\bar{c}+b)\cdot\downarrow(a\cdot b+c)$	$a\cdot\bar{b}\cdot\downarrow c$	120

tableau 7 Temps de calcul pour différentes expressions

1. Analyser plus de cent milles expression lors de la génération d'un graphe des situations accessibles est courant.

4.2.5.2. Prise en compte des contentions sur les entrées

En raison de son utilisation, il est intéressant d'étendre ce module de calcul symbolique pour permettre la prise en compte de contentions qui peuvent exister pour certains couples d'entrées, c'est-à-dire des couples d'entrées pour lesquelles il existe une relation de dépendance entre les valeurs. Il est nécessaire pour cela, d'introduire de nouvelles règles de simplification.

Dans l'annexe B, nous avons traité les principaux cas de contention que sont : $a = \bar{b}$, $a \cdot b = 0^*$, $a \rightarrow b$ (a implique b).

le cas de la contention $a = \bar{b}$ correspond, par exemple, au cas de deux entrées mutuellement exclusives opposées l'une à l'autre. Les deux positions d'un sélecteur bistable utilisé en MARCHE/ARRÊT respectent ce type de contention.

le cas de la contention $a \cdot b = 0^*$ correspond, par exemple, au cas de deux entrées qui ne peuvent jamais être présentes simultanément. Les deux capteurs de fin course d'un même vérin respectent ce type de contention.

le cas de la contention $a \rightarrow b$ correspond, par exemple, au cas de deux entrées pour lesquelles la présence de la première entraîne nécessairement la présence de la seconde. Ce cas de contention se rencontre lorsque plusieurs entrées sont utilisées pour la détection de seuil (lorsque l'entrée «niveau haut» est présente alors l'entrée «niveau bas» est également présente). Cette contention peut également s'exprimer par l'égalité $a \cdot \bar{b} = 0^*$.

Dans ce chapitre, nous avons présenté comment nous prenons en compte les variations des entrées d'un grafcet. Nous considérons les entrées comme un seule entité et nous nous intéressons uniquement aux ensembles de variations de cette entité ; l'intérêt étant de bénéficier de la théorie des ensembles pour le calcul des conditions de franchissement simultané des ensembles de transitions simultanément franchissables.

Le module de calcul symbolique sur $\mathbb{I}$ nécessaire pour mettre en œuvre cette technique venant d'être présenté, nous pouvons maintenant exposer dans le détail la démarche retenue pour générer automatiquement l'automate à états équivalent à un grafcet.

Chapitre 5
La génération de l'automate équivalent

Ce chapitre est consacré à la description de la méthode de génération de l'automate équivalent à un grafcet. Comme nous l'avons exposé dans le premier chapitre, cette méthode s'effectue en trois temps : détermination des évolutions à l'échelle de temps interne, construction de l'automate décrivant le comportement à l'échelle de temps externe, et réduction de cet automate pour tenir compte de l'historique des entrées.

5.1. Détermination des évolutions à l'échelle de temps interne

5.1.1. Modèle de représentation et vocabulaire employé

Pour décrire l'enchaînement des situations accessibles à l'échelle de temps interne depuis une situation stable, nous avons choisi une représentation sous forme d'arbre dans lequel chaque nœud est l'image d'une situation et chaque branche celle d'une évolution (nous avons choisi une représentation par arbre car les situations identiques ne doivent pas être regroupées).

Avant de présenter la technique mise au point pour la détermination des situations et des évolutions entre ces situations, nous allons nous attarder sur le vocabulaire que nous emploierons puisqu'il nous a été nécessaire d'apporter certaines précisions aux définitions couramment admises. Ces définitions sont données en deux temps pour plus de commodités.

Définition 4 :

Nous définissons par **situation** d'un grafcet l'ensemble des données concernant l'état du modèle nécessaires aux calculs d'évolution. Chaque situation est définie par deux ensembles : le premier est l'ensemble des étapes actives du grafcet à un instant donné, le second est l'ensemble des étapes dont l'activité vient d'être modifiée.

Nous n'avons pas pu retenir la définition courante de la situation ([Blanchard 79], [Afcet 83]) car à l'échelle de temps interne, la connaissance de l'ensemble des étapes actives d'un grafcet à un instant donné, ne suffit pas pour évaluer toutes les expressions relatives aux variables internes (par exemple, pour évaluer une expression comme $\uparrow X10$, il ne suffit pas de connaître l'activité de l'étape 10 mais il faut également savoir si l'activité de cette étape vient d'être modifiée).

Définition 5 :

Nous appelons **évolution** tout passage d'une situation à une autre, que nous distinguons en les nommant respectivement *situation amont* et *situation aval* de l'évolution.

Définition 6 :

Nous appelons **contexte** les expressions combinatoires sur $\mathbb{I}$ que nous associons à chaque situation, évolution, transition et ensemble de transitions.

Du point de vue fonctionnel, chacun de ces contextes[1] caractérise un ensemble précis de variations des entrées. Du point de vue mathématique, le contexte est l'expression sur $\mathbb{I}$ qui permet la définition en compréhension de cet ensemble (cf. chapitre 4).

1. Le terme de contexte a été repris de [Grepa 85].

Définition 7 :

A chaque situation atteinte, il est associé deux contextes :

- le **contexte résiduel (Cres)** qui décrit l'ensemble des variations des entrées qui ont permis d'atteindre cette situation,
- le **contexte de stabilité (Csta)** qui décrit la partie du contexte résiduel pour laquelle cette situation devient stable.

Définition 8 :

A chaque transition ou ensemble de transitions, il est associé deux contextes :

- le **contexte minimal (Cmin)** qui décrit l'ensemble des variations des entrées qui permettent de franchir au moins cette transition ou cet ensemble de transitions (condition nécessaire au franchissement),
- le **contexte maximal (Cmax)** qui décrit la partie du contexte minimal pour laquelle cette transition ou cet ensemble de transitions sera seul franchi (condition nécessaire et suffisante pour le franchissement).

Définition 9 :

A chaque évolution, est associé un contexte appelé **contexte d'évolution (Cévo)**. Il décrit l'ensemble des variations des entrées qui permettent cette évolution.

Pour illustrer ces définitions, nous nous servirons du grafcet présenté figure 29. La figure 30 représente les ensembles de variations qui correspondent à l'analyse des possibilités d'évolution depuis la situation {10}, situation qui a été atteinte depuis la situation stable {9}.

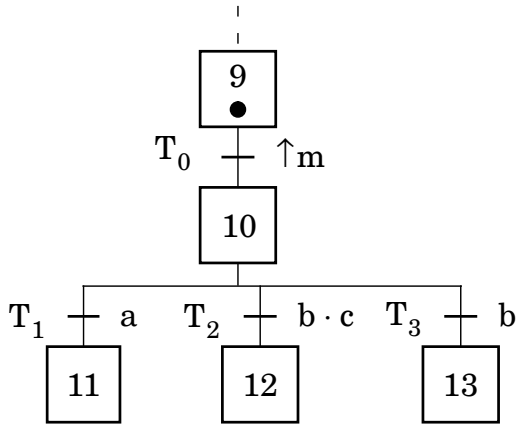


figure 29. Grafcet support de l'illustration de la notion de contexte

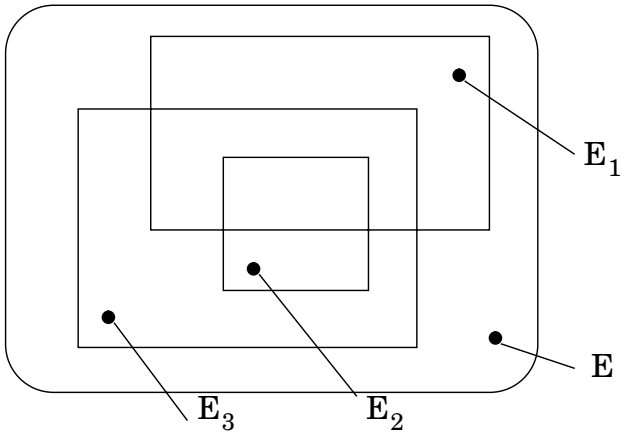


figure 30. Illustration de la notion de contexte

Les quatre ensembles de variations des entrées sont définis de la manière suivante :

- E : Ensemble des variations des entrées qui permettent d'atteindre la situation {10} depuis la situation stable {9},

$$E = \{x \mid \bar{x} + \uparrow m = 1^{*}\}$$

- E_1 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_1 depuis la situation {10},

$$E_1 = \{x \mid \bar{x} + \uparrow m \cdot a = 1^{*}\}$$

- E_2 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_2 depuis la situation {10},

$$E_2 = \{x \mid \bar{x} + \uparrow m \cdot b \cdot c = 1^{*}\}$$

- E_3 : Ensemble des variations des entrées qui permettent encore de franchir la transition T_3 depuis la situation {10}.

$$E_3 = \{ x \mid \bar{x} + \uparrow m \cdot b = 1^* \}$$

Dans le tableau 8, nous présentons les différents contextes que nous venons de définir en donnant leur expression dans $\mathbb{I}$ et l'ensemble de vairations d'entrées qu'il représente dans le cas de l'étude de la situation {10} du grafcet figure 29.

Contexte	expression dans $\mathbb{I}$	Ensemble représenté
contexte résiduel de {10}	$\uparrow m$	E
contexte de stabilité de {10}	$\uparrow m \cdot \bar{a} \cdot \bar{b}$	$E - (E_1 \cup E_2 \cup E_3)$
contexte minimal de $\{T_1\}$	$\uparrow m \cdot a$	E_1
contexte minimal de $\{T_2\}$	$\uparrow m \cdot b \cdot c$	E_2
contexte minimal de $\{T_3\}$	$\uparrow m \cdot b$	E_3
contexte minimal de $\{T_1, T_2\}$	$\uparrow m \cdot a \cdot b \cdot c$	$E_1 \cap E_2$
contexte minimal de $\{T_1, T_3\}$	$\uparrow m \cdot a \cdot c$	$E_1 \cap E_3$
contexte minimal de $\{T_2, T_3\}$	$\uparrow m \cdot b \cdot c$	$E_2 \cap E_3$
contexte minimal de $\{T_1, T_2, T_3\}$	$\uparrow m \cdot a \cdot b \cdot c$	$E_1 \cap E_2 \cap E_3$
contexte maximal de $\{T_1\}$	$\uparrow m \cdot a \cdot \bar{b}$	$E_1 - (E_2 \cup E_3)$
contexte maximal de $\{T_2\}$	0^*	$\varnothing$
contexte maximal de $\{T_3\}$	$\uparrow m \cdot \bar{a} \cdot b \cdot \bar{c}$	$E_3 - (E_1 \cup E_2)$
contexte maximal de $\{T_1, T_2\}$	0^*	$\varnothing$
contexte maximal de $\{T_1, T_3\}$	$\uparrow m \cdot a \cdot b \cdot \bar{c}$	$(E_1 \cap E_3) - E_2$
contexte maximal de $\{T_2, T_3\}$	$\uparrow m \cdot \bar{a} \cdot b \cdot c$	$(E_3 \cap E_2) - E_1$
contexte maximal de $\{T_1, T_2, T_3\}$	$\uparrow m \cdot a \cdot b \cdot c$	$E_1 \cap E_2 \cap E_3$

tableau 8 Exemples de contextes

Définition 10 :

Une **situation** est dite **stable** relativement à son contexte résiduel si aucune des transitions validées n'est franchissable quelles que soient les variations des entrées que décrit ce contexte.

Définition 11 :

Une **situation** est dite **instable** relativement à son contexte résiduel si au moins une des transitions validées est franchissable pour une des variations des entrées que décrit ce contexte.

Cette définition est conforme à celle proposée au chapitre 2 car ce franchissement de transitions s'effectue sans nouvelle occurrence d'événement externe.

Définition 12 :

Une **situation** est dite **totalement instable** relativement à son contexte résiduel lorsqu'elle est la situation aval d'une évolution qui s'est déjà produite pour ce contexte résiduel.

Définition 13 :

Une **situation** est dite **incohérente du point de vue des ordres de forçage** si elle comporte des étapes actives auxquelles sont associées au moins deux ordres de forçage d'un même grafcet partiel.

5.1.2. Méthode de génération

La génération des arbres des situations accessibles à l'échelle de temps interne, depuis chaque situation stable atteignable, s'effectue arbre par arbre (c'est-à-dire, situation stable par situation stable), tant qu'un arbre ne sera pas associé à chaque nouvelle situation stable atteignable. La racine de chaque arbre représente la situation stable à partir de laquelle sont calculées les évolutions à l'échelle de temps interne. Chaque élément terminal d'un arbre représente une des situations stables

atteignables, une situation totalement instable ou une situation incohérente du point de vue des ordres de forçage.

Le premier arbre généré correspond aux évolutions possibles à l'échelle interne depuis la situation initiale.

La génération des arbres s'effectue arbre par arbre et, dans un arbre donné, par niveau (il s'agit donc d'une génération en largeur et non en profondeur [Gaudel 87]). Au cours de cette génération, il est nécessaire de distinguer deux cas en fonction de la position du nœud dans l'arbre :

- Lorsque le nœud est la racine de l'arbre, il correspond à la situation stable pour laquelle nous cherchons les évolutions possibles à l'échelle de temps interne. Cette situation est réceptive à toute nouvelle occurrence d'un événement externe. Pour cette situation, il n'existe pas de contexte résiduel.

- Lorsque le nœud à analyser ne se situe pas à la racine de l'arbre, cette situation est le résultat d'une évolution précédente à l'échelle de temps interne. Pour cette situation, un contexte résiduel est défini. Il est cherché les transitions franchissables sans nouvelle occurrence d'un événement externe.

En raison de ces différences, deux techniques complémentaires sont nécessaires pour l'analyse d'une situation. Pour illustrer la démarche appliquée nous utiliserons l'extrait de grafcet présenté figure 31.

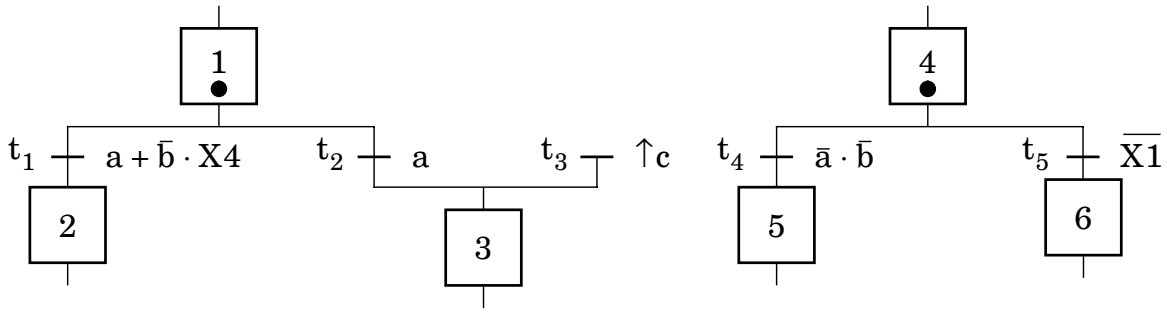


figure 31. Exemple d'illustration de la méthode de génération

5.1.3. Analyse d'une situation stable (racine d'un arbre)

Cette analyse s'effectue en trois temps. La première phase consiste à identifier toutes les transitions sensibilisées[1] depuis cette situation stable. La seconde phase est l'élaboration de tous les ensembles de transitions simultanément franchissables et de leurs contextes. La dernière étape consiste à déterminer les situations qui sont atteintes par le franchissement de l'un de ces ensembles de transitions et des évolutions correspondantes.

5.1.3.1. Détermination des transitions sensibilisées avec leur contexte minimal

La détermination de la sensibilité des transitions s'effectue uniquement pour les transitions qui n'appartiennent pas à un grafcet partiel forcé. Il est vérifié que la réceptivité associée aux transitions validées est compatible avec la situation courante. Dans la pratique, cette vérification consiste à établir le contexte minimal de la transition. Si celui-ci s'avère être nul, la transition n'est donc pas sensibilisée depuis cette situation. Le contexte minimal est quant-à-lui construit à partir de la réceptivité associée à la transition, dans laquelle toutes les expressions portant sur les valeurs des variables d'étapes ont été calculées[2] tandis que celles portant sur les changements de valeurs des variables d'étapes ont été mises à zéro[3].

Pour l'exemple de la figure 31, les seules transitions sensibilisées avec leur contexte minimal sont :

$$\left(\{t_1\}, a + \bar{b} \right), \left(\{t_2\}, a \right), \left(\{t_3\}, \uparrow c \right), \left(\{t_4\}, \bar{a} \cdot \bar{b} \right)$$

1. transition validée pour laquelle la partie de la réceptivité portant sur les variables internes est vraie [Banchard 79].

2. La situation étant entièrement définie, la valeur de chaque variable d'étape est donc connue.

3. Comme le modèle GRAFCET postule que tous les événements externes sont pris en compte dès leur occurrence et pour toutes leurs incidences, une situation ne peut pas être à la fois sensible aux événements internes et externes.

5.1.3.2. Détermination des ensembles de transitions simultanément franchissables avec leur contexte maximal

La détermination de tous les ensembles de transitions simultanément franchissables s'effectue en deux temps. Le premier temps consiste à construire de proche en proche tous les ensembles de transitions simultanément franchissables en déterminant pour chacun son contexte minimal. Si ce dernier s'avère être nul, l'ensemble est alors rejeté. Un ensemble de cardinalité n est obtenu en ajoutant une nouvelle transition dans un ensemble de cardinalité $n-1$. Le contexte minimal de cet ensemble est construit à partir de celui de l'ensemble de cardinalité $n-1$ et de celui de la transition ajoutée. Cette construction s'effectue ainsi :

$$\mathrm{Cmin}\left(\{T_i, ..., T_j, T_k\} \right) = \mathrm{Cmin}\left(\{T_i, ..., T_j\} \right) \cdot \mathrm{Cmin}\,(T_k)$$

A l'issu de cette phase, chaque ensemble de transitions est caractérisé par son contexte minimal qui définit la condition nécessaire pour permettre le franchissement de cet ensemble. Cependant, ce contexte ne correspond pas à la condition de franchissement car plusieurs ensembles de transitions peuvent être simultanément franchies, (les contextes minimaux ne sont pas mutuellement exclusifs).

Pour l'exemple de la figure 31, les ensembles de transitions simultanément franchissables avec leur contexte minimal sont :

$$\left(\{t_1\}, a + \bar{b} \right), \left(\{t_2\}, a \right), \left(\{t_3\}, \uparrow c \right), \left(\{t_4\}, \bar{a} \cdot \bar{b} \right), \left(\{t_1, t_2\}, a \right),$$

$$\left(\{t_1, t_3\}, \uparrow c \cdot (a + \bar{b}) \right), \left(\{t_1, t_4\}, \bar{a} \cdot \bar{b} \right), \left(\{t_2, t_3\}, \uparrow c \cdot a \right),$$

$$\left(\{t_3, t_4\}, \uparrow c \cdot \bar{a} \cdot \bar{b} \right), \left(\{t_1, t_2, t_3\}, \uparrow c \cdot a \right), \left(\{t_1, t_3, t_4\}, \uparrow c \cdot \bar{a} \cdot \bar{b} \right)$$

Dans un deuxième temps, il est déterminé la condition qui permet le seul franchissement de chaque ensemble de transitions simultanément franchissables. Cette condition est le contexte maximal de chaque ensemble. Pour un ensemble de cardinalité n, son contexte maximal est construit à partir de son contexte minimal et de ceux des ensembles de cardinalité $n+1$ dans lesquels il est inclus :

$$\mathrm{Cmax}\left(\{T_i, ..., T_j\} \right) = \mathrm{Cmin}\left(\{T_i, ..., T_j\} \right) \cdot \overline{\sum \mathrm{Cmin}\left(\{T_i, ..., T_j, T_k\} \right)}$$

Si la simplification de ce contexte maximal aboutit à l'élément nul, l'ensemble est rejeté.

Pour l'exemple de la figure 31, les seuls ensembles de transitions simultanément franchissables avec leur contexte maximal sont :

$$\left(\{t_3\}, \uparrow c \cdot \bar{a} \cdot b \right), \left(\{t_1, t_2\}, a \cdot \overline{\uparrow c} \right), \left(\{t_1, t_4\}, \bar{a} \cdot \bar{b} \cdot \overline{\uparrow c} \right),$$

$$\left(\{t_1, t_2, t_3\}, \uparrow c \cdot a \right), \left(\{t_1, t_3, t_4\}, \uparrow c \cdot \bar{a} \cdot \bar{b} \right)$$

5.1.3.3. Détermination des situations accessibles et des évolutions correspondantes

La troisième phase est la détermination des situations atteintes par le franchissement de l'un de ces ensembles de transitions et des évolutions correspondantes.

Pour chaque ensemble de transitions simultanément franchissables, il est d'abord déterminé quel serait l'ensemble des étapes actives à l'issue de ce seul franchissement. La situation est ensuite corrigée pour prendre en compte les ordres de forçage.

A partir de l'ensemble des étapes actives, il est recherché **par niveau de hiérarchie**[1], et pour chaque grafcet partiel s'il est soumis à un ordre de forçage. Quatre cas peuvent se présenter :

- il n'est soumis à aucun ordre de forçage, aucune correction de situation n'est à effectuer,
- il n'est soumis qu'à un ordre de forçage dans la situation courante (figeage), aucune correction de situation n'est effectuée,
- il n'est soumis qu'à un ordre de forçage dans une situation pré-définie, l'ensemble des étapes actives est modifié pour que soit satisfait l'ordre de forçage,
- Il est soumis à au moins deux ordres de forçage, la nouvelle situation calculée est déclarée comme étant incohérente du point de vue des ordres de forçage.

1. Nous supposons que la hiérarchie de forçage entre grafcets partiels est cohérente (dans [Lesage 93], nous avons proposé une technique pour vérifier cette cohérence et déterminer le niveau hiérarchique de chaque grafcet partiel).

A la suite de cette correction, il est nécessaire de définir le deuxième ensemble d'étapes qui caractérise cette nouvelle situation, c'est-à-dire, l'ensemble des étapes dont l'activité vient d'être modifiée. Si E_1 est l'ensemble des étapes actives avant franchissement et E_2 l'ensemble des étapes actives après franchissement et application des ordres de forçage, l'ensemble des étapes dont l'activité vient d'être modifiée par ce franchissement est $(E_1 \cup E_2) - (E_1 \cap E_2)$.

Les nouvelles situations étant maintenant définies, il est nécessaire de calculer leur contexte résiduel. Lorsque la situation est atteinte par le franchissement d'un seul ensemble de transitions, le contexte résiduel de cette situation est égal au contexte maximal de cet ensemble. Dans le cas contraire, il est alors égal à la somme du contexte maximal de chaque ensemble de transitions dont le franchissement permet d'atteindre cette situation.

La dernière opération consiste à construire le niveau de l'arbre correspondant à ces évolutions. Il est constitué de branches dont la situation amont est la situation qui vient d'être analysée tandis que la situation aval est la situation atteinte. Le contexte d'évolution est le contexte résiduel de la situation atteinte.

Pour l'exemple présenté figure 31, seul le franchissement des ensembles $\{t_1, t_2, t_3\}$ et $\{t_1, t_2\}$ conduit à une situation commune. Les situations accessibles depuis la situation $\{1, 4\}$ sont représentées sur la figure 32.

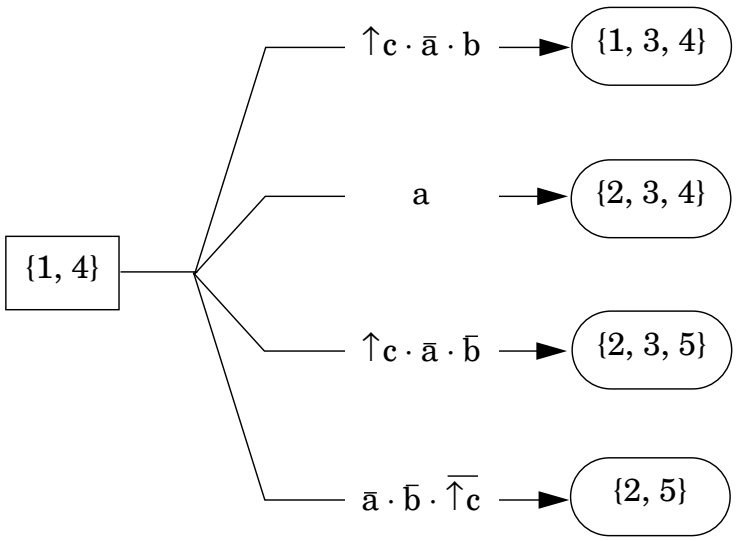


figure 32. Evolutions à l'échelle de temps interne du grafcet présenté figure 31 depuis la situation {1, 4} considérée comme stable

5.1.4. Analyse d'une situation quelconque

A l'opposé du cas précédent, le caractère stable ou instable de la situation étudiée n'est pas connu lorsque débute l'analyse[1] : la situation est seulement décrite par la donnée de son ensemble d'étapes actives, de son ensemble d'étapes dont l'activité a été modifiée et de son contexte résiduel. L'analyse de cette situation s'effectue en cinq phases. La première de celles-ci consiste à vérifier si la situation n'est pas une situation totalement instable. La seconde phase est la détermination des transitions sensibilisées depuis cette situation et de leur contexte résiduel. La troisième phase est l'élaboration de tous les ensembles de transitions simultanément franchissables et de leurs contextes. La quatrième phase consiste à déterminer les situations qui sont atteintes par le franchissement de l'un de ces ensembles de transitions et les évolutions correspondantes. La dernière phase de l'analyse est le traitement de la stabilité partielle de la situation analysée.

Pour illustrer les différents étapes de l'analyse nous nous servons toujours de l'exemple présenté figure 31, la situation $\{1, 4\}$ sera maintenant considérée accessible depuis une autre situation. Nous supposons également que la situation $\{1, 4\}$ a pour un contexte résiduel $a + b$.

5.1.4.1. Vérification de l'instabilité totale de la situation

Pour vérifier si la situation donnée n'est pas une situation totalement instable, il suffit de remonter l'arbre depuis le nœud en cours d'analyse jusqu'à sa racine en vérifiant qu'aucun de ces prédécesseurs n'est cette situation. Si tel était le cas, cette situation serait en effet une situation totalement instable pour toutes les variations des entrées décrites par le contexte résiduel.

En effet, au chapitre 2, nous avons montré que nous étions en présence d'instabilité totale lorsqu'un grafcet effectuait une deuxième fois la même évolution sur occurence d'un événement externe. Compte-tenu de la définition que nous avons retenue (cf. paragraphe 5.1.1.), il nous suffit de comparer les situations, puisque celles-ci se composent de l'ensemble des étapes actives et de l'ensemble des étapes dont l'activité a été modifiée.

---

1. Bien naturellement, cette analyse ne s'effectue pas sur les situations déclarées situation incohérente du point de vue des ordres de forçage.

5.1.4.2. Détermination des transitions sensibilisées avec leur contexte minimal

La détermination de la sensibilité des transitions depuis la situation analysée ne s'effectue que pour les transitions qui n'appartiennent pas à un grafcet partiel forcé. Lors de cette opération, il est vérifié que la réceptivité associée aux transitions validées est compatible avec la situation analysée et son contexte résiduel. Dans la pratique, cette vérification consiste à établir le contexte minimal de cette transition. Si celui-ci s'avère être nul, cette transition n'est pas définie comme étant sensibilisée. Le contexte minimal est construit à partir d'une expression déduite de la réceptivité associée à cette transition et du contexte résiduel de la situation analysée :

$$Cmin\,(T_i) \;=\; Cres \cdot expression\_déduite$$

L'expression déduite de la réceptivité est en fait la réceptivité associée à la transition dans laquelle toutes les expressions portant sur les valeurs et les changements de valeurs des variables d'étapes ont été calculés tandis que les expressions portant sur les changements de valeurs des entrées ont été supprimées[1].

Si, à l'issue de cette étape, aucune transition n'est sensibilisée, la situation analysée est déclarée situation stable, terminant ainsi son analyse pour cet arbre. Dans le cas contraire, la situation est instable pour au moins une partie de son contexte résiduel.

Pour l'exemple de la figure 31, depuis la situation $\{1, 4\}$ ayant comme contexte résiduel $a + b$, les seules transitions sensibilisées avec leur contexte minimal sont :

$$\left(\{t_1\}, a \right), \left(\{t_2\}, a \right)$$

5.1.4.3. Détermination des ensembles de transitions simultanément franchissables avec leur contexte maximal

L'élaboration de tous les ensembles de transitions simultanément franchissables avec leur contexte maximal s'effectue de la même façon que dans le cas précédent.

---

1. Une situation considérée comme instable n'est pas sensible aux événements externes.

Pour l'exemple de la figure 31, depuis la situation $\{1, 4\}$ ayant comme contexte résiduel $a + b$, le seul ensemble de transitions simultanément franchissables avec son contexte maximal est :

$$\left(\{t_1, t_2\}, a \right)$$

5.1.4.4. Détermination des situations accessibles et des évolutions correspondantes

La détermination des situations atteintes par le franchissement de l'un de ces ensembles et des évolutions correspondantes s'effectue de la même façon que dans le cas précédent.

Pour l'exemple de la figure 31, un seul franchissement est possible. Ce dernier est décrit sur la figure 33.

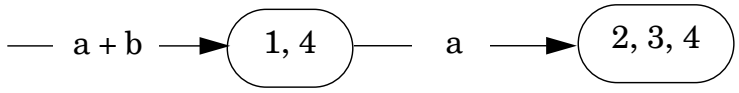


figure 33. Evolutions à l'échelle de temps interne du grafcet présenté figure 31 depuis la situation {1, 4} ayant comme contexte résiduel (a+b)

5.1.4.5. Traitement de la stabilité partielle de la situation analysée

La dernière phase de la méthode consiste à vérifier si cette situation instable n'est pas en fait «partiellement stable». Nous avons caractérisé d'instable la situation analysée car il existe au moins une variation des entrées élément de l'ensemble décrit par le contexte résiduel pour laquelle une transition validée est franchissable. Cependant, il n'existe pas forcément une possibilité d'évolution pour **toutes** les variations des entrées décrites par le contexte résiduel, cette situation peut donc présenter des possibilités de stabilité.

Nous avons introduit pour cela, la notion de contexte de stabilité. Le contexte de stabilité est élaboré à partir du contexte résiduel de la situation analysée et de chaque contexte minimal des transitions sensibilisées de la façon suivante :

$$Csta = Cres \cdot \overline{\sum Cmin\,(T_i)}$$

Si le contexte de stabilité est non nul, la situation instable analysée est également partiellement stable. Pour donner une représentation à ce problème de stabilité partielle, tout en évitant qu'une situation accessible soit déclarée à la fois stable et instable, nous ajoutons dans l'arbre un nouveau nœud et une nouvelle branche (cf. figure 34).

La nouvelle branche a comme situation amont la situation analysée et comme situation aval la situation stable identique à la situation analysée. Le contexte d'évolution de cette branche comme le contexte résiduel du nouveau nœud seront tous deux égaux au contexte de stabilité de la situation analysée.

Pour l'exemple de la figure 31, la partie de l'arbre représentant les évolutions à l'échelle interne depuis la situation {1, 4} ayant comme contexte résiduel $(a + b)$ est présenté figure 34.

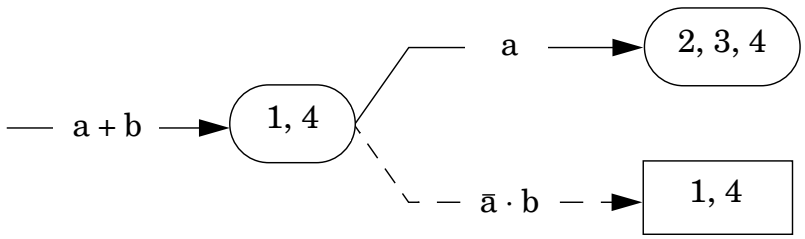


Figure 34. Evolutions à l'échelle interne depuis la situation {1, 4} partiellement stable ayant comme contexte résiduel (a+b)

5.1.5. Remarques sur le résultat

Lorsque s'achève la détermination des situations accessibles à l'échelle de temps interne depuis chaque situation stable atteignable, nous disposons d'un ensemble d'arbres dont la racine représente à chaque fois une situation stable différente. Pour chaque élément terminal d'un arbre, trois possibilités existent :

- il représente une situation stable, cette situation est donc atteignable depuis la situation stable qui est à la racine de l'arbre,
- il représente une situation totalement instable, il s'agit tout probablement d'une erreur de modélisation qu'il est nécessaire de signaler au concepteur du grafcet,

- il représente une situation incohérente du point de vue des ordres de forçage, il s'agit également d'une erreur de modélisation qu'il est nécessaire de signaler au concepteur du grafcet.

A partir des informations établies lors de la détermination des situations accessibles à l'échelle de temps interne depuis chaque situation stable, il nous est maintenant possible de déterminer les évolutions à l'échelle de temps externe.

5.2. Construction de l'automate décrivant le comportement à l'échelle de temps externe

Cette construction s'effectue en deux temps. Nous déterminons tout d'abord é les évolutions à l'échelle de temps externe puis prenons en compte les actions associées aux étapes du grafcet.

5.2.1. Détermination des évolutions à l'échelle de temps externe

5.2.1.1. Modèle de représentation et critères de sélection des éléments représentés

Nous avons choisi de décrire les évolutions à l'échelle de temps externe par un graphe orienté aux arcs étiquetés où :

- chaque sommet du graphe correspond à une situation que peut prendre le grafcet,
- chaque arc du graphe représente une évolution du grafcet à l'échelle de temps externe,
- l'étiquette associée à chaque arc, définit la condition que doivent respecter les entrées du grafcet pour permettre cette évolution.

Avant de présenter la méthode de génération de ce graphe, il est nécessaire de préciser quels sont les éléments (situations et évolutions) qui y seront représentés. Si l'analyste doit être averti de toutes les caractéristiques du grafcet analysé, nous

pensons que certaines d'entre elles n'ont pas à apparaître sur le graphe décrivant les évolutions à l'échelle de temps externe afin de ne pas nuire à son analyse ultérieure.

Nous avons ainsi choisi de ne pas faire apparaître sur ce graphe les situations qui correspondent à des erreurs de modélisation telles les situations totalement instables et les situations incohérentes du point de vue des ordres de forçage. En effectuant ce choix, nous choisissons de considérer qu'une situation donnée n'est réceptive qu'aux variations des entrées qui permettent d'atteindre une situation stable.

Nous avons également décidé de ne pas faire apparaître les évolutions qui ont comme situation amont et situation aval la même situation stable. Nous avons enfin choisi de regrouper sous une seule évolution toutes les évolutions qui ont même situation amont et même situation aval.

5.2.1.2. Méthode de génération

La génération de ce graphe ne présente aucune difficulté en raison du travail effectué lors de la détermination des évolutions à l'échelle de temps interne, la majorité des calculs étant déjà effectués.

Les seules situations prises par le grafcet à l'échelle de temps externe étant les situations stables atteignables à l'échelle de temps interne, le graphe possède donc autant de sommets qu'il y a eu d'arbres générés à l'étape précédente de la démarche et chacun de ces sommets correspond à une racine de ces arbres (à ce niveau, seule la connaissance de l'ensemble des étapes actives est nécessaire).

Pour déterminer tous les arcs issus d'un sommet S donné, il suffit d'en construire autant que l'arbre de racine S possède de nœuds terminaux correspondant à des situations stables différentes entre elles et différentes de S. L'étiquette de chacun de ces arcs est égal au contexte résiduel de la situation stable atteignable lorsque celle-ci est unique dans l'arbre, ou le OU logique de chaque contexte résiduel lorsque la situation stable atteignable est représentée plusieurs fois.

5.2.1.3. Remarques sur le résultat

La méthode de détermination des évolutions à l'échelle de temps externe est telle que ce graphe présente certaines caractéristiques. Il s'agit d'un 1-graphe sans

boucle [Berge 7]3 puisqu'il existe entre deux sommets (S_1, S_2) au plus un arc issu de S_1 et aboutissant sur S_2 et à la condition que S_1 soit différent de S_2. De plus, pour chacun des sommets de ce graphe, les deux propriétés suivantes sont vérifiées :

- Pour tout couple d'arc (a_1, a_2) issu de S, d'étiquette respective E_1, E_2

$$E_1 \cdot E_2 = 0$$

- Le passage d'une situation à une autre n'est pas possible, sans occurrence d'une nouvelle variation des entrées.

Bien que ce graphe ne correspond pas à la définition donnée pour un automate fini[1] dans [Marchand 89] ni à celle donnée dans [Zahnd 87], nous appellerons ce graphe **automate** car il correspond parfaitement à l'image que se fait l'automaticien de l'automate qu'il manipule à travers des graphes d'états.

Dans la suite de ce mémoire, pour rester homogène avec le terme d'automate, nous parlerons d'états et de transitions : un **état** représentant une situation prise par le grafcet à l'échelle de temps externe et une **transition** représentant une évolution entre ces situations. Nous parlerons également de **condition** associée à une transition. Il s'agira de l'expression combinatoire sur $\mathbb{I}$ qui décrit l'ensemble des variations des entrées qui provoque cette évolution.

5.2.2. Prise en compte des actions

Jusqu'à présent, nous ne nous sommes intéressés au grafcet qu'au travers de synchronisation[2] (interprétation limitée aux réceptivités). Pour que l'automate équivalent soit une description du comportement d'un grafcet, il est nécessaire de tenir compte des actions qui sont définies dans ce grafcet.

5.2.2.1. Modèle de représentation

D'une manière générale, toute action[3] d'un grafcet peut être définie par un triplet $(\text{Etape}, \text{Condition}, \text{Sortie})$ où :

---

1. En informatique, le terme d'automate fini correspond à une machine qui permet de reconnaître un langage. Pour cela, il possède un ensemble d'états initiaux et d'états terminaux et l'étiquette associée à chaque arc est une lettre d'un alphabet. De plus, il n'est pas nécessairement déterministe.

2. Le terme est de M. Blanchard [Blanchard 79].

- le premier élément décrit l'étape à laquelle est associée cette action,
- le second élément est une expression combinatoire **formée à partir des entrées et des étapes,**
- le dernier élément définit la sortie qui sera émise quand l'action sera exécutée.

Pour décrire les actions d'un grafcet, nous munissons l'automate d'actions définies également par un triplet (Etat, Condition, Sortie) où :

- le premier élément décrit l'état auquel est associé cette action,
- le second élément est une expression sur Π **formée seulement à partir des entrées**,
- le dernier élément définit la sortie qui sera émise quand l'action sera exécutée.

5.2.2.2. Méthode d'obtention

Pour obtenir les actions de l'automate, il faut construire, tout en évitant les répétitions, une action automate pour chaque action grafcet qui est exécutée dans chaque situation stable. La méthode d'obtention est donnée dans le détail en pseudo-code figure 35.

Nous disposons maintenant d'une description du comportement d'un grafcet sous la forme d'un automate. Cette description n'est cependant pas optimale car il n'a pas été complètement pris en compte l'hypothèse relative à la non occurrence simultanée d'événements externes. L'automate qui vient d'être généré peut présenter des transitions qui ne seront jamais franchies car elle nécessiterait l'occurrence simultanée d'événements externes. Dans la section suivante, nous montrons comment de telles transitions peuvent être supprimées.

---

3. A l'exception des actions impulsionnelles et limitées dans le temps qui ne seront pas traitées dans ce mémoire.

```
Pour chaque action grafcet a_g , faire
   Pour chaque état e, faire
      Si Etape (a_g) est élément de la situation décrit par e, alors
         Calculer les tests relatifs aux étapes dans Condition (a_g)
         Si Condition (a_g) ≠ 0 , alors
            Action_trouvée ⇐ faux
            Pour chaque action automate a, faire
               Si (Etat (a) = e) ET (Sortie (a) = Sortie (a_g)) , alors
                  Action_trouvée ⇐ vrai
                  Condition (a) ⇐ Condition (a) + Condition (a_g)
               Fin si
            Fin pour
            Si Action_trouvée = faux , alors
               Construire une nouvelle action automate a où
                        Etat (a) ⇐ e
                        Condition (a) ⇐ Condition (a_g)
                        Sortie (a) ⇐ Sortie (a_g)
            Fin si
         Fin si
      Fin si
   Fin pour
Fin pour
```

figure 35. Description en pseudo-code de la méthode d'obtention des actions de l'automate

5.3. Prise en compte de l'historique des entrées

Dans les premières sections de ce chapitre nous avons présenté une méthode qui nous a permis d'identifier de manière automatique toutes les situations stables que peut atteindre un grafcet ainsi que toutes ses possibilités d'évolution entre ces situations stables. Lors de ce premier travail, nous avons considéré que depuis chaque situation stable toutes les variations d'entrées compatibles avec les règles de simplification utilisées dans le module de calcul symbolique étaient possibles. Cependant, pour chaque situation il existe toujours un certain nombre de variations d'entrées qui ne peuvent théoriquement se produire car elles sont incompatibles avec l'historique des évolutions des entrées.

Sur la figure 36, il est présenté deux extraits d'automates dans lesquels l'arc (S2, S4) peut être à chaque fois supprimé : pour le graphe figure 36(a), l'évolution est impossible puisqu'elle nécessite que les entrées a et b changent de valeurs simultanément tandis que pour le graphe figure 36(b), elle ne sera jamais réalisée car elle nécessite que l'entrée a passe de nouveau de 0 à 1 sans passer de 1 à 0,

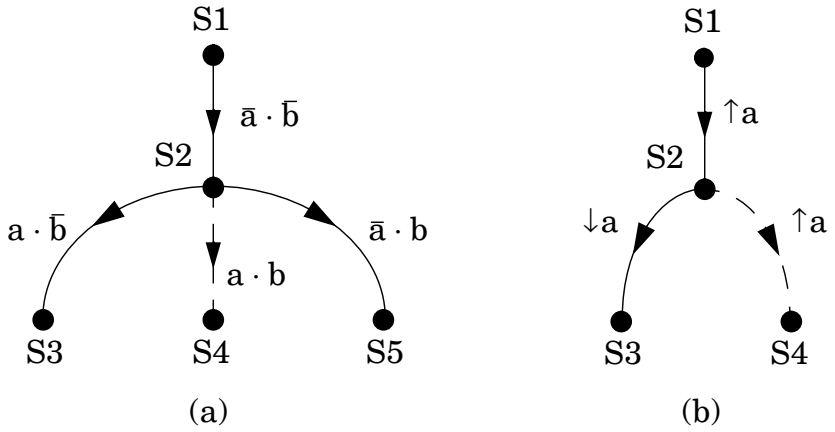


figure 36. Exemples de possibilités de réduction d'un automate en tenant compte de la chronologie des entrées

En n'ayant pas tenu compte de l'historique des évolutions des entrées lors de la génération de l'automate, certaines de ses transitions font ainsi référence à des variations d'entrées impossibles à obtenir. Pour que l'automate soit la fidèle description du comportement du grafcet, il est donc nécessaire que toutes les références à des variations irréalistes des entrées soient supprimées. C'est cette opération que nous nous proposons d'exposer dans cette section.

5.3.1. Modèle du comportement des entrées d'un grafcet

Pour répondre aux besoins exprimés ci dessus, ce modèle doit permettre la représentation des valeurs prises par les entrées et le changement de ces valeurs. Il doit également permettre d'identifier pour chaque variation d'entrées, les variations qui ont dû s'effectuer avant et celles qui pourront l'être après.

Pour être opérationnel, ce modèle doit pouvoir être exploité de manière ensembliste car l'analyse des variations d'entrées les unes après les autres est inenvisageable en raison de leur trop grand nombre.

Nous avons choisi de modéliser le comportement des entrées par un graphe orienté pour lequel chaque sommet correspond à une valeur des entrées tandis que chaque arc matérialise un événement élémentaire, c'est-à-dire la variation d'une entrée. Ainsi, nous disposons sur un modèle unique, d'une représentation formelle et homogène pour les valeurs des entrées, les variations d'entrées ou les successions de variations d'entrées puisque ces trois familles éléments sont respectivement représentées par un sommet, une arête ou un chemin de ce graphe.

Pour répondre aux impératifs opérationnels primordiaux en raison de la taille du graphe (pour un grafcet à n entrées, le graphe décrivant le comportement de celles-ci possède 2^n sommets et $n \times 2^n$ arcs orientés), nous avons pris soin d'associer à chaque sommet l'expression correspondante des entrées et à chaque arc la variation exacte des entrées qu'il représente.

Pour trois entrées (a, b et c) d'un grafcet une représentation graphique du modèle que nous utilisons pour décrire leur comportement est donnée figure 37.

Grâce à ce codage, tout ensemble d'arc comme tout ensemble de sommets peut toujours être défini par une expression combinatoire sur $\mathbb{I}$. Le principal atout de ce codage est certainement de permettre, **uniquement par manipulations élémentaires d'expressions combinatoires sur $\mathbb{I}$** l'obtention de :

- l'ensemble des extrémités terminales d'un ensemble d'arcs ; c'est-à-dire les valeurs que peuvent prendre les entrées à l'issue d'une de leurs variations représentées par ces arcs,

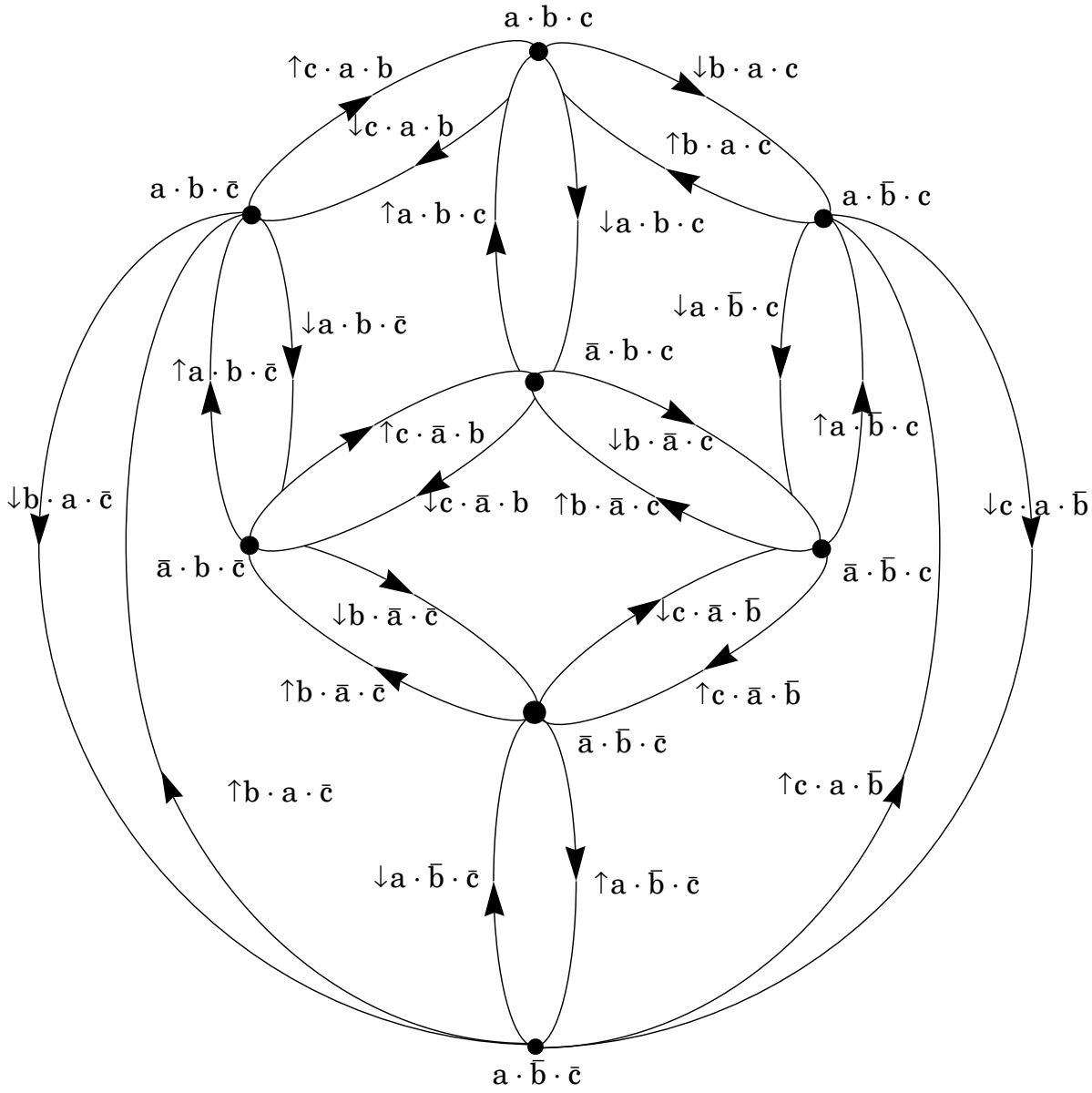


figure 37. Représentation exhaustive des évolutions de trois entrées a, b et c.

- l'ensemble des arcs qui permettent de parcourir un ensemble de sommets ; c'est-à-dire les variations d'entrées possibles entre certaines valeurs des entrées,
- l'ensemble des arcs qui permettent de quitter un ensemble de sommets ; c'est-à-dire les variations d'entrées, possibles depuis certaines valeurs des entrées, qui conduisent à d'autres valeurs des entrées,
- l'ensemble des successeurs directs d'un ensemble de sommets sur le graphe complet ou sur une partie de ce graphe.

Disposant d'un modèle du comportement des entrées, il nous est maintenant possible de formuler notre problème d'identification des seules variations d'entrées qui sont possibles depuis une situation donnée.

5.3.2. Formulation du problème

Pour identifier les seules variations d'entrées qui sont possibles depuis une situation donnée, nous proposons cette condition nécessaire et suffisante :

Théorème 2 :

Pour qu'une variation d'entrées permette de quitter une situation stable d'un grafcet, il est nécessaire et suffisant qu'elle soit le dernier élément d'un enchaînement possible de variations d'entrées dont :

- le premier élément est une variation d'entrées qui permet d'atteindre la situation stable considérée,
- les autres éléments sont des variations d'entrées qui ne permettent pas de quitter la situation stable considérée.

Cette condition nécessaire et suffisante peut également se formuler d'une manière plus adaptée au modèle de comportement des entrées :

Une variation d'entrées permet de quitter une situation stable si et seulement si il existe dans le graphe décrivant le comportement des entrées un chemin de longueur quelconque tel que :

- aucun de ces éléments ne représente une variation d'entrées qui permettent de quitter la situation stable considérée,
- son extrémité initiale est l'extrémité terminale d'un arc représentant une variation d'entrée qui permet d'accéder à la situation stable considérée,
- son extrémité terminale est l'extrémité initiale de l'arc qui représente cette variation.

Grâce à cette seconde formulation, la recherche des variations d'entrées qui permettent d'atteindre une situation stable est ramené à un problème classique d'existence de chemins dans un graphe. En remarquant que l'identification du chemin

n'est pas nécessaire (la connaissance de son existence est suffisante), ce problème ainsi formulé peut maintenant être résolu par un simple calcul de descendant[1].

En effet, si on appelle S l'ensemble des extrémités terminales des arcs qui représente une variation d'entrées qui permet d'accéder à la situation stable considérée, et G l'ensemble des arcs qui représentent une variation d'entrées qui ne permettent pas de quitter cette situation stable alors, une variation d'entrées permet de quitter la situation stable considérée si et seulement si l'extrémité initiale de l'arc qui la représente fait partie des descendants de S sur G.

Sur la figure 38 sont représentés les sommets S, le graphe G et les descendants de S sur G pour l'extrait d'automate donné en encart en considérant que l'ensemble des entrées est {a, b, c}.

5.3.3. Identification des variations d'entrées possibles depuis une situation stable donnée

Pour identifier les variations d'entrées possibles depuis une situation stable, les seules informations pertinentes relatives a la situation stable sont la connaissance de l'ensemble des variations d'entrées qui permettent d'y accéder et de l'ensemble des variations qui permettent d'en repartir. Ces informations sont toutes deux contenues dans l'automate équivalent qui à été généré selon la démarche proposée dans les sections précédentes. Pour obtenir une description de l'ensemble des variations d'entrées qui permet d'accéder à (respectivement de repartir de) la situation, il suffit d'effectuer la somme sur $\mathbb{I}$ des conditions associées aux transitions qui sont en amont (respectivement en aval) de l'état qui représente la situation stable considérée.

Sur le plan calculatoire, la principale difficulté à résoudre est le calcul des descendants d'un sommet sans avoir recours à une description de manière exhaustive du graphe. En effet, en théorie des graphes, il s'agit d'un problème mineur résolu par différents algorithmes [Desbazeille 76]. Dans notre cas, nous ne pouvons pas employer directement ces algorithmes puisque nous nous refusons à décrire le graphe de manière exhaustive. Nous allons cependant utiliser les concepts de ces techniques pour mettre au point une solution spécifique à notre besoin. Sur le plan formel, le

1. y est un descendant de x dans G si et seulement si il existe un chemin de x vers y [Gaudel 87].

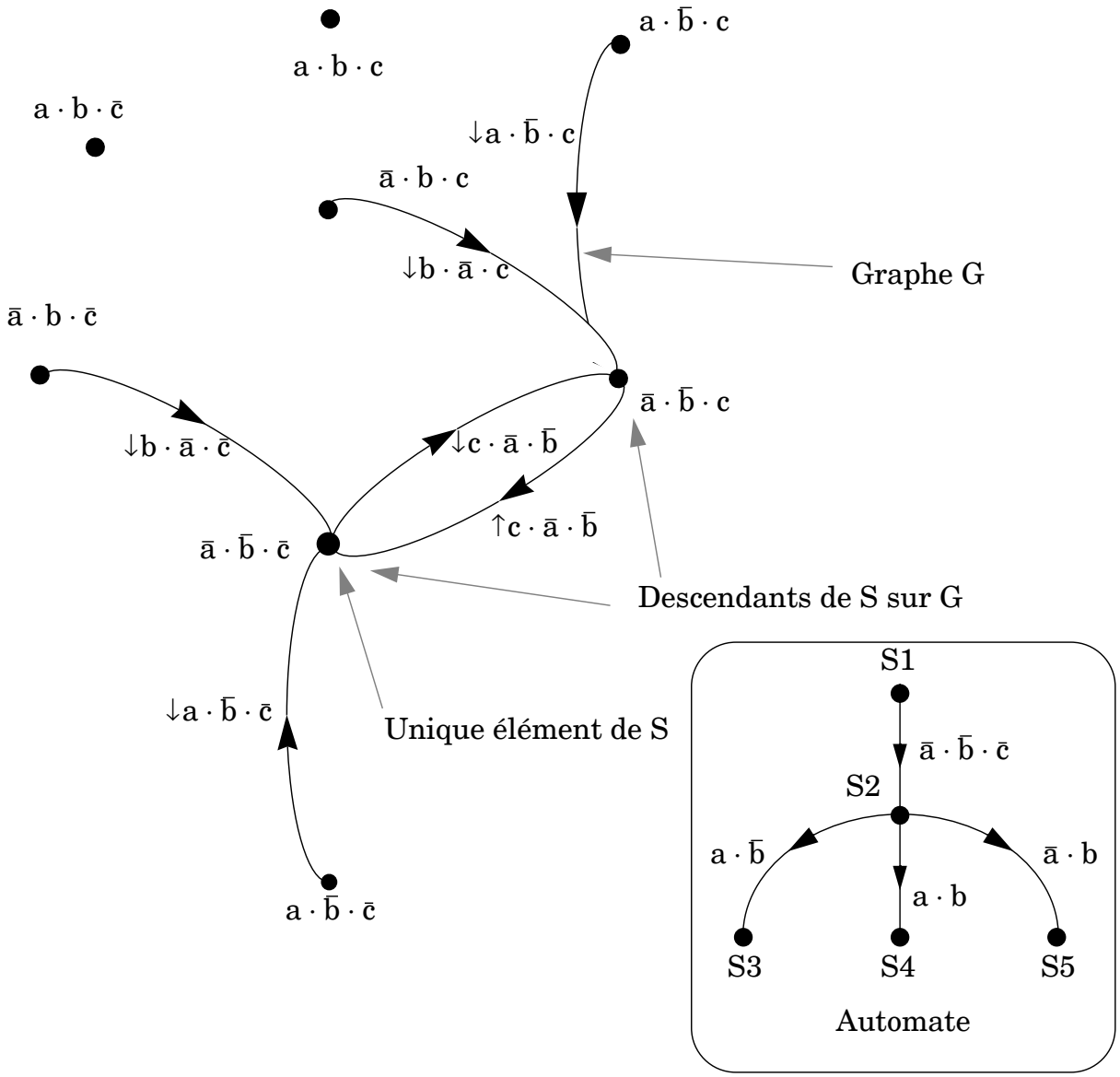


figure 38. Illustration de la possibilité de résoudre notre problème par un calcul de descendants

calcul des descendants d'un ensemble de sommets peut être vu comme un calcul par point fixe[1]. En effet, il s'agit d'établir l'ensemble D de sommets, contenant S, qui reste invariant lorsqu'il est complété par les extrémités terminales des arcs de G dont l'extrémité initiale est déjà élément de D.

Pour obtenir les descendants de S sur G, il nous suffit de reproduire ce calcul de point fixe. Pour cela, nous avons besoin de manipulations d'expressions qui permettent d'obtenir :

- à partir de l'expression qui caractérise un ensemble d'arcs, l'expression qui décrit l'ensemble de leurs extrémités terminales,

1. Cette technique est d'ailleurs celle employée dans l'outil MEC [Crubillé 89].

- à partir de l'expression qui caractérise un ensemble de sommets, l'expression qui décrit l'ensemble des arcs qui permettent de quitter cet ensemble de sommets,
- à partir de l'expression qui caractérise un ensemble de sommets, l'expression qui décrit l'ensemble des arcs qui permettent de parcourir cet ensemble de sommets.

C'est ces manipulations que nous allons maintenant présenter.

5.3.3.1. Obtention des extrémités terminales d'un ensemble d'arcs

A partir de l'expression "Exp" qui caractérise un ensemble d'arcs, l'expression "terminaux(Exp)" qui décrit l'ensemble de leurs extrémités terminales s'obtient simplement en remplaçant dans "Exp", les expressions portant sur le front d'une variable d'entrée suivant le tableau 9.

Ancienne valeur	Nouvelle valeur
$\uparrow a$	a
$\downarrow a$	$\bar{a}$
$\overline{\uparrow a}$	1^*
$\overline{\downarrow a}$	1^*

tableau 9 Remplacement des tests relatifs aux fronts

Cette simplicité est due aux liens étroits qui lient les valeurs des entrées et leur variations. En effet, une expression qui caractérise un ensemble de sommets caractérise également l'ensemble des arcs qui aboutissent à cet ensemble de sommets. Par exemple, pour le comportement des entrées présenté figure 37, l'expression "$b \cdot c$" décrit simultanément l'ensemble S de sommets, et l'ensemble A d'arcs :

$$S = \{a \cdot b \cdot c, \bar{a} \cdot b \cdot c\}$$
$$A = \{\uparrow c \cdot a \cdot b, \uparrow a \cdot b \cdot c, \uparrow b \cdot a \cdot c, \downarrow a \cdot b \cdot c, \uparrow c \cdot \bar{a} \cdot b, \uparrow b \cdot \bar{a} \cdot c\}$$

5.3.3.2. Obtention des arcs extérieurs à un ensemble de sommets

A partir de l'expression "Exp" qui caractérise un ensemble de sommets, il est possible de déterminer l'expression "extérieurs (Exp) " qui caractérise l'ensemble des arcs dont l'extrémité initiale est élément de l'ensemble des sommets tandis que l'extrémité terminale ne l'est pas. "extérieurs (Exp) " s'obtient ainsi :

$$\text{extérieurs}(\text{Exp}) = \downarrow(\text{Exp})$$

Démonstration :

Pour démontrer cette relation, il suffit de raisonner sur les éléments que représentent les sommets et arcs. L'ensemble d'arcs recherché correspond à l'ensemble des variations d'entrées qui permettent de passer des valeurs d'entrées où "Exp" est vérifiée aux valeurs d'entrées où elle ne l'est pas. Les seules variations possibles sont nécessairement celles qui vérifient "$\downarrow$(Exp) ".

C.Q.F.D.

5.3.3.3. Obtention des arcs qui existent entre les éléments d'un ensemble de sommets

A partir de l'expression "Exp" qui caractérise un ensemble de sommets, il est possible de déterminer l'expression "intérieurs (Exp) " qui caractérise l'ensemble des arcs dont les deux extrémités sont éléments de l'ensemble de sommets. "intérieurs (Exp) " s'obtient de la façon suivante :

$$\text{intérieurs}(\text{Exp}) = \text{Exp} \cdot \overline{\uparrow(\text{Exp})}$$

Démonstration :

Si A_1 est l'ensemble des arcs dont l'extrémité terminale est élément de l'ensemble de sommets et A_2 le sous-ensemble de A_1 qui regroupe l'ensemble des arcs dont l'extrémité initiale n'est pas élément de l'ensemble de sommets, alors l'ensemble recherché est égal à $A_1 - A_2$.

Si Exp_1 et Exp_2 sont les expressions qui caractérisent respectivement A_1 et A_2 alors l'expression qui caractérise $A_1 - A_2$ est $\text{Exp}_1 \cdot \overline{\text{Exp}_2}$.

Dans notre cas, comme une expression qui caractérise un ensemble de sommets caractérise également l'ensemble des arcs dont l'extrémité terminale est élément de

cet ensemble de sommets, nous avons $Exp_1 = Exp$. En raisonnant comme au paragraphe 5.3.3.2., on peut démontrer que $Exp_2 = \uparrow(Exp)$.

C.Q.F.D.

5.3.3.4. Détermination des descendants de S sur G

A partir des manipulations d'expressions que nous venons de présenter, l'expression "Desc" qui caractérise l'ensemble des sommets que l'on peut atteindre depuis les sommets de S en suivant uniquement des arcs contenus dans G, s'obtient par le calcul de point fixe de l'expression :

$$Desc_g(s) = s + Desc_g(s) + terminaux(g \cdot extérieurs(Desc_g(s)))$$

où s est l'expression qui caractérise l'ensemble S de sommets et g est l'expression qui caractérise l'ensemble G d'arcs.

Justification :

s permet l'initialisation du calcul de point fixe.

$terminaux(g \cdot extérieurs(Desc_g(s)))$ caractérise l'ensemble des extrémités terminales des arcs qui sont élément de G et dont seul l'extrémité initiale est élément de l'ensemble de sommets recherché.

Grâce à toutes ces opérations, il est nous maintenant possible de déterminer l'expression qui caractérise les variations d'entrées possibles depuis une situation stable donnée.

5.3.3.5. Détermination des variations d'entrées possibles depuis une situation stable donnée

Considérons une situation stable pour laquelle les expressions "Amont" (respectivement "Aval") caractérisent l'ensemble des variations d'entrées qui permet d'accéder à (respectivement de repartir de) la situation. Alors, l'expression "possible" qui caractérise l'ensemble des seules variations d'entrées possibles qui permettent de repartir de la situation en respectant l'historique des évolutions des entrées s'obtient ainsi :

$$possible = (intérieurs(Exp) + extérieurs(Exp)) \cdot Aval$$

où Exp est le résultat du calcul de point fixe suivant :

$$\text{Exp} = \text{terminaux}(\text{Amont}) + \text{Exp} + \text{terminaux}(\overline{\text{Aval}} \cdot \text{extérieurs}(\text{Exp}))$$

En effet, une variation d'entrées permet de quitter la situation stable considérée si et seulement si l'extrémité initiale de l'arc qui la représente fait partie des descendants de S sur G. Ici :

- terminaux (Amont) caractérise l'ensemble S,
- $\overline{\text{Aval}}$ caractérise l'ensemble G,
- Exp caractérise les descendants des sommets de S sur G,
- (intérieurs (Exp) + extérieurs (Exp)) caractérise l'ensemble des arcs dont l'extrémité initiale est un descendant de S sur G.

5.3.3.6. Exemple d'application

Nous nous proposons d'illustrer cette technique d'identification des variations d'entrées possibles en cherchant les seules variations d'entrées possibles pour la situation S_2 de l'extrait d'automate présenté figure 39.

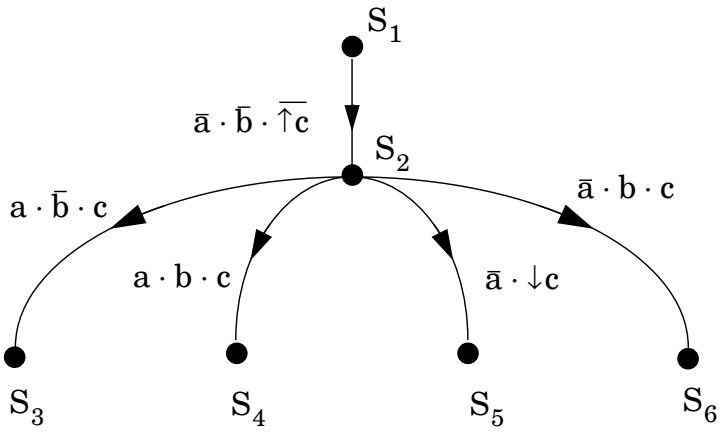


figure 39. Extrait d'automate présentant des transitions sensibles à des variations d'entrées impossibles à obtenir en raison de l'historique des évolutions des entrées.

Détails des calculs :

$$\begin{aligned}
\text{Amont} &= \bar{a} \cdot \bar{b} \cdot \overline{\uparrow c} \\
\text{Aval} &= a \cdot \bar{b} \cdot c + a \cdot b \cdot c + \bar{a} \cdot \downarrow c + \bar{a} \cdot b \cdot c = \bar{a} \cdot \downarrow c + c \cdot (a + b) \\
\text{terminaux}(\text{Amont}) &= \bar{a} \cdot \bar{b} \\
\text{Exp}(0) &= \bar{a} \cdot \bar{b} \\
\text{Exp}(1) &= \bar{a} \cdot \bar{b} + \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{c} \\
\text{Exp}(2) &= \bar{a} \cdot \bar{b} + \bar{c} \\
\text{Exp} &= \bar{a} \cdot \bar{b} + \bar{c} \\
\text{possible} &= \uparrow c \cdot (a + b) + \uparrow a \cdot \bar{b} \cdot c + \bar{a} \cdot \uparrow b \cdot c + \bar{a} \cdot \bar{b} \cdot \downarrow c
\end{aligned}$$

L'extrait d'automate présenté figure 40 correspond à celui présenté figure 39 une fois qu'ont été supprimées toutes les références aux variations d'entrées impossibles à obtenir depuis la situation S_2.

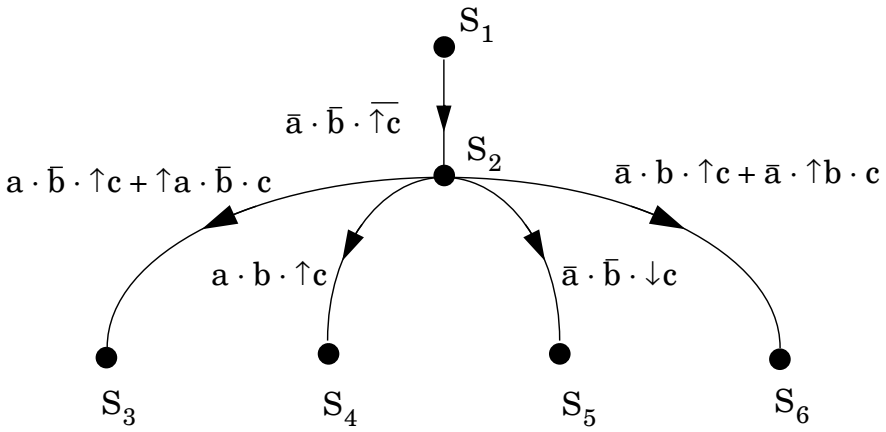


figure 40. Extrait d'automate ne présentant des transitions sensibles qu'à des variations d'entrées cohérente avec l'historique des évolutions des entrées.

En appliquant cette technique à l'automate à états qui a été généré, nous pouvons supprimer toutes les références à des variations irréalistes des entrées et ainsi obtenir un automate réduit qui soit la fidèle description du comportement du grafcet. C'est cette opération de réduction que nous nous proposons maintenant d'exposer.

5.3.4. La technique de réduction

Cette technique a été élaborée pour supprimer d'un automate toutes les références à des variations d'entrées irréalistes. Dans de nombreux cas, cette suppression aboutit à l'élimination de nombreuses transitions et parfois à celle de certains états de l'automate.

A l'issue de cette réduction, chaque condition associée à une transition caractérise les seules variations possibles des entrées qui permettent l'évolution représentée par cette transition.

Avant d'exposer les détails de la technique de réduction, il est nécessaire de préciser quelque points.

5.3.4.1. Remarques préalables

Traitement particulier de la situation initiale :

Comme la situation initiale est la situation prise par le grafcet lors de son initialisation, nous considérons que celle-ci nécessite un traitement particulier. En effet, le modèle GRAFCET ne proposant aucune hypothèse concernant les valeurs ou les changements de valeur des entrées lors de l'initialisation, il n'est pas possible de dire que telle ou telle valeur ne pourra jamais être prise par les entrées lors de l'initialisation. Nous estimons donc qu'aucune évolution depuis la situation initiale ne peut être modifiée même lorsqu'elle fait uniquement référence à des variations d'entrées impossibles à obtenir lorsque cette situation peut être de nouveau atteinte.

Nécessité d'un traitement avec recherche de point fixe :

Comme l'identification des variations d'entrées impossibles à obtenir depuis une situation donnée s'obtient à partir de l'ensemble des variations d'entrées qui permettent d'atteindre cette situation, toute réduction de cet ensemble nécessite de reprendre cette identification. Pour être la plus performante possible, l'analyse d'une situation ne devrait être faite qu'après l'analyse de toutes ses situations amont. Cependant, cette condition est impossible à respecter car l'automate présente de nombreux circuits. Pour cela, la technique de réduction que nous proposons demande un traitement avec recherche de point fixe.

A l'issue de l'analyse d'une situation, il est suffisant de reprendre l'analyse d'une situation en aval si l'ensemble des valeurs que doivent prendre les entrées pour atteindre cette situation a été modifié.

Nécessité d'une méthode de réduction moins performante pour certains automates :

Pour traiter les automates obtenus à partir de grafcets dans lesquels les opérateurs fronts ne sont pas employés, cette technique de réduction doit être bridée si le non-emploi des opérateurs fronts est dû à un choix méthodologique. Pour respecter ce choix, nous proposons une seconde version de la technique de réduction, moins performante mais qui permet d'obtenir l'automate à état équivalent dans lequel il est fait uniquement référence à des valeurs d'entrées.

5.3.4.2. Détail de la méthode de réduction dans le cas général

Pour réduire l'automate, on aurait pu opérer pour chacun de ses états comme il l'a été fait sur l'exemple présenté figure 39. Cependant, en raison du traitement par point fixe, il est nécessaire d'apporter quelques légères modifications pour optimiser les calculs et mettre en place le critère d'arrêt de la procédure.

A chaque état[1] de l'automate, sont associées les quatre expressions suivantes :

- "Atteintes" qui caractérise l'ensemble des *valeurs* que peuvent prendre les entrées à l'issue d'une variation qui a permis d'atteindre l'état considéré,
- "Atteignables" qui caractérise l'ensemble des *valeurs* que peuvent prendre les entrées sans provoquer un changement d'état,
- "Sans_chgt" qui caractérise l'ensemble des *variations d'entrées* qui ne provoquent pas de changement d'état,
- "Possibles" qui caractérise l'ensemble des *variations d'entrées* possibles depuis une des valeurs d'entrées définies par "Atteignables".

Nous associons également à chaque état le booléen "A_reprendre" qui permet la réitération des calculs si nécessaire.

La description algorithmique de la méthode est donnée sur la figure 41 en pseudo-code.

1. hormis l'état initial

Initialisation de la technique

Pour chaque état, faire

Atteintes $\Leftarrow 1^{*}$

A_reprendre $\Leftarrow$ vrai

Sans_chgt $\Leftarrow \overline{\sum_{\text{transition aval}} \text{Cond}}$

Fin pour

Corps de le technique

Pour chaque état où A_reprendre = vrai , faire

A_reprendre $\Leftarrow$ faux

Si Atteintes $\neq$ terminaux$\left(\sum_{\text{transition amont}} \text{Cond}\right)$, alors

Atteintes $\Leftarrow$ terminaux$\left(\sum_{\text{transition amont}} \text{Cond}\right)$

Si Atteintes = 0^{*}, alors

supprimer toutes les transitions en aval

supprimer toutes les actions associées à l'état

supprimer l'état

Sinon

Calcul avec point fixe de :

Atteignables = terminaux (Sans_chgt · extérieurs (Atteignables)) + Atteintes + Atteignables

Possibles $\Leftarrow$ intérieurs (Atteignables) + extérieurs (Atteignables)

Pour chaque transition en aval faire

Si Cond $\neq$ Cond · Possibles , alors

Cond $\Leftarrow$ Cond · Possibles

Pour tout état en aval de la transition, faire

A_reprendre $\Leftarrow$ vrai

Fin pour

Si Cond = 0^{*}, alors

supprimer la transition

Fin si

Fin si

Fin pour

Fin si

Fin si

Fin pour

figure 41. Description en pseudo-code de la méthode de réduction

5.3.4.3. Détail de la méthode de réduction simplifiée

Cette méthode est conçue pour réduire les automates obtenus à partir de grafcets dans lesquels les opérateurs fronts ne sont pas employés. Dans ces automates, les transitions ne font jamais référence à des variations d'entrées mais seulement à des valeurs d'entrées. Nous nous servons de cette caractéristique pour simplifier les calculs.

A chaque état[1] de l'automate, sont associées les quatre expressions suivantes :

- "Atteintes" qui caractérise l'ensemble des *valeurs* que peuvent prendre les entrées en atteignant l'état considéré,
- "Atteignables" qui caractérise l'ensemble des *valeurs* que peuvent prendre les entrées sans provoquer un changement d'état,
- "Sans_chgt" qui caractérise l'ensemble des *valeurs* qui ne provoquent pas un changement d'état,
- "Possibles" qui caractérise l'ensemble des *valeurs* qu'il est possible d'atteindre depuis une des valeurs d'entrées définies par "Atteignables".

Nous associons également à chaque état le booléen "A_reprendre" qui permet la réitération des calculs si nécessaire.

La description de la méthode en pseudo-code est donnée sur la figure 42.

1. hormis l'état initial

Initialisation de la technique

Pour chaque état, faire

Atteintes $\Leftarrow 1^{*}$

A_reprendre $\Leftarrow$ vrai

Sans_chgt $\Leftarrow \overline{\sum_{\text{transition aval}} \text{Cond}}$

Fin pour

Corps de le technique

Pour chaque état où A_reprendre = vrai , faire

A_reprendre $\Leftarrow$ faux

Si Atteintes $\neq \sum_{\text{transition amont}} \text{Cond}$, alors

Atteintes $\Leftarrow \sum_{\text{transition amont}} \text{Cond}$

Si Atteintes = 0^{*} , alors

supprimer toutes les transitions en aval

supprimer toutes les actions associées à l'état

supprimer l'état

Sinon

Calcul avec point fixe de :

Atteignables = terminaux (extérieurs (Atteignables)) $\cdot$ Sans_chgt + Atteintes + Atteignables

Possibles $\Leftarrow$ terminaux (extérieurs (Atteignables))

Pour chaque transition en aval faire

Si Cond $\neq$ Cond $\cdot$ Possibles , alors

Cond $\Leftarrow$ Cond $\cdot$ Possibles

Pour tout état en aval de la transition, faire

A_reprendre $\Leftarrow$ vrai

Fin pour

Si Cond = 0^{*} , alors

supprimer la transition

Fin si

Fin si

Fin pour

Fin si

Fin si

Fin pour

figure 42. Description en pseudo-code de la méthode de réduction simplifiée

5.3.5. Quelques remarques sur le résultat obtenu

Nous avons pu expérimenter sur plusieurs exemples de différente nature cette technique de réduction d'automates basée sur la suppression des références à des variations des entrées incompatibles avec l'historique des entrées.

Pour juger son efficacité, nous l'avons testé sur des exemples comme ceux reproduits sur la figure 43.

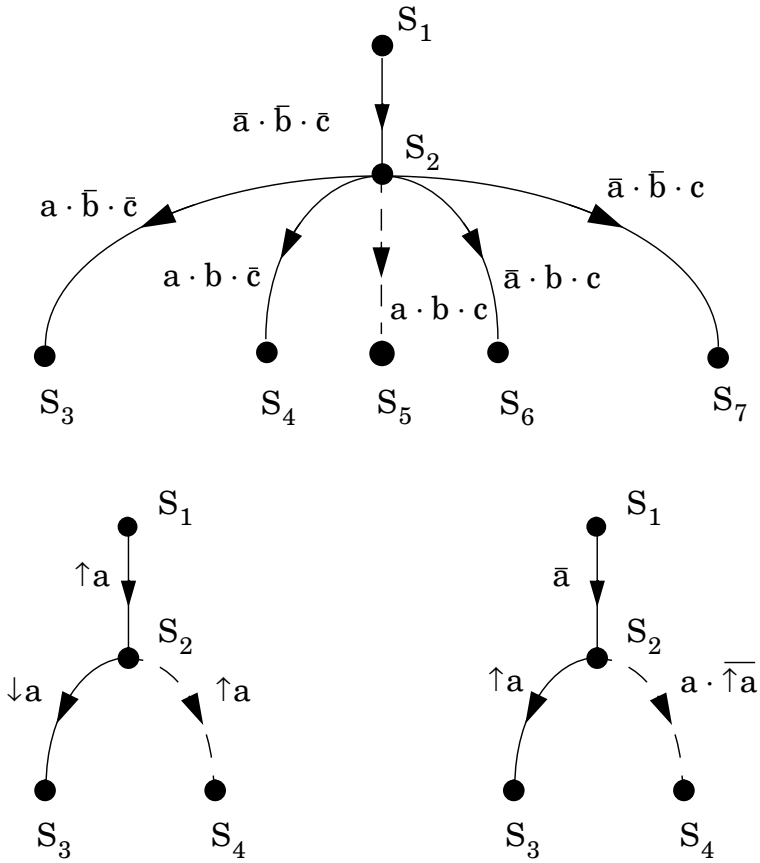


figure 43. Exemples de possibilités de réduction de l'automate en tenant compte de la chronologie des entrées.

Pour juger de sa robustesse, nous l'avons également testé sur des automates plus importants. Au chapitre 7, nous présentons un grafcet pour lequel il a été généré l'automate équivalent qui a été ensuite réduit en supprimant les références aux variations d'entrées incompatibles avec leur historique. Cet automate qui possédait 32 états et 512 transitions avant application de cette technique a été optimisé et ne possède plus que 32 états et 171 transitions.

Sur le plan de l'efficacité, il peut paraître surprenant d'identifier dans un premier temps toutes les variations d'entrées qui provoquent une évolution du grafcet, d'en calculer toutes leurs conséquences pour ensuite rechercher quelles sont celles qui n'auraient pas du être identifiées et éliminer les conséquences de cette iden-

tification erronée. Cependant, il ne peut être fait autrement car il est nécessaire de connaître l'ensemble des valeurs que peuvent prendre les entrées en atteignant une situation pour identifier les variations d'entrées qui ne pourront pas se produire. Comme cet ensemble de valeurs ne peut être établi qu'après avoir identifié toutes les évolutions théoriques du grafcet, cette opération ne peut donc pas être réalisée conjointement à la génération de l'automate.

5.4. Résultats expérimentaux

Nous avons validé cette méthode de génération de l'automate équivalent par le développement d'une maquette informatique écrite en langage C. Au chapitre 7, nous donnons quelques résultats expérimentaux concernant les possibilités de calculs de l'automate équivalent à un grafcet pour un problème classique de l'informatique, le "problème des philosophes".

Nous avons expérimenté notre maquette en générant l'automate à état équivalent pour une modélisation en GRAFCET à l'ordre 10 (c'est-à -dire, 10 philosophes ayant le même comportement). Pour dix philosophes, le grafcet se compose de 80 étapes regroupés dans 20 graphes connexes ; certaines des situations du grafcet contiennent jusqu'à jusqu'à 29 étapes simultanément actives. L'automate à état équivalent se compose de 15 125 états et de 151 250 transitions.

Sur la figure 44, nous donnons quelques chiffres relatifs à la génération des automates pour le problème des philosophes.

A l'issue de cette phase de nos travaux qui permet de générer automatiquement l'automate équivalent à un grafcet, il est maintenant possible d'obtenir **automatiquement** le comportement de nombreux grafcets exprimé dans une machine d'état rudimentaire (sans aucun parallélisme). Dans le chapitre suivant, nous allons présenter comment arriver à la **validation** d'un grafcet.

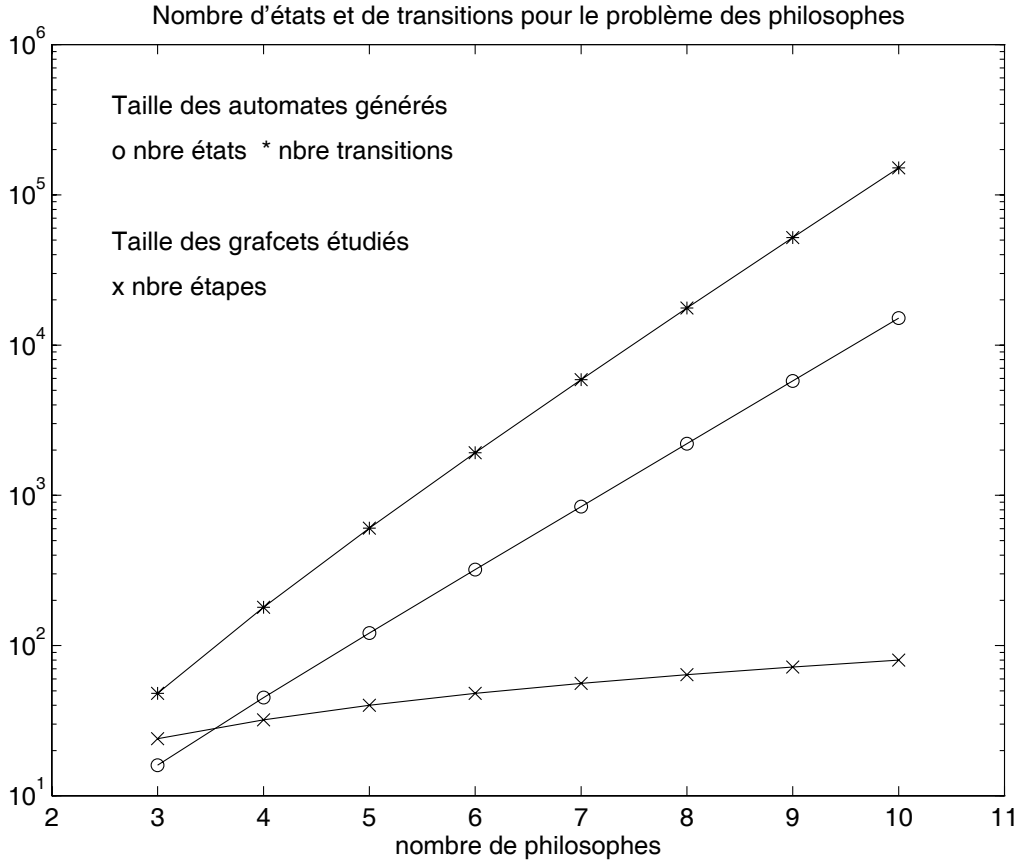


figure 44. Nombre d'états et de transitions pour le problème des philosophes.

Chapitre 6
Validation de grafcets par analyse de l'automate équivalent

Le précédent chapitre a été consacré à la description de la méthode d'obtention de l'automate équivalent à un grafcet. Dans ce chapitre, nous nous intéressons maintenant à son utilisation en présentant les moyens mis en œuvre pour vérifier les propriétés invoquées lors de la validation d'un grafcet. A l'élaboration d'une liste de propriétés type, liste qui s'avérerait toujours incomplète, nous avons préféré développer un cadre pour la validation de grafcets par vérification de propriétés sur leur automate équivalent. Ce cadre se compose :

- de deux définitions mathématiques pour l'automate à états,
- d'une théorie pour l'expression des propriétés,
- d'un certain nombre de propriétés de base.

Nous offrons donc à tout analyste la possibilité de définir les propriétés qu'il souhaite vérifier sur ses grafcets. Une fois celles-ci définies formellement, il est alors possible de mettre en place des moyens logiciels permettant de calculer automatiquement les parties de l'automate qui les satisfont.

Nous allons d'abord présenter les quelques propriétés qui doivent être vérifiées lors de la génération de l'automate équivalent et non à l'issue de cette génération car les informations sur lesquelles elles reposent, ne sont pas contenues dans l'automate.

6.1. Propriétés établies à partir d'informations obtenues durant la génération de l'automate équivalent

Parmi les informations obtenues lors de la génération de l'automate équivalent, certaines d'entre elles ne sont pas reprises dans l'automate car celui-ci est conçu pour être la représentation exhaustive - et la plus compacte possible - du comportement à l'échelle de temps externe. Pour cela, nous rapellons ci-dessous les choix qui ont été effectués pour sa constitution :

- seules les situations stables et la situation initiale sont représentées,
- si une même situation est à la fois la source et le but d'une évolution, alors cette évolution n'est pas représentée (nous considérons qu'il ne s'agit pas d'une évolution pertinente à l'échelle de temps externe),
- si deux évolutions ont même source et même but, alors elles sont représentées par un seule transition de l'automate,
- si deux actions d'un grafcet sont exécutées depuis la même situation et concernent la même sortie, alors elles sont représentées par un seule action dans l'automate.

Si ces options nous ont permis d'avoir une meilleure représentation du comportement à l'échelle de temps externe, elles ont également conduit à dissimuler certaines particularités du grafcet que l'analyste peut vouloir détecter. Afin que ces choix n'aient aucune conséquence néfaste pour la validation du grafcet, il est nécessaire d'avertir systématiquement l'analyste :

- s'il était rencontré des situations totalement instables ou incohérentes du point de vue des ordres de forçage,
- s'il était identifié des évolutions à l'échelle de temps interne sans conséquence à l'échelle de temps externe,
- s'il était effectué des regroupements d'évolutions ou des fusionnements d'actions.

Grâce à cette précaution, bien que les informations nécessaires pour ces vérifications n'apparaissent pas dans l'automate, l'analyste peut vérifier les propriétés suivantes :

- *la non-existence de situation totalement instable,*
- *la non-existence de cas de forçages simultanés d'un même grafcet partiel,*
- *la non-existence d'évolution à l'échelle de temps interne qui sont sans conséquence à l'échelle de temps externe,*
- *la non-existence d'évolutions identiques à l'échelle de temps externe mais ne passant pas par les mêmes situations à l'échelle de temps interne,*
- *la non-existence d'actions exécutées dans la même situation stable et concernant la même sortie*.

A l'opposé de ces cinq propriétés, toutes les autres propriétés que nous allons présenter s'obtiennent par analyse de l'automate équivalent.

6.2. Définition mathématique de l'automate équivalent

Définition 14 :

Sur le plan mathématique, l'automate à états qui a été généré est une machine de Mealy binaire complètement spécifiée dont la machine séquentielle est uniforme avec une condition initiale réduite à un singleton.

Cette définition a été réalisée à partir de celles proposées dans l'ouvrage de J. Zahnd «Machines séquentielles» [Zahnd 87].

Il s'agit bien d'une machine de Mealy puisque les sorties de l'automate dépendent de son état et de ses entrées. Comme cette machine est déterministe et mono-état (la cardinalité des variables secondaires présentes est de 1), elle est nécessairement complètement spécifiée, sa machine séquentielle est uniforme et la condition initiale de celle-ci est réduite à un singleton.

6.2.1. Représentation complète

A partir d'une des descriptions possibles donnée dans [Zahnd 87] pour une machine de Mealy (que nous avons complété pour tenir compte de toutes les spécificités de celle que nous utilisons) nous proposons pour l'automate équivalent la définition suivante :

Définition 15 :

L'automate équivalent à un grafcet est un sextuple [X, Y, Z, Y_0, T, A] où :

- X est l'ensemble des entrées,
- Y est l'ensemble des états,
- Z est l'ensemble des sorties,
- $Y_0 \subset Y$ est la condition initiale,
- T est l'ensemble des transitions t ($t = (Am, Cond, Av)$ où $(Am, Av) \in Y^2$ et Cond est une fonction sur Π des éléments de X).
- A est l'ensemble des actions a ($a = (Etat, Cond, Sortie)$ avec $Etat \in Y$, $Sortie \in Z$ et Cond est une fonction sur Π des éléments de X),

possédant les caractéristiques suivantes :

X, Y, Z, T et A sont des ensembles deux à deux disjoints.

$$Card(Y_0) = 1^*$$

$$\forall t \in T : Am(t) \neq Av(t)$$

$$\forall (t_1, t_2) \in T^2, (t_1 \neq t_2) : (Am(t_1) = Am(t_2)) \Rightarrow (Av(t_1) \neq Av(t_2))$$

$$\forall (t_1, t_2) \in T^2, (t_1 \neq t_2) : (Am(t_1) = Am(t_2)) \Rightarrow \left(Cond(t_1) \cdot Cond(t_2) = 0^* \right)$$

Comme ce sextuple est l'image d'un grafcet, la méthode d'obtention que nous proposons lui confère les propriétés suivantes :

- $\forall t \in T : Cond(t) \neq 0^*$
- $\forall a \in A : Cond(a) \neq 0^*$
- $\forall y \in Y - Y_0 : \sum_{t_i,\, Am(t_i) = y} Cond(t_i) \neq 1^*$

- Toute occurrence d'un événement permettant d'accéder à un état, ne peut permettre de le quitter.

- Soit E, l'ensemble des étapes de ce grafcet, il existe une correspondance[1] $Sit : Y \rightarrow E$ décrivant le fait que chaque état représente une situation composée d'un ensemble d'étapes. Cette correspondance possède la propriété suivante :

$$\forall (y_1, y_2) \in Y^2, (y_1 \neq y_2) : Sit(y_1) \neq Sit(y_2)$$

6.2.2. Représentation partielle

Si nous considérons uniquement la partie [Y, T] du sextuple [X, Y, Z, Y_0, T, A] composée de l'ensemble des états et de l'ensemble des transitions, cette partie correspond à un graphe orienté étiqueté dont Y est l'ensemble des sommets et T l'ensemble des arcs. Le triplet $t = (Am, Cond, Av)$ qui caractérise l'arc, représente respectivement l'extrémité initiale, l'étiquette et l'extrémité terminale de l'arc.

Par sa construction ce graphe (que nous appellerons G) est un 1-graphe sans boucle [Berge 73]. En effet, ce graphe vérifie les deux propriétés suivantes :

$$\forall (t_1, t_2) \in T^2, (t_1 \neq t_2) : (Am(t_1) = Am(t_2)) \Rightarrow (Av(t_1) \neq Av(t_2))$$

$$\forall t \in T : Am(t) \neq Av(t)$$

G étant un 1-graphe, nous pourrons alors définir tout chemin par la séquence d'arcs qui le compose ou par la succession des sommets qu'il rencontre.

Pour conserver des notations homogènes, nous utilisons les notations suivantes :
- l'ensemble des chemins de G est notée C,
- l'extrémité initiale d'un chemin c est notée $Am(c)$,
- l'extrémité terminale d'un chemin c est notée $Av(c)$,
- l'ensemble des arcs qui compose le chemin c est noté $Arc(c)$,
- l'ensemble des sommets rencontré le long du chemin c est noté $Som(c)$.

---

1. Une correspondance f entre un ensemble A et un ensemble B est définie par la donnée des ensembles A et B et par une règle qui associe à tout élément x de A un sous-ensemble de B, noté f(x) ([Zahnd 87] page 8).

6.3. Proposition d'un cadre théorique pour l'expression de propriétés

Nous avons choisi délibérement une représentation ensembliste pour la définition mathématique de l'automate équivalent pour bénéficier de la théorie des ensembles lors de son analyse. En effet, grâce aux lois de composition des ensembles, il est facile d'obtenir tous les éléments qui vérifient des propriétés complexes, car dans la majorité des cas ces propriétés ne sont qu'une composition de propriétés élémentaires.

Cependant, les trois lois de compositions élémentaires ($\cap$, $\cup$, -) ne sont pas suffisantes pour répondre à tous nos besoins. En effet, l'automate que nous analysons se compose de cinq ensembles de nature différentes et il est parfois nécessaire de passer d'un ensemble à un autre. Pour cela, nous proposons d'utiliser dix autres opérateurs de composition.

Les six opérateurs suivants correspondent à des commandes de calculs de base définies dans l'outil MEC [Crubillé 89] :

Définition 16 :

Amont (T_1) regroupe l'ensemble des états qui sont l'origine d'une des transitions de T_1.

Aval (T_1) regroupe l'ensemble des états qui sont l'extrémité d'une des transitions de T_1.

Int (Y_1) regroupe l'ensemble des transitions qui aboutissent sur un des états de Y_1.

Ext (Y_1) regroupe l'ensemble des transitions qui sont issues d'un des états de Y_1.

Succ (Y_1, T_1) regroupe l'ensemble des états que l'on peut atteindre depuis un état contenu dans Y_1 en suivant une ou plusieurs transitions contenus dans T_1.

Pred (Y_1, T_1) regroupe l'ensemble des états à partir desquels on peut atteindre un état contenu dans Y_1 en suivant une ou plusieurs transitions contenues dans T_1.

Leur définition mathématique est la suivante :

$$\begin{aligned}
\text{Amont}(T_1) &= \{y \in Y \mid \exists t \in T_1 : (\text{Am}(t) = y)\} \\
\text{Aval}(T_1) &= \{y \in Y \mid \exists t \in T_1 : (\text{Av}(t) = y)\} \\
\text{Int}(Y_1) &= \{t \in T \mid (\text{Av}(t) \in Y_1)\} \\
\text{Ext}(Y_1) &= \{t \in T \mid (\text{Am}(t) \in Y_1)\} \\
\text{Succ}(Y_1, T_1) &= \{y \in Y \mid \exists c \in C : (\text{Am}(c) \in Y_1), (\text{Arc}(c) \subset T_1), (\text{Av}(c) = y)\} \\
\text{Pred}(Y_1, T_1) &= \{y \in Y \mid \exists c \in C : (\text{Am}(c) = y), (\text{Arc}(c) \subset T_1), (\text{Av}(c) \in Y_1)\}
\end{aligned}$$

Les quatre derniers opérateurs sont plus spécifiques à notre cas :

Définition 17 :

$\text{Act}(E_1)$ regroupe l'ensemble des états qui représentent une situation pour laquelle toute les étapes de E_1 sont actives.

$\text{Des}(E_1)$ regroupe l'ensemble des états qui représentent une situation pour laquelle toute les étapes de E_1 sont désactives.

$\text{Pres}(X_1, D)$ regroupe l'ensemble des éléments de D (D peut être un ensemble d'actions ou un ensemble de transitions) qui nécessitent la présence de toutes les entrées de X_1 pour être exécutée quand D est un ensemble d'actions, ou pour être franchie quand D est un ensemble de transitions.

$\text{Abs}(X_1, D)$ regroupe l'ensemble des éléments de D (D peut être un ensemble d'actions ou un ensemble de transitions) qui nécessitent l'absence de toutes les entrées de X_1 pour être exécutée quand D est un ensemble d'actions, ou pour être franchie quand D est un ensemble de transitions.

Leur définition mathématique est la suivante :

$$\begin{aligned}
\text{Act}(E_1) &= \{y \in Y \mid (E_1 \subset \text{Sit}(y))\} \\
\text{Des}(E_1) &= \{y \in Y \mid (E_1 \subset (E - \text{Sit}(y)))\} \\
\text{Pres}(X_1, D) &= \{d \in D \mid \left(\sum_{X_1} \overline{x_i}\right) \cdot \text{Cond}(d) = 0^*\} \\
\text{Abs}(X_1, D) &= \{d \in D \mid \left(\sum_{X_1} x_i\right) \cdot \text{Cond}(d) = 0^*\}
\end{aligned}$$

l'ensemble D peut être un ensemble d'actions ou un ensemble de transitions.

Grâce à la définition ensembliste de l'automate équivalent et cette liste d'opérateurs de composition, il est aisé d'exprimer et de calculer les propriétés que l'on souhaite vérifier sur un grafcet.

6.4. Les propriétés établies par analyse de l'automate équivalent

la liste de propriétés que nous proposons ici ne se veut pas exhaustive mais suffisamment variée pour donner une idée des importantes possibilités pour la validation d'un grafcet, que permet l'analyse de son automate équivalent.

Pour en faciliter la lecture nous donnons pour chaque propriété qui sont regroupées par thème, une définition des termes recherchés en langage naturel suivie de la définition mathématique correspondante.

6.4.1. Propriétés relatives aux possibilités d'évolution

6.4.1.1. Absence de situations de blocage

En GRAFCET, en raison de l'utilisation des variables internes dans les réceptivités, il est parfois difficile de s'assurer qu'en aucun cas, il ne peut avoir des cas de blocage. Ce blocage peut être total (plus aucune évolution n'est possible) ou seulement partiel lorsqu'il ne concerne qu'une partie du grafcet global.

Pour vérifier l'absence de situations totalement bloquées, il suffit de rechercher s'il existe des états de l'automate équivalent qui ne sont à l'origine d'aucune transition.

Pour réaliser cette opération, il faut vérifier que l'ensemble suivant est vide :

$$Y - \text{Amont}(T)$$

Dans le cas contraire, l'ensemble des situations totalement bloquées est défini par :

$$\{\text{Sit}(y) \mid (y \in Y - \text{Amont}(T))\}$$

Le cas du blocage partiel est certainement le plus courant. En effet, comme les grafcets ne sont pas tous synchronisés entre eux, il existe toujours au moins un grafcet libre d'évoluer. Si nous considérons qu'une situation donnée ne présente un

blocage partiel que si et seulement si cette situation ainsi que toutes les situations accessibles depuis cette situation présente le même sous-ensemble d'étapes, alors la détection des blocages partiels peut s'effectuer ainsi :

Soit E_1 un ensemble donné d'étapes, pour vérifier que cet ensemble d'étapes n'est pas la source d'un blocage partiel, il suffit de vérifier que l'ensemble suivant est vide :

$$Act(E_1) - Pred((Y - Act(E_1)), T)$$

En effet, cet ensemble se compose des états représentant des situations où toutes les étapes de E_1 sont actives, depuis lequel il n'est pas possible d'atteindre un état qui représente une situation où une étape de E_1 est inactive.

Par exemple, pour détecter si les synchronisations présentées figure 45 ne provoquent pas de blocage partiel, il suffit de s'assurer que l'ensemble suivant est vide :

$$(Act(\{10\}) - Pred(Des(\{10\}), T)) \cup (Act(\{20\}) - Pred(Des(\{20\}), T))$$

$$\text{nota} \begin{cases} Y - Act(\{10\}) = Des(\{10\}) \\ Y - Act(\{20\}) = Des(\{20\}) \end{cases}$$

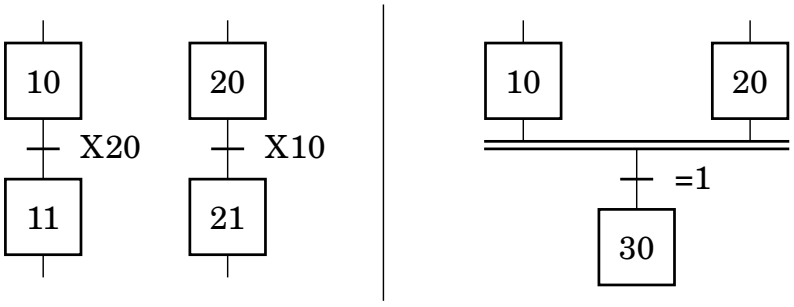


figure 45. Exemples pouvant conduire à des cas de blocage partiel

6.4.1.2. Possibilités de retour à la situation initiale

En GRAFCET, la situation initiale joue un rôle privilégié car elle correspond souvent à la situation dans laquelle se trouve la commande lorsque le processus commandé est au repos. Il est donc parfois intéressant de connaître les situations depuis lesquelles la situation initiale ne pourra jamais être atteinte. De plus, par sa nature, la situation initiale est une situation toujours répertoriée dans l'automate équivalent bien qu'elle est parfois instable ou jamais atteinte.

Pour vérifier si la situation initiale correspond à une situation stable, il suffit de calculer l'expression suivante et de vérifier si elle est bien différente de 1*.

$$Exp = \sum Cond(t_i) \qquad t_i \in Ext(Y_0)$$

Dans le cas contraire, la situation initiale est instable.

Pour déterminer si la situation initiale peut être de nouveau atteinte, il suffit de vérifier si l'automate équivalent contient au moins une transition qui aboutit à l'état initial. Il suffit donc de construire l'ensemble suivant et vérifier qu'il est non vide.

$$Int(Y_0)$$

Dans le cas contraire, la situation initiale est une situation jamais atteinte.

Pour vérifier si la situation initiale peut être atteinte depuis chaque situation il faut construire l'ensemble suivant et vérifier qu'il est vide :

$$Y - Pred(Y_0, T)$$

Dans le cas contraire, depuis les situations suivantes il n'est pas possible d'atteindre la situation initiale :

$$\{Sit(y) \mid (y \in Y - Pred(Y_0, T))\}$$

6.4.2. Propriétés relatives aux émissions de sorties

6.4.2.1. Emissions simultanées de deux sorties

Au chapitre 1, lorsqu'il a été abordé les besoins des utilisateurs, nous avons présenté un exemple de sur-spécification visant à s'assurer que deux actions mécaniquement incompatibles ne peuvent jamais être exécutées simultanément.

A l'aide de l'automate à état, il est possible de vérifier si deux sorties s_1 et s_2 peuvent être émises simultanément. Pour cela, il faut rechercher les états depuis lesquels sont exécutés simultanément deux actions a_1 et a_2 dont les sorties concernent s_1 et s_2. Si l'ensemble Y_1 est vide, alors les sorties ne sont jamais émises simultanément.

$$Y_1 = \{y \in Y \mid \exists (a_1, a_2) \in A^2 : (Sortie(a_1) = s_1), (Sortie(a_2) = s_2), \left(Cond(a_1) \cdot Cond(a_2) \neq 0^*\right), (Etat(a_1) = Etat(a_2) = y)\}$$

Dans le cas contraire, il est possible d'émettre simultanément s_1 et s_2 depuis les situations de l'ensemble suivant :

$$\{Sit(y) \mid (y \in Y_1)\}$$

6.4.2.2. Emission séquentielle de sorties dangereuses

La vérification que deux actions mécaniquement incompatibles ne pourront jamais s'exécuter simultanément ne suffit pas pour s'assurer que la commande de ces deux sorties sera toujours correcte. En effet, prenons par exemple le cas de deux vérins qui partagent le même espace de travail et pilotés par des pré-actionneurs bistables (cf. figure 46). Pour s'assurer qu'il n'y ait jamais collision entre les vérins, il faut contrôler que la sortie d'un vérin se fasse toujours après la rentrée de l'autre.

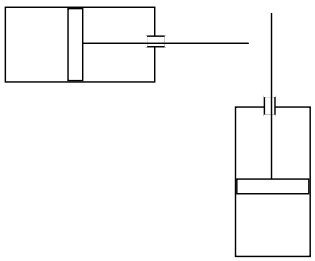


figure 46. Exemples d'interaction entre deux actionneurs

Si s_1, s_2, r_1, r_2 sont respectivement les sorties qui correspondent aux commandes de sortie et de rentrée de ces deux vérins et si ces sorties ne sont pas émises par des actions conditionnelles[1], vérifier l'absence de collision peut se faire ainsi.

Dans un premier temps, il est nécessaire de construire les ensembles suivants (Y_{S1} regroupe les états qui représentent une situation depuis laquelle la sortie s_1 est émise) :

$$Y_{S1} = \{y \in Y \mid \exists a \in A : (Sortie(a) = s_1), (Etat(a) = y)\}$$
$$Y_{S2} = \{y \in Y \mid \exists a \in A : (Sortie(a) = s_2), (Etat(a) = y)\}$$
$$Y_{R1} = \{y \in Y \mid \exists a \in A : (Sortie(a) = r_1), (Etat(a) = y)\}$$
$$Y_{R2} = \{y \in Y \mid \exists a \in A : (Sortie(a) = r_2), (Etat(a) = y)\}$$

Puis, il faut s'assurer que les cinq ensembles suivants sont vides :

1. En cas d'actions conditionnelles, l'absence totale de collisions ne peut être assurée.

$$Y_{S1} \cap Y_{S2} \qquad Y_{S1} \cap Y_{R2} \qquad Y_{R1} \cap Y_{S2}$$
$$Y_{S1} \cap \mathrm{Pred}\,(Y_{S2},\ (T - \mathrm{Ext}\,(Y_{R1})))$$
$$Y_{S2} \cap \mathrm{Pred}\,(Y_{S1},\ (T - \mathrm{Ext}\,(Y_{R2})))$$

$Y_{S1} \cap Y_{S2}$ regroupe les états qui représentent une situation dans laquelle les deux vérins sortent simultanément.
$Y_{S1} \cap Y_{R2}$ regroupe les états qui représentent une situation dans laquelle le premier vérin sort tandis que le second rentre.
$Y_{S1} \cap \mathrm{Pred}\,(Y_{S2},\ (T - \mathrm{Ext}\,(Y_{R1})))$ regroupe les états qui représentent une situation dans laquelle le premier vérin sort et depuis laquelle il est possible d'atteindre une situation où le second vérin sort sans passer obligatoirement par une situation où le premier vérin rentre.

Dans le cas contraire, les situations représentées par un état contenu dans l'un de ces cinq ensembles correspondent à des situations dangereuses ou qui pourront l'être.

6.4.2.3. Cas des actions mémorisées

Nous aborderons ici uniquement le cas des actions mémorisées non conditionnelles.

Au chapitre 1 section 2, nous avons rappelé que l'usage des actions mémorisées est en pratique souvent interdit car elles introduisent un parallélisme interprété de la commande (une partie de la commande du processus n'est pas directement représenté dans le modèle de commande). Il est alors très difficile de vérifier si l'usage qui en est fait est correct. A l'aide de l'automate équivalent, il est possible de vérifier si l'emploi des actions mémorisées est fonctionnel. On peut par exemple vérifier :

- qu'une mémoire n'est jamais simultanément mise à un et à zéro,
- que pour toute situation stable, il n'existe toujours qu'un seul état pour la mémoire,
- qu'une mémoire n'est jamais mise à un alors qu'elle y est déjà,
- qu'une mémoire n'est jamais mise à zéro alors qu'elle y est déjà.

Pour illustrer ces possibilités de vérification, nous supposons que Y_{set} (respectivement Y_{reset}) regroupe les états qui représentent une situation où la mémoire extérieure est mise à un (respectivement mise à zéro).

Pour vérifier qu'une mémoire extérieure n'est jamais simultanément mise à un et à zéro, il faut s'assurer que l'ensemble suivant est vide :

$$Y_{set} \cap Y_{reset}$$

Pour vérifier si pour toute situation stable, il n'existe toujours qu'un seul état pour la mémoire, il faut s'assurer que chaque état de l'automate ne peut pas à la fois être atteint depuis des états de Y_{reset} et de Y_{set} sans passer obligatoirement par des états de Y_{reset} ou depuis des états de Y_{set} et de Y_{reset} sans passer obligatoirement par des états de Y_{set}. De plus, si on suppose qu'à l'initialisation du grafcet, la mémoire est à zéro, il faut également contrôler, pour toutes les situations accessibles depuis la situation initiale sans mise à un de la mémoire, que l'état de la mémoire est ensuite toujours à zéro.

Pour vérifier ces deux propriétés, il faut s'assurer que l'ensemble suivant est vide :

$$Succ\,(Y_{set},\ (T - Int\,(Y_{reset}))) \cap Succ\,((Y_{reset} \cup Y_0),\ (T - Int\,(Y_{set})))$$

Pour vérifier qu'une mémoire extérieure n'est jamais mise à un alors qu'elle y est déjà, il faut contrôler depuis toutes les situations où la mémoire à été mise à un, qu'il n'est pas possible d'atteindre une situation où la mémoire est remise à un. Pour vérifier cette propriété, il suffit de s'assurer que l'ensemble suivant est vide :

$$(Succ\,(Y_{set},\ (T - Int\,(Y_{reset}))) - Y_{set}) \cap Pred\,(Y_{set},\ (T - Ext\,(Y_{reset})))$$

6.4.3. Propriétés relatives aux modes de marche

En GRAFCET, les modes de marche sont souvent décrits à l'aide de grafcets connexes. Grâce à l'automate équivalent, un grand nombre de propriétés relatives aux modes de marches sont vérifiables. On peut ainsi contrôler :

- l'unicité du mode de marche courant,
- la compatibilité des modes de marche entre deux parties du processus commandé,

- le respect des modes de marche (seules certaines sorties sont émises dans certains modes),
- le respect de l'enchaînement des modes,
- les conditions de changements de modes de marches.

Pour illustrer ces possibilités de vérification, nous supposons que Y_{repos}, Y_{fonc} et $Y_{défaut}$ regroupent respectivement les états qui représentent les situations dans les trois modes suivant «repos», «fonctionnement», et «défaut».

Ainsi, pour vérifier l'unicité des modes de marches, il est nécessaire et suffisant de s'assurer que l'ensemble suivant est vide :

$$(Y_{repos} \cap Y_{fonc}) \cup (Y_{repos} \cap Y_{défaut}) \cup (Y_{défaut} \cap Y_{fonc})$$

Il est également possible de vérifier que le mode «défaut» ne peut pas être quitter sans passer dans le mode «repos». Pour cela, il est nécessaire et suffisant de s'assurer que l'ensemble suivant est vide :

$$(Ext(Y_{défaut}) - Int(Y_{défaut})) - Int(Y_{repos})$$

$(Ext(Y_{défaut}) - Int(Y_{défaut}))$ regroupe les transitions dont seule l'origine est un élément de $Y_{défaut}$.

A l'aide des fonctions $Pres(X_1, T_1)$ et $Abs(X_1, D)$, il est également possible de contrôler les conditions de changements de modes de marche. Ainsi, pour vérifier que le mode «fonctionnement» ne peut être atteint que si l'entrée «dcy» est présente, il est nécessaire et suffisant de s'assurer que l'ensemble suivant est vide :

$$(Int(Y_{fonc}) - Ext(Y_{fonc})) - Pres(\{dcy\}, T)$$

Nous venons de détailler quelques unes des possibilités offertes par l'analyse de l'automate équivalent pour établir et vérifier des propriétés d'un grafcet. Pour obtenir certaines caractéristiques sur le comportement du grafcet analysé, il est parfois plus intéressant de travailler sur une image réduite de l'automate plutôt que sur l'automate lui-même. Pour cela, nous présentons maintenant une technique de **réduction d'automates suivant un critère d'observation** et son utilisation dans le cadre d'une validation de grafcets.

6.5. Les propriétés établies à l'issue d'une réduction de l'automate

Pour répondre à des besoins spécifiques pour la validation d'un grafcet, nous avons établi une technique de réduction qui permet d'obtenir une image du comportement d'un grafcet définie en observant uniquement certaines de ses sorties.

Cette méthode est conçue pour établir un automate de quelques états par regroupement de tous les états qui ne pouvaient être discernés les uns des autres en surveillant uniquement les sorties définies pour l'observation.

Proposer une réduction de l'automate pour en faciliter l'analyse est une idée déjà présente dans différents travaux comme ceux de Vergamini [Vergamini 87]. Cependant, notre technique est sensiblement différente car le critère de réduction retenu n'est pas basé sur les équivalences de traces mais sur l'observation des sorties de l'automate.

6.5.1. Présentation de la méthode de réduction

Pour que cette méthode de réduction soit opérationnelle pour l'analyse d'un grafcet, il est nécessaire qu'elle repose sur un critère de réduction possédant une définition mathématique stricte et que le résultat de cette réduction soit indépendant de l'ordre d'analyse des états et des transitions. Pour éviter tout problème, nous avons tenu à ce que la relation d'indiscernabilité entre les états de l'automate soit une relation d'équivalence (relation réflexive, symétrique et transitive).

Définition 18 :

Deux états distincts (y_1 et y_2) de l'automate sont indiscernables l'un de l'autre si et seulement si il est possible de passer de y_1 à y_2 et de y_2 à y_1 sans provoquer de variation dans les possibilités d'émission des sorties observées.

Il s'agit bien de possibilités d'émission car il ne sera pas tenu compte des conditions associée aux actions.

Le critère de réduction étant présenté, nous pouvons maintenant donner le détail de cette technique de réduction qui s'effectue en trois temps :

- partition entre les états de l'automate d'après les possibilités d'émission des sorties définies comme étant observées,
- détermination de tous les ensembles d'états indiscernables et réduction de chaque ensemble un seul état,
- détermination des transitions entre les états de l'automate réduit.

La partition entre les états de l'automate s'effectue par détermination de tous les ensembles d'états depuis lesquels un même sous-ensemble de sorties observables peut être émis. Le résultat forme bien une partition de l'ensemble des états puisque chacun de ses éléments ne fait partie que d'une et une seule classe.

A l'issue de cette partition, sont recherchés les sous-ensembles d'états indiscernables en considérant que deux états éléments de la même classe de la partition sont indiscernables l'un de l'autre s'il existe un circuit contenu entièrement dans cette classe et passant par ces deux états. Si Y_1 est l'ensemble des états contenus dans une classe et y un élément de cette classe on peut aisément démontrer que l'ensemble des états indiscernables de y est[1] :

$$\mathrm{Succ}(\{y\}, (\mathrm{Int}(Y_1) \cap \mathrm{Ext}(Y_1))) \cap \mathrm{Pred}(\{y\}, (\mathrm{Int}(Y_1) \cap \mathrm{Ext}(Y_1)))$$

Une fois tous les sous-ensembles indiscernables définis, il est associé à chacun d'entre-eux un "macro-état" de l'automate réduit.

Pour déterminer l'existence de transitions entre les états de l'automate réduit, il suffit de rechercher pour chaque couple (y_1, y_2) d'états de l'automate réduit l'ensemble T_{1_2} des transitions de l'automate initial pour lesquelles l'origine est contenue dans le sous-ensemble que représente y_1 et dont l'extrémité est contenue dans le sous-ensemble que représente y_2. La condition associée à cette transition issue de y_1 et aboutissant sur y_2 est alors la somme des conditions des transitions de T_{1_2}.

1. Dans la pratique, le calcul des ensembles d'états indiscernables s'effectue également de cette façon.

6.5.2. Utilisation dans le cadre d'une validation

A l'aide de cette image synthétique de l'automate équivalent, l'analyste peut cerner le comportement de son modèle. En choisissant judicieusement l'ensemble des sorties observées, la lecture de l'automate réduit peut lui permettre de vérifier de nombreuses propriétés. Dans le deuxième exemple présenté au chapitre7, nous nous sommes servis de cette technique de réduction pour vérifier le respect des sécurités de fonctionnement pour chaque sortie et de la procédure de mise en route (les automates réduits suivant l'observation des différentes sorties sont également présentés).

Cependant, il est nécessaire d'insister sur l'absence d'équivalence stricte entre l'automate initial et sa réduction, absence due à la perte d'informations qui se produit nécessairement lors de la réduction. Cette absence d'équivalence stricte nécessite donc de prendre certaines précautions lors de l'élaboration des conclusions sur l'analyse d'un automate réduit.

Par exemple, la condition associée à une transition de l'automate réduit n'est qu'une **condition nécessaire** pour accéder à l'ensemble d'états indiscernables mais pas une condition nécessaire et suffisante puisqu'elle est obtenue par une réunion des conditions d'un ensemble de transitions de l'automate initial. Ainsi, sa lecture permet de vérifier si certaines contraintes d'évolution sont bien respectées mais ne permet pas forcément de conclure qu'elles ne le sont pas !

De plus, en raison de la méthode de réduction, les caractéristiques suivantes de l'automate initial ne sont plus forcément respectés :

- $\forall (t_1, t_2) \in T^2, (t_1 \neq t_2) : (Am(t_1) = Am(t_2)) \Rightarrow (Cond(t_1) \cdot Cond(t_2) = 0)$
- $\forall y \in Y - Y_0 : \sum_{t_i,\, Am(t_i) = y} Cond(t_i) \neq 1$
- Toute occurrence d'un événement permettant d'accéder à un état, ne peut permettre de le quitter.

En prenant quelques précautions lors de l'analyse de la réduction de l'automate, de nombreuses propriétés d'un grafcet sont vérifiables de manière «ergonomique» à

partir de la réduction de son automate équivalent. Nous allons en donner un exemple dans le chapitre qui suit.

Chapitre 7
Exemples de validation de grafcets

Dans ce chapitre, nous présentons deux exemples de validation de grafcets que nous avons réalisés à l'aide de notre maquette informatique «AGGLAÉ» et l'outil de vérification des systèmes de transitions «MEC» développé par le LABRI de Bordeaux.

Nous avons retenu ces deux exemples pour leur complémentarité. Le premier est le «problème des philosophes», problème classique en informatique comme en automatique. Le deuxième, est un exemple industriel : un «problème de distribution d'eau» pour lequel il existe des contraintes importantes relatives aux modes de marches et aux procédures de fonctionnement.

7.1. Le problème des philosophes

7.1.1. Références et intérêts

Ce problème intitulé "les philosophes mangeurs de riz" ou bien "les philosophes mangeurs de spaghetti" est souvent cité pour illustrer les subtilités de la synchronisation des programmes parallèles. Ce problème traite du partage de plusieurs ressources communes et de ses conséquences comme le blocage d'une ressource ou la non accessibilité à l'une des ressources pour l'un des processus (on parle alors de famine).

Dans [Vergamini 87], ce problème est l'exemple support pour le chapitre 3 relatif au principe de vérification de processus. L'auteur donne un tableau relatif à la taille de l'automate obtenu en fonction du nombre de philosophes traités. Dans [Arnold 92b], les auteurs citent également cet exemple comme étant systématiquement utilisé pour tester la puissance et l'efficacité de nouvelles primitives de synchronisation. Ce problème est modélisé en RdP dans [Denham 88] [David 89].

Nous avons souhaité modéliser ce problème en GRAFCET à la fois pour nous permettre de positionner nos travaux par rapport à d'autres recherches dans des domaines connexes et également parce qu'il nous permet, en faisant varier le nombre de philosophe en présence, de traiter des problèmes de complexité croissante.

7.1.2. Définition du problème [Arnold 92b]

N philosophes se partagent une table ronde. Autour de la table se trouvent n chaises, chacune appartenant à un philosophe, et sur la table sont disposées n baguettes, n assiettes et un plat de riz, qui est supposé toujours plein (cf. figure 47).

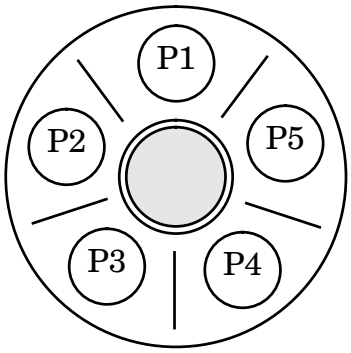


figure 47. La table des philosophes (illustration à l'ordre 5)

Chaque philosophe passe alternativement par des phases où *il pense* à des phases où *il a faim* et où *il tente de manger*. Ainsi, chaque philosophe pense et, de temps en temps, lorsqu'il a faim essaie de prendre les deux baguettes situées de part et d'autre de son assiette pour manger du riz. Si un des deux philosophes voisins a déjà pris une baguette, l'opération est impossible.

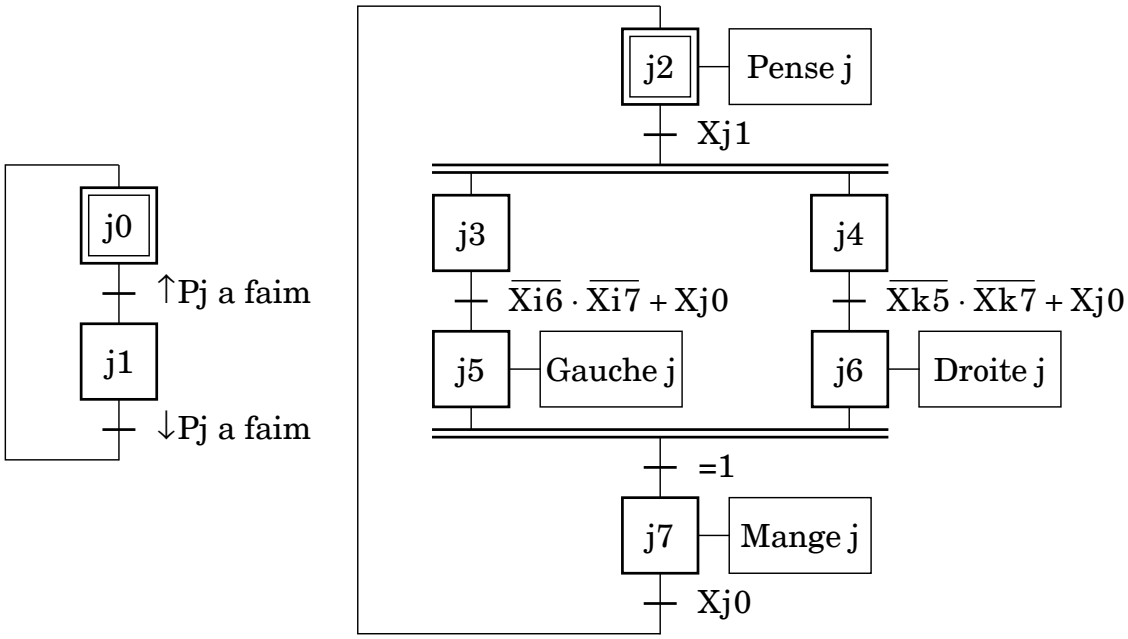
Il est demandé de modéliser le comportement de chacun des philosophes en évitant les blocages et la famine (au sens propre !) d'un philosophe.

7.1.3. Modélisation

Nous proposons une modélisation de ce problème en GRAFCET dans laquelle tous les philosophes ont un comportement rigoureusement identique, comportement défini par deux grafcets connexes.

A chaque philosophe j est associé une entrée «Pj a faim» et quatre sorties «Pense j» «Gauche j», «Droite j» et «Mange j» traduisant le fait que ce philosophe pense, détient seulement la baguette qui est à sa gauche ou à sa droite, ou qu'il est en train de manger.

Le comportement de chaque philosophe est ainsi décrit par le modèle présenté sur la figure 48.



i est le numéro du philosophe placé à gauche du philosophe j ($i = j - 1$)

k est le numéro du philosophe placé à droite du philosophe j ($k = j + 1$)

figure 48. Modélisation du philosophe j

Dans cette modélisation, nous avons permis à un philosophe d'abandonner sa quête de nourriture en considérant qu'il pouvait à tout moment, retourner dans l'état ou il pense.

Pour cinq convives, la modélisation complète est donnée figure 49.

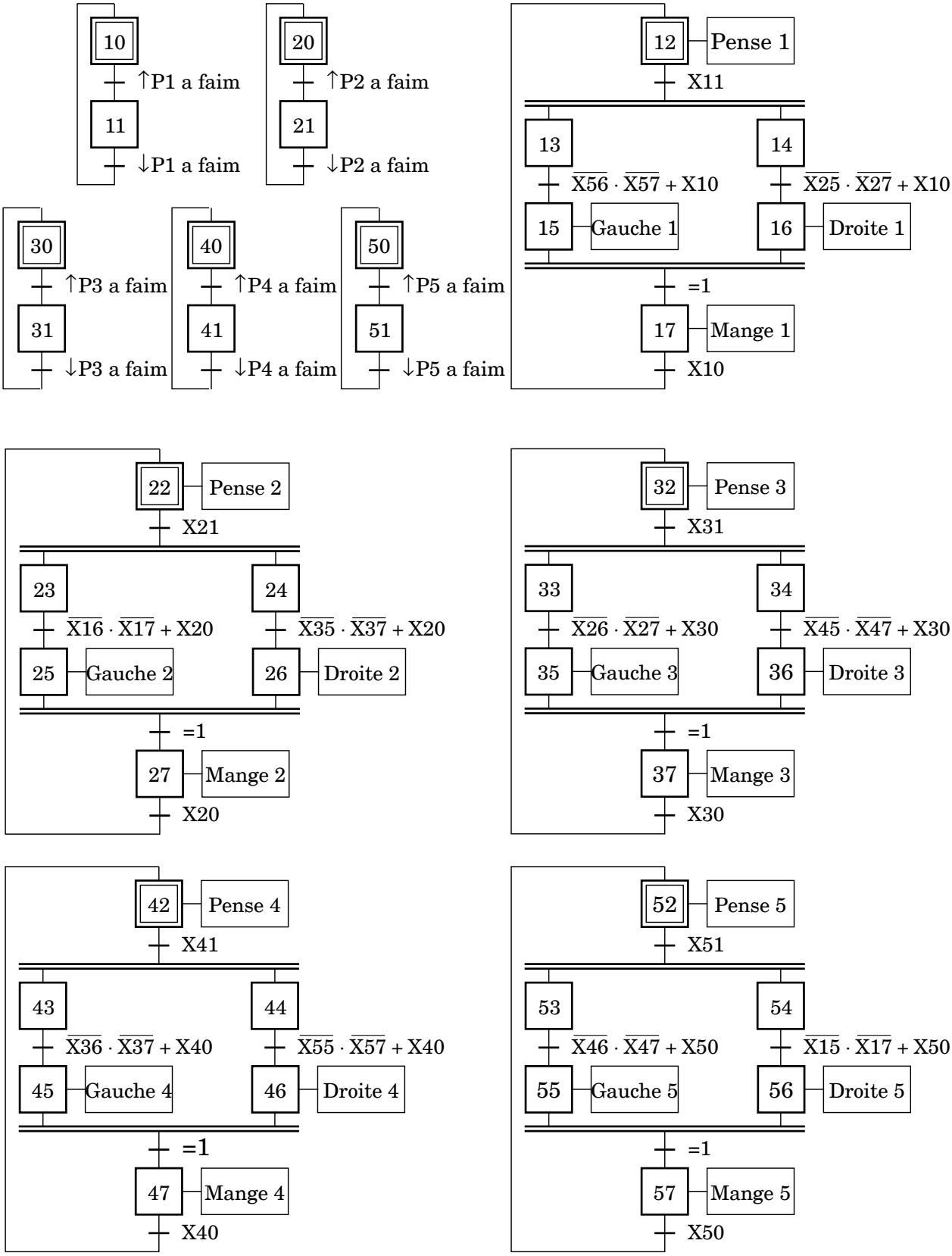


figure 49. Modélisation du problème à l'ordre 5

7.1.4. Validation

7.1.4.1. Validation du grafcet par analyse de l'automate équivalent

Une fois la modélisation effectuée, il est nécessaire de procéder à sa validation, qui consiste à vérifier :

- qu'une baguette n'est jamais utilisée par deux philosophes en même temps,
- qu'un philosophe qui a faim essaie toujours de prendre les baguettes,
- qu'aucun blocage n'est possible lorsque personne ne souhaite abandonner,
- qu'aucune alliance entre philosophes n'est possible pour en affamer un autre.

Pour effectuer ces vérifications sur l'automate équivalent, nous allons devoir au préalable construire les ensembles d'états et de transitions suivants :

Définition donnée en terme Grafcet	Définition mathématique
Etats dans lesquels le philosophe j pense.	$P_j = Act(\{Xj2\})$
Etats dans lesquels le philosophe j a faim mais ne tient pas de baguettes.	$F_j = Act(\{Xj3, Xj4\})$
Etats dans lesquels le philosophe j ne détient que la baguette droite.	$G_j = Act(\{Xj5\})$
Etats dans lesquels le philosophe j ne détient que la baguette gauche.	$D_j = Act(\{Xj6\})$
Etats dans lesquels le philosophe j mange.	$M_j = Act(\{Xj7\})$
Transitions franchies lorsque le philosophe j a faim	$T_{fj} = \{t \in T \mid (Cond(t) = \uparrow Pj \text{ a faim})\}$
Transitions franchies lorsque le philosophe j est repu	$T_{rj} = \{t \in T \mid (Am(t) \in M_j), (Cond(t) = \downarrow Pj \text{ a faim})\}$
Transitions franchies lorsque le philosophe j abandonne	$T_{aj} = \{t \in T \mid (Am(t) \notin M_j), (Cond(t) = \downarrow Pj \text{ a faim})\}$

tableau 10 Paramétrage des états et des transitions de l'automate équivalent

En utilisant l'outil MEC, il nous est possible de vérifier que les cinq ensembles d'états P_j, F_j, G_j, D_j, M_j sont deux à deux disjoints et que l'union des cinq est l'ensemble de tous les états possibles. En tenant compte de ces informations, nous pouvons maintenant facilement vérifier les contraintes imposées par le cahier des charges.

Pour prouver qu'une baguette n'est jamais utilisée par deux philosophes en même temps, il suffit de vérifier que l'ensemble $(D_j \cup M_j) \cap (G_k \cup M_k)$ avec $(k = j + 1)$ est vide.

Pour prouver qu'un philosophe qui a faim essaie toujours de prendre les baguettes, il suffit de vérifier que les ensembles suivants sont vides :

$$\begin{aligned} F_j &- ((G_k \cup M_k) \cap (D_i \cup M_i)) \\ G_j &- (G_k \cup M_k) \\ D_j &- (D_i \cup M_i) \end{aligned} \qquad \begin{cases} i = j - 1 \\ k = j + 1 \end{cases}$$

Pour prouver qu'aucun blocage n'est possible lorsque personne ne souhaite abandonner, il faut vérifier que l'ensemble suivant est vide :

$$Y - \text{Amont}\left(T - \bigcup_{i=1}^{n} T_{aj} \right)$$

La dernière propriété consiste à vérifier l'impossibilité pour un ensemble de philosophes de manger à leur guise tout en empêchant systématiquement le philosophe j de disposer des baguettes. Pour contrôler cette propriété, il suffit de s'assurer que lorsque l'un des philosophes voisins de j sera rassasié, alors nécessairement il ne pourra plus empêcher le philosophe j de manger en utilisant les baguettes. Pour cela, il suffit de démontrer que l'ensemble $Y_1 \cup Y_2$ est vide où :

$$\begin{aligned} Y_1 &= (\text{Succ}((\text{Aval}(T_{ri}) \cap (F_j \cup D_j \cup G_j)), T_1) \cap (D_i \cup M_i)) \\ Y_2 &= (\text{Succ}((\text{Aval}(T_{rk}) \cap (F_j \cup D_j \cup G_j)), T_1) \cap (G_k \cup M_k)) \\ T_1 &= \text{Int}(F_j \cup D_j \cup G_j \cup M_j) \cap \text{Ext}(F_j \cup D_j \cup G_j) \end{aligned}$$

T_1 est l'ensemble des transitions franchissables lorsque le philosophe j attend pour manger.

$(\text{Aval}(T_{ri}) \cap (F_j \cup D_j \cup G_j))$ est l'ensemble des états atteints lorsque le philosophe i est repu et que le philosophe j attend pour manger.

7.1.4.2. Validation de l'automate équivalent en raisonnant sur le cahier des charges

Dans le tableau 11, nous avons récapitulé le nombre d'états et de transitions de l'automate équivalent pour une modélisation du problème des philosophes de 3 à 10 convives.

Philosophes	3	4	5	6	7	8	9	10
Etats	16	45	121	320	841	2 205	5 776	15 125
Transitions	48	131	605	1820	5 887	17 640	51 984	151 250

tableau 11 Nombre d'états et de transitions de l'automate équivalent pour une modélisation du problème des philosophes de 3 à 10 convives

Si à l'ordre 3, il est encore possible d'envisager une vérification de l'automate équivalent par une construction manuelle de celui-ci, cela est inenvisageable pour les ordres supérieurs. Nous avons préféré justifier la pertinence de ces résultats en raisonnant directement à partir du cahier des charges. Pour ce problème, nous pouvons borner le nombre d'états et disposer d'une relation liant le nombre d'états et de transitions :

- $Nbre_{Etats} > 2^n$ en considérant que 2^n est le nombre de combinaisons possibles concernant l'appétit des philosophes (les philosophes peuvent avoir faim sans nécessairement pouvoir manger).
- $Nbre_{Etats} < 3^n$ en considérant que 3^n est le nombre de combinaisons possibles concernant l'état des baguettes (chaque baguette est au plus dans trois états «sur la table», «dans la main du philosophe de gauche», «dans la main du philosophe de droite»).
- $Nbre_{Transitions} = n \cdot Nbre_{Etats}$ puisque chaque état est sensible à un changement d'appétit de chaque philosophe.

D'autre part, nous avons pu également vérifier le nombre d'états par comparaison avec une analyse systématique qui a consisté à prévoir toutes les combinaisons possibles pour l'ensemble des philosophes («repos», «affamé sans baguette», «muni de la baguette droite», «muni de la baguette gauche», «en train de manger») puis à

supprimer toutes celles qui sont impossibles. Le tableau 12 donne, en fonction de l'état d'un philosophe, les états possibles du philosophe à sa droite.

Etats d'un philosophe	Etats possibles pour son voisin de droite
repos	repos muni de la baguette gauche en train de manger
affamé sans baguette	muni de la baguette gauche en train de manger
muni de la baguette droite	repos affamé sans baguette muni de la baguette droite
muni de la baguette gauche	muni de la baguette gauche en train de manger
en train de mangé	repos affamé sans baguettes muni de la baguette droite

tableau 12 Relation de compatibilité entre les états de philosophes voisins

A chaque fois, le nombre de combinaisons restantes est égal au nombre d'états trouvés majoré de deux. Ces deux combinaisons supplémentaires correspondent à des blocages (chaque philosophe à une baguette dans la main droite ou dans la main gauche).

7.1.5. Eléments de comparaison

Nous nous sommes donc servis de cet exemple pour tester la maquette informatique qui génère l'automate équivalent. Dans le tableau 13 nous avons récapitulé les principaux résultats de ces tests.

Nombre de philosophes	Nombre de simplifications effectuées	Nombre de situations analysées	Nombre d'états	Nombre de transitions	Temps de calcul
3	1 491	142	16	48	1 s
4	7 202	541	45	180	4 s
5	29 605	1 826	121	605	19 s
6	110 784	5 774	320	1 920	1 min 23 s
7	390 824	17 613	841	5 887	5 min 13 s
8	1 323 636	52 517	2 205	17 640	19 min 55 s
9	4 349 295	154 105	5 776	51 984	1 h 18 min
10	13 958 505	446 765	15 125	151 250	5 h 25 min

tableau 13 Données relatives à la génération de l'automate équivalent pour le problème des philosophes (de 3 à 10)

Sur la figure 50, nous avons représenté le facteur de croissance entre une modélisation à l'ordre i et une modélisation à l'ordre i-1 pour chacun des paramètres du tableau 13. A l'exception du facteur temps[1], tous les facteurs multiplicatifs sont convergents. Pour ce problème, il est donc possible de prévoir de manière assez fine le nombre de simplifications qu'il sera nécessaire d'effectuer et le nombre de situations à analyser en fonction du nombre de philosophes. Cette particularité fait de ce problème un très bon exemple pour des tests de performance.

1. La remontée de la courbe facteur temps est due à une gestion de la mémoire de la machine non optimale qui conduit à une utilisation intempestive de la zone de mémoire écrite sur le disque.

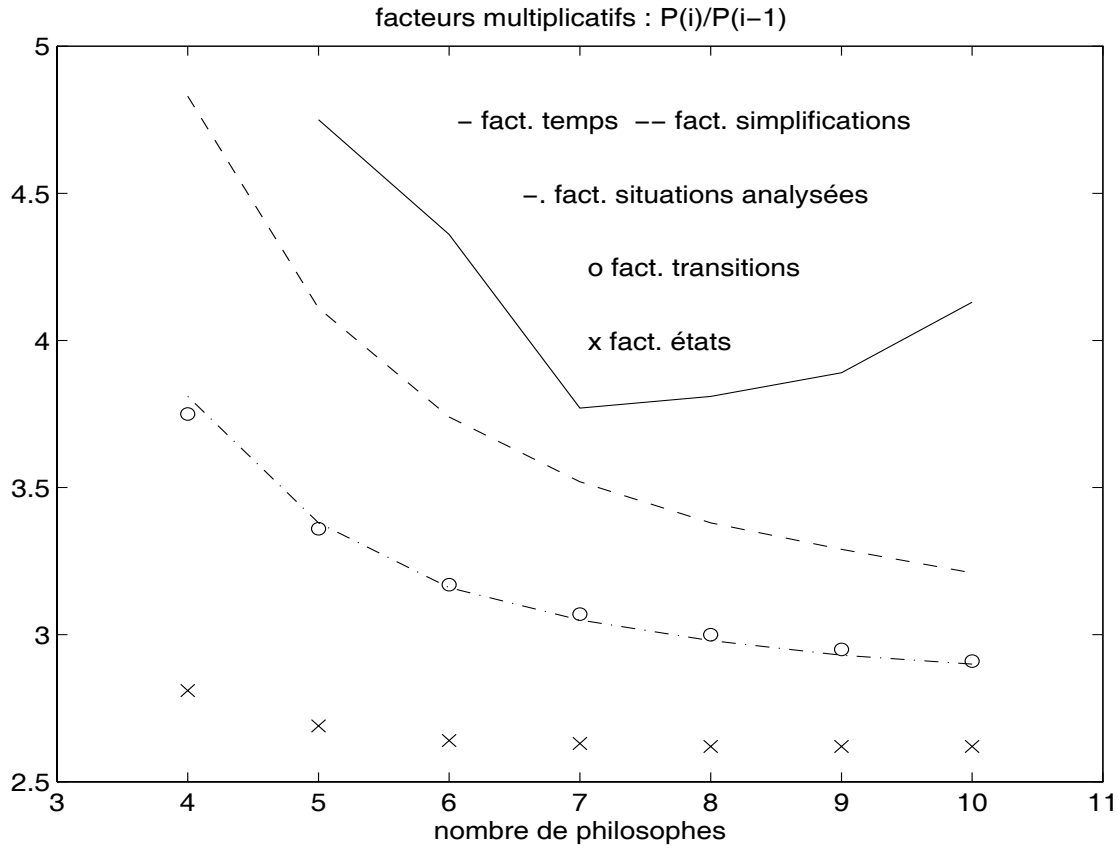


figure 50. Facteurs de croissance pour le problème des philosophes.

Dans [Vergamini 87], il est donné la taille de l'automate obtenu avec une modélisation à l'aide de processus. Sur la figure 51, nous présentons une comparaison des deux approches. En 1987, pour la modélisation à l'aide de processus, les possibilités informatiques ne permettaient que la génération à l'ordre 5. En 1994, nous pensons qu'il en serait autrement mais, à notre connaissance, aucun autre essai n'a été depuis réalisé.

Cependant pour 5 philosophes, la modélisation en processus donne déjà un automate de 3 111 états et de 53 500 transitions. Sur ce plan, une modélisation en GRAFCET est nettement plus performante puisque le nombre d'états et de transitions est bien moindre. Cette performance est due à la puissance d'expression qu'apporte la notion de réceptivité et la recherche de stabilité qui permet d'éliminer tous les états intermédiaires.

Sur cet exemple issu de l'informatique, le GRAFCET s'avère donc être un modèle particulièrement performant pour la modélisation de systèmes parallèles.

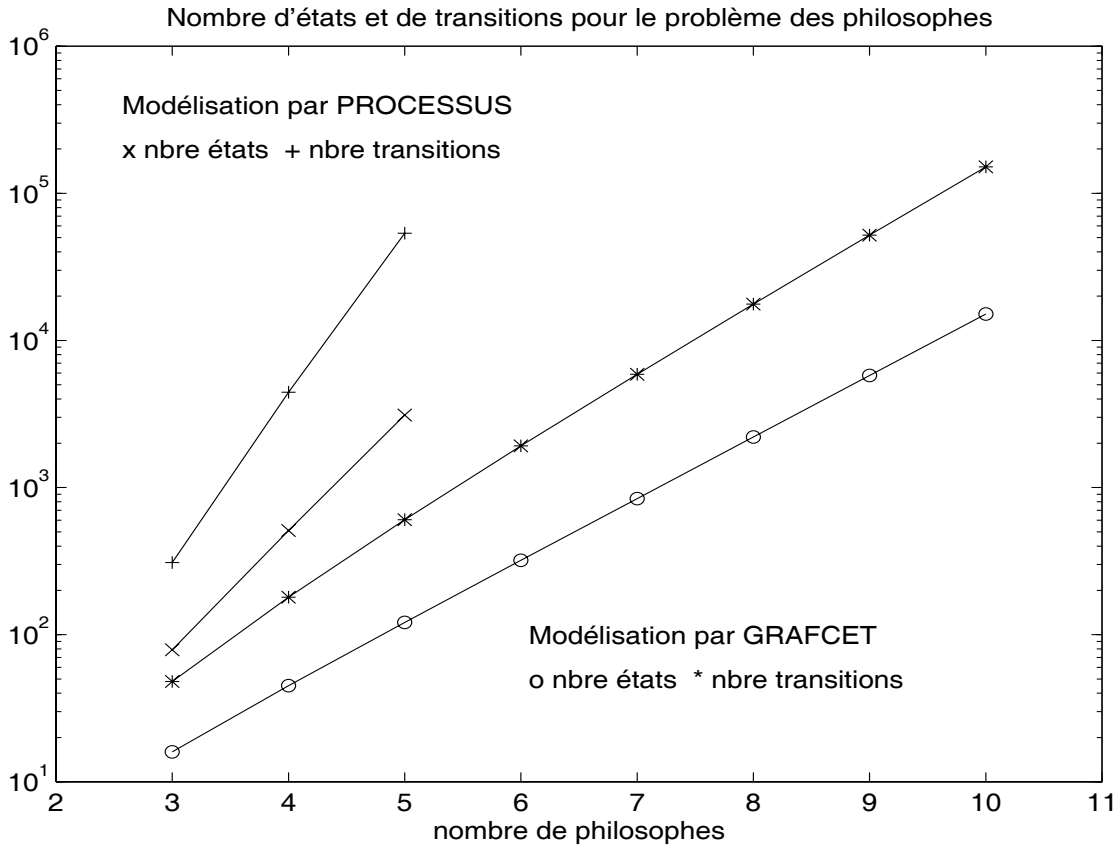


figure 51. Nombre d'états et de transitions pour le problème des philosophes.

7.2. Problème de distribution d'eau

7.2.1. Présentation du problème

L'exemple traité concerne la distribution d'eau contenue dans un réservoir unique et utilisé par plusieurs chaînes de production. L'installation physique comprend le réservoir, deux pompes, six vannes et les dispositifs de distribution. Il est demandé de gérer le démarrage et l'arrêt des pompes, ainsi que l'ouverture et la fermeture des vannes, en tenant compte des pannes éventuelles des pompes, des problèmes qui peuvent survenir sur le circuit et des différentes requêtes de distribution.

Chaque pompe possède deux vannes (une en amont et une en aval). La vanne de sortie, comme celle de refoulement, est commune aux deux pompes (cf. figure 52).

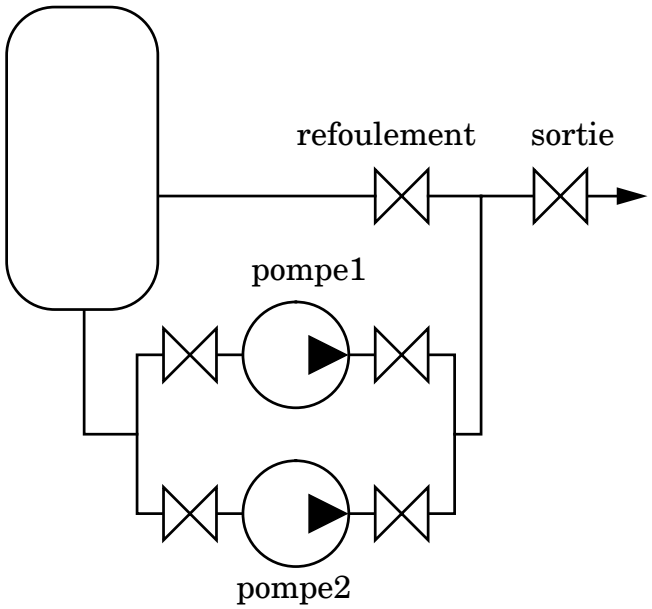


figure 52. Vue globale de l'installation

Ce problème, posé par un industriel spécialiste de la distribution d'eau, nous a été confié par la société ILOG qui l'a elle même utilisé pour montrer la performance de son logiciel AGEL, atelier de développement de systèmes réactifs. Pour mettre en valeur les possibilités de ce vérificateur de programmes, il était présenté sur cet exemple, des propriétés de comportement vérifiables en visualisant l'automate réduit équivalent à l'automate d'origine issu d'une modélisation en Esterel.

7.2.2. Le cahier des charges

Il s'agit d'une mise en forme du cahier des charges fourni par la société ILOG.

- Les requêtes de distribution

II y a deux types de requêtes de distribution : une requête «bas débit» et des requêtes «haut débit», en nombre non précisé. On ne peut servir qu'une requête «haut débit» à la fois mais on peut servir simultanément une requête «bas débit» et une requête «haut débit».

- Les pompes

Il y a deux pompes, appelées «pompe1» et «pompe2». Les deux pompes ne fonctionnent jamais en même temps, on utilise l'une ou l'autre. En fonctionnement normal, chaque pompe fonctionne pendant 24 heures. Chaque jour, on commute d'une pompe sur l'autre pour assurer un fonctionnement équilibré des deux pompes. Cette commutation n'est effectuée qu'à la fin de toutes les distributions en cours.

Une pompe peut tomber en panne. Lorsque la pompe qui fonctionne est en panne, on stoppe toute distribution et on ferme toutes les vannes qui ont été ouvertes. Lors de la prochaine requête, on utilisera l'autre pompe si elle est en état de fonctionner. Lorsque la pompe du jour sera en état, on attendra la fin de toutes les requêtes de distribution en cours et on commutera pour l'utiliser à nouveau.

- Les vannes

Chaque pompe a deux vannes associées ; ces vannes s'appellent «amont pompe1», «aval pompe1», «amont pompe2», «aval pompe2».

La vanne «sortie» est commune aux deux pompes.

La vanne «refoulement» est ouverte lorsqu'on ne sert que la requête «bas débit». Si on sert une requête «haut débit», ou une requête «haut débit» en même temps que la requête «bas débit», elle est fermée.

- La panne générale

On peut détecter une panne générale du circuit (concrètement, il s'agit de la détection d'une baisse importante du débit qui signale un engorgement). Dans ce cas,

il faut interrompre les distributions en cours, arrêter la pompe utilisée et fermer toutes les vannes. On ne répond plus à aucune requête jusqu'à la fin de la panne.

- Le fonctionnement du système

Décrivons le fonctionnement du système lorsqu'arrive une requête de distribution, et que l'on dispose d'une pompe pour servir la requête.

Si la pompe est arrêtée lorsqu'on reçoit la requête de distribution, on ouvre la vanne «sortie» et la vanne en amont à la pompe. On attend 5 secondes. Après ce délai, on démarre la pompe, et on ouvre la vanne en aval. La distribution demandée commence.

Si la pompe est en marche, la distribution demandée commence immédiatement.

Lorsqu'on reçoit une fin de requête de distribution, on arrête la distribution. Si c'est la dernière en cours, on arrête la pompe et on ferme toutes les vannes.

7.2.3. Modélisation

7.2.3.1. Remarques préalables

Les requêtes «haut débit» ont été regroupées sous une seule entrée. Ce regroupement n'est pas pénalisant car le comportement du système ne diffère pas selon le nombre de requêtes «haut débit» et leur provenance.

Les requêtes «haut débit» et la requête «bas débit» sont considérées comme des informations à valeur booléenne. La notion de fin de requête n'existe donc pas.

7.2.3.2. Le grafcet

La modélisation en GRAFCET est présentée sur la figure 53. Dans cette modélisation, les grafcets connexes «MMP1» (étapes 10, 11, 12) et «MMP2» (étapes 20, 21, 22) décrivent les modes de marche des pompes 1 et 2. Le grafcet connexe «ChP» (étapes 30, 31) gère le changement de pompes tandis que le grafcet connexe «Ref» (étapes 40, 41) gère le fonctionnement de la vanne de refoulement. Le grafcet «Fonc» (étapes 50, 51, 52, 53, 54) décrit les procédures de mise en route.

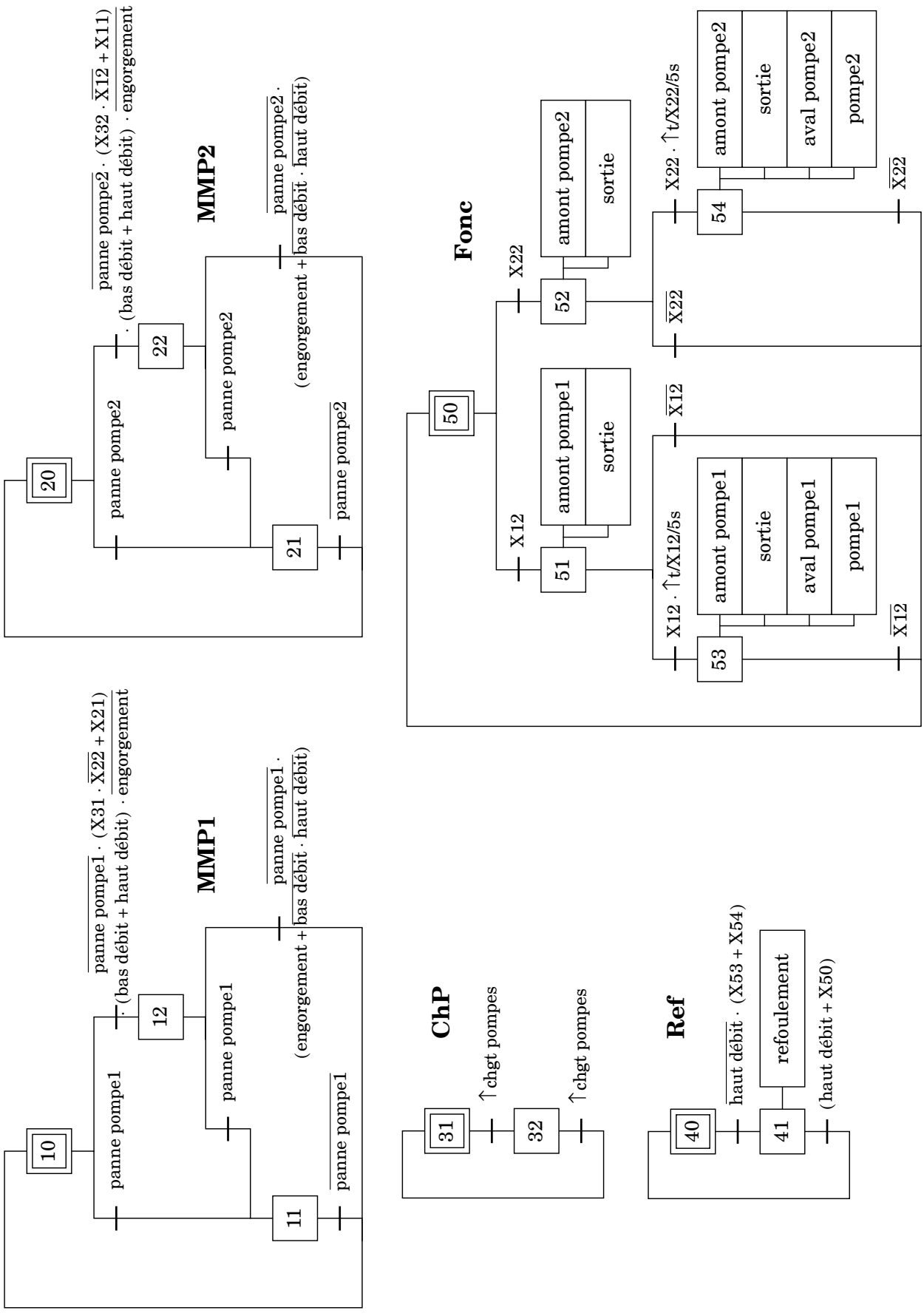


figure 53. Modélisation du problème de distribution d'eau

7.2.4. Validation

7.2.4.1. Quelques chiffres sur la génération

Pour cette application, la génération de l'automate équivalent demande moins d'une minute durant laquelle il a été nécessaire d'envisager les possibilités d'évolutions depuis 728 situations et d'effectuer 12 762 calculs d'expressions combinatoires.

Lors de la génération, sont identifiés 32 situations stables différentes et 752 possibilités d'évolutions entre ces situations. L'automate construit ne se compose que de 32 états et de 512 transitions, car parmi les 752 possibilités d'évolutions, certaines sont parallèles (même origine y_1 et même but y_2).

Cet automate a été ensuite réduit pour tenir compte de l'historique des entrées. A l'issue de cette opération, l'automate ne présente plus que 32 états et 171 transitions. C'est sur cet automate que sera réalisée la validation. Il convient de signaler que la modélisation en Esterel conduit à un automate de plus de 1 000 états.

7.2.4.2. Validation du comportement modélisé

Pour cette application, il nous a été possible de vérifier en analysant l'automate équivalent un certain nombre de propriétés d'ordre général comme l'absence de situation totalement instable, de possibilité de blocage, le respect de l'unicité des modes de marches.

Il nous a été également permis de vérifier des propriétés plus spécifiques comme :

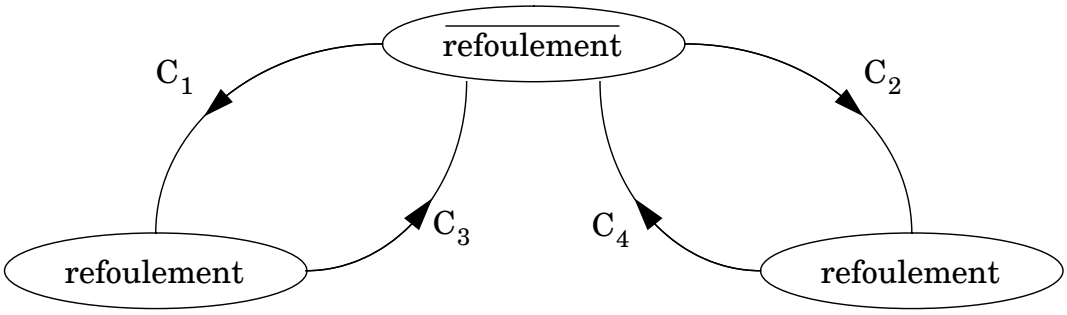
- l'absence de situations où les étapes 12 et 22 sont actives simultanément (les deux pompes ne sont jamais simultanément dans le mode de marche «fonctionnement»),
- les deux sorties «pompe1» et pompe2» ne sont jamais émises simultanément,
- la sortie «refoulement» n'est émise que conjointement avec les sorties «pompe1» ou «pompe2».

En utilisant les possibilités de réduction de l'automate, on a pu obtenir les automates présentés de la figure 54 à la figure 57. Grâce à ces automates réduits, nous avons pu contrôler que les conditions nécessaires pour l'ouverture et la fermeture des vannes sont bien vérifiées.

7.2.4.3. Quelques remarques sur les automates réduits

Sur l'automate réduit présenté figure 54, il existe deux états dans lesquels la sortie «refoulement» est émise. Cette particularité traduit le fait qu'il n'est pas possible de passer d'un état de l'automate initial émettant les sorties «refoulement» et «pompe1» à un état émettant les sorties «refoulement» et «pompe2» sans passer par un état ou la sortie «refoulement» n'est pas émise.

Sur certains automates réduits, certaines conditions d'évolutions ne portent pas sur des événements mais sur des valeurs des entrées. Ce phénomène est du à la méthode proposée pour prendre en compte la chronologie des entrées dans laquelle il n'est pas prévu d'affiner les conditions des évolutions issues de la situation initiale.



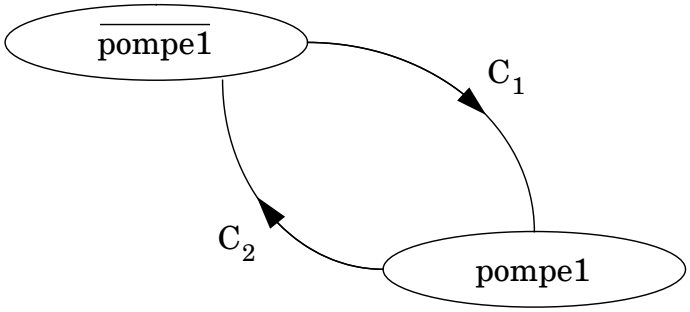
$$C_1 = \overline{\text{panne pompe1}} \cdot \overline{\text{engorgement}} \cdot \text{bas debit} \cdot \left(\left(\overline{\text{haut debit}} \cdot \uparrow \text{t/X12/5s} \right) + \downarrow \text{haut debit} \right)$$

$$C_2 = \overline{\text{panne pompe2}} \cdot \overline{\text{engorgement}} \cdot \text{bas debit} \cdot \left(\left(\overline{\text{haut debit}} \cdot \uparrow \text{t/X22/5s} \right) + \downarrow \text{haut debit} \right)$$

$$C_3 = \downarrow \left(\overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}} \cdot \text{bas debit} \cdot \overline{\text{haut debit}} \right)$$

$$C_4 = \downarrow \left(\overline{\text{engorgement}} \cdot \overline{\text{panne pompe2}} \cdot \text{bas debit} \cdot \overline{\text{haut debit}} \right)$$

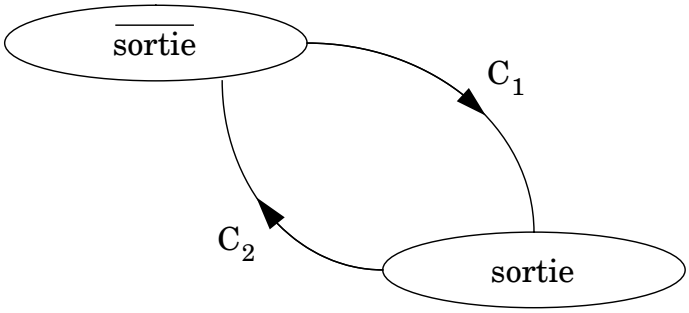
figure 54. Automate réduit en observant la sortie «refoulement»



$$C_1 = \overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}} \cdot \uparrow t/X12/5s \cdot (\text{bas debit} + \text{haut debit})$$

$$C_2 = \downarrow ((\text{bas debit} + \text{haut debit}) \cdot \overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}})$$

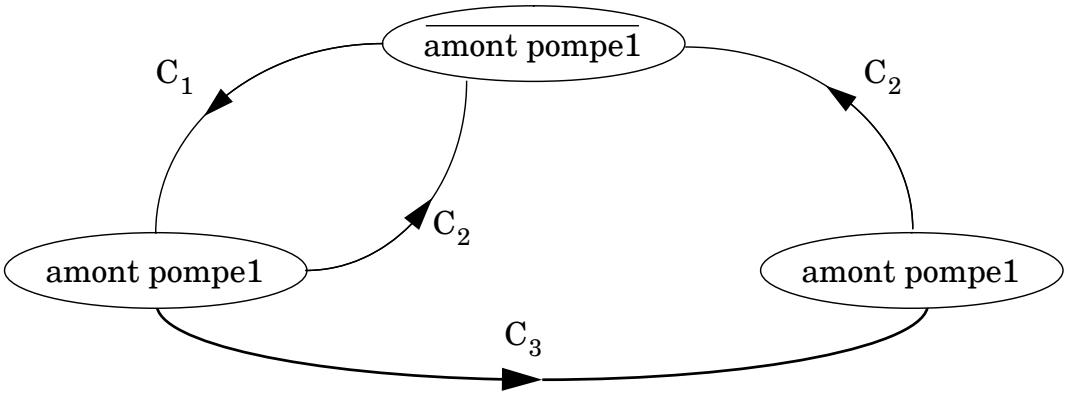
figure 55. Automate réduit en observant la sortie «pompe1»



$$C_1 = \overline{\text{engorgement}} \cdot (\text{bas debit} + \text{haut debit}) \cdot (\overline{\text{panne pompe1}} + \overline{\text{panne pompe2}})$$

$$C_2 = \downarrow (\overline{\text{engorgement}} \cdot (\text{bas debit} + \text{haut debit}) \cdot (\overline{\text{panne pompe1}} + \overline{\text{panne pompe2}}))$$

figure 56. Automate réduit en observant la sortie «sortie»



$C_1 = \overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}} \cdot (\text{bas debit} + \text{haut debit})$

$C_2 = \downarrow((\text{bas debit} + \text{haut debit}) \cdot \overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}})$

$C_3 = \overline{\text{engorgement}} \cdot \overline{\text{panne pompe1}} \cdot \uparrow t/X12/5s \cdot (\text{bas debit} + \text{haut debit})$

figure 57. Automate réduit en observant la sortie «amont pompe1»

7.2.4.4. Fichiers résultats

Nous avons reproduit dans cette section, les fichiers ou des extraits de fichiers générés par AGGLAÉ.

Fichier d'informations concernant la génération

```
Temps de calcul en secondes : 58
Nombre de simplification d'expressions combinatoires effectuees : 12762
Nombre de situations analysees : 728
Nombre de situations stables : 32
Nombre de situations totalement instables : 0
Nombre d'evolutions a l'echelle externe : 752
Generation effectuee le 25/07/94 de 16:18:18 a 16:19:18
```

Extraits du fichier décrivant l'automate équivalent

```
Situation 1 : X10 X20 X31 X40 X50
Situation 2 : X10 X21 X31 X40 X50
Situation 3 : X10 X20 X32 X40 X50
Situation 4 : X11 X21 X31 X40 X50
Situation 5 : X10 X21 X32 X40 X50
Situation 6 : X11 X21 X32 X40 X50
Situation 7 : X11 X20 X31 X40 X50
Situation 8 : X12 X20 X31 X40 X51
Situation 9 : X11 X20 X32 X40 X50
Situation 10 : X12 X21 X31 X40 X51
Situation 11 : X12 X20 X32 X40 X51
Situation 12 : X12 X21 X32 X40 X51
Situation 13 : X11 X22 X31 X40 X52
Situation 14 : X11 X22 X32 X40 X52
```

```
Situation 15 : X10 X22 X32 X40 X52
Situation 16 : X10 X22 X31 X40 X52
Situation 17 : X12 X20 X31 X41 X53
Situation 18 : X12 X20 X31 X40 X53
Situation 19 : X12 X21 X31 X41 X53
Situation 20 : X12 X21 X31 X40 X53
Situation 21 : X12 X20 X32 X41 X53
Situation 22 : X12 X20 X32 X40 X53
Situation 23 : X12 X21 X32 X41 X53
Situation 24 : X12 X21 X32 X40 X53
Situation 25 : X11 X22 X31 X41 X54
Situation 26 : X11 X22 X31 X40 X54
Situation 27 : X10 X22 X31 X41 X54
Situation 28 : X10 X22 X31 X40 X54
Situation 29 : X11 X22 X32 X41 X54
Situation 30 : X11 X22 X32 X40 X54
Situation 31 : X10 X22 X32 X41 X54
Situation 32 : X10 X22 X32 X40 X54
...
Or : 2 Ex : 4 Cd : (OU (ET panne_pompe2 (FM panne_pompe1 ) (NON bas_debit
) (NON haut_debit ) ) (ET panne_pompe2 (FM panne_pompe1 ) engorgement ) )
Or : 2 Ex : 1 Cd : (OU (ET (NON haut_debit ) (NON bas_debit ) (NON
panne_pompe1 ) (FD panne_pompe2 ) ) (ET engorgement (NON panne_pompe1 ) (FD
panne_pompe2 ) ) )
Or : 2 Ex : 5 Cd : (OU (ET panne_pompe2 (NON haut_debit ) (NON bas_debit )
(NON panne_pompe1 ) (FM chgt_pompes ) ) (ET panne_pompe2 engorgement (NON
panne_pompe1 ) (FM chgt_pompes ) ) )
Or : 2 Ex : 10 Cd : (OU (ET panne_pompe2 (NON panne_pompe1 ) (NON
engorgement ) (NON bas_debit ) (FM haut_debit ) ) (ET panne_pompe2
haut_debit (NON panne_pompe1 ) (FD engorgement ) ) (ET panne_pompe2 (NON
panne_pompe1 ) (NON engorgement ) (FM bas_debit ) (NON haut_debit ) ) (ET
panne_pompe2 bas_debit (NON panne_pompe1 ) (FD engorgement ) ) )
...
```

Fichier contenant les résultats de l'analyse enregistrés par l'analyste

```
Degre interieur et exterieur de chaque situation :
Situation : 1  deg-int : 9 deg-ext : 13
Situation : 2  deg-int : 6 deg-ext : 4
...
Situation : 32  deg-int : 4 deg-ext : 5
Pas de situation source.
Pas de blocage.

Situation comprenant        : X12 X22
          ne comprenant pas :
Il existe 0 situation(s).

Situation comprenant        : X53 X54
          ne comprenant pas :
Il existe 0 situation(s).

Situation comprenant        : X41
          ne comprenant pas : X51 X52 X53 X54
```

```
Il existe 0 situation(s).

Situation comprenant        : X11 X21
          ne comprenant pas :
Situation 4 : X11 X21 X31 X40 X50
Situation 6 : X11 X21 X32 X40 X50
Il existe 2 situation(s).

Evolutions depuis : 4 6
           vers   :
Or : 4 Ex : 6 Cd : (ET panne_pompe1 panne_pompe2 (FM chgt_pompes ) )
Or : 4 Ex : 2 Cd : (OU (ET (NON haut_debit ) (NON bas_debit ) panne_pompe2
(FD panne_pompe1 ) ) (ET engorgement panne_pompe2 (FD panne_pompe1 ) ) )
Or : 4 Ex : 7 Cd : (OU (ET (NON haut_debit ) (NON bas_debit ) panne_pompe1
(FD panne_pompe2 ) ) (ET engorgement panne_pompe1 (FD panne_pompe2 ) ) )
Or : 4 Ex : 10 Cd : (OU (ET bas_debit (NON engorgement ) panne_pompe2 (FD
panne_pompe1 ) ) (ET haut_debit (NON engorgement ) panne_pompe2 (FD
panne_pompe1 ) ) )
Or : 4 Ex : 13 Cd : (OU (ET bas_debit (NON engorgement ) panne_pompe1 (FD
panne_pompe2 ) ) (ET haut_debit (NON engorgement ) panne_pompe1 (FD
panne_pompe2 ) ) )
Or : 6 Ex : 4 Cd : (ET panne_pompe2 panne_pompe1 (FM chgt_pompes ) )
Or : 6 Ex : 5 Cd : (OU (ET (NON haut_debit ) (NON bas_debit ) panne_pompe2
(FD panne_pompe1 ) ) (ET engorgement panne_pompe2 (FD panne_pompe1 ) ) )
Or : 6 Ex : 9 Cd : (OU (ET (NON haut_debit ) (NON bas_debit ) panne_pompe1
(FD panne_pompe2 ) ) (ET engorgement panne_pompe1 (FD panne_pompe2 ) ) )
Or : 6 Ex : 12 Cd : (OU (ET bas_debit (NON engorgement ) panne_pompe2 (FD
panne_pompe1 ) ) (ET haut_debit (NON engorgement ) panne_pompe2 (FD
panne_pompe1 ) ) )
Or : 6 Ex : 14 Cd : (OU (ET bas_debit (NON engorgement ) panne_pompe1 (FD
panne_pompe2 ) ) (ET haut_debit (NON engorgement ) panne_pompe1 (FD
panne_pompe2 ) ) )


Condition globale :  (OU (ET panne_pompe1 panne_pompe2 (FM chgt_pompes ) )
(ET (NON haut_debit ) (NON bas_debit ) panne_pompe2 (FD panne_pompe1 ) )
(ET engorgement panne_pompe2 (FD panne_pompe1 ) ) (ET (NON haut_debit )
(NON bas_debit ) panne_pompe1 (FD panne_pompe2 ) ) (ET engorgement
panne_pompe1 (FD panne_pompe2 ) ) (ET bas_debit panne_pompe2 (FD
panne_pompe1 ) ) (ET haut_debit panne_pompe2 (FD panne_pompe1 ) ) (ET
bas_debit panne_pompe1 (FD panne_pompe2 ) ) (ET haut_debit panne_pompe1 (FD
panne_pompe2 ) ) )


Il existe 10 evolution(s).
```

7.3. Conclusions relatives au traitement d'exemples

En traitant ces exemples, nous nous sommes convaincus que la possibilité de valider les grafcets permettait une approche totalement différente de leur conception. Il devient en effet possible de *spécifier uniquement ce que le système doit faire* et de *contrôler ensuite que le système ne fait pas ce qu'il ne doit pas faire*. Comme il n'est plus nécessaire de rechercher a priori les configurations particulières des entrées qui sont problématiques puisqu'il suffit de laisser «AGGLAÉ» les identifier automatiquement, la spécification est grandement facilitée.

Il devient également possible d'envisager sereinement l'utilisation des possibilités offertes par la recherche de stabilité du modèle GRAFCET pour élaborer des grafcets de spécification. Nous avons utilisé cette caractéristique dans les deux exemples et elle nous est apparue très performante, pour aborder les problèmes de modes de marches par exemple. Dans le cas de la distribution d'eau, nous avons d'abord établi et analysé les grafcets relatifs aux modes de marche. Une fois ceux-ci validés, nous avons élaboré les grafcets décrivant le fonctionnement normal en s'assurant qu'ils sont toujours réceptifs à un changement de mode de marches. Le fonctionnement de l'ensemble est le suivant : il y a d'abord évolution des grafcets décrivant les modes de marches puis évolution des grafcets décrivant le fonctionnement. Grâce à la recherche de stabilité, la séquentialité n'est pas visible à l'extérieur du modèle.

Lors de nos expérimentations, nous avons également analysé des grafcets dont nous n'étions pas les auteurs. Grâce à cela, nous avons pu enrichir la base de propriétés qu'il est possible de vérifier pour valider un grafcet et améliorer la génération de l'automate. Nous tenons à remercier leurs auteurs pour cette aide précieuse et surtout pour avoir accepté que leur travail de modélisation soit soumis à notre analyse.

Conclusion générale

La traduction de grafcets en graphe d'états a souvent été décrite comme étant à la fois séduisante sur le plan théorique et inenvisageable sur le plan opérationnel en raison de l'explosion combinatoire qui en résulte. Nous venons de montrer dans ce mémoire - et plus concrètement encore par notre maquette informatique AGGLAÉ - que cette explosion combinatoire pouvait être maîtrisée par une modélisation judicieuse de l'évolution des entrées.

Cependant, il nous paraît important de souligner que nous ne proposons pas une technique de *traduction* d'un grafcet en un automate à états, mais plutôt une technique d'*extraction* de l'automate qu'un grafcet contient puisque nous reconstituons l'automate à partir des évolutions possibles du grafcet.

Lors de ces travaux, nous avons pu juger de l'important pouvoir d'expression du GRAFCET (réceptivités, parallélismes, forçages, ...) qui rendaient cette machine d'états «trop» performante pour être validée directement. En utilisant la représentation par automate à états pour la validation des modèles réalisés, le GRAFCET est encore plus performant et sans nul doute l'outil idéal pour la spécification de la dynamique des systèmes à événements discrets.

Nos travaux ont également permis de renforcer les bases théoriques du GRAFCET, notamment en établissant une définition formelle à la notion de fronts. Pour cela, nous avons défini une algèbre de Boole pour les fonctions du temps à valeurs booléennes, algèbre qui a été complétée par deux opérateurs unaires qui ont permis de formaliser les fronts. Grâce à cette algèbre, nous avons pu démontrer des

propriétés couramment admises comme $\downarrow\overline{u} = \uparrow u$ ou de nouvelles propriétés comme

$$\uparrow\left(\sum_{i=1}^{n} u_i\right) = \sum_{i=1}^{n}\left(\uparrow u_i \cdot \prod_{(j=1),\,(j\neq i)}^{n} \uparrow u_j + \left(\overline{u_j} \cdot \overline{\downarrow u_j}\right)\right).$$

Notre contribution à la validation de grafcets de spécification a donc atteint les objectifs énoncés initialement puisque nous avons pu expérimenter avec succès la faisabilité de la génération automatique du graphe des situations accessibles et la preuve de propriétés du grafcet analysé tout en renforçant les bases théoriques du modèle. Il n'en reste pas moins qu'il s'agit de travaux pour lesquels de nombreuses perspectives s'ouvrent devant nous.

La première va dans le sens de l'amélioration des possibilités de validation de grafcets de spécification en multipliant les capacités d'analyse de l'automate équivalent et en optimisant les techniques de codage des algorithmes de génération du graphe des situations accessibles afin de l'appliquer à des cas industriels de grande ampleur. Pour que la validation de grafcets soit utilisable par tous, il sera également nécessaire de toujours veiller à ce que l'expression des propriétés à vérifier puissent se faire avec le vocabulaire du GRAFCET. Pour améliorer cette praticabilité de la démarche proposée dans ce mémoire, il parait intéressant d'y associer les industries spécialisées dans la production d'automatismes surs comme le nucléaire ou l'avionique ; il s'agit tout probablement des utilisateurs potentiels les plus directement concernés par une telle approche.

Une deuxième perspective peut être dégagée au niveau de la formalisation du GRAFCET car l'algèbre de Boole que nous proposons pour les fonctions du temps à valeurs booléennes peut encore être étendue. Nous pensons que de nouveaux opérateurs peuvent être définis permettant ainsi d'introduire des décalages temporels. Nous comptons sur cette possibilité pour la modélisation des temporisations.

Tous ces travaux devront bien entendu, rester homogène aux concepts et définitions établis en 1977 car ce sont ces concepts et définitions qui ont fait du GRAFCET ce qu'il est actuellement, un modèle doté d'une grande capacité de modélisation qui a l'avantage d'être très ergonomique.

Bibliographie

Les références techniques ont toutes été regroupées à la fin de la bibliographie.

[Afcet 77]

Groupe AFCET Systèmes Logiques ; “Pour une représentation normalisée du cahier des charges d’un automatisme logique” ; RAII ; Vol. 61, pages 27 à 32 & Vol. 62, pages 36 à 40 ; Novembre & Décembre 1977

[Afcet 83]

Groupe AFCET Systèmes Logiques ; “Les interprétations algébriques et algorithmiques et les temporisations du GRAFCET” ; Document de synthèse édité par l’AFCET ; Juin 1983

[Afcet 87]

Groupe AFCET Systèmes Logiques ; “Formalisation d’extensions du GRAFCET : Macro étape et Forçage” ; Document de synthèse édité par l’AFCET ; Janvier 1987

[André 92]

C. ANDRÉ, M.A. PERALDI ; “Grafcet et langages synchrones” ; Actes du Congrès GRAFCET’92 - Editions AFCET - pages 91 à 100 ; Paris ; Mars 1992

[André 93]

C. ANDRÉ, M.A. PÉRALDI ; “Synchronous Programming : introduction and application to industrial process control” ; Invited Paper of 1993 IEEE International Conference on Computers in Design, Manufacturing and Production ; pages 461 à 470 ; Paris-Evry ; 24-27 Mai 1993

[André 94a]

C. ANDRÉ, D. GAFFÉ ; “Coopération Grafcet/Esterel” ; Actes du Colloque “Automatique, Génie informatique, Image 94” - pages 221 à 224 ; Poitiers ; Juin 1994

[André 94b]

C. ANDRÉ, D. GAFFÉ ; “Evénements et Conditions en Grafcet” ; APII - Editions HERMES ; Vol 28 N°4 - pages 331 à 352 ; 1994

[Arnold 90]

A. ARNOLD ; “Systèmes de transitions finis et sémantique des processus communicants” ; TSI - Editions DUNOD ; Vol. 9, N°3 - pages 193 à 216 ; 1990

[Arnold 92a]

A. ARNOLD ; “Systèmes de transitions finis et sémantique des processus communicants” ; Editions MASSON ; 196 pages ; 1992

[Arnold 92b]

A. ARNOLD J. BEAUQUIER, B. BÉRARD, B. ROZOY ; “Programmes parallèles : modèles et validation” ; Editions ARMAND COLIN ; 164 pages ; Janvier 1992

[Aygalinc 92]

P. AYGALINC, J.P. DENAT ; “Validation de modèles GRAFCET fonctionnels et évaluation de performances du système associé par l’utilisation des Réseaux de Petri “ ; Actes du Congrès GRAFCET’92 - Editions AFCET - pages 135 à 145 ; Paris ; Mars 1992

[Berge 73]

C. BERGE ; “Graphes et hypergraphes” ; Editions DUNOD UNIVERSITÉ, 516 pages ; 1973

[Blanchard 79]

M. BLANCHARD ; “Comprendre maîtriser et appliquer le GRAFCET” ; Editions CEPADUES, 174 pages ; 1979

[Bouteille 92]

N. BOUTEILLE, P. BRARD, G. COLOMBARI, N. COTAINA, D. RICHET ; “Le GRAFCET” ; Editions CEPADUES, 144 pages ; 1992

[Chapurlat 93]

V.CHAPURLAT, G. MONNERET, F. PRUNET ; “Discrete events system modelling and software engineering : ACSY” ; Proc. Comp Euro 93 ; IEE Computer Society Press Ed.” ; Paris-Evry ; Mai 1993

[Colombari 90]

G.COLOMBARI ; “Formalisation temporelle du Grafcet” ; Actes du Congrès Temps Réel - pages 143 à 154 ; Nantes ; Octobre 1990

[Corbier 87]

F. CORBIER ; “Simuler la partie opérative pour tester les automatismes de commandes” ; Actes des Conférences “AUTOMATION 87” ; page 33 à 46 ; Paris ; 24 Mars 1987

[Crubillé 89]

P. CRUBILLÉ ; “Réalisation de l’outil Mec : spécification fonctionnelle et architecture” ; Thèse de l’Université Bordeaux I ; 169 pages ; Novembre 1989

[Davenport 87]

J. Davenport, Y. Siret E. Tournier ; “Calcul formel : systèmes et algorithmes de manipulations algébriques” ; Editions Masson 264 pages ; 1987

[David 89]

R. DAVID, H. ALLA ; “Du Grafcet au réseaux de Petri” ; Editions HERMES, 424 pages ; 1989

[Deneux 83]

H. DENEUX, R. DAVID ; “Spécification et mise en œuvre d’automates interconnectés à l’aide du GRAFCET” ; RAIRO ; Vol. 17, N°4 - pages 339 à 358 ; 1983

[Denham 88]

M.J. DENHAM ; “A Petri-net Approach to the Control of Discrete-event Systems” ; Avanced Computing Concepts and Techniques in Control Engineering ; NAto ASI Series, Springer-Verlag Berlin Heidelberg ; Vol. F47 - pages 191 à 214 ; 1988

[Desbazeille 76]

G. DESBAZEILLE ; “Exercices et problèmes de recherche opérationnelle” ; Editions DUNOD ; 346 pages ; 1976

[Dutertre 93]

B. DUTERTRE ; “Spécification et preuve de systèmes dynamiques” ; Thèse de l’Université Rennes 1 ; 195 pages ; Mai 1993

[Frachet 93]

J.P. FRACHET, G. COLOMBARI ; “Elements for a semantics of the time in GRAFCET and dynamic systems using non-standard analys” ; APII - Editions HERMES ; Vol 27 N°1 - pages 107 à 125 ; 1993

[Gaudel 87]

M.C. GAUDEL, M. SORIA, C. FROIDEVAUX ; “Types de données et algorithmes : Recherche, Tri, Algorithmes sur les graphes” ; Edité par INRIA ; 240 pages ; 1987

[Giaccone 91]

T. GIACCONE ; “Modèle structuré de spécification, de conception et de mise au point de systèmes à événements discrets” ; Thèse de l’Université de Montpellier II ; Novembre 1991

[Girard 73]

P. GIRARD, P. Naslin ; “Construction des machines séquentielles industrielles” ; Editions DUNOD, 244 pages ; 1973

[Gochet 91]

P. GOCHET, P. GRIBOMONT ; “Logique : méthodes pour l’informatique fondamentale” ; Editions HERMES ; 456 pages ; 1991

[Grepa 85]

GREPA ; “Le GRAFCET de nouveaux concepts” ; Editions CEPADUES, 104 pages ; 1985

[Le Parc 94]

P. LE PARC ; “Apports de la méthodologie synchrone pour la définition et l’utilisation du langage GRAFCET” ; Thèse de l’Université Rennes 1 ; 172 pages ; Janvier 1994

[Lesage 93]

J.J. LESAGE, J.M. ROUSSEL ; “Hierarchical approach to GRAFCET using forcing order” ; APII - Editions HERMES ; Vol 27 N°1 - pages 25 à 38 ; 1993

[Lhoste 94]

P. LHOSTE ; “Contribution au génie automatique : concepts, modeles, méthodes et outils” ; Habilitation à diriger des recherches de l’université de Nancy I ; Février 94

[Marchand 89]

M. MARCHAND ; “Mathématique discrète” ; Editions DE BOECK UNIVERSITÉ ; 499 pages ; 1989

[Moalla 81]

M. MOALLA ; “Spécification et conception sûre d’automatismes discrets complexes, basées sur l’utilisation du Grafcet et des réseaux de Petri” ; Thèse d’état de l’Université scientifique et médicale de Grenoble ; Juillet 1981

[Muller 89]

J.L. MULLER ; “Modélisation Grafcet de la commande et simulation de la partie opérative des systèmes de production et de manutention” ; Thèse de l’Université de Montpellier II ; 1989

[Naslin 70]

P. NASLIN ; “Circuits logiques et automatismes à séquences” ; Editions DUNOD ; 522 pages ; 1970

[Panetto 91]

H. PANETTO ; “Une contribution au génie automatique : le prototypage des machines et systèmes automatisés de production” ; Thèse de l’Université de Nancy I ; Janvier 1991

[Permingeat 91]

N. PERMINGEAT, D. GLAUDE ; “Algèbre de Boole” ; 2^{e} tirage corrigé Editions MASSON, 211 pages ; 1991

[Prunet 87]

F. PRUNET, J.L. STRULESE, C. CAZALOT, E. GINESTET, D. PANAGET, G. DECHENAUX, P. LLORCA ; “Méthodologie et implantation automatique de commande d’automatisme à l’aide de la chaîne PIASTRE” ; APII - Editions DUNOD ; Vol 21 N°4 - pages 299 à 321 ; 1987

[Pruvost 85]

J.N. PRUVOST ; “Application des méthodes de fiabilité et de sécurité des systèmes aux automatismes” ; Actes des Conférences “AUTOMATION 85” ; 20 pages ; Paris ; 17 Avril 1985

[Roussel 93]

J.M. ROUSSEL, J.J. LESAGE ; “Une algèbre de Boole pour l’approche événementielle des systèmes logiques” ; APII - Editions HERMES ; Vol 27 N°5 - pages 541à 560 ; 1993

[Roux 92]

O. ROUX, D. CREUSOT, F. CASSEZ, J.P. ELLOY ; “Le langage réactif asynchrone Electre” ; TSI - Editions HERMES ; Vol 11 N°5 - pages 35 à 66 ; 1992

[Roux 94]

O. ROUX, V. RUSU ; “Du Grafcet au langage réactif Electre” ; APII - Editions HERMES ; Vol 28 N°2 - pages 131 à 157 ; 1994

[Sourisse 85]

C. SOURISSE ; “Incidence de la sécurité et de la disponibilité sur la conception des équipements pilotés par API” ; Actes des Conférences “AUTOMATION 85” ; 15 pages ; Paris ; 17 Avril 1985

[Toulotte 81]

J.M. TOULOTTE ; “Automates programmables” ; ITET N°227 ; N°227 - pages 5 à 14 ; 1981

[Vergamini 87]

D. VERGAMINI ; “Vérification de réseaux d’automates finis par équivalences observationnelles : le système AUTO ; Thèse de l’Université de Nice ; Décembre 1987

[Zahnd 87]

J. ZAHND ; “Machines séquentielles” ; Editions DUNOD, 265 pages ; 1987

Références techniques

[CEI 88]

Norme CEI 848 ; “Etablissement de diagrammes fonctionnels pour systèmes de commande” ; 1988

[Citroën 88]

Automobiles CITROEN direction des méthodes & moyens industriels ; “Cahier des charges général automatisme des moyens de production” ; Version 2.2 ; 1988

[Pagnol 93]

J.L. PAGNOL : “Grafcet + VME : couple gagnant” ; Compte rendu de la journée “Automatisation de processus indistriels” du “Club AUTOMATION” ; Paris ; 4 Février 1993

[UTE 82]

Union Technique de l’Électricité ; Norme NF C 03-190 “Diagramme fonctionnel GRAFCET pour la description des systèmes logiques de commande” ; Paris ; Juin 1982

[UTE 93]

Union Technique de l’Électricité ; Norme C 03-191 “Diagramme fonctionnel GRAFCET. Extension des concepts de base” ; Paris ; Juin 1993

[3IP 88]

Société 3IP ; “Documentation technique d’OMEGA” ; 1988

Annexe A
Illustration à l'aide de chronogrammes des 14 propriétés établies au chapitre 3

Dans le chapitre 3, nous avons établi 14 propriétés concernant le développement des différents opérateurs par rapport aux opérateurs fronts. Dans cette annexe, nous donnons une illustration de celles-ci à l'aide de chronogrammes. Les chronogrammes des fonctions u et v ont été choisis pour faire apparaître toutes les combinaisons possibles y compris les évolutions simultanées.

- $\downarrow\bar{u} = \uparrow u$,

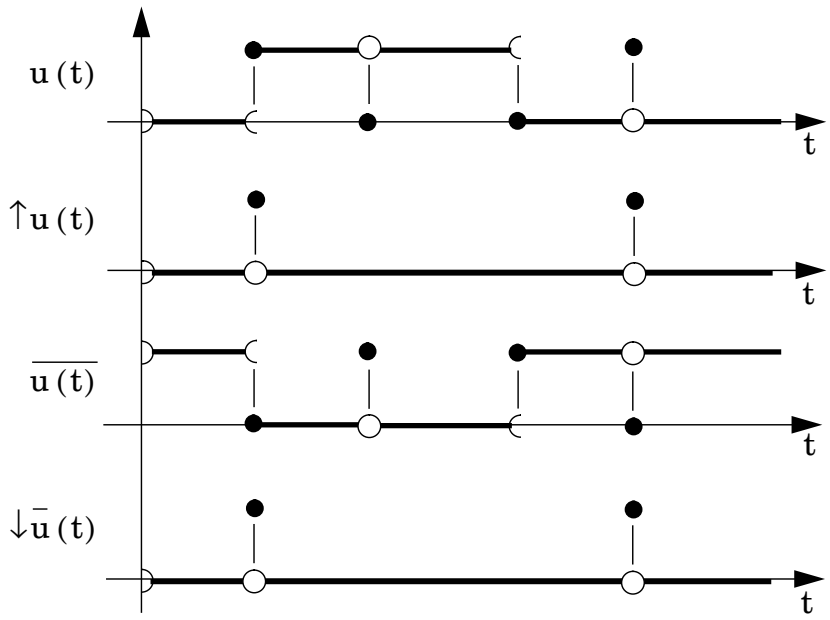


figure 58. Chronogrammes d'illustration des quatorze propriétés de développement des opérateurs fronts

- $\downarrow\bar{u} = \uparrow u$, $\uparrow\bar{u} = \downarrow u$, $u + \uparrow u = u$, $u \cdot \uparrow u = \uparrow u$, $\bar{u} + \downarrow u = \bar{u}$, $\bar{u} \cdot \downarrow u = \downarrow u$

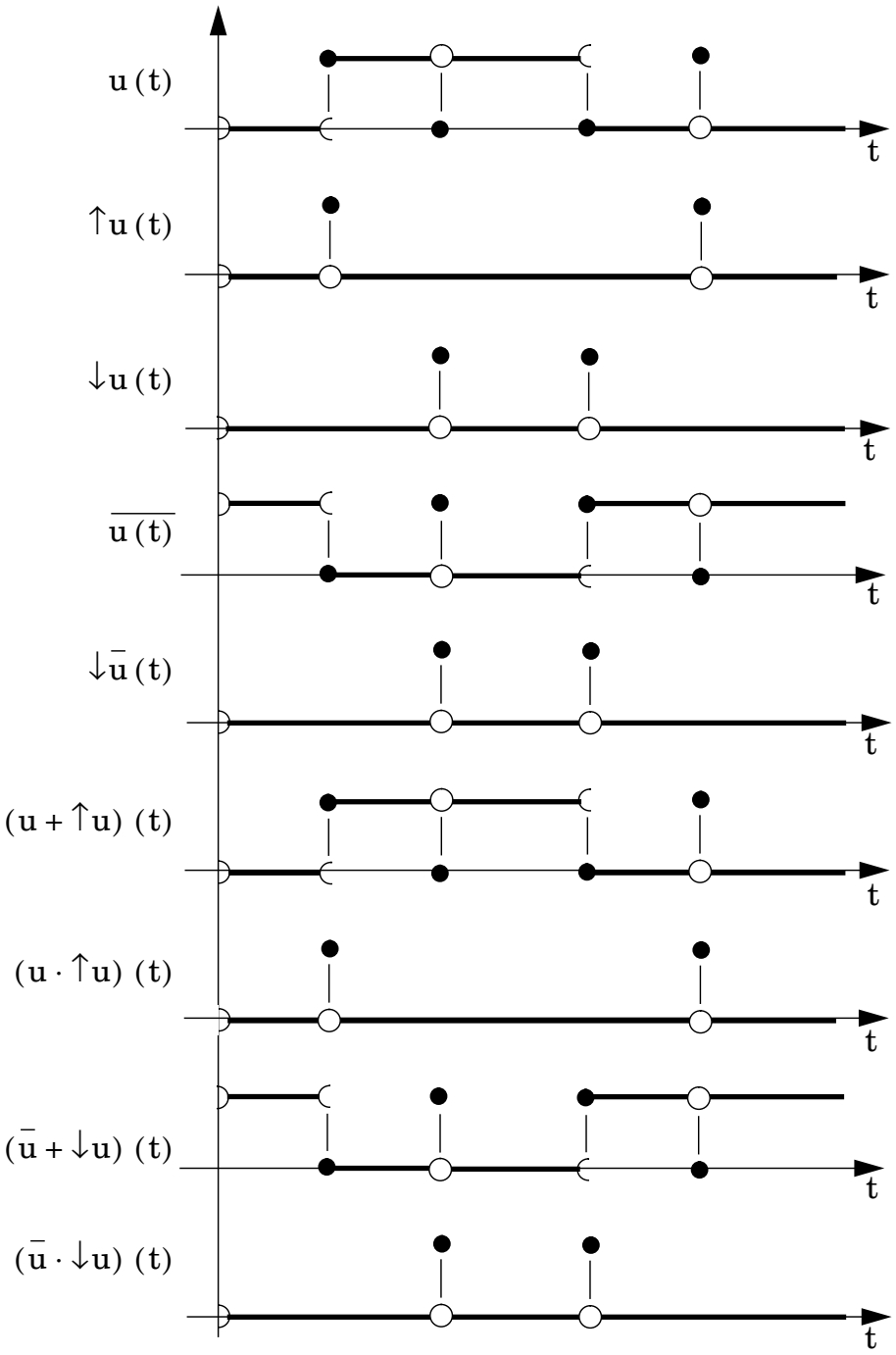


figure 59. Chronogrammes d'illustration des quatorze propriétés de développement des opérateurs fronts

- $\uparrow(\uparrow u) = \uparrow u$, $\uparrow(\downarrow u) = \downarrow u$, $\downarrow(\uparrow u) = 0$, $\downarrow(\downarrow u) = 0$

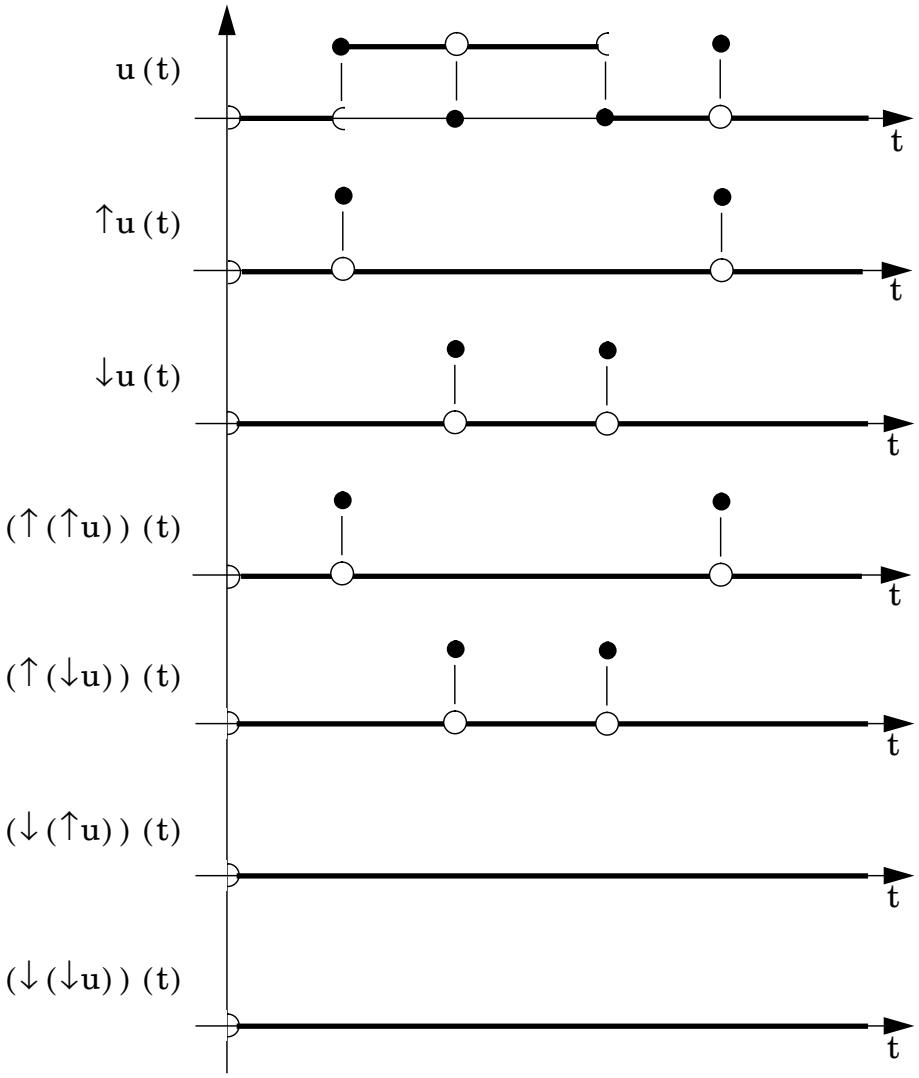


figure 60. Chronogrammes d'illustration des quatorze propriétés de développement des opérateurs fronts

- $\uparrow(u+v) = \uparrow u \cdot (\uparrow v + \bar{v} \cdot \overline{\downarrow v}) + \uparrow v \cdot (\uparrow u + \bar{u} \cdot \overline{\downarrow u})$

  $\downarrow(u+v) = \downarrow u \cdot \bar{v} + \downarrow v \cdot \bar{u}$

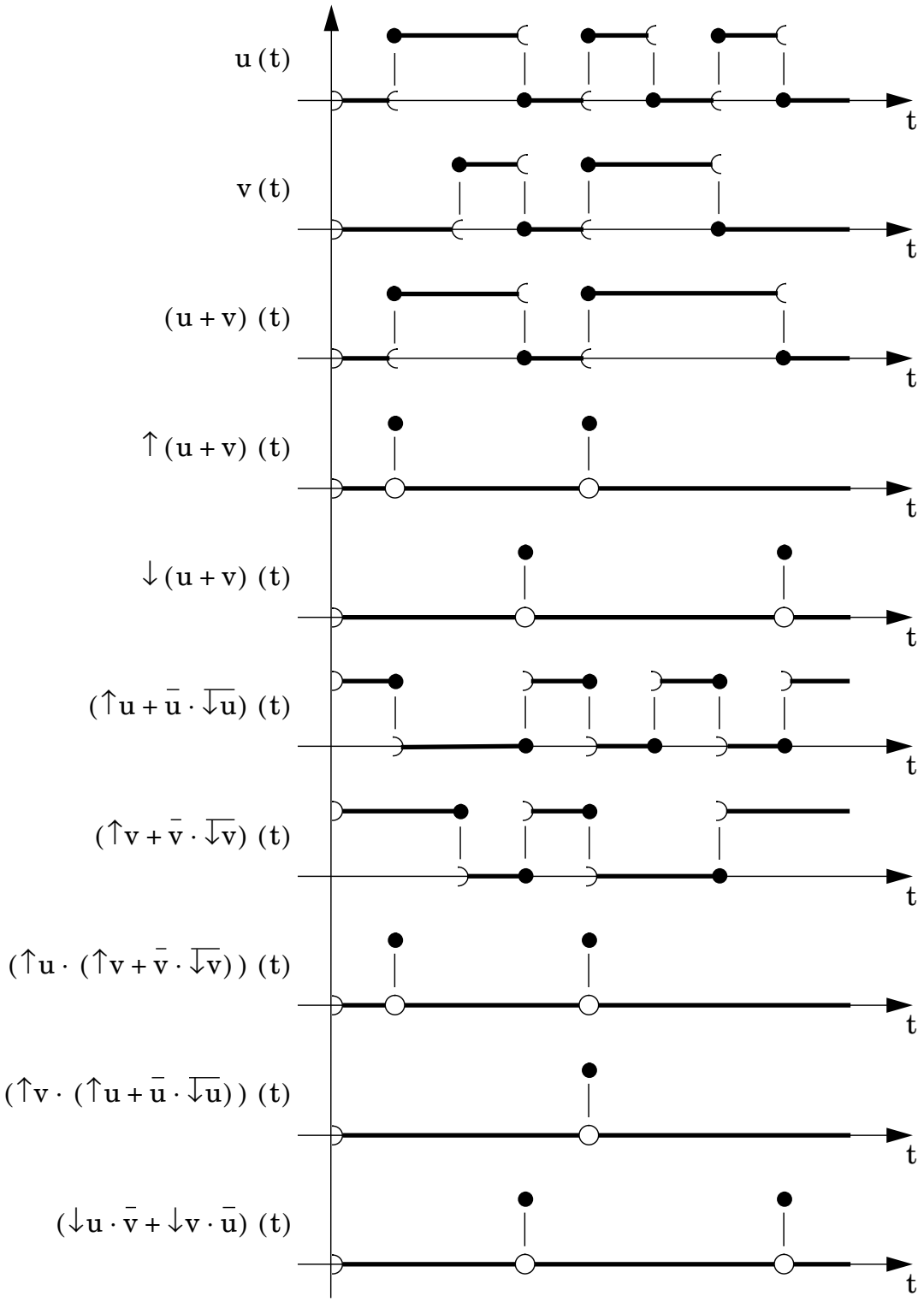


figure 61. Chronogrammes d'illustration des quatorze propriétés de développement des opérateurs fronts

- $\uparrow(u \cdot v) = (\uparrow u \cdot v) + (u \cdot \uparrow v)$

  $\downarrow(u \cdot v) = \downarrow u \cdot (\downarrow v + v \cdot \overline{\uparrow v}) + \downarrow v \cdot (\downarrow u + u \cdot \overline{\uparrow u})$

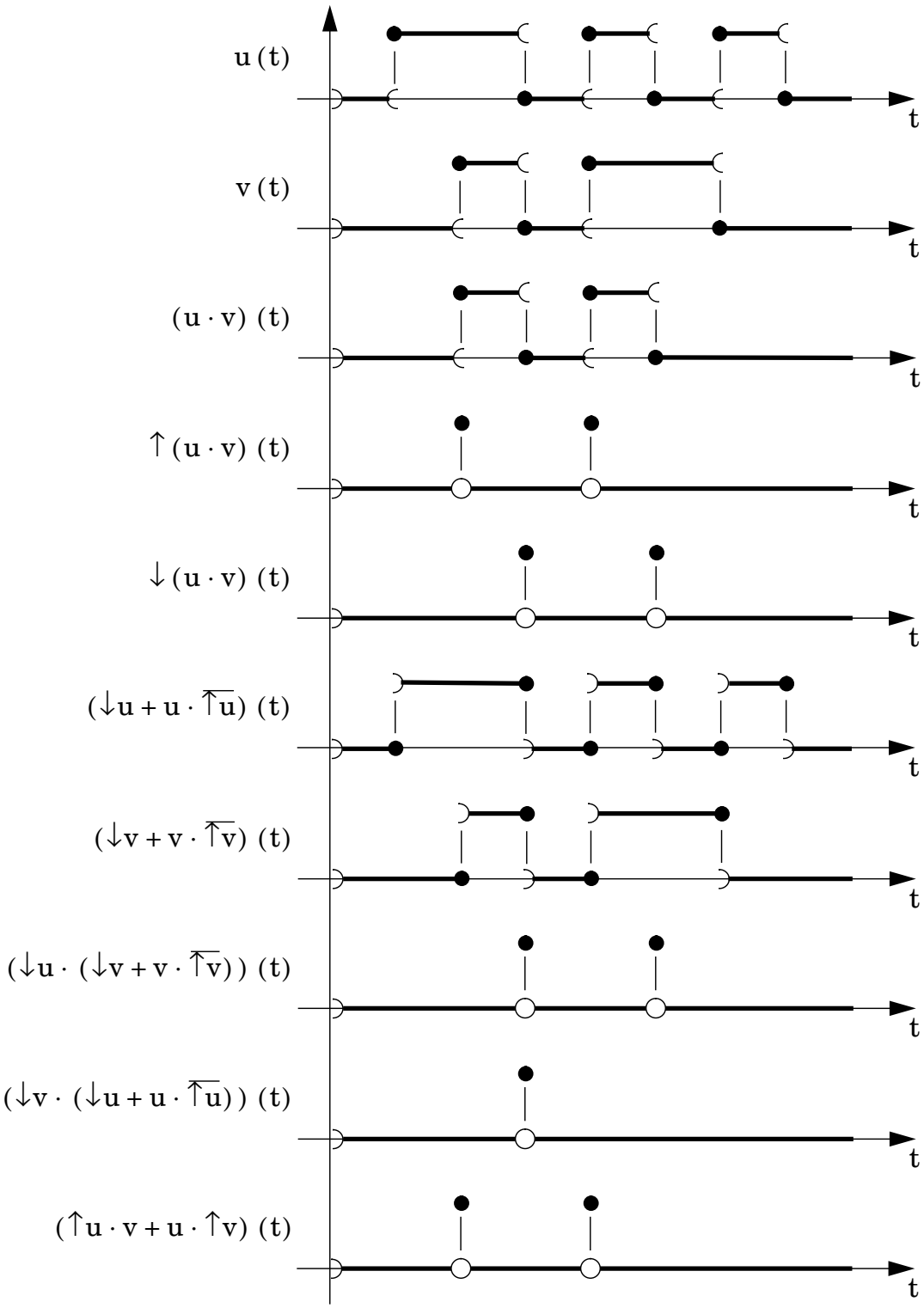


figure 62. Chronogrammes d'illustration des quatorze propriétés de développement des opérateurs fronts

Annexe B
Calcul symbolique sur $\mathbb{I}$
avec contention des entrées

Au chapitre 4, à l'issue de la présentation du module de calcul symbolique, nous avons proposé une extension pour envisager un traitement qui tiennent compte des cas de contention des entrées. Dans cette annexe, nous présentons les travaux nécessaires à cette extention.

Lorsque deux entrées (a et b) sont liés par une contention, les règles de simplifications employées classiquement ne sont pas toutes utilisables. Pour cela, il est nécessaire de reprendre les possibilités de simplification pour les trois expressions $f(a)\cdot g(b)$, $f(a)+g(b)$ et $f(a)\cdot P+g(b)$ lorsque f et g sont une de ces six structures suivantes : u, $\bar{u}$, $\uparrow u$, $\downarrow u$, $\overline{\uparrow u}$, $\overline{\downarrow u}$.

Le résultat sera, comme au chapitre 4, présenté sous forme de tableaux. Pour certaines combinaisons, nous avons jugé bon de joindre une démonstration.

B.1.Cas de la contention : $a = \bar{b}$

Ce cas de contention correspond au cas de deux entrées mutuellement exclusives et opposées l'une de l'autre.

Pour ce cas de contention, chaque possibilité de simplification pouvait exprimée soit en fonction de a, soit en fonction de b. En raison de l'utilisation faite de ce module,

nous avons, dans les trois tableaux qui suivent, privilégié la forme positive de chaque simplication.

Pour ce cas de contention, aucune démonstration est donnée. Pour établir chaque résultat, il suffit d'effectuer le changement de variable b en $\bar{a}$ dans les tableaux donnés dans le chapitre 4.

Dans le tableau 1, sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot g(b)$.

ET	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	0^*	a	0^*	$\uparrow a$	a	-
$\bar{a}$	b	0^*	$\uparrow b$	0^*	-	b
$\uparrow a$	0^*	$\uparrow a$	0^*	$\uparrow a$	$\uparrow a$	0^*
$\downarrow a$	$\uparrow b$	0^*	$\uparrow b$	0^*	0^*	$\uparrow b$
$\overline{\uparrow a}$	b	-	$\uparrow b$	0^*	-	$\overline{\uparrow a}$
$\overline{\downarrow a}$	-	a	0^*	$\uparrow a$	$\overline{\uparrow b}$	-

tableau 14 Possibilités de simplification pour $f(a) \cdot g(b)$ *lorsque les deux entrées a et b sont mutuellement exclusives.*

Dans le tableau 15 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) + g(b)$.

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	1^*	a	-	a	$\overline{\uparrow b}$	1^*
$\bar{a}$	b	1^*	b	-	1^*	$\overline{\downarrow b}$
$\uparrow a$	-	a	-	$\uparrow a$	$\overline{\uparrow b}$	1^*
$\downarrow a$	b	-	$\uparrow b$	-	1^*	$\overline{\downarrow b}$
$\overline{\uparrow a}$	$\overline{\uparrow a}$	1^*	$\overline{\uparrow a}$	1^*	1^*	$\overline{\uparrow a}$
$\overline{\downarrow a}$	1^*	$\overline{\downarrow a}$	1^*	$\overline{\downarrow a}$	$\overline{\uparrow b}$	1^*

tableau 15 Possibilités de simplification pour $f(a) + g(b)$ *lorsque les deux entrées a et b sont mutuellement exclusives.*

Dans le tableau 16 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot P + g(b)$

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$a \cdot P$	$b + P$	a	-	-	$\overline{\uparrow b}$	-
$\bar{a} \cdot P$	b	$a + P$	-	-	-	$\overline{\uparrow a}$
$\uparrow a \cdot P$	-	a	-	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\uparrow a} + P$
$\downarrow a \cdot P$	b	-	$\uparrow b$	-	$\overline{\uparrow b} + P$	$\overline{\uparrow a}$
$\overline{\uparrow a} \cdot P$	-	-	-	$\downarrow b + P$	$\overline{\uparrow b} + P$	$\overline{\uparrow a}$
$\overline{\downarrow a} \cdot P$	-	-	$\uparrow b + P$	-	$\overline{\uparrow b}$	$\overline{\uparrow a} + P$

tableau 16 Possibilités de simplification pour $f(a) \cdot P + g(b)$ *lorsque les deux entrées a et b sont mutuellement exclusives.*

B.2.Cas de la contention : $a \cdot b = 0^*$

Ce cas de contention correspond au cas de deux entrées qui ne sont jamais présentes en même temps mais qui peuvent être simultanément absentes. Nous supposons que les deux entrées ne peuvent pas changer d'état simultanément.

Pour ce cas de contention, nous proposons une démonstration lorsque celle-ci n'est pas immédiate.

Dans le tableau 17, sont regroupés les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot g(b)$ (Pour obtenir les 15 combinaisons situées sous la diagonale du tableau, nous utilisons les symétries de l'expression et de la contention).

ET	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	0^*	a	0^*	0^*	a	a
$\bar{a}$	b	-	$\uparrow b$	$\downarrow b$	-	-
$\uparrow a$	0^*	$\uparrow a$	0^*	0^*	$\uparrow a$	$\uparrow a$
$\downarrow a$	0^*	$\downarrow a$	0^*	0^*	$\downarrow a$	$\downarrow a$
$\overline{\uparrow a}$	b	-	$\uparrow b$	$\downarrow b$	-	-
$\overline{\downarrow a}$	b	-	$\uparrow b$	$\downarrow b$	-	-

tableau 17 Possibilités de simplification pour $f(a) \cdot g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \cdot b = 0^*$.

Démonstration de certaines propositions :

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

$$a \cdot \uparrow b = a \cdot \uparrow b \cdot b = 0^*$$

$$\begin{cases} \downarrow(a \cdot b) = 0^* \\ \downarrow(a \cdot b) = \downarrow a \cdot b + a \cdot \downarrow b \end{cases} \Rightarrow \begin{cases} \downarrow a \cdot b = 0^* \\ a \cdot \downarrow b = 0^* \end{cases}$$

$$a \cdot \overline{\uparrow b} = a \cdot \overline{\uparrow b} + a \cdot \uparrow b = a$$

$$a \cdot \overline{\downarrow b} = a \cdot \overline{\downarrow b} + a \cdot \downarrow b = a$$

$$\bar{a} \cdot \uparrow b = \bar{a} \cdot \uparrow b + a \cdot \uparrow b = \uparrow b$$

$$\bar{a} \cdot \downarrow b = \bar{a} \cdot \downarrow b + a \cdot \downarrow b = \downarrow b$$

C.Q.F.D.

Dans le tableau 18, sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) + g(b)$ (Pour obtenir les 15 combinaisons situées sous la diagonale du tableau, nous utilisons les symétries de l'expression et de la contention).

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\bar{a}$	$\bar{a}$	1^*	$\bar{a}$	$\bar{a}$	1^*	1^*
$\uparrow a$	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\downarrow a$	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\overline{\uparrow a}$	$\overline{\uparrow a}$	1^*	$\overline{\uparrow a}$	$\overline{\uparrow a}$	1^*	1^*
$\overline{\downarrow a}$	$\overline{\downarrow a}$	1^*	$\overline{\downarrow a}$	$\overline{\downarrow a}$	1^*	1^*

tableau 18 Possibilités de simplification pour $f(a) + g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \cdot b = 0^*$.

Démonstration de certaines propositions :

$$a + \bar{b} = a \cdot (\bar{b} + b) + \bar{b} = a \cdot \bar{b} + \bar{b} = \bar{b}$$
$$a + \overline{\uparrow b} = a \cdot (\overline{\uparrow b} + \uparrow b) + \overline{\uparrow b} = a \cdot \overline{\uparrow b} + \overline{\uparrow b} = \overline{\uparrow b}$$
$$a + \overline{\downarrow b} = a \cdot (\overline{\downarrow b} + \downarrow b) + \overline{\downarrow b} = a \cdot \overline{\downarrow b} + \overline{\downarrow b} = \overline{\downarrow b}$$
$$\bar{a} + \bar{b} = \overline{(a \cdot b)} = 1^{*}$$
$$\bar{a} + \uparrow b = \bar{a} + \uparrow b \cdot (a + \bar{a}) = \bar{a} + \uparrow b \cdot \bar{a} = \bar{a}$$
$$\bar{a} + \downarrow b = \bar{a} + \downarrow b \cdot (a + \bar{a}) = \bar{a} + \downarrow b \cdot \bar{a} = \bar{a}$$
$$\bar{a} + \overline{\uparrow b} = \bar{a} + \uparrow b + \overline{\uparrow b} = 1^{*}$$
$$\bar{a} + \overline{\downarrow b} = \bar{a} + \downarrow b + \overline{\downarrow b} = 1^{*}$$

C.Q.F.D.

Dans le tableau 19 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot P + g(b)$.

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$a \cdot P$	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\bar{a} \cdot P$	-	$\bar{b} + P$	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$
$\uparrow a \cdot P$	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\downarrow a \cdot P$	-	$\bar{b}$	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\overline{\uparrow a} \cdot P$	-	$\bar{b} + P$	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$
$\overline{\downarrow a} \cdot P$	-	$\bar{b} + P$	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$

tableau 19 Possibilités de simplification pour $f(a) \cdot P + g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \cdot b = 0^{*}$.

Démonstration de certaines propositions :

$$\bar{b} + a \cdot P = \bar{b} + a \cdot \bar{b} \cdot P = \bar{b}$$
$$\bar{b} + \bar{a} \cdot P = \bar{b} + \bar{a} \cdot (\bar{b} + b) \cdot P = \bar{b} + \bar{a} \cdot b \cdot P = \bar{b} + b \cdot P = \bar{b} + P$$
$$\bar{b} + \uparrow a \cdot P = \bar{b} + \uparrow a \cdot \bar{b} \cdot P = \bar{b}$$
$$\bar{b} + \downarrow a \cdot P = \bar{b} + \downarrow a \cdot \bar{b} \cdot P = \bar{b}$$
$$\bar{b} + \overline{\uparrow a} \cdot P = \bar{b} + \overline{\uparrow a} \cdot (\bar{b} + b) \cdot P = \bar{b} + \overline{\uparrow a} \cdot b \cdot P = \bar{b} + b \cdot P = \bar{b} + P$$
$$\bar{b} + \overline{\downarrow a} \cdot P = \bar{b} + \overline{\downarrow a} \cdot (\bar{b} + b) \cdot P = \bar{b} + \overline{\downarrow a} \cdot b \cdot P = \bar{b} + b \cdot P = \bar{b} + P$$

C.Q.F.D.

B.3.Cas de la contention : $a \rightarrow b$

Nous supposons que les deux entrées ne peuvent pas changer d'état simultanément. De plus, l'entrée b peut être présente tandis que l'entrée a ne l'est pas.

Ce cas de contention peut également s'exprimer par l'égalité $a \cdot \bar{b} = 0^*$. En raison de la disymétrie de la contention, il est nécessaire d'envisager la simplification des expressions $f(a) \cdot g(b)$, $f(a) + g(b)$ et $f(a) \cdot P + g(b)$ mais également de l'expression $f(a) + g(b) \cdot P$.

Dans le tableau 20 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot g(b)$.

ET	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	a	0^*	0^*	0^*	a	a
$\bar{a}$	-	$\bar{b}$	$\uparrow b$	$\downarrow b$	-	-
$\uparrow a$	$\uparrow a$	0^*	0^*	0^*	$\uparrow a$	$\uparrow a$
$\downarrow a$	$\downarrow a$	0^*	0^*	0^*	$\downarrow a$	$\downarrow a$
$\overline{\uparrow a}$	-	$\bar{b}$	$\uparrow b$	$\downarrow b$	-	-
$\overline{\downarrow a}$	-	$\bar{b}$	$\uparrow b$	$\downarrow b$	-	-

tableau 20 Possibilités de simplification pour $f(a) \cdot g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \rightarrow b$.

Démonstration de certaines propositions :

$$a \cdot b = a \cdot b + a \cdot \bar{b} = a$$
$$\bar{a} \cdot \bar{b} = \bar{a} \cdot \bar{b} + a \cdot \bar{b} = \bar{b}$$
$$\begin{cases} \downarrow(a \cdot \bar{b}) = 0^{*} \\ \downarrow(a \cdot \bar{b}) = \downarrow a \cdot \bar{b} + a \cdot \uparrow b \end{cases} \Rightarrow \begin{cases} a \cdot \uparrow b = 0^{*} \\ \downarrow a \cdot \bar{b} = 0^{*} \end{cases}$$
$$a \cdot \downarrow b = a \cdot \downarrow b \cdot \bar{b} = 0^{*}$$
$$a \cdot \overline{\uparrow b} = a \cdot \overline{\uparrow b} + a \cdot \uparrow b = a$$
$$\bar{a} \cdot \uparrow b = \bar{a} \cdot \uparrow b + a \cdot \uparrow b = \uparrow b$$
$$\uparrow a \cdot b = \uparrow a \cdot a \cdot b = \uparrow a \cdot a = \uparrow a$$
$$\uparrow a \cdot \bar{b} = \uparrow a \cdot a \cdot \bar{b} = 0^{*}$$
$$\downarrow a \cdot b = \downarrow a \cdot b + \downarrow a \cdot \bar{b} = \downarrow a$$
$$\overline{\uparrow a} \cdot \bar{b} = \overline{\uparrow a} \cdot \bar{b} + \uparrow a \cdot \bar{b} = \bar{b}$$

C.Q.F.D.

Dans le tableau 18 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) + g(b)$.

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
a	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\bar{a}$	1^{*}	$\bar{a}$	$\bar{a}$	$\bar{a}$	1^{*}	1^{*}
$\uparrow a$	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\downarrow a$	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\overline{\uparrow a}$	1^{*}	$\overline{\uparrow a}$	$\overline{\uparrow a}$	$\overline{\uparrow a}$	1^{*}	1^{*}
$\overline{\downarrow a}$	1^{*}	$\overline{\downarrow a}$	$\overline{\downarrow a}$	$\overline{\downarrow a}$	1^{*}	1^{*}

tableau 21 Possibilités de simplification pour $f(a) + g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \rightarrow b$.

Démonstration de certaines propositions :

$$a + b = a \cdot b + b = b$$
$$\bar{a} + b = \overline{(a \cdot \bar{b})} = \overline{0}^{*} = 1^{*}$$
$$\bar{a} + \bar{b} = \bar{a} + \bar{b} \cdot (\bar{a} + a) = \bar{a}$$
$$\bar{a} + \uparrow b = \bar{a} + \uparrow b \cdot \bar{a} = \bar{a}$$
$$\bar{a} + \overline{\uparrow b} = (\bar{a} + \uparrow b) + \overline{\uparrow b} = 1^{*}$$
$$\uparrow a + b = \uparrow a \cdot b + b = b$$
$$\overline{\uparrow a} + b = \overline{\uparrow a} + b + \uparrow a \cdot \bar{b} = 1^{*}$$
$$\overline{\uparrow a} + \bar{b} = \overline{\uparrow a} + \bar{b} \cdot \overline{\uparrow a} = \overline{\uparrow a}$$

C.Q.F.D.

Dans le tableau 22 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) \cdot P + g(b)$.

OU	b	$\bar{b}$	$\uparrow b$	$\downarrow b$	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$a \cdot P$	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\bar{a} \cdot P$	$b + P$	-	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$
$\uparrow a \cdot P$	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\downarrow a \cdot P$	b	-	-	-	$\overline{\uparrow b}$	$\overline{\downarrow b}$
$\overline{\uparrow a} \cdot P$	$b + P$	-	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$
$\overline{\downarrow a} \cdot P$	$b + P$	-	-	-	$\overline{\uparrow b} + P$	$\overline{\downarrow b} + P$

tableau 22 Possibilités de simplification pour $f(a) \cdot P + g(b)$ *lorsque les deux entrées a et b respectent la contention* $a \rightarrow b$.

Démonstration de certaines propositions :

$$b + a \cdot P = b + a \cdot b \cdot P = b$$
$$\overline{\uparrow b} + \bar{a} \cdot P = \overline{\uparrow b} + \bar{a} \cdot (\overline{\uparrow b} + \uparrow b) \cdot P = \overline{\uparrow b} + \bar{a} \cdot \uparrow b \cdot P = \overline{\uparrow b} + \uparrow b \cdot P = \overline{\uparrow b} + P$$
$$b + \uparrow a \cdot P = b + \uparrow a \cdot b \cdot P = b$$
$$b + \downarrow a \cdot P = b + \downarrow a \cdot b \cdot P = b$$
$$b + \overline{\uparrow a} \cdot P = b + \uparrow a + \overline{\uparrow a} \cdot P = b + \uparrow a + P = b + P$$
$$b + \overline{\downarrow a} \cdot P = b + \downarrow a + \overline{\downarrow a} \cdot P = b + \downarrow a + P = b + P$$

C.Q.F.D.

Dans le tableau 23 sont regroupées les 36 combinaisons possibles pour une expression de la forme $f(a) + g(b) \cdot P$.

OU	$b \cdot P$	$\bar{b} \cdot P$	$\uparrow b \cdot P$	$\downarrow b \cdot P$	$\overline{\uparrow b} \cdot P$	$\overline{\downarrow b} \cdot P$
a	-	-	-	-	-	-
$\bar{a}$	$\bar{a} + P$	-	$\bar{a}$	$\bar{a}$	$\bar{a} + P$	$\bar{a} + P$
$\uparrow a$	-	-	-	-	-	-
$\downarrow a$	-	-	-	-	-	-
$\overline{\uparrow a}$	$\overline{\uparrow a} + P$	$\overline{\uparrow a}$	$\overline{\uparrow a}$	$\overline{\uparrow a}$	$\overline{\uparrow a} + P$	$\overline{\uparrow a} + P$
$\overline{\downarrow a}$	$\overline{\downarrow a} + P$	$\overline{\downarrow a}$	$\overline{\downarrow a}$	$\overline{\downarrow a}$	$\overline{\downarrow a} + P$	$\overline{\downarrow a} + P$

tableau 23 Possibilités de simplification pour $f(a) + g(b) \cdot P$ *lorsque les deux entrées a et b respectent la contention* $a \rightarrow b$.

Démonstration de certaines propositions :

$$\bar{a} + b \cdot P = \bar{a} + \bar{b} + b \cdot P = \bar{a} + \bar{b} + P = \bar{a} + P$$

$$\bar{a} + \uparrow b \cdot P = \bar{a} + \uparrow b \cdot \bar{a} \cdot P = \bar{a}$$

$$\bar{a} + \overline{\uparrow b} \cdot P = \bar{a} + \uparrow b + \overline{\uparrow b} \cdot P = \bar{a} + \uparrow b + P = \bar{a} + P$$

$$\overline{\uparrow a} + b \cdot P = \overline{\uparrow a} + \bar{b} + b \cdot P = \overline{\uparrow a} + \bar{b} + P = \overline{\uparrow a} + P$$

$$\overline{\uparrow a} + \bar{b} \cdot P = \overline{\uparrow a} + \bar{b} + \bar{b} \cdot P = \overline{\uparrow a} + \bar{b} = \overline{\uparrow a}$$

C.Q.F.D.

Résumé :

En Génie Automatique, le GRAFCET [IEC 848] est couramment employé pour la modélisation de la dynamique des systèmes à événements discrets, en raison de ses capacités de modélisation et de son ergonomie. Cependant, il lui est reproché de ne pas être défini de manière suffisamment formelle pour que tous les grafcets établis soient sans ambiguïté et puissent être validés.

L'objectif des travaux est double : contribuer à la formalisation du GRAFCET de manière à renforcer ses fondements théoriques et offrir à tout analyste les moyens nécessaires pour valider une modélisation exprimée en GRAFCET en vérifiant les propriétés des modèles et leur comportement par rapport à leurs entrées/sorties.

Le GRAFCET étant une machine d'état complexe - essentiellement à cause des parallélismes importants qu'il permet de décrire - nous proposons à l'analyste d'utiliser le graphe des situations accessibles, ou grafcet d'état équivalent pour valider sa spécification.

Nous avons conçu une technique de génération automatique du graphe des situations accessibles d'un grafcet global (qui est un automate fini «équivalent»), de manière à pouvoir établir un ensemble de preuves et propriétés sur la cohérence intrinsèque du grafcet et sur sa pertinence par rapport au cahier des charges. Une algèbre de Boole, dans laquelle la notion de fronts a été formalisée par deux opérateurs unaires a été construite. Les 14 propriétés qui ont été démontrées ont permis d'établir un module de calcul symbolique utilisé pour tenir compte de l'historique des évolutions des entrées. Nos travaux intègrent les extensions du modèles GRAFCET.

Pour valider notre approche, une maquette informatique en C a été développée et permet de calculer l'automate équivalent au grafcet à valider. Nous utilisons pour vérifier certaines propriétés l'outil MEC développé pour l'étude des systèmes de transitions. Deux exemples de validation de grafcets par analyse de leur automate sont donnés dans le mémoire.

Mots clés :

GRAFCET, modélisation, validation, preuve, événements, systèmes logiques, algèbre de Boole, systèmes à événements discrets

Abstract :

In Automation Engineering, GRAFCET is currently used for modelling sequential system control, for its modeling and ergonomic capacities. It has nevertheless been reproached with not being defined formally enough for all established grafcets to be unequivocal and validated.

The aim of this work is twofold: to contribute to the formalization of GRAFCET so as to strengthen its theoretical foundations and to provide all analysts with the necessary means to validate GRAFCET models by proving their properties and their behavior in relation with inputs/outputs.

As GRAFCET is a complex state machine - essentially because of the important parallelisms the description of which it allows - the analyst can use the graph of accessible situations to validate his specification.

We have conceived a technique to automatically generate the graph of accessible situations of a global grafcet (i.e the equivalent finite state machine) so as to establish a set of proofs and properties about the inner coherence of the grafcet and its relevance in relation to requirements. A Boolean algebra in which the notion of events has been formalized by two unary operations has been built. The 14 properties that have been demonstrated have allowed to establish a module of formal calculus used to review the evolution of inputs.Our works include extensions of GRAFCET models.

To validate our approach, a C software has been developed which is used to calculate the finite state machine equivalent to the grafcet to be validated. To check certain properties, we use the MEC software which has been developed to study transition systems. Two examples of validations of gafcets by analysis of their automate are given in the paper.

Key words :

Sequential function chart, GRAFCET, validation, proof, events, logical systems, Boolean algebra, discret events systems