Université de Strasbourg
Magistère de Mathématiques 2e année

Théorie algébrique du routage informatique par les *path-vector protocols*

Adrien RODAU

Rapport de Stage supervisé par :

Cristel PELSSER
Professeure des Universités

Pascal MÉRINDOL
Maître de Conférences

ICube
300, boulevard Sébastien Brand
67412 Illkirch-Graffenstaden

UFR de Mathématique et d'Informatique
7, rue René Descartes
67084 Strasbourg

Juin-Juillet 2017

Résumé

Le routage informatique n'est pleinement opérationnel que si l'on peut s'assurer que les protocoles choisis convergeront. Après une présentation du mécanisme de fonctionnement des *path-vector protocols*, nous en proposons une retranscription algébrique détaillée et ordonnée. Celle-ci sert ensuite de cadre pour obtenir plusieurs conditions suffisantes de convergence, plus ou moins restrictives, dont la dernière est également nécessaire. Nous étendons ensuite notre théorie pour la rendre aussi constructive que descriptive, tout en examinant le comportement des constructeurs vis-à-vis des conditions de convergence. Enfin, nous tentons de dégager des pistes pour rendre nos résultats de convergence plus généraux.

Présentation du laboratoire ICube

Ce rapport est la synthèse d'un stage effectué au sein de l'équipe Réseaux du laboratoire ICube, rattaché à l'Université de Strasbourg.

Le laboratoire ICube (Sciences de l'Ingénieur, de l'Informatique et de l'Imagerie) est un organisme de recherche pluri-disciplinaire fondé à l'initiative de l'Université de Strasbourg, de l'INSA Strasbourg, de l'ENGEES et du CNRS. Son but est d'offrir une structure commune aux chercheurs strasbourgeois travaillant dans les domaines liés aux sciences et technologies appliquées. L'ICube est divisé en 4 départements englobant 16 équipes de recherche et regroupe plus de 580 chercheurs titulaires.

L'équipe Réseaux fait partie du Département de Recherche en Informatique. Composée de neuf enseignants-chercheurs, un chargé de recherche et un ingénieur d'études, elle concentre son travail sur le développement de protocoles de communication informatique. À ce titre, l'équipe s'intéresse au routage dans les réseaux informatique aussi bien dans son aspect théorique que dans son implémentation.

Première partie

Le routage dans les réseaux informatiques

Le mot « routage » désigne l'action globale visant à transporter des données entre leur origine et leur destination, à travers un réseau composé de routeurs actifs et de liens. Le rôle de chaque routeur est d'aiguiller ces données vers le prochain routeur avec évidemment l'intention de les renvoyer sur une route les amenant à leur destination finale. Mais un objectif aussi simple peut se décliner en pratique par une multitude d'implémentations différentes. Le langage de communication utilisé entres les routeurs est le cadre de chacune de ces implémentations. Ces langages et les algorithmes associés sont donc appelés *protocoles de routage*.

1 Les différentes classes de protocoles

Tous les protocoles de routage ont pour base commune que leurs routeurs tentent toujours de sélectionner une route sur la base de leurs propres caractéristiques et des données apportées par des informations de routage. En règle générale, ces informations circulent à travers le même réseau que les données à utiliser. En effet, un réseau informatique n'est physiquement pas restrictif sur la nature des données qui le traversent, et de plus presque tous les administrateurs préfèrent, à des fins de simplicité, séparer les protocoles qui utilisent des cartes différentes du même réseau physique.

La première distinction fondamentale entre les protocoles vient donc de la nature de ces informations de routage mise à disposition des routeurs. On distingue ainsi deux grandes classes :

***Distance-vector protocols* et *path-vector protocols* :** les informations de routage ne concernent que le chemin qu'elles ont traversé et sont cloisonnées en fonction de la destination qu'elles concernent.

***Link-state protocols* :** chaque routeur possède une carte complète du réseau, qu'il met à jour grâce aux informations de routage.

Il est important de noter que nous utilisons les appellations anglophones de chaque classe, qui sont issues de leurs variantes les plus usuelles, mais qui ne les représentent pas vraiment dans

toute leur généralité. Ainsi, les *distance-vector protocols* peuvent être vus comme une sous-classe des *path-vector protocols*.

C'est précisément sur ces derniers que notre étude se concentre. En effet, ils représentent sans doute la classe de protocoles qui offre la plus grande flexibilité, tout en restant techniquement très efficace.

2 Les *path-vector protocols*

La spécificité de cette classe de protocoles est que le routage y est mis en place grâce à des informations créées, traitées, et échangées au niveau local par chaque routeur. Ceci offre une grande liberté aux administrateurs pour décider de leur propre politique tout en conservant un protocole commun. À titre de comparaison, dans un *link-state protocol* le meilleur chemin est imposé globalement par la carte du réseau.

À cause de cette conception locale, les *path-vector protocols* nécessitent une phase d'initialisation dans laquelle le réseau va en quelque sorte se découvrir lui-même. Ce mécanisme devient ensuite diffus pour adapter le routage en fonctions des aléas susceptibles de se produire (lien brisé ou surchargé, routeur déconnecté, ou à l'inverse mise en place d'un nouveau lien ou routeur).

Afin de mieux comprendre le fonctionnement de ces protocoles, nous allons d'abord l'expliquer à l'aide d'une analogie.

Imaginons un campement composé de tentes isolées les unes des autres en pleine forêt. Une seule d'entre elles est désignée comme point de rassemblement. Des volontaires partent de ce point et explorent tous les chemins possibles dans la forêt pour rejoindre la première tente. Lorsqu'un volontaire atteint une tente, il explique à ses occupants tous les détails du chemin qu'il a traversé. S'ils le trouvent satisfaisant, les occupants de la tente pourront prendre le même chemin, mais en sens inverse. En attendant, ils désignent un autre volontaire pour aller informer la prochaine tente. Au bout d'un moment, un volontaire peut finir par arriver à une tente où les occupants lui expliqueront que le chemin qu'il décrit n'est pas intéressant et qu'ils en ont déjà un qui les satisfait. Puisqu'il n'a plus rien d'intéressant à enseigner, le volontaire s'arrête là. Lorsque tous les volontaires sont arrêtés, l'initialisation est terminée. En cas de besoin, les occupants des tentes sauront dès lors tous par quel chemin passer. Mais pour que cela soit clair pour eux, il faudra que le volontaire leur ait décrit le chemin en inversant toutes les directions que lui-même avait prises. Si un jour un occupant découvre que le chemin est devenu impraticable, on reprendra un autre chemin qu'on avait délaissé, mais il faudra exceptionnellement désigner un nouveau volontaire pour aller prévenir les prochaines tentes de l'évolution de la situation.

Bien évidemment, les volontaires correspondent aux informations de routage et les occupants aux données. Pour les raisons évoquées, l'étude du cheminement des informations de routage suffit à analyser le protocole, mais nous continuerons toutefois à décrire les directions des chemins du point de vue des données. Ainsi, les informations de routages partent de la *destination*, arrivent aux nœuds par une arête *sortante* et sont réémises par une arête *entrante*. Il en sera de même pour les flèches représentant les liens sur toutes nos figures.

Nous allons maintenant expliquer rigoureusement l'algorithme de fonctionnement des *path-vector protocols* :

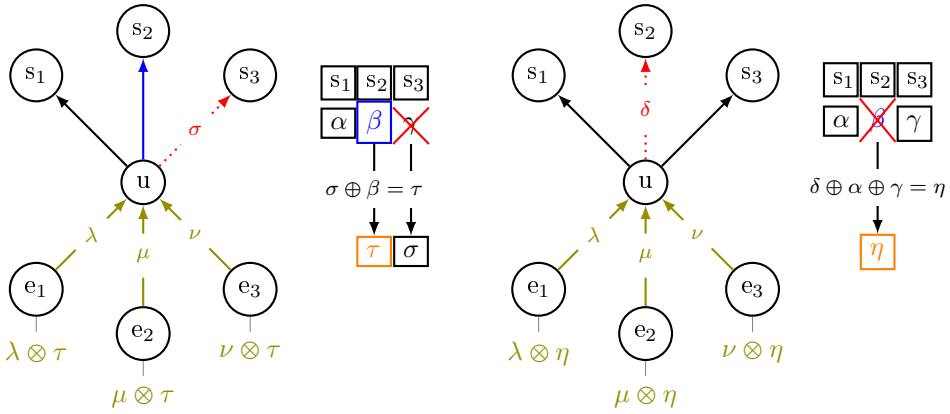
À tout instant et pour chaque destination, chaque routeur garde en mémoire au minimum :

— le lien sortant à utiliser pour envoyer des données vers la destination en question.

— un tableau stockant les dernières informations de routage reçues pour chacun des liens sortants du routeur.

Lorsque le protocole est initialisé, chaque nœud autorisé à être une destination se déclare comme tel en envoyant des informations de routage en ce sens à tous ses voisins.

À partir de ce moment-là, des informations de routage vont remonter le réseau, le chemin qu'elles ont parcouru étant exactement celui qu'elles annoncent, mais en sens inverse. Ces informations sont traitées et modifiées par chaque routeur.



(a) Cas général (b) Cas particulier

FIGURE 1 – Illustration de l'algorithme des *path-vector protocols*. Le routeur u reçoit de nouvelles informations (en rouge). Il les traite (de bleu à orange), puis annonce une éventuelle modification de son choix à ses voisins (en kaki).

En détail, lorsqu'un routeur reçoit de nouvelles informations de routages concernant une destination spécifique, il met en œuvre l'algorithme suivant :

Cas général : Les nouvelles informations proviennent d'un lien sortant différent du dernier ayant induit une modification.

1. Le routeur traite les nouvelles informations reçues avec celles correspondant à son choix actuel.
2. Il en déduit le nouveau lien à utiliser ou décide de ne rien changer.

Cas particulier : Les nouvelles informations de routage proviennent du même lien sortant que le dernier ayant induit une modification.

1. Le routeur annule ce choix devenu caduc.
2. Il compare ensuite ces nouvelles informations avec chacune de celles qu'il a stockées dans son tableau. [1]
3. Il en déduit le nouveau lien à utiliser.

Dans tous les cas : Le routeur met ensuite en œuvre les actions suivantes :

1. Si le routeur a changé son choix : il envoie les informations de routage correspondantes à tous ses voisins, en les modifiant en fonction de l'arête entrante utilisée pour les joindre.
2. Le routeur met à jour son tableau : il écrit les informations de routage reçues dans la case correspondante à l'arête sortante par laquelle elles sont arrivée et écrase les anciennes devenues invalides.

1. Cette partie de l'algorithme n'est cohérente avec le reste de la théorie que si l'axiome 1 est valide. Par ailleurs, si le routeur dispose d'un ordre total sur les chemins comme dans *Border Gateway Protocol* (BGP), alors cela revient simplement pour lui à choisir le meilleur chemin encore disponible. Dans le cas le plus général, le routeur opère une sélection arbitraire locale.

Cet algorithme est illustré sur la figure 1 (en utilisant les opérateurs $\oplus$ et $\otimes$ qui seront développés dans la partie II).

Même si l'initialisation se termine, l'algorithme est susceptible de se relancer si un routeur découvre qu'un lien du réseau ne fonctionne plus. Il décide alors simplement d'annuler son choix de chemin passant par ce lien et reprend la partie de l'algorithme correspondante.

Nous sommes resté volontairement très flou en employant l'expression « traiter les informations ». Il s'agit là de retranscrire fidèlement en mots la véritable généralité de la structure algébrique sous-jacente à l'algorithme, qui sera présentée dans la partie II.

Cette généralité et cette flexibilité font partie des raisons pour lesquelles le seul protocole utilisé de fait à l'heure actuelle par Internet au niveau inter-systèmes fait partie de la classe des *path-vector protocols*. Son nom est BGP, et sa présentation va nous donner un exemple précis des enjeux liés à cette classe de protocoles.

3 Le protocole BGP

En pratique, BGP existe en deux variantes : *Internal Border Gateway Protocol* (IBGP) et *External Border Gateway Protocol* (EBGP). Nous ne présenterons ici que EBGP, qui est la version correspondant le plus aisément à notre théorie à venir. Dans ce protocole, les informations de routages sont les suivantes :

Destination (toujours présente quel que soit le protocole).

`as_path` : Vecteur retraçant les nœuds constituant le chemin.

`origin` : Message du dernier routeur indiquant s'il a connu le chemin par un voisin intra ou extra-régional. Peut être non renseigné.

`multi_exit_discriminator (med)` : Valeur numérique utilisée seulement si plusieurs liens distincts joignent les mêmes nœuds, afin de les discriminer.

`community` : Champ optionnel éditable par chaque nœud pour indiquer un niveau de préférence global particulier pour le chemin.

À partir de ces informations, chaque routeur va comparer deux chemins selon plusieurs critères ordonnés :

1. `loc_pref` : Préférence locale du routeur.
2. `as_path` : Le chemin le plus court, c'est-à-dire traversant le moins de noeuds, est préféré. Si le routeur découvre qu'il fait déjà partie d'un chemin, celui-ci est immédiatement rejeté afin d'éviter de former une boucle.
3. `origin` : Intra-régional est préféré à extra-régional qui est lui-même préféré à un champ non renseigné.
4. `med`
5. `router_id` : Dernier recours arbitraire. Compare les identifiants des routeurs ayant transmis les informations.

Chaque critère n'est utilisé que si tous les précédents n'ont pas permis de départager les chemins, suivant donc un ordre lexicographique.

4 Problématiques

A priori, rien ne garantit que toutes les informations de routage vont cesser leur parcours. Si c'est le cas, on dit que le protocole *converge*, et la phase d'initialisation se termine effectivement.

Obtenir la convergence de son protocole tout en restreignant au minimum la liberté qu'il lui offre est le but de tout administrateur réseau. Notre étude tentera donc d'identifier des guides permettant d'atteindre la convergence pour les *path-vector protocols*. Pour y parvenir, nous choisissons une généralisation de l'approche analytique introduite dans [3]. Elle consiste à formaliser les mécanismes de l'initialisation sous une forme algébrique. Ce travail va occuper toute la partie suivante.

Deuxième partie
Théorie algébrique du routage

La formalisation algébrique du routage tente de retranscrire par une structure abstraite les informations de routage utilisées par les protocoles et la façon dont elles sont exploitées. Pour reprendre la séparation naturelle entre un programme de routage et le réseau sur lequel il est exécuté, nous distinguons ce que nous appelons l'algèbre de routage, qui formalise le premier, du graphe pondéré qui représente le second.

5 Algèbres de routage

Pour construire la théorie des algèbres de routage, nous adoptons une méthode constructive consistant non pas à retranscrire directement les mécanismes de protocoles établis tels que BGP, mais plutôt à formaliser les mécanismes fondamentaux du routage informatique, afin d'en poser une base la plus générale possible. Toutefois, même à un tel niveau de généralité, la théorie reste conçue pour les *path vector protocols* afin de pouvoir étudier le comportement de l'algorithme décrit dans la section 2.

Définition 1. Une *algèbre de routage* est un quadruplet $\mathcal{A} = \langle S, \oplus, \otimes, \mathcal{O} \rangle$ où $S = L \uplus \Sigma$ est un ensemble (fini non-vide) partitionné et muni de deux opérations :

$$\oplus : \Sigma \times \Sigma \longrightarrow \Sigma \qquad \otimes : L \times \Sigma \longrightarrow \Sigma$$

et dont on distingue un sous-ensemble non-vide $\mathcal{O} \subset \Sigma$.

Détaillons le rôle de chaque composante :
- Σ est l'ensemble des *signatures*, qui caractérisent les chemins du réseau. Elles représentent les informations de routage transmises par le protocole.
- L est l'ensemble des *étiquettes* (*labels*) associées aux arêtes du graphe.
- $\oplus$ est une opération binaire infixe et interne de Σ qui modélise le traitement comparé de deux signatures pour en choisir une troisième.
- $\otimes$ est l'opération naturelle d'extension d'un chemin à travers une arête, appliquée aux informations de routage associées.
- $\mathcal{O}$ est un ensemble de signatures spéciales pouvant être annoncées par un nœud lorsqu'il se déclare comme destination. Ces nœuds sont représentés comme doubles sur nos figures.
- $\phi \in \Sigma$ est la signature attribuée à tous les chemins interdits. Elle vérifie :

$$\forall\, \alpha \in \Sigma : \phi \oplus \alpha = \alpha \oplus \phi = \alpha$$

et est donc le neutre de $\oplus$.

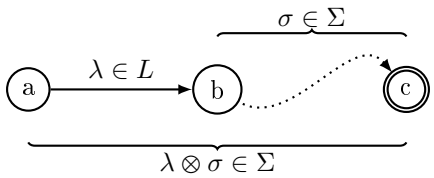
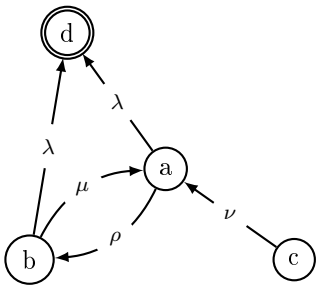


FIGURE 2 – Fonctionnement de $\otimes$. Les flèches sont dans le sens de parcours des données. Un nœud double représente la destination.



(a) Un réseau

Chemin	Signature
d	ϵ
ad	1
bd	1
abd	2
bad	3
cad	4
cabd	5

$\Sigma \setminus \{\phi\}$

(b) Correspondance avec une algèbre compatible

L / $\otimes$	ϵ	1	2	3	4	5	ϕ
λ	1	ϵ	ϕ	ϕ	ϕ	ϕ	ϕ
μ	ϕ	3	1	ϕ	ϕ	ϕ	ϕ
ν	ϕ	2	ϕ	1	ϕ	ϕ	ϕ
ρ	ϕ	4	5	ϕ	ϕ	ϕ	ϕ

(c) Table de $\otimes$

$\oplus$	ϵ	1	2	3	4	5	ϕ
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
1	ϵ	1	1	1	4	5	1
2	ϵ	1	2	4	1	5	2
3	ϵ	1	4	3	ϕ	3	3
4	ϵ	ϕ	1	ϕ	1	3	4
5	ϵ	5	2	3	2	5	5
ϕ	ϵ	1	2	3	4	5	ϕ

(d) Table de $\oplus$. Seule la partie orange est effectivement appliquée sur le réseau présenté ici.

FIGURE 3 – Un exemple de réseau et d'algèbre de routage

La distinction entre signatures et étiquettes n'est pas nécessaire théoriquement. En effet, on aurait aussi bien pu considérer qu'une arête est un chemin élémentaire de longueur 1, et que l'opération $\otimes$ s'apparente à une concaténation. Nous incluons toutefois cette séparation précisément parce qu'elle n'induit aucune perte de généralité mais permet de gagner en clarté. On peut alors d'un point de vue purement formel dupliquer les informations associées aux arêtes pour leur attribuer à la fois une étiquette et une signature pour le chemin élémentaire correspondant (à condition que le bout de l'arête puisse être un nœud de destination).

Il est important de clarifier ce que représente exactement l'algèbre dans le protocole de routage.

D'un point de vue pratique, le logiciel du routeur est conçu selon deux principes. Premièrement, il est totalement indépendant de la carte du réseau sur lequel il sera employé. Deuxièmement, il fournit une structure rigide et définitive aux informations de routages (c'est-à-dire aux signatures), comme dans le cas de BGP, et il fournit aussi les algorithmes employés pour les comparer. Toutefois, ce n'est qu'au moment d'implémenter le logiciel dans un réseau que l'administrateur distribue les étiquettes sur les arêtes d'une part, et fixe une correspondance entre les

chemins du réseau et certaines signatures d'autre part. Concrètement, dans le cas de BGP, cela revient à attribuer les valeurs du champs `loc_pref`.

Ces correspondances sont illustrées sur l'exemple de la figure 3. Les opérateurs d'une algèbre de routage sont pensés pour représenter le logiciel. Ainsi, le fait que seule une petite partie de la table de $\oplus$ témoigne de l'indépendance entre l'algèbre et le réseau.

Toutefois, représenter le logiciel ne signifie pas le modéliser. L'algèbre de routage n'a pas en soi vocation à préciser les structures internes de ses composantes. Elle est avant tout une construction mathématique abstraite, et en tant que telle, sa généralité repose sur sa définition phénoménologique. Ainsi, signatures et étiquettes ne sont formellement rien d'autre que des notations, et les opérateurs ne sont connus que par leurs tables. C'est cela même qui permet de donner le cadre le plus général possible capable de représenter les protocoles de type *path-vector*.

Parce qu'elles donnent une structure unique aux protocoles des plus simples aux plus complexes, ce ne sera qu'après avoir épuisé l'analyse théorique phénoménologique des algèbres de routage que nous les inscrirons dans une démarche effective de construction, dans la partie IV.

6 Axiomes de construction

La définition de base des algèbres de routage, bien que satisfaisant un objectif de généralité, est trop dépouillée pour être exploitée immédiatement dans une analyse théorique. Il est d'abord nécessaire, tout en restant pertinent, de distinguer des sous-classes d'algèbres à l'aide d'axiomes dits « de construction » . Ces axiomes, bien que toujours exprimés phénoménologiquement, représentent chacun une étape importante de structuration de l'algèbre de routage qui apportera à terme suffisamment d'informations pour commencer à étudier des résultats de convergence.

6.1 Axiomes sur $\oplus$

Les axiomes sont présentés selon un ordre doublement hiérarchique. Cela signifie d'une part que chaque axiome n'est proprement cohérent que s'il s'ajoute à tous les précédents, et d'autre part qu'il représente une perte effective de généralité par rapport à eux.

Axiome 1

> $\oplus$ *est* ***associatif*** *:* $\forall\ (\alpha, \beta, \gamma) \in \Sigma^3 : (\alpha \oplus \beta) \oplus \gamma = \alpha \oplus (\beta \oplus \gamma) = \alpha \oplus \beta \oplus \gamma$

Cet axiome garantit une certaine forme de cohérence dans le traitement comparé des signatures. Il permet d'étendre $\oplus$ à plus de deux arguments en décomposant la chaîne d'opérations comme on le souhaite.

Par ailleurs, on peut faire immédiatement le lien avec les structures mathématiques classiques :

Corollaire 1 (Axiome 1)**.** $(\Sigma, \oplus)$ *est un* monoïde*, c'est-à-dire un ensemble muni d'une loi interne associative possédant un neutre.*

Axiome 2

> $\oplus$ *est* ***commutatif*** *:* $\forall\ (\alpha, \beta) \in \Sigma^2 : \alpha \oplus \beta = \beta \oplus \alpha$

On impose ici une symétrie dans le traitement des signatures.

Corollaire 2 (Axiomes 1 et 2)**.** $(\Sigma, \oplus)$ *est muni d'une relation de pré-ordre (réflexive et transitive) canonique :*

$$\beta \preccurlyeq \alpha \iff \exists\ \gamma \in \Sigma : \beta = \gamma \oplus \alpha$$

Démonstration. Tout d'abord, $\preccurlyeq$ est bien définie grâce à la commutativité. On entend par là qu'il n'existe pas deux relations similaires mais différentes en fonction du positionnement d'un côté ou de l'autre de $\oplus$.

Pour tout $\alpha \in \Sigma$, on a : $\alpha = \alpha \oplus \phi$. Donc $\alpha \preccurlyeq \alpha$ et ainsi $\preccurlyeq$ est réflexive.

Soient $(\alpha, \beta, \gamma) \in \Sigma^3$ telles que $\gamma \preccurlyeq \beta \preccurlyeq \alpha$. Alors il existe β', α' telles que $\gamma = \beta' \oplus \beta$ et $\beta = \alpha' \oplus \alpha$. Par associativité, on a $\gamma = \beta' \oplus \alpha' \oplus \alpha = (\beta' \oplus \alpha') \oplus \alpha$ et ainsi $\gamma \preccurlyeq \alpha$. Donc $\preccurlyeq$ est transitive. □

Nous avons désormais assez de structure pour commencer à parler de « comparaison » entre les signatures. Il convient toutefois de signaler qu'à ce stade, $\beta \preccurlyeq \alpha \Leftrightarrow \beta = \gamma \oplus \alpha$ ne peut s'interpréter en toute généralité que par « β *peut être obtenue* à partir de α ».

Axiome 3

> $\oplus$ *est* ***idempotent*** *:* $\forall\ \alpha \in \Sigma : \alpha \oplus \alpha = \alpha$

Bien que cet axiome semble totalement naturel dans les protocoles de type *path-vector* complets, il n'existe aucune raison de le considérer comme tel en toute généralité. La construction mathématique montre au contraire que cet axiome simple représente en réalité une étape de structuration majeure, à travers le corollaire suivant :

Corollaire 3 (Axiomes 1 à 3)**.** *La relation de pré-ordre canonique* $\preccurlyeq$ *devient une relation d'ordre.*

Il est intéressant de noter que ce corollaire reste toujours valable à partir d'une hypothèse légèrement plus faible : la m-idempotence, définie par :

$$\forall\ \alpha \in \Sigma : \underbrace{\alpha \oplus \alpha \oplus \cdots \oplus \alpha}_{m \text{ itérations}} = \underbrace{\alpha \oplus \alpha \oplus \cdots \oplus \alpha \oplus \alpha}_{m+1 \text{ itérations}}$$

Démonstration. Il suffit de montrer l'anti-symétrie. Soient $(\alpha, \beta) \in \Sigma^2$ telles que :

$$\beta \preccurlyeq \alpha \Longleftrightarrow \exists\ \gamma \in \beta = \gamma \oplus \alpha$$
$$\beta \succcurlyeq \alpha \Longleftrightarrow \exists\ \delta \in \alpha = \delta \oplus \beta$$

D'où :

$$\begin{aligned} \alpha &= \delta \oplus \gamma \oplus \alpha \\ &= \delta \oplus \gamma \oplus \delta \oplus \gamma \oplus \alpha \\ &= 2\delta \oplus 2\gamma \oplus a \\ &= m\delta \oplus m\gamma \oplus \alpha \end{aligned} \qquad \begin{aligned} \beta &= \gamma \oplus \delta \oplus \beta \\ &= m\gamma \oplus m\delta \oplus \beta \\ &= (m+1)\gamma \oplus m\delta \oplus \alpha \\ &= m\gamma \oplus m\delta \oplus \alpha \\ &= \alpha \end{aligned}$$

□

Le dernier axiome permet enfin d'accéder à une représentation proche de celle de BGP.

Axiome 4

> $\oplus$ *est* ***sélectif*** *:* $\alpha \oplus \beta = \alpha$ *ou* β

On peut alors enfin dire que $\oplus$ *compare* véritablement deux signatures, en ce sens qu'elle en choisit une des deux.

On ne sera donc pas surpris d'obtenir le résultat suivant :

Corollaire 4. *La relation d'ordre canonique $\preccurlyeq$ est totale.*

Démonstration. Soient $(\alpha, \beta) \in \Sigma^2$ deux signatures quelconques. Supposons (par exemple) que $\alpha \oplus \beta = \beta$. Alors par définition $\beta \preccurlyeq \alpha$. Dans tous les cas β et α sont comparables. □

À ce stade, on obtient une structure très forte qui, pour $\oplus$, a rejoint le mécanisme de base des protocoles tels que BGP.

Il n'est plus nécessaire d'ajouter d'autres axiomes de construction sur $\oplus$. En effet, le cœur même des *path-vector protocols* reposant sur la comparaison des signatures, l'ordre total sur Σ est le terminus structurel naturel.

6.2 Axiomes gratuits

L'opérateur $\otimes$ étant en lui-même la retranscription d'une étape parfaitement indécomposable du concept même de routage, à savoir l'extension d'un chemin par une arête, les axiomes le concernant ne sont pas à proprement parler constructifs. En effet, ils ne caractérisent pas l'opération mais représentent uniquement des ajouts théoriques qui n'occasionnent aucune perte de généralité.

Axiome A

> *Il existe un neutre (à gauche) de $\otimes$ noté e. ϕ est absorbant pour $\otimes$*

e est un étiquette qui représente le fait de ne pas bouger, de ne pas étendre un chemin. La deuxième partie de cet axiome est simplement l'idée intuitive selon laquelle si un chemin est interdit, il le reste même si on essaie de l'étendre à travers une arête.

Axiome B

> *Il existe un « anti-neutre » $\epsilon \in \Sigma$ tel que : $\forall\ \sigma \in \Sigma : \epsilon \oplus \sigma = \epsilon$.*

Encore une fois traduit en mots, cet axiome signifie que quoi qu'il arrive, rester au même endroit est toujours la meilleure solution comparée à n'importe quelle autre. Ainsi ϵ peut être vu comme le pendant de l'étiquette e chez les signatures.

6.3 Précisions d'interprétation

On pourrait penser qu'une description telle que celle du dernier axiome représente nécessairement une restriction par rapport au cas le plus général. Mais en réalité cette affirmation ne concerne que e, et absolument rien ne nous empêche d'introduire une étiquette supplémentaire dans notre modèle, sans qu'elle ne soit implémentée réellement. C'est en ce sens que l'ajout de e, tout comme celui de ϕ, n'est qu'une simple facilité théorique qui simplifiera beaucoup la compréhension de l'analyse des structures, sans les dénaturer en aucune manière dans leur représentativité.

Par exemple, supposons que l'on désire « violer » l'axiome B et décréter qu'il existe un certain chemin de signature σ et une certaine arête d'étiquette λ pour lesquels l'extension serait préférée à l'immobilité, c'est-à-dire exprimé formellement : $(e \otimes \sigma) \oplus (\lambda \otimes \sigma) = \lambda \oplus \sigma$. Alors rien n'empêche d'introduire une deuxième étiquette e', elle aussi purement théorique et sans implémentation, et qui aurait exactement le comportement voulu en lieu et place de e.

Ceci illustre une autre notion fondamentale de la théorisation du routage : puisqu'elle le représente plutôt qu'elle le modélise, les propriétés qu'elle énonce sont absolues. En clair, si un chemin a pour signature ϕ et qu'on dit qu'il est interdit, alors il est *totalement* interdit. On ne peut formellement changer ce fait. Mais on peut toujours modifier notre *interprétation* des

signatures d'une autre algèbre pour y voir une variante de la première, où ce qui était « interdit » deviendrait « autorisé » . La partie IV décrira quelques façons de formaliser ainsi de telles réinterprétations sous la forme de constructeurs sur les algèbres de routage.

7 Parallèle mathématique

L'ordre des axiomes que nous avons proposé est basé sur l'enchaînement logique de leurs corollaires. Une telle construction est donc également applicable à d'autres domaines utilisant eux aussi une structure algébrique à deux opérateurs. De fait, notre théorie possède un parallèle mathématique (largement développé dans [1]) dont elle diffère uniquement par l'absence d'un axiome fondamental : la distributivité de $\otimes$ sur $\oplus$. Les correspondances sont résumées dans la figure 4.

Axiomes communs	Structure mathématique	Algèbres de routage
–	Magma	Algèbre de routage
Axiome 1	Monoïde	
Axiomes 1 et 2, Axiomes A et B	Semi-anneau	
Axiomes 1 à 3, Axiomes A et B	Dioïde	Algèbre de routage ordonnée
Axiomes 1 à 4, Axiomes A et B	Dioïde sélectif	Base de routage

FIGURE 4 – Résumé des structures parallèles

En réalité, puisque le mécanisme algébrique des deux constructions est identiques, leurs différences reposent dans l'inclusion des axiomes. Or celle-ci est entièrement liée à l'objectif poursuivi par chacune des voies. La théorie mathématique est constructive : chaque axiome est pensé et inclus en fonction des ses corollaires, le but étant d'obtenir des structures cohérentes aux propriétés fortes. *A contrario*, la théorie informatique est descriptive : les axiomes représentent des comportements effectifs à inclure ou non en fonction de la généralité voulue, et dont les corollaires peuvent alors être valorisés. Mais la hiérarchisation est reprise afin de mieux cerner comment procéder à une telle inclusion. Or, la distributivité de $\otimes$ sur $\oplus$ induit une très large perte de généralité, tout en n'ayant aucune conséquence qui puisse être considérée comme structurelle. Au contraire, sa formulation :

$$\forall\ \lambda \in L, \forall\ (\sigma, \tau) \in \Sigma^2 : \lambda \otimes (\sigma \oplus \tau) = (\lambda \otimes \sigma) \oplus (\lambda \otimes \tau)$$

signifie précisément que les choix des routeurs doivent être les mêmes avant et après chaque arête. Ceci va à l'encontre de la philosophie des *path-vector protocols*, notamment BGP où `local_pref`, la préférence propre du routeur, est privilégiée devant tous les autres critères.

Ces trois raisons expliquent l'absence de cette condition de la théorie algébrique du routage à ce niveau.

Troisième partie
Convergence des protocoles

8 Définitions

La notion de convergence est depuis le départ l'objectif ultime de notre étude théorique. Il est donc nécessaire de clarifier quelles sont nos attentes précises à l'aide d'une série de définitions qui vont compartimenter toutes les variantes de la convergence.

8.1 Types de convergence

Définition 2. On dit qu'un protocole *converge unilatéralement* si, pour une *unique destination* annoncée d, les transferts d'informations de routage cessent au bout d'un temps fini.

Cette définition est la plus simple et la plus universelle. Presque toutes les fois où nous avons employé et où nous emploierons le mot « convergence » seul, c'est cette définition qui est sous-entendue.

Même s'il est possible de s'arrêter là, la théorie nous offre de pousser plus loin nos ambitions en développant des convergences plus fortes, à commencer par la *convergence multilatérale*.

La différence majeure est que cette fois la destination est un *ensemble de nœuds* noté D. Tous les nœuds du réseau cherchent toujours à atteindre D, ce qui signifie maintenant atteindre un nœud quelconque de D.

Mais les nœuds de D ne sont pas tous identiques en tant que destinations. On définit l'*assignation initiale* par :

$$R : D \longrightarrow \mathcal{O} \subset \Sigma$$

On peut de façon équivalente voir R comme une fonction $N \to \mathcal{O}$ (où N est l'ensemble des nœuds du réseau) et poser $D = \mathrm{Supp}(R) = \{u \in N \mid R(u) \neq \phi\}$.

Chaque nœud de destination peut ainsi se déclarer avec une signature de départ qui lui est propre.

Définition 3. On dit qu'un protocole *converge multi-latéralement* si pour une assignation initiale R (et donc un ensemble de destinations D) annoncée, les transferts d'informations de routage cessent au bout d'un temps fini.

En réalité la convergence multi-latérale vers D peut être ramenée à une convergence unilatérale. Il suffit d'ajouter un nœud supplémentaire « fictif » noté d de signature δ et des arêtes elles aussi « fictives » d $\to$ p pour tout $p \in D$ vérifiant $l(d,p) \otimes \delta = R(p)$. La situation est illustrée sur la figure 5.

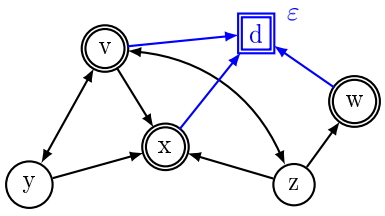


FIGURE 5 – Correspondance entre convergence unilatérale et multilatérale

Du point de vue théorique, la convergence multilatérale est donc totalement équivalente à sa variante unilatérale. En pratique cependant, elles correspondent à deux types d'implémentations distinctes. Celles qui autorisent plusieurs nœuds à être considérées comme destination dans la même instance sont dites *anycast*. Les autres, qui représentent l'immense majorité, sont dites *unicast*.

Pour définir une convergence véritablement plus forte, nous allons considérablement élargir la portée du concept initial :

Définition 4. Un protocole est dit *correct* s'il converge multi-latéralement *pour toute* assignation initiale R et sur *tout sous-réseau*, et qu'aucun des chemins choisis au final ne forme de boucle.

Un protocole correct se situe dans le stade le plus élevé en termes de stabilité, car il convergera toujours quelles que soient les destinations, et peu importe les liens du réseau qui seraient coupés. Un tel objectif n'est pourtant pas inaccessible, comme nous le montrera le théorème 2.

8.2 Modes de convergences

Si la convergence concerne en elle-même la seule terminaison de l'algorithme, il est possible de s'interroger au-delà sur les caractéristiques de ses résultats.

Le *mode de convergence* précise ainsi la nature des chemins finalement choisis par le protocole pour atteindre une destination unique d :

Optimal : Pour tout nœud u, le chemin choisi P_u est optimal parmi tous les chemins de u à d.

Localement optimal : Pour tout nœud u, le chemin choisi P_u est optimal parmi tous les chemins de la forme uv, P_v avec v voisin de u.

Ce dernier mode est en réalité celui obtenu par défaut par tout protocole de type *path-vector*.

Pour expliquer ce point, il est temps de ré-exprimer formellement dans l'algèbre de routage le fonctionnement de ces protocoles.

On note $F^{[k]}$ l'assignation partielle intermédiaire obtenue une fois terminées toutes les transmissions d'informations de routages sur k arêtes. On ajoute la *matrice d'adjacence du réseau* l à composantes dans L telle que pour tous nœuds u et v, $l(u, v)$ désigne s'il y a lieu l'étiquette de l'arête reliant u à v (sinon on pose $l(u, v) = e$). Alors l'application du protocole se traduit par la formule :

$$F^{[k+1]} = l \otimes F^{[k]} \oplus R \tag{J}$$

En toute logique, l'*assignation finale* F obtenue en cas de convergence est un point fixe :

$$F = l \otimes F \oplus R \tag{P}$$

Ainsi, le protocole n'est rien d'autre qu'un algorithme distribué de résolution de l'équation (P).

Nous voyons ici ressurgir Le parallèle mathématique. En effet, l'équation (J) n'est rien d'autre que la *méthode de Jacobi*[2]. Or, dans un dioïde, une solution naturelle de l'équation serait $l^* \otimes R$ où l^* est la *quasi-inverse* de l définie par :

$$l^* = e \oplus l \oplus l^2 \oplus l^3 \oplus \cdots = \sum_{k=0}^{\infty} l^k$$

2. La méthode de Jacobi dans sa déclinaison informatique est parfois aussi connue sous le nom d'*algorithme de Bellman*

(où $l^k = l \otimes l \otimes \cdots \otimes l$ avec k itérations).

Mais avant même que l'on puisse s'interroger sur les conditions d'existence de cette solution providentielle, il faut se résoudre à l'évidence : elle ne fonctionne pas pour les bases de routage. La raison est exactement ce qui marque la frontière entre les deux théories mathématique et informatique : la non-distributivité de $\otimes$ sur $\oplus$.

Malgré tout, comme nous le montrera le théorème 2, il est tout de même possible de faire converger la méthode de Jacobi dans certaines circonstances que nous allons maintenant introduire.

9 Conditions suffisantes

Dans une optique de construction, nous avions présenté dans la partie II la base de routage comme une structure très particulière, par opposition à la généralité des structures utilisant moins d'axiomes. Il s'agissait alors surtout de donner une idée de l'étendue de la théorie, et surtout de hiérarchiser clairement les propriétés de la structure naturelle. C'est donc sur celle-ci que nous nous basons désormais. Ainsi, toutes les algèbres que nous évoquerons dans cette partie III sont des bases de routages, qui disposent pour rappel d'un ordre total noté $\preccurlyeq$ sur l'ensemble Σ des signatures.

Puisque la base de routage ne peut utiliser les outils déjà développés en mathématiques par manque de distributivité, nous somme amenés à nous poser la question suivante : existe-t-il d'autres propriétés permettant d'obtenir la convergence ?

Pour répondre à cette problématique, nous allons procéder cette fois-ci à un travail de déconstruction en partant d'une structure très forte que l'on sait correcte. On extrait donc trois propriétés :

$$\forall\, (\lambda, \sigma) \in L \times \Sigma \;\; : \sigma \preccurlyeq \lambda \otimes \sigma \qquad \text{Monotonie} \qquad \text{(M)}$$

$$\forall\, (\lambda, \sigma) \in L \times \Sigma \;\; : \sigma \prec \lambda \otimes \sigma \qquad \text{Monotonie stricte} \qquad \text{(SM)}$$

$$\forall\, (\lambda, \sigma, \tau) \in L \times \Sigma^2 \;\; : \sigma \preccurlyeq \tau \Longrightarrow \lambda \otimes \sigma \preccurlyeq \lambda \otimes \tau \qquad \text{Isotonie} \qquad \text{(I)}$$

Une algèbre de routage strictement monotone et isotone[3] est en réalité équivalente à la structure de routage la plus naturelle : celle d'un graphe pondéré, où la signature d'un chemin est simplement la somme des poids positifs des arêtes qui le composent. En conséquence, nous baptisons *algèbre des poids* une telle algèbre de routage.

En fait, appliquée à une base de routage, l'isotonie est alors équivalente à la distributivité de $\otimes$ sur $\oplus$. En effet, puisque :

$$\sigma \preccurlyeq \tau \Longleftrightarrow \sigma \oplus \tau = \sigma$$

alors

$$\text{(I)} \Longleftrightarrow \lambda \otimes (\sigma \oplus \tau) = (\lambda \otimes \sigma) \oplus (\lambda \otimes \tau)$$

Il est donc possible d'appliquer la quasi-inverse, à condition de montrer qu'elle existe, ce que la monotonie stricte assure. Une démonstration peut être trouvée dans [1].

En tant que telle, cette structure de routage est optimale du point de vue de la convergence :

3. Certains auteurs utilisent le terme *monotonie* pour ce que nous appelons *isotonie*.

Propriété 1. *Un protocole utilisant une algèbre des poids est correct et converge de façon globale.*

En pratique, l'algèbre des poids est beaucoup trop contraignante pour être applicable à des réseaux inter-régionaux, précisément parce qu'elle utilise la distributivité de $\otimes$ sur $\oplus$, qui impose une cohérence absolue entre tous les choix des routeurs. Pour cette raison, la *monotonie stricte seule* représente un objectif plus adapté. Comme toujours, le gain de généralité entre les deux niveaux se traduit par une convergence plus faible, qui pour cette étape s'exprime par le fait que la convergence globale n'est en général pas garantie. En revanche, le résultat suivant est conservé :

Théorème 1. *Tout protocole utilisant une algèbre de routage* monotone *est correct.*

Le passage de la monotonie stricte à la monotonie simple représente quant à elle le gain suivant :

Propriété 2. *Si une algèbre n'est pas monotone, alors il existe un réseau sur laquelle elle ne converge pas.*

Démonstration. Si l'algèbre n'est pas monotone alors il existe $(\lambda, \sigma) \in L \times \Sigma$ tels que : $\lambda \oplus \sigma \prec \sigma$. Soit un réseau sur lequel l'algèbre s'applique et où se trouve la configuration suivante : Si a et

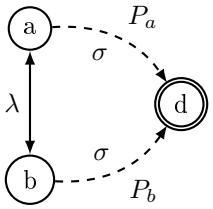


FIGURE 6 – L'instance *bad gadget*

b choisissent P_a et P_b respectivement alors le protocole diverge. En effet, P_b est meilleur que P_a et aP_a est meilleur que P_b. Les deux routeurs changent donc d'avis mais cela même rend leurs nouveaux choix inapplicables. Ils reviennent donc à la situation initiale, et entrent ainsi dans une boucle infinie. □

Le circuit représenté sur la figure 6 est connu sous le nom de *bad gadget* (vilain circuit) et fut introduit dans [5]. Bien évidemment, ce n'est en général pas le seul sous-circuit sur lequel une algèbre non-monotone donnée peut diverger.

C'est à ce stade qu'il convient d'insister sur l'importance de la distinction entre le *réseau* et l'*algèbre*. Dans la pratique, le réseau est en général assez peu flexible, en particulier au niveau inter-régional. Cela explique en partie notre attachement à rendre les algèbres de routage les plus générales possibles. Mais une algèbre a toujours pour vocation de s'appliquer sur un réseau. On peut dès lors entreprendre deux stratégies différentes :

— Créer des algèbres convergentes sur n'importe quel réseau, mais dont les tables d'opérateurs sont difficilement adaptables aux particularités d'un réseau donné.
— Ou bien partir d'un réseau précis et identifier parmi toutes les algèbres possibles celles qui convergeront.

Nous avons établi que la première approche nécessite comme base la monotonie. Les outils spécifiques de construction seront quant à eux développés dans la partie IV.

La seconde approche, en revanche, nous invite à descendre encore plus bas dans la structure, puisqu'elle nous libère de la contrainte d'universalité de la convergence sur tous les réseaux.

10 Théorème général de convergence

Il existe bien une condition nécessaire et suffisante de convergence, dont l'énoncé peut sembler assez naturel, mais qui est pourtant difficile à démontrer :

Définition 5. Dans un réseau associé à une algèbre de routage $A = \langle \Sigma, \preccurlyeq, L, \oplus, \mathcal{O} \rangle$, un cycle de nœuds $u_0 u_1 \dots u_n$ (avec $u_0 = u_n$) est dit *strictement éjectif* si : pour toutes signatures $(\alpha_1, \alpha_2, \dots, \alpha_n) \in (\Sigma \setminus \{\phi\})^n$ telles qu'il existe des chemins de signatures α_i ayant pour origines u_i pour tout $i \in [\![1; n]\!]$, alors il existe un $j \in [\![1; n]\!]$ vérifiant :

$$l(u_j, u_{j+1}) \otimes \alpha_{j+1} \succ \alpha_j \tag{1}$$

où $l(u_j, u_{j+1})$ désigne l'étiquette de l'arête correspondante.

En clair, cette condition signifie simplement qu'**on finira toujours par sortir d'un cycle** strictement éjectif quelle que soit la façon dont on y entre.

Définition 6. Un cycle qui n'est pas strictement éjectif est dit *enfermant*. Il est donc associé à un ensemble de signatures $(\alpha_1, \alpha_2, \dots, \alpha_n) \in (\Sigma \setminus \{\phi\})^n$ et de chemins correspondants tels que pour tout $i \in [\![1; n]\!]$:

$$l(u_i, u_{i+1}) \otimes \alpha_{i+1} \preccurlyeq \alpha_i \tag{2}$$

On peut alors énoncer le résultat majeur de cette partie :

Théorème 2 (Théorème général de convergence, [4])**.** *Un protocole de type* path-vector *composé d'un algèbre de routage et d'un réseau est correct si et seulement si* tous les cycles sont strictement éjectifs.

Le théorème 1 est un corollaire direct du théorème 2.
La démonstration à venir du théorème repose sur l'outil suivant :

Définition 7 (Digraphe)**.** Le *digraphe* d'un réseau associé à une base de routage est un graphe dont les nœuds sont les chemins utilisables du réseau de destination d et qui possède deux types d'arêtes :

$P \hookrightarrow Q$ **:** Q est une extension directe de P : on a $Q = uv \circ P$.

$P \dashrightarrow Q$ **:** P et Q ont même origine, et si α et β sont leurs signatures respectives, on a $\alpha \prec \beta$.

Le digraphe est une synthèse de l'application d'une algèbre sur un réseau puisqu'il représente tous les cheminements possibles des informations de routage.
Un exemple de digraphe sur un réseau simple est donné par la figure 7. Il présente un cycle enfermant qui n'est en réalité qu'une reprise du *bad gadget* présenté sur la figure 6. À noter que l'algèbre n'est pas précisée. En effet, puisque le digraphe ne nous renseigne que sur les comparaisons qui peuvent effectivement se produire sur un réseau donné, de nombreuses algèbres ayant cette même base commune ont le même digraphe sur le réseau en question.
Nous sommes maintenant prêts pour la démonstration du théorème 2.

Sens direct Par l'équivalence vue en 8.1, il suffit de montrer la convergence unilatérale sur tout sous-circuit ainsi que l'absence de boucle au final.
On relie tout d'abord le digraphe au théorème par le résultat suivant :

Lemme 1. *Si tous les cycles du réseau sont strictement éjectifs alors le digraphe est acyclique.*

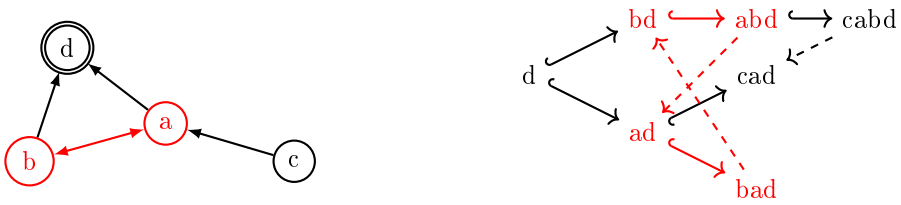


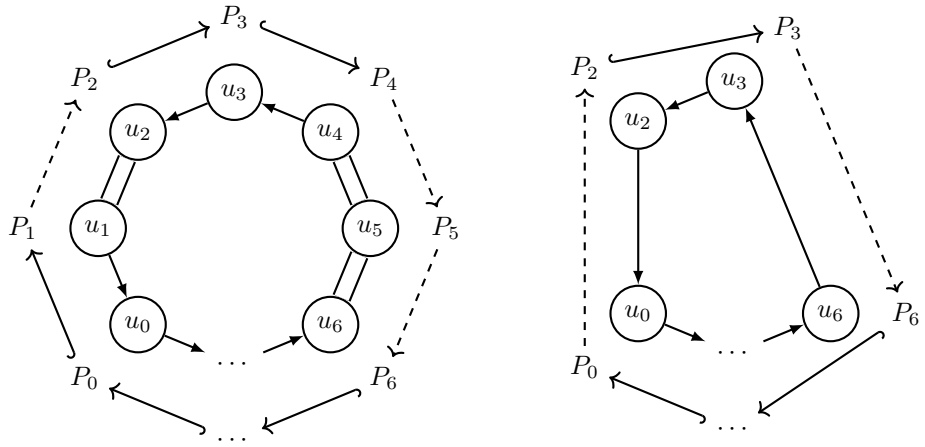
FIGURE 7 – Un réseau et un digraphe possible. Un cycle enfermant est en rouge.

Démonstration. On démontre la contraposée :

Soit $\mathcal{C} = (P_0, \alpha_0)(P_1, \alpha_1)\dots(P_n, \alpha_n)$ (avec $(P_0, \alpha_0) = (P_n, \alpha_n)$) un cycle minimal dans le digraphe.

Notons $\mathcal{K} = u_0 u_1 \dots u_n$ les origines respectives des chemins de $\mathcal{C}$. Montrons que toutes les flèches $P \dashrightarrow Q$ où P et Q ont même origine u_i doivent être consécutives dans $\mathcal{C}$.

En effet, si ce n'est pas le cas, il existe $(j, q) \in [\![1; n]\!]^2$ tels que P_j et P_q ont même origine u_i. Puisque l'ordre de comparaison est totale, on a (par exemple) $P_j \dashrightarrow P_q$. Ceci implique que $(P_j, \alpha_j)(P_{j+1}\alpha_{j+1})\dots(P_q, \alpha_q)$ est un sous-cycle de $\mathcal{C}$, ce qui contredit sa minimalité.



(a) $\mathcal{K}$ (b) $\mathcal{K}'$

FIGURE 8 – Simplification d'un cycle du digraphe

Soit maintenant $\mathcal{K}' = u'_0 u'_1 \dots u'_m$ le chemin obtenu à partir de $\mathcal{K}$ en supprimant toutes les redondances (nécessairement consécutives), comme illustré sur la figure 8. Alors $\mathcal{K}'$ est un cycle minimal dans le réseau.

Étudions une par une les arêtes de $\mathcal{K}'$:

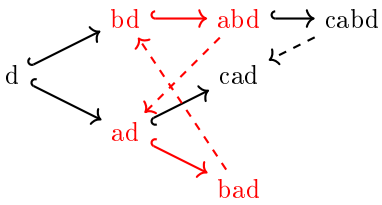
— Si $u'_i \to u'_{i+1}$ était déjà dans $\mathcal{K}$, alors l'arête correspondante dans $\mathcal{C}$ est nécessairement de type $P'_i \hookrightarrow P'_{i+1}$. On a donc par définition $\alpha'_i = l(u'_i, u'_{i+1}) \otimes \alpha'_{i+1}$.

— Sinon, $u'_i \to u'_{i+1}$ correspond à plusieurs arêtes de $\mathcal{C}$, la première étant de type $\hookrightarrow$ et les suivantes de type $\dashrightarrow$. En conséquence, on a : $\alpha'_i \succ l(u'_i, u'_{i+1}) \otimes \alpha'_{i+1}$

Au final, $\mathcal{K}'$ n'est pas strictement éjectif.

□

On peut maintenant démontrer explicitement la convergence unilatérale. Cette deuxième



Rang	Chemin
0	d
1	–
2	abd, bad
3	cabd, bd, ad
4	cad

FIGURE 9 – Tableau des rangs du digraphe 7

partie de la démonstration est indépendante de la première. En effet, elle utilise uniquement le fait que le digraphe est acyclique pour obtenir un invariant de boucle dans l'algorithme de convergence.

Soit $\mathcal{G}$ le digraphe du réseau. Puisque $\mathcal{G}$ est *acyclique* alors on peut le voir comme un arbre de racine d. Pour tout chemin utilisable P du réseau, qui est donc un nœud de $\mathcal{G}$, la distance maximale de P à d est bien définie. On l'appelle *rang* de P et on note M le rang maximal.

Soit la variable temporelle $F = (f_1, f_2, \dots, f_M) \in \mathbb{N}^M$ telle qu'à chaque instant t, et pour tout $j \in [\![1; M]\!]$, on ait :

$$f_j = \text{Nombre de messages en transit à l'instant } t \text{ annonçant un chemin de rang } j \\ + \\ \text{Nombre de routeurs ayant choisi à l'instant } t \text{ un chemin de rang } j.$$

Notez que le temps t peut très bien être continu, du moment que chacune des opérations effectuée par le réseau l'est en temps fini. De même, la simultanéité n'est aucunement nécessaire.

$\mathbb{N}^M$ est muni de l'ordre lexicographique. Nous allons montrer qu'à chaque mise à jour d'un routeur, F décroît strictement. Cet évènement est illustré sur la figure 10.

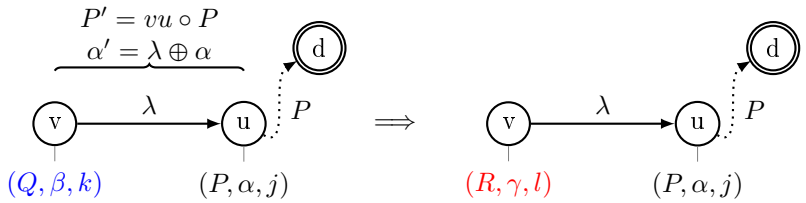


FIGURE 10 – Illustration de la mise à jour d'un routeur

Le routeur v a déjà choisi le chemin Q de signature β et de rang k. À l'instant t, il reçoit un message du routeur w annonçant le chemin P de signature α et de rang j qui devient $P' = vu \circ P$ de signature $\alpha' = \lambda \otimes \alpha$. À l'instant $t' > t$, v décide alors de choisir le chemin R de signature γ et de rang l. Quatre cas sont possibles :

- $R = Q$ **:** v ne change rien et n'annonce rien. L'annonce de P s'éteint : f_j diminue. Donc F diminue.
- $R = P' = vu \circ P$ **:** v annonce R : f_l augmente. L'annonce de P s'éteint : f_j diminue. Mais on a $l > j$ car $P \hookrightarrow R$. Donc F diminue.
- $R \neq P'$ **:** v annonce R : f_l augmente. Q est abandonné : f_k diminue. On a $R \dashrightarrow Q$ donc $l > k$. Ainsi F diminue.

R **est inutilisable :** v n'a plus de choix. Q est abandonné : f_k diminue. L'annonce de P s'éteint : f_j diminue. Donc F diminue.

Dans tous les cas, F diminue. On a donc obtenu un invariant de boucle strictement décroissant : le protocole converge.

□

Réciproque La stratégie de la démonstration de la réciproque est totalement différente de celle du sens directe. On procède ici par contraposée. L'idée consiste alors à ré-exprimer la définition de cycle enfermant sous la forme de sous-conditions intermédiaires sur lesquelles la non-correction sera plus facile à mettre en lumière.

Soit $\mathcal{K} = u_0 u_1 \cdots u_n$ un cycle du réseau. On introduit les définitions suivantes :

Condition T(m) : Il existe des indices $i_0 < i_1 < \cdots < i_{m-1}$ et des signatures distinctes $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ tels que :

$$\forall\, k \in [\![0; m-1]\!] : \alpha_{i_k} \succcurlyeq l(u_{i_k} \mathcal{K} u_{i_{k+1}}) \otimes \alpha_{i_{k+1}}$$

où $l(u_{i_k} \mathcal{K} u_{i_{k+1}}) = l(u_{i_k}, u_{i_k+1}) \otimes l(u_{i_k+1}, u_{i_k+2}) \otimes \cdots \otimes l(u_{i_{k+1}-1}, u_{i_{k+1}})$. En clair : il existe un *sous-cycle* enfermant de longueur m.

Condition S : Il existe $i \in [\![1; n-1]\!]$ et $\alpha \in \Sigma \setminus \{\phi\}$ tels que : $\alpha \succcurlyeq l(u_i \mathcal{K} u_i) \otimes \alpha$. En clair : en partant d'une certaine façon, faire le tour du cycle est mieux ou équivalent.

Condition M : Il existe des indices $i_0 < i_1 < \cdots < i_{m-1}$ pour un certain $m \leq n$ et des signatures distinctes $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ tels que :

$$\forall\, k \in [\![0; m-1]\!] : \begin{cases} \alpha_{i_{k+1}} \succ l(u_{i_{k+1}} \mathcal{K} u_{i_{k+2}}) \otimes \alpha_{i_{k+2}} \\ \alpha_{i_k} \prec l(u_{i_k} \mathcal{K} u_{i_{k+2}}) \otimes \alpha_{i_{k+2}} \end{cases}$$

La condition **M** est illustrée sur la figure 11 à l'aide d'un code couleur : une couleur plus claire correspond à un meilleur chemin.

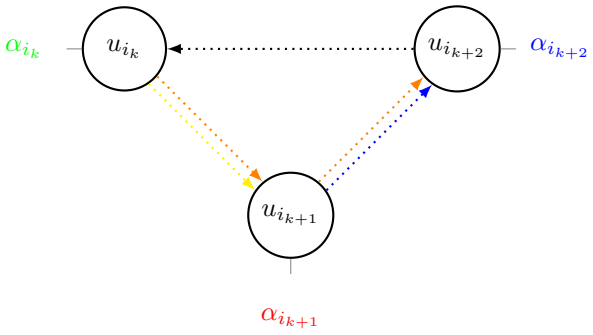


FIGURE 11 – Condition **M**

Lemme 2. *Si $\mathcal{K}$ est un cycle enfermant alors il vérifie une des deux conditions **S** ou **M**.*

Démonstration. $\mathcal{K}$ vérifie **T**(n) où n est sa longueur. Soit m le plus petit entier tel que $\mathcal{K}$ vérifie **T**(m).

Si $m = 1$, on a simplement $\mathbf{T(1)} = \mathbf{S}$.

Sinon, le sous-cycle $(u_{i_0}, \alpha_{i_0}), \ldots, (u_{i_{m-1}}, \alpha_{i_{m-1}})$ vérifiant $\mathbf{T}(m)$ vérifie également $\mathbf{M}$. En effet, supposons que ce ne soit pas le cas. Alors il existe $k \in [\![0; m-1]\!]$ tel que :

$$\alpha_{i_{k+1}} \preccurlyeq l(u_{i_{k+2}} \mathcal{K} u_{i_{k+1}}) \otimes \alpha_{i_{k+2}} \text{ ou } \alpha_{i_k} \succcurlyeq l(u_{i_k} \mathcal{K} u_{i_{k+2}}) \otimes \alpha_{i_{k+2}}$$

Dans le premier cas, d'après $\mathbf{T}(m)$, on a donc :

$$\alpha_{i_{k+1}} = l(u_{i_{k+2}} \mathcal{K} u_{i_{k+1}})$$

D'où :

$$\begin{aligned} \alpha_{i_k} &\succcurlyeq l(u_{i_k} \mathcal{K} u_{i_{k+1}}) \otimes \alpha_{i_{k+1}} \\ &= l(u_{i_k} \mathcal{K} u_{i_{k+1}}) \otimes l(u_{i_k+1} \mathcal{K} u_{i_{k+2}}) \otimes \alpha_{i_{k+2}} \\ &= l(u_{i_k} \mathcal{K} u_{i_{k+2}}) \otimes \alpha_{i_{k+2}} \end{aligned}$$

On obtient donc que le sous-cycle $(u_{i_0}, \alpha_{i_0}), \ldots, (u_{i_k}, \alpha_{i_k}), (u_{i_{k+2}}, \alpha_{i_{k+2}}), \ldots, (u_{i_{m-1}}, \alpha_{i_{m-1}})$ vérifie $\mathbf{T}(m-1)$, ce qui contredit la minimalité du premier sous-cycle.

Dans le second cas, le même argument est directement applicable et permet d'aboutir à la même contradiction. □

Comme annoncé, nous allons maintenant démontrer que chacune des deux conditions **S** et **M** induit la non-correction. Pour rappel, il suffit qu'une seule assignation initiale dans un seul sous-réseau ne converge pas pour que le protocole ne soit pas correct.

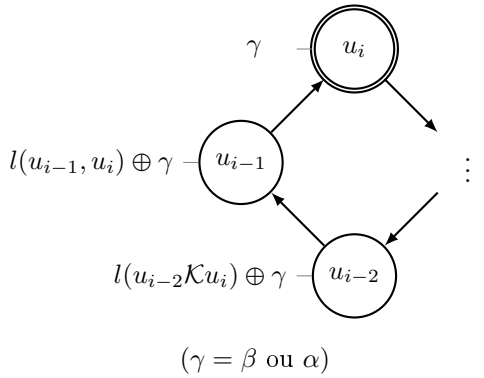


FIGURE 12 – Illustration de la non-correction avec **S**

Pour la condition **S**, il faut distinguer deux sous-cas :

— S'il existe $\beta \preccurlyeq \alpha$ tel que $\beta = (u_i \mathcal{K} u_i) \oplus \beta$, alors le chemin autour de $\mathcal{K}$ possède la même signature que celui de départ. L'assignation initiale qui attribue β à u_i converge. Mais si le routeur physique (la machine) décide de préférer le chemin $\mathcal{K}$, alors on crée une boucle de routage.

— Si on a $\alpha \succ l(u_i \mathcal{K} u_i) \otimes \alpha$ et pas d'égalité pour tout $\beta \prec \alpha$, alors l'assignation qui attribue α à u_i ne converge pas.

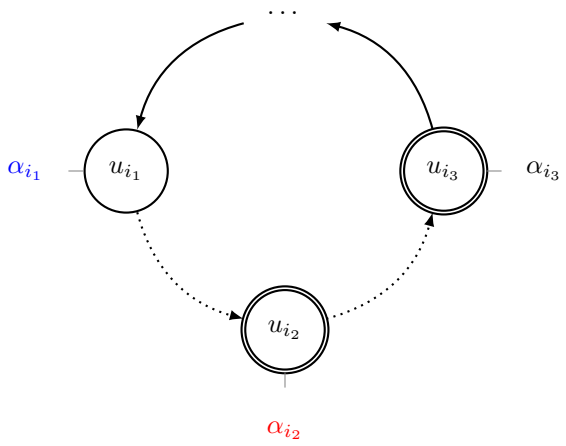
Pour la condition **M**, l'assignation initiale qui attribue chaque α_{i_k} au nœud correspondant u_{i_k} va elle aussi diverger.

Pour cela, nous regroupons les nœuds concernés par **M** par trois. Dans chaque groupe, les deux premiers nœuds ont une influence bien précise sur le troisième, ce qui l'entraîne dans une boucle illustrée sur la figure 13 :

1. u_{i_3} et u_{i_2} annoncent leurs signatures initiales.
2. Au bout d'un certain temps, u_{i_1} reçoit l'information de routage en provenance de u_{i_2}. Puisque $\alpha_{i_i} \succ (u_{i_i} \mathcal{K} u_{i_2}) \otimes \alpha_{i_2}$, u_{i_2} choisit le chemin passant par u_{i_3}. Il annonce ensuite sa nouvelle signature à travers le cycle.
3. Parallèlement, u_{i_2} reçoit l'information de routage en provenance de u_{i_3}. Puisque $\alpha_{i_2} \succ (u_{i_2} \mathcal{K} u_{i_3}) \otimes \alpha_{i_3}$, u_{i_2} modifie également son choix de chemin en faveur de celui passant par u_{i_2} et l'annonce en direction de u_{i_1}.
4. Après un autre intervalle de temps, u_{i_1} reçoit la nouvelle annonce de u_{i_2}. Il annule donc son choix précédent, et puisque $(u_{i_1} \mathcal{K} u_{i_3}) \otimes \alpha_{i_3} \succ \alpha_{i_1}$, il revient à son choix initial de signature α_{i_1}.

Chacun des nœuds impliqués dans **M** joue le rôle de u_{i_1} dans un triplet différent, et donc chacun va effectuer au moins une boucle pour revenir à sa signature d'origine. Toutefois, c'est bien l'annonce coordonnée de leurs signatures initiales par chacun des nœuds (l'étape 1) qui a permis cela, et à ce stade rien ne garantit que ce phénomène se reproduise, et donc que les boucles de chaque nœud se répètent plus d'une fois.

Supposons donc par exemple que les boucles de u_{i_1} et u_{i_2} soient désynchronisées. Autrement dit, lorsque u_{i_2} revient à sa signature α_{i_2} au temps t_2, et lorsque sa nouvelle annonce parvient à u_{i_1} au temps $t'_2 > t_2$, celui-ci n'a pas α_{i_1} comme choix actif. Puisqu'il ne reçoit des informations de routage que de u_{i_2}, son seul autre choix est $(u_{i_1} \mathcal{K} u_{i_2}) \otimes \alpha_{i_2}$. Mais pour que ce soit le cas, il faut nécessairement que u_{i_2} n'ai jamais annoncé autre chose que α_{i_2}, sinon u_{i_1} serait revenu à α_{i_1}. D'où une contradiction. Donc la désynchronisation n'est pas possible et au temps t'_2, u_{i_1} a bien α_{i_1} pour choix actif.

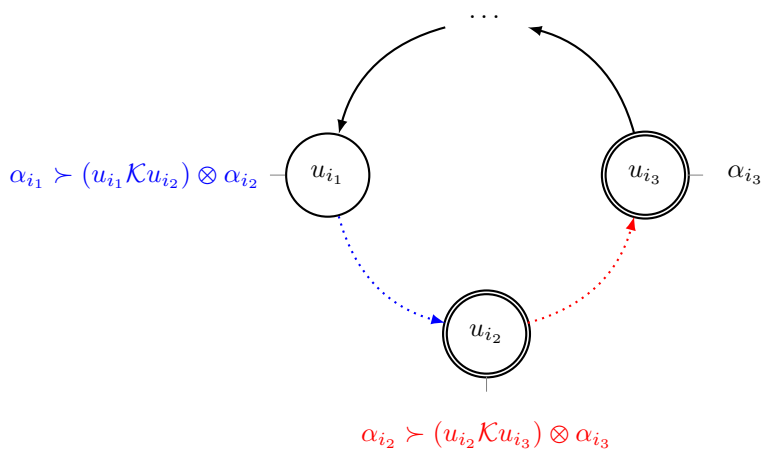


(a) Étape 1 : Annonces de départ

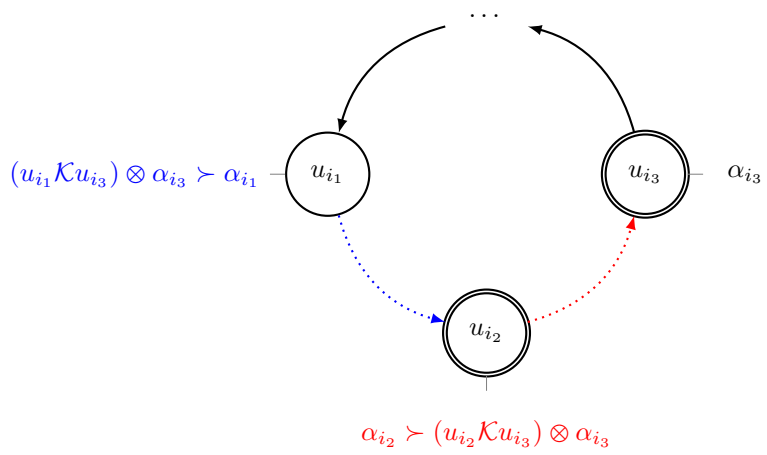
FIGURE 13 – Étapes du sous-processus divergent avec **M**

En étendant le raisonnement à toute paire de nœuds consécutifs, on en déduit que c'est bien l'assignation initiale toute entière qui revient en boucle, et donc que le protocole diverge.

□



(b) Étapes 2 et 3 : u_{i_2} préfère sortir du cycle par u_{i_3} et u_{i_1} préfère sortir par u_{i_2}



(c) Étape 4 : u_{i_1} ne peut plus sortir par u_{i_2} et préfère revenir à son choix initial plutôt que de sortir par u_{i_3}

Il est particulièrement intéressant de noter que la spécificité du multilatéral n'intervient qu'à la toute fin de la démonstration, pour montrer que la sous-condition **M** peut conduire à la divergence.

Bien qu'utile grâce à la rigueur qu'il apporte, le cadre des algèbres de routage n'est pas pleinement nécessaire à la démonstration du théorème. En effet, l'ordre total sur les chemins est une base suffisamment forte et intuitive pour permettre de considérer le problème dans un cadre moins formel. C'est dans dans cette perspective que fut introduit dans [5] le Stable Paths Problem (SPP), dont le résultat principal n'est rien d'autre que le sens direct du théorème 2[4]. Toutefois, contrairement au SPP, la théorie des algèbres de routage n'est pas dépendante de l'ordre total. Elle offre donc un bien meilleur cadre pour la généralisation des résultats, que nous prospecterons dans la partie V.

11 Algorithmes de rectification

Le théorème 2 nous annonce que la façon la moins restrictive d'obtenir la correction sur une base de routage est de supprimer tous les cycles enfermants. Dans cette section, nous allons tenter d'appliquer ce résultat dans divers algorithmes de rectification destinés à modifier une algèbre pour rendre correct (unilatéralement) le protocole associé.

11.1 Rectification locale

Tout d'abord, avant de s'intéresser à *comment* rectifier le protocole, il faut savoir *où* le rectifier. Il est donc nécessaire d'exécuter sur le réseau ou son modèle un algorithme de recherche des cycles enfermants. Nous rappelons simplement que la détection des cycles enfermants signifie la localisation des nœuds mais également des signatures associées incriminées. Une même boucle du réseau peut être liés à plusieurs jeux de signatures formant donc formellement des cycles enfermants distincts, comme l'illustre l'exemple de la figure 14.

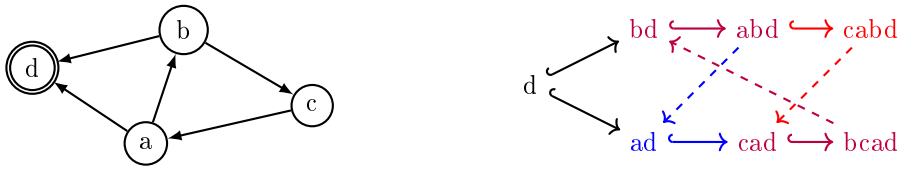


FIGURE 14 – Le digraphe met en évidence deux cycles enfermants supportés par une même boucle du réseau (en bleu et en rouge, avec une partie commune violette).

Heureusement, la définition d'un cycle enfermant implique toutes les arêtes, et il suffit donc de modifier les signatures associées à une seule d'entre elles pour que le cycle devienne strictement éjectif. Si les signatures en question sont impliquées dans plusieurs cycles enfermants, il est donc possible de les corriger tous en même temps avec une unique modification. Ainsi, après la détection, la seconde étape de tout algorithme est celle de l'optimisation. Elle consiste précisément à comparer les cycles pour identifier d'éventuelles arêtes communes (en termes de signatures, s'entend) afin de minimiser le nombre et l'impact des modifications.

Toutes ces consignes sont très générales car leur implémentation technique dépasse le cadre de notre étude puisqu'elle dépend fortement de celle du protocole.

4. L'analogue des cycles enfermants y est appelé *dispute wheel*

Nous nous concentrerons donc sur la troisième étape : la rectification proprement dite. Considérons deux nœuds $u_{i+1} \longleftarrow u_i$ reliés par une arête appartenant à un cycle enfermant, et notons α_{i+1}, α_i les signatures associées. On a donc :

$$\alpha_i \succcurlyeq l(u_{i+1}, u_i) \otimes \alpha_{i+1}$$

Le but de l'opération est d'inverser cette relation. Sachant que les signatures sont organisées selon un ordre total, cela ne sera pas sans conséquences.

Notons par ordre de préférence $\alpha_i^1, \alpha_i^2, \ldots, u_i^n$ les signatures effectives de u_i, c'est-à-dire susceptibles d'être effectivement annoncé au nœud. On pose de plus que $\alpha_i = \alpha_i^k$ et $l(u_{i+1}, u_i) \otimes \alpha_{i+1} = \alpha_i^l$:

$$\alpha_i^1 \prec \alpha_i^2 \prec \cdots \prec \alpha_i^k \prec \cdots \alpha_i^l \prec \cdots \prec \alpha_i^n$$

Supposons qu'il existe un nœud u'_{i+1} relié à u_i par une arête et une signature α_i^r placée comme ceci :

$$\alpha_i^1 \prec \alpha_i^2 \prec \cdots \prec \alpha_i^k \prec \cdots \prec \alpha_i^r \prec \cdots \prec \alpha_i^l \prec \cdots \prec \alpha_i^n$$

et que les deux signatures α_i^r et α_i^l soient les seules empêchant un hypothétique cycle dont ferait partie u'_{i+1} et u_i d'être enfermant. Alors il ne faut surtout pas inverser les positions de α_i^r et α_i^l. Pour régler le problème initial, la solution la plus simple est donc de remonter α_i^k juste au dessus de α_i^l :

$$\alpha_i^1 \prec \alpha_i^2 \prec \cdots \prec \alpha_i^r \prec \cdots \prec \alpha_i^l \prec \alpha_i^k \prec \cdots \prec \alpha_i^n$$

Bien évidemment, le raisonnement doit être inversé si c'est α_i^k qui est concernée. Si aucune des deux signatures n'est concernée alors les deux solutions sont équivalentes. Mais si les deux sont concernées, alors la rectification est impossible par cette approche.

11.2 Rectification inter-régionale

Un aspect sensible du routage que nous n'avons que très peu évoqué est le secret des protocoles. Pour des raisons diverses, de nombreux administrateurs ne communiquent que le minimum d'informations nécessaires dans les signatures, et essaient notamment de faire en sorte qu'elles ne donnent aucune indication sur la nature et la topographie du réseau. Ainsi, les très grands réseaux sont souvent divisés en régions administratives autonomes gérées isolément les unes des autres. Or, les cycles enfermants peuvent très bien traverser plusieurs de ces régions et elles doivent donc impérativement communiquer certaines informations pour les supprimer. Heureusement, nous savons exactement lesquelles. Nous présentons donc un algorithme récursif qui complète le précédent pour un réseau multi-régional :

On suppose que chaque protocole régional est déjà correct. On imagine également que les vérifications inter-régionales sont effectuées par une sorte d'autorité indépendante à laquelle les régions fournissent uniquement les renseignements nécessaires.

1. Chaque région liste tous les chemins utilisables joignant deux de ses routeurs douaniers (ceux qui la relient à une autre, carrés sur la figure 15) et dont les arêtes vérifient toutes l'équation (2). Elle attribue un indicatif à chacun et transmet la liste de ces chemins en précisant uniquement leurs points d'entrée et de sortie.
2. Une fois les informations de toutes les régions obtenues, l'autorité vérifie si les chemins en question forment effectivement un cycle enfermant inter-régional (sachant qu'elle connait le comportement des arêtes inter-régionales).

3. Si un tel cycle existe, il suffit qu'une seule algèbre régionale applique l'algorithme de rectification à une seule arête pour régler le problème. On peut tout aussi bien modifier une arête inter-régionale si nécessaire.

Par récursivité, chaque région peut elle-même appliquer l'algorithme à ses propres sous-régions, et ainsi de suite, en partant toujours du niveau le plus bas.

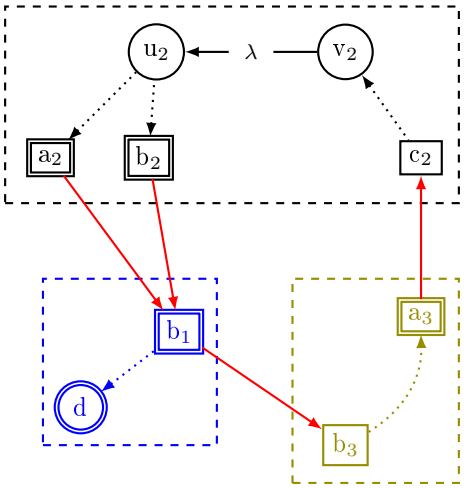


FIGURE 15 – Exemple d'un réseau multi-régional. Les nœuds carrés sont « douaniers ». Ceux qui sont en plus reliés à une arête sortante sont vus comme destination par les autres nœuds de leur région s'ils s'annoncent.

Quatrième partie
Constructeurs sur les algèbres

Dans la partie II, nous avions insisté sur le caractère phénoménologique des algèbres de routage : en toute généralité, elle ne font que décrire l'action des opérateurs, sans plus. Toutefois, cela ne signifie pas que la structure ne se prête pas à inclure une telle description. C'est dans ce sens que nous allons présenter ici des opérateurs internes sur les algèbres ayant pour but de les structurer, repris de [2], et que nous désignons sous le terme de *constructeurs*.

12 Définitions

Tout au long de cette section, on considère deux algèbres de routages $\mathcal{A} = \langle S_\mathcal{A} = \Sigma_\mathcal{A} \uplus L_\mathcal{A}, \oplus_\mathcal{A}, \otimes_\mathcal{A}, \mathcal{O}_\mathcal{A} \rangle$ et $\mathcal{B} = \langle S_\mathcal{B} = \Sigma_\mathcal{B} \uplus L_\mathcal{B}, \oplus_\mathcal{B}, \otimes_\mathcal{B}, \mathcal{O}_\mathcal{B} \rangle$.

12.1 Constructeurs ensemblistes

Le constructeur le plus simple est bien évidemment le produit cartésien :

$$\mathcal{A} \times \mathcal{B} = \langle (S_\mathcal{A} \setminus \{\phi_\mathcal{A}\} \times S_\mathcal{B} \setminus \{\phi_\mathcal{B}\}) \cup \{\phi\}, \oplus_\mathcal{A} \times \oplus_\mathcal{B}, \otimes_\mathcal{A} \times \otimes_\mathcal{B}, \mathcal{O}_\mathcal{A} \times \mathcal{O}_\mathcal{B} \rangle$$

où les opérateurs agissent composantes par composantes. De manière générale, tous les opérateurs ensemblistes classiques peuvent être vus comme des constructeurs de la même façon. Ils peuvent alors s'appliquer à n'importe quelle algèbre.

Il faut toutefois avertir sur une subtilité commune à tous les constructeurs binaires utilisant le produit cartésien : ils ne doivent contenir qu'un seul ϕ qui a la propriété d'être absorbant à travers les composantes, c'est-à-dire que $\forall\ (\alpha, \beta) \in S_{\mathcal{A}} \times S_{\mathcal{B}}\ : (\alpha, \phi_{\mathcal{B}}) = (\phi_{\mathcal{A}}, \beta) = \phi$.

Nous nous intéressons désormais aux constructeurs dont la structure est propre au routage. En conséquence, ils nécessitent un ou plusieurs axiomes de construction pour fonctionner.

12.2 Produit lexical

Définition 8 (Produit lexical)**.** Le *produit lexical* $\mathcal{A} \boxtimes \mathcal{B}$ est défini sur deux algèbres muni d'une relation d'ordre (axiomes 1 à 3) par :

— $S = S_{\mathcal{A}} \times S_{\mathcal{B}}$, $\otimes = \otimes_{\mathcal{A}} \times \otimes_{\mathcal{B}}$.

— $(\alpha, \beta) \preccurlyeq (\alpha', \beta') = \begin{cases} \alpha \preccurlyeq \alpha' & \text{si } \alpha \neq \alpha' \\ \beta \preccurlyeq \beta' & \text{sinon.} \end{cases}$

Ce constructeur est très classique dans le domaine du routage car il permet enfin de structurer l'ordre des signatures en y introduisant l'ordre lexicographique. C'est exactement ce type de structure qui est utilisé dans BGP et dans la majorité des protocoles de type *path-vector*.

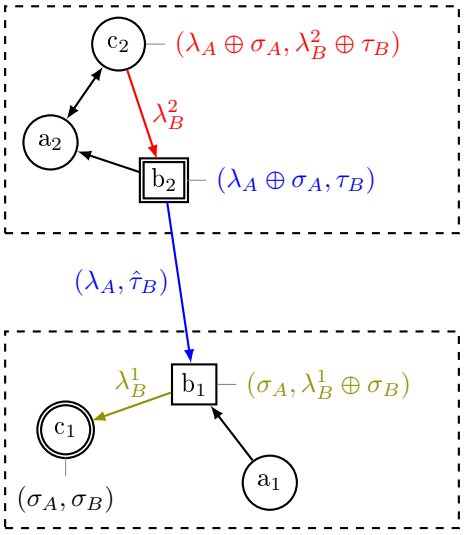


FIGURE 16 – Fonctionnement du produit régional

12.3 Produit régional

Définition 9 (Produit régional)**.** Le *produit régional* $\mathcal{A} \boxdot \mathcal{B}$ est défini sur une algèbre $\mathcal{A}$ idempotente (axiome 3) et une algèbre $\mathcal{B}$ quelconque par :

— $\Sigma = \Sigma_{\mathcal{A}} \times \Sigma_{\mathcal{B}}$, $L = (L_{\mathcal{A}} \setminus \{e\} \times \mathcal{O}_{\mathcal{B}}) \cup L_{\mathcal{B}}$.

— $\lambda_{\mathcal{B}} \otimes (\sigma_{\mathcal{A}}, \sigma_{\mathbf{B}}) = (\sigma_{\mathcal{A}}, \lambda_{\mathcal{B}} \otimes \sigma_{\mathcal{B}})$.

— $(\lambda_{\mathcal{A}}, \tau_{\mathcal{B}}) \otimes (\sigma_{\mathcal{A}}, \sigma_{\mathcal{B}}) = (\lambda_{\mathcal{A}} \otimes \sigma_{\mathcal{A}}, \tau_{\mathcal{B}})$.

Le produit régional permet de formaliser ce que nous avions esquissé dans la section 11.2 : une construction régionalisée des algèbres de routage. L'algèbre $\mathcal{A}$ représente le niveau inter-régional, tandis que l'algèbre $\mathcal{B}$ représente l'union de toutes les algèbres intra-régionales. Les signatures propres à une région sont systématiquement écrasées à la sortie de celle-ci, préservant ainsi le secret du protocole.

À noter que le choix de la politique de $\oplus$ est laissé à l'administrateur. Ainsi, $\oplus$ peut être égal à $\oplus_{\mathcal{A}} \times \oplus_{\mathcal{B}}$ ce qui revient à donner autant d'importance aux deux niveaux. Si les deux algèbres sont munies d'un ordre, $\oplus$ peut être lexicographique, et alors la topologie des régions n'intervient qu'à courte distance pour départager les cas d'égalités. Enfin, $\oplus$ peut être lexicographique inverse (c'est-à-dire avec une préférence ordonnée de droite à gauche) si l'administrateur souhaite privilégier la convergence dans sa propre région.

12.4 Disjonction

Définition 10 (Disjonction)**.** La *disjonction* $\mathcal{A} \lhd \mathcal{B}$ est définie sur deux algèbres quelconques par :

— $S = S_{\mathcal{A}} \uplus S_{\mathcal{B}}$, $\otimes = \otimes_{\mathcal{A}} \Cup \otimes_{\mathcal{B}}$
— $\oplus$ fonctionne comme $\oplus_{\mathcal{A}}$ sur $\Sigma_{\mathcal{A}}$ et réciproquement pour $\mathcal{B}$.
— $\forall\ (\alpha, \beta) \in \mathcal{A} \times \mathcal{B}\ : \alpha \oplus \beta = \beta \oplus \alpha = \alpha$

La disjonction hiérarchise brutalement les algèbres là où le produit lexical conserve les informations des deux. Elle peut cependant être utile dans des situations où une telle préservation n'est pas nécessaire (si la bande passante moyenne est faible par exemple). Elle a également l'avantage d'être plus générale car ne faisant pas appel à la notion d'ordre.

De manière optionnelle, on peut compléter la disjonction pour pouvoir « revenir en arrière » en sautant de $\mathcal{A}$ à $\mathcal{B}$ (ce qui n'est pas possible avec la construction normale). Pour cela, à l'aide d'une injection $f : \Sigma_{\mathcal{A}} \hookrightarrow \Sigma_{\mathcal{B}}$ on introduit une étiquette spéciale s vérifiant :

$$\forall\ \alpha \in \Sigma_{\mathcal{A}} : s \otimes \alpha = f(\alpha) \in \Sigma_{\mathcal{B}}$$
$$\forall\ \beta \in \Sigma_{\mathcal{B}} : s \otimes \beta = \phi$$

13 Préservation de la convergence

Comme nous l'avions évoqué au début de la section 9, une des approches pour obtenir un protocole convergent revient à utiliser une algèbre *strictement monotone*, ce qui garantit la convergence sur n'importe quel circuit. La condition d'absence de circuits enfermants est quant à elle peu adaptée à une telle méthode car elle est essentiellement locale (puisqu'elle ne concerne que les cycles de l'algèbre) là où les autres sont globales. Puisqu'elle vise à créer des algèbres pour elles-mêmes et non pas en fonction du circuit, les constructeurs s'inscrivent naturellement dans cette approche, mais il faut pour cela connaître leurs comportements vis-à-vis de la monotonie stricte.

13.1 Produit cartésien et convergence

Il est assez évident que le produit cartésien conserve la monotonie stricte à condition que les deux algèbres la vérifient. En effet, si $\oplus$ s'applique composante par composante, alors :

$$(\alpha, \beta) \oplus (\lambda \otimes \alpha, \mu \otimes \beta) = (\alpha, \beta) \iff \begin{cases} \alpha \oplus (\lambda \otimes \alpha) = \alpha \\ \beta \oplus (\mu \otimes \beta) = \beta \end{cases}$$

13.2 Produit lexical et convergence

Le comportement du produit lexical vis-à-vis de la monotonie stricte est le suivant :

Propriété 3. *$\mathcal{P} = \mathcal{A}_1 \boxslash \mathcal{A}_2 \boxslash \cdots \boxslash \mathcal{A}_n$ est strictement monotone si et seulement si il existe $k \in [\![1;n]\!]$ tel que : toutes les algèbres $\mathcal{A}_1, \ldots, \mathcal{A}_{k-1}$ sont au moins monotones et $\mathcal{A}_k$ est strictement monotone.*

Démonstration. On démontre la propriété par contraposée. Supposons que $\mathcal{P}$ ne soit pas strictement monotone. Alors il existe $(\alpha, \lambda) \in \Sigma \times L$ telles que $\lambda \otimes \alpha \preccurlyeq \alpha$. De par la structure de $\otimes$, il existe $k \in [\![1;n]\!]$ telle que : $\forall\, i \in [\![1;k-1]\!] : \alpha_i = \lambda_i \otimes_i \alpha_i$ et $\alpha_k \succcurlyeq_k \lambda_k \otimes_k$. Donc $\mathcal{A}_k$ n'est pas monotone. Réciproquement, supposons que pour tout $k \in [\![1;n]\!]$, soit il existe une algèbre non monotone entre $\mathcal{A}_1$ et $\mathcal{A}_{k-1}$, soit $\mathcal{A}_k$ n'est pas strictement monotone. Pour $k = 1$, on obtient que $\mathcal{A}_1$ n'est pas strictement monotone. On a alors deux cas possibles :

- Si $\mathcal{A}_1$ n'est pas non plus monotone, alors $\mathcal{P}$ ne peut être strictement monotone, ni même monotone, car l'ordre de $\mathcal{A}_1$ ne l'est pas et prédomine sur les suivants.
- Si $\mathcal{A}_1$ est monotone, en appliquant la propriété pour $k = 2$ on en déduit que c'est $\mathcal{A}_2$ qui n'est pas monotone, et donc, pour les mêmes raisons que précédemment, $\mathcal{P}$ n'est pas strictement monotone.

□

13.3 Produit régional et convergence

Le comportement du produit régional vis-à-vis des cycles enfermants a déjà été évoqué dans la section 11.2. L'obtention de la monotonie stricte dépend quant à elle du choix qui a été fait pour $\oplus$:

Si $\oplus$ est lexicographique : Il suffit que $\mathcal{A}$ soit strictement monotone. Il n'y a aucun intérêt à utiliser une algèbre $\mathcal{A}$ uniquement monotone, c'est-à-dire avec un ou plusieurs cas d'égalité $\lambda \otimes \alpha = \alpha$. De fait, une arête qui n'aurait aucun effet sur la signature inter-régionale n'est rien d'autre qu'une arête intra-régionale.

Si $\oplus$ est le produit cartésien : $\mathcal{A}$ et $\mathcal{B}$ doivent bien sûr être tous les deux strictement monotones, mais l'écrasement des signatures entre les régions impose des conditions supplémentaires. En effet, dans le cas où deux régions utiliseraient la même algèbre, alors il faudrait s'assurer que pour chaque nœud d'entrée toutes les signatures écrasées seront toujours préférées à la signature écrasante. Avec un léger abus de notation, on peut résumer cela par la formule :

$$\mathcal{O}_{\mathcal{B}} \prec (\Sigma_{\mathcal{B}} \smallsetminus \mathcal{O}_{\mathcal{B}}) \tag{N}$$

Il est possible d'obtenir la monotonie stricte avec une condition moins forte en l'adaptant directement au réseau, mais on abandonne alors l'objectif d'universalité.

Si $\oplus$ est lexicographique inverse : les conditions sont les mêmes que pour le produit lexical, mais avec la nécessité supplémentaire que $\mathcal{B}$ vérifie la condition (N) ci-dessus.

13.4 Disjonction et convergence

Comme on peut s'y attendre, une disjonction n'est strictement monotone que si $\mathcal{A}$ et $\mathcal{B}$ le sont déjà. En effet, elle ne fait qu'accoler les ordres des deux algèbres. À noter que l'étiquette optionnelle s est elle même monotone, car puisque $\forall \in \mathcal{A} : s \otimes \alpha \in \Sigma_{\mathcal{B}}$ on a par définition de la disjonction $\alpha \oplus (s \otimes \alpha) = \alpha$.

Nous avons décrit ici les constructeurs reprenant les structures les plus courantes dans les algèbres de routages. Il en existe d'autres plus poussés que l'on peut retrouver dans [2]. Ce sont essentiellement des perfectionnement de ceux que nous avons présentés, dont le comportement est identique ou similaire. Il n'est absolument pas exclus de développer à l'avenir de nouveaux constructeurs pour un usage particulier. En réalité, c'est même par cette voie que la théorie algébrique du routage et tous les résultats liés pourront le mieux s'intégrer aux protocoles courants.

Cinquième partie

Généralisation des résultats

Dans les parties III et IV, nous avons presque exclusivement considéré des bases de routages, c'est-à-dire des algèbres vérifiant les axiomes 1 à 3. Dans cette partie, nous allons tenter de prospecter ce que deviendrait nos résultats de convergence avec des algèbres utilisant moins d'axiomes, et donc plus générales et plus libres.

Dans cette perspective, il convient de revenir sur les démonstrations afin de bien y cerner le rôle de chaque axiome, et en particulier du dernier l'axiome 4 de sélectivité.

Comme nous l'avions signalé, la démonstration du théorème 2 est en fait composée de deux grandes parties indépendantes :

1. Montrer qu'un cycle enfermant correspond à un cycle du digraphe.
2. Montrer que l'absence de cycle dans le digraphe donne un invariant de l'algorithme de convergence.

La seconde partie est purement algorithmique et ne fait pas appel aux algèbres de routage. Elle pourrait donc potentiellement être réutilisée dans une démonstration généralisée.

En revanche, la première partie, qui se démontre par contraposée, fait dès le départ usage de l'axiome 4 de sélectivité. En réalité, la définition de cycle enfermant est calibrée pour correspondre aux cycles du digraphe, mais l'axiome est cruciale dans cette définition car c'est lui qui permet de garantir que le cycle associé du réseau est minimal.

Il est donc possible de généraliser le théorème et le concept de cycle enfermant en supprimant l'axiome 4 mais au prix de la minimalité. Ces nouveaux cycles enfermants n'étant pas minimaux, ils peuvent inclure plusieurs fois le même nœud, ce qui les rends en pratique impossibles à détecter. Ce résultat n'est donc malheureusement pas utilisable.

S'il est possible de généraliser encore le théorème 2, il sera d'abord nécessaire de trouver un autre invariant de boucle de l'algorithme de routage utilisant une méthode totalement différente de celle que nous avons appliquée avec le digraphe.

Conclusion

Le routage informatique implique souvent une multitude d'aménagements locaux de la part des administrateurs pour s'adapter à leurs contraintes régionales. Malgré cela, nous avons proposé une théorie abstraite descriptive capable par sa généralité de décrire tous les *path-vector protocols*.

La rigueur d'une telle approche permet une description claire des résultats de convergence qu'elle énonce ainsi que de leur portée. Le même principe permet dans l'autre sens de connaître à l'avance les étapes de généralisation de ces résultats, posant les bases d'un nouveau champ de recherche sur la convergence des protocoles.

Remarques personnelles

Ce stage a été l'occasion pour moi de découvrir le fonctionnement d'une équipe de recherche universitaire et appliquée. J'ai ainsi pu travailler sur des sujets complètement ouverts dans le cadre même où la communauté de recherche se retrouve pour les étudier. Cette expérience sera sans doute une référence personnelle dans mes projets futurs aux seins d'autres structures de recherche. Le routage informatique m'est apparu comme une discipline extrêmement spécifique, alors qu'elle est littéralement globale. Elle nous apparait extérieurement comme étant purement technique, et le fait est que sa formalisation est loin d'être exempte de difficultés. La comprendre et la développer m'ont demandé de nombreux efforts pour adapter la rigueur absolue de l'abstraction mathématique à un système qui se devait d'être aussi cohérent que représentatif. La décomposition et la reformulation d'idées qui semblaient au départ purement propres à l'informatique m'ont ainsi ouvert un autre regard sur les théories formelles. Leur généralité n'est donc pas toujours auto-portée mais permet de réelle avancées techniques.

Références

[1] Michel Gondran et Michel Minoux, *Graphes, dioïdes et semi-anneaux, nouveaux modèles et algorithmes*, Tec & Doc, 2001.

[2] Timothy G. Griffin et João Luís Sobrinho, *Metarouting*, SIGCOMM' 05, Philadelphie, Pennsylvanie, Août 2005.

[3] João Luís Sobrinho, *An Algebraic Theory of Dynamic Network Routing*, IEEE/ACM Transactions on Networking, pp. 1160-1173, Octobre 2005.

[4] João Luís Sobrinho, *Correctness of Routing Vector Protocols as a Property of Network Cycles*, IEEE/ACM Transactions on Networking, 2016.

[5] Timothy G. Griffin, F. Bruce Shepherd et Gordon Wilfong, *The Stable Paths Problem and Interdomain Routing*, IEEE Transactions on Networking, Volume 10, Numéro 2 (Avril 2002), pp. 232-243.