Modules

5.1 Introduction

La prédiction de la dynamique d'un réseau d'automates est un problème difficile, de par le fait que cette dynamique est exponentielle en la taille du réseau. Une des approches pour contourner ce problème est de diviser le réseau étudié en plusieurs sous-réseaux, dans l'espoir que la prédiction des éléments séparés permettent la prédiction de l'ensemble.

Cette approche modulaire n'est pas nouvelle et a été explorée de plusieurs façons dans la littérature. Certains travaux (MILO, SHEN-ORR, ITZKOVITZ et al. 2002 ; U. ALON 2003) ont rassemblés des réseaux d'influences tirés de la biologie, de la biochimie, de l'écologie et de l'ingénierie et étudient la réccurence de motifs, c'est-à-dire des sous-graphes spécifiques dont la présence est anormalement haute dans ces réseaux naturels comparés à des réseaux générés aléatoirement. D'autres travaux (BERNOT et TAHI 2009 ; SIEBERT 2009 ; DELAPLACE, KLAUDEL, MELLITI et al. 2012) étudient des formalismes qui permettent la définition de sous-réseaux dans un réseau d'automate, avec l'objectif de comprendre la dynamique d'un réseau plus large à partir de l'étude de parties de ce réseau. Ces travaux développent également des méthodes qui permettent cette division en parties d'une façon efficace et pertinente. Un travail récent (ALKHUDHAYR et STEGGLES 2019) introduit une notion de composition de réseau d'automates booléens qui repose sur la fusion de deux automates. La fonction locale de l'automate obtenu par cette fusion est définie comme la composition par un opérateur booléen des fonctions locales des automates fusionnés.

Notre approche peut être rapprochée de travaux dans le cadre non discret des réseaux de réactions (BAEZ et POLLARD 2017), où des entrées sont rajoutées à ces systèmes qui modélisent des réactions chimiques par l'emploi de réseaux de Petri ouverts.

Dans le cadre des réseaux d'automates, notre formalisme diffère des précédents par le fait qu'il ne vise pas à décrire la nature d'un sous-réseau, mais bien la possibilité de tout réseau d'être influencé par un réseau extérieur par le biais d'entrées que nous rajoutons au modèle. Ainsi, la propriété de faire partie d'un réseau plus large peut être comprise comme la réduction de l'évolution de ces entrées. Nous appellons ces réseaux disposant d'entrées les *modules*. Ce chapitre présente les définitions associées aux modules en section 5.2, et présente un ensemble de résultats les concernant en section 5.3.

5.2 Définitions

5.2.1 Entrées, fonctions locales, modules et graphe d'interaction

Les modules sont une généralisation des réseaux d'automates. En ces termes, la plupart des définitions développées pour les réseaux d'automates s'appliquent également aux modules. Ainsi, si un réseau d'automates est défini avec un ensemble d'automates S ainsi qu'un ensemble d'états Λ, un module a également un ensemble d'*entrées*, que nous appellons par convention I . Par convention de nouveau, les entrées contenues dans cet ensemble sont désignées par des lettres grecques, en commencant par α.

Les entrées d'un module représentent des influences extérieures, qui possèdent une valeur dans Λ à tout instant, et qui peuvent influencer les fonctions locales du module. Pour pouvoir définir cette influence, il est nécessaire de définir un état pour ces entrées. Cela est fait par le biais d'une configuration d'entrée.

Définition 43 (Configuration d'entrée)**.** *Soit I un ensemble d'entrées et Λ un ensemble d'états. Une* configuration d'entrée *est un vecteur défini sur Λ^I.*

Par convention, nous noterons les configurations d'entrée par la lettre i.

Concaténer une configuration d'entrée i avec une configuration standard x nous donne un vecteur $x \cdot i$ défini sur $\Lambda^{S \cup I}$. Ce vecteur contient toute l'information nécessaire pour mettre à jour les fonctions locales du module. Ainsi, nous pouvons désormais développer la définition des fonctions locales des modules.

Définition 44 (Fonction locale d'un module)**.** *Soit S un ensemble d'automates, I un ensemble d'entrées et Λ un ensemble d'états. Une* fonction locale *d'un module est une fonction f définie sur $\Lambda^{S \cup I} \to \Lambda$.*

Nous pouvons à présent définir un module par une définition similaire à celle d'un réseau d'automates.

Définition 45 (Module)**.** *Soit S un ensemble d'automates, I un ensemble d'entrées et Λ un ensemble d'états. Un* module *associe à tout $s \in S$ une fonction locale $f_s : \Lambda^{S \cup I} \to \Lambda$.*

En ces termes, les entrées d'un module se comportent comme des automates dans le sens où leur valeur est définie dans une configuration et utilisée pour influencer les automates du réseau. Cependant, les entrées diffèrent des automates dans le fait qu'elles ne possèdent pas elles-mêmes de fonctions locales. Les entrées sont conçues comme une influence extérieure au réseau, et leur valeur est arbitrairement définie à tout instant.

Exemple 38. *Soit $S = \{a, b, c\}$, $I = \{\alpha\}$ et $\Lambda = \{0, 1\}$. Soit M le module qui à a, b et c associe les fonctions suivantes : $f_a(x \cdot i) = \neg x_a \wedge i_\alpha$, $f_b(x \cdot i) = x_a \vee x_c$, et $f_c(x \cdot i) = (\neg x_a \wedge \neg x_c) \vee i_\alpha$.*

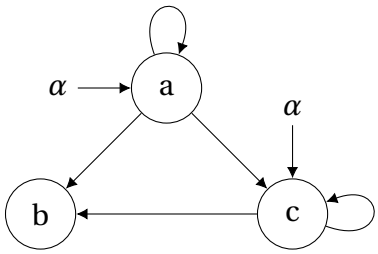

FIGURE 5.1 – Graphe d'interaction du module développé dans l'exemple 38.

L'influence qu'une entrée porte sur un automate se définit de façon similaire à l'influence entre deux automates.

Définition 46 (Influence d'une entrée)**.** *Soit M un module, α une entrée de I et s un automate de M. On dit que α* influence *s si et seulement s'il existe une configuration x et une paire de configurations d'entrée i et i' telles que $i_\beta = i'_\beta \iff \beta \neq \alpha$, et $f_s(x \cdot i) \neq f_s(x \cdot i')$.*

Tout comme les réseaux d'automates, les modules sont propices à la représentation sous la forme d'un graphe. Le *graphe d'interaction* d'un module se définit de façon pratiquement équivalente au graphe d'interaction d'un réseau d'automates, à l'ajout prêt de flèches qui représentent l'influence des entrées du module sur les automates.

Définition 47 (Graphe d'interaction de module)**.** *Soit M un module. On définit le* graphe d'interaction *de M comme le graphe orienté qui est défini sur les sommets de S, qui possède une arête de s vers r si et seulement si s influence r. De plus, chaque sommet s est décoré d'une flèche entrante annotée α pour chaque entrée α qui influence s.*

L'exemple 38 est représenté sous la forme d'un graphe d'interaction dans la figure 5.1.

5.2.2 Mises à jour, exécutions et graphe de la dynamique

Pour x une configuration et i une configuration d'entrée, un module M peut être mis à jour de la même façon qu'un réseau d'automates ; par le biais d'une sélection de fonctions locales de M qui s'appliquent simultanément pour altérer la configuration x. Cette sélection est représentée par une mise à jour δ, dont la définition ne change pas ; il s'agit toujours d'un sous-ensemble de S. La définition de l'application d'une telle mise à jour sur un module est intuitive.

Définition 48 (Application de mise à jour sur un module)**.** *Soit M un réseau d'automates, x une configuration, i une configuration d'entrée et δ une mise à jour. L'*application de la mise à jour *δ sur les configurations x et i dans M est une nouvelle configuration notée $M_\delta(x \cdot i)$, qui est définie par :*

$$\forall s \in S, M_\delta(x \cdot i)_s = \begin{cases} f_s(x \cdot i) & si\ s \in \delta \\ x_s & sinon \end{cases} .$$

Exemple 39. *Considérons le module M développé dans l'exemple 38, ainsi que trois mises à jour $\delta_1 = \{a\}$, $\delta_2 = \{b, c\}$ et S. Nous observons les faits suivants :*

$$\begin{aligned} M_{\delta_1}(000 \cdot 0) &= 000 \\ M_{\delta_2}(000 \cdot 0) &= 001 \\ M_S(000 \cdot 0) &= 001 \\ M_{\delta_1}(101 \cdot 1) &= 001 \\ M_{\delta_2}(101 \cdot 1) &= 011 \\ M_S(101 \cdot 1) &= 011 \end{aligned}$$

On note que l'application d'une mise à jour sur un module ne génère pas une nouvelle configuration pour les entrées de ce module. Les configurations d'entrée sont considérées indépendantes du fonctionnement interne du module. Cela signifie que sur une séquence de mises à jour, l'affectation d'états aux entrées du module pourra évoluer de façon complètement arbitraire. Pour refléter cette nature arbitraire, nous définissons la notion de séquence d'entrée.

Définition 49 (Séquence d'entrée)**.** *Soit I un ensemble d'entrée. Une* séquence d'entrée *est une séquence $J = (i_1, i_2, \ldots)$ finie ou infinie, où chaque i_k est une configuration d'entrée définie sur I.*

Ainsi, tout ce qui est nécessaire pour mettre à jour un module sur plus d'une mise à jour à la suite, est une configuration initiale x, un mode de mise à jour Δ, et une séquence d'entrée J au moins aussi longue que Δ.

Définition 50 (Exécution de module)**.** *Soit M un module, x une configuration, $\Delta = (\delta_1, \delta_2, \ldots)$ un mode de mise à jour fini et $J = (i_1, i_2, \ldots)$ une séquence d'entrée au moins aussi longue que Δ. On appelle* exécution *de M selon Δ sur x et J la configuration définie par les équations récursives suivantes :*

$$\begin{aligned} M_{(\delta_1, \delta_2, \ldots)}(x, (i_1, i_2, \ldots)) &= M_{(\delta_2, \ldots)}(M_{\delta_1}(x \cdot i_1), (i_2, \ldots)) \\ M_{(\,)}(x, J) &= x \end{aligned}$$

Exemple 40. *Nous considérons M le module développé dans l'exemple 38. Soit $\Delta_1 = (\{a\}, \{b\}, \{c\})$, $\Delta_2 = (\{b, c\}, \{a\})$, et $J = (1, 0, 1)$. Nous observons les faits suivant :*

$$M_{\Delta_P \cdot \Delta_P \cdot \Delta_P}(000, J) = 101$$

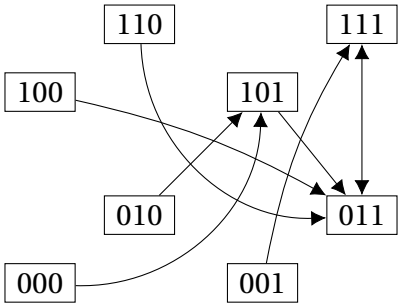

FIGURE 5.2 – Graphe de la dynamique parallèle avec séquence d'entrée $J = (1)$ du module développé dans l'exemple 38.

$$M_{\Delta_1}(000, J) = 011$$
$$M_{\Delta_2}(000, J) = 001$$
$$M_{\Delta_P \cdot \Delta_P \cdot \Delta_P}(111, J) = 111$$
$$M_{\Delta_1}(111, J) = 011$$
$$M_{\Delta_2}(111, J) = 011$$

Nous étendons la définition de graphe de la dynamique d'un réseau d'automates aux modules. Pour ce faire, il nous faut rajouter une information supplémentaire : en plus du mode de mise à jour Δ, l'on doit fournir une séquence d'entrée J de taille suffisament longue pour définir une exécution. On définit ensuite intuitivement le graphe de la dynamique en exécutant le module sur ces paramètres sur toutes les configurations possibles, et en représentant le résultat sous la forme d'un graphe.

Définition 51 (Graphe de la dynamique d'un module)**.** *Soit M un module, Δ un mode de mise à jour et J une séquence d'entrée au moins aussi longue que Δ. Le graphe de la dynamique de M selon Δ et J est le graphe orienté avec sommets dans Λ^S, tel que toute paire de configuration (x, y) est une arête du graphe si et seulement si $M_\Delta(x, J) = y$.*

La figure 5.2 représente le graphe de la dynamique selon le mode de mise à jour parallèle et la séquence d'entrée $J = (1)$ du module développé dans l'exemple 38.

Une telle représentation souffre d'un manque de variation sur la configuration d'entrée. Là où le graphe de la dynamique d'un mode de mise à jour donné d'un réseau d'automates était réduit à décrire uniquement le comportement pour un seul mode de mise à jour, le graphe de la dynamique d'un module est réduit à ne décrire que son comportement pour un mode de mise à jour et une séquence d'entrée. Ainsi, nous ne considérons pas cet objet comme une représentation fidèle des très nombreux comportements dont un module est capable.

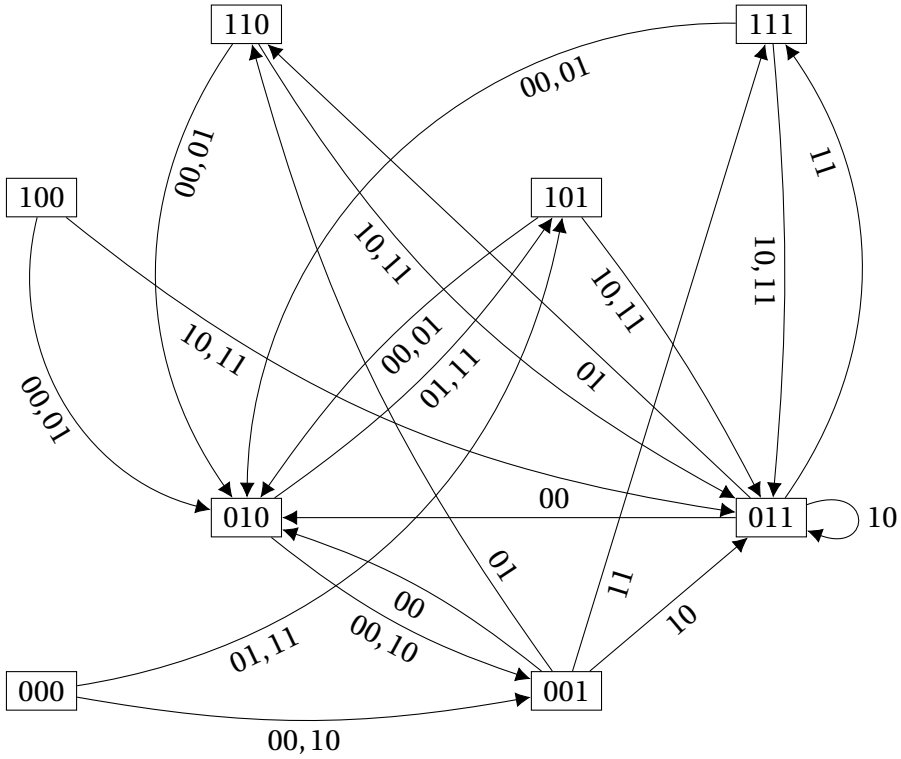

FIGURE 5.3 – Graphe de la dynamique d'entrée avec mode de mise à jour $\Delta = (\{b,c\},a)$ du module développé dans l'exemple 38. L'étiquette 00 représente la séquence d'entrée $J = (0,0)$.

Si la définition ci-dessus est une extension intuitive de la définition équivalente dans le cadre des réseaux d'automates, nous proposons une définition nouvelle propre au cas des modules. Ce que nous appellons graphe de la dynamique d'entrée est défini à partir d'un module et d'un mode de mise à jour, et représente toutes les exécutions possibles pour toutes les séquences d'entrée possibles.

Définition 52 (Graphe de la dynamique d'entrée)**.** *Soit M un module et Δ un mode de mise à jour. Le* graphe de la dynamique d'entrée *de M selon Δ est le graphe orienté et étiqueté avec sommets dans Λ^S et dont l'ensemble d'étiquetage est l'ensemble des séquences d'entrée de longueur égale à la taille de Δ, tel que le tuple (x,J,y) est une arête du graphe si et seulement si $M_\Delta(x,J) = y$.*

La figure 5.3 représente le graphe de la dynamique d'entrée selon le mode de mise à jour $\Delta = (\{b,c\},\{a\})$ du module développé dans l'exemple 38.

En plus des graphes de la dynamique donnés par des modes de mises à jours fixés, nous considérons l'équivalent du graphe de la dynamique asynchrone dans le contexte des modules. Ce graphe qui, dans le contexte des réseaux d'automates renseigne l'effet de toutes les mises à jours d'un seul automate à partir de toutes les configurations, doit

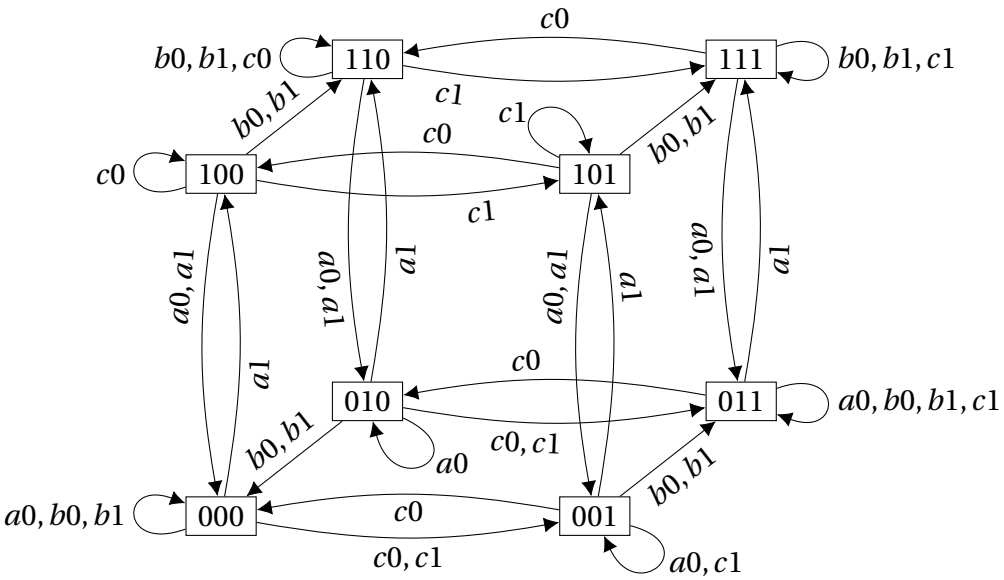

FIGURE 5.4 – Graphe de la dynamique asynchrone du module développé dans l'exemple 38. L'étiquette $a0$ signifie la mise à jour du sommet a avec l'entrée α évaluée à 0.

ici être augmenté de l'information concernant l'affectation des entrées au moment de la mise à jour. Cela signifie que l'ensemble des étiquettes sur les arêtes sortantes à chaque nœud sera le produit cartésien de l'ensemble des automates et de l'ensemble des configurations d'entrée.

Définition 53 (Graphe de la dynamique asynchrone d'un module)**.** *Soit M un module. Le* graphe de la dynamique asynchrone *de M est le graphe orienté étiqueté avec sommets dans Λ^S et étiquettes dans $S \times \Lambda^I$ tel que $(x,(a,i),y)$ est une arête étiquetée du graphe si et seulement si $F_{\{a\}}(x \cdot i) = y$.*

La figure 5.4 représente le graphe de la dynamique asynchrone du module développé dans l'exemple 38.

Ainsi, le graphe de la dynamique asynchrone d'un module permet de décrire le comportement de tout mode de mise à jour composé de singletons, et ce quelle que soit la séquence d'entrée utilisée. Afin de pouvoir étendre cette représentation à tout mode de mise à jour, nous devons introduire la notion correspondante du *graphe de la dynamique totale,* qui ici devra contenir l'information de la configuration d'entrée en addition à l'information de la mise à jour considérée.

Définition 54 (Graphe de la dynamique totale d'un module)**.** *Soit M un module. Le* graphe de la dynamique totale *de M est le graphe orienté étiqueté avec sommets dans Λ^S et étiquettes dans $\wp(S) \times \Lambda^I$ tel que $(x,(\delta,i),y)$ est une arête étiquetée du graphe si et seulement si $M_\delta(x \cdot i) = y$.*

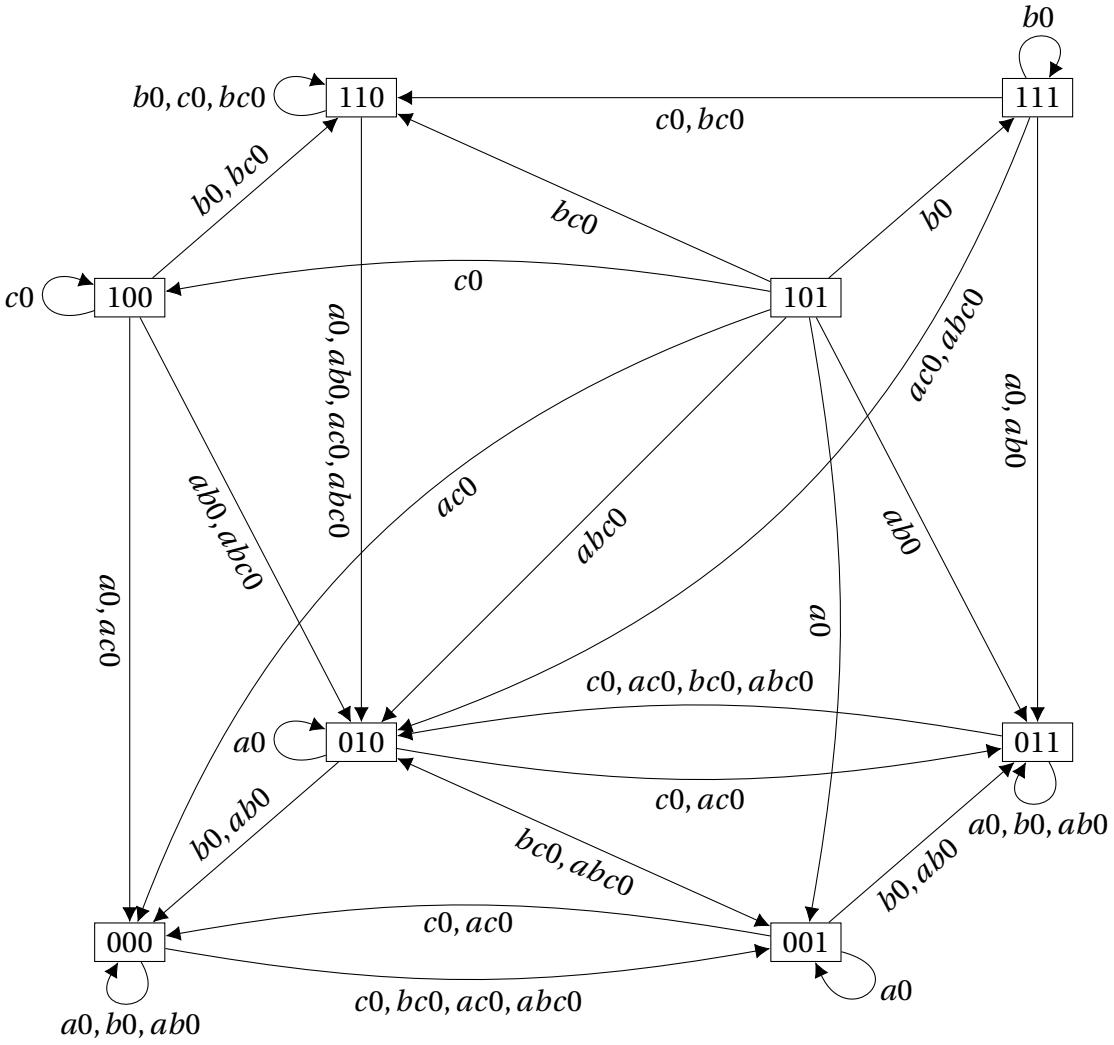

FIGURE 5.5 – Graphe de la dynamique totale du module développé dans l'exemple 38, réduit aux transitions avec la configuration d'entrée $i_\alpha = 0$. L'étiquette $ab0$ représente la mise à jour $\delta = \{a, b\}$ associée à la configuration d'entrée qui associe 0 à α.

Les figures 5.5 et 5.6 représentent le graphe de la dynamique totale du module développé dans l'exemple 38.

Nous faisons le constat assez prévisible que la taille de ces graphes est d'autant plus grande que le nombre d'entrées augmente. Plus spécifiquement, là où le nombre de sommets de chaque graphe est toujours le même, le nombre total d'arêtes étiquetées explose encore plus lourdement que dans le cas des réseaux d'automates. La taille de ces objets est détaillée dans la table en figure 5.7.

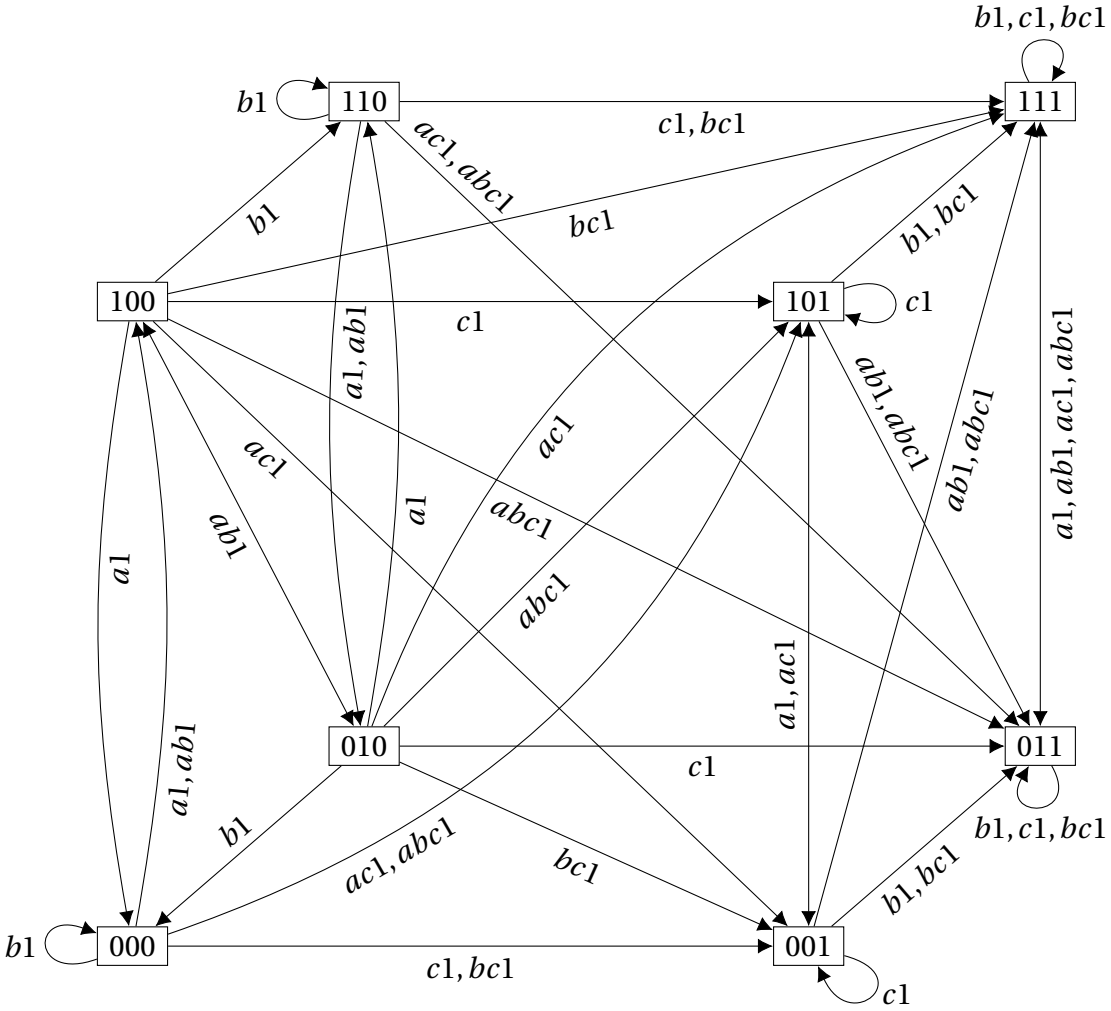

FIGURE 5.6 – Graphe de la dynamique totale du module développé dans l'exemple 38, réduit aux transitions avec la configuration d'entrée $i_\alpha = 1$. L'étiquette $ab1$ représente la mise à jour $\delta = \{a, b\}$ associée à la configuration d'entrée qui associe 1 à α.

5.2.3 Branchements

Les modules peuvent être transformés pour remplacer l'influence extérieure d'une entrée par l'influence spécifique d'un automate. Cette transformation est faite par le biais de branchements, qui sont de deux types : récursif, et non récursif. Ces deux types de modification opèrent par la substitution de variables dans les configurations d'une exécution. Dans le cas du branchement non récursif, les variables d'entrée retirées sont remplacées par les variables d'automates d'un autre réseau. Dans le cas du branchement récursif, les variables d'entrées retirées sont remplacées par les

Type de réseau	Module		Réseau d'automates	
Mesure	Nœuds	Arêtes	Nœuds	Arêtes
G. d'interaction	n	$n \times (n+i)$	n	n^2
G. de la dynamique	l^n	l^n	l^n	l^n
G. de la dynamique d'entrée	l^n	l^{nid}	N/A	N/A
G. de la dynamique asynchrone	l^n	$n \times l^{ni}$	l^n	$n \times l^n$
G. de la dynamique totale	l^n	$2^n \times l^{ni}$	l^n	$2^n \times l^n$

FIGURE 5.7 – Table renseignant la taille maximale des différents graphes de la dynamique explorés dans cette section, lorsqu'applicable. Les nombres l, n, i et d représentent respectivement le nombre d'états, le nombre d'automates, le nombre d'entrées et la longueur du mode de mise à jour appliqué.

variables d'automates du réseau lui-même.

Pour un ensemble de réseaux donné, il existe une multitude de façons différentes de combiner entrées et automates. Même en considérant le cas plus simple du branchement récursif d'un réseau seul, le nombre total de branchements possibles est de $i \times (n+1)$, où i est le nombre d'entrées, et n le nombre d'automates. C'est pour cette raison que nous définissons pour chaque branchement une fonction $\mathtt{w} : I \nrightarrow S$, où $\mathrm{dom}(\mathtt{w})$ est l'ensemble d'entrées du réseau influencé par le branchement, et $\mathrm{img}(\mathtt{w})$ est l'ensemble des automates du réseau dont l'influence va remplacer celle des entrées dans $\mathrm{dom}(\mathtt{w})$. Cette fonction, dite fonction de branchement, est une fonction partielle ; en effet, il est tout à fait possible de laisser des entrées non affectées par un branchement.

Nous rappellons que l'utilisation de l'opérateur de composition $\circ$ entre vecteurs et fonctions est définie en section 1.3.

Définition 55 (Branchement récursif)**.** *Soit M un module, et $\mathtt{w} : I \nrightarrow S$ une fonction de branchement. On définit le branchement récursif de M par $\mathtt{w}$, noté $\circlearrowright_{\mathtt{w}} M$, le module défini sur l'ensemble d'automates S et l'ensemble d'entrées $I \setminus dom(\mathtt{w})$ qui, pour tout automate s, définit la fonction locale f'_s telle que*

$$f'_s(x' \cdot i') = f_s((x' \cdot i') \circ \hat{\mathtt{w}})$$

où f_s est la fonction locale de s dans M, et $\hat{\mathtt{w}}$ est la fonction définie sur $S \cup I \to S \cup I \setminus dom(\mathtt{w})$, telle que

$$\hat{\mathtt{w}}(a) = \begin{cases} \mathtt{w}(a) & si\ a \in dom(\mathtt{w}) \\ a & sinon \end{cases}.$$

Ainsi, le résultat d'un branchement récursif est un nouveau module dans lequel, pour tout α dans le domaine de $\mathtt{w}$, l'entrée α est absente du module, et son emploi dans le réseau est remplacé par l'automate $\mathtt{w}(\alpha)$.

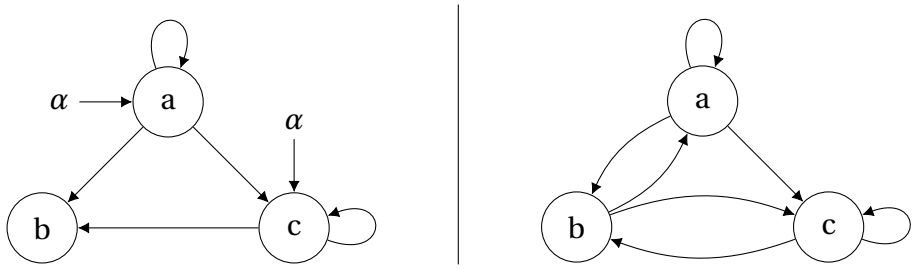

FIGURE 5.8 – Illustration du branchement décrit dans l'exemple 41. À gauche de la figure, le graphe d'interaction du module M. À droite de la figure, le graphe d'interaction du réseau d'automates $\circlearrowleft_{\mathtt{w}} M$.

Exemple 41. *Nous considérons le module M développé dans l'exemple 38. Soit $\mathtt{w}$ la fonction de branchement définie par $dom(\mathtt{w}) = \{b\}$ et $\mathtt{w}(\alpha) = b$. Le résultat du branchement récursif $\circlearrowleft_{\mathtt{w}} M$ est le réseau d'automate qui définit les fonctions locales*

$$f'_a(x \cdot i) = f_a((x \cdot i) \circ \hat{\mathtt{w}}) = \neg(x \cdot i)_{\hat{\mathtt{w}}(a)} \wedge (x \cdot i)_{\hat{\mathtt{w}}(\alpha)} = \neg(x \cdot i)_a \wedge (x \cdot i)_b = \neg x_a \wedge x_b$$
$$f'_b(x \cdot i) = f_b((x \cdot i) \circ \hat{\mathtt{w}}) = (x \cdot i)_{\hat{\mathtt{w}}(a)} \vee (x \cdot i)_{\hat{\mathtt{w}}(c)} = (x \cdot i)_a \vee (x \cdot i)_c = x_a \vee x_c$$
$$f'_c(x \cdot i) = f_c((x \cdot i) \circ \hat{\mathtt{w}}) = (\neg(x \cdot i)_{\hat{\mathtt{w}}(a)} \wedge \neg(x \cdot i)_{\hat{\mathtt{w}}(c)}) \vee (x \cdot i)_{\hat{\mathtt{w}}(\alpha)} = (\neg x_a \wedge \neg x_c) \vee x_b.$$

La figure 5.8 représente les graphes d'interaction des deux modules développés dans l'exemple 41.

L'exemple 41 fait la démonstration d'un phénomène propre à certains branchements récursifs. Lorsque la fonction de branchement $\mathtt{w}$ est une fonction totale, toutes les entrées sont remplacées par des automates et, par conséquent, le module obtenu ne dispose pas d'entrées. Dans ce cas, il est correct de désigner le module obtenu comme un réseau d'automates.

Là où le branchement récursif permet d'introduire de nouveaux cycles dans un module, le branchement non récursif permet d'introduire de nouveaux automates. Pour ce faire, deux modules différents sont combinés.

Définition 56 (Branchement non récursif)**.** *Soient M et M' deux modules tels que $S \cap S' = I \cap I' = \emptyset$, et $\mathtt{w}$ une fonction de branchement définie sur $I' \twoheadrightarrow S$. On définit le branchement non récursif de M sur M' d'après $\mathtt{w}$, noté $M \rightarrowtail_{\mathtt{w}} M'$, comme le module ayant pour ensemble d'automates $S \cup S'$ et pour ensemble d'entrées $I \cup I' \setminus dom(\mathtt{w})$ tel que, pour tout automate s, le module définit la fonction locale*

$$f''_s(x'' \cdot i'') = \begin{cases} f_s(x''|_S \cdot i''|_I) & si\ s \in S \\ f'_s((x'' \cdot i''|_{I'}) \circ \hat{\mathtt{w}}) & si\ s \in S' \end{cases}$$

où f_s et f'_s sont les fonctions locales de s dans M et M' respectivement, et $\hat{\mathtt{w}}$ est la fonction

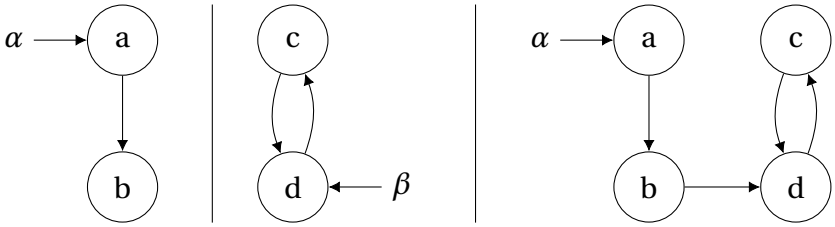

FIGURE 5.9 – Illustration du branchement décrit dans l'exemple 42. De gauche à droite dans la figure sont représentés les graphes d'interaction des modules M, M' et $M \rightarrowtail_{\mathtt{w}} M'$.

définie sur $S' \cup I' \rightarrow S \cup S' \cup I' \setminus dom(\mathtt{w})$ *telle que*

$$\hat{\mathtt{w}}(a) = \begin{cases} \mathtt{w}(a) & si\ a \in dom(\mathtt{w}) \\ a & sinon \end{cases} .$$

Exemple 42. *Soit M le module tel que $S = \{a, b\}$, $I = \{\alpha\}$, $f_a(x \cdot i) = i_\alpha$ et $f_b(x \cdot i) = \neg x_a$. Soit M' le module tel que $S' = \{c, d\}$, $I' = \{\beta\}$, $f_c(x \cdot i) = \neg x_d$ et $f_d(x \cdot i) = x_c \vee i_\beta$. Soit $\mathtt{w}$ la fonction de branchement définie sur $\{\beta\} \rightarrow S$ telle que $\mathtt{w}(\beta) = b$. Le module $M \rightarrowtail_{\mathtt{w}} M'$ est défini sur les ensembles $S'' = \{a, b, c, d\}$, $I'' = \{\alpha\}$ et définit les fonctions locales*

$$\begin{aligned} f''_a(x \cdot i) &= f_a(x|_S \cdot i|_I) = i_\alpha \\ f''_b(x \cdot i) &= f_b(x|_S \cdot i|_I) = \neg x_a \\ f''_c(x \cdot i) = f_c((x \cdot i) \circ \hat{\mathtt{w}}) &= \neg (x \cdot i)_{\hat{\mathtt{w}}(d)} = \neg (x \cdot i)_d = \neg x_d \\ f''_d(x \cdot i) = f_d((x \cdot i) \circ \hat{\mathtt{w}}) = (x \cdot i)_{\hat{\mathtt{w}}(c)} \vee (x \cdot i)_{\hat{\mathtt{w}}(\beta)} &= (x \cdot i)_c \vee (x \cdot i)_b = x_c \vee x_b. \end{aligned}$$

La figure 5.9 représente les graphes d'interaction des trois modules développés dans l'exemple 42.

Dans notre utilisation des outils que sont les branchement non récursifs, un cas de fonction de branchement ressort comme particulièrement utile. Nous parlons ici du cas où un branchement non récursif est appliqué avec une fonction de branchement dont le domaine est vide. Cette opération se contente de simplement juxtaposer deux modules sans altérer leurs comportements. Cette opération parait suffisament distincte et utile pour mériter une notation particulière, et nous l'utilisons pour définir l'union sur les modules.

Définition 57 (Union de modules)**.** *Soient M et M' deux modules. On définit l'*union *de ces modules, notée $M \cup M'$, comme le résultat de l'opération $M \rightarrowtail_{\mathtt{w}} M'$, pour $\mathtt{w}$ la fonction telle que $dom(\mathtt{w}) = \emptyset$.*

Similairement, on peut définir le cas particulier du branchement récursif qui utilise une fonction de branchement définie sur un modèle vide. Le comportement de cet

opérateur, noté $\circlearrowleft_\emptyset$, est trivial ; aucun automate n'est rajouté ou modifié, et le module reste le même. L'opérateur $\circlearrowleft_\emptyset$ est donc l'identité sur les modules.

Enfin et par cohérence avec les définitions précédentes, nous définissons le module vide, noté $M_\emptyset$, comme le module défini sur les ensembles d'automates et d'entrées $S = I = \emptyset$.

5.3 Résultats

5.3.1 Universalité des branchements

Les opérateurs union et identité que nous décrivons dans la section précédente vérifient toutes le propriétés que l'on attend de tels opérateurs. Elles sont englobées par la propriété suivante.

Propriété 9. *Soient M, M' et M'' trois modules. Les équations suivantes sont vérifiées :*

$$\circlearrowleft_\emptyset M = M \tag{5.1}$$
$$M \cup M_\emptyset = M \tag{5.2}$$
$$M \cup M' = M' \cup M \tag{5.3}$$
$$(M \cup M') \cup M'' = M \cup (M' \cup M'') \tag{5.4}$$

Démonstration. Pour M un module, on considère le module $\circlearrowleft_\emptyset M$. Pour tout automate s, la fonction locale de s dans $\circlearrowleft_\emptyset M$ est définie par $f'_s(x \cdot i) = f_s((x \cdot i) \circ \hat{\mathtt{w}})$, où $\mathtt{w}$ est la fonction de branchement vide et f_s est la fonction locale de s dans M. Comme $\mathrm{dom}(\mathtt{w}) = \emptyset$, on observe $\hat{\mathtt{w}} = \mathrm{id}$ ce qui implique $f'_s = f_s$ et l'équation 5.1 est donc vérifiée.

Pour M un module, le module $M \cup M_\emptyset$ est défini sur l'ensemble d'automates $S \cup \emptyset = S$ et l'ensemble d'entrées $I \cup \emptyset = I$ tel que, pour tout automate s dans S, la fonction locale de s dans $M \cup M_\emptyset$ est définie comme $f'_s(x \cdot i) = f_s(x|_S \cdot i|_I) = f_s(x \cdot i)$, pour f_s la fonction locale de s dans M. On en déduit l'équation 5.2.

Soient M et M' deux modules. Le module $M \cup M'$ a pour ensemble d'automates $S \cup S'$ et pour ensemble d'entrées $I \cup I'$ tel que, pour tout automate $s \in S$, le module définit la fonction locale $g_s(x \cdot i) = f_s(x|_S \cdot i_I)$, et pour tout automate $s' \in S'$, le module définit la fonction locale $g_{s'}(x \cdot i) = f'_{s'}((x \cdot i|_{I'}) \circ \hat{\mathtt{w}})$. Cependant, comme la fonction de branchement considérée a pour domaine l'ensemble vide, on en déduit de nouveau que $\hat{\mathtt{w}} = \mathrm{id}$, ce qui implique que

$$\begin{aligned} g_{s'}(x \cdot i) &= f'_{s'}((x \cdot i|_{I'})|_{S' \cup I' \setminus \mathrm{dom}(\mathtt{w})}) \\ &= f'_{s'}((x \cdot i|_{I'})|_{S' \cup I'}) \\ &= f'_{s'}(x|_{S'} \cdot i|_{I'}). \end{aligned}$$

Maintenant, considérons le module $M' \cup M$. Ce module a pour ensemble d'automates $S' \cup S = S \cup S'$, et pour ensemble d'entrées $I' \cup I = I \cup I'$. Pour tout automate $s' \in S'$, le module définit la fonction locale $h_{s'}(x \cdot i) = f'_{s'}(x|_{S'} \cdot i|_{I'})$ et, pour tout automate $s \in S$, la fonction locale de s dans $M' \cup M$ vérifie

$$\begin{aligned} h_s(x \cdot i) &= f_s((x \cdot i|_I) \circ \hat{\mathtt{w}}) \\ &= f_s((x \cdot i|_I)|_{S \cup I \setminus \mathrm{dom}(\mathtt{w})}) \\ &= f_s((x \cdot i|_I)|_{S \cup I}) \\ &= f_s(x|_S \cdot i|_I). \end{aligned}$$

On en déduit l'équation 5.3.

En nous fondant sur les développements du paragraphe précédent, on observe que l'union de deux modules M et M' est un module qui est défini sur l'union des ensembles d'automates et des ensembles d'entrées de ces deux modules, et qui vérifie

$$f''_s(x \cdot i) = \begin{cases} f_s(x|_S \cdot i|_I) & \text{si } s \in S \\ f'_s(x|_{S'} \cdot i|_{I'}) & \text{si } s \in S' \end{cases}.$$

On en déduit que le module $(M \cup M') \cup M''$ est le module défini sur les ensembles $S \cup S' \cup S''$ et $I \cup I' \cup I''$ tel que

$$g_s(x \cdot i) = \begin{cases} f_s(x|_S \cdot i|_I) & \text{si } s \in S \\ f'_s(x|_{S'} \cdot i|_{I'}) & \text{si } s \in S' \\ f''_s(x|_{S''} \cdot i|_{I''}) & \text{si } s \in S'' \end{cases}.$$

Par le même développement et en interchangeant M et M'', le module $M \cup (M' \cup M'')$ est défini de la même façon, ce qui implique l'équation 5.4. □

On peut de plus observer que tout branchement non récursif peut être reproduit par la succession d'une union de modules et d'un branchement récursif, fait caractérisé par la propriété suivante.

Propriété 10. *Soient M, M' deux modules et $\mathtt{w}$ une fonction de branchement qui définit $M \rightarrowtail_{\mathtt{w}} M'$. L'équation suivante est vérifiée :*

$$M \rightarrowtail_{\mathtt{w}} M' = \circlearrowleft_{\mathtt{w}} (M \cup M').$$

Démonstration. Soient M et M' deux modules et $\mathtt{w} : I' \nrightarrow S$ une fonction de branchement. On observe que la fonction de branchement $\mathtt{w}$, définie comme fonction partielle de I' vers S, est également définie comme une fonction partielle de $I \cup I'$ vers $S \cup S'$. Ainsi elle est une fonction de branchement adéquate pour opérer le branchement $\circlearrowleft_{\mathtt{w}} (M \cup M')$.

Pour s dans S, la fonction locale définie par $M \rightarrowtail_{\mathtt{w}} M'$ est la fonction $g_s(x \cdot i) = f_s(x|_S \cdot i|_I)$. Pour s' dans S', cette fonction locale devient $g_{s'}(x \cdot i) = f'_{s'}((x \cdot i) \circ \hat{\mathtt{w}})$.

Dans le cas du module $\circlearrowleft_{\mathtt{w}}(M \cup M')$, la fonction locale de l'automate s dans S est $h_s(x \cdot i) = f''_s((x \cdot i) \circ \hat{\mathtt{w}})$, pour f''_s la fonction locale de s dans $M \cup M'$. Cela est égal à $f_s((x|_S \cdot i|_I) \circ \hat{\mathtt{w}})$, ce qui, par définition de $\mathtt{w}$, est égal à $f_s(x|_S \cdot i|_I)$. Pour tout automate s' dans S', la fonction locale de s' dans $\circlearrowleft_{\mathtt{w}}(M \cup M')$ est, par le même raisonnement, $h_{s'}(x \cdot i) = f'_{s'}((x|_{S'} \cdot i|_{I'}) \circ \hat{\mathtt{w}})$, ce qui est équivalent à $h_{s'}(x \cdot i) = f'_{s'}((x \cdot i) \circ \hat{\mathtt{w}})$, d'où le résultat. □

Les opérations d'union et de branchement récursif sont assez expressives pour permettre la construction de tout module à partir de modules de taille un. Cette expressivité est capturée par le théorème suivant.

Théorème 4 (Universalité des branchements)**.** *Soit M un module. Il existe un ensemble $\{M_1, M_2, \ldots, M_k\}$ de modules de taille 1 et $\mathtt{w}$ une fonction de branchement tels que*

$$M = \circlearrowleft_{\mathtt{w}} \bigcup_{i=0}^{k} M_k.$$

Démonstration. Pour prouver ce résultat, nous construisons l'ensemble de modules de taille 1 à partir des fonctions locales du module M.

Premièrement, pour tout s dans S, nous définissons la variable d'entrée α_s telle que $\alpha_s \notin I$. L'ensemble de ces variables constitue un ensemble I' distinct de l'ensemble I. De plus, nous définissons $\mu_s : \{s\} \cup I \cup I' \to S \cup I$ la fonction définie telle que

$$\mu(r) = \begin{cases} r & \text{si } r = s \\ r & \text{si } r \in I \\ q & \text{si } r \in I' \text{ et } r = \alpha_q \end{cases}.$$

Pour chaque automate s dans S de nouveau, nous définissons le module M_s comme le module défini sur l'ensemble d'automates $\{s\}$ et l'ensemble d'entrées $I \cup I'$ tel que la fonction locale de s est définie par $f^s_s(x \cdot i) = f_s((x \cdot i) \circ \mu_s)$.

Pour M un module, nous prenons l'ensemble de modules $\{M_s | s \in S\}$ et la fonction de branchement $\mathtt{w} : I \cup I' \twoheadrightarrow S$ définie sur le domaine $\mathrm{dom}(\mathtt{w}) = I'$ et telle que $\mathtt{w}(\alpha_s) = s$ pour tout automate s. Nous vérifions maintenant que la recomposition de ces éléments nous redonne le module de base.

Pour tout s dans S, la fonction locale de s dans $\bigcup\{M_s | s \in S\}$ est $f^s_s(x \cdot i) = f_s((x \cdot i) \circ \mu_s)$. Dans le module $\circlearrowleft_{\mathtt{w}} \bigcup\{M_s | s \in S\}$ qui est défini sur les ensembles S et I, cette fonction locale devient $f'_s(x \cdot i) = f^s_s((x \cdot i) \circ \hat{\mathtt{w}}) = f_s((x \cdot i) \circ \mu_s \circ \hat{\mathtt{w}})$.

On observe que, par définition, $\mu_s|_I = id$, et $\hat{\mathtt{w}}|_{S \cup I} = id$. De plus, $(\mu_s \circ \hat{\mathtt{w}})|_{I'} = id$. On en déduit que $\mu_s \circ \hat{\mathtt{w}} = id$, et donc que $f'_s(x \cdot i) = f_s(x \cdot i)$, ce qui implique $M = \bigcup\{M_s | s \in S\}$. □

5.3.2 Effets des branchements sur la dynamique

Les branchements sont des opérations simples qui modifient la description des modules sur lesquels ils opèrent. Ces modifications entraînent inévitablement des modifications sur la dynamique de ces modules. Dans les quelques résultats suivants, nous nous intéressons à observer les effets des différents types de branchement sur la dynamique des modules étudiés.

L'étude de l'évolution de la dynamique d'un module requiert une méthode de comparaison entre dynamiques. Ici, nous nous concentrons sur la dynamique totale des modules, et nous définissons une relation d'inclusion spécifique à ces graphes permettant d'étendre la définition d'inclusion sur les graphes. Afin de permettre cette définition, nous introduisons la notation $D(M)$ qui décrit le graphe de la dynamique totale du module M.

Définition 58 (Inclusion de dynamiques totales)**.** *Soient M et M' deux modules, tels que $S \subseteq S'$ et $I \subseteq I'$. On définit $D(M) \subseteq D(M')$ si et seulement si, pour toute arête $(x, (\delta, i), y)$ dans $D(M)$, il existe une arête $(x', (\delta', i'), y')$ dans $D(M')$ qui vérifie $x'|_S = x$, $y'|_S = y$, $\delta' \cap S = \delta$, et $i'|_I = i$.*

Nous définissons également l'extension intuitive du produit cartésien de deux graphes de la dynamique totale.

Définition 59 (Produit cartésien de dynamiques totales)**.** *Soient M et M' deux modules. Nous définissons le produit cartésien $D(M) \times D(M')$ comme le graphe étiqueté défini sur l'ensemble de sommets $\Lambda^{S \cup S'}$, et l'ensemble d'étiquettes $\wp(S \cup S') \times \Lambda^{I \cup I'}$, tel que le graphe $D(M) \times D(M')$ possède l'arête $(x \cdot x', (\delta \cup \delta', i \cdot i'), y \cdot y')$ si et seulement si le graphe $D(M)$ possède l'arête $(x, (\delta, i), y)$ et le graphe $D(M')$ possède l'arête $(x', (\delta', i'), y')$.*

Les effets des opérations d'union et de branchement récursif vide sur la dynamique des modules sont triviaux, et explicités par la propriété suivante.

Propriété 11 (Effet dynamique des branchements vides)**.** *Soient M, M' deux modules. Les équations suivantes sont vérifiées :*

$$D(\circlearrowleft_{\emptyset} M) = D(M) \tag{5.5}$$
$$D(M \cup M') = D(M) \times D(M'). \tag{5.6}$$

Démonstration. L'équation 5.5 se vérifie par l'application de la propriété 9.

Soient M et M' deux modules. Le module $M \cup M'$ est défini sur les automates $S \cup S'$ et les entrées $I \cup I'$ tel que, pour tout automate S, la fonction locale de s dans $M \cup M'$ vérifie

$$f''_s(x \cdot i) = \begin{cases} f_s(x|_S \cdot i|_I) & \text{si } s \in S \\ f'_s(x|_{S'} \cdot i|_{I'}) & \text{si } s \in S' \end{cases}$$

pour f_s et f'_s les fonctions locales de s dans M et M' respectivement.

Le graphe de la dynamique $D(M \cup M')$ est défini sur les sommets $\Lambda^{S \cup S'}$ et les étiquettes $\wp(S \cup S') \times \Lambda^{I \cup I'}$, et tel que $(x \cdot x', (\delta \cup \delta', i \cdot i'), y \cdot y')$ est une arête du graphe si et seulement si $(M \cup M')_{\delta \cup \delta'}(x \cdot x' \cdot i \cdot i') = y \cdot y'$. On procède aux étapes logiques suivantes :

$$
\begin{aligned}
& (M \cup M')_{\delta \cup \delta'}(x \cdot x' \cdot i \cdot i') = y \cdot y' \\
\iff & \forall s \in S \cup S', (M \cup M')_{\delta \cup \delta'}(x \cdot x' \cdot i \cdot i')_s = (y \cdot y')_s \\
\iff & (\forall s \in S, (M \cup M')_{\delta \cup \delta'}(x \cdot x' \cdot i \cdot i')_s = y_s) \\
& \wedge (\forall s' \in S', (M \cup M')_{\delta \cup \delta'}(x \cdot x' \cdot i \cdot i')_{s'} = y'_{s'}) \\
\iff & \forall s \in S, \begin{cases} f''_s(x \cdot x' \cdot i \cdot i') & \text{si } s \in \delta \cup \delta' \\ (x \cdot x')_s & \text{sinon} \end{cases} = y_s \\
& \wedge \forall s' \in S', \begin{cases} f''_{s'}(x \cdot x' \cdot i \cdot i') & \text{si } s' \in \delta \cup \delta' \\ (x \cdot x')_{s'} & \text{sinon} \end{cases} = y'_{s'} \\
\iff & \forall s \in S, \begin{cases} f_s((x \cdot x')|_S \cdot (i \cdot i')|_I) & \text{si } s \in \delta \\ x_s & \text{sinon} \end{cases} = y_s \\
& \wedge \forall s' \in S', \begin{cases} f'_{s'}((x \cdot x')|_{S'} \cdot (i \cdot i')|_{I'}) & \text{si } s' \in \delta' \\ x'_{s'} & \text{sinon} \end{cases} = y'_{s'} \\
\iff & \forall s \in S, \begin{cases} f_s(x \cdot i) & \text{si } s \in \delta \\ x_s & \text{sinon} \end{cases} = y_s \\
& \wedge \forall s' \in S', \begin{cases} f'_{s'}(x' \cdot i') & \text{si } s' \in \delta' \\ x'_{s'} & \text{sinon} \end{cases} = y'_{s'} \\
\iff & (\forall s \in S, M_\delta(x \cdot i)_s = y_s) \wedge (\forall s' \in S', M'_{\delta'}(x' \cdot i')_{s'} = y'_{s'}) \\
\iff & (M_\delta(x \cdot i) = y) \wedge (M'_{\delta'}(x' \cdot i') = y').
\end{aligned}
$$

Il résulte de ce développement qu'il existe une arête $(x \cdot x', (\delta \cup \delta', i \cdot i'), y \cdot y')$ dans la dynamique $D(M \cup M')$ si et seulement si le graphe $D(M)$ accepte l'arête $(x, (\delta, i), y)$ et si le graphe $D(M')$ accepte l'arête $(x', (\delta', i'), y')$. Ainsi, le graphe $D(M \cup M')$ coïncide avec la définition du graphe $D(M) \times D(M')$, et l'équation 5.6 est vérifiée. □

Lorsqu'un réseau est altéré par un branchement récursif, le module résultant est identique au réseau initial à la différence près que l'influence de certaines entrées est maintenant remplacée par l'influence d'automates. Cela signifie en particulier que tous les comportements du réseau final peuvent être reproduits dans le réseau initial en prenant les séquences d'entrées qui miment le comportement des automates concernés par le branchement. Cette observation implique que le branchement récursif ne peut que réduire la dynamique des réseaux étudiés ; plus précisément, la dynamique du nouveau réseau est strictement incluse dans la dynamique initiale.

Propriété 12 (Effet dynamique du branchement récursif)**.** *Un module M et une fonction de branchement non vide* `w` *vérifient*

$$D(\circlearrowleft_{\mathtt{w}} M) \subsetneq D(M).$$

Démonstration. Soit M un module et $\mathtt{w}$ une fonction de branchement. Les graphes $D(M)$ et $D(\circlearrowleft_{\mathtt{w}} M)$ sont tous deux définis sur le même ensemble de sommets Λ^S. Le graphe $D(M)$ accepte ses étiquettes dans l'ensemble $\wp(S) \times \Lambda^I$, là où le graphe $D(\circlearrowleft_{\mathtt{w}} M)$ accepte ses étiquettes dans l'ensemble $\wp(S) \times \Lambda^{I \setminus \mathrm{dom}(\mathtt{w})}$.

Soit $(x, (\delta, i), y)$ une arête dans $D(\circlearrowleft_{\mathtt{w}} M)$. On définit i' la configuration d'entrée définie dans Λ^I telle que

$$i'_\alpha = \begin{cases} x_{\mathtt{w}(\alpha)} & \text{si } \alpha \in \mathrm{dom}(\mathtt{w}) \\ i_\alpha & \text{sinon} \end{cases}.$$

On observe que, par définition du branchement récurif $\mathtt{w}$, l'arête $(x, (\delta, i'), y)$ est une arête du graphe $D(M)$. Comme $i'|_{I \setminus \mathrm{dom}(\mathtt{w})} = i$, $D(\circlearrowleft_{\mathtt{w}} M) \subseteq D(M)$. Enfin, comme le branchement $\mathtt{w}$ n'est pas vide, $D(\circlearrowleft_{\mathtt{w}} M) \neq D(M)$, ce qui implique le résultat. □

Cette propriété est illustrée en figure 5.10 sur la dynamique totale du réseau d'automate obtenu dans l'exemple 41. La dynamique illustrée en figure 5.10 est incluse dans la dynamique illustrée dans les figures 5.5 et 5.6.

Lorsque l'on combine deux modules par le biais d'un branchement non récursif, la dynamique obtenue est une sous-dynamique du produit cartésien des dynamiques des réseaux initiaux.

Propriété 13 (Effet dynamique du branchement non récursif)**.** *Deux modules M, M' et une fonction de branchement non vide $\mathtt{w}$ vérifient*

$$D(M \rightarrowtail_{\mathtt{w}} M') \subsetneq D(M) \times D(M').$$

Démonstration.

$$\begin{aligned} D(M \rightarrowtail_{\mathtt{w}} M') &= D(\circlearrowleft_{\mathtt{w}} (M \cup M')) && \text{(Propriété 10)} \\ &\subsetneq D(M \cup M') && \text{(Propriété 12)} \\ &= D(M) \times D(M'). && \text{(Propriété 11)} \end{aligned}$$

□

5.3.3 Modularité et simulation

Les modules sont une généralisation des réseaux d'automates booléens, qui sont eux-mêmes une généralisation des automates cellulaires. En cela, les modules possèdent une place forte pour proposer des résultats généraux de simulation sur les modèles de calcul naturel. Ainsi, nous proposons d'utiliser la modularité de ce modèle afin de construire un méta-théorème qui dérive des résultats très variés de simulation à partir d'une preuve de simulation locale.

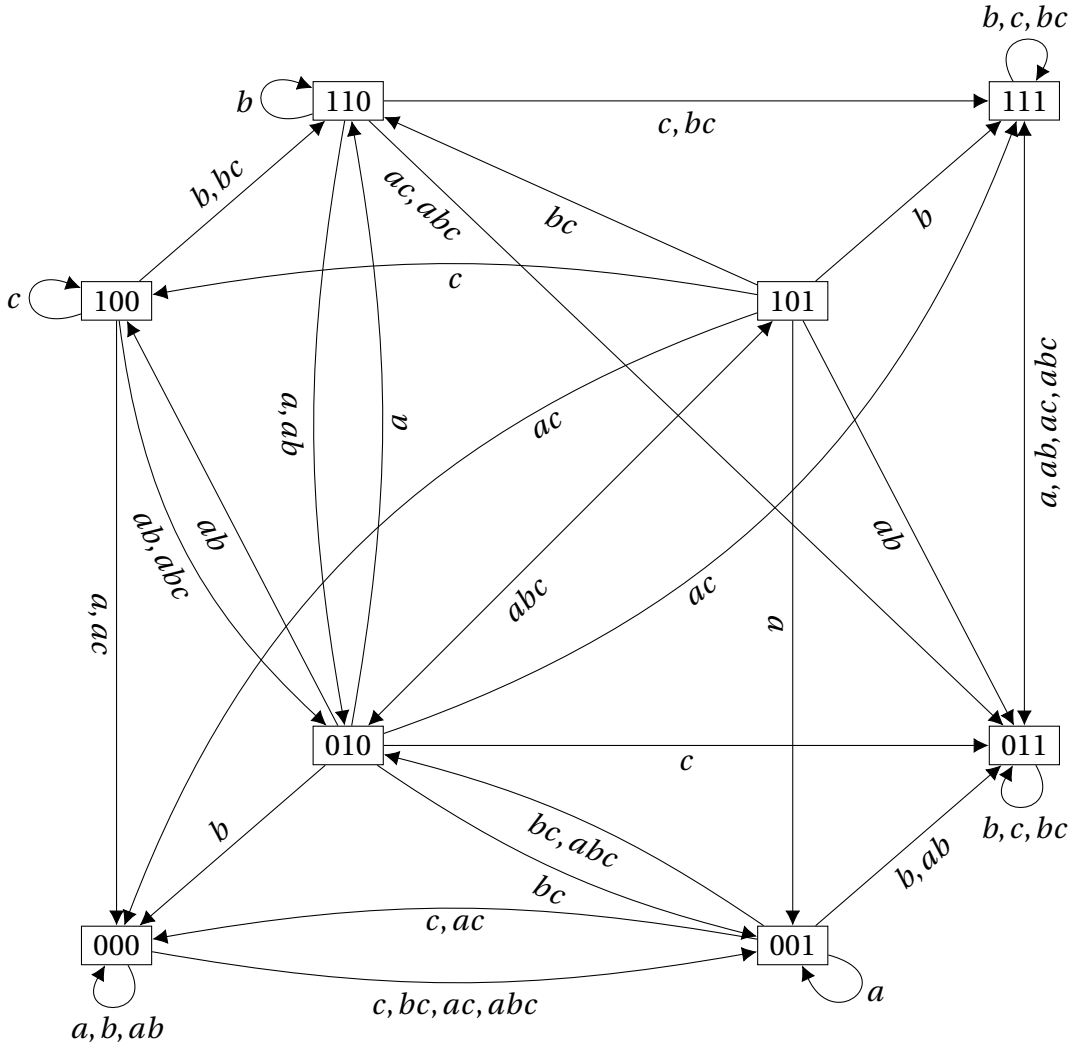

FIGURE 5.10 – Graphe de la dynamique totale du réseau d'automates booléen obtenu par le branchement récursif développé dans l'exemple 41. L'étiquette ab représente la mise à jour $\delta = \{a, b\}$.

La simulation locale nous permet de définir une famille de modules qui reproduisent le comportement local d'automates ou de cellules dans d'autres modèles. Ainsi, la composition de ces modules en un super-réseau de modules simule naturellement les réseaux dont les éléments sont localement simulés.

Plus formellement, supposons un réseau d'automates F composé d'un ensemble d'automates S à états dans l'ensemble Λ. Nous simulons localement chaque automate a dans S par un module M_a, défini sur un ensemble d'automates S_a et un ensemble d'entrées I_a. Chaque module M_s utilise un ensemble d'états Λ'.

Le module M_a représente l'automate a dans notre composition finale. Ainsi, la configuration de M_a a pour mission d'encoder l'état de a à tout moment. Cela est

exprimé par le biais de ce que nous appellons un encodage.

Définition 60 (Encodage)**.** *Soit R un ensemble d'automates, et Λ, Λ' deux ensembles d'états. Un* encodage *sur R de Λ' vers Λ est une fonction partielle surjective $\phi : \Lambda'^R \twoheadrightarrow \Lambda \cup \{\bullet\}$.*

Exemple 43. *Soit les ensembles $Q = \{a, b\}$, $\Lambda' = \{0, 1, 2\}$ et $\Lambda = \{0, 1\}$. On définit ϕ l'encodage sur Q de Λ' vers Λ défini tel que*

$$\phi(00) = 0$$
$$\phi(01) = 1$$
$$\phi(10) = 2$$
$$\phi(11) = \bullet$$

La valeur $\bullet$ signifie que la configuration considérée n'a pas de transcription dans l'encodage considéré.

Dans notre construction, chaque module M_a joue le rôle d'un automate de M. Nous nommons ϕ_a l'encodage sur S_a de Λ' vers Λ, pour chaque a dans S.

Si, dans M, les différents automates interagissent par influence, alors les modules $\{M_s | s \in S\}$ doivent être interconnectés. Pour toute paire d'automates distincts a, b dans S, nous définissons le branchement $\mathtt{w}_{a,b} : I_b \nrightarrow S_a$ qui précise la façon dont le module M_a est branché sur le module M_b. Ici, l'ensemble $\mathrm{codom}(\mathtt{w}_{a,b})$ sert d'interface de communication de a vers b. Avant que le branchement soit opéré, cette interface est représentée virtuellement par l'ensemble des entrées décrit par $\mathrm{dom}(\mathtt{w}_{a,b})$. Les états de ces nœuds et de ces entrées permettent au module M_a de transférer son état à l'automate M_b. Afin de s'assurer que cette communication soit cohérente avec les encodages précédemment définis, nous introduisons l'encodage $\phi_{a,b}$ défini sur $\mathrm{codom}(\mathtt{w}_{a,b})$ de Λ' vers Λ, pour chaque paire d'automates distincts a, b dans S. Dans la pratique, la valeur donnée par l'encodage $\phi_{a,b}$ devra être cohérente avec la valeur de l'encodage ϕ_a, pour tout b différent de a dans S.

En condition supplémentaire, il est nécessaire que les fonctions de branchement soient injectives. Ainsi, nous définissons la fonction $\mathtt{w}^{-1}_{a,b} : \mathrm{codom}(\mathtt{w}_{a,b}) \to \mathrm{dom}(\mathtt{w}_{a,b})$ telle que $\mathtt{w}^{-1}_{a,b} \circ \mathtt{w}_{a,b} = id$. Nous pouvons maintenant définir la simulation locale d'une fonction de notre réseau d'automates F par un module.

Définition 61 (Simulation locale de réseau d'automates)**.** *Pour F un réseau d'automates, a un automate dans S et Δ un mode de mise à jour de M_a, le module M_a* simule localement *f_a, noté $f_a \preccurlyeq_\Delta M_a$, si et seulement si pour toute configuration $x \in \Lambda^S$,*

- *et pour toute configuration $x' \in \Lambda'^{S_a}$ telle que $\phi_a(x') = x_a$,*
- *et pour toute configuration d'entrée $i' : \Lambda'^{I_a}$ telle que pour tout b dans S, si $a \neq b$ alors $\phi_{b,a}(i' \circ \mathtt{w}^{-1}_{b,a}) = x_b$*

on a

$$\phi_a(M_{a\Delta}(x' \cdot i')) = f_a(x).$$

Lorsque les modules $\{M_s | s \in S\}$ sont branchés en un seul réseau, il est primordial que les modules de mise à jour Δ considérés ne parasitent pas le fonctionnement du réseau global. Pour cela, nous posons la condition que tous mode de mise à jour utilisé sur un sous-réseau M_s doit mettre à jour au moins tout les nœuds qui possèdent des entrées à la première étape de mise à jour. Lorsque cela est fait, il n'est pas possible qu'un module change son encodage avant qu'un autre module ait pu acquérir la valeur de son encodage à temps.

Définition 62 (Mode de mise à jour entrée-prioritaire)**.** *Soit Δ un mode de mise à jour pour un module M. Le mode de mise à jour Δ est dit entrée-prioritaire si et seulement si, pour tout $k > 1$, $s \in \Delta_k$ implique que s n'est influencé par aucune entrée.*

Nous pouvons maintenant introduire notre définition spécialisée de simulation entre réseaux d'automates.

Définition 63 (Simulation entre réseaux d'automates)**.** *Soient F, F' deux réseaux d'automates, définis sur des ensembles d'états Λ et Λ'. Le réseau F' simule le réseau F, noté $F \preccurlyeq F'$, si et seulement s'il existe une fonction $\Phi : \Lambda'^{S'} \to \Lambda^S \cup \{\bullet\}$ dite encodage global telle que, pour toute mise à jour $\delta \subset S$, il existe un mode de mise à jour fini Δ' tel que pour toutes configurations x de F et x' de F' telles que $\Phi(x') = x$,*

$$\Phi(F'_{\Delta'}(x')) = F_\delta(x).$$

Depuis un réseau F, et en définissant l'ensemble de modules $\{M_s | s \in S\}$, de branchements $\{\mathtt{w}_{a,b} | a, b \in S \,|a \neq b\}$ ainsi que les encodages $\{\phi_s | s \in S\}$ et $\{\phi_{a,b} | a, b \in S | a \neq b\}$, nous proposons le théorème suivant.

Théorème 5. *Soit $\mathtt{w}$ le branchement défini comme la concaténation des branchements $\{\mathtt{w}_{a,b} | a, b \in S \,|a \neq b\}$. Si, pour tout $s \in S$, le module M_s simule localement la fonction f_s, et si $\mathtt{w}$ est une fonction de branchement proprement définie qui associe un sommet à tous les sommets dans l'union des modules $\{M_s | s \in S\}$, alors*

$$F \preccurlyeq \circlearrowleft_{\mathtt{w}} \left(\bigcup_{s \in S} M_s \right).$$

Démonstration. Nous allons d'abord démontrer que le comportement du réseau F' peut être décrit comme la somme des comportements des modules $\{M_s | s \in S\}$ pris séparément. Puis nous utiliserons la propriété de simulation locale de chaque module pour extrapoler le comportement global de simulation du réseau F'.

Soit $F' = \circlearrowleft_{\mathtt{w}} (\bigcup_{s \in S} M_s)$. Par l'énoncé de notre théorème, F' est un réseau d'automates. Soit T son ensemble d'automates, défini comme l'union des ensembles d'automates des modules $\{M_s | s \in S\}$. Pour tout a dans S, tout mode de mise à jour entrée-prioritaire Δ pour le module M_a, pour tout mode de mise à jour Δ' sur $T \setminus M_a$, et pour toute configuration x, nous énonçons que

$$F'_{\Delta \cup \Delta'}(x)|_{S_a} = M_{a\Delta}(x|_{S_a} \cdot (x \circ \mathtt{w}|_{I_a})). \tag{5.7}$$

Pour voir que cette équation est vraie, considérons qu'à la première étape de l'exécution, le branchement $\mathtt{w}$ implique que pour tout s dans S_a, et toute configuration x, $F'(x)_s = (\bigcup_{s\in S} M_s)(x\cdot(x\circ\mathtt{w}))$. En particulier, cela signifie que $F'(x)_s = M_a(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))$.

Considérons A le sous-ensemble des automates de S_a qui sont influencés par au moins une entrée, et $B = S_a \setminus A$ l'ensemble des automates qui ne sont influencés par aucune entrée. Par la nature entrée-prioritaire de Δ, nous savons que si $s \in \Delta_k$ et $k > 1$, alors $s \in B$.

Considérons d'abord la mise à jour des automates dans A. Nous posons $\delta = \Delta_1$ et $\delta' = \Delta'_1$. Nous pouvons clairement déduire de la proposition précédente que

$$F'_{\delta\cup\delta'}(x)|_A = M_{a\delta}(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))|_A.$$

De plus, il n'existe aucun s dans A tel que s est également dans Δ_k pour $k > 1$. Nous pouvons donc conclure, puisqu'aucune fonction locale de A n'est mise à jour dans le reste de l'exécution, que $F'_{\delta\cup\delta'}(x)|_A = F'_{\Delta\cup\Delta'}(x)|_A$, et que $M_{a\delta}(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))|_A = M_{a\Delta}(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))|_A$. En conclusion de notre travail sur l'ensemble A,

$$F'_{\Delta\cup\Delta'}(x)|_A = M_{a\Delta}(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))|_A.$$

Nous considérons maintenant la mise à jour de l'ensemble d'automates B. Pour tout $s \in B$, nous savons que $F'(x)_s = M_a(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))_s$. Par la définition de B, l'automate s n'est influencé par aucune entrée. Cela signifie que, pour toutes mises à jour $\delta \subseteq S_a$ et $\delta' \subseteq T\setminus S_a$ et pour toute configuration d'entrée $i \in \Lambda'^{I_a}$, on a $F'_{\delta\cup\delta'}(x)|_B = M_{a\delta}(x|_{S_a}\cdot i)|_B$. Par un simple raisonnement récursif cela implique que $F'_{\Delta\cup\Delta'}(x)|_B = M_{a\Delta}(x|_{S_a}\cdot i)|_B$.

En rassemblant nos raisonnements concernant les ensembles A et B, nous obtenons que $F'_{\Delta\cup\Delta'}(x) = M_{a\Delta}(x|_{S_a}\cdot(x\circ\mathtt{w}|_{I_a}))|_A\cdot M_{a\Delta}(x|_{S_a}\cdot i)|_B$, pour toute configuration d'entrée i. Cela signifie qu'en prenant en particulier $i = x\circ\mathtt{w}|_{I_a}$, on vérifie la proposition 5.7.

Nous considérons à présent la fonction $\Phi : \Lambda'^T \to \Lambda^S \cup \{\bullet\}$ qui vérifie que, pour tout $x' \in \Lambda'^T$, $\Phi(x') = \bullet$ s'il existe $a \in S$ tel que $\phi_a(x|_{S_a}) = \bullet$, et $\Phi(x')_a = \phi_a(x|_{S_a})$ dans le cas contraire. Soient x et x' tels que $\Phi(x') = x$ et $x \neq \bullet$. Soit $\delta \subseteq S$ une mise à jour de F. Pour tout a dans δ, nous définissons Δ'_a un mode de mise à jour entrée-prioritaire avec lequel M_a simule localement la fonction f_a, ce qui est toujours possible par hypothèse du théorème. Nous définissons le mode de mise à jour Δ' sur T tel que $\Delta' = \bigcup\{\Delta'_a | a \in \delta\}$.

Nous allons maintenant prouver que $\Phi(F'_{\Delta'}(x')) = F_\delta(x)$. Premièrement, nous observons que

$$F'_{\Delta'}(x') = \mathrm{concat}(F'_{\Delta'}(x')|_{S_a} | a \in S)$$

ce qui se développe en

$$F'_{\Delta'}(x') = \mathrm{concat}(F'_{\Delta'}(x')|_{S_a} | a \in \delta)\cdot\mathrm{concat}(x'|_{S_a} | a \in S\setminus\delta)$$

ce dont nous déduisons que

$$\begin{aligned} & F'_{\Delta'}(x') = \operatorname{concat}(F'_{\Delta'_a \cup \bigcup_{b\in\delta, b\neq a}\Delta'_b}(x')|_{S_a} | a \in \delta) \cdot \operatorname{concat}(x'|_{S_a} | a \in S\setminus\delta) \\ \Longrightarrow\ & F'_{\Delta'}(x') = \operatorname*{concat}_{a\in\delta}(M_{a\Delta'_a}(x'|_{S_a} \cdot (x \circ \mathtt{w}|_{I_a}))) \cdot \operatorname*{concat}_{a\in S\setminus\delta}(x'|_{S_a}) \qquad \text{(Équation 5.7)} \end{aligned}$$

Comme le résultat de l'exécution du module M_a est une configuration sur S_a, nous pouvons déduire la valeur de $\Phi(F'_{\Delta'}(x))$ suivante :

$$\Phi(F'_{\Delta'}(x'))_a = \begin{cases} \phi_a(M_{a\Delta_a}(x'|_{S_a} \cdot (x \circ \mathtt{w}|_{I_a}))) & \text{si } a \in \delta \\ \phi_a(x'|_{S_a}) & \text{si } a \in S\setminus\delta \end{cases}.$$

Par définition de x et x', nous savons que $\phi_a(x'|_{S_a}) = x_a$ et que $\phi_{b,a}(x' \circ \mathtt{w}_{b,a} \circ \mathtt{w}_{b,a}^{-1}) = \phi_{b,a}(x'|_{\operatorname{codom}(\mathtt{w}_{b,a})}) = \phi_b(x'|_{S_b}) = x_b$, par la définition de $\phi_{b,a}$. De cela, nous appliquons la définition de la simulation locale et obtenons

$$\Phi(F'_{\Delta'}(x'))_a = \begin{cases} f_a(\Phi(x')) & \text{si } a \in \delta \\ \phi_a(x'|_{S_a}) & \text{si } a \in S\setminus\delta \end{cases} = \begin{cases} f_a(\Phi(x)) & \text{si } a \in \delta \\ \Phi(x')_a & \text{si } a \in S\setminus\delta \end{cases}.$$

De plus, par la définition de la mise à jour de F, on obtient que

$$F_\delta(x)_a = \begin{cases} f_a(x) & \text{si } a \in \delta \\ x_a & \text{si } a \in S\setminus\delta \end{cases}.$$

Enfin, par l'hypothèse que $x = \Phi(x')$, nous obtenons

$$F_\delta(x)_a = \begin{cases} f_a(\Phi(x')) & \text{si } a \in \delta \\ \Phi(x')_a & \text{si } a \in S\setminus\delta \end{cases},$$

ce qui implique le résultat.

□

Ce théorème permet la généralisation de résultats simples de simulation, et est présenté ici principalement pour sa qualité d'illustration du formalisme des modules en tant que modèle de décomposition et recomposition des réseaux d'automates. Dans cet esprit, nous présentons trois résultats simples dérivés de l'application de notre théorème. Il s'agit de l'universalité intrinsèque des réseaux d'automates booléens, lire la capacité des réseaux d'automates booléens de simuler l'ensemble des réseaux d'automates, ainsi que la simulation de tout réseau d'automate booléen par un réseau d'automates booléen disjonctif, ainsi que par un réseau d'automates booléen localement monotone.

Corollaire 1. *Soit F un réseau d'automates défini sur l'ensemble d'états Λ. Il existe un réseau d'automates booléens F' avec au plus* $\log(|\Lambda|)$ *fois plus d'automates tel que $F \preccurlyeq F'$.*

Démonstration. Soit F un réseau défini sur l'ensemble d'états Λ. Dans cette preuve, nous construisons pour chaque automate a dans F un module booléen M_a qui encode l'état de a dans Λ à l'aide de $\log(|\Lambda|)$ automates avec état dans $\mathbb{B}$. Chacun de ces automates calcule la fonction locale de f_a et dérive le bit d'information qui l'intéresse, de façon à ce que l'ensemble des automates de M_a encodent dans leur ensemble l'état de a dans F.

Le module M_a est défini sur l'ensemble d'automates $S_a = \{a_0, a_1, \dots, a_{\log(|\Lambda|)-1}\}$. Pour tout automate b de F différent de a, on définit l'ensemble d'entrées $I_{b,a} = \{\alpha_{b_0,a}, \alpha_{b_1,a}, \dots, \alpha_{b_{\log(|\Lambda|)-1},a}\}$. L'ensemble des entrées de M_a est l'ensemble $I_a = \bigcup_b I_{b,a}$.

Soit $\mathrm{bit}_k(\lambda)$ la fonction qui, pour tout état λ, retourne la valeur du bit numéro k dans le nombre booléen de taille $\log(|\Lambda|)$ qui encode la valeur de λ. Cette fonction est canonique en supposant un ordre arbitraire sur les éléments de Λ, qui est un ensemble fini.

Pour a_k un automate de M_a, x une configuration sur S_a, on définit ϕ_a l'encodage qui à x associe la valeur dans Λ qui est cohérente avec la fonction bit, et renvoie $\bullet$ si aucune valeur n'est cohérente. En particulier, pour $\lambda \in \Lambda$, si $x_{a_k} = \mathrm{bit}_k(\lambda)$, alors $\phi_a(x) = \lambda$.

On définit le branchement $\mathtt{w}_{b,a}$ comme le branchement avec domaine $I_{b,a}$ et co-domaine S_b tel que $\mathtt{w}_{b,a}(\alpha_{b_k,a}) = b_k$. Le branchement $\mathtt{w}_{b,a}$ est injectif.

Pour x une configuration sur S_b, on définit l'encodage $\phi_{b,a}$ comme la fonction qui vérifie $\phi_{b,a}(x) = \lambda$ si pour tout k, $x_{b_k} = bit_k(\lambda)$, et $\phi_{b,a}(x) = \bullet$ si aucun λ n'existe.

Soient x et i des configurations d'automates et d'entrées du module M_a. Depuis ces éléments, on peut reconstruire une configuration y sur le réseau F. Pour ce faire, supposons qu'il existe un λ_a tel que $\phi_a(x) = \lambda_a$ et que, pour tout $b \neq a$, il existe un λ_b tel que $\phi_{b,a}(i) = \lambda_b$. La recomposition de cette configuration, que nous notons $\mathrm{recomp}(x,i)$ vérifie $\mathrm{recomp}(x,i)_s = \lambda_s$ dans ce cas. Dans le cas contraire, on évalue $\mathrm{recomp}(x,i)_s = \lambda_\bullet$, où $\lambda_\bullet$ est une valeur arbitraire dans Λ.

Enfin, la fonction locale f_{a_k} est définie par

$$f_{a_k}(x \circ i) = \mathrm{bit}_k(f_a(\mathrm{recomp}(x,i)).$$

Pour simuler localement la fonction f_a, le module M_a opère avec la mise à jour parallèle.

Par la structure de notre construction, il est trivial de vérifier que le module M_a simule localement la fonction f_a. Nous déduisons le résultat par application du Théorème 5. □

Corollaire 2. *Soit F un réseau d'automates booléens. Il existe un réseau d'automates booléens F' tel que chaque fonction locale de F' est une clause disjonctive, et $F \preccurlyeq F'$.*

Démonstration. Soit F un réseau d'automates booléens. Dans cette preuve, nous construisons, pour chaque automate a de F, un module M_a exclusivement composé de fonctions booléennes qui peuvent être écrites comme une clause disjonctive, qui simule f_a localement.

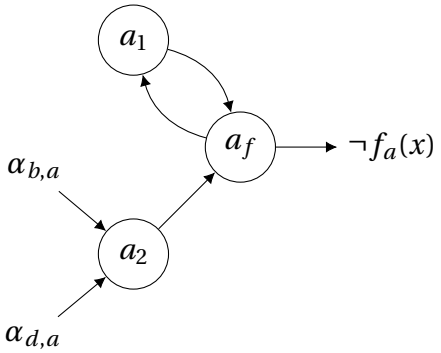

FIGURE 5.11 – Graphe d'interaction du module localement disjonctif qui simule localement la fonction $f_a(x) = x_a \wedge (\neg x_b \vee x_d)$. Nous nommons $C_{a,1} = x_a$ et $C_{a,2} = \neg x_b \vee x_d$ les clauses de la fonction f_a. Nous observons $f_{a_1} = \neg x_{a_f}$, $f_{a_2} = \neg \alpha_{b,a} \vee \alpha_{d,a}$ et $f_{a_f} = \neg x_{a_1} \vee \neg x_{a_2}$.

Premièrement, nous faisons l'observation que toute fonction booléenne peut être représentée par une formule booléenne sous la forme normale disjonctive. Cela signifie que la fonction f_a peut être réalisée par une formule $\mathbf{g}_a = \bigvee_{k=1}^{n_a} C_{a,k}$, pour n_a le nombre de clauses de la formule $\mathbf{g}_a$, et où chaque $C_{a,k}$ est une clause conjonctive.

Notre construction repose sur la composition d'un module M_a qui possède un nœud dénoté a_k pour chaque clause $C_{a,k}$, ainsi qu'un nœud dénoté a_f. Le module définit également un ensemble d'entrées $\alpha_{b,a}$ pour chaque automate b dans F différent de a. L'encodage ϕ_a est défini tel que $\phi_a(x) = x_{a_f}$, le branchement $\mathtt{w}_{b,a}$ a pour domaine $\{\alpha_{b,a}\}$ et co-domaine $\{x_{b_f}\}$ ce qui implique $\mathtt{w}_{b,a}(\alpha_{b,a}) = x_{b_f}$, et l'encodage $\phi_{b,a}(x|_{\{b_f\}}) = x_{b_f}$.

La fonction locale de a_k calcule la négation de la valeur de $C_{a,k}$. Pour ce faire, le littéral associé à a dans $\mathbf{g}_a$ est substitué par x_{a_f} et, pour tout automate b différent de a, le littéral associé à b dans $\mathbf{g}_a$ est substitué par $i_{\alpha_{b,a}}$. Comme $C_{a,k}$ est une clause conjonctive, sa négation est une clause disjonctive.

La fonction locale de a_f calcule la valeur de $\mathbf{g}_a$ par le biais du calcul $\bigvee_{k=1}^{n_a} C_{a,k}$, qui est opéré en subsituant la valeur de $C_{a,k}$ par la négation $\neg x_{a_k}$.

Pour simuler localement l'automate a, le module M_a opère sur le mode de mise à jour $\Delta_a = (\{a_1, a_2, \ldots, a_{n_a}\}, \{a_f\})$, qui est un mode de mise à jour entrée-prioritaire. Par notre construction, il est facile de vérifier que le module M_a simule localement la fonction f_a. Nous déduisons le résultat par application du Théorème 5. □

Nous illustrons un exemple de module constitué de fonctions disjonctives qui simule localement la fonction $f_a(x) = x_a \wedge (\neg x_b \vee x_d)$ en figure 5.11.

Nous définissons qu'une fonction locale f d'un réseau d'automates booléens est localement monotone si, pour toute paire de configurations x, y telles que $x \leq y$, $f(x) \leq f(y)$. Ici, l'opération de comparaison utilisée est la comparaison binaire bit à bit : la configuration 00 est inférieure à 10, 01 ou 11, mais 10 et 01 ne sont pas

comparables.

Corollaire 3. *Soit F un réseau d'automates booléens. Il existe un réseau d'automates booléens F' tel que chaque fonction locale de F' est une fonction localement monotone, et $F \preccurlyeq F'$.*

Démonstration. Soit F un réseau d'automates booléens. Dans cette preuve, nous construisons, pour chaque automate a de F, un module M_a, exclusivement composé de fonctions localement monotones, qui simule f_a.

Pour toute paire d'entrées x, y, une fonction localement monotone f vérifie $x \leq y \implies f(x) \leq f(y)$. Dans le cas des fonctions locales d'un réseau d'automates, nous définissons qu'une configuration x est inférieure ou égale à une configuration y définie sur les mêmes ensembles si et seulement si $x_s \leq y_s$ pour tout automate s.

Soit a un automate de F. Nous définissons le module M_a composé de deux automates a_0 et a_1, et un ensemble d'entrées I_a qui définit deux entrées $\alpha_{b,a,0}, \alpha_{b,a,1}$ pour chaque automate b différent de a. L'encodage ϕ_a vérifie $\phi_a(x) = 1$ si et seulement si $x_{a_0} = 0$ et $x_{a_1} = 1$, $\phi_a(x) = 0$ si et seulement si $x_{a_0} = 1$ et $x_{a_1} = 0$, et $\phi_a(x) = \bullet$ dans tous les autres cas. Le branchement injectif $\mathtt{w}_{b,a}$ a pour domaine $\{\alpha_{b,a,0}, \alpha_{b,a,1}\}$ et codomaine $\{b_0, b_1\}$ tel que $\mathtt{w}_{b,a}(\alpha_{b,a,k}) = b_k$ pour $k \in \mathbb{B}$. Enfin, l'encodage $\phi_{b,a}(x) = \phi_b(x)$.

Pour tous x et i tels que les encodages sont proprement définis, soit $\mathrm{recomp}(x, i)$ la configuration de F telle que $\mathrm{recomp}(x, i)_a = \phi_a(x)$ et $\mathrm{recomp}(x, i)_b = \phi_b(i \circ \mathtt{w}_{b,a}^{-1})$, pour tout $b \neq a$. Dans le cas où $\mathrm{recomp}(x, i)$ est défini, les fonctions locales de a_0 et a_1 vérifient $f_{a_1}(x \circ i) = f_a(\mathrm{recomp}(x, i))$ et $f_{a_0}(x \circ i) = \neg f_{a_1}(x \circ i)$.

Cet encodage n'est pas défini si et seulement s'il existe au moins une paire (x_{a_0}, x_{a_1}) ou $(\alpha_{b,a,0}, \alpha_{b,a,1})$ dont la valeur des éléments est identique. S'il existe une telle paire de valeur 0, on dit que (x, i) encode une paire 0. S'il existe une telle paire de valeur 1, on dit que (x, i) encode une paire 1. Dans le cas où (x, i) encode une paire 0 ou une paire 1, nous définissons l'évaluation des fonctions locales f_{a_0} et f_{a_1} comme

$$f_{a_0}(x \circ i) = f_{a_1}(x \circ i) = \begin{cases} 0 & \text{si } (x, i) \text{ n'encode pas de paire 1} \\ 1 & \text{si } (x, i) \text{ n'encode pas de paire 0} \\ 0 & \text{si } (x, i) \text{ encode au moins une paire 0 et une paire 1} \end{cases}.$$

Soient x et y deux configurations du module $\bigcup_{s \in S} M_s$. Si $x = y$ alors $f_{a_0}(x) = f_{a_0}(y)$ et $f_{a_1}(x) = f_{a_1}(y)$. Supposons que $x \leq y$ et $x \neq y$. Nous construisons (x', i') et (y', j') les paires de configurations d'automates et d'entrées équivalentes à x et y.

Supposons que (x', i') encode une paire 0, et pas de paire 1. Alors $f_{a_0}(x) = f_{a_1}(x) = 0$ ce qui est toujours inférieur ou égal à $f_{a_0}(y)$ et $f_{a_1}(y)$.

Supposons que (x', i') encode une paire 1 et pas de paire 0. Alors (y', j') encode au moins les même paires 1 et ne peut pas encoder de paire 0, ce qui implique que $f_{a_0}(y) = f_{a_1}(y) = 1$, ce qui est toujours supérieur ou égal à $f_{a_0}(x)$ et $f_{a_1}(x)$.

Supposons que (x', i') n'encode aucune paire 0 ni aucune paire 1. Cela signifie que (y', j') n'encode pas de paire 0 et encode au moins une paire 1, ce qui signifique que $f_{a_0}(y) = f_{a_1}(y) = 1$, ce qui est toujours supérieur ou égal à $f_{a_0}(x)$ et $f_{a_1}(x)$.

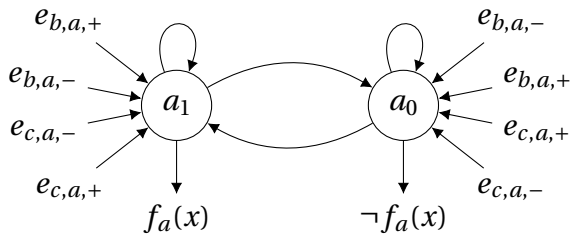

FIGURE 5.12 – Graphe d'interaction du module composé de fonctions localement monotones qui simule localement la fonction $f_a(x) = x_a \wedge (\neg x_b \vee x_c)$.

Supposons enfin que (x', i') encode au moins une paire 0 et au moins une paire 1. Cela signifie que $f_{a_0}(x) = f_{a_1}(x) = 0$, ce qui est toujours inférieur ou égal à $f_{a_0}(y)$ et $f_{a_1}(y)$.

En conclusion, les fonctions locales du module M_a sont localement monotones. Notre construction permet de vérifier que le module M_a simule localement la fonction f_a grâce au mode de mise à jour parallèle. Nous déduisons le résultat par application du théorème 5. □

Nous illustrons un exemple de module constitué de fonctions localement monotones qui simule localement la fonction $f_a(x) = x_a \wedge (\neg x_b \vee x_d)$ en figure 5.12.

5.4 Conclusion

La caractérisation de la dynamique des réseaux d'automates est une tâche difficile. Dans ce chapitre, nous avons introduit le formalisme de modules, qui sont une généralisation des réseaux d'automates dont la dynamique ne peut être qu'encore plus complexe. Il est cependant important de ne pas percevoir les modules comme une simple généralisation des réseaux d'automates ; mais plutôt comme une opportunité de caractériser le comportement de ces derniers lorsque l'on considère non pas un réseau dans son ensemble, mais un morceau particulier qui est sujet à l'influence du reste. Dans ce cadre, les entrées d'un module permettent de simplifier son étude.

Malgré cette approche, l'étude des modules reste un sujet complexe, et il semble que la continuation de cette étude repose avant tout sur une restriction à des familles simples de modules. Un exemple d'une telle restriction est développée au chapitre suivant, qui est dédié à l'étude des modules acycliques.