Acyclicité

Les réseaux d'automates acycliques

Les réseaux d'automates acycliques sont une famille de réseaux d'automates parmi les plus simples, et consécutivement une des rares familles de réseaux d'automates dont nous comprenons entièrement le comportement. L'acyclicité ici est définie dans le contexte du graphe d'interaction du réseau ; un réseau d'automate est dit acyclique si son réseau d'interaction ne dispose d'aucun cycle.

Définition 64 (Réseau d'automates acyclique)**.** *Soit F un réseau d'automates. Le réseau F est dit* acyclique *si et seulement si son graphe d'interaction est un graphe acyclique.*

Exemple 44. *Soit F le réseau d'automates acyclique défini sur les automates $S = \{a, b, c\}$ et qui admet les fonctions locales suivantes :*

$$f_a(x) = 1$$
$$f_b(x) = \neg x_a$$
$$f_c(x) = x_a \vee x_b$$

Le graphe d'interaction du réseau d'automates acyclique F est représenté en figure 6.1, et le graphe de sa dynamique totale en figure 6.2.

Cela signifie plus formellement qu'il n'existe aucun chemin partant d'un automate s et retournant sur l'automate s dans le graphe d'interaction du réseau d'automates. Cela signifie qu'il existera toujours au moins un automate qui n'est influencé par aucun autre automate ; dans le cas contraire, on pourrait remonter l'influence dans le graphe sans fin, ce qui décrit naturellement un cycle. Ces automates sans aucune influence ont une fonction locale constante, puisqu'elle ne peut pas être influencée par aucun facteur, y compris la valeur de l'automate où elle est installée. Le reste du réseau prenant son influence toujours depuis ce sous-ensemble constant, et ne disposant d'aucun cycle, l'entièreté du réseau est destinée à se figer en un point fixe. Ce théorème est également évoqué en section 4.6.

Théorème 6 (Robert 1980)**.** *Soit F un réseau acyclique. La dynamique totale de F n'a qu'un seul attracteur qui est un point fixe.*

Démonstration. Soit F un réseau d'automates acyclique. On construit la séquence $(S_1, S_2, \ldots, S_k)$ telle que S_1 est l'ensemble des automates de F qui ne sont influencés

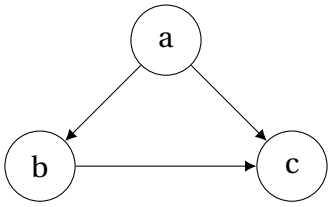

FIGURE 6.1 – Graphe d'interaction du réseau d'automates développé dans l'exemple 44.

par aucun autre automate, S_2 est l'ensemble des automates qui ne sont influencés que par des automates de S_1, et en général S_i décrit les automates qui ne sont influencés que par des sommets dans les ensembles qui le précèdent dans la séquence. En pratique, k décrit la longueur du plus long chemin dans le graphe d'interaction de F. Par définition, l'ensemble des éléments de cette séquence produit une partition de l'ensemble S, et aucun élément de cette séquence n'est vide.

On considère le graphe de la dynamique totale de F. Par définition, toute fonction locale d'un automate de S_1 est une fonction constante. Par conséquent, mettre à jour n'importe quel automate dans S_1 fige cet automate dans une valeur qui ne change plus. Plus formellement, toute configuration x qui vérifie $x_s \neq f_s(x)$ pour s dans S_1 ne fait pas partie d'un attracteur. Par conséquent, l'ensemble des attracteurs de la dynamique doit être inclus dans le sous-graphe dont les configurations vérifient $x_s = f_s(x)$ pour tout s dans S_1.

Nous considérons ce sous-graphe. Dans ce sous-graphe et par définition, toute fonction locale d'un automate de S_2 est une fonction qui ne dépend que d'automates dans S_1. Par définition de nouveau, le sous-graphe de la dynamique ne considère qu'une valeur constante pour chaque automate dans S_1. Par conséquent, le même raisonnement s'applique, et les attracteurs du réseau ne peuvent être que dans le sous-graphe de ce sous-graphe qui ne contient que les configurations qui vérifient $x_s = f_s(x)$ pour $s \in S_2$.

On applique ce raisonnement pour l'ensemble de la séquence $(S_1, S_2, \ldots, S_k)$, et obtenons un sous-graphe de la dynamique qui contient nécessairement l'ensemble des attracteurs de F. De plus, l'ensemble de ce raisonnement a selectionné les configurations qui figent l'ensemble des automates du réseau à une valeur constante. Par conséquent, le sous-graphe que nous obtenons ne possède qu'une seule configuration, qui est nécessairement le seul attracteur du graphe, et donc un point fixe. □

Ce résultat connu illustre parfaitement que la complexité des réseaux d'automates réside dans les cycles de son graphe d'interaction.

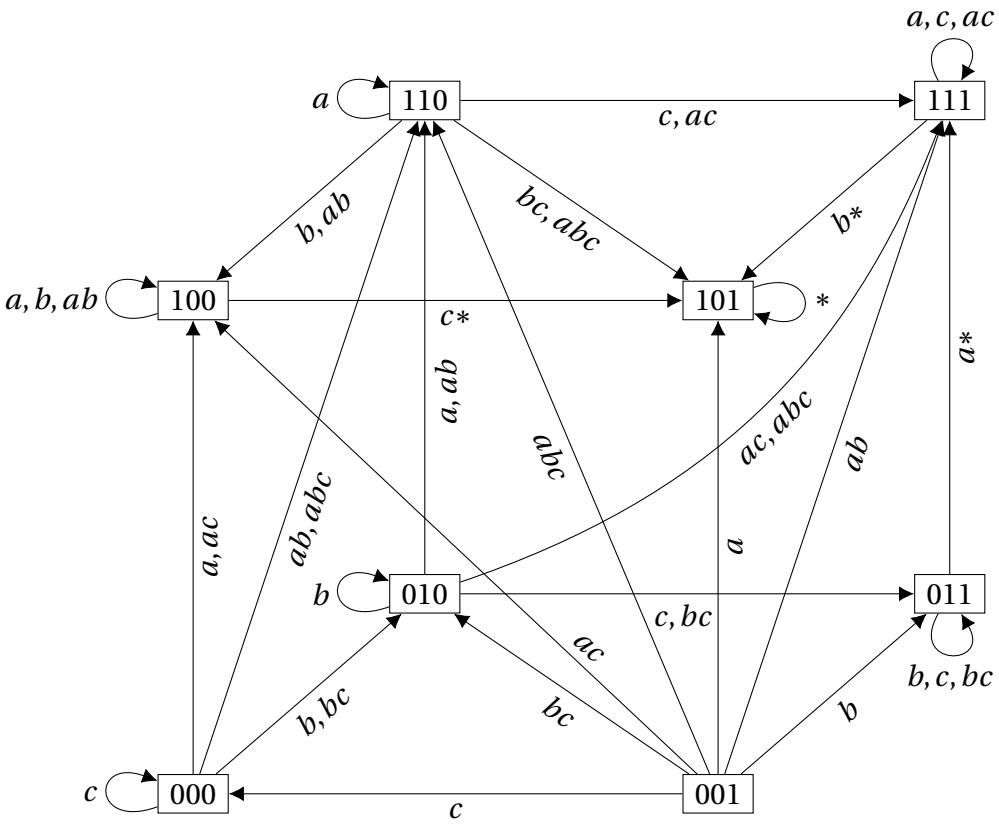

FIGURE 6.2 – Graphe de la dynamique totale du réseau d'automates développé dans l'exemple 44. L'étiquette ab représente la mise à jour $\delta = \{a, b\}$. L'étiquette $a*$ représente l'ensemble des mises à jour qui contiennent a, et $*$ l'ensemble de toutes les mises à jour possibles. On remarque que l'ensemble des attracteurs de cette dynamique se limite au point fixe 101.

6.2 Modules acycliques

Les modules sont une généralisation qui complexifie le comportement des réseaux étudiés. Nous développons dans cette section la restriction des modules sur des réseaux acycliques. Les réseaux d'automates acycliques étant des objets dont la dynamique est simple à comprendre, leur version modulaire dispose d'un comportement dynamique qui semble accessible à la caractérisation.

Définition 65 (Module acyclique)**.** *Soit M un module. Le réseau M est* acyclique *si et seulement si son graphe d'interaction est un graphe acyclique.*

Exemple 45. *Soit M le module acyclique défini sur les automates $S = \{a, b, c\}$, les entrées*

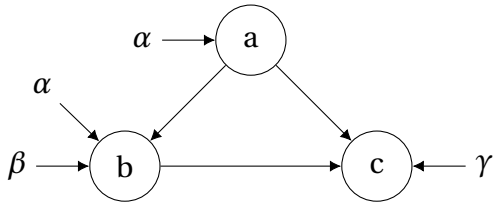

FIGURE 6.3 – Graphe d'interaction du module acyclique développé dans l'exemple 45.

$I = \{\alpha, \beta, \gamma\}$ et qui admet les fonctions locales suivantes :

$$f_a(x \cdot i) = i_\alpha$$
$$f_b(x \cdot i) = x_a \vee i_\beta \vee \neg i_\alpha$$
$$f_c(x \cdot i) = \neg x_b \wedge x_a \wedge \neg i_\gamma.$$

Le graphe d'interaction de M est représenté en figure 6.3.

Les modules acycliques diffèrent des réseaux d'automates acycliques car, en général, leurs entrées les empêchent de converger. Lorsque la valeur d'une entrée change dans une exécution, cette entrée provoque une cascade potentielle de réévaluations de fonctions locales le long du module. L'absence de cycle empêche le stockage de toute information ; et dès que les entrées sont figées sur une certaine configuration d'entrée, le réseau entier reprend le fonctionnement d'un réseau d'automates acyclique, et atteint un point fixe.

L'étude des modules acycliques nous semble justifiée par la simple réalisation que tout réseau d'automates, et même tout module, peut être exprimé comme le branchement récursif d'un module acyclique. Ce dernier se construit facilement en trouvant un coupe-cycle de sommets (ou Feedback Vertex Set) du graphe d'interaction du réseau, c'est-à-dire un ensemble d'automates qui, lorsque l'on remplace leur influence par des entrées, confèrent un réseau d'interaction acyclique. Afin de conserver au mieux la structure du réseau initial, il est préférable de trouver un Minimal Feedback Vertex Set du graphe d'interaction, ce qui est un problème NP-complet (KARP 1972).

Propriété 14. *Soit M un module. Il existe un module acyclique M' et un branchement récursif $\mathtt{w}$ tel que*

$$\circlearrowleft_{\mathtt{w}} M' = M.$$

Démonstration. Soit M un module. Nous considérons une solution du problème Minimal Feedback Vertex Set du graphe d'interaction de M que nous dénotons comme l'ensemble d'arêtes orientées $A \subseteq S \times S$. Par définition, retirer ces arêtes au graphe d'interaction de M nous donne un graphe acyclique.

On considère l'ensemble des automates $S' \subseteq S$, défini tel que $s' \in S'$ si et seulement s'il existe s' tel que $(s', s) \in A$. On définit un ensemble de variables $I' = \{\alpha'_s \mid s' \in S'\}$ tel

que $I \cap I' = \emptyset$.

Soit M' le module basé sur l'ensemble d'automates S et l'ensemble d'entrées $I \cup I'$, et qui, pour tout $s \in S$, définit la fonction locale f'_s comme la copie de la fonction f_s dans laquelle chaque variable x'_s est remplacée par $i_{\alpha_{s'}}$, pour tout s' dans S'. Soit $\mathtt{w}$ la fonction de branchement avec domaine I' qui définit $\mathtt{w}(\alpha_{s'}) = s'$. Ces constructions impliquent $\circlearrowleft_{\mathtt{w}} M' = M$. Par définition, le graphe d'interaction de M' est acyclique. □

Il semble clair qu'une caractérisation du comportement des modules acycliques permet une compréhension plus approfondie du fonctionnement des modules en général, et en particulier des réseaux d'automates.

6.2.1 Fonctions de sortie

Le comportement d'un automate d'un module acyclique peut être caractérisé comme une fonction de l'historique récent des entrées données au réseau. Ainsi, lorsqu'un module acyclique est mis à jour pour assez longtemps à partir d'une configuration initiale et avec une séquence de configurations d'entrée, l'influence de la configuration initiale sur l'évaluation des automates devient nulle. Cela se comprend clairement lorsque l'on se rappelle qu'un module acyclique ne dispose d'aucun mécanisme pour retenir de l'information. Nous caractérisons cette propriété par l'association d'une fonction de sortie à tout automate.

Ici, nous faisons une différence entre séquence d'entrée (composée par des configurations d'entrées mises dans un certain ordre) et historique d'entrée (composé de configurations d'entrées organisées par ancienneté). Si l'ensemble des séquences d'entrées d'un module est $(\Lambda^I)^*$, les historiques d'entrées de longueur n se définissent dans l'ensemble $\Lambda^{I\times\{1,\dots,n\}}$.

Définition 66. *Pour tout séquence d'entrée J et tout entier n tel que $|J| \geq n$, on définit l'historique de taille n extrait de J comme l'historique noté $H_n(J)$, qui pour tout $\alpha \in I$ et $k \in \{1,\dots,n\}$ définit*

$$H_n(J)_{(\alpha,k)} = (J_{|J|-k+1})_\alpha.$$

Une fonction de sortie est définie sur $O : \Lambda^{I\times\{1,\dots,D\}} \to \Lambda$, pour D un entier suffisamment grand. L'historique définit dans $\Lambda^{I\times\{1,\dots,D\}}$ passé en entrée d'une fonction de sortie contient les configurations d'entrée récentes. Ainsi, pour α une variable de I, l'index $(\alpha,1)$, que nous notons α_1, de ce vecteur, correspond à la valeur de l'entrée α à la mise à jour actuelle. L'index α_2 représente la valeur de l'entrée α une mise à jour dans le passé, et en général α_k représente la valeur de l'entrée α $k-1$ mises à jour dans le passé. Une fonction de sortie est une fonction qui, pour tout historique de longueur D, retourne une valeur dans Λ.

Pour O une fonction de sortie donnée, la valeur k maximale telle qu'il existe $\alpha \in I$ qui vérifie que α_k a une influence sur l'évaluation de O s'appelle le délai de O. Nous notons ce délai par la lettre $\mathtt{d}$, et il correspond à la profondeur minimale de l'historique nécessaire pour calculer O. Ainsi l'évaluation de O sur une séquence d'entrée consiste

à en extraire un historique de longueur $\mathtt{d}$, et d'évaluer O sur cet historique. On note que si $\mathtt{d} < D$, alors un historique de taille $\mathtt{d}$ ne contient pas assez d'information pour permettre un appel correct de la fonction O. Cependant, puisque par définition aucune variable qui manque à cet historique n'a d'influence sur le calcul de O, ne prenons le raccourcis de noter $O(H_{\mathtt{d}}(J))$ comme une évaluation correcte de O.

Définition 67 (Évaluation d'une fonction de sortie)**.** *Soit $O : \Lambda^{I\times\{1,\dots,D\}} \to \Lambda$ une fonction de sortie avec délai $\mathtt{d}$, et J une séquence d'entrée de taille $k \geq \mathtt{d}$. On définit l'*évaluation *de O sur J, notée $O(J)$, comme*

$$O(J) = O(H_{\mathtt{d}}(J)).$$

Nous définissons maintenant l'incrémentation d'une fonction de sortie, qui consiste à déplacer la fenêtre des variables considérées par la fonction dans le temps, ce qui a l'effet immédiat d'augmenter son délai de 1. Cette notion simple est centrale dans les preuves qui reposent sur le calcul de ces fonctions de sortie dans le contexte de la prédiction de la valeur d'automates dans un module acyclique.

Définition 68 (Incrémentation de fonction de sortie)**.** *Pour O une fonction de sortie de délai $\mathtt{d}$, nous définissons l'*incrémentation *de la fonction O, notée O^{+1}, comme la fonction de délai $\mathtt{d}+1$ définie telle que, pour toute séquence d'entrée J de taille k telle que $k \geq \mathtt{d}+1$,*

$$O^{+1}(J) = O(J_{[1,\dots,k-1]}).$$

Notre caractérisation du comportement sous mode de mise à jour parallèle des modules acycliques consiste à énoncer que, partant d'une configuration x, et pour une séquence d'entrée suffisament longue J, le comportement de tout automate peut être prédit par une fonction de sortie, dont l'évaluation ne dépend que de J et non de x.

Propriété 15. *Soit M un module acyclique. Pour s un automate, et J une séquence de configurations d'entrée suffisament longue de taille k, il existe au moins une fonction de sortie O_s de délai $\mathtt{d} \leq k$ telle que*

$$M(x,J)_s = O_s(J).$$

Démonstration. Dans cette preuve nous supposons disposer des fonctions locales du module M sous la forme de formules.

Soit prof(s) la fonction qui donne la profondeur d'un automate dans le graphe d'interaction définie telle que prof(s) = 1 si et seulement si s n'est influencé par aucun automate, et prof(s) = $\max_{s'\in \mathrm{prec}(s)} \mathrm{prof}(s') + 1$ pour prec(s) l'ensemble des automates qui influencent s dans le cas contraire.

Soit s tel que prof(s) = 1. Nous construisons O_s comme la copie de la fonction locale f_s dans laquelle chaque variable α est substituée par la variable α_1, pour tout α dans

I. Nous pouvons vérifier que pour toute séquence J de taille k au moins supérieure à 0, et toute configuration x, que $M(x,J)_s = O_s(J)$.

Supposons que, pour un certain entier i, nous disposons d'une fonction de sortie O_s pour chaque entier s qui vérifie $\text{prof}(s) = i$ telle que, pour toute séquence J de taille au moins i, $M(x,J)_s = O_s(J)$. Dans la suite de cette preuve, nous montrons que cette hypothèse implique la même proposition pour l'entier $i+1$, ce qui implique le résultat par induction.

Soit s un entier tel que $\text{prof}(s) = i+1$. Nous construisons la fonction de sortie O_s comme la copie de la fonction locale f_s dans laquelle chaque variable $x_{s'}$ est substituée par la fonction $O^{+1}_{s'}$, et chaque variable α par α_1. Soit J une séquence d'entrée de taille k au moins $i+1$. Nous observons que

$$M(x,J)_s = M(M(x,J_{[1,\dots,k-1]}),J_{[k]})_s$$
$$M(x,J)_s = f_s(M(x,J_{[1,\dots,k-1]}) \cdot J_k).$$

Cependant, par définition la fonction f_s n'est influencée que par des automates dont les fonctions de sorties sont déjà définies. Soit R l'ensemble des automates dont la profondeur est inférieure à s, et T l'ensemble des automates dont la profondeur est supérieure ou égale à s. Soit $O_R(J)$ la configuration définie sur R qui pour tout $r \in R$ définit la valeur $O_r(J)$. Pour toute configuration x_T sur T, nous observons que

$$M(x,J)_s = f_s(O_R(J_{[1,\dots,k-1]}) \cdot x_T \cdot J_k)$$
$$M(x,J)_s = O_s(J),$$

ce qui implique le résultat. □

L'ensemble des fonctions de sortie qui prédisent la valeur de s sont toutes très similaires. En effet, il ne s'agit que des variantes de la même fonction qui décalent leur analyse de l'historique dans le temps d'une valeur arbitraire, entraînant des délais plus grands. Plus formellement, chacune de ces différentes fonctions sont obtenues comme les incrémentations successives d'une fonction de sortie à délai minimal. Nous proposons donc la définition naturelle de fonction de sortie canonique associée à un automate comme la fonction de sortie qui prédit la valeur de l'automate, tout en ayant le délai minimal.

Définition 69 (Fonction de sortie canonique)**.** *Soit M un module acyclique et s un automate de M. La* fonction de sortie canonique *de s, notée O_s, est la fonction de sortie qui vérifie la propriété 15, avec délai minimal.*

Exemple 46. *Soit $S = \{a,b,c\}$ un ensemble d'automates. Soit M le module qui définit les fonctions locales suivantes :*

$$f_a(x \cdot i) = i_\alpha$$
$$f_b(x \cdot i) = x_a \vee i_\beta$$

$$f_c(x \cdot i) = x_b \wedge \neg x_a$$

Ce module vérifie les fonctions de sortie canoniques suivantes :

$$\begin{aligned} O_a &= \alpha_1 \\ O_b &= \alpha_2 \vee \neg\beta_1 \\ O_c &= (\alpha_3 \vee \neg\beta_2) \wedge \neg\alpha_2 \end{aligned}$$

Considérons la séquence d'entrée $J = (i, i', i'')$, avec $i_\alpha = i_\beta = i''_\alpha = 0$, et $i'_\alpha = i'_\beta = i''_\beta = 1$. Pour prédire la valeur de a, b ou c après que la séquence d'entrée J soit appliquée au module M, on calcule $O_a(J)$, $O_b(J)$, et $O_c(J)$. Pour cela, on doit transformer J, qui est une séquence d'entrée, en un historique des valeurs de α et β défini sur $\Lambda^{\{\alpha,\beta\}\times\{1,2,3\}}$. Ici, la longueur de notre séquence est $k = 3$, et nos fonctions de sorties ont chacune un délai que nous notons $\mathtt{d}_a = 1$, $\mathtt{d}_b = 2$ et $\mathtt{d}_c = 3$. En appliquant la définition 67, on obtient que $O_a(J) = O_a(H_1(J))$, $O_b(J) = O_b(H_2(J))$ et $O_c(J) = O_c(H_3(J))$. Les historiques $H_1(J)$, $H_2(J)$ et $H_3(J)$ sont tous issus de J, et tous de longueur différentes. On peut les évaluer de cette façon :

$$\begin{aligned} \forall k \in \{1,2,3\}, H_k(J)_{(\alpha,1)} &= (J_{|J|-1+1})_\alpha = i''_\alpha \\ \forall k \in \{1,2,3\}, H_k(J)_{(\beta,1)} &= (J_{|J|-1+1})_\beta = i''_\beta \\ \forall k \in \{2,3\}, H_k(J)_{(\alpha,2)} &= (J_{|J|-2+1})_\alpha = i'_\alpha \\ \forall k \in \{2,3\}, H_k(J)_{(\beta,2)} &= (J_{|J|-2+1})_\beta = i'_\beta \\ H_3(J)_{(\alpha,3)} &= (J_{|J|-3+1})_\alpha = i_\alpha \\ H_3(J)_{(\beta,3)} &= (J_{|J|-3+1})_\beta = i_\beta. \end{aligned}$$

Pour obtenir la valeur de $O_a(H_1(J))$, sachant que $O_a = \alpha_1$, il suffit de prendre la valeur de α dans la première configuration de l'historique, ce qui est $H_1(J)_{(\alpha,1)} = i''_\alpha = 0$. On a donc $O_a(J) = O_a(H_1(J)) = H_1(J)_{(\alpha,1)} = i''_\alpha = 0$.

Similairement, pour calculer $O_b(H_2(J))$, sachant que $O_b = \alpha_2 \vee \neg\beta_1$, on note que α_2 est la valeur de α donnée par la seconde configuration de l'historique, et que β_1 est la valeur de β donnée dans la première configuration de l'historique. Ces valeurs sont $H_2(J)_{(\alpha,2)} = i'_\alpha = 1$ et $H_2(J)_{(\beta,1)} = i''_\beta = 1$, ce qui signifie que $O_b(J) = O_b(H_2(J)) = H_2(J)_{(\alpha,2)} \vee \neg H_2(J)_{(\beta,1)} = i'_\alpha \vee \neg i''_\beta = 1 \vee \neg 1 = 1$.

Enfin, pour calculer $O_c(H_3(J))$, en sachant que $O_c = (\alpha_3 \vee \neg\beta_2) \wedge \neg\alpha_2$, on calcule $\alpha_3 = H_3(J)_{(\alpha,3)} = i_\alpha = 0$, $\beta_2 = H_3(J)_{(\beta,2)} = i'_\beta = 1$ et $\alpha_2 = H_3(J)_{(\alpha,2)} = i'_\alpha = 1$. On a donc $O_c(J) = O_c(H_3(J)) = (H_3(J)_{(\alpha,3)} \vee \neg H_3(J)_{(\beta,2)}) \wedge H_3(J)_{(\alpha,2)} = (i_\alpha \vee \neg i'_\beta) \wedge \neg i'_\alpha = (0 \vee \neg 1) \wedge \neg 1 = 0$.

L'évaluation des fonctions de sortie O_b et O_c sur J est illustrée en figure 6.4.

Exemple 47. *Soit M le module développé dans l'exemple 45, dont les fonctions locales*

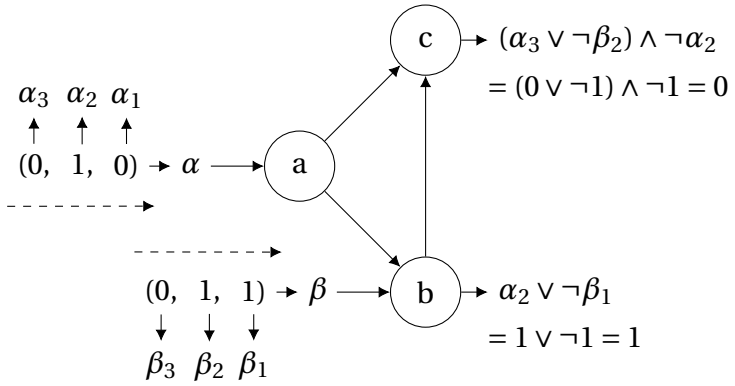

FIGURE 6.4 – Illustration du calcul opéré dans l'exemple 46. Sur la gauche, les séquences $(0,1,0)$ et $(0,1,1)$ sont les sous-séquences d'entrée extraites de J qui s'appliquent à α et β respectivement. Ces séquences se lisent de gauche à droite dans l'ordre de leur application sur le module. L'historique des entrées α et β est obtenu en inversant l'ordre de ces séquences : la variable α_1 dénote la plus récente valeur de α dans la séquence J, α_2 dénote la valeur qui précède α_1 dans le temps, et ainsi de suite. Sur la droite, on note les fonctions de sorties O_b et O_c sous forme de formules, ainsi que l'évaluation de ces formules sur la séquence d'entrée J.

sont

$$
\begin{aligned}
f_a(x \cdot i) &= i_\alpha \\
f_b(x \cdot i) &= x_a \vee i_\beta \vee \neg i_\alpha \\
f_c(x \cdot i) &= \neg x_b \wedge x_a \wedge \neg i_\gamma.
\end{aligned}
$$

Le module M vérifie les fonctions de sortie canoniques suivantes :

$$
\begin{aligned}
O_a &= \alpha_1 \\
O_b &= \alpha_2 \vee \beta_1 \vee \neg\alpha_1 \\
O_c &= \neg\alpha_3 \wedge \neg\beta_2 \wedge \alpha_2 \wedge \alpha_2 \wedge \neg\gamma_1.
\end{aligned}
$$

Les fonctions O_a, O_b et O_c ont délai 1, 2 et 3 respectivement.

6.2.2 Calcul d'une fonction de sortie booléenne

Pour M un module acyclique, nous nous intéressons au problème de calculer efficacement les fonctions de sortie canoniques des automates de M. La méthode que nous explorons est récursive, car les fonctions de sortie des automates plus profonds dans le graphe d'interaction de M dépendent directement de l'incrémentation des fonctions de sortie des automates à plus faible profondeur dans le graphe d'interaction. La

question centrale de l'efficacité de notre méthode repose sur le choix de l'encodage de ces fonctions de sortie.

En effet, pour n'importe laquelle de ces méthodes d'encodage, nous pouvons décrire un algorithme très général qui calcule les fonctions de sortie d'un module acyclique. Cette méthode repose sur les fonctions locales du module étudié, dans lesquelles les variables d'entrées sont substituées par les variables correspondantes de profondeur 1. Les variables d'automates sont quant à elles remplacées par l'incrémentation de la fonction de sortie de l'automate correspondant. Ce processus est bien défini car la structure acyclique du module abolit tout cycle d'influence.

Exemple 48. *Soit k un entier. On définit M_k le module acyclique constitué des automates $S = \{1, \dots, k\}$, et des entrées $\{\alpha^0, \alpha^1, \dots, \alpha^k\}$, tel que*

$$f_1(x \cdot i) = i_{\alpha^0} \oplus i_{\alpha^1}$$
$$\forall 1 < i \leq k, f_i(x \cdot i) = x_{i-1} \oplus i_{\alpha^i}.$$

Pour $k \in \mathbb{N}$, le module M_k développé dans l'exemple 48 est de taille une fonction linéaire en k. Considérons le calcul des fonctions de sortie de M_k sous la forme de formules booléennes avec les portes logiques $\{\wedge, \vee, \neg\}$. Par la procédure décrite précédemment, nous pouvons définir les fonctions de sortie du module de la façon suivante :

$$O_1(J) = (\alpha_1^0 \vee \alpha_1^1) \wedge (\neg\alpha_1^0 \wedge \neg\alpha_1^1)$$
$$\forall 1 < i \leq k, O_k(J) = (O_{i-1}^{+1} \vee \alpha_1^i) \wedge (\neg O_{i-1}^{+1} \vee \neg\alpha_1^i).$$

On note que la longueur du développement naïf de la formule booléenne qui encode la fonction de sortie O_i est deux fois supérieure à celle de la formule qui encode la fonction de sortie O_{i-1}. En général, cela signifie qu'il est possible que le coût du calcul de la fonction de sortie O_k soit de l'ordre de 2^k.

La même fonction de sortie possède une taille toujours polynomiale en la taille du module acyclique lorsque l'on considère l'encodage des fonctions de sortie du module sous la forme d'un circuit booléen. La différence provoquée par ce choix est que les multiples références à des fonctions de sortie précédentes n'explosent pas la taille du circuit obtenu grâce à la structure plus permissive des circuits booléens par rapport aux formules booléennes.

Propriété 16 (Perrot, Perrotin et Sené 2020b)**.** *Soit M un module acyclique, G son graphe d'interaction, et s un automate de M. Il existe un circuit booléen C calculable en temps polynomial en la taille de M et G qui encode la fonction de sortie O_s.*

Démonstration. Nous développons une méthode pour calculer en temps polynomial le circuit booléen qui encode la fonction de sortie de tout automate s dans M. Cette méthode calcule d'abord une liste de besoins. Cette liste est construite inductivement,

et vaut initialement $R_0 = \{(s,0)\}$, ce qui s'interprète par le besoin de construire la fonction de sortie de l'automate s avec le délai supplémentaire 0.

Nous construisons récursivement la liste de la façon suivante : $(t', \mathtt{d}+1) \in R_{k+1}$ si et seulement s'il existe $(t, \mathtt{d}) \in R_k$ tel que t' influence t dans le module M. La liste finale obtenue par cette méthode est définie par $R = \bigcup_i R_i$.

Proposition 1. *La liste R est calculable en temps polynomial en la taille de M.*

Afin de vérifier cette proposition, considérons D la longueur du plus long chemin dans M. Premièrement, il est facile de voir que, pour tout $k > D$, notre définition vérifie $R_k = \emptyset$. Ensuite, pour tout entier $k \leq D$, la valeur $\mathtt{d}$ maximale qui vérifie $(t, \mathtt{d}) \in R_k$ pour un certain automate t vérifie également $\mathtt{d} \leq k$. Nous pouvons donc conclure que la liste R_k est toujours de taille $n \times k$ au pire, pour n le nombre d'automates de M, ce qui est toujours moins que $n \times D$. Nous concluons donc que la liste totale est au maximum de taille $n \times D^2$, ce qui est polynomial. Notre méthode de construction recursive permettra donc de la calculer en temps polynomial, ce qui vérifie la Proposition 1.

Nous construisons le circuit C à partir de R de la façon suivante : nous prenons une instance du circuit qui encode la fonction locale f_t pour chaque paire $(t, \mathtt{d}) \in R$. Nous combinons l'ensemble des circuits obtenus en prenant toute variable du circuit étiquetée par i_α, pour $\alpha \in I$, et en la remplaçant par son équivalent $\alpha_{\mathtt{d}+1}$, pour $\mathtt{d}$ le délai ajouté du circuit actuellement modifié. Par exemple, pour $(t, \mathtt{d}) \in R$, tel que t dépend d'une entrée $\alpha \in I$, nous ajoutons au circuit C une copie du circuit qui encode f_t dans lequel toute occurence de la variable i_α a été remplacée par $\alpha_{\mathtt{d}+1}$. Enfin, nous connectons l'ensemble des circuits générés de cette façon en remplaçant toute occurence de la variable $x_{t'}$ dans le circuit de la paire $(t, \mathtt{d})$ par le littéral qui correspond à la porte de sortie du circuit qui encode l'automate t' avec le délai ajouté $\mathtt{d}+1$. Par définition de R, ce circuit fait toujours partie de notre assemblage. Le circuit total obtenu calcule la fonction de sortie O_s, et l'algorithme qui le calcule est au pire en $\mathcal{O}(n \times D^2 \times K)$, pour K la taille maximale d'encodage des circuits booléens donnés en entrée. □

6.2.3 Dynamique des modules acycliques

Les modules acycliques sont une restriction des modules dont le comportement, sous mise à jour parallèle, est caractérisé comme une fonction des séquences d'entrée utilisées pour les mettre à jour. Cette compréhension peut nous aider à mieux comprendre la dynamique des réseaux qui sont obtenus par branchement récursif de ces modules acycliques.

Considérons un module acyclique M, et F un réseau d'automates obtenu par le branchement récursif de M. Le nombre d'entrées de M nous permet une observation simple sur le nombre d'attracteurs contenus dans la dynamique parallèle de F. Par exemple, considérons les points fixes de F. Tout point fixe de F peut également être exprimé comme un point fixe de M, à condition que les entrées de M soient figées sur les valeurs qui correspondent aux automates dont l'influence remplace ces entrées

pour obtenir F. Autrement dit, tout point fixe, et en général tout attracteur de F, peut être observé sur M à condition de sélectionner la séquence d'entrée qui mime le branchement qui construit F à partir de M. Ce simple fait est une observation directe de la priopriété 12.

Dans le cas acyclique, ce cas implique que nous pouvons définir une borne supérieure sur le nombre d'attracteurs de F, par le nombre d'attracteurs similaires que l'on peut observer sur M en supposant une séquence d'entrée qui fonctionne en ce sens. Cette dernière valeur est bornée par le nombre d'entrées de M. En effet, lorsque les valeurs des entrées de M sont figées pour reproduire un point fixe, l'ensemble des états du reste du réseau est une fonction de ces entrées. Cela signifie que le nombre de points fixes de F est borné par le nombre de configurations d'entrée possibles de M. Cet argument, porté pour toute taille d'attracteur, nous permet d'obtenir le résultat suivant.

Théorème 7 (PERROT, PERROTIN et SENÉ 2020a)**.** *Soit M un module acyclique avec k entrées, et $\mathtt{w}$ une fonction de branchement récursif telle que $\circlearrowleft_{\mathtt{w}} M$ est un réseau d'automates. Notons $a(k,c)$ le nombre d'attracteurs de taille c dans la dynamique parallèle de $\circlearrowleft_{\mathtt{w}} M$. Nous énonçons que $a(k,c) \leq A(k,c)$, pour :*

$$A(k,1) = |\Lambda|^k$$
$$A(k,c) = |\Lambda|^{kc} - \sum_{c'<c, c' \mid c} A(k,c').$$

Démonstration. Soit M un module avec k entrées, et $\mathtt{w}$ une fonction de branchement qui vérifie $\mathrm{dom}(\mathtt{w}) = I$. Soit $\mathtt{w}(I)$ la notation de l'ensemble $\mathtt{w}(I) = \{\mathtt{w}(\alpha) \mid \alpha \in I\}$. Nous observons le fait suivant :

$$|\mathtt{w}(I)| \leq |I| = k. \tag{6.1}$$

Considérons maintenant le réseau d'automates $F = \circlearrowleft_{\mathtt{w}} M$, et $X = (x_1, x_2, \ldots, x_c)$ un attracteur de sa dynamique. Par définition, X vérifie $F(x_i) = x_{i+1}$ pour $i < c$, $F(x_c) = x_1$ et pour tout i, j, $x_i = x_j \implies i = j$. Pour $R \subseteq S$, nous définissons $X|_R$ la séquence qui vérifie $X|_R = (x_1|_R, x_2|_R, \ldots, x_c|_R)$. Supposons X' un autre attracteur de taille c. Nous faisons la proposition suivante.

Proposition 2. $X|_{\mathtt{w}(I)} = X'|_{\mathtt{w}(I)} \implies X = X'$.

Pour vérifier cette proposition, supposons que $X|_{\mathtt{w}(I)} = X'|_{\mathtt{w}(I)}$. Par hypothèse, M est acyclique, et il existe un ensemble non vide $S_1 \subseteq S$ qui contient exactement tous les automates de M qui ne sont influencés que par des entrées. Cela signifie que notre hypothèse implique $X|_{\mathtt{w}(I) \cup S_1} = X'|_{\mathtt{w}(I) \cup S_1}$. Maintenant, considérons S_2 l'ensemble non vide qui contient exactement tous les automates influencés seulement par des entrées et des automates dans S_1. Cela nous permet de déduire que $X|_{\mathtt{w}(I) \cup S_1 \cup S_2} = X'|_{\mathtt{w}(I) \cup S_1 \cup S_2}$. En construisant l'ensemble S_i composé exactement des automates influencés par des entrées et des automates dans les ensembles qui précèdent, et sachant qu'aucun de ces ensembles n'est vide, nous prouvons la proposition par induction.

Cette proposition nous permet de conclure qu'il ne peut y avoir dans la dynamique de F plus d'attracteurs de taille c que d'ensembles $X|_{\mathtt{w}(I)}$ distincts. Plus formellement,

$$a(k,c) \leq |\Lambda|^{kc}. \tag{6.2}$$

Soit X une des possibles $|\Lambda|^{kc}$ différentes séquences de c configurations. Supposons que $F(x_i) = x_{i+1}$ pour $i < c$ et $F(x_c) = x_1$. Par définition, s'il existe i, j tels que $i \neq j$ et $x_i = x_j$, alors X est une séquence périodique, et n'est pas un attracteur. Cela est équivalent à trouver une séquence X' plus petite telle que X peut être composé comme la répétition X'^q, pour q un entier positif. Cela implique que, pour toute séquence X' telle que la taille de X' divise c, il existe une séquence $X = X'^q$ pour q un entier positif tel que X est compté par la formule $|\Lambda|^{kc}$, et X n'est pas un attracteur. Nous pouvons donc raffiner la borne précédente et affirmer que $a(k,c)$ n'est pas plus grand que $|\Lambda|^{kc} - \sum_{x'<c, c'|c} A(k,c')$. □

De ce théorème central, nous pouvons établir un ensemble de résultats simples qui permettent d'illustrer que la source principale de la complexité des modules acycliques dans le cas booléen est le nombre de leurs entrées, et non le nombre de leurs automates. Pour ce faire, nous décrivons les problèmes de décision suivant.

▸ ACYCLIC MODULE ATTRACTOR PROBLEM

Entrée: Un module acyclique booléen M, k entrées et n automates, une fonction de branchement $\mathtt{w}$ et un nombre c encodé en unaire.

Promesse: L'encodage des fonctions locales de M ne comporte que des variables essentielles.

Question: Existe-t-il un attracteur de taille c dans la dynamique parallèle du réseau d'automates $\circlearrowleft_{\mathtt{w}} M$?

▸ ACYCLIC MODULE FIXED POINT PROBLEM

Entrée: Un module acyclique booléen M, k entrées et n automates et une fonction de branchement $\mathtt{w}$.

Promesse: L'encodage des fonctions locales de M ne comporte que des variables essentielles.

Question: Existe-t-il une configuration x telle que $\circlearrowleft_{\mathtt{w}} M(x) = x$?

Ces problèmes possèdent les complexités suivantes.

Théorème 8 (PERROT, PERROTIN et SENÉ 2020a)**.** *Le problème Acyclic Module Attractor peut être résolu en temps $\mathcal{O}(f(k \times c)q(n))$ pour f une fonction et q un polynôme,* i.e. *il est dans FPT.*

Démonstration. L'algorithme qui prouve ce théorème vérifie toutes les séquences d'entrées pour k entrées et de longueur c possibles, dont il existe $|\mathbb{B}|^{k\times c}$ représentantes au total. Pour vérifier si une telle séquence J décrit un attracteur de taille c dans la dynamique du réseau $\circlearrowleft_{\mathtt{w}} M$, il suffit de mettre à jour le module M avec une séquence d'entrées composée comme la répétition de la séquence J jusqu'à ce que sa taille

dépasse le délai maximal parmi les fonctions de sortie de M. Un attracteur sera obtenu si et seulement si la séquence récurrente obtenue en sortie est identique à la séquence J, à l'application de la fonction `w` près. Cette procédure s'exécute en temps $r(n \times k \times c)$, pour r un polynôme.

Nous observons que tout attracteur de taille c dans la dynamique de $\circlearrowleft_{\mathtt{w}} M$ correspond nécessairement à une séquence d'entrée J, qui peut être récupérée simplement par la lecture des automates branchés par la fonction `w` pendant c étapes. Notre procédure est donc fiable.

En testant chaque séquence J, on établit un algorithme qui vérifie l'existence d'un attracteur de taille c en une complexité $\mathcal{O}(|\mathbb{B}|^{k\times c} \times r(n \times k \times c)$, ce qui signifie qu'il existe une fonction f et un polynôme q tels que la complexité de cet algorithme est $\mathcal{O}(f(k \times c)q(n))$. □

Théorème 9 (PERROT, PERROTIN et SENÉ 2020a)**.** *Le problème Acyclic Module Fixed Point est NP-complet.*

Démonstration. Tout d'abord, ce problème est dans NP car, pour toute configuration, vérifier qu'il s'agit d'un point fixe demande simplement d'exécuter le module une fois.

Pour démontrer qu'il s'agit d'un problème NP-difficile, on procédons à une réduction depuis le problème SAT. Pour f une formule, on construit un module qui dispose d'un automate pour chaque variable dans f, ainsi que deux autres automates `solver` et `oscillator`. Chaque automate correspondant à une variable dispose d'une entrée unique qui sont toutes branchées par `w` à l'automate qui la contient ; de façon à ce qu'une fois branchés, chacun de ces automates a dispose d'une fonction locale triviale $f_a(x) = x_a$.

La fonction locale de l'automate `solver` est une réécriture de la formule f dans laquelle chaque variable est remplacée par l'influence de l'automate qui lui correspond. Enfin, la fonction locale de l'automate `oscillator` est définie comme $f_{\mathtt{oscillator}}(x) = \neg\mathtt{solver} \wedge \neg\mathtt{oscillator}$; cet automate oscille entre les valeurs 0 et 1, excepté lorsque l'automate `solver` est évalué à 1, alors la valeur de `oscillator` se fixe à 0.

Par construction, l'évaluation de chaque automate qui n'est pas `solver` ni `oscillator` ne change pas dans le temps. Cela implique que la valeur de l'automate `solver` se fixe après une mise à jour. On en conclut que la valeur du dernier automate, `oscillator`, va se fixer, et le réseau va obtenir un point fixe si et seulement si la formule f est satisfaite ; on en déduit la réduction, et le résultat. □

6.2.4 Caractérisation de la dynamique par les fonctions de sortie

Les fonctions de sortie d'un module acyclique M permettent l'exécution complète du module M en ne prenant en compte que les dernières d configurations dans la séquence de configurations d'entrée, pour d le délai maximal parmi les fonctions de sortie de M. Nous établissons que les fonctions de sortie de M caractérisent son

comportement asymptotique. Il est en effet intéressant de noter que les fonctions de sortie d'un module acyclique ne permettent pas de prédire l'évaluation d'un automate sur de courtes exécutions. Cela s'explique par l'observation que la configuration initiale d'une module, pour une exécution, possède une influence sur le calcul de ses automates, jusqu'à un nombre de mise à jour égal à la profondeur maximale du réseau, point auquel le comportement de tous les automates du module devient une fonction de l'historique de ses entrées.

Cette idée de caractérisation du comportement asymptotique d'un module par ses fonctions de sortie se raffine de façon concrète par le théorème suivant. Celui-ci énonce que, si deux modules acycliques possèdent suffisament de fonctions de sortie en commun et sont branchés récursivement de façon similaire relativement à ces fonctions de sortie pour obtenir des réseaux d'automates, alors les attracteurs des dynamiques parallèles respectives de ces réseaux sont isomorphes.

Théorème 10 (Perrot, Perrotin et Sené 2020a)**.** *Soient M et M' deux modules acycliques, avec T et T' des sous-ensembles de leurs ensembles d'automates tels que $|T| = |T'|$. S'il existe g une bijection de I vers I' et h une bijection de T vers T' telle que pour tout s dans T, les fonctions O_s et $O'_{h(s)}$ ont même délai, et que pour toute séquence d'entrée J de longueur au moins le délai de O_s,*

$$O_s(J) = O'_{h(s)}(J \circ g^{-1})$$

alors, pour toute fonction de branchement $\mathtt{w} : I \to T$, les graphes de la dynamique parallèle des réseaux d'automates $\circlearrowleft_{\mathtt{w}} M$ et $\circlearrowleft_{h\circ\mathtt{w}\circ g^{-1}} M'$ ont des attracteurs isomorphes, au nommage des automates près.

Démonstration. En préliminaire, nous remarquons que $\mathtt{w}$ a pour domaine I, et donc que $\circlearrowleft_{\mathtt{w}} M$ est un réseau d'automates que nous noterons F. La définition de g nous permet la même remarque pour $\circlearrowleft_{h\circ\mathtt{w}\circ g^{-1}} M'$, que nous noterons F'. Soient G et G' les attracteurs des graphes de la dynamique parallèle de F et F' respectivement. L'objectif de cette preuve est de prouver que les sous-graphes G et G' sont isomorphes.

Pour toute configuration $x \in \Lambda^S$, nous définissons la séquence d'entrée de taille k générée par x, que nous notons $\hat{J}(x)^k$, comme la séquence qui vérifie

$$\hat{J}(x)^k_l = F^{l-1}(x)|_T \circ \mathtt{w}, \text{pour } 1 \leq l \leq k.$$

Intuitivement, la séquence $J(x)^k_l$ contient les valeurs successives des automates de T qui sont considérés comme les sorties du réseau F. Cet historique démarre à la configuration x et s'arrête après k mises à jour. La séquence obtenue est combinée avec le branchement $\mathtt{w}$, ce qui en fait une séquence d'entrée.

Proposition 3. *Soit k un entier qui vérifie $d_s \leq k$ pour tout automate s, et pour d_s le délai de la fonction de sortie canonique de s. Pour J une séquence d'entrée de longueur k, l'évaluation $M(x, J)$ ne dépend que de J, et non de la configuration de départ x.*

Cette proposition se vérifie immédiatement par l'application de la propriété 15. Le calcul d'une fonction de sortie ne dépend pas de la configuration initiale sur des exécutions suffisament longues. En nous fondant sur cette proposition, nous établissons la notation $M(x, J) = M(J)$ dans le reste de cette demonstration lorsqu'applicable, c'est-à-dire lorsque la séquence J est plus longue que le délai maximal parmi les fonctions de sortie de M.

Proposition 4. *Soit J une séquence d'entrée telle que $M(J)$ est bien définie. Nous énonçons que $\hat{J}(M(J)^k = J \implies M(J) \in V(G)$, pour $V(G)$ l'ensemble des sommets du graphe des attracteurs de F.*

Cette proposition énonce que si la configuration $M(J)$ génère la séquence d'entrée J lorsque utilisée pour mettre à jour le réseau F, alors $M(J)$ fait partie d'un attracteur. Pour le prouver, notons $x = M(J)$. Par hypothèse, $\hat{J}(x)^k = J$. Cela implique

$$\begin{aligned}
F^k(x) =& F(F^{k-1}(x)) = M(F^{k-1}(x), F^{k-1}(x)|_T \circ \mathtt{w}) \\
=& M(M(\dots M(x, F^0(x)|_T \circ \mathtt{w}) \dots, F^{k-2}(x)|_T \circ \mathtt{w}), F^{k-1}(x)|_T \circ \mathtt{w}) \\
=& M(M(\dots M(x, \hat{J}(x)_1^k) \dots, \hat{J}(x)_{k-1}^k), \hat{J}(x)_k^k) \\
=& M(x, \hat{J}(x)^k) = M(x, J) = M(J) = x.
\end{aligned}$$

En conclusion, $F^k(x) = x$, ce qui implique que x fait partie d'un attracteur de taille k, ce qui vérifie la proposition.

Proposition 5. *Soit x dans $V(G)$. Il existe x' dans $V(G')$ tel que $\hat{J}(x)^k \circ g^{-1} = \hat{J}(x')^k$, pour tout entier k.*

Cette proposition implique que pour toute configuration récursive de F, il existe une configuration récursive de F' qui génère une séquence d'entrées identique à l'application de la bijection g près.

Pour le prouver, considérons $x \in V(G)$ et k un entier supérieur au délai de toute fonction de sortie de M et M', tels que $F^k(x) = x$. Considérons également les séquences d'entrées $\hat{J}(x)^k$ et $\hat{J}(x)^k \circ g^{-1}$. La proposition 3 implique que $M'(\hat{J}(x)^k \circ g^{-1})$ est une configuration de M' bien définie, que nous notons x'. Nous devons maintenant prouver que $\hat{J}(x)^k \circ g^{-1} = \hat{J}(x')^k$. D'après nos définitions,

$$\hat{J}(x)_1^k \circ g^{-1} = F^0(x)|_T \circ \mathtt{w} \circ g^{-1} = x|_T \circ \mathtt{w} \circ g^{-1},$$

alors que

$$\begin{aligned}
\hat{J}(x')_1^k = F'^0(x')|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} =& x'|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
=& M'(\hat{J}(x)^k \circ g^{-1})|_{T'} \circ h \circ \mathtt{w} \circ g^{-1}.
\end{aligned}$$

Nous observons que, pour tout s' dans T', $M'(\hat{J}(x)^k \circ g^{-1})_{s'} = O'_{s'}(\hat{J}(x)^k \circ g^{-1})$, ce qui,

par l'hypothèse du théorème, est égal à $O_{h^{-1}(s')}(\hat{J}(x)^k)$. Cela implique que

$$M'(\hat{J}(x)^k \circ g^{-1})|_{T'} \circ h = M(\hat{J}(x)^k)|_T \circ h^{-1} \circ h = x|_T$$

ce dont nous déduisons

$$\hat{J}(x')_1^k = x|_T \circ \mathtt{w} \circ g^{-1} = \hat{J}(x)_1^k \circ g^{-1}.$$

En conclusion, $\hat{J}(x)_1^k \circ g^{-1} = \hat{J}(x')_1^k$.

Ce fait marque la première étape d'une démonstration par induction pour prouver $\hat{J}(x)^k \circ g^{-1} = \hat{J}(x')^k$. Supposons donc que, pour un certain ℓ inférieur à k,

$$\hat{J}(x)_{[1,\ell]}^k \circ g^{-1} = \hat{J}(x')_{[1,\ell]}^k.$$

Montrons désormais que $\hat{J}(x)_{[1,\ell+1]}^k \circ g^{-1} = \hat{J}(x')_{[1,\ell+1]}^k$. Pour cela, il nous est simplement nécessaire de montrer que $\hat{J}(x)_{\ell+1}^k \circ g^{-1} = \hat{J}(x')_{\ell+1}^k$. Nous savons que

$$\begin{aligned}
\hat{J}(x)_{\ell+1}^k \circ g^{-1} &= F^\ell(x)|_T \circ \mathtt{w} \circ g^{-1} \\
&= M(x, \hat{J}(x)_{[1,\ell]}^k)|_T \circ \mathtt{w} \circ g^{-1} \\
&= M(M(\hat{J}(x)^k), \hat{J}(x)_{[1,\ell]}^k)|_T \circ \mathtt{w} \circ g^{-1} \\
&= M(\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k)|_T \circ \mathtt{w} \circ g^{-1},
\end{aligned}$$

et

$$\begin{aligned}
\hat{J}(x')_{\ell+1}^k &= F'^\ell(x')|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
&= M'(x', \hat{J}(x')_{[1,\ell]}^k)|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
&= M'(x', \hat{J}(x)_{[1,\ell]}^k \circ g^{-1})|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
&= M'(M'(\hat{J}(x)^k \circ g^{-1}), \hat{J}(x)_{[1,\ell]}^k \circ g^{-1})|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
&= M'((\hat{J}(x)^k \circ g^{-1}) \cdot (\hat{J}(x)_{[1,\ell]}^k \circ g^{-1}))|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\
&= M'((\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k) \circ g^{-1})|_{T'} \circ h \circ \mathtt{w} \circ g^{-1}.
\end{aligned}$$

Nous observons que la séquence $(\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k) \circ g^{-1})$ est de taille au moins k, ce qui nous permet de l'utiliser pour calculer le résultat d'une fonction de sortie. De l'hypothèse du théorème, nous déduisons que pour tout s' dans T',

$$\begin{aligned}
M'((\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k) \circ g^{-1})_{s'} &= O'_{s'}((\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k) \circ g^{-1}) \\
&= O_{h^{-1}(s')}(\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k) \\
&= M(\hat{J}(x)^k \cdot \hat{J}(x)_{[1,\ell]}^k)_{h^{-1}(s')}
\end{aligned}$$

ce qui, par une autre utilisation de l'hypothèse du théorème, implique

$$\begin{aligned}\hat{J}(x')^k_{\ell+1} &= M'((\hat{J}(x)^k \cdot \hat{J}(x)^k_{[1,\ell]}) \circ g^{-1})|_{T'} \circ h \circ \mathtt{w} \circ g^{-1} \\ &= M(\hat{J}(x)^k \cdot \hat{J}(x)^k_{[1,\ell]})_T \circ h^{-1} \circ h \circ \mathtt{w} \circ g^{-1} \\ &= M(\hat{J}(x)^k \cdot \hat{J}(x)^k_{[1,\ell]})_T \circ \mathtt{w} \circ g^{-1} \\ &= \hat{J}(x)^k_{\ell+1} \circ g^{-1}\end{aligned}$$

ce qui termine notre preuve par induction, de laquelle nous déduisons que $\hat{J}(x)^k \circ g^{-1} = \hat{J}(x')^k$. Nous en déduisons $\hat{J}(M'(\hat{J}(x)^k \circ g^{-1}))^k = \hat{J}(x')^k = \hat{J}(x)^k \circ g^{-1}$, ce qui, à l'aide de la proposition 4, implique que x' fait partie de $V(G')$, et que x' fait partie d'un attracteur dont la taille divise k, tout comme x. Ceci conclut la preuve de la proposition 5 en supposant k suffisament grand, mais nous notons que la proposition se maintient pour tout entier k.

Nous observons une séquence symétrique d'arguments pour prouver que, pour toute configuration x' dans $V(G')$, il existe x dans $V(G)$ tel que $\hat{J}(x')^k \circ g = \hat{J}(x)^k$. De ceci, nous déduisons que, pour tout x dans $V(G)$, il existe une unique configuration x' dans $V(G')$ qui vérifie cette équation. Cela se voit par le fait que s'il existait x', x'' dans $V(G')$ telles que $\hat{J}(x')^k \circ g = \hat{J}(x)^k$, et $\hat{J}(x'')^k \circ g = \hat{J}(x)^k$, alors $\hat{J}(x')^k = \hat{J}(x'')^k$, ce qui implique pour k un multiple des tailles des attracteurs qui contiennent x' et x'' que $x' = M'(\hat{J}(x')^k) = M'(\hat{J}(x'')^k) = x''$, et x' x'' sont la même configuration.

Soit $\hat{h} : V(G) \to V(G')$ la bijection qui, à tout x dans $V(G)$, associe x' dans $V(G')$ telle que $\hat{J}(x')^k \circ g = \hat{J}(x)^k$, ce qui implique que $\hat{h}(x) = M'(\hat{J}(x)^k \circ g^{-1})$, pour k plus grand que le délai de toute fonction de sortie dans M et M' et multiple de la taille des attracteurs qui contiennent x et $\hat{h}(x)$. Le reste de cette preuve montre que $\hat{h}$ est un isomorphisme de G vers G', ce qui demande de montrer que, pour tout x dans $V(G)$, $\hat{h}(F(x)) = F'(\hat{h}(x))$.

Soit x dans $V(G)$ et k un entier multiple de la longueur de l'attracteur auquel x appartient, tel que k est plus grand que le délai de toutes les fonctions de sortie de M et M'. Nous savons que

$$\begin{aligned}\hat{h}(F(x)) &= M'(\hat{J}(F(x))^k \circ g^{-1}) \\ &= M'((F^0(F(x))|_T \circ \mathtt{w}, F^1(F(x))|_T \circ \mathtt{w}, \ldots, F^{k-1}(F(x))|_T \circ \mathtt{w}) \circ g^{-1}) \\ &= M'((F^1(x)|_T \circ \mathtt{w}, F^2(x)|_T \circ \mathtt{w}, \ldots, F^k(x)|_T \circ \mathtt{w}) \circ g^{-1}) \\ &= M'((F^1(x)|_T \circ \mathtt{w} \circ g^{-1}, F^2(x)|_T \circ \mathtt{w} \circ g^{-1}, \ldots, F^k(x)|_T \circ \mathtt{w} \circ g^{-1})).\end{aligned}$$

Considérons un élément individuel de la séquence çi-dessus, $F^\ell(x)|_T \circ \mathtt{w} \circ g^{-1}$. Pour tout s dans S,

$$\begin{aligned}F^\ell(x)_s &= M(x, \hat{J}(x)^\ell)_s \\ &= M(M(\hat{J}(x)^k), \hat{J}(x)^\ell)_s \\ &= M(\hat{J}(x)^k \cdot \hat{J}(x)^\ell)_s\end{aligned}$$

$$
\begin{aligned}
&= O_s(\hat{J}(x)^k \cdot \hat{J}(x)^\ell) \\
&= O'_{h(s)}((\hat{J}(x)^k \cdot \hat{J}(x)^\ell) \circ g^{-1}) \\
&= M'((\hat{J}(x)^k \cdot \hat{J}(x)^\ell) \circ g^{-1})_{h(s)} \\
&= M'(M'(\hat{J}(x)^k \circ g^{-1}), \hat{J}(x)^\ell \circ g^{-1})_{h(s)} \\
&= M'(\hat{h}(x), \hat{J}(x)^\ell \circ g^{-1})_{h(s)} \\
&= F'^\ell(\hat{h}(x))_{h(s)}
\end{aligned}
$$

ce qui implique que $F^\ell(x)|_T \circ \mathtt{w} \circ g^{-1} = F'^\ell(\hat{h}(x))|_{T'} \circ h \circ \mathtt{w} \circ g^{-1}$. En remplaçant ces termes dans l'équation initiale, nous obtenons

$$
\begin{aligned}
\hat{h}(F(x)) &= M'((F'^1(\hat{h}(x))|_{T'} \circ h \circ \mathtt{w} \circ g^{-1}, F'^2(\hat{h}(x))|_{T'} \circ h \circ \mathtt{w} \circ g^{-1}, \dots \\
&\qquad \dots, F'^k(\hat{h}(x))|_{T'} \circ h \circ \mathtt{w} \circ g^{-1})) \\
&= M'(\hat{J}(\, F'(\hat{h}(x)))) \\
&= F'(\hat{h}(x)),
\end{aligned}
$$

ce qui conclut cette démonstration. □

Exemple 49. *Soit M le module acyclique défini sur les automates $S = \{a, b, c, d\}$ et l'entrée $I = \{\alpha\}$, et qui admet les fonctions locales*

$$
\begin{aligned}
f_a(x \cdot i) &= i_\alpha \\
f_b(x \cdot i) &= x_a \vee i_\alpha \\
f_c(x \cdot i) &= \neg x_a \\
f_d(x \cdot i) &= (x_b \wedge \neg x_c) \vee i_\alpha.
\end{aligned}
$$

Nous observons les fonctions de sortie canoniques

$$
\begin{aligned}
O_a(J) &= \alpha_1 \\
O_b(J) &= \alpha_2 \vee \alpha_1 \\
O_c(J) &= \neg \alpha_2 \\
O_d(J) &= ((\alpha_3 \vee \alpha_2) \wedge \alpha_3) \vee \alpha_1 \\
&= \alpha 3 \vee \alpha_1.
\end{aligned}
$$

Soit M' le module acyclique défini sur les automates $S' = \{l, m, n\}$ et l'entrée $I' = \{\beta\}$ qui admet les fonctions locales

$$
\begin{aligned}
f'_l(x \cdot i) &= i_\alpha \\
f'_m(x \cdot i) &= x_l \\
f'_n(x \cdot i) &= x_m \vee i_\alpha.
\end{aligned}
$$

Nous observons les fonctions de sortie canoniques

$$O'_l(J) = \beta_1$$
$$O'_m(J) = \beta_2$$
$$O'_n(J) = \beta_3 \vee \beta_1.$$

Soient $T = \{d\}, T' = \{n\}$, $g : I \to I'$ la bijection qui définit $g(\alpha) = \beta$, et $h : T \to T'$ la bijection qui définit $h(d) = n$. Les modules M et M' vérifient $O_d(J) = O'_{h(d)}(J \circ g^{-1})$, et par conséquent, pour $\mathtt{w} : T \to I$ le branchement qui définit $\mathtt{w}(d) = \alpha$, les réseaux d'automates $\circlearrowleft_{\mathtt{w}} M$ et $\circlearrowleft_{h \circ \mathtt{w} \circ g^{-1}} M'$ ont des attracteurs isomorphes.

Ce théorème permet une notion nouvelle d'équivalence sur les modules acycliques pour lesquels on désigne un sous-ensemble d'automates T comme sorties. Cette notion d'équivalence est vérifiée pour toute paire de modules pour lesquels les conditions nécessaires sont vérifiées pour que le théorème 10 s'applique. Afin d'explorer ces classes d'équivalence de façon productive, nous définissons une restriction sur les ensembles T définissables pour un module M. Cette restriction n'accepte que les modules qui ne possèdent pas de branches inutiles, à savoir ceux dont tous les automates influencent directement ou indirectement un automate dans T.

Définition 70 (Ensemble de sortie)**.** *Soit M un module acyclique, et T un sous-ensemble de S. On définit que T est un* ensemble de sortie *de M si et seulement si, pour tout automate s, il existe un chemin dans le graphe d'interaction de M depuis s vers t, pour t un automate dans T.*

Naturellement, cette définition restreint notre étude à des ensembles de sorties non vides. Nous pouvons maintenant définir les classes d'équivalence sur les paires (M, T), pour M un module acyclique et T un ensemble de sortie de M.

Définition 71 (Équivalence en sortie entre modules acycliques)**.** *Soient M et M' deux modules, et T, T' des ensembles de sorties de M et M' respectivement. On définit que (M, T) et (M', T') sont équivalents en sortie, noté $(M, T) \equiv_O (M', T')$, si et seulement si toutes les conditions sont vérifiées pour appliquer le théorème 10 sur les modules M, M' et les sous-ensembles T, T'.*

Cette relation d'équivalence permet la construction de classes d'équivalence. Ainsi, pour M un module acyclique et T un sous-ensemble de ses automates, on définit par $C_{(M,T)} = \{(M', T') | \ (M', T') \equiv_O (M, T)\}$ la classe d'équivalence qui contient (M, T).

Parmi les très nombreuses classes d'équivalence possibles, nous notons de prime abord l'existence de classes triviales. Ces classes triviales d'équivalence ne sont composées que de fonctions de sortie de délai 1. Cette contrainte provoque une restriction drastique de la structure des modules qui les réalisent, puisqu'aucun automate ne peut influencer un automate dont la fonction de sortie est de délai 1.

Propriété 17. *Soit (M, T) tel que les fonctions de sortie de T soient toutes de délai 1. La classe $C_{(M,T)}$ ne contient que des paires (M', T') égales à (M, T) au renommage des automates et des entrées près.*

Démonstration. Soit M un module et T un ensemble de sorties de M tels que la fonction de sortie de tout s dans T a pour délai 1. Puisque le délai de ces fonctions de sortie est 1, cela implique que les automates dans T ne sont influencés par aucun autre automate dans M. Donc, par la définition d'ensemble de sorties, il n'y a aucun automate dans M à part T. Cela est vérifié pour toute paire (M', T') telle que $(M', T') \equiv_O (M, T)$. De plus, comme toutes les fonctions de sortie de M et M' sont équivalentes, toutes les fonctions locales de M et M' sont également équivalentes, au renommage des automates et entrées près. Cela s'applique pour l'entièreté de la classe $C_{(M,T)}$. □

Chacune des classes d'équivalence non triviales fait preuve d'une très grande liberté de structure parmi ses modules. En effet, pour M un module acyclique et T un ensemble de sorties de M, il existe des paires (M', T') équivalentes en sortie à (M, T) pour M' un module de taille arbitraire, et ce malgré la restriction qui fait de T un ensemble de sortie.

Propriété 18. *Soit M un module et T un ensemble de sorties. Pour tout entier N, il existe (M', T') tel que $(M, T) \equiv_O (M', T')$ et $|M'| > N$.*

Démonstration. Soit (M, T) une paire telle que M possède au moins une fonction de sortie O_s avec délai au moins 2. Cela implique qu'il existe un ensemble d'automates non vide qui influence l'automate s. Soit s' un tel automate. Nous pouvons produire une paire (M', T) équivalente en sortie à (M, T), telle que $|M| > |M'|$. Pour ce faire, on construit le module M' comme une copie du module M, auquel on rajoute un automate s'' dont la fonction locale est une copie de la fonction locale de l'automate s'. De plus, dans la fonction locale de l'automate s du module M', on substitue chaque instance de la variable $x_{s'}$ par $(x_{s'} \wedge x_{s''})$. Puisque $f_{s'} = f_{s''}$, on a $O_{s'} = O_{s''}$, ce qui implique que l'évaluation de la fonction de sortie de s dans M' reste équivalente à celle s dans M. Nous avons ainsi créé un module disposant d'un automate supplémentaire et équivalent en sortie au module initial. □

6.3 Modules 1-pour-1

Parmi les nombreuses classes d'équivalence qui organisent les modules acycliques et leurs sorties, nous nous intéressons ici aux classes produites par les modules acycliques qui ne définissent qu'une seule entrée et qu'une seule sortie, ou $|I| = |T| = 1$. Comme nous allons le voir, ces cas particuliers représentent un grand nombre de familles de réseaux d'automates.

L'ensemble des réseaux d'automates qui peuvent se décomposer simplement en un module 1-pour-1 sont facilement caractérisés comme les réseaux qui possèdent un automate en particulier qui, une fois retiré, laisse le graphe d'interaction du réseau acyclique.

Propriété 19. *Soit F un réseau d'automates. Les deux propositions suivantes sont équivalentes :*

— *Le cardinal d'un feedback vertex set minimal du graphe d'interaction de F est 1 ou moins.*
— *Il existe un module 1-pour-1 M et une fonction de branchement $\mathtt{w}$ tels que $\circlearrowleft_{\mathtt{w}} M = F$.*

Démonstration. Supposons que le graphe d'interaction de F dispose d'un feedback vertex set de taille 0. Alors F est déjà un module acyclique, et l'on peut rajouter une entrée redondante α pour définir un module 1-pour-1 M. Pour $\mathtt{w}$ la fonction de sortie définie sur α qui y associe une valeur arbitraire, on observe $\circlearrowleft_{\mathtt{w}} M = F$.

Supposons que le graphe d'interaction de F dispose d'un feedback vertex set de taille 1. Soit $s \in S$ l'automate concerné. On construit le module M défini sur l'ensemble des automates de F et sur l'ensemble d'entrées $I = \{\alpha\}$ tel que pour tout automate s', la fonction locale $f'_{s'}$ dans M est définie par $f'_{s'}(x' \cdot i') = f_s((x' \cdot i') \circ g)$, où g est la fonction $g : \{S\} \to \{S \cup I\}$ telle que

$$g(r) = \begin{cases} \alpha & \text{si } r = s \\ r & \text{sinon} \end{cases}.$$

Nous observons que, dans M, aucun automate n'est influencé par s. En effet, l'influence de la variable s a été remplacée par g par l'influence de l'entrée α. Nous en concluons que M est acyclique : en effet, par hypothèse, tout cycle dans le graphe d'interaction de M doit passer par s. Or, comme aucun sommet n'est influencé par s, aucun cycle n'existe dans le graphe d'interaction de M. Le module M dispose d'une seule entrée, α et d'une seule sortie, s. Pour conclure cette inférence, nous construisons la fonction de branchement $\mathtt{w} : \{\alpha\} \to S$ telle que $\mathtt{w}(\alpha) = s$. Le module $\circlearrowleft_{\mathtt{w}} M$ n'a pas d'entrée et dispose des automates S, et pour tout $s' \in S$, sa fonction locale $f''_{s'}$ vérifie

$$f''_{s'}(x'') = f'_{s'}(x'' \circ \hat{\mathtt{w}})$$
$$f''_{s'}(x'') = f_{s'}(x'' \circ \hat{\mathtt{w}} \circ g).$$

On observe que $\hat{\mathtt{w}} \circ g(s) = \hat{\mathtt{w}}(\alpha) = s$ et que, pour tout s' différent de s, $\hat{\mathtt{w}} \circ g(s') = \hat{\mathtt{w}}(s') = s'$, et $\hat{\mathtt{w}} \circ g$ est donc la fonction identité sur S. En conclusion, $f''_{s'}(x'') = f_{s'}(x'')$ pour tout $s' \in S$, et $\circlearrowleft_{\mathtt{w}} M = F$.

Supposons maintenant qu'il existe un module 1-pour-1 M et une fonction de branchement $\mathtt{w}$ tels que $\circlearrowleft_{\mathtt{w}} M = F$. Puisque le module M n'a qu'une seule entrée et une seule sortie, les domaine et image de $\mathtt{w}$ sont des singletons. Soit s l'automate inclus dans cette image. On peut définir une fonction g définie comme précédemment qui transforme F en M. Puisque M est acyclique, cela signifie que tout cycle dans le graphe d'interaction de F passe par s. Supposons qu'il existe un tel cycle : alors le graphe d'interaction de F dispose d'un feedback vertex set de taille 1 qui contient s. Si aucun tel cycle n'existe, alors, par définition, le graphe d'interaction de F dispose d'un feedback vertex set de taille 0. □

Ces modules ont la particularité de posséder une dynamique bien plus facile à pré-

dire, ce qui est illustré par les problèmes suivants, conçus comme des cas particuliers des problèmes développés en fin de section 6.2.3.

▸ ONE-TO-ONE MODULE ATTRACTOR PROBLEM
Entrée: Un module acyclique booléen M, une seule entrée et n automates, une fonction de branchement `w` et un nombre c encodé en unaire.
Promesse: L'encodage des fonctions locales de M ne comporte que des variables essentielles.
Question: Existe-t-il un attracteur de taille c dans la dynamique parallèle du réseau d'automates $\circlearrowleft_{\mathtt{w}} M$?

▸ ONE-TO-ONE MODULE FIXED POINT PROBLEM
Entrée: Un module acyclique booléen M, une seule entrée et n automates et une fonction de branchement `w`.
Promesse: L'encodage des fonctions locales de M ne comporte que des variables essentielles.
Question: Existe-t-il une configuration x telle que $\circlearrowleft_{\mathtt{w}} M(x) = x$?

Ces problèmes sont associés aux théorèmes suivants.

Théorème 11 (PERROT, PERROTIN et SENÉ 2020a)**.** *Le problème One-to-one Module Attractor est NP-complet.*

Démonstration. Ce problème est dans NP car, depuis toute configuration, on peut vérifier que cette configuration fait partie d'un attracteur de taille c en mettant à jour le réseau c fois, et en vérifiant que la configuration obtenue est la même que la configuration initiale.

Pour en prouver la NP-difficulté, nous présentons une réduction depuis le problème SAT. Pour f une formule à m variables nommées $v_1, \dots, v_m$, on construit un module 1-pour-1 avec $m+e+1$ automates (pour e borné par une $2m$) tel que, lorsque la sortie de ce module est branchée à son entrée, sa dynamique dispose d'un attracteur de taille $c = m+e+1$ si et seulement si f est satisfaisable.

Le module 1-pour-1, nommé M, est composé de deux parties. La première partie est un *ruban*, composé de $m+e$ automates nommés $t_1, \dots, t_{m+e}$ avec e le plus petit nombre tel que $m+e+1$ est un nombre premier (la valeur de e peut être calculée efficacement par l'algorithme décrit par (AGRAWAL, KAYAL et SAXENA 2004)). Pour tout $1 < k \leq m+e$, on définit la fonction locale $f_{t_k}(x) = t_{k-1}$, et $f_{t_1}(x) = \alpha$ avec α l'entrée unique du module. Pour $i \in \{1, \dots, m\}$, l'état de l'automate t_i encode l'évaluation de la variable v_i.

La seconde partie du réseau est composée d'un automate unique nommé q, qui est la sortie du réseau sur laquelle `w` branche l'entrée α. Cet automate a le rôle de laisser le ruban de taille $m+e$ devenir un ruban rotatif de taille $m+e+1$, ou d'arrêter le processus et de forcer le réseau à converger vers le point fixe 0^{m+e+1}. Sa fonction

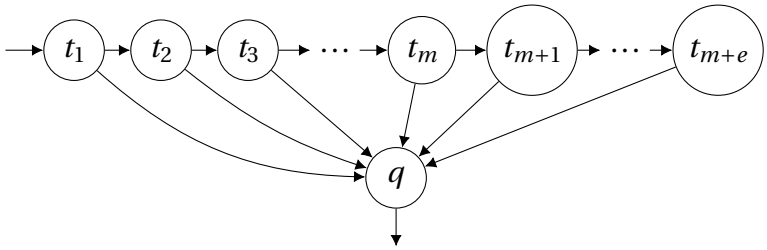

FIGURE 6.5 – Illustration du module M développé dans la preuve du théorème 11. Si f dispose d'une évaluation positive, alors l'automate q permet au ruban de taille $m+e$ de devenir un ruban rotatif de taille $m+e+1$, et sinon f_q s'évalue à 0 et toute configuration converge vers le point fixe 0^{m+e}.

locale est comme suit :

$$f_q(x) = \begin{cases} x_{t_{m+e}} & \text{si les automates } t_1,\dots,t_m \text{ encodent} \\ & \qquad \text{une évaluation positive de } f \\ & \text{ou si un décalage de cette configuration encode} \\ & \qquad \text{une évaluation positive de } f, \\ 0 & \text{sinon.} \end{cases}$$

Puisque le module M est acyclique, l'automate q ne peut qu'indirectement connaître son propre état par le branchement `w`. Après ce branchement, le sommet q possède l'information de l'entièreté du réseau, et peut donc décider si un certain décalage du ruban permet d'évaluer la formule f à vrai. Par conséquent, s'il existe une évaluation positive de f, alors il existe une configuration du réseau qui se met en rotation et en découle un attracteur de taille $\mathcal{O}(m)$. Si le réseau dispose d'un attracteur de taille supérieure à 1, alors, par la nature première de la taille du réseau, cet attracteur sera de taille $m+e+1$ et décrira une évaluation positive de f. La formule f est insatisfaisable si et seulement si toute configuration du réseau converge vers le point fixe 0^{m+e+1}.

Cette construction est illustrée par la figure 6.5, et est de taille polynomiale. □

Corollaire 4 (PERROT, PERROTIN et SENÉ 2020a)**.** *Le problème One-to-one Module Fixed Point est dans P.*

Démonstration. Ce résultat est une application du théorème 8. □

6.3.1 Les registres à décalage à rétroaction linéaire comme des modules 1-pour-1

Les registres à décalage à rétroaction linéaire sont des modèles de circuits étudiés principalement pour leur capacité à générer des séquences pseudo-aléatoires. Ces circuits disposent d'une rangée de registres qui retiennent une valeur booléenne. Lors

de la mise à jour, l'ensemble des valeurs est déplacée vers l'avant : la nouvelle valeur du premier registre est calculée par une fonction linéaire de l'état général du système.

Ces objets sont facilement modélisés par des réseaux d'automates. Nous définissons l'ensemble des réseaux d'automates qui les modélisent comme suit.

Définition 72 (Réseau d'automates à décalage à rétroaction)**.** *Un réseau d'automates est un* réseau d'automates à décalage à rétroaction *si et seulement si son réseau d'interaction dispose d'un automate s dont la fonction locale est arbitraire, et dont le reste des automates peut être décrit comme une séquence $(s_1, s_2, \ldots, s_{n-1})$ qui vérifie que $f_{s_1}(x) = x_s$, et $f_{s_i}(x) = x_{s_{i-1}}$ pour tout i tel que $1 < i < n$.*

Nous ne nommons pas ces objets "réseaux d'automates à décalage à rétroaction *linéaire*" car ce "*linéaire*" signifie, dans le cas des registres du même nom, que la fonction de réatroaction est une fonction linéaire. Les résultats qui suivent ne font sens que si l'on se permet de considérer toute fonction de rétroaction, et cela justifie notre appellation.

La caractérisation de ces réseaux par le biais de modules 1-pour-1 est triviale, et repose sur le remplacement de l'influence de l'automate s par une l'influence d'une entrée.

Propriété 20. *Soit F un réseau d'automates à décalage à rétroaction. Il existe M un module ayclique 1-pour-1 et une fonction de branchement $\mathtt{w}$ telle que $F = \circlearrowleft_{\mathtt{w}} M$.*

Démonstration. On observe que le réseau d'interaction de F dispose toujours d'un feedback vertex set défini par $\{s\}$, car tout cycle de ce graphe passe trivialement par s. La propriété est obtenue par application de la propriété 19. □

Si cette propriété permet d'observer que les réseaux d'automates à décalage à rétroaction sont bien caractérisés par les modules 1-pour-1, nous pouvons également démontrer que la fonction de sortie de tout module 1-pour-1 peut être réalisée par un réseau d'automates à décalage à rétroaction. Nous décrivons ce fait par la propriété suivante.

Propriété 21. *Soit O une fonction de sortie définie sur une unique entrée. Il existe un module 1-pour-1 M et une fonction de branchement $\mathtt{w}$ tels que $\circlearrowleft_{\mathtt{w}} M$ est un réseau d'automates à décalage à rétroaction et la fonction de sortie de M est égale à O.*

Démonstration. Soit O une fonction de sortie définie sur l'entrée α, et d le délai de O. On peut facilement composer M un module qui réalise O de la façon suivante. Le module M est défini sur les automates $S = \{s, s_1, s_2, \ldots, s_{d-1}\}$ et l'entrée $I = \{\alpha\}$ tels que $(s_1, s_2, \ldots, s_{d-1})$ est une séquence d'automates dont les fonctions locales vérifient $f_{s_1}(x) = i_\alpha$, et $f_{s_i}(x) = x_{s_{i-1}}$ pour tout i tel que $1 < i < d$. La fonction locale de l'automate s est définie comme la fonction O dans laquelle toute instance de la variable α_1 est substituée par la variable i_α, et pour tout k tel que $1 < k \leq d$, la variable α_d est substituée par la variable $x_{s_{k-1}}$. Soit $\mathtt{w} : \{\alpha\} \to S$ la fonction qui vérifie $\mathtt{w}(\alpha) = s$.

On observe que $\circlearrowleft_{\mathtt{w}} M$ est un réseau d'automate à décalage à rétroaction linéaire. De plus, la fonction de sortie de l'automate s dans M réalise par construction la fonction de sortie O : cela se voit facilement par le fait que la fonction de sortie de l'automate x_k se trouve être $O_{x_k}(J) = \alpha_k$. □

Cette propriété exprime pour nous une forme d'universalité intrinsèque des réseaux d'automates à décalage à rétroaction dans la famille plus large des réseaux d'automates obtenus par le branchement récursif d'un module 1-pour-1. En effet, pour tout réseau obtenu par le branchement récursif d'un module 1-pour-1, il existe par l'application de la propriété 20 un réseau d'automates à décalage à rétroaction qui définit une fonction de sortie équivalente. Par application du théorème 10, ce réseau d'automates à décalage à rétroaction disposera d'attracteurs isomorphes. Cette notion d'universalité présuppose que le calcul pertinent d'un réseau d'automates est l'ensemble de ses attracteurs.

6.3.2 Les fleurs d'automates comme des modules 1-pour-1

Parmi les autres familles incluses dans cette classe, nous pouvons nommer les ⊕-BAF, ou fleurs d'automates booléens localement non-monotones telles que définies dans (ALCOLEI, PERROT et SENÉ 2016). Nous reproduisons ici une définition plus générale que la définition initiale donnée dans (DIDIER et REMY 2012) :

Définition 73 (Fleur d'automates)**.** *Un réseau d'automate est une* fleur *si et seulement si son graphe d'interaction est connexe, et si tous ses sommets sauf un ont une arité entrante et sortante de un.*

Les fleurs ne sont pas des modules acycliques par cette définition. Il existe cependant une transformation intuitive de ces réseaux en modules acycliques 1-pour-1.

Propriété 22. *Soit F une fleur d'automates. Il existe M un module ayclique 1-pour-1 et une fonction de branchement* $\mathtt{w}$ *telle que $F = \circlearrowleft_{\mathtt{w}} M$.*

Démonstration. Soit F une fleur d'automates et soit s le sommet dont l'arité sortante et entrante n'est pas un. On observe que l'ensemble $\{s\}$ est un feedback vertex set du graphe d'interaction de F. Pour le voir, supposons qu'il existe un cycle dans le graphe d'interaction de F qui ne contient pas s. Par définition, tous les sommets de ce cycle ont pour arité entrante et sortante un. Cela signifie que ce cycle est fermé du reste du réseau, et donc que ce cycle et le sommet s sont sur des sous-graphes non connectés, ce qui est en contradiction avec l'hypothèse que le graphe d'interaction de F est connexe.

Le graphe d'interaction d'une fleur d'automates définit toujours un feedback vertex set de taille 1. Nous déduisons le résultat par application de la propriété 19. □

On remarque que les fleurs d'automates sont une généralisation des réseaux d'automates à décalage à rétroaction. Cela permet d'obtenir trivialement que les fleurs

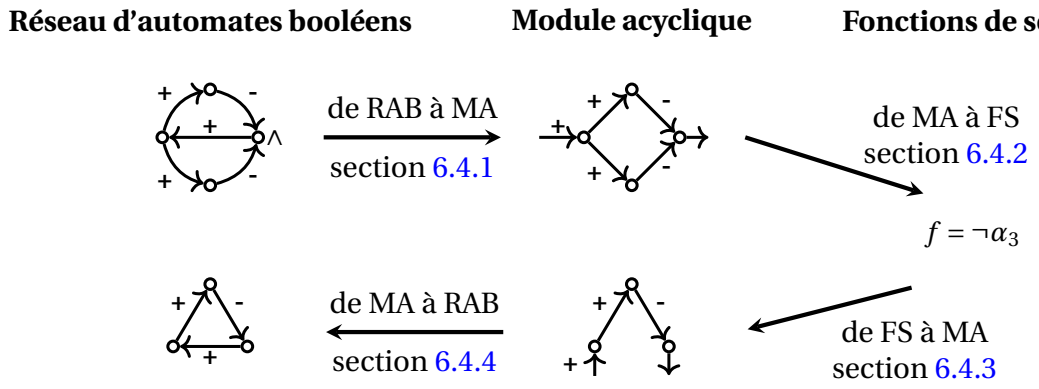

FIGURE 6.6 – Illustration de la méthode d'optimisation de module acyclique, qui permet l'optimisation de réseaux d'automates booléens pour le calcul de leur dynamique. Chaque étape de cette méthode est représentée par une flèche, qui est annotée de la section qui la décrit.

d'automates sont aussi intrinsèquement universelles que les réseaux d'automates à décalage à rétroaction.

Propriété 23. *Soit O une fonction de sortie définie sur une unique entrée. Il existe un module 1-pour-1 M et une fonction de branchement $\mathtt{w}$ tels que $\circlearrowleft_{\mathtt{w}} M$ est une fleur d'automates et la fonction de sortie de M est égale à O.*

Démonstration. Puisque tout réseau d'automates à décalage à rétroaction est une fleur d'automates, on obtient le résultat simplement par application de la propriété 21. □

6.4 Optimisation d'un module acyclique

La définition de classes d'équivalence nous pousse à l'observation d'une propriété naturelle. Pour chaque classe, il existe un ensemble de modules de cette classe qui ont une taille minimale. Cette observation nous inspire un processus d'optimisation de réseaux d'automates, qui repose sur une méthode d'optimisation de modules acycliques. Pour F un réseau d'automates, cette méthode procède par les étapes suivantes, qui sont également représentées en figure 6.6 :

1. Construire un module acyclique M tel que $\circlearrowleft_{\mathtt{w}} M = F$, pour $\mathtt{w}$ une fonction de branchement.
2. Extraire T l'ensemble de sorties qui permet la recomposition de F.
3. Trouver (M', T') tel que $(M, T) \equiv_O (M', T')$ et M' minimal en taille.
4. Recomposer M' par un branchement équivalent à $\mathtt{w}$, au renommage des automates et des entrées près.

Il ressort de cette méthode et pour tout réseau d'automate F un nouveau réseau F', de taille possiblement plus petite, et qui, par définition de l'équivalence en sortie, possède des attracteurs isomorphes aux attracteurs de F. Ainsi, cette méthode est motivée par le calcul des attracteurs d'un réseau d'automate, qui est une procédure au coût exponentiel en la taille du réseau. Nous proposons d'explorer les détails de cette méthode qui permet de réduire dans certains cas la taille des réseaux étudiés tout en préservant leurs attracteurs en dynamique parallèle.

Pour se faire, nous présentons chaque étape de cette méthode par le biais d'un problème fonctionnel, accompagné par notre analyse de la complexité de ce problème, lorsqu'applicable. Ces problèmes de complexité sont, pour la plupart, des problèmes fonctionnels avec promesse, dont la définition est détaillée en section 1.8.6. Cette promesse consiste à exprimer que les fonctions locales des modules et réseaux d'automates passés en entrée de ces modèles sont encodés de façon à n'avoir que des variables essentielles. Cela veut dire que l'ensemble des automates et entrées qui influencent la mise à jour d'un automate donné sont exactement les automates et entrées dont les variables sont présentes dans l'encodage de la fonction locale de cet automate. Cette hypothèse est forte : en effet, sa vérification est un procédé coNP-difficile. L'utilité de cette promesse est que le graphe d'interaction du réseau considéré peut alors être dessiné en temps polynomial.

Nous justifions l'utilisation de cette promesse en deux points : premièrement, la procédure d'optimisation que nous développons prend pour contexte les applications des réseaux d'automates booléens dans un contexte biologique ; dans ces applications, les réseaux étudiés sont créés de façon qui ne permet pas l'apparition de variables redondantes. Secondement, l'emploi de cette promesse nous permet l'observation de problèmes difficiles, et l'exhibition de cette difficulté sans avoir à considérer le cas particulier des variables redondantes nous semble pertinent pour comprendre les limites de notre méthode.

Les exemples suivants accompagnent la définition de notre méthode et illustrent chaque étape qui la compose.

Exemple 50. *Pour $S_A = \{a, b, c, d\}$, soit F_A le réseau d'automates booléens défini par les fonctions locales $f_a(x) = x_d$, $f_b(x) = f_c(x) = x_a$, et $f_d(x) = \neg x_b \vee \neg x_c$. Le graphe d'interaction de F_A est représenté en figure 6.7.*

Exemple 51. *Pour $S_B = \{St, Sl, Sk, Pp, Ru, S9, C, C25, M, C^*\}$, soit F_B le réseau d'automates booléens défini par les fonctions locales $f_{St}(x) = \neg x_{St}$, $f_{Sl}(x) = \neg x_{Sl} \vee x_{C^*}$, $f_{Sk}(x) = x_{St} \vee \neg x_{Sk}$, $f_{Pp}(x) = x_{Sl} \vee \neg x_{Pp}$, $f_{Ru}(x) = f_{S9}(x) = \neg x_{Sk} \vee x_{Pp} \vee \neg x_C \vee \neg x_{C^*}$, $f_C(x) = \neg x_{Ru} \vee \neg x_{S9} \vee \neg x_{Sl}$, $f_{C25}(x) = \neg x_{Pp} \vee x_C$, $f_M(x) = x_{Pp} \vee \neg x_C$, and $f_{C^*}(x) = \neg x_{Ru} \vee \neg x_{S9} \vee x_{C25} \vee \neg x_M$. Le graphe d'interaction de F_B est représenté en figure 6.8.*

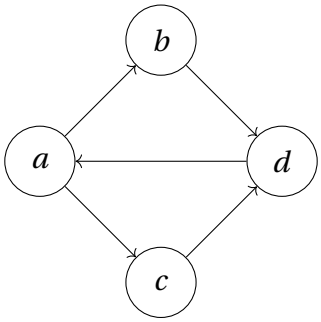

FIGURE 6.7 – Graphe d'interaction du réseau d'automates F_A défini dans l'exemple 50.

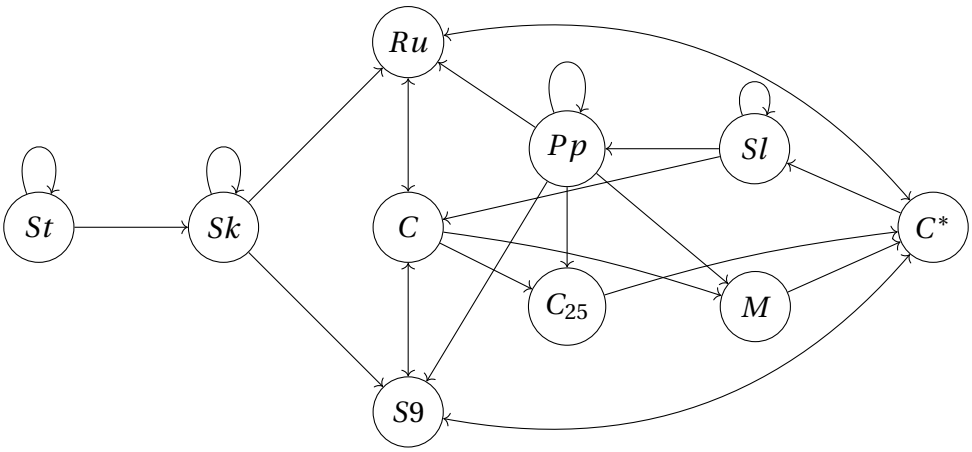

FIGURE 6.8 – Graphe d'interaction du réseau d'automates F_B défini dans l'exemple 51.

6.4.1 Transformation d'un réseau d'automate en module acyclique

La première étape de notre processus consiste à déplier, ou découper, transformer un réseau d'automates F en un module acyclique. Nous formalisons cette étape par le biais d'un problème fonctionnel.

▸ FUNCTIONAL ACYCLIC UNFOLDING PROBLEM

Entrée: Un réseau d'automates F et un entier k.
Promesse: L'encodage des fonctions locales de F ne comporte que des variables essentielles.
Sortie: Un module acyclique M avec au plus k entrées et une fonction de branchement récursif $\mathtt{w}$ tels que $\circlearrowleft_{\mathtt{w}} M = F$.

Théorème 12 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème Functional Acyclic Unfolding est dans FNP.*

Démonstration. Rappellons que la promesse de ce problème permet le calcul du

graphe d'interaction de F en temps polynomial.

Considérons l'algorithme non déterministe suivant : en première étape, deviner M et $\mathtt{w}$; vérifier ensuite que le nombre d'entrées de M n'est pas supérieur à k, et que le module $\circlearrowleft_{\mathtt{w}} M$ est syntaxiquement égal à F.

Cet algorithme est calculable en temps polynomial non déterministe car un branchement récursif n'opère qu'une simple substitution de variables. Peu importe la façon dont les fonctions locales de F sont encodées en entrée, on pourra choisir M et $\mathtt{w}$ tels que les fonctions locales de $\circlearrowleft_{\mathtt{w}} M$ sont encodées de façon identique à F. □

Théorème 13 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème Functional Acyclic Unfolding est NP-difficile.*

Démonstration. Nous procédons à une réduction many-one polynomiale depuis le problème Feedback Vertex Set. Soit f une fonction qui, pour tout graphe G et tout nombre k tel que (G,k) est une instance du problème Feeback Vertex Set, donne une instance (F,k) du problème Functionnal Acyclic Unfolding telle que $S = V(G)$, et f_s est une fonction OR de tous les automates s' dans S tels que (s',s') est une arête de G. Cette construction implique que le graphe d'interaction et le graphe G soient identiques, et est clairement calculable en temps polynomial. On en déduit que l'instance (G,k) est positive si et seulement si (F,k) possède une solution.

Cette réduction fonctionne sur l'observation que remplacer l'influence d'un automate par une entrée dans un module retire du graphe d'interaction tous les cycles qui passent par cet automate. En effet, une fois que l'influence de l'automate est remplacée par une entrée sur l'ensemble du réseau, cet automate n'a plus aucune arête sortante dans le graphe d'interaction. En termes de l'existence de cycles dans le graphe d'interaction, ce remplacement d'influence est équivalent à retirer l'automate du module. □

Cette transformation est toujours possible grâce à l'application de la propriété 14. Afin d'assurer que le module obtenu n'aura qu'un nombre minimal de sorties, nous procédons ici par réduction vers le problème Feedback Vertex Set.

En sortie de ce problème, nous obtenons M et $\mathtt{w}$. L'ensemble de sorties T considéré est simplement l'ensemble des sommets dont l'influence a été remplacée par des entrées, ce qui correspond à l'ensemble $\mathrm{codom}(\mathtt{w})$.

Exemple 52. *Nous considérons S_A et F_A de l'exemple 50. Pour $I_A = \{\alpha\}$, soit M_A le module acyclique défini par les fonctions locales $f'_a(x) = x_\alpha$, $f'_b(x) = f'_c(x) = x_a$, et $f'_d(x) = \neg x_b \vee \neg x_c$. Le module M_A est une réponse valide à l'instance $F_A, k = 1$ du problème Functional Acyclic Unfolding, et il n'existe pas de solution pour $k = 0$. Le graphe d'interaction du module M_A est représenté en figure 6.9.*

Exemple 53. *Nous considérons S_B et F_B de l'exemple 51. Pour $I_B = \{\alpha_{St}, \alpha_{Sl}, \alpha_{Sk}, \alpha_{Pp}, \alpha_C, \alpha_{C^*}\}$, soit M_B le module acyclique défini par les fonctions locales $f'_{St}(x) = \neg x_{\alpha_{St}}$, $f'_{Sl}(x) = \neg x_{\alpha_{Sl}} \vee x_{\alpha_{C^*}}$, $f'_{Sk}(x) = x_{\alpha_{St}} \vee \neg x_{\alpha_{Sk}}$, $f'_{Pp}(x) = x_{\alpha_{Sl}} \vee \neg x_{\alpha_{Pp}}$, $f'_{Ru}(x) = f_{S9}(x) = \neg x_{\alpha_{Sk}} \vee x_{\alpha_{Pp}} \vee \neg x_{\alpha_C} \vee \neg x_{\alpha_{C^*}}$, $f'_C(x) = \neg x_{Ru} \vee \neg x_{S9} \vee \neg x_{\alpha_{Sl}}$, $f'_{C25}(x) = \neg x_{\alpha_{Pp}} \vee x_{\alpha_C}$,*

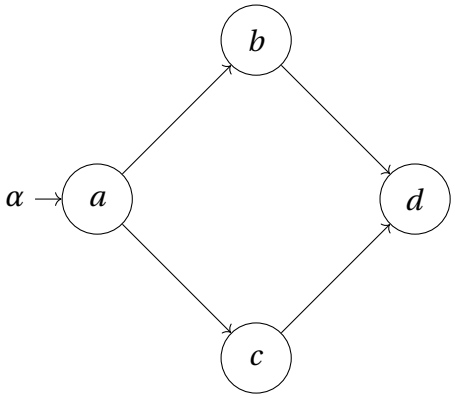

FIGURE 6.9 – Graphe d'interaction du module M_A défini dans l'exemple 52.

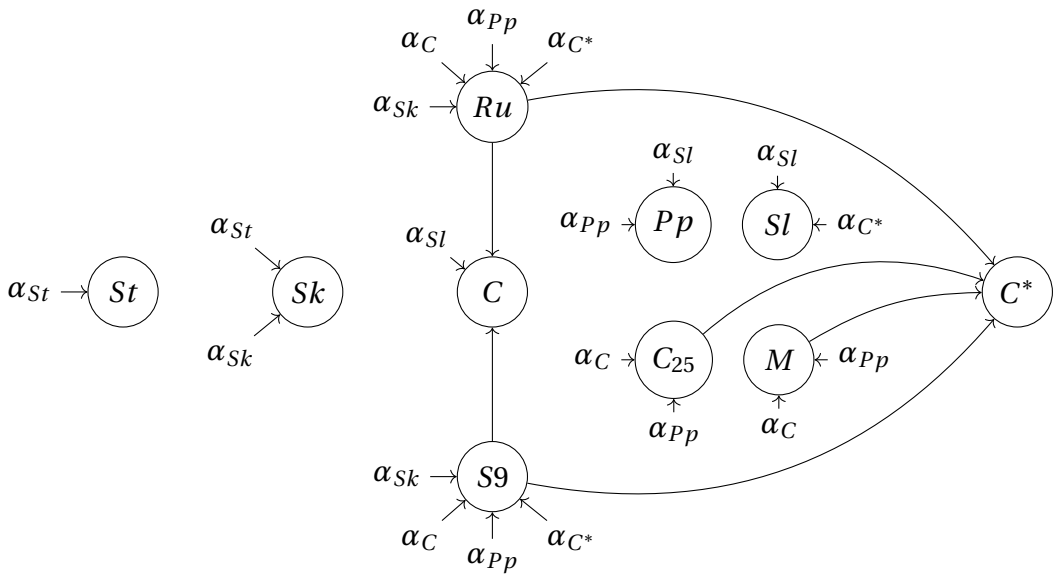

FIGURE 6.10 – Graphe d'interaction du module M_B défini dans l'exemple 53.

$f'_M(x) = x_{\alpha_{Pp}} \vee \neg x_{\alpha_C}$, *et* $f'_{C^*}(x) = \neg x_{Ru} \vee \neg x_{S9} \vee x_{C25} \vee \neg x_M$. *Le module* M_B *est une réponse valide à l'instance* $F_B, k = 6$ *du problème Functional Acyclic Unfolding, et il n'existe pas de solution pour* $k < 6$. *Le graphe d'interaction du module* M_B *est représenté en figure 6.10.*

6.4.2 Calcul efficace des fonctions de sortie

Soit (M, T) la paire d'éléments obtenus en sortie du problème précédent. Afin d'obtenir une paire (M', T') possiblement de plus petite taille, nous procédons en deux étapes. La première consiste à calculer les fonctions de sortie des automates

T du module M sous la forme de circuits booléens. La seconde consiste à générer un module minimal M' qui calcule les même fonctions de sortie sur un ensemble d'automates T'.

Comme discuté en section 6.2.2, nous encodons les fonctions de sortie par des circuits booléens car cela nous permet de les calculer en temps polynomial, là où les encoder par des formules booléennes ne nous donne pas cette assurance.

La première partie de la minimisation du module est le calcul de ses fonctions de sortie, que nous caractérisons par le problème fonctionnel suivant.

▸ OUTPUT CIRCUIT COMPUTATION PROBLEM

Entrée: Un module acyclique M avec $X \subseteq S$ un sous-ensemble de ses automates.

Promesse: L'encodage des fonctions locales de M ne comporte que des variables essentielles.

Sortie: Une fonction de sortie pour chaque automate de X, encodée sous la forme d'un circuit booléen.

Ainsi formulée, cette partie du processus d'optimisation peut être calculée en temps polynomial.

Théorème 14 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème Output Circuit Computation est dans FP.*

Démonstration. La promesse de ce problème permet la construction du graphe d'interaction de M en temps polynomial. Le résultat est obtenu par application de la propriété 16. □

Exemple 54. *Nous considérons le module M_A de l'exemple 52. Soit $M_A, X_A = \{d\}$ une instance du problème Output Circuit Computation. Le circuit $O_d = \neg\alpha_3$ est une réponse valide à cette instance.*

Exemple 55. *Nous considérons le module M_B de l'exemple 53. Soit $M_B, X_B = \{St, Sk, Sl, Pp, C, C^*\}$ une instance du problème Output Circuit Computation. L'ensemble des circuits $O_{St} = \neg\alpha_{St,1}$, $O_{Sl} = \neg\alpha_{Sl,1} \vee \alpha_{C^*,1}$, $O_{Sk} = \alpha_{St,1} \vee \neg\alpha_{Sk,1}$, $O_{Pp} = \alpha_{Sl,1} \vee \neg\alpha_{Pp,1}$, $O_C = (\alpha_{Sk,2} \wedge \neg\alpha_{Pp,2} \wedge \alpha_{C,2} \wedge \alpha_{C^*,2}) \vee \neg\alpha_{Sl,1}$ et $O_{C^*} = \alpha_{C,2} \vee \neg\alpha_{Pp,2}$ est une solution valide à cette instance.*

6.4.3 Génération d'un module acyclique optimal

Une fois que nous obtenons une collection de fonctions de sortie pour le sous-ensemble T du module M, nous pouvons nous intéresser à contruire un module acyclique optimal qui réalise ces fonctions de sortie. Cette question est caractérisée par le problème suivant.

▸ MODULE SYNTHESIS PROBLEM
Entrée: Un ensemble I d'entrées, un ensemble de fonctions de sortie encodées par des circuits booléens O définies sur les entrées I et k un entier.
Sortie: Un module acyclique M avec au plus k automates tel que chaque fonction de O est une fonction de sortie d'au moins un automate de M.

Notre compréhension de ce problème ne nous permet pour l'instant que de dessiner des bornes de complexité assez lâches. Il est en effet très facile de prouver les résultats suivants.

Théorème 15 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème Module Synthesis est coNP-difficile.*

Démonstration. On considère f une formule instance du problème Tautology, avec I l'ensemble de variables propositionnelles contenues dans f. Soit f' la fonction de sortie obtenue à partir de f en remplaçant chaque instance d'une variable α par l'équivalent de délai 1, α_1. Nous définissons également $\mathbf{1}$ la fonction de sortie constante à 1. Nous composons $O = \{f', \mathbf{1}\}$ et $k = 1$. Cette instance a une solution si et seulement si ces deux fonctions de sortie peuvent être réalisées par un seul automate, autrement si et seulement si f' est une tautologie. Cette instance peut clairement être construite en temps polynomial, et toute solution positive de cette instance implique que f est une tautologie. □

Théorème 16 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème Module Synthesis est dans* FNP^{coNP}.

Démonstration. On considère l'algorithme suivant. Premièrement, deviner un module acyclique M de taille k. Calculer ensuite chaque fonction de sortie du module en temps polynomial. Vérifier ensuite que chaque fonction dans O est équivalente à au moins une de ces fonctions de sortie, ce qui peut être fait avec au plus $|M| \times |O|$ appels à un oracle coNP. □

L'avancée actuelle de notre travail ne nous permet pas de raffiner cette borne de complexité plus précisément. Bien que les deux bornes ici formalisées se trouvent être des résultats relativement simples, les tentatives de preuve de raffinement de ces bornes que nous avons explorées n'ont retourné aucun résultat.

Question ouverte 2. *Quelle est la complexité précise du problème Module Synthesis ?*

Prouver que ce problème est dans FcoNP nécessite une réalisation de ce problème par un algorithme de même complexité. Un tel algorithme dispose des objets I, O et k en entrée, et a la capacité de construire de façon non déterministe un objet de taille polynomiale en la taille de ces entrées, puis pourra vérifier en temps polynomial déterministe que cet objet implique que l'instance n'a pas de solution valide. Cependant, résoudre le problème Module Synthesis nécessite au moins la génération d'un module acyclique M, à partir duquel il est un problème coNP de vérifier que ce module réalise

les bonne fonctions de sortie car, en effet, il s'agit de la complexité du problème de l'égalité de deux circuits booléens. Il nous semble raisonnable de conclure que tout calcul d'une solution en temps FcoNP semble impossible.

Une autre possibilité de preuve est un algorithme qui procède en générant une solution naïve, et en optimisant cette solution par la fusion de ses automates deux à deux. Cette approche nécessite cependant la résolution d'un problème de la forme suivante.

▶ MODULE LOCAL FUSION PROBLEM

Entrée: Une module acyclique M et a, b deux automates différents de M.

Question: Existe-t-il une fonction locale f_c telle qu'il existe un module acyclique M' défini sur les automates $S \cup \{c\} \setminus \{a, b\}$, tel que $f_c \equiv f'_c$ et $O_s \equiv O'_s$ pour tout s dans $S \setminus \{a, b\}$?

La résolution d'un ensemble d'instances du problème Module Local Fusion est une étape nécessaire pour tout algorithme qui optimise un module par la fusion de ses automates. Ce problème est au moins un problème dans $\mathrm{NP}^{\mathrm{coNP}}$. En effet, son énoncé repose sur le problème de deviner un nouveau module M', tel que ce module vérifie l'égalité des fonctions f_c et f'_c, ainsi que l'égalité des fonctions de sortie, ce qui, en soi, est un ensemble de problèmes coNP-difficiles. Un algorithme reposant sur la fusion ne pourra donc pas être d'une complexité inférieure à $\mathrm{NP}^{\mathrm{coNP}}$.

Prouver que ce problème est $\mathrm{NP}^{\mathrm{coNP}}$-difficile, aussi appelé $\Sigma_2 P$-difficile, requiert la réduction d'un problème $\Sigma_2 P$-complet vers le problème Module Synthesis. Un exemple classique de problème $\Sigma_2 P$-complet est le problème de la satisfaisabilité d'une formule $\exists x_1, x_2, \ldots, \forall y_1, y_2, \ldots, \Phi(x, y)$, pour Φ une formule booléenne. Nous n'avons pas d'intuition sur la façon dont le problème Module Synthesis permettrait l'encodage d'un problème aussi complexe.

Nous concluons cette discussion sur la complexité évasive du problème Module Synthesis en le rapprochant à d'autres problèmes en surface très similaires. Un problème similaire est le problème Circuit Minimisation, qui repose sur le calcul d'un circuit booléen de taille optimale qui calcule une table de vérité donnée en entrée. Ce problème est trivialement prouvé d'être dans NP, mais la connaissance actuelle du domaine ne permet pas de décider s'il s'agit d'un problème dans P ou d'un problème NP-complet.

Bien que les problèmes Circuit Minimisation et Module Synthesis semblent similaires, ces similarités sont en vérité superficielles. En effet, si les modules acycliques et les circuits booléens ont en commun d'être représentés comme des graphes orientés acycliques, sont considérés d'avoir des entrées et des sorties et dont le calcul est caractérisé par des fonctions booléennes, ces similarités ne sont d'aucune aide pour comparer les problèmes de leur optimisation en taille. Si optimiser un circuit booléen repose sur la question de réaliser une fonction booléenne de façon la plus compacte et efficace en utilisant un ensemble limité de portes, optimiser un module acyclique demande d'optimiser un calcul qui est réalisé à travers le temps. Les contraintes qui concernent l'optimisation d'un circuit booléen reposent sur la structure même des

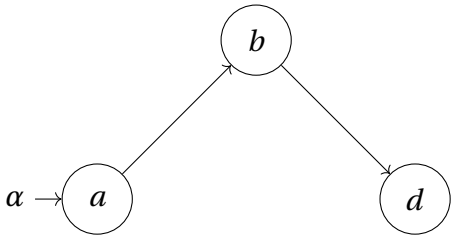

FIGURE 6.11 – Graphe d'interaction du module M'_A défini dans l'exemple 56.

fonctions booléennes, alors que les contraintes qui concernent l'optimisation d'un module acyclique reposent sur la difficulté de créer un flot d'information qui permet la réalisation d'une sortie composée d'entrées à différents délais. Ainsi, ces problèmes nous semblent trop différents pour permettre une réduction de l'un vers l'autre, et ce malgré leurs nombreuses similarités.

Le problème Module Synthesis est un problème difficile, et sa résolution dans le cas général parait compromettre la recherche d'une solution d'optimisation efficace. Cependant, son application sur de petites instances permet d'observer de réelles optimisations dans la taille de certain des réseaux étudiés. De telles optimisations sont illustrées par les exemples suivants.

Exemple 56. *Nous considérons la fonction de sortie O_d définie dans l'exemple 54. Pour $S'_A = \{a, b, d\}$ et $I_A = \{\alpha\}$, soit M'_A le module défini par les fonctions locales $f''_a = x_\alpha$, $f''_b = x_a$ et $f_d = \neg x_b$. Le module M'_A est une réponse valide à l'instance $I_A, \{O_d\}, k = 3$ du problème Module Synthesis. Le graphe d'interaction du module M'_A est représenté en figure 6.11.*

Exemple 57. *Nous considérons les fonctions de sorties $O_B = \{O_{St}, O_{Sl}, O_{Sk}, O_{Pp}, O_C, O_{C^*}\}$ définies dans l'exemple 55. Pour $S'_B = \{St, Sl, Sk, Pp, Ru, C25, C^*\}$ et $I_B = \{\alpha_{St}, \alpha_{Sl}, \alpha_{Sk}, \alpha_{Pp}, \alpha_C, \alpha_{C^*}\}$, soit M'_B le module défini par les fonctions locales $f''_{St}(x) = \neg x_{\alpha_{St}}$, $f''_{Sl}(x) = \neg x_{\alpha_{Sl}} \vee x_{\alpha_{C^*}}$, $f''_{Sk}(x) = x_{\alpha_{St}} \vee \neg x_{\alpha_{Sk}}$, $f''_{Pp}(x) = x_{\alpha_{Sl}} \vee \neg x_{\alpha_{Pp}}$, $f''_{Ru}(x) = \neg x_{\alpha_{Sk}} \vee x_{\alpha_{Pp}} \vee \neg x_{\alpha_C} \vee \neg x_{\alpha_{C^*}}$, $f''_C(x) = \neg x_{Ru} \vee \neg x_{\alpha_{Sl}}$, $f''_{C25}(x) = \neg x_{\alpha_{Pp}} \vee x_{\alpha_C}$, et $f'_{C^*}(x) = x_{C25}$. Le module M'_B est une réponse valide à l'instance $I_B, O_B, k = 8$ du problème Module Synthesis. Le graphe d'interaction du module M'_B est représenté en figure 6.12.*

6.4.4 Recomposition du réseau final

Enfin, la dernière étape de l'optimisation d'un réseau est la recomposition d'un réseau d'automates à partir du module optimal généré à la section précédente.

Cette recomposition est opérée simplement par l'application d'un branchement récursif qui est défini par l'étape précédente du processus qui a transformée un réseau d'automates en un module acyclique. Lors de cette opération, l'influence d'un ensemble d'automates a été remplacé par des entrées. L'étape finale du processus

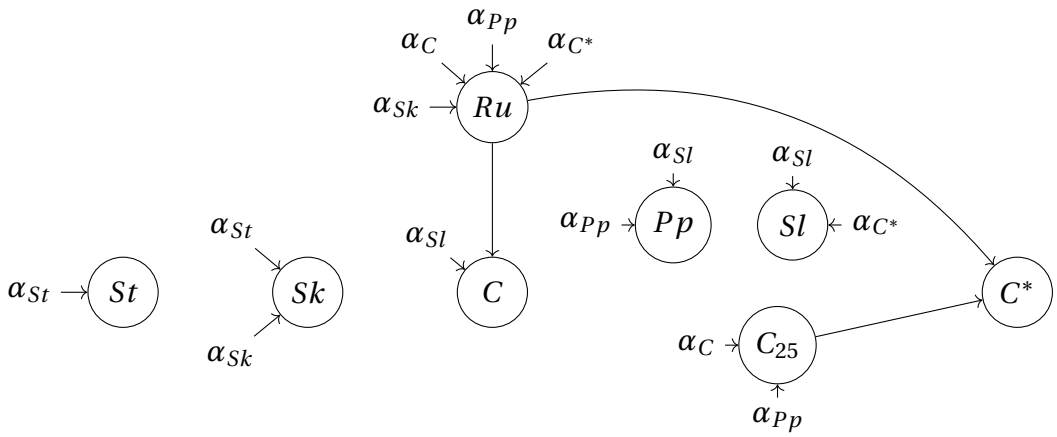

FIGURE 6.12 – Graphe d'interaction du module M'_B défini dans l'exemple 57.

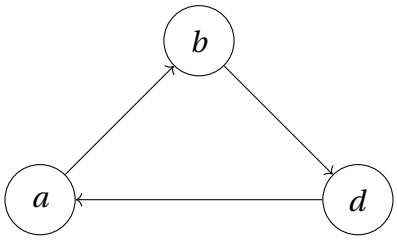

FIGURE 6.13 – Graphe d'interaction du réseau d'automates S'_A défini dans l'exemple 58.

consiste en un branchement de ces entrées vers les automates dont l'influence a été substituée.

Exemple 58. *Nous considérons le module acyclique M'_A de l'exemple 56. Soit $\mathtt{w}_A$ la fonction de branchement qui vérifie $\mathtt{w}_A(\alpha) = d$. Le réseau d'automates booléens $\circlearrowleft_{\mathtt{w}_A} M'_A$ est défini sur les automates $S'_A = \{a, b, d\}$ et les fonctions locales $f'''_a(x) = x_d$, $f'''_b(x) = x_a$, et $f'''_d(x) = \neg x_b$. Le graphe d'interaction de ce module est représenté en figure 6.13.*

Exemple 59. *Nous considérons le module acyclique M'_B de l'exemple 57. Soit $\mathtt{w}_B$ la fonction de branchement qui vérifie $\mathtt{w}_B(\alpha_s) = s$ pour tout s dans X_B. Le réseau d'automates booléens $\circlearrowleft_{\mathtt{w}_B} M'_B$ est défini sur les automates $S'_B = \{St, Sl, Sk, Pp, Ru, C25, C^*\}$ et le fonctions locales $f'''_{St}(x) = \neg x_{St}$, $f'''_{Sl}(x) = \neg x_{Sl} \vee x_{C^*}$, $f'''_{Sk}(x) = x_{St} \vee \neg x_{Sk}$, $f'''_{Pp}(x) = x_{Sl} \vee \neg x_{Pp}$, $f'''_{Ru}(x) = \neg x_{Sk} \vee x_{Pp} \vee \neg x_C \vee \neg x_{C^*}$, $f'''_C(x) = \neg x_{Ru} \vee \neg x_{Sl}$, $f'''_{C25}(x) = \neg x_{Pp} \vee x_C$, et $f'_{C^*}(x) = x_{C25}$. Le graphe d'interaction de ce module est représenté en figure 6.14.*

À la fin de tout ce processus, nous obtenons un réseau d'automates dont la taille est égale ou inférieure à la taille du réseau initial. De plus, nous avons la garantie par application du théorème 10 que les attracteurs du réseau final sont isomorphes

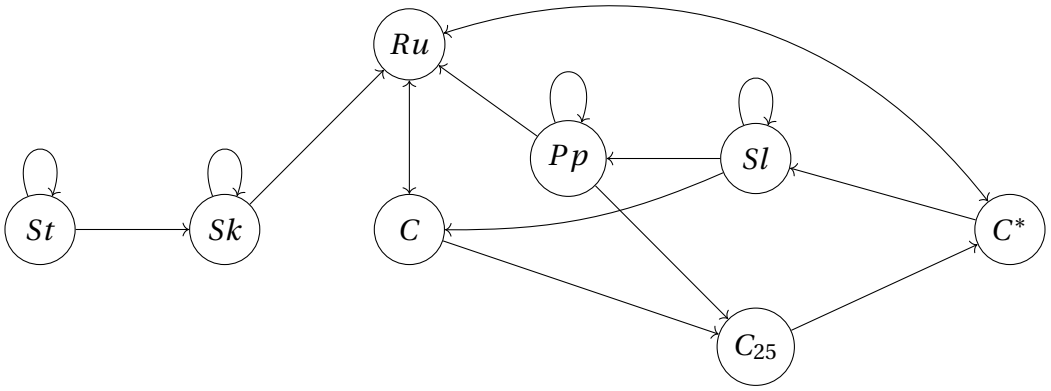

FIGURE 6.14 – Graphe d'interaction du réseau d'automates S'_B défini dans l'exemple 59.

aux attracteurs du réseau initial. L'application de ce théorème est faite sur la paire de modules acycliques constituée du module obtenu après application du problème Functional Acyclic Unfolding sur le réseau initial et le module obtenu par application du problème du Module Synthesis sur le premier module. Cette paire de modules acycliques possèdent des fonctions de sortie équivalentes au renommage des entrées et des automates près, et sont sont reconstitués par un branchement récursif équivalents par le même renommage des entrées et des automates en la paire de réseaux constituée du réseau initial, et du réseau final.

Dans la pratique, nous n'avons pas de garantie que ce processus permette une véritable optimisation. Cependant, nous illustrons certains cas positifs par le biais des séries d'exemples 50, 52, 54, 56, 58 et 51, 53, 55, 57, 59. La première série illustre l'application de notre méthode sur un réseau de taille 4 simple, construit pour posséder deux automates redondant qui sont "compactés" par notre méthode, qui obtient un réseau final de taille 3 décrit dans l'exemple 58. La seconde série illustre l'application de notre méthode sur un réseau plus complexe, emprunté dans (DAVIDICH et BORNHOLDT 2008), qui modélise le comportement de la division d'une cellule de levure. Ce réseau, initialement composé de 10 automates (exemple 51), est optimisé pour n'en contenir plus que 8 (exemple 59). Notre première série d'exemples permet donc le calcul des attracteurs du réseau initial par le développement d'un graphe de la dynamique de taille $2^3 = 8$ plutôt que par le développement d'un graphe de la dynamique de taille $2^4 = 16$. Similairement, la seconde série permet le calcul des attracteurs du réseau initial par le développement d'un graphe de la dynamique de taille $2^8 = 256$ plutôt que par le développement d'un graphe de la dynamique de taille $2^10 = 1024$.

En plus de permettre une réduction par une puissance de 2 du graphe de la dynamique parallèle du réseau étudié, cette méthode met en avant une notion formelle de la redondance dans un réseau d'automates. En effet, il semble que si cette méthode permet l'optimisation d'un réseau, c'est qu'il existe un ensemble d'automates dans ce

réseau dont le rôle peut être effectué par un ensemble d'automates plus petit. Bien que la méthode présentée ne s'applique que pour le mode de mise à jour parallèle, elle souligne que le nombre de telles "redondances" dans un réseau d'automates booléen implique une borne supérieure sur la taille totale des attracteurs de sa dynamique, car chaque redondance implique l'existence d'un réseau plus petit dont les attracteurs sont équivalents.

6.4.5 Cas des modules 1-pour-1

Bien que la complexité du cas général de l'optimisation d'un réseau d'automates reste une question difficile, nous détaillons dans cette section la complexité précise de l'optimisation d'un réseau qui peut être déplié en modules 1-pour-1. La définition de ces réseaux, qui sont caractérisés par un feedback vertex set minimal de taille 1 ou moins, est disponible en section 6.3.

Lors de l'application de la première étape de la procédure d'optimisation développée en section 6.4, l'utilisation d'un tel réseau permet l'obtension d'un module 1-pour-1, dont la seconde étape de la procédure pourra extraire une fonction de sortie définie sur une seule entrée.

Cette unique fonction de sortie peut être transformée en un module acyclique optimal par l'application de la propriété 21. En effet, cette propriété énonce qu'il existe un module 1-pour-1 qui réalise cette fonction, tel que sa taille est le délai de cette fonction. La construction de ce module peut être opérée en temps polynomial en la taille de l'encodage de la fonction de sortie, et nous montrons que sa taille est optimale.

▸ 1-FOR-1 MODULE SYNTHESIS PROBLEM
Entrée: Une fonction de sortie O définie sur une entrée unique α, et de délai d.
Sortie: Un module acyclique M optimal tel que la fonction O est la fonction de sortie d'au moins un automate dans M.

Théorème 17 (PERROT, PERROTIN et SENÉ 2020b)**.** *Le problème 1-for-1 Module Synthesis est dans FP.*

Démonstration. Pour $\mathtt{d}$ le délai de la fonction de sortie O, nous observons qu'il n'est pas possible de produire un module de taille inférieure à $\mathtt{d}$: la taille de la plus longue chaîne dans le module donne une borne supérieure à la variable de plus grand délai parmi les fonctions de sortie du module.

Nous reproduisons la construction détaillée dans la preuve de la propriété 21. Cette construction est de taille $\mathtt{d}$, et ne requiert que l'assemblage d'une chaîne d'automates de taille $\mathtt{d}-1$ avec des fonctions locales de taille constante et un automate dont la fonction locale est obtenue par substitution depuis O. Ce procédé est polynomial et optimal. □

Ce résultat est pertinent lorsque l'on considère que les modules 1-pour-1 ne sont pas une réduction triviale des modules acycliques, mais généralisent de nombreuses

classes de réseaux dont la caractérisation des attracteurs nous échappe encore. Ce résultat permet de réduire l'étude des attracteurs de cette famille de réseaux à l'étude des attracteurs des réseaux d'automates à décalage à rétroaction, qui sont optimaux en taille.

6.5 Conclusion

Dans ce chapitre, nous avons étudié le cas particulier que représentent les modules acycliques. Ces objets qui généralisent les réseaux d'automates acycliques – dont le comportement est simple – possèdent un comportement suffisamment complexe pour que leur étude soit pertinente dans le cadre de la compréhension des réseaux d'automates, mais également suffisamment simple pour permettre l'élaboration de résultats.

S'il existe une méthode d'optimisation efficace des modules 1-pour-1 mais que le cas général est difficile à caractériser, la meilleure direction future semble l'étude progressive et attentive des modules k-pour-k. Pour chaque entier k, notre formalisme offre une famille de modules plus générale que la famille précédente, qui englobe une portion plus large de la faune des réseaux d'automates booléens, et dont la difficulté de l'optimisation est plus grande. Cette étude offre une hiérarchie intuitive, permettant de représenter l'évolution de la difficulté de ces questions mais aussi de nombreuses autres – la nature et les propriétés de leur fonctions de sortie, le nombre total de cycles que les réseaux associés admettent – à travers l'augmentation de la connectivité des modules étudiés.