Modalités de ressource et contrôle en logique tensorielle

Nicolas Tabareau

Université Paris Diderot (Paris 7)
UFR d'Informatique

Thèse

pour l'obtention du diplôme de

Docteur de l'Université Paris Diderot, spécialité informatique

Modalités de ressource et contrôle en logique tensorielle

présentée et soutenue publiquement par

Nicolas Tabareau

le 3 décembre 2008

devant le jury composé de

M. Nick	Benton	
M. Thomas	Ehrhard	
M. Jean	Goubault-Larrecq	
M. Martin	Hyland	Rapporteur
M. Paul-André	Melliès	Directeur de thèse
M. Luke	Ong	Rapporteur

Remerciements

Mains aux fleurs, Picasso (1958)

Je voulais dire *merci* à Paul-André Melliès pour son encadrement lumineux et son utopie fragile. J'espère que sa vision de la recherche résistera encore longtemps aux assauts politiques.

Je voulais dire *merci* à Martin Hyland pour m'avoir fait profiter de son rayonnement et pour sa lecture précieuse de ce manuscrit.

Je voulais dire *merci* à Luke Ong pour avoir, par une lecture approfondie, fait bénéficier ce manuscrit d'une expertise qui dépasse de loin le cadre de l'informatique théorique.

Je voulais dire *merci* à Nick Benton pour son humour british et pour m'avoir initié à la sémantique de la compilation lors de mon séjour de trois mois chez Microsoft Research.

Je voulais dire *merci* aux membres du jury. Qu'ils sachent que je suis très honoré de leur présence à ma soutenance.

Je voulais dire *merci* aux thésards et aux membres de PPS, ainsi qu'à Odile, pour la bonne humeur et la solidarité qui règnent à Chevaleret.

Je voulais dire *merci* à Christine, Pierre, Sammy, Sylvain et à mon père pour les corrections qu'ils ont apportées à ce manuscrit.

Je voulais dire *merci* à Samuel, mon jumeau de thèse, pour nos discussions chroniques par dessus nos écrans.

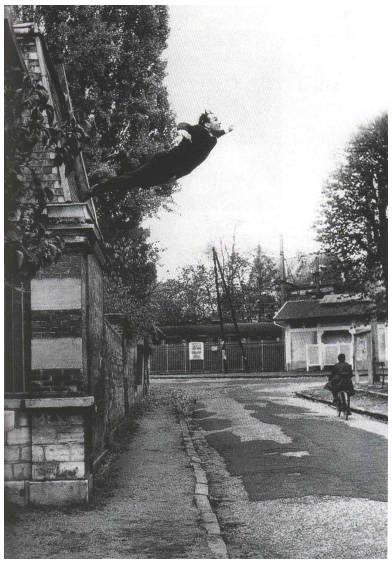
Je voulais dire *merci* à mes amis dont la présence m'a toujours permis de faire la part des choses.

Je voulais dire *merci* à ma famille qui, bien qu'extérieure au monde de la recherche, m'a toujours été d'un grand soutien.

Je voulais dire *merci* à ma princesse, à qui je dois bien plus que notre enfant qui va naître.

Table des matières

Introduction



Le saut dans le vide,
Klein (1960)

Sommaire

Sémantique des jeux et logique linéaire

La sémantique des jeux contemporaine est la petite sœur de la logique linéaire : née (ou plutôt ressuscitée à partir des travaux de l'école de Paul Lorenzen [Lor61, LL78]) au début des années 90, dans le tourbillon intellectuel engendrée par la récente découverte de la logique linéaire (LL) par Jean-Yves Girard [Gir87], la sémantique des jeux resta pendant longtemps sous son influence spirituelle. À cette époque, le rayonnement de la logique linéaire sur la sémantique des jeux était très fructueux. Grâce à ce tuteur, la sémantique des jeux a fleuri en suivant uniformément l'intuition que toute *formule* de la logique linéaire décrit un *jeu* ; et que toute *preuve* de cette formule décrit une *stratégie* pour jouer à ce jeu.

Cette correspondance entre les formules de la logique linéaire et les jeux est entretenue par plusieurs analogies frappantes que nous nous attacherons à décrire ici. Tout d'abord, un des principes de base de la logique linéaire est que la négation

$$A \mapsto \neg A$$

est *involutive*. Cela signifie en substance que toute formule A est égale (ou au moins isomorphe) à sa double négation

$$A \quad \cong \quad \neg\neg A. \tag{1}$$

Ce principe est très bien reflété en sémantique des jeux par l'intuition que nier un jeu A consiste à permuter le rôle de Joueur et Opposant. Ainsi, nier un jeu deux fois revient à permuter deux fois Joueur et Opposant, c'est-à-dire à ne rien faire. Typiquement, si A est

le jeu d'échecs standard où les Blancs commencent, $\neg A$ est le jeu d'échecs où les Noirs commencent, et $\neg\neg A$ est le jeu d'échecs standard à nouveau. Un autre principe de base de la logique linéaire est que toute formule agit comme une ressource qui disparaît une fois consommée. En particulier, une preuve de la formule

$$A \multimap B$$

permet de déduire la conclusion B en utilisant (nous disons plutôt : en consommant) exactement une fois l'hypothèse A, vue ici comme une ressource. À nouveau, ce principe est reflété en sémantique des jeux par l'intuition que jouer à un jeu, c'est comme consommer une ressource – le jeu lui-même.

Les connecteurs de la logique linéaire sont reflétés de manière convaincante en sémantique des jeux. Par exemple, le produit tensoriel

$$A \otimes B$$

de deux formules A et B est interprété par le jeu (ou la formule) A joué en parallèle avec le jeu (ou la formule) B, où Opposant est le seul à pouvoir décider de changer de jeu. Cela revient à poser deux échiquiers sur la même table et à dire que les Noirs doivent répondre sur l'échiquier où les Blancs viennent de jouer. Similairement, la somme

$$A \oplus B$$

de deux formules A et B est interprétée par le jeu où Joueur joue en premier, en choisissant entre le jeu A et le jeu B, et continue dans la composante sélectionnée. Cela revient à poser deux échiquiers sur la même table et à laisser les Noirs décider s'ils veulent jouer sur celui de gauche ou sur celui de droite. Ce choix est ensuite irréversible. Enfin, la modalité exponentielle de la logique linéaire

$$!A$$

appliquée à la formule A est interprétée comme le jeu où plusieurs copies du jeu A sont jouées en parallèle, et où Opposant est le seul à pouvoir

- passer d'une copie à une autre ;
- ouvrir une nouvelle copie du jeu A.

Cela revient à jouer sur des échiquiers en parallèle comme pour le tenseur avec en plus la capacité pour les blancs d'aller chercher un nouvel échiquier et de l'ajouter à ceux déjà présents.

Ce que nous venons de décrire correspond en essence à la sémantique des jeux introduite par Andreas Blass dans [Bla92]. Simple et élégant, ce modèle reflète fidèlement la gestion des ressources de la logique linéaire. De manière amusante, cette sémantique des jeux est pour ainsi dire antérieure à la logique linéaire elle-même [Bla72]. Elle a d'abord été abordée indirectement par Gérard Berry et Pierre-Louis Curien [BC82] à travers le modèle des structures de données concrètes qui constitue la première sémantique interactive des algorithmes séquentiels. La sémantique des jeux fut ensuite utilisée directement pour obtenir des modèles complets de la logique linéaire multiplicative [AJ94, HO92]. Ces travaux ont été poursuivis ensuite sur des fragments plus étendus de la logique linéaire [Lam95, BDER97]. La sémantique des jeux a aussi été rapprochée du calcul de la réduction linéaire de tête effectuée par certaines machines abstraites [DHR96].

Une sécession avec la logique linéaire. La sémantique des jeux s'est ensuite émancipée de la logique linéaire dans le milieu des années 90, afin de réaliser ses propres desseins hérités de la sémantique dénotationnelle :

1. Le désir d'interpréter des *programmes* écrits dans des langages utilisant des effets (récursion, états, etc.) et de caractériser exactement leurs comportements interactifs à l'intérieur de modèles *pleinement adéquats* (*fully abstract* en anglais).
2. Le désir de comprendre les principes algébriques des langages de programmation et des effets, en utilisant la machinerie catégorique.

Une nouvelle génération de sémantiques des jeux s'est alors développée, conduite par (au moins) deux lignes différentes de recherche :

1. Samson Abramsky et Radha Jagadeesan [AJ94] ont signalé que le modèle (ou plutôt la variante alternée du modèle) de Blass ne définit pas de modèle catégorique de la logique linéaire. Pire : il ne définit même pas une catégorie, car la composition n'est pas associative. Samson Abramsky a décrit ce phénomène dans [Abr03] en le baptisant *le problème de Blass*.
2. Martin Hyland et Luke Ong [HO00] ont introduit la notion de *jeu d'arène*, et ont caractérisé le comportement interactif des programmes écrits dans le langage fonctionnel PCF – correspondant au λ-calcul pur muni d'un test conditionnel, de l'arithmétique et d'un opérateur de récursion. Samson Abramsky, Radha Jagadeesan et Pasquale Malacaria ont obtenu le même résultat avec un modèle légèrement différent [AJM00]. Notons que malgré leurs dates de publication, ces travaux ont tous les deux été réalisés au cours de l'année 1994.

Ainsi, le problème de Blass indique qu'il est difficile de construire une sémantique de jeux (séquentiels) pour la logique linéaire ; les jeux d'arène sont devenus dominants vers le milieu des années 90, bien qu'ils ne donnent pas lieu à un modèle de logique linéaire. Ces deux raisons (au moins) ont opéré une scission entre la sémantique des jeux et la logique linéaire : il fut soudainement accepté que les catégories de jeux (séquentiels) et de stratégies ne captureraient que des *fragments* (intuitionniste ou polarisé) de la logique linéaire mais pas la totalité de cette logique.

Une réconciliation à travers la logique tensorielle. Afin de comprendre en profondeur les modalités de ressource de la logique linéaire en sémantique des jeux, il faut donc réunir ces deux sujets. Comme le contentieux est apparu avec la théorie des catégories, il semble naturel que leur réunion apparaisse au niveau catégorique.

Nous expliquons comment réaliser cette réconciliation en *relâchant* les contraintes de la négation involutive, ce qui donne lieu à la définition d'une négation tensorielle. Cette négation induit à son tour une monade de *continuation linéaire* dont l'unité

$$A \longrightarrow \neg\neg A \tag{2}$$

raffine l'isomorphisme (1) de la logique linéaire. Passer de la négation involutive à la négation tensorielle signifie que nous remplaçons la logique linéaire par une logique plus primitive que nous appelons *logique tensorielle*. La notion catégorique associée, c'est à dire une catégorie monoïdale symétrique munie d'une négation tensorielle, est appelée *catégorie de dialogue*.

La logique tensorielle fournit une relecture de la logique polarisée introduite par Jean-Yves Girard dans son travail sur la logique classique et le système LC [Gir91]. Un phénomène inattendu apparaît dans ces logiques polarisées : les modalités de ressource changent la polarité des formules. Ce phénomène curieux s'explique en logique tensorielle par une décomposition de l'opérateur de soulèvement $^{!}_{\mathrm{pol}}$ en deux constructeurs logiques : la modalité exponentielle ! qui ne change pas la polarité des formules, et la négation tensorielle,

notée dans ce cadre $\downarrow$ (plutôt que $\neg$), dont le rôle de négation est d'échanger le point de vue Opposant et Joueur sur la formule – ce qui revient à en renverser la polarité

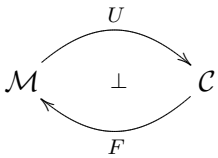
$$_{\text{pol}}!A \;=\; !\downarrow A.$$

Il s'agit de promouvoir un changement de perspective radical sur les logiques polarisées. En effet, la logique tensorielle n'est pas réduite pour nous à un fragment de la logique linéaire, comme on a pris l'habitude de penser les logiques polarisées. Au contraire, nous défendrons la thèse selon laquelle la logique tensorielle est une logique plus primitive que la logique linéaire, et plus proche des mécanismes de continuation décrits par la sémantique des jeux. Et de la même manière que la logique classique s'interprète en logique intuitionniste par traduction de Gödel, nous verrons que la logique linéaire s'interprète par une traduction similaire, de nature catégorique (construction de Kleisli). En un mot, la logique tensorielle est à la logique linéaire ce que la logique intuitionniste est à la logique classique : un formalisme plus proche du calcul et des programmes.

Calculer des algèbres libres

Notre point de vue sur la logique tensorielle est axé sur sa sémantique catégorique. Aussi nous reformulons et généralisons la notion d'exponentielle présente dans la logique linéaire grâce à un concept majeur en théorie des catégories : l'adjonction.

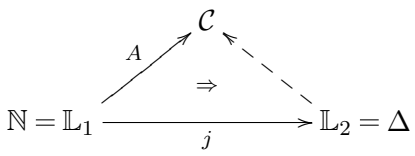
Calculer des modalités de ressource. Une modalité de ressource est par définition décrite par une adjonction monoïdale



entre une catégorie monoïdale symétrique $\mathcal{C}$, modèle de logique tensorielle, et une catégorie monoïdale symétrique $\mathcal{M}$, internalisant les propriétés de la modalité considérée. Dans le cas de la modalité exponentielle, la catégorie $\mathcal{M}$ est cartésienne avec sa diagonale modélisant la contraction et la counité de la comonade associée modélisant l'affaiblissement. On retrouve la construction de l'exponentielle du modèle Linéaire/Non Linéaire introduit par Nick Benton [Ben95]. Cette formulation par adjonction avec une catégorie cartésienne met en valeur le fait que pratiquement toutes les exponentielles présentes dans la littérature sont en fait des comonoïdes commutatifs. Mais en y regardant de plus près, on s'aperçoit que ces exponentielles partagent une propriété supplémentaire qui n'est pas requise dans l'axiomatique : elle sont souvent obtenues de manière libre. Il semble alors naturel de se tourner vers les travaux d'Eduardo Dubuc [Dub74], et plus récemment de Bruno Vallette [Val04] et Steve Lack [Lac08] sur le calcul des monoïdes libres afin d'en déduire une construction générique des exponentielles ; et de l'utiliser en sémantique des jeux.

Malheureusement calculer des comonoïdes commutatifs libres en sémantique des jeux n'est pas chose aisée et les résultats déjà présents dans la littérature algébrique ne permettent pas de les appréhender ; le problème principal étant que les catégoriciens ont été motivés par des exemples mathématiques où tout est assez lisse. Les exemples qui nous motivent sont à contrario très pauvres en propriétés, les catégories de jeux ayant par exemple très peu de limites. Nous avons donc été amenés à développer notre propre théorie pour comprendre le calcul des comonoïdes commutatifs libres sous des climats plus arides.

Théories linéaires. Le point de départ de notre travail est d'harmoniser le calcul de monoïdes libres et le calcul de comonoïdes commutatifs libres en considérant les modèles libres d'une théorie linéaire (PRO) ou symétrique (PROP) – les notions de théorie linéaire et symétrique étant les versions monoïdales et monoïdales symétriques de la notion de théorie algébrique due à William Lawvere [Law63]. De ce point de vue, le calcul du monoïde libre dans une catégorie monoïdale $\mathcal{C}$ correspond au calcul de l'extension de Kan monoïdale d'un objet A de $\mathcal{C}$ – vu comme un foncteur monoïdal de la catégorie discrète des entiers $\mathbb{N}$ dans $\mathcal{C}$ – le long de l'injection j de $\mathbb{N}$ dans la catégorie simpliciale Δ.



Dans le glossaire des théories linéaires, on dit qu'on calcule le $\mathbb{L}_2$-modèle libre du $\mathbb{L}_1$-modèle A selon le transport de théories linéaires $j : \mathbb{L}_1 \to \mathbb{L}_2$. On peut alors s'appuyer sur les formules génériques bien connues pour le calcul d'extensions de Kan ; la seule chose qui reste à faire est de garantir que cette extension soit monoïdale. Il s'avère que cette tâche est plus ardue qu'il n'y paraît et nécessite encore un cran supplémentaire d'abstraction. Cela va être réalisé en plaçant notre raisonnement au niveau 2-dimensionnel.

La vague 2-dimensionnelle. Emporté par la vague 2-dimensionnelle déferlant actuellement dans la communauté des catégoriciens, nous avons abstrait notre raisonnement à un niveau bicatégorique basé sur la théorie des distributeurs de Jean Bénabou [Bé73]. L'utilisation du concept d'équipement en distributeurs développé par Richard Wood [Woo82, Woo85] nous a permis de décrire de manière minimale les propriétés nécessaires aux théories linéaires en question et à la catégorie $\mathcal{C}$ pour que le calcul de modèles libres soit possible. Notre résultat général repose sur deux hypothèses fondamentales :

- **Opéradicité** : le transport de théories linéaires vérifie une hypothèse *combinatoire*. Intuitivement, cette hypothèse indique une propriété de décomposition en arbres des morphismes de $\mathbb{L}_2$; propriété toujours vérifiée lorsque les deux théories linéaires proviennent d'opérades.
- **Complétude algébrique** : la catégorie $\mathcal{C}$ sur laquelle on calcule le $\mathbb{L}_2$-modèle libre vérifie une hypothèse *algébrique*. Intuitivement, cette hypothèse indique que $\mathcal{C}$ possède certaines colimites et que celles-ci commutent avec la structure monoïdale. Pour pouvoir parler abstraitement de complétude pour certaines colimites, nous avons dû développer la notion de *système de Yoneda* dans un équipement en distributeurs.

Notre résultat est conservatif par rapport aux travaux de Dubuc, Vallette et Lack au sens où leurs théorèmes sont des cas particuliers du nôtre. Mais la notion de complétude algébrique que nous utilisons est beaucoup plus flexible car elle requiert l'existence de très peu de colimites, capturant ainsi le cas des exponentielles libres dans les espaces de cohérence et en sémantique des jeux.

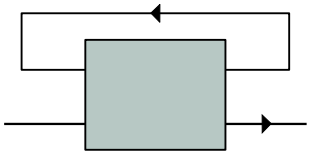
Grâce à ces outils, il nous est maintenant possible de définir une catégorie de jeux modélisant la logique tensorielle, où les modalités de ressources sont calculées librement comme des limites de diagrammes appropriés. Nous nous appuyons pour cette construction sur le modèle des jeux de Conway introduit par André Joyal [Joy77]. Ce choix a été motivé par l'existence, dans ce modèle, d'une structure compacte fermée dont découle une notion de trace qui est au cœur de notre interprétation des références.

Trace et références

Les catégories monoïdales tracées [JSV96] ont été introduites par André Joyal, Ross Street et Dominic Verity afin de fournir une description uniforme de divers constructions mathématiques ayant un comportement cyclique. Ces catégories sont équipées d'un opérateur de trace

$$\mathrm{Tr}^{X}_{A,B} \quad : \quad \mathcal{C}_1(X \otimes A, X \otimes B) \;\to\; \mathcal{C}_1(A, B)$$

qui permet intuitivement de brancher la sortie sur X à l'entrée en X. Il est usuel de représenter la trace, au travers de cette intuition, par le schéma suivant



Parmi les constructions les plus notables décrites par les catégories monoïdales tracées, nous citerons la fermeture des tresses en théorie des nœuds et l'opérateur de trace en algèbre linéaire.

Ici, nous nous intéressons à la trace comme moyen de description des variables locales dans les langages de programmation. Traditionnellement en sémantique [Mog91], on interprète un langage de programmation dans une catégorie en distinguant les objets A décrivant les valeurs, des objets TA décrivant les calculs de type A, où T est une monade. Dans le cas des références, la monade considérée est la monade d'état $S \multimap (S \otimes \_)$ qui permet d'interpréter un programme de type $A \to B$ comme un programme prenant une valeur A et renvoyant un calcul $S \multimap (S \otimes B)$, ce qui, via la clôture monoïdale, correspond à un morphisme de $S \otimes A \to S \otimes B$. Dès lors, si on est capable de prendre la trace sur S de f, on obtient un morphisme avec mémoire interne de type $A \longrightarrow B$. C'est l'analogue de la restriction (ou localisation) utilisée dans les calculs de processus pour un langage où les canaux sont remplacés par des adresses mémoires.

Pour étayer cette idée, nous voulons utiliser notre modèle de logique tensorielle basé sur les jeux de Conway.

Multiparenthésage. Samson Abramsky, Kohei Honda et Guy McCusker [AHM98] ont introduit une sémantique des jeux pour donner un modèle pleinement adéquat d'un langage de type ML avec références d'ordre supérieur. Il apparaît que la catégorie des jeux de Conway que nous avons définie est étroitement liée à ce modèle. Il nous manque seulement une notion de parenthésage afin de contraindre le déroulement d'une interaction. Mais avoir une telle notion, sans pour autant briser la structure compacte fermée de la catégorie des jeux de Conway, n'est pas une mince affaire. Cela nous a amené à une refonte totale du parenthèsage pour aboutir à la notion de *multiparenthésage*.

Cette notion repose sur la gestion de requêtes initiées par Joueur et Opposant. Cette vision duale du parenthésage permet de préserver la structure compacte fermée des jeux de Conway. Ces requêtes sont gérées par une relation de résidus – empruntée à la théorie de la réécriture – qui permet de détecter si un joueur à accéder à une requête. Nous avons ensuite donné une description axiomatique de la gestion de ces requêtes à travers un opérateur de ressource. Celui-ci s'apparente à un calcul de distance entre deux positions d'un jeu ; ou plus précisément au calcul d'une pseudométrique où la notion de temporalité est apparente. Ainsi, la distance entre deux positions est un couple d'entiers positifs, chacun correspondant au nombre de requêtes initiées par Opposant (resp. Joueur) entre ces deux positions et auxquelles on a pas encore accédé. Cet opérateur vérifie deux propriétés naturelles ayant des équivalents standards en théorie des mots ou en géométrie : la

domination par suffixe et la *sous-additivité*. Bien que très simple, cette axiomatique nous permet de construire, de manière purement algébrique, des preuves souvent laborieuses en sémantique des jeux (comme la compositionalité des stratégies bien parenthésées).

Tout cela rend possible la définition d'une catégorie de jeux multiparenthésés équipée d'une trace. Cette catégorie fournit un modèle de la logique tensorielle où les modalités de ressource sont calculées grâce à notre travail sur le calcul des algèbres libres. Il apparaît que la catégorie de jeux introduite par Abramsky et al. est isomorphe à la catégorie de Kleisli induite par la modalité de ressource exponentielle sur notre catégorie de jeux multiparenthésés. Nous utilisons donc directement leur résultat pour obtenir la pleine adéquation de notre modèle. L'intérêt de notre travail réside dans la définition de la stratégie $\texttt{cell}_A$ qui interprète une cellule mémoire de type A. Contrairement aux travaux précédents, cette stratégie est obtenue graduellement en utilisant la négation tensorielle, la structure compacte fermée et la notion d'accès mémoire pour une modalité exponentielle. Cet accès mémoire réduit la gestion des copies accomplie par une cellule mémoire à la présence d'une transformation naturelle de composante

$$\xi_{A,B} \quad : \quad !(A \otimes !B) \longrightarrow !A \otimes !B$$

satisfaisant deux lois de cohérence avec la déréliction et la contraction de la modalité exponentielle. Ainsi, on va construire pas à pas trois types de cellules :

- la cellule linéaire dans laquelle on écrit une seule fois et on lit une seule fois (cette construction utilise la négation tensorielle et la structure compacte fermée) ;
- la cellule constante dans laquelle on écrit une fois et on lit autant de fois que l'on veut (cette construction utilise la modalité exponentielle) ;
- la cellule mémoire dans laquelle on écrit et on lit autant de fois que l'on veut (cette construction utilise l'accès mémoire de la modalité exponentielle).

Nous montrons alors, de manière diagrammatique, que ces stratégies valident des équations sémantiques sur les références. Cela nous permet donc de reformuler la correction de notre système en évitant les longues preuves par étude de cas sur les parties jouées par une stratégie – preuves qui sont d'ailleurs bien souvent absentes des textes car elles sont très difficiles à mettre en place.

Multicatégories de contrôle

Une des motivations de la logique tensorielle est de fournir un cadre plus souple à l'interprétation des langages de bas niveau de type assembleur. Ceci permettrait d'appréhender la sémantique de la compilation de façon plus uniforme en ayant un langage commun pour décrire les modèles haut et bas niveau. L'étude du langage de bas niveau que nous avons en tête est construite sur une sémantique de « style passage par continuation » (CPS). Elle permet d'établir des propriétés sur les fragments de code assembleur parlant de la divergence plutôt que de la terminaison.

Logique tensorielle vs théorie des continuations. Nous souhaitons combiner harmonieusement la logique tensorielle et la théorie des continuations. Si on se penche du côté des langages de programmation, il est naturel de voir l'environnement – aussi appelé contexte d'évaluation ou continuation – d'un programme de type A comme un programme du type dual $\neg A$. Nous avons mentionné plus haut que dans la logique tensorielle, il existe une preuve canonique de A vers $\neg\neg A$ mais que ces deux formules n'y sont pas identiques. Il est bien connu en théorie des continuations qu'appliquer la négation une nouvelle fois sur le type $\neg A$ ne redonne pas le type A lui-même, mais plutôt le type un peu plus libéral $\neg\neg A$.

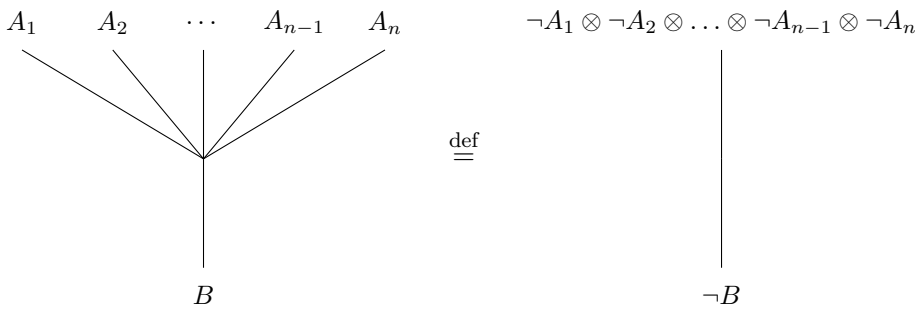


FIG. 1 – Un morphisme dans la multicatégorie de continuation

Ce phénomène est reflété en sémantique catégorique par l'existence d'une monade de continuation $\neg\neg$ associée à toute négation tensorielle sur une catégorie monoïdale $\mathcal{C}$. La preuve canonique (2) est ensuite interprétée dans la catégorie $\mathcal{C}$ par l'unité de la monade.

Il est aussi bien connu en théorie des continuations que la monade $\neg\neg$ est forte. Ceci crée donc un lien tangible entre continuation et négation tensorielle.

Un cadre cartésien bien exploré. Nous voulons préciser ce lien à un niveau catégorique. Pour cela, nous pouvons nous appuyer sur une littérature assez riche tant qu'il s'agit du cadre cartésien. Yves Lafont, Bernhard Reus et Thomas Streicher [LRS93] ont observé, en utilisant une traduction par continuation, que le fragment de la logique intuitionniste, où l'implication est remplacée par la négation, est suffisant pour interpréter le λ-calcul. Plus tard, Martin Hofmann et Thomas Streicher [HS97] ont défini une sémantique du $\lambda\mu$-calcul en appel par nom, à l'aide d'une sémantique catégorique des continuations induite par une catégorie de réponses. Ils ont établi de plus que leurs modèles des continuations sont complets parmi tous les modèles du $\lambda\mu$-calcul en appel par nom.

La première tentative de caractérisation algébrique des catégories de continuation a été faite par Hayo Thielecke [Thi97] au cours de son étude des $\otimes\neg$-catégories. Pour cela, Thielecke a introduit l'idée importante selon laquelle ces catégories doivent être prémonoïdales et non monoïdales.

Peter Selinger a ensuite introduit la notion de catégorie de contrôle [Sel01] et a établi un théorème fondamental de structure, stipulant qu'une telle catégorie est toujours la catégorie de continuation d'une catégorie de réponses. Notre travail est largement inspiré de ce résultat, et des techniques que Selinger a introduites pour le démontrer.

Dans sa thèse, Olivier Laurent [Lau02a] a suggéré une notion de catégorie de contrôle linéaire, à mi-chemin entre les notions de catégorie de contrôle et de catégorie linéairement distributive de Robin Cockett, et Robert Seely [CS92]. Laurent a établit que de telles catégories fournissent toutes un modèle de MALLP, une version polarisée du fragment multiplicatif et additif de la logique linéaire. La définition d'une catégorie de contrôle linéaire, contrairement aux catégories de contrôle, ne capture pas les propriétés précises des modèles de continuation induits par une catégorie de réponses monoïdale.

Une version monoïdale basée sur les multicatégories. Notre tâche est donc précisément de combler ce manque. Il apparaît, après réflexion, que l'objet canonique de notre étude est plus naturellement une multicatégorie de continuation. Un morphisme avec n entrées $A_1, A_2, \ldots, A_{n-1}, A_n$ et une sortie B de cette multicatégorie est interprété par un morphisme de $\neg A_1 \otimes \neg A_2 \otimes \cdots \otimes \neg A_{n-1} \otimes \neg A_n$ vers $\neg B$ dans la catégorie de dialogue associée, comme le décrit la Figure 1.

Nous nous attachons alors à définir une notion de multicatégorie de contrôle afin de décrire exhaustivement la structure présente dans une multicatégorie de continuation.

Afin de donner une définition synthétique d'une multicatégorie de contrôle comme une multicatégorie équipée d'une notion d'idéal exponentiel et d'une structure prémonoïdale, nous proposons un nouvel éclairage sur les multicatégories, basé sur la notion d'empreinte d'une multicatégorie. Nous montrons ainsi que toute multicatégorie de contrôle est équivalente à une multicatégorie de continuation. La clef de voûte de la preuve repose sur la définition du centre d'une multicatégorie de contrôle, définition inspirée de la notion de centre d'une catégorie prémonoïdale. Nous montrons ensuite que ce centre définit une catégorie de dialogue dont la multicatégorie de continuation associée est équivalente à la multicatégorie de contrôle de départ.

Une vision uniforme de la compilation

Comme nous l'avons indiqué plus haut, nous avons l'intuition que la logique tensorielle peut aussi nous aider à comprendre les langages de bas niveau et la compilation d'un langage de haut niveau vers ces langages. En utilisant nos résultats sur la négation tensorielle et les multicatégories de contrôle, nous aimerions avoir une sémantique uniforme tout au long de la compilation et ce, du langage de haut niveau jusqu'à l'assembleur. On sait déjà que la logique linéaire est un bon candidat pour interpréter les termes du langage de haut niveau. On sait aussi que les continuations linéaires fournissent un cadre solide pour donner la sémantique d'un langage assembleur. Idéalement, on doit pouvoir interpréter ces deux langages dans le cadre commun de la logique tensorielle. Cette idée est d'ailleurs confortée par le développement récent d'une sémantique des jeux pour la logique de séparation (ou plus précisément de la logique BI) introduite par Guy McCusker et David Pym [MP07]. Nous n'avons pas encore réalisé ce saut conceptuel mais nous pouvons présenter ici les premiers résultats obtenus.

Un compilateur certifié ? Affirmatif, et quoi d'autre ? En particulier, nous souhaitons construire un compilateur dont on sait prouver la correction. En effet, on suppose généralement que les compilateurs ont une sémantique transparente : le code compilé « se comporte comme le dit la sémantique du programme source ». Malheureusement, les compilateurs – et plus particulièrement les compilateurs optimisés – sont des programmes complexes qu'on ne peut pas déboguer facilement par une simple phase de tests. La vérification d'un compilateur est un problème largement étudié comme l'atteste la bibliographie de Maulik Dave [Dav03]. La certification en Coq a même été récemment étudiée par Xavier Leroy et son équipe [Ler08]. La nouveauté de notre approche réside dans l'utilisation d'une sémantique relationnelle permettant à la fois de prouver la correction du compilateur mais aussi de prouver la correction de certaines optimisations du compilateur de manière modulaire. L'approche relationnelle permet aussi d'avoir un raisonnement équationnel sur les fragments de code : on peut montrer l'égalité de deux termes compilés directement au niveau assembleur sans passer par le langage de haut niveau.

Une sémantique relationnelle. Comme le langage de haut niveau que nous voulons analyser contient les fonctions récursives, il est difficile de garantir la terminaison lors du typage. Il est donc insuffisant de s'intéresser uniquement à des fragments de code qui terminent. Dans ce cadre, la notion qui devient pertinente pour relier deux programmes est la *divergence*, c'est à dire le fait que l'exécution se poursuit pour toujours. On dira que deux programmes p et p' sont reliés par la relation R aux points de programmes l et l',

ce que nous noterons

$$\models p, p' \rhd l, l' : R^{\top}$$

si pour n'importe quel couple (s, s') d'états mémoire reliés par R, la divergence de p à partir de l'état s au point de programme l est équivalente à la divergence de p' à partir de l'état s' au point de programme l'. On dira dans ce cas que les deux programmes *équidivergent* pour la condition R. La propriété de correction de fragments de code assembleur reposera alors sur une version relationnelle de style « passage par continuation » (CPS) du triplet de Hoare [Hoa69]. Rappelons que le triplet de Hoare indique comment un fragment de code c modifie la mémoire en passant d'un état vérifiant le prédicat P_{pre} à un état vérifiant le P_{post}. On note usuellement ce triplet

$$\{P_{pre}\}\ c\ \{P_{post}\}.$$

La version relationnelle et CPS du triplet de Hoare est alors

$$\models p, p' \rhd l_{post}, l'_{post} : {R_{post}}^{\top} \ \Rightarrow \models p, p' \rhd l_{pre}, l'_{pre} : {R_{pre}}^{\top}.$$

Ce jugement indique que si les programmes p et p' équidivergent pour la condition R_{post} aux points l_{post} et l'_{post}, alors ils équidivergent pour la condition R_{pre} aux points l_{pre} et l'_{pre}.

Pour cette étude, nous avons choisi comme langage de haut niveau PCF et nous avons considéré un langage assembleur basique. Nous avons donné une sémantique du langage assembleur en nous inspirant de la théorie des continuations, de la réalisabilité et de la logique de séparation. Nous avons ensuite écrit un compilateur en Coq du langage PCF vers notre assembleur, et nous avons prouvé, toujours en Coq, que le code compilé avait le comportement souhaité. La formalisation en Coq comporte un peu moins de 14000 lignes de code et repose pour l'essentiel sur la réécriture offerte par le module **Setoid**. Ce travail a été réalisé en collaboration avec Nick Benton du laboratoire Microsoft Research à Cambridge.

Plan de la thèse

Cette thèse s'articule autour de cinq chapitres relativement indépendants, précédés d'un chapitre général rappelant quelques préliminaires catégoriques.

Chapitre 1. Le Chapitre 1 est un rappel de certaines notions catégoriques indispensables à la compréhension de ce manuscrit : 2-catégorie, adjonction, monade, catégorie monoïdale ou prémonoïdale, idéal exponentiel, théories algébriques, PRO, etc.

Chapitre 2. Le Chapitre 2 introduit la *négation tensorielle* ainsi que les *modalités de ressource*, le tout formant ce que nous appelons la *logique tensorielle*. Nous présentons ensuite une reformulation commune des espaces cohérents et de finitude comme des modèles de logique linéaire obtenus à partir d'un même modèle de logique tensorielle dont on fait varier la négation. Ce modèle, basée sur une construction par recollement à partir d'une catégorie de modules, est muni de plusieurs négations dont les monades de continuation sont toutes commutatives. Enfin, nous présentons un modèle de logique tensorielle inspiré des jeux de Conway.

Chapitre 3. Le Chapitre 3 présente une construction d'algèbres libres d'une T-théorie enrichie. Cette construction, basée sur la notion d'équipement en distributeurs, repose sur deux propriétés : l'une de nature combinatoire, *l'opéradicité* ; et l'autre de nature algébrique, *la complétude algébrique*. Nous expliquons ensuite comment cette construction permet de reformuler les résultats déjà connus d'Eduardo Dubuc, et plus récemment de Bruno Vallette et Steve Lack, sur le calcul des monoïdes libres. Enfin, nous utilisons notre outil pour calculer les exponentielles libres dans les espaces de cohérence et les jeux de Conway.

Chapitre 4. Le Chapitre 4 présente un modèle pleinement adéquat d'un langage de type ML avec références d'ordre supérieur. Ce modèle est construit à partir des jeux de Conway que l'on a munit d'une notion de *multiparenthésage* permettant de contraindre les interactions, tout en préservant la structure compacte fermée. Nous utilisons le résultat d'Abramsky, Honda et Mc Cusker pour démontrer la pleine adéquation de notre modèle. Notre modèle se distingue par le fait que la stratégie $\texttt{cell}_A$, interprétant la cellule mémoire de type A, est définie de manière interne en utilisant la négation tensorielle, la trace, et une notion d'accès mémoire pour la modalité exponentielle. Nous sommes aussi capable de décrire d'autres types de cellules comme la cellule mémoire linéaire ou la cellule mémoire constante.

Chapitre 5. Le Chapitre 5 présente une extension au cadre monoïdal du travail de Peter Selinger sur les catégories de contrôle. Le passage du cadre cartésien au cadre monoïdal nous impose de considérer des *multicatégories de contrôle*. On définit alors une multicatégorie de continuation induite par une catégorie de dialogue (catégorie monoïdale symétrique munie d'une négation tensorielle) ; et on montre un théorème de structure : toute multicatégorie de contrôle est équivalente à une multicatégorie de continuation.

Chapitre 6. Le Chapitre 6 présente la certification en Coq d'un compilateur d'un langage de type PCF vers un langage assembleur. Les propriétés vérifiées par les fragments de code assembleur sont décrites par une sémantique relationnelle constituant une catégorie monoïdale symétrique et cartésienne, munie d'une notion de dualité proche de la négation tensorielle. Nous utilisons, de manière cruciale dans notre preuve, le module **Setoid** de Coq permettant de faire de la réécriture sur les arbres de tenseurs décrivant l'état de la mémoire.

Chapitre 1

Préliminaires

Nu bleu II, Matisse (1952)

Sommaire

1.1 Catégories

La nécessité de dégager la structure abstraite d'un objet mathématique pour mieux en comprendre son essence est à la base du développement de la théorie des catégories. Nous renvoyons le lecteur novice à l'incontournable monographie de Saunders MacLane [ML71] et supposons donc connues les notions de base comme les foncteurs, les adjonctions, les limites etc. Nous prenons néanmoins quelques pages pour fixer les notations en rappelant certaines définitions et pour livrer notre point de vue sur certaines notions souvent abordées de façon différente en fonction de la communauté mathématique à laquelle on appartient.

En guise d'exemple introductif que nous filerons au cours de ces rappels, considérons la catégorie des groupes. Le groupe libre F_S généré par un ensemble S peut être décrit par le diagramme universel

$$\begin{array}{ccc} S & \xrightarrow{\eta_S} & F_S \\ & \searrow^{f} & \downarrow{\scriptstyle \exists!\ f^\dagger} \\ & & G \end{array} \tag{1.1}$$

qui indique que pour toute fonction ensembliste f de S dans un groupe G – pour lequel on a oublié momentanément sa structure de groupe – il existe une unique homomorphisme

de groupe $f^\dagger$ – appelé *relèvement de* f vérifiant

$$f^\dagger \circ \eta_S \;=\; g.$$

Au lieu d'étudier isolement l'objet (ici un groupe) qui possède une structure donnée, la théorie des catégories met l'accent sur les morphismes et les processus qui préservent la structure entre deux objets. Il apparaît qu'en étudiant ces morphismes, on est capable d'en apprendre plus sur la structure des objets.

Dans notre illustration, les morphismes étudiés sont les homomorphismes de groupes. Un homomorphisme de groupe entre deux groupes préserve la structure de groupe. L'étude des homomorphismes de groupe fournit alors un outil pour étudier les propriétés générales des groupes et les conséquences des axiomes relatifs aux groupes.

La théorie des représentations illustre bien ce propos. En effet, elle consiste à étudier un groupe G via son action sur un espace vectoriel V, cette action étant décrite par un homomorphisme de groupes de G vers $GL(V)$ – le groupe des automorphismes de V.

Il existe un mécanisme similaire dans la plupart des théories mathématiques. Une catégorie est une formulation axiomatique qui permet de relier des structures mathématiques aux fonctions qui préservent leur structure. Une étude systématique des catégories permet de prouver des résultats généraux à partir des axiomes d'une catégorie.

Rappel 1.1 (Catégorie)
Une *catégorie* $\mathcal{C}$ est une structure composée de quatre données :

- d'une classe $\mathcal{C}_0$ dont les éléments sont appelés *objets* ;
- d'un ensemble $\mathcal{C}_1(A, B)$ pour chaque paire d'objets A et B dont les éléments sont appelés *morphismes* (ou *flèches*), et sont souvent notés

  $$f : A \to B$$

  l'objet A étant alors appelé *source* de f et l'objet B étant appelé *but* de f. On a l'habitude d'appeler cet ensemble de flèches le *Hom-set* de A vers B ;
- d'une opération

  $$- \circ - : \mathcal{C}_1(B, C) \times \mathcal{C}_1(A, B) \to \mathcal{C}_1(A, C)$$

  appelée *composition*, associant à tous morphismes $f : A \to B$ et $g : B \to C$, le morphisme *composé*

  $$g \circ f : A \to C;$$

- d'un morphisme

  $$\mathrm{id}_A : A \to A$$

  pour tout objet A appelé *identité* sur A ;

qui satisfont les axiomes suivants :

- *associativité* : pour tous objets A, B, C et D, et tous morphismes $f : A \to B, g : B \to C$ et $h : C \to D$, on a

  $$(h \circ g) \circ f \;=\; h \circ (g \circ f);$$

- *élément neutre* : pour tous objets A et B, et tout morphisme $f : A \to B$, on a

  $$\mathrm{id}_B \circ f \;=\; f \;=\; f \circ \mathrm{id}_A.$$

Remarque 1.2 (problème de taille)
Il est usuel de se restreindre aux catégories dites *localement petites*, c'est-à-dire pour lesquelles chaque Hom-set est un ensemble et pas uniquement une classe. Cela évite des problèmes de taille. Nous ne pouvons pas le faire dans cette thèse car nous aurons besoin par exemple de parler de la catégorie des catégories localement petite qui n'est pas localement petite.

Par la suite, nous allons rajouter des structures sur des catégories, et utiliser ces structures pour démontrer des résultats généraux. Nous conseillons au lecteur – qui peut se trouver parfois pris sous une avalanche de concepts – de toujours spécialiser les définitions dans les deux cas de bases suivant :

1. Si la catégorie $\mathcal{C}$ ne possède qu'un seul objet ($\mathcal{C}_0 = \{*\}$), alors les lois d'associativité et d'élément neutre assurent que les morphismes sur l'unique objet de $\mathcal{C}$ ont une structure de *monoïde* ;
2. Si il y a au plus un morphisme entre deux objets, alors l'ensemble des morphismes peut être vu comme une relation et les lois d'associativité et d'élément neutre assure que cette relation est un *préordre*.

Remarquons que nous aurions pu annoter la composition et les identités par la catégorie $\mathcal{C}$ dans laquelle elles vivent. Ainsi, lorsqu'on parle de plusieurs catégories différentes, cela évite la confusion. Pourtant, nous avons choisi de ne pas alourdir la notation car la catégorie sous-jacente de la composition se déduit des morphismes auxquels elle s'applique (idem pour les identités).

Exemple 1.3
À titre d'exemple donnons quelques catégories fondamentales.

- La catégorie **Ens** dont les objets sont les ensembles et les morphismes sont les fonctions ensemblistes avec la composition et les identités habituelles.
- La catégorie **Rel** dont les objets sont les ensembles finis et les morphismes sont les relations avec la composition et les identités habituelles.
- La catégorie **Grp** dont les objets sont les groupes et les morphismes entre deux objets $(A, +, 0, -)$ et $(B, \times, 1, ()^{-1})$ sont les homomorphismes de groupe, c'est-à-dire les fonctions $f : A \to B$ telles que

$$\forall m, n \in A, \quad f(m+n) = f(m) \times f(n)$$

  avec la composition et les identités habituelles.
- La catégorie **Grph** dont les objets sont les graphes et les morphismes sont les morphismes de graphes. Pour nous, un graphe G est donné par un ensemble de sommets V_G et un ensemble d'arêtes $E_G \subseteq V_G \times V_G$; ou de manière plus catégorique, par un graphe est un diagramme

$$E_G \underset{\text{but}}{\overset{\text{source}}{\rightrightarrows}} V_G$$

  dans **Ens**. Un morphisme de graphes $f : G \to G'$ est donné par une fonction qui associe un sommet de G' à chaque sommet de G, et qui à chaque arête $v \xrightarrow{e} v'$ de G associe une arête $f(v) \xrightarrow{f(e)} f(v')$ de G'.

Rappel 1.4
Étant donnée une catégorie $\mathcal{C}$, on définit sa *catégorie opposée* $\mathcal{C}^{\text{op}}$ qui a les mêmes objets que $\mathcal{C}$ mais dont les flèches sont inversées

$$\mathcal{C}_1^{\text{op}}(A, B) = \mathcal{C}_1(B, A).$$

Une catégorie est elle-même un type de structure mathématique pour lequel il existe une notion de morphismes préservant sa structure. De tels morphismes sont appelés foncteurs.

Rappel 1.5 (Foncteur)
Un *foncteur* $F : \mathcal{C} \to \mathcal{D}$ entre deux catégories $\mathcal{C}$ et $\mathcal{D}$ est la donnée :

- d'une fonction $F : \mathcal{C}_0 \to \mathcal{D}_0$ qui à tout objet A de $\mathcal{C}$ associe un objet FA de $\mathcal{D}$;
- d'une fonction $F_{A,B} : \mathcal{C}_1(A, B) \to \mathcal{D}_1(FA, FB)$ (souvent simplement notée F) qui à tout morphisme $f : A \to B$ de $\mathcal{D}$ associe un morphisme $Ff : FA \to FB$ de $\mathcal{D}$;

qui satisfont les axiomes suivants :

- *préservation de la composition* : pour tous objets A, B et C, et tous morphismes $f : A \to B$ et $g : B \to C$ de $\mathcal{C}$, on a

$$F(g \circ f) \;=\; Fg \circ Ff;$$

- *préservation des identités* : pour tout objet A de $\mathcal{C}$, on a

$$F\ \mathrm{id}_A \;=\; \mathrm{id}_{FA}.$$

Un foncteur F est dit *plein* (resp. *fidèle*) lorsque la fonction $F_{A,B}$ est surjective (resp. *injective*) pour toute paire d'objets A et B de $\mathcal{C}$.

On voit apparaître pour la première fois le *caractère introspectif* de la théorie des catégories : les foncteurs définissent eux-mêmes une notion de morphismes sur les catégories.

Exemple 1.6 (Catégorie des catégories)
On note **Cat** la catégorie dont les objets sont les catégories et dont les foncteurs $F : \mathcal{C} \to \mathcal{D}$ sont les morphismes entre deux objets $\mathcal{C}$ et $\mathcal{D}$. La composée $G \circ F$ de deux foncteurs F et G – parfois simplement notée GF – est obtenue en composant de manière ensembliste les fonctions définies sur les objets et sur les morphismes. Le foncteur identité sur une catégorie $\mathcal{C}$, noté $\mathrm{Id}_{\mathcal{C}}$, en obtenu en prenant l'identité (ensembliste) sur les objets et les morphismes.

Lorsque la classe $\mathcal{C}_0$ d'objets de $\mathcal{C}$ est elle-même un ensemble, on dit que la catégorie est *petite*. On note alors **SCat** la restriction (pleine) de **Cat** aux catégories petites.

Après cette dynamique de mise en abîme si propre aux catégories, mentionnons un deuxième phénomène majeur : *l'élévation dans les dimensions supérieures*. Ainsi, il n'est pas seulement possible de définir des morphismes (foncteurs) entre deux catégories, mais il est aussi possible de définir des morphismes (transformations naturelles) entre deux foncteurs. Ces transformations naturelles sont dites *évoluées à la dimension 2* (les catégories représentant la dimension 0, et les foncteurs la dimension 1).

Rappel 1.7 (Transformation naturelle)
Soit deux catégories $\mathcal{C}$ et $\mathcal{D}$ et deux foncteurs $F : \mathcal{C} \to \mathcal{D}$ et $G : \mathcal{C} \to \mathcal{D}$, une *transformation naturelle* $\theta : F \to G$ entre les foncteurs F et G est une famille $\theta_A : FA \to GA$ de morphismes de $\mathcal{D}$, indexée sur les objets A de $\mathcal{C}$ telle que pour tous objets A et B et tout morphisme $f : A \to B$ de $\mathcal{C}$ le diagramme

$$\begin{array}{ccc} FA & \xrightarrow{Ff} & FB \\ {\scriptstyle\theta_A}\downarrow & & \downarrow{\scriptstyle\theta_B} \\ GA & \xrightarrow[Gf]{} & GB \end{array}$$

commute dans $\mathcal{D}$.

Revenons maintenant sur notre exemple du groupe libre. Nous avons pu décrire à l'intérieur de la catégorie des groupes **Grp** la notion de groupe libre via un diagramme universel. Mais il est aussi possible de décrire le groupe libre de manière externe en étudiant

le lien entre la catégorie des ensembles **Ens** et la catégorie des groupes **Grp**. Il existe un foncteur

$$U \quad : \quad \mathbf{Grp} \ \to \ \mathbf{Ens},$$

appelé *foncteur d'oubli*, qui consiste simplement à voir un groupe comme l'ensemble des éléments qui le compose. Maintenant, il apparaît que la fonction qui à un ensemble associe le groupe libre sur cet ensemble peut être étendue à un foncteur

$$F \quad : \quad \mathbf{Ens} \ \to \ \mathbf{Grp}.$$

Il existe un lien entre U et F qui caractérise (à isomorphisme près) le foncteur F : c'est la notion d'*adjonction*.

Rappel 1.8 (Adjonction)
Une *adjonction* entre deux catégories $\mathcal{C}$ et $\mathcal{D}$ est la donnée de deux foncteurs $F : \mathcal{C} \to \mathcal{D}$ et $G : \mathcal{D} \to \mathcal{C}$ et d'une famille de bijections

$$\varphi_{A,B} : \mathcal{D}_1(FA, B) \ \cong \ \mathcal{C}_1(A, GB)$$

pour tous objets A de $\mathcal{C}$ et B de $\mathcal{D}$, naturelle en A et B. On utilise la notation $F \dashv G$ et on dit que le foncteur F est adjoint à gauche de G pour indiquer une telle situation.

On note souvent la bijection de la façon suivante :

$$\frac{FA \xrightarrow{\ f\ } B}{A \xrightarrow{\ \varphi(f)\ } GB}$$

Cette notation permet d'exprimer la naturalité en A et B de façon plaisante

$$\frac{FA' \xrightarrow{\ Fa\ } FA \xrightarrow{\ f\ } B}{A' \xrightarrow{\ a\ } A \xrightarrow{\ \varphi(f)\ } GB} \qquad \frac{FA \xrightarrow{\ f\ } B \xrightarrow{\ b\ } B'}{A \xrightarrow{\ \varphi(f)\ } GB \xrightarrow{\ Gb\ } GB'}$$

Si on regarde à nouveau notre exemple du groupe libre, l'existence d'une adjonction $F \dashv U$ signifie qu'il y a autant de morphismes ensemblistes de S vers UG que de morphismes de groupes de FA vers G. Ceci (plus la naturalité de la bijection) nous donne une formulation équivalente à celle utilisant la propriété universelle de l'unité $\eta_s : S \to F_S$ décrite par le Diagramme 1.1 en début de chapitre.

Remarquons qu'une adjonction peut aussi être présentée par les deux familles de morphismes canoniques

$$\varphi_{A,FA}(\mathrm{id}_{FA}) : A \to GFA \quad \text{et} \quad \varphi^{-1}_{GB,B}(\mathrm{id}_{GB}) : FGB \to B,$$

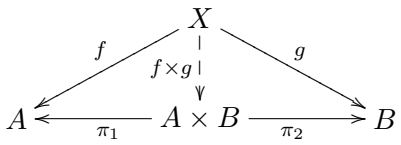
appelés respectivement *unité* et *counité*. Ces deux familles définissent des transformations naturelles

$$\eta : \mathrm{id}_{\mathcal{C}} \to GF \quad \text{et} \quad \varepsilon : FG \to \mathrm{id}_{\mathcal{D}}.$$

Lorsque celles-ci sont des isomorphismes, on parle alors d'une *équivalence de catégories*.

Nous terminons ces petits rappels en présentant les catégories cartésiennes. Ces catégories ont un statut un peu spécial en informatique car elles sont à la base de la sémantique du λ-calcul et donc de la sémantique des langages fonctionnels en général.

Rappel 1.9 (Produit)
Le produit de deux objets A et B dans un catégorie $\mathcal{C}$ est la donnée d'un objet souvent noté $A \times B$ et de deux morphismes $\pi_1 : A \times B \to A$ et $\pi_2 : A \times B \to B$ appelés respectivement *première et deuxième projection* vérifiant la propriété universelle : pour toute paire de morphismes $f : X \to A$ et $g : X \to B$, il existe un unique morphisme noté $f \times g : X \to A \times B$ qui fasse commuter le diagramme



Rappel 1.10 (Objet terminal)
Un objet 1 d'une catégorie $\mathcal{C}$ est dit *terminal* lorsque pour tout objet A de $\mathcal{C}$, il existe un unique morphisme de A dans 1. Par exemple, la catégorie **1** qui a un seul objet et un seul morphisme est terminal dans **Cat**.

Rappel 1.11 (Catégorie cartésienne)
Un *catégorie cartésienne* $\mathcal{C}$ est une catégorie équipée à la fois d'un objet terminal 1 et d'un produit $A \times B$ pour toute paire d'objets (A, B) de la catégorie $\mathcal{C}$.

1.2 2-Catégories

De nombreuses notions catégoriques comme les adjonctions ou les monades (voir Définition 1.19) ont une définition qui semble indépendante du cadre dans lequel on se situe (monoïdale, cartésien, etc.). Ainsi, la définition d'une adjonction monoïdale semble être juste une spécialisation de la définition d'adjonction pour les catégories monoïdales. En fait, cela vient du fait que la notion d'adjonction se décrit naturellement dans une structure appelé *2-catégorie* munie de morphismes à la dimension 2 *appelés 2-cellules*. On peut voir une 2-catégorie comme une catégorie enrichie sur **Cat** (voir la Section 3.2.1 pour plus de détails). Nous souhaitons pourtant passer cela sous silence pour le moment, car cela compliquerait inutilement les préliminaires de ce chapitre. Nous donnons donc préférablement une définition directe.

Définition 1.12 (2-catégorie)
Une *2-catégorie* $\mathcal{C}$ est la donnée :
- d'une classe $\mathcal{C}_0$ dont les éléments sont appelés *0-cellules* ;
- d'une catégorie $\mathcal{C}_1(A, B)$ pour toute paire d'objets A et B, les objets $f : A \to B$ de cette catégorie sont appelés *1-cellules* et les morphismes $\alpha : f \Rightarrow g$ de cette catégorie sont appelés *2-cellules*, la loi de composition de cette catégorie est notée $\bullet$ et est appelée *composition verticale* ;
- d'un foncteur $- \circ - : \mathcal{C}_1(B, C) \times \mathcal{C}_1(A, B) \to \mathcal{C}_1(A, C)$ appelé *composition horizontale* ;
- d'un foncteur $\mathrm{id}_A : \mathbf{1} \to \mathcal{C}_1(A, A)$ pour tout objet A, appelé *identité horizontale* sur A ;

qui satisfont les axiomes suivants :

– $\circ$ est associative : le diagramme

$$\begin{array}{ccc}
(\mathcal{C}_1(C,D)\times\mathcal{C}_1(B,C))\times\mathcal{C}_1(A,B) & \xrightarrow{\alpha} & \mathcal{C}_1(C,D)\times(\mathcal{C}_1(B,C)\times\mathcal{C}_1(A,B)) \\
\downarrow{\scriptstyle \circ\times\mathcal{C}_1(A,B)} & & \downarrow{\scriptstyle \mathcal{C}_1(C,D)\times\circ} \\
\mathcal{C}_1(C,D)\times\mathcal{C}_1(A,C) & & \mathcal{C}_1(B,D)\times\mathcal{C}_1(A,B) \\
& {\scriptstyle\circ}\searrow \quad \swarrow{\scriptstyle\circ} & \\
& \mathcal{C}_1(A,D) &
\end{array} \tag{1.2}$$

commute ;

– id_A est l'élément neutre de la composition $\circ$: le diagramme

$$\begin{array}{ccccc}
\mathbf{1}\times\mathcal{C}_1(A,B) & & & & \mathcal{C}_1(A,B)\times\mathbf{1} \\
\downarrow{\scriptstyle \mathrm{id}_B\times\mathcal{C}_1(A,B)} & \searrow^{\lambda} & & \swarrow_{\rho} & \downarrow{\scriptstyle \mathcal{C}_1(A,B)\times\mathrm{id}_A} \\
\mathcal{C}_1(B,B)\times\mathcal{C}_1(A,B) & \xrightarrow[\circ]{} & \mathcal{C}_1(A,B) & \xleftarrow[\circ]{} & \mathcal{C}_1(A,B)\times\mathcal{C}_1(A,A)
\end{array} \tag{1.3}$$

commute.

Exemple 1.13

- On peut étendre la catégorie des catégories **Cat** a une 2-catégorie $\mathbf{Cat}_2$ en considérant les transformations naturelles comme des 2-cellules entre les foncteurs.
- Les catégories monoïdales, foncteurs relâchés et transformations monoïdales forment une 2-catégorie **MonCat** . Si on considère plutôt les foncteurs forts, on la note $\mathbf{MonCat_{fort}}$. Si on considère les foncteurs stricts, on la note $\mathbf{MonCat_{str}}$.
- Les catégories monoïdales symétriques, foncteurs relâchés symétriques et transformations monoïdales forment une 2-catégorie **SMonCat** . Si on considère plutôt les foncteurs forts, on la note $\mathbf{SMonCat_{fort}}$. Si on considère plutôt les foncteurs stricts, on la note $\mathbf{SMonCat_{str}}$.

On peut déduire de la fonctorialité de la composition quelques lois de base qui permettent de ne pas se préoccuper de l'ordre de composition des deux cellules.

Propriété 1.14 (Lois d'échange et de Godement) Dans toute 2-catégorie $\mathcal{C}$, la *loi d'échange* entre les compositions horizontale et verticale est vérifiée : pour toutes 2-cellules

$$\begin{array}{c} \alpha : f \Rightarrow f' : A \to B \\ \alpha' : f' \Rightarrow f'' : A \to B \end{array} \quad \text{et} \quad \begin{array}{c} \beta : g \Rightarrow g' : B \to C \\ \beta' : g' \Rightarrow g'' : B \to C \end{array}$$

on a

$$(\beta' \bullet \beta) \circ (\alpha' \bullet \alpha) \;=\; (\beta' \circ \alpha') \bullet (\beta \circ \alpha)$$

et pour toutes 1-cellules $f : A \to B$ et $g : B \to C$, on a

$$\mathrm{id}_{g\circ f} \;=\; \mathrm{id}_g \circ \mathrm{id}_f.$$

De la même manière, dans toute 2-catégorie $\mathcal{C}$, la *loi de Godement* est vérifiée : pour toute paire de 2-cellules

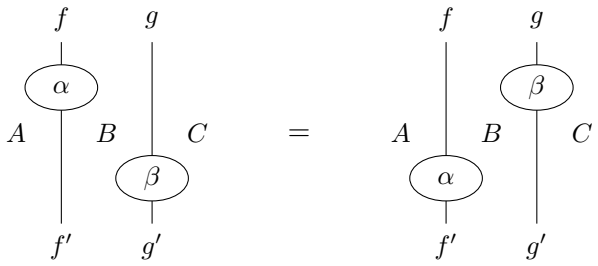
$$\alpha : f \Rightarrow f' : A \to B \quad \text{et} \quad \beta : g \Rightarrow g' : B \to C$$

on a

$$(\beta \circ f') \bullet (g \circ \alpha) \;=\; (g' \circ \alpha) \bullet (\beta \circ f).$$

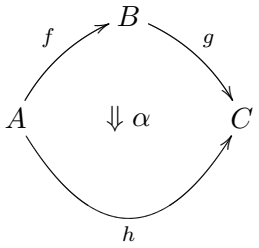
1.2.1 Diagrammes de cordes

Les lois d'échange et de Godement permettent de voir les diagrammes 2-catégoriques de façon géométrique via *des diagrammes de cordes*, introduits par Roger Penrose dans les années 70 [Pen71, PR84]. Ces diagrammes sont basés sur la dualité de Poincaré et permettent de remplacer la notion d'égalité modulo les lois 2-catégoriques par la notion d'égalité modulo déformation topologique. Par exemple, l'égalité de Godement décrite à la Propriété 1.14 peut être représentée par

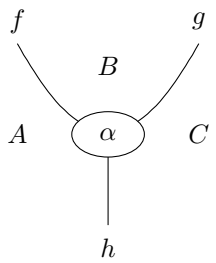


Ainsi, Joyal et Street [JS91] ont montré que deux 2-cellules sont égales modulo les lois 2-catégoriques si et seulement si leurs représentations par un diagramme de cordes sont égales à déformation près.

Nous donnons maintenant un algorithme plus précis de transformation d'une 2-cellule en un diagramme de cordes. Supposons donnée une 2-catégorie $\mathcal{C}$. Soient A, B et C des 0-cellules et $f : A \to B$, $g : B \to C$ et $h : A \to C$ des 1-cellules de cette 2-catégorie. Une 2-cellule $\alpha : g \circ f \Rightarrow h$ est habituellement représentée par



De ce diagramme de dimension 2, on peut déduire un diagramme de cordes en suivant le principe de dualité de Poincaré qui nous amène à transformer éléments de dimension m de ce diagramme en éléments de dimension $2 - m$. Ainsi, les 0-cellules A, B et C deviennent des régions du plan, les 1-cellules f, g et h deviennent des fils et la 2-cellule α devient un point. On obtient alors le diagramme

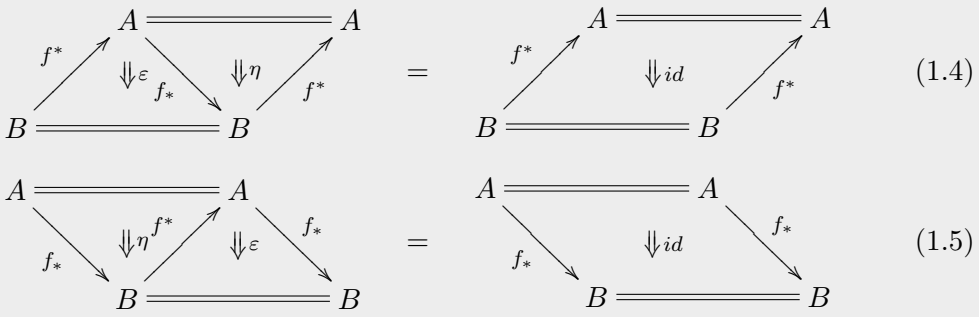


La composition horizontale de deux diagrammes est obtenue en plaçant ces deux diagrammes « côte à côte » et la composition verticale est obtenue en plaçant les deux diagrammes « l'un au-dessus de l'autre » puis en recollant les fils d'entrée et de sortie. La cellule identité est représentée par un simple fil.

1.2.2 Adjonctions dans une 2-catégorie

Définition 1.15 (Adjonction)
Une *adjonction* $f_* \dashv f^* : A \to B$ entre deux flèches $f_* : A \to B$ et $f^* : B \to A$ d'une 2-catégorie $\mathcal{C}$ est la donnée :
- d'une 2-cellule $\eta : \mathrm{id}_A \to f^* \circ f_*$ appelée *unité* ;,
- d'une 2-cellule $\varepsilon : f_* \circ f^* \to \mathrm{id}_B$ appelée *counité* ;

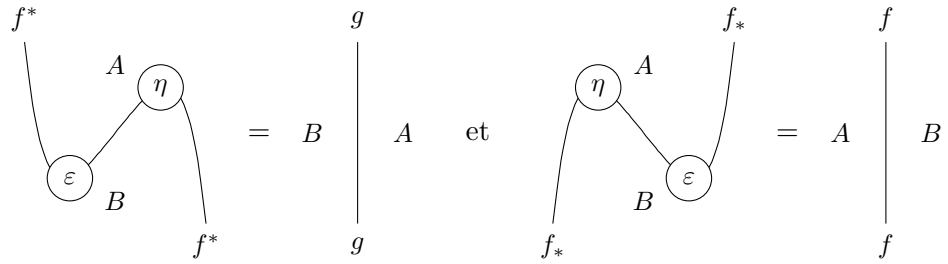
satisfaisant les égalités triangulaires (aussi appelées « identités de zigzag ») :



(1.4)

(1.5)

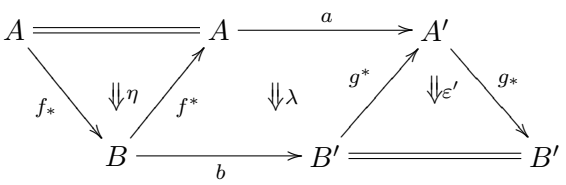
Graphiquement, les égalités triangulaires sont représentées par les diagrammes de cordes suivant



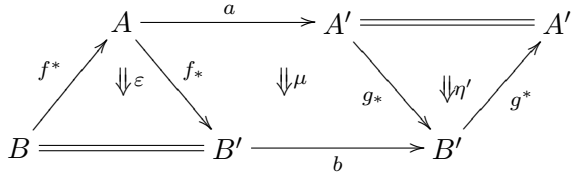
Exemple 1.16
- Une adjonction dans la 2-catégorie $\mathbf{Cat}_2$ redonne la définition habituelle 1.8 de foncteurs adjoints.
- Une adjonction dans la 2-catégorie **MonCat** est appelé *adjonction monoïdale*.
- Considérons la 2-catégorie **poSet** des ensembles partiellement ordonnés et fonctions croissantes. Les 2-cellules sont données par l'ordre point-à-point entre deux fonctions croissantes. Une adjonction dans la 2-catégorie **poSet** est plus connue sous le nom de *correspondance de Galois*.

Proposition 1.17 ([KS72]) Soit $\eta, \varepsilon : f_* \dashv f^* : A \to B$, $\eta', \varepsilon' : g_* \dashv g^* : A' \to B'$, $a : A \to A'$ et $b : B \to B'$. Il existe un bijection entre les 2-cellules $\lambda : af^* \Rightarrow g^*b$ et les 2-cellules $\mu : g_*a \Rightarrow bf_*$ où

μ est la composée



λ est la composée



De plus, la bijection respecte la composition et les identités (à la fois horizontales et verticales).

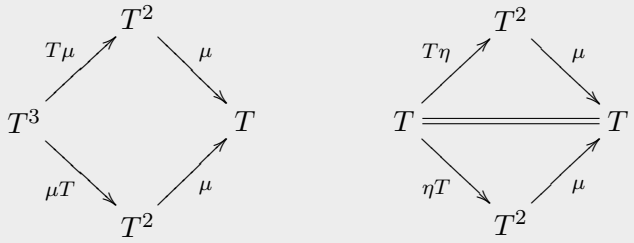
En utilisant la Proposition 1.17 avec $\mathrm{id}_C \dashv \mathrm{id}_C : C \to C$ pour la deuxième adjonction, on obtient le corollaire suivant.

Corollaire 1.18 Soit $\eta, \varepsilon : f_* \dashv f^* : A \to B$, $a : A \to C$ et $b : B \to C$. Il existe un bijection entre les 2-cellules $\lambda : af^* \Rightarrow b$ et les 2-cellules $\mu : a \Rightarrow bf_*$.

1.2.3 Monades dans une 2-catégorie

Définition 1.19 (Monade)
Une *monade* (T, μ, η) sur une 2-catégorie $\mathcal{C}$ est la donnée :
- d'une 1-cellule $T : A \to A$;
- d'une 2-cellule $\mu : T^2 \to T$ appelée *multiplication* de la monade ;
- d'une 2-cellule $\eta : 1 \to T$ appelée *unité* de la monade ;

satisfaisant les deux digrammes commutatifs suivants :



On dit que la monade est *idempotente* lorsqu'elle vérifie en plus

$$\eta_T \circ \mu = T\eta \circ \mu = \mathrm{id}$$

Exemple 1.20
- Une monade dans la 2-catégorie $\mathbf{Cat}_2$ redonne la définition usuelle de monade sur une catégorie.
- Une monade dans la 2-catégorie **MonCat** est appelé *monade monoïdale*.

1.2.4 Algèbres et constructions libres

Adjonction et monade sont deux notions intimement liées.

Propriété 1.21 Toute adjonction $F \dashv G : \mathcal{C} \to \mathcal{D}$ donne lieu à une monade $G \circ F$.

Nous allons maintenant rappeler que toute monade provient d'une adjonction et qu'en général celle-ci n'est pas unique. On suppose donnée une monade T dans une 2-catégorie $\mathcal{C}$.

Définition 1.22 (*T*-algèbre)
Une *T-algèbre* (A, a) est la donnée d'un objet A – appelé *objet sous-jacent de l'algèbre* – et d'un morphisme $a : TA \to A$ – appelé *morphisme de structure* – tels que les diagrammes

$$\begin{array}{ccc} T^2A & \xrightarrow{Ta} & TA \\ {\scriptstyle\mu}\downarrow & & \downarrow{\scriptstyle a} \\ TA & \xrightarrow[a]{} & A \end{array} \qquad \text{et} \qquad \begin{array}{ccc} A & \xrightarrow{\eta} & TA \\ & \searrow & \downarrow{\scriptstyle a} \\ & & A \end{array}$$

commutent. Un *morphisme de T-algèbres* $f : (A, a) \to (B, b)$ est un morphisme de $f : A \to B$ de $\mathcal{C}$ tel que le diagramme

$$\begin{array}{ccc} TA & \xrightarrow{Tf} & TB \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle b} \\ A & \xrightarrow[f]{} & B \end{array}$$

commute.

Proposition 1.23 (catégorie d'Eilenberg-Moore) Les T-algèbres et les morphismes de T-algèbres définissent une catégorie $\mathcal{C}^T$ appelé *catégorie d'Eilenberg-Moore.*

Toute T-algèbre (A, a) peut être vue comme un objet $A = U^T(A, a)$ de $\mathcal{C}$ dont on a oublié la structure. Réciproquement, tout objet A induit une algèbre libre

$$F^T(A) = (TA, \mu_A)$$

Ces constructions se relèvent aisément en deux foncteurs donnant lieu à l'adjonction

$$F^T \dashv U^T : \mathcal{C} \to \mathcal{C}^T$$

Définition 1.24 (catégorie de Kleisli)
La catégorie de Kleisli $\mathcal{C}_T$ d'une monade T a pour objets ceux de $\mathcal{C}$ et pour morphismes de A vers B, les morphismes $f : A \to TB$. L'identité est donnée par l'unité de la monade, $\mathrm{id}_A = \eta_A$, et la composition de deux morphismes $f : A \to TB$ et $g : B \to TC$ est donnée par le morphisme

$$A \xrightarrow{f} TB \xrightarrow{Tg} T^2C \xrightarrow{\mu_C} TC$$

La catégorie $\mathcal{C}$ s'envoie dans la catégorie de Kleisli $\mathcal{C}_T$ par le foncteur :

$$F_T A = A \quad \text{et} \quad F_T(A \xrightarrow{f} B) = A \xrightarrow{f} B \xrightarrow{\eta_B} TB.$$

Réciproquement, $\mathcal{C}_T$ s'envoie dans $\mathcal{C}$ par le foncteur :

$$G_T A = TA \quad \text{et} \quad F_T(A \xrightarrow{f} TB) = TA \xrightarrow{Tf} T^2B \xrightarrow{\mu_B} TB.$$

Ces construction se relève aisément en deux foncteurs donnant lieu à l'adjonction $F_T \dashv U_T : \mathcal{C} \to \mathcal{C}_T$

On voit donc que toute adjonction définit une monade ; et réciproquement, que toute monade définit les deux adjonctions ci-dessus (qui peuvent être identiques). Ces deux adjonctions tiennent une position particulière parmi toutes les adjonctions qui définissent la monade T. En effet, on sait construire une catégorie dont les objets sont les adjonctions de monade T. Il apparaît alors que l'adjonction de la catégorie de Kleisli est initiale dans cette catégorie tandis que l'adjonction de la catégorie d'Eilenberg-Moore y est finale.

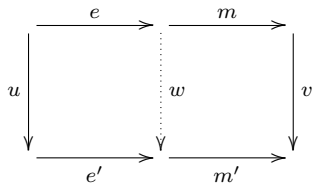
Définition 1.25 (adjonction monadique)
On dit qu'une adjonction $F \dashv G : \mathcal{C} \to \mathcal{D}$ est *monadique* lorsque $\mathcal{D}$ est équivalente à la catégorie d'Eilenberg-Moore sur la monade $T = G \circ F$ induite par l'adjonction. Dans ce cas, on dit que le foncteur F construit la T-algèbre libre sur les objets de $\mathcal{C}$.

Remarquons que les adjonctions ne sont pas toutes monadiques. Le foncteur de **Ens** vers **SCat** qui envoie un ensemble S sur la catégorie discrète dont les objets sont ceux de S et les flèches sont uniquement les identités admet un adjoint à droite. Celui-ci envoie une catégorie petite $\mathcal{C}$ sur son ensemble d'objets $\mathcal{C}_0$, en oubliant toutes les flèches. La comonade induite sur **SCat** est le foncteur qui à $\mathcal{C}$ associe sa catégorie discrète $\underline{\mathcal{C}}$ et la monade induite sur **Ens** est l'identité. La catégorie d'algèbres de cette monade est évidement **Ens** elle-même.

1.2.5 Systèmes de factorisation

Rappel 1.26 (système de factorisation)
Un *système de factorisation* $(\mathcal{E}, \mathcal{M})$ au sens de Peter Freyd et Max Kelly [FK72] pour une catégorie $\mathcal{C}$ est la donnée de deux classes de morphismes $\mathcal{E}$ (pour épi) et $\mathcal{M}$ (pour mono) telles que

- $\mathcal{E}$ et $\mathcal{M}$ contiennent tous les isomorphismes de $\mathcal{C}$ et sont closes par composition ;
- tout morphisme f de $\mathcal{C}$ peut être factorisé comme $f = m \circ e$ avec $e \in \mathcal{E}$ et $m \in \mathcal{M}$;
- la factorisation est fonctorielle : si u et v sont deux morphismes tels que $v \circ m \circ e = m' \circ e' \circ u$ pour des morphismes $e, e' \in \mathcal{E}$ et $m, m' \in \mathcal{M}$, alors il existe un unique morphisme w tel que le diagramme



commute.

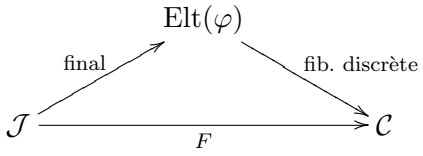
Exemple 1.27
Nous donnons ici les exemples les plus classiques de systèmes de factorisation

Catégorie	$\mathcal{E}$	$\mathcal{M}$
Ens	fonctions surjectives	fonctions injectives
Cat	foncteurs bijectifs sur les objets	foncteurs pleins et fidèles
MultiCat	morphismes bijectifs sur les objets	morphismes pleins et fidèles
Cat	foncteurs finaux	fibrations discrètes

Le lecteur intéressé par l'exemple sur la catégorie des multicatégories **MultiCat** trouvera plus de détails à la Section 5.2.1. Le dernier exemple, un peu plus subtil, a été donné par Ross Street et Robert Walters dans [SW73] – Street et Walters ont originellement présenté le cas dual des foncteurs initiaux et cofibrations discrètes. Il vient du fait que l'on peut associer un préfaisceau φ sur une catégorie $\mathcal{C}$ à tout foncteur F d'une catégorie $\mathcal{J}$ vers $\mathcal{C}$ (i.e. un diagramme sur $\mathcal{J}$ dans $\mathcal{C}$). Le préfaisceau φ est obtenu en calculant la colimite dans **Ens** du diagramme

$$\begin{array}{rccc} \mathcal{C}(-,F) & : & \mathcal{J} & \to & \mathcal{C} \\ & & j & \mapsto & \mathcal{C}(-,Fj) \end{array}$$

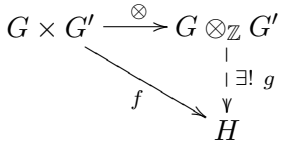
La factorisation est alors obtenue en « injectant » $\mathcal{J}$ dans la catégorie $\mathrm{Elt}(\varphi)$ des éléments de φ puis en « projetant » dans la catégorie $\mathcal{C}$, comme l'indique le diagramme



1.3 Catégories monoïdales

En informatique, la théorie des catégories sert essentiellement à pouvoir abstraire une structure qui apparaît commune à de nombreux objets d'étude. Nous allons ici présenter les structures se dégageant autour de la notion de monoïdalité. C'est en effet une notion cruciale dès lors que l'on peut combiner des objets dans une catégorie. Plus particulièrement pour nous, c'est un moyen d'interpréter en termes abstraits le connecteur $\otimes$ de la logique tensorielle.

Revenons encore une fois sur notre exemple mais restreignons nous à la catégorie des groupes abéliens **Ab** . Il est bien connu que l'on peut définir deux façons de « multiplier » deux groupes abéliens G et G'. La première consiste à prendre simplement le produit des ensembles sous-jacents et à définir la loi interne composante à composante. Ceci définit le produit $G \times G'$ des deux groupes abéliens. Il existe aussi une autre combinaison venant de la théorie des modules. En tant que $\mathbb{Z}$-modules, les groupes abéliens héritent de la multiplication $G \otimes_{\mathbb{Z}} G'$, défini comme étant l'unique groupe abélien (à isomorphisme près) pour lequel il existe une application bilinéaire $\otimes$ de $G \times G'$ dans $G \otimes_{\mathbb{Z}} G'$ qui factorise de manière unique toute application bilinéaire f de $G \times G'$ dans un groupe abélien H en un homomorphisme de groupe g de $G \otimes_{\mathbb{Z}} G'$ vers H.



Cette façon de combiner deux groupes est beaucoup plus subtile mais elle vérifie les mêmes propriétés algébriques que sa prédécessrice ; propriétés encapsulées dans la définition suivante.

Définition 1.28 (Catégorie monoïdale)
Une *catégorie monoïdale* $(\mathcal{C}, \otimes, I, \alpha, \lambda, \rho)$ est une catégorie $\mathcal{C}$ munie :

– d'un foncteur

$$\otimes \quad : \quad \mathcal{C} \times \mathcal{C} \to \mathcal{C}$$

appelé *produit tensoriel* ;
– d'un objet I de C appelé *unité* du produit tensoriel ;
– d'une transformation naturelle inversible α du foncteur d'association à gauche $(- \otimes -) \otimes -$ vers le foncteur d'association à droite $- \otimes (- \otimes -)$ de composante

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C);$$

– de deux transformations naturelles inversibles λ et ρ, de composantes

$$\lambda_A : I \otimes A \to A \quad \text{et} \quad \rho_A : A \otimes I \to A;$$

tels que les diagrammes suivants commutent pour tous objets A, B, C et D :

$$\begin{array}{ccccc}
((A \otimes B) \otimes C) \otimes D & \xrightarrow{\alpha_{A,B,C} \otimes D} & (A \otimes (B \otimes C)) \otimes D & \xrightarrow{\alpha_{A,B \otimes C,D}} & A \otimes ((B \otimes C) \otimes D) \\
\downarrow{\scriptstyle \alpha_{A \otimes B,C,D}} & & & & \downarrow{\scriptstyle A \otimes \alpha_{B,C,D}} \\
(A \otimes B) \otimes (C \otimes D) & & \xrightarrow[\alpha_{A,B,C \otimes D}]{} & & A \otimes (B \otimes (C \otimes D))
\end{array}$$

$$\begin{array}{ccc}
(A \otimes I) \otimes B & \xrightarrow{\alpha_{A,I,B}} & A \otimes (I \otimes B) \\
& {\scriptstyle \rho_A \otimes B} \searrow \quad \swarrow {\scriptstyle A \otimes \lambda_B} & \\
& A \otimes B &
\end{array}$$

Une catégorie monoïdale est *stricte* lorsque les trois transformations naturelles α, λ et ρ sont des identités.

Toute catégorie cartésienne donne directement lieu à une catégorie monoïdale avec pour produit tensoriel de A et B le produit $A \times B$ et pour élément neutre l'objet terminal 1.

Exemple 1.29

– La catégorie **Ens** munie du produit cartésien est monoïdale, elle admet l'ensemble singleton comme élément neutre.
– La catégorie **Rel** munie du produit cartésien est monoïdale, elle admet l'ensemble singleton comme élément neutre.
– La catégorie **Ab** munie du produit de $\mathbb{Z}$-modules $\otimes_{\mathbb{Z}}$ est monoïdale, elle admet $\mathbb{Z}$ comme élément neutre.
– Le produit synchrone $G \times G'$ de deux graphes G et G' a pour sommets les éléments de $V_G \times V_{G'}$ et pour arêtes de (v, w) vers (v', w') les couples d'arêtes $(f : v \to v', g : w \to w')$. La catégorie **Grph** munie du produit *synchrone* est monoïdale, elle admet comme élément neutre le graphe réduit à un sommet et une arête. C'est la produit cartésien dans **Grph**.
– Le produit asynchrone $G \otimes G'$ de deux graphes G et G' a pour sommets l'ensemble $V_G \times V_{G'}$. Les arêtes de (v, w) vers (v', w') sont données par

$$\begin{cases} \text{les arêtes de } v \text{ vers } v' \text{ si } w = w', \text{ ou bien} \\ \text{les arêtes de } w \text{ vers } w' \text{ si } v = v' \end{cases}$$

La catégorie **Grph** munie du produit *asynchrone* est monoïdale, elle admet comme élément neutre le graphe réduit à un sommet et aucune arête.
– La catégorie des $[\mathcal{C},\mathcal{C}]$ des endofoncteurs sur $\mathcal{C}$ et transformations naturelles munie de la composition de foncteurs est monoïdale, elle admet le foncteur identité comme élément neutre.

La plupart des catégories monoïdales qui font partie du quotidien du mathématicien ont naturellement une structure de symétrie.

Définition 1.30 (Catégorie monoïdale symétrique)
Une catégorie monoïdale est dite *symétrique* lorsqu'elle est munie d'un isomorphisme naturel

$$\gamma_{A,B} \quad : \quad A \otimes B \longrightarrow B \otimes A$$

tel que les diagrammes suivants commutent

$$\begin{array}{ccc} A\otimes B & \xrightarrow{\gamma_{A,B}} & B\otimes A \\ & {\scriptstyle \mathrm{id}_{A\otimes B}}\searrow \quad \downarrow{\scriptstyle \gamma_{B,A}} & \\ & A\otimes B & \end{array} \qquad (1.6)$$

$$\begin{array}{ccc} A\otimes I & \xrightarrow{\gamma_{A,I}} & I\otimes A \\ & {\scriptstyle \rho_A}\searrow \quad \swarrow{\scriptstyle \lambda_A} & \\ & A & \end{array} \qquad (1.7)$$

$$\begin{array}{ccccc} & & A\otimes(B\otimes C) & \xrightarrow{\gamma_{A,B\otimes C}} & (B\otimes C)\otimes A \\ & {\scriptstyle \alpha_{A,B,C}}\nearrow & & & \searrow{\scriptstyle \alpha_{B,C,A}} \\ (A\otimes B)\otimes C & & & & B\otimes(C\otimes A) \\ & {\scriptstyle \gamma_{A,B}\otimes C}\searrow & & & \nearrow{\scriptstyle B\otimes\gamma_{A,C}} \\ & & (B\otimes A)\otimes C & \xrightarrow[\alpha_{B,A,C}]{} & B\otimes(A\otimes C) \end{array} \qquad (1.8)$$

On peut montrer que le Diagramme 1.7 est superflu. Si on enlève le Diagramme 1.6 qui indique que $\gamma_{B,A}$ est l'inverse de $\gamma_{A,B}$, on obtient presque la définition d'une *catégorie monoïdale tressée* (la définition nécessite un deuxième diagramme similaire à 1.8 mais faisant intervenir α^{-1}).

Exemple 1.31
Toutes les catégories monoïdales citées en exemple sont symétriques excepté la catégorie des foncteurs et transformations naturelles. En effet, l'existence d'une symétrie dans $[\mathcal{C},\mathcal{C}]$ impliquerait la commutativité de la composition des fonctions ensemblistes.

Comme on peut le voir dans les exemples ci-dessus, une structure monoïdale sur une catégorie n'est pas unique. Il est donc important de pouvoir relier ces structures entre elles via des foncteurs vérifiant des propriétés de préservation par rapport au tenseur.

Définition 1.32 (Foncteur monoïdal)
Un *foncteur monoïdal relâché* entre deux catégories monoïdales $(\mathcal{C},\otimes,I,\alpha,\lambda,\rho)$ et $(\mathcal{D},\bullet,u,\alpha',\lambda',\rho')$ est la donnée d'un foncteur $F : \mathcal{C} \to \mathcal{D}$, d'une transformation naturelle ϕ de composantes $\phi_{A,B} : FA \bullet FB \to F(A\otimes B)$ et d'un morphisme $\phi_I : u \to FI$ tels que les diagrammes

$$\begin{array}{ccc}
(FA \bullet FB) \bullet FC & \xrightarrow{\alpha'_{A,B,C}} & FA \bullet (FB \bullet FC) \\
\downarrow{\scriptstyle \phi_{A,B} \bullet FC} & & \downarrow{\scriptstyle FA \bullet \phi_{B,C}} \\
F(A \otimes B) \bullet FC & & FB \bullet F(B \otimes C) \\
\downarrow{\scriptstyle \phi_{A\otimes B,C}} & & \downarrow{\scriptstyle \phi_{A,B\otimes C}} \\
F((A \otimes B) \otimes C) & \xrightarrow[F\alpha_{A,B,C}]{} & F(A \otimes (B \otimes C))
\end{array}$$

$$\begin{array}{ccc}
u \bullet FA & \xrightarrow{\lambda'_{FA}} & FA \\
\downarrow{\scriptstyle \phi_I \bullet FA} & & \uparrow{\scriptstyle F\lambda_A} \\
FI \bullet FA & \xrightarrow[\phi_{I,A}]{} & F(I \otimes A)
\end{array}
\qquad
\begin{array}{ccc}
FA \bullet u & \xrightarrow{\rho'_{FA}} & FA \\
\downarrow{\scriptstyle FA \bullet \phi_I} & & \uparrow{\scriptstyle F\rho_A} \\
FA \bullet FI & \xrightarrow[\phi_{A,I}]{} & F(A \otimes I)
\end{array}$$

commutent dans $\mathcal{D}$ pour tous objets A, B et C.

Un *foncteur monoïdal fort* (resp. *strict*) est un foncteur monoïdal relâché dans lequel $\phi_{A,B}$ et ϕ_I sont des isomorphismes (resp. des identités) pour tous objets A et B de $\mathcal{C}$. On définit de la même manière les foncteurs monoïdaux *corelâchés* en renversant toutes les flèches structurelles.

Lorsque la catégorie $\mathcal{C}$ est symétrique et que le foncteur préserve la symétrie, c'est-à-dire

$$\begin{array}{ccc}
FA \bullet FB & \xrightarrow{\gamma_{FA,FB}} & FB \bullet FA \\
\downarrow{\scriptstyle \phi_{A,B}} & & \downarrow{\scriptstyle \phi_{B,A}} \\
F(A \otimes B) & \xrightarrow[F\gamma_{A,B}]{} & F(B \otimes A)
\end{array}$$

on dit que le foncteur est *monoïdale symétrique*.

Exemple 1.33
Le foncteur d'oubli U : **Ab** $\to$ **Ens** est monoïdal strict si on munit **Ab** et **Ens** de leur produit respectif. Cela vient du fait (déjà signaler) que ce foncteur est adjoint à droite du foncteur F : **Ens** $\to$ **Ab** qui construit le groupe abélien libre. Plus subtilement, on peut montrer que l'application bilinéaire $G \times G' \to G \otimes_{\mathbb{Z}} G'$ fait de U un foncteur monoïdal relâché si on munit **Ab** du tenseur $\otimes_{\mathbb{Z}}$. Nous verrons un peu plus loin que cette propriété se déduit du fait que son adjoint à gauche, le foncteur F, est monoïdale fort de **Ens** muni du produit, vers **Ab** muni du tenseur.

À ce stade de la lecture, le lecteur novice aura déjà pu s'apercevoir de l'importance des diagrammes de cohérence en théorie des catégories. Mais la question qu'il se pose alors est simple : Pourquoi doit-on imposer ces diagrammes et pourquoi ceux-ci plutôt que d'autres ?

La réponse, pour les catégories monoïdales, réside dans un théorème de cohérence. En effet, les diagrammes de cohérence doivent nous assurer que n'importe quelle application successive de α, λ et ρ menant d'un objet A à un objet B dans une catégorie monoïdale sont identiques. On résume souvent ce principe par la maxime :

Tous les diagrammes commutent.

Pour pouvoir établir formellement ce théorème, il nous faut une notion d'équivalence de catégories monoïdales et donc de transformations naturelles monoïdales. En effet, la version abstraite du théorème de cohérence se formule en disant que toute catégorie monoïdale $\mathcal{C}$ est équivalente à une catégorie monoïdale stricte $\mathcal{D}$. Comme tous les diagrammes formés de morphismes canoniques (associativité, etc.) d'une catégorie monoïdale stricte commutent trivialement (ce ne sont que des identités), on en déduit que les diagrammes de $\mathcal{C}$ commutent eux aussi.

Définition 1.34 (Transformation naturelle monoïdale)
Une transformation naturelle monoïdale $\theta : (F, \phi) \to (G, \psi)$ entre deux foncteurs monoïdaux relâchés $(F, \phi), (G, \psi) : (\mathcal{C}, \otimes, I) \to (\mathcal{D}, \bullet, u)$ est la donnée d'une transformation naturelle $\theta : F \to G$ entre les deux foncteurs sous-jacents pour laquelle les diagrammes

$$\begin{array}{ccc} FA \bullet FB & \xrightarrow{\theta_A \bullet \theta_B} & GA \bullet GB \\ \downarrow{\scriptstyle \phi_{A,B}} & & \downarrow{\scriptstyle \psi_{A,B}} \\ F(A \otimes B) & \xrightarrow[\theta_{A\otimes B}]{} & G(A \otimes B) \end{array} \qquad \begin{array}{ccc} & u & \\ {\scriptstyle \phi_u}\swarrow & & \searrow{\scriptstyle \psi_u} \\ FI & \xrightarrow[\theta_I]{} & GI \end{array}$$

commutent dans $\mathcal{D}$ pour tous objets A et B de $\mathcal{C}$.

Une adjonction entre deux foncteurs relâchés pour laquelle l'unité et la counité sont monoïdales est appelée *adjonction monoïdale*.

Une équivalence de catégories pour laquelle les foncteurs sont forts et l'unité et la counité sont monoïdales est appelé *équivalence de catégories*. On peut maintenant énoncer le théorème de cohérence.

Propriété 1.35 (Cohérence des catégories monoïdales) Toute catégorie monoïdale est équivalente, via une équivalence de catégories monoïdales, à une catégorie monoïdale stricte. Réciproquement, toute catégorie équivalente à une catégorie monoïdale stricte est monoïdale.

En conclusion, plusieurs présentations des lois de cohérence d'une catégorie monoïdale sont possibles. Certaines sont minimales, d'autres pratiques à utiliser mais l'important est qu'elles satisfassent toutes le théorème de cohérence. Par exemple, dans la définition des catégories monoïdales symétriques, le Diagramme (1.7) peut être déduit du Diagramme (1.8). Pourtant, comme cette propriété est loin d'être évidente, nous pensons qu'il est préférable de laisser cette axiome superflu.

1.3.1 Catégories monoïdales symétriques fermées

On a vu comment, en théorie des catégories, les morphismes entre des objets peuvent servir à étudier les objets eux-mêmes. Réciproquement, nous voudrions une notion qui permet de voir les morphismes comme des objets de la catégorie. Ce processus d'« internalisation des morphismes » est très important en informatique car il permet de décrire l'abstraction du λ-calcul et plus généralement des langages fonctionnels.

Définition 1.36 (Catégorie monoïdale symétrique fermée)
Une *catégorie monoïdale symétrique fermée* est une catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I)$ pour laquelle le produit tensoriel $A \otimes -$ à un adjoint à droite $A \multimap -$ pour tout objet A. Plus simplement, pour tous objets A, B et C de $\mathcal{C}$, on a un isomorphisme

$$\mathcal{C}_1(A \otimes B, C) \;\cong\; \mathcal{C}_1(B, A \multimap C)$$

naturel en B et C.

La notion de fermeture existe aussi dans un cadre non symétrique mais nécessite alors la donnée d'une fermeture à gauche $A \multimap -$ et d'une fermeture à droite $- \multimap A$.

Exemple 1.37
- La catégorie **Ens** munie du produit cartésien est fermée avec pour fermeture $A \multimap B$, l'ensemble des fonctions de A vers B.
- La catégorie **Rel** munie du produit cartésien est fermée avec pour fermeture $A \multimap B$, l'ensemble des relations sur $A \times B$.
- La catégorie **Ab** munie du produit de $\mathbb{Z}$-modules $\otimes_{\mathbb{Z}}$ est fermée avec pour fermeture $G \multimap G'$, le groupe des homomorphismes de G vers G' muni de la loi point-à-point

  $$(f + g)(a) \;\stackrel{\text{def}}{=}\; f(a) + g(a),$$

  l'inverse est aussi défini point-à-point.
- La catégorie **Grph** munie du produit *asynchrone* est fermée avec pour fermeture $G \multimap G'$, le graphe des homomorphismes de graphes de G vers G' ayant pour arêtes les familles η_a vérifiant

  $$f \xrightarrow{\eta}_{G \multimap G'} f' \quad \text{ssi} \quad \forall a \in G \;,\; fa \xrightarrow{\eta_a}_{G'} fa'.$$

  Remarquons au passage que la situation est en tout point similaire à celle de **Cat** munie du produit tensoriel de Gray.

Remarque 1.38
La catégorie **Ab** munie du produit usuel n'est pas fermée. En effet, si elle l'était, le produit commuterait aux colimites et en particulier aux sommes. Or la somme est donnée par le produit dans la catégorie **Ab**. On en déduit l'isomorphisme saugrenu

$$G \times (H \times H') \;\cong\; (G \times H) \times (G \times H')$$

qui n'est évidement pas valide. Cela montre que **Ab** n'est pas cartésienne fermée.

La fonction $\mathcal{C}_1(-,-)$ qui – dans une catégorie $\mathcal{C}$ – associe à deux objets A et B l'ensemble des flèches de A dans B peut être étendue à un foncteur

$$\mathcal{C}_1(-,-) \quad : \quad \mathcal{C}^{\text{op}} \times \mathcal{C} \longrightarrow \mathbf{Ens}$$

Nous donnons maintenant une définition de nature 2-catégorique (au sens où elle s'appuie sur la notion de transformations naturelles dans **Cat**) de la fermeture, basée sur l'action du produit tensorielle $\otimes$ et de la fermeture $\multimap$ sur ce foncteur $\mathcal{C}_1(-,-)$

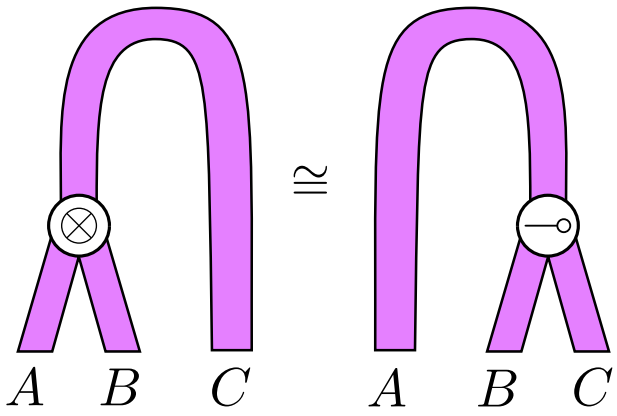


FIG. 1.1 – Dessin de la fermeture dans une catégorie monoïdale symétrique

Définition bis 1.38 (Catégorie monoïdale symétrique fermée)
Une *catégorie monoïdale symétrique fermée* est une catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I)$ pour laquelle il existe un foncteur

$$\multimap \quad : \quad \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \longrightarrow \mathcal{C}$$

et un isomorphisme naturel φ de la forme

$$\begin{array}{ccc} \mathcal{C}^{\mathrm{op}} \times \mathcal{C}^{\mathrm{op}} \times \mathcal{C} & \xrightarrow{\otimes^{op} \times \mathcal{C}} & \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \\ \downarrow{\scriptstyle \mathcal{C}^{\mathrm{op}} \times \multimap} & \cong \varphi & \downarrow{\scriptstyle \mathcal{C}_1(-,-)} \\ \mathcal{C}^{\mathrm{op}} \times \mathcal{C} & \xrightarrow[\mathcal{C}_1(-,-)]{} & Set \end{array}$$

Cette reformulation de la notion de fermeture pour une catégorie monoïdale nous permet de donner une représentation graphique basée sur les diagrammes de cordes. La Figure 1.1 exprime que l'espace d'une catégorie monoïdale symétrique est suffisamment « mou » pour que l'on puisse déplacer un objet du domaine vers le codomaine d'un morphisme. Nous verrons que ce diagramme fait sens dans une certaine bicatégorie de modules catégoriques, ou distributeurs.

On peut maintenant exprimer de manière élégante et concise la notion d'idéal exponentiel qui jouera un rôle important dans notre étude algébrique de la logique tensorielle.

1.3.2 Idéal exponentiel

Dans certaines situations, on ne peut pas définir de fermeture sur toute la catégorie $\mathcal{C}$ mais seulement sur une sous-catégorie $\mathcal{I}$. On a donc besoin d'une définition de clôture faisant intervenir un foncteur $\mathcal{I} \to \mathcal{C}$ identifiant la sous-catégorie de $\mathcal{C}$ acceptant une exponentiation. Cela donne lieu à la définition d'un idéal exponentiel.

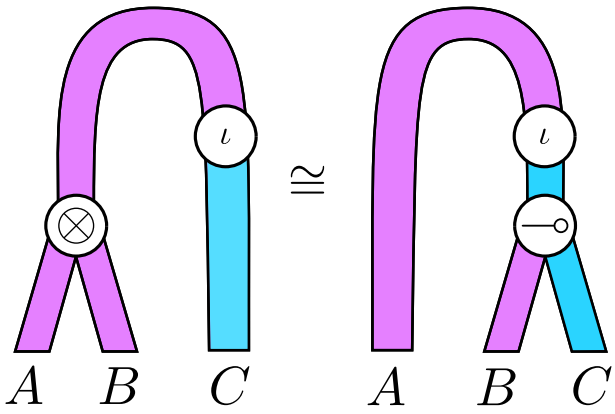


FIG. 1.2 – Dessin d'un idéal exponentiel

Définition 1.39 (idéal exponentiel)
Un *idéal exponentiel* $(\iota, \mathcal{I}, \multimap)$ sur une catégorie monoïdale symétrique $\mathcal{C}$ est constituée d'une paire de foncteurs

$$\iota \quad : \quad \mathcal{I} \to \mathcal{C} \qquad\qquad \multimap \quad : \quad \mathcal{C}^{\mathrm{op}} \times \mathcal{I} \to \mathcal{I}$$

et d'un isomorphisme naturel φ de la forme

$$\begin{array}{ccccc}
\mathcal{C}^{\mathrm{op}} \times \mathcal{C}^{\mathrm{op}} \times \mathcal{I} & & \xrightarrow{\quad \otimes^{op} \times \mathcal{I} \quad} & & \mathcal{C}^{\mathrm{op}} \times \mathcal{I} \\
\downarrow{\scriptstyle \mathcal{C}^{\mathrm{op}} \times \multimap} & & \cong \varphi & & \downarrow{\scriptstyle \mathcal{C}^{\mathrm{op}} \times \iota} \\
\mathcal{C}^{\mathrm{op}} \times \mathcal{I} & \xrightarrow[\mathcal{C}^{\mathrm{op}} \times \iota]{} & \mathcal{C}^{\mathrm{op}} \times \mathcal{C} & \xrightarrow[\mathcal{C}_1(-,-)]{} & Set
\end{array}$$

On peut, comme pour la fermeture, décrire de manière diagrammatique ce processus de déplacement d'objet de la gauche vers la droite de l'espace. La Figure 1.2 montre que cette fois le déplacement nécessite que l'espace de droite soit « gardé » par le foncteur ι.

Toute catégorie monoïdale symétrique fermée donne lieu à un idéal exponentiel avec $\mathcal{I} = \mathcal{C}$ et $\iota = \mathrm{id}$.

1.3.3 Catégories $*$-autonomes

Définition 1.40 (catégorie $*$-autonome)
Une catégorie monoïdale symétrique $\mathcal{C}$ est *$*$-autonome* lorsque elle est munie
- d'un foncteur $(-)^* : \mathcal{C}^{\mathrm{op}} \to \mathcal{C}$ appelé *foncteur de dualisation* ;
- d'un isomorphisme naturel
$$A \cong A^{**}$$
pour tout A ;
- d'une bijection naturel
$$\varphi_{A,B,C} \quad : \quad \mathcal{C}(A \otimes B, C^*) \;\cong\; \mathcal{C}(A, (B \otimes C)^*)$$
pour tous objets A, B et C de $\mathcal{C}$ de telle sorte que le diagramme
$$\begin{array}{ccc} \mathcal{C}(A \otimes (B \otimes C), D^*) & \xrightarrow{\mathcal{C}(\alpha_{A,B,C}, C^*)} & \mathcal{C}((A \otimes B) \otimes C, D^*) \\ & & \downarrow{\scriptstyle \varphi_{A\otimes B,C,D}} \\ \downarrow{\scriptstyle \varphi_{A,B\otimes C,D}} & & \mathcal{C}(A \otimes B, (C \otimes D)^*) \\ & & \downarrow{\scriptstyle \varphi_{A,B,C\otimes D}} \\ \mathcal{C}(A, ((B \otimes C) \otimes D)^*) & \xrightarrow[\mathcal{C}(A, (\alpha^{-1}_{B,C,D})^*)]{} & \mathcal{C}(A, (B \otimes (C \otimes D))^*) \end{array}$$
commute.

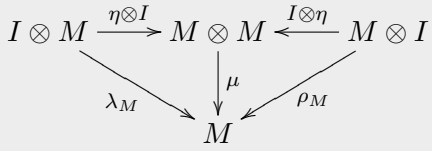
L'objet A^* est appelé *dual* de A.

Une catégorie $*$-autonome définit un idéal exponentiel avec
$$\mathcal{I} \;=\; \mathcal{C}^{\mathrm{op}}, \quad \iota \;=\; (-)^* \quad \text{et} \quad \multimap \;=\; \otimes^{\mathrm{op}}.$$
Comme $(-)^*$ est un isomorphisme naturel, elle donne même lieu à une catégorie monoïdale symétrique fermée en posant
$$A \multimap B = (B \otimes C^*)^*.$$

1.3.4 Monoïdes et comonoïdes

Définition 1.41 (Monoïde)
Un *monoïde* (M, η, μ) dans un catégorie monoïdale $(\mathcal{C}, \otimes, I)$ est un objet M donné avec deux morphismes
- $\mu : M \otimes M \to M$ appelé *multiplication* ;
- $\eta : I \to M$ appelé *unité* ;

de telle sorte que les diagrammes
$$\begin{array}{ccccc} (M \otimes M) \otimes M & \xrightarrow{\alpha} & M \otimes (M \otimes M) & \xrightarrow{M\otimes\mu} & M \otimes M \\ \downarrow{\scriptstyle \mu\otimes M} & & & & \downarrow{\scriptstyle \mu} \\ M \otimes M & & \xrightarrow[\mu]{} & & M \end{array}$$



commutent.

Dualement, un *comonoïde* sur une catégorie monoïdale $\mathcal{C}$ est un monoïde dans la catégorie opposée $\mathcal{C}^{\mathrm{op}}$.

Un *morphisme de monoïdes* f entre deux monoïdes (M, η, μ) et (M', η', μ') est un morphisme vérifiant :

$$f \circ \mu = \mu' \circ (f \otimes f) \quad \text{et} \quad f \circ \eta = \eta'.$$

On définit dualement les *morphismes de comonoïdes*.

Exemple 1.42

- Un monoïde dans la catégorie $(\mathbf{Ens}, \times, 1)$ est un monoïde au sens usuel.
- Un monoïde dans la catégorie $(\mathbf{Ab}, \otimes_{\mathbb{Z}}, \mathbb{Z})$ est la même chose qu'un anneau.
- Un monoïde dans la catégorie $[\mathcal{C}, \mathcal{C}]$ des endofoncteurs sur une catégorie C est une monade (voir Définition 1.19).

1.3.5 Catégories prémonoïdales

Il est bien connu que la catégorie **Cat** possède exactement deux structures monoïdales symétriques fermées [FLK80]. La structure habituelle – où le produit tensoriel est donné par le produit cartésien et où la fermeture est donnée par la catégorie des foncteurs et transformations naturelles – donne lieu à une 2-catégorie cartésienne $\mathbf{Cat}_2$. Il est possible de voir les catégories monoïdales strictes comme des monoïdes dans $\mathbf{Cat}_2$ – et même les catégories monoïdales comme des pseudomonoïdes dans $\mathbf{Cat}_2$.

L'autre catégorie monoïdale symétrique fermée induite par **Cat** est généralement notée $\mathbf{Cat}'$. De manière informelle, le produit tensoriel $\mathcal{C} \otimes \mathcal{D}$ de deux catégories $\mathcal{C}$ et $\mathcal{D}$ est la catégorie ayant pour objets $\mathcal{C}_0 \times \mathcal{D}_0$ et pour morphismes les suites finies alternées de morphismes de $\mathcal{C}$ et $\mathcal{D}$ différents de l'identité dont la suite extraite des morphismes de $\mathcal{C}$ (resp. $\mathcal{D}$) forment un chemin dirigé dans $\mathcal{C}$ (resp. $\mathcal{D}$). La composition est donnée par la concaténation formelle suivie des simplifications induites par les deux catégories $\mathcal{C}$ et $\mathcal{D}$. Ce produit tensoriel peut être défini conceptuellement par la somme amalgamée suivante dans **Cat** (définition tirée de [PR97])

$$\begin{array}{ccc} \underline{\mathcal{C}} \times \underline{\mathcal{D}} & \longrightarrow & \mathcal{C} \times \underline{\mathcal{D}} \\ \downarrow & & \downarrow \\ \underline{\mathcal{C}} \times \mathcal{D} & \longrightarrow & \mathcal{C} \otimes \mathcal{D} \end{array}$$

où $\underline{\mathcal{C}}$ dénote la catégorie discrète sur $\mathcal{C}$. La fermeture $[\mathcal{C}, \mathcal{D}]$ est donnée par la catégorie des foncteurs de $\mathcal{C}$ vers $\mathcal{D}$ et des transformations (pas forcement naturelles) entre ces foncteurs.

John Power et Edmund Robinson [PR97] ont observé qu'un monoïde dans $\mathbf{Cat}'$ est la même chose qu'une catégorie prémonoïdale stricte. Malheureusement, on n'obtient pas les catégories prémonoïdales en passant aux pseudomonoïdes. En effet, les isomorphismes structurels d'une catégorie prémonoïdale doivent être naturels mais aussi « centraux ».

Soit $\parr$ un foncteur de $\mathcal{C} \otimes \mathcal{C}$ vers $\mathcal{C}$. Un morphisme $f : A \to A'$ de la catégorie $\mathcal{C}$ est dit *central* si pour tout morphisme $g : B \to B'$, les deux composées $(f \parr B') \circ (A \parr g)$ et $(A' \parr g) \circ (f \parr B)$ sont égales, et les deux composées $(B' \parr f) \circ (g \parr A)$ et $(g \parr A') \circ (B \parr f)$ sont égales. Dans ce cas, on peut utiliser les notations $f \parr g$ et $g \parr f$ qui ne sont pas ambiguës.

Définition 1.43 (naturelle en A vis-à-vis des morphismes centraux)
On dit qu'une transformation $\eta_A : FA \to GB$ entre deux foncteurs F et G est *naturelle en A vis-à-vis des morphismes centraux* lorsque pour tout morphisme central $f : A \to B$, le diagramme

$$\begin{array}{ccc} FA & \xrightarrow{Ff} & FB \\ {\scriptstyle \eta_A}\downarrow & & \downarrow{\scriptstyle \eta_B} \\ GA & \xrightarrow[Gf]{} & GB \end{array}$$

commute.

Cette notion de centralité permet maintenant de définir les catégories prémonoïdales.

Définition 1.44 (catégorie prémonoïdale)
Une catégorie *prémonoïdale* $(\mathcal{C}, \parr, \bot, \alpha, \lambda, \rho)$ est une catégorie munie
- d'un foncteur $\parr : \mathcal{C} \otimes \mathcal{C} \to \mathcal{C}$ appelé *produit tensoriel* ;
- d'un objet $\bot$ appelé *unité* ;
- d'une transformation naturelle inversible α du foncteur d'association à gauche $(- \parr -) \parr -$ vers le foncteur d'association à droite $- \parr (- \parr -)$ de composante

$$\alpha_{A,B,C} : (A \parr B) \parr C \to A \parr (B \parr C)$$

  présumée centrale pour tous objets A, B et C de $\mathcal{C}$;
- de deux transformations naturelles inversibles λ et ρ, de composantes

$$\lambda_A \ : \ \bot \parr A \to A \quad \text{et} \quad \rho_A \ : \ A \parr \bot \to A$$

  présumées centrales pour tout objet A ;

assujetties au pentagone de MacLane et au triangle unitaire identiques à ceux des catégories monoïdales.

Une *catégorie prémonoïdale symétrique* possède en plus une famille d'isomorphismes naturels et centraux

$$\gamma_{A,B} : A \parr B \ \longrightarrow \ A \parr B,$$

satisfaisant

$$\gamma_{B,A} \circ \gamma_{A,B} = \mathrm{id}_{A \parr B}$$

et les deux diagrammes de compatibilité avec α, λ et ρ identiques à ceux des catégories monoïdales symétriques.

Les morphismes centraux d'une catégorie prémonoïdales $\mathcal{C}$ forment une catégorie monoïdale $\mathcal{Z}(\mathcal{C})$ appelé *centre* de $\mathcal{C}$. En particulier, une catégorie prémonoïdale est monoïdale si et seulement si elle est égale à son centre.

Définition 1.45 (foncteur prémonoïdal)
Un *foncteur prémonoïdal relâché* entre deux catégories prémonoïdales $(\mathcal{C}, \parr, \bot, \alpha, \lambda, \rho)$ et $(\mathcal{D}, \otimes, \bot, \alpha', \lambda', \rho')$ est la donnée d'un foncteur $F : \mathcal{C} \to \mathcal{D}$ qui envoie les morphismes centraux sur des morphismes centraux, d'une transformation naturelle centrale ϕ de composantes $\phi_{A,B} : FA \otimes FB \to F(A \parr B)$ et d'un morphisme central $\phi_I : \bot \to F\bot$ tels que les diagrammes

$$\begin{array}{ccc}
(FA \otimes FB) \otimes FC & \xrightarrow{\alpha'_{A,B,C}} & FA \otimes (FB \otimes FC) \\
\downarrow{\scriptstyle \phi_{A,B} \otimes FC} & & \downarrow{\scriptstyle FA \otimes \phi_{B,C}} \\
F(A \parr B) \otimes FC & & FB \otimes F(B \parr C) \\
\downarrow{\scriptstyle \phi_{A \parr B, C}} & & \downarrow{\scriptstyle \phi_{A, B \parr C}} \\
F((A \parr B) \parr C) & \xrightarrow[F\alpha_{A,B,C}]{} & F(A \parr (B \parr C))
\end{array}$$

$$\begin{array}{ccc}
\bot \otimes FA & \xrightarrow{\lambda'_{FA}} & FA \\
\downarrow{\scriptstyle \phi_I \otimes FA} & & \uparrow{\scriptstyle F\lambda_A} \\
F\bot \otimes FA & \xrightarrow[\phi_{I,A}]{} & F(\bot \parr A)
\end{array}
\qquad
\begin{array}{ccc}
FA \otimes \bot & \xrightarrow{\rho'_{FA}} & FA \\
\downarrow{\scriptstyle FA \otimes \phi_I} & & \uparrow{\scriptstyle F\rho_A} \\
FA \otimes F\bot & \xrightarrow[\phi_{A,I}]{} & F(A \parr I)
\end{array}$$

commutent dans $\mathcal{D}$ pour tous objets A, B et C.

Un *foncteur prémonoïdal fort* (resp. *strict*) est un foncteur prémonoïdal relâché dans lequel $\phi_{A,B}$ et ϕ_I sont des isomorphismes (resp. des identités) pour tous objets A et B de $\mathcal{C}$. On définit de la même manière les foncteurs prémonoïdaux *corelâchés* en renversant toutes les flèches structurelles.

Lorsque la catégorie $\mathcal{C}$ est symétrique et que le foncteur préserve la symétrie, c'est-à-dire

$$\begin{array}{ccc}
FA \otimes FB & \xrightarrow{\gamma_{FA,FB}} & FB \otimes FA \\
\downarrow{\scriptstyle \phi_{A,B}} & & \downarrow{\scriptstyle \phi_{B,A}} \\
F(A \parr B) & \xrightarrow[F\gamma_{A,B}]{} & F(B \parr A)
\end{array}$$

on dit que le foncteur est *prémonoïdale symétrique*.

Remarque 1.46
On peut donner une définition 2-dimensionnelle des catégories prémonoïdales. Une catégorie enrichie sur $\mathbf{Cat}'$ est appelée une *sesqui-catégorie*. Comme $\mathbf{Cat}'$ est monoïdale symétrique fermée, elle est enrichie sur elle-même et forme donc une sesqui-catégorie $\mathbf{Cat}'_2$. $\mathbf{Cat}'_2$ n'est pas une 2-catégorie car la loi d'échange n'est pas toujours satisfaite. Cependant, une transformation entre deux foncteurs est naturelle si et seulement si elle vérifie la loi d'échange avec toutes les compositions à gauche avec des 2-cellules pour lesquelles la loi a un sens.

On définit donc une *2-cellule naturelle* dans une sesqui-catégorie comme étant une 2-cellule qui commute avec toutes les 2-cellules à gauche (au sens de la loi d'échange).

Ensuite, on définit un catégorie prémonoïdale comme un pseudomonoïde dans la sesqui-catégorie $\mathbf{Cat}'_2$ pour lequel toutes les 2-cellules apparaissant dans la structure sont naturelles.

Cela donne une définition purement conceptuelle des catégories prémonoïdales équivalentes à la définition plus pédestre donnée plus haut.

1.3.6 Monades fortes et monades commutatives

Définition 1.47 (Monade forte)
Une *monade* (T, μ, η) dans sur une catégorie monoïdale $(\mathcal{C}, \otimes, I)$ est dite *forte à droite* s'il existe une transformation naturelle

$$t_{A,B} \quad : \quad A \otimes TB \to T(A \otimes B)$$

appelée *force (tensorielle)* telle que les diagrammes

$$\begin{array}{ccccc} (A \otimes B) \otimes TC & \xrightarrow{\alpha_{A,B,TC}} & A \otimes (B \otimes TC) & \xrightarrow{A \otimes t_{B,C}} & A \otimes T(B \otimes C) \\ \downarrow{\scriptstyle t_{A\otimes B,C}} & & & & \downarrow{\scriptstyle t_{A,B\otimes C}} \\ T((A \otimes B) \otimes C) & & \xrightarrow[T\alpha_{A,B,C}]{} & & T(A \otimes (B \otimes C)) \end{array} \tag{1.9}$$

$$\begin{array}{ccccc} A \otimes T^2B & \xrightarrow{t_{A,TB}} & T(A \otimes TB) & \xrightarrow{Tt_{A,B}} & T^2(A \otimes B) \\ \downarrow{\scriptstyle A\otimes\mu_B} & & & & \downarrow{\scriptstyle \mu_{A\otimes B}} \\ A \otimes TB & & \xrightarrow[t_{A,B}]{} & & T(A \otimes B) \end{array} \tag{1.10}$$

$$\begin{array}{ccc} I \otimes TA & \xrightarrow{t_{I,A}} & T(I \otimes A) \\ & {\scriptstyle \lambda_{TA}} \searrow & \downarrow{\scriptstyle T\lambda_A} \\ & & TA \end{array} \tag{1.11}$$

$$\begin{array}{ccc} A \otimes B & \xrightarrow{A\otimes\eta_B} & A \otimes TB \\ & {\scriptstyle \eta_{A\otimes B}} \searrow & \downarrow{\scriptstyle t_{A,B}} \\ & & T(A \otimes B) \end{array} \tag{1.12}$$

commutent. De la même manière, une monade est dite *forte à gauche* si elle est munie d'une *force* (à gauche)

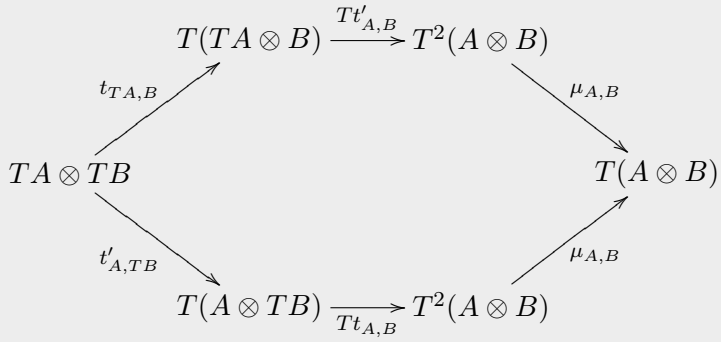
$$t'_{A,B} \quad : \quad TA \otimes B \to T(A \otimes B)$$

satisfaisant les diagrammes idoines. Une monade est dite *forte* lorsqu'elle est forte à gauche et à droite.

Lorsque la catégorie monoïdale $\mathcal{C}$ est symétrique, une monade forte à droite est aussitôt équipée d'une force à gauche qui la rend forte

$$t'_{A,B} = T(\gamma_{B,A}) \circ t_{B,A} \circ \gamma_{A,TB}$$

Définition 1.48 (Monade commutative)
Une monade forte est dite *commutative* lorsque la force à gauche et à droite se marient à travers le diagramme commutatif suivant :



(1.13)

Proposition 1.49 ([Koc72],[GLLN02]) Les monades commutatives sont en bijection avec les monades monoïdales symétriques.

Proposition 1.50 ([PR97] : Corollaire 4.2) Soit $\mathcal{C}$ une catégorie monoïdale symétrique et T une monade sur $\mathcal{C}$. La monade T est forte si et seulement si la catégorie de Kleisli $\mathcal{C}_T$ est prémonoïdale symétrique et le foncteur $F_T : \mathcal{C} \to \mathcal{C}_T$ est prémonoïdal symétrique strict.

Proposition 1.51 ([PR97] : Corollaire 4.3) Soit $\mathcal{C}$ une catégorie monoïdale symétrique et T une monade sur $\mathcal{C}$. La monade T est commutative si et seulement si la catégorie de Kleisli $\mathcal{C}_T$ est monoïdale symétrique et le foncteur $F_T : \mathcal{C} \to \mathcal{C}_T$ est monoïdal symétrique strict.

1.3.7 Dualité entre adjonctions monoïdales

Proposition 1.52 ([Mel08] : section 5.14) Soit $F \dashv G : \mathcal{C} \to \mathcal{D}$ une adjonction entre deux catégories monoïdales. Il y a une bijection entre les structures corelâchées sur l'adjoint à gauche F et les structures relâchées sur l'adjoint à droite G.

Proposition 1.53 ([Mel08] : section 5.15) Soit $F \dashv G : A \to B$ une adjonction entre deux catégories monoïdales. Supposons aussi que F soit relâché. Alors l'adjonction est monoïdales si et seulement si F est fort.

Proposition 1.54 ([Mel08] : section 5.16) Soit $F \dashv G : A \to B$ une adjonction entre deux catégories monoïdales symétriques. Supposons aussi que F soit relâché symétrique. Alors l'adjonction est monoïdale symétrique si et seulement si F est symétrique fort.

1.4 Théories algébriques, PRO et PROP

1.4.1 Théories de Lawvere.

Dans sa thèse [Law63], William Lawvere reformule ainsi la notion habituelle de théorie algébrique (par exemple des monoïdes, des algèbres de Lie, etc.)

Définition 1.55 (théorie algébrique)
Une *théorie algébrique* est une catégorie $\mathbb{L}$ équipée de produits finis, dont les objets sont les entiers naturels : $0, 1, 2, \ldots$ et dans laquelle le produit de k objets $m_1, \ldots, m_k$ est donné par leur somme arithmétique $m_1 + \cdots + m_k$.

L'idée est de représenter chaque opération n-aire d'une théorie algébrique par un morphisme $m \to 1$ dans la catégorie $\mathbb{L}$; et d'encoder la théorie équationnelle de ces opérations n-aires au moyen de la composition de la catégorie $\mathbb{L}$.

Un $\mathbb{L}$-modèle sur une catégorie $\mathcal{C}$ avec produits finis (noté $\times$) est défini comme un foncteur

$$A \;:\; \mathbb{L} \longrightarrow \mathcal{C}$$

préservant les produits, au sens où le morphisme canonique

$$A[m_1 + \cdots + m_k] \longrightarrow A[m_1] \times \cdots \times A[m_k]$$

est un isomorphisme (cf la Définition 1.32 d'un foncteur fort). Cela implique que le foncteur A transporte (à isomorphisme près) l'entier naturel n vers la n-ème puissance de l'objet $\underline{A} = A[1]$:

$$A[n] \;\cong\; \underline{A}^{\times n}.$$

De même, le foncteur A transporte (toujours à isomorphisme près) chaque opération n-aire $n \to 1$ de la théorie algébrique vers une opération n-aire sur l'objet $\underline{A}$:

$$\underline{A}^{\times n} \to \underline{A}. \tag{1.14}$$

Pour cette raison, $\underline{A}$ est appelé *objet sous-jacent* du $\mathbb{L}$-modèle, et le morphisme (1.14) est appelé *interprétation* de l'opération n-aire $n \to 1$. Cette terminologie est justifiée par le fait que le $\mathbb{L}$-modèle A est caractérisé (à isomorphisme naturel près) par son objet sous-jacent et l'interprétation de chaque opération de la théorie algébrique $\mathbb{L}$.

Une théorie algébrique $\mathbb{L}$ et une catégorie $\mathcal{C}$ avec produits finis définissent une catégorie Modèle$(\mathbb{L}, \mathcal{C})$ dont les objets sont les $\mathbb{L}$-modèles sur la catégorie $\mathcal{C}$, et dont les morphismes $A \to B$ sont les transformations naturelles de A vers B vues comme des foncteurs.

En guise d'exemple, considérons la catégorie avec produits finis *libre* $\mathbb{F}^{op}$ sur la catégorie à un seul objet : cela définit une théorie algébrique dont les opérations n-aires sont précisément les n projections $\pi_1, \ldots, \pi_n : n \to 1$. La catégorie $\mathbb{F}^{op}$ peut être définie plus explicitement comme la catégorie opposée (voir la Définition 1.4) de la catégorie FinSet des ensembles finis $[m] = \{1, \ldots, m\}$ et fonctions ensemblistes. Cette théorie algébrique $\mathbb{F}^{op}$ est dite *triviale* car un $\mathbb{F}^{op}$-modèle est caractérisé (à isomorphisme près) par son objet sous-jacent de telle sorte que la catégorie Modèle$(\mathbb{F}^{op}, \mathcal{C})$ est équivalente à la catégorie $\mathcal{C}$ elle-même.

Après cette illustration pour le moins basique, considérons une théorie algébrique un peu plus riche : la théorie des monoïdes $\mathbb{M}$.

Exemple 1.56
La théorie des monoïdes $\mathbb{M}$ est la théorie linéaire dont les opérations n-aires sont les mots finis (de longueur arbitraire) construits sur l'alphabet à n lettres $[n] = \{1, \ldots, n\}$. Par exemple, l'opération ternaire codée par le mot

$$2 \cdot 3 \cdot 1 \cdot 3 \cdot 3 \cdot 1$$

décrit l'opération

$$(m_1, m_2, m_3) \mapsto m_2 + m_3 + m_1 + m_3 + m_3 + m_1$$

Par construction, la catégorie Modèle$(\mathbb{M}, \mathcal{C})$ est équivalente à la catégorie des monoïdes et morphismes de monoïdes dans $\mathcal{C}$ (voir la Définition 1.41).

Il existe un (unique) morphisme algébrique $f : \mathbb{F}^{op} \to \mathbb{M}$, qui transporte la i-ème projection $\pi_i \in \mathbb{F}^{op}(n, 1)$ vers le mot à une lettre $i \in \mathbb{M}(n, 1)$. Il est aisé de vérifier que le foncteur d'oubli associé U_f transporte chaque monoïde de la catégorie $\mathcal{C}$ vers son objet sous-jacent.

1.4.2 Théories linéaires (PROs).

La version monoïdale des théories algébriques est obtenue en adaptant les idées de Lawvere en replaçant « produit fini » par « produit tensoriel » dans toutes les définitions ayant trait aux théories algébriques. Cela donne lieu à la définition de théories linéaires.

Définition 1.57 (théorie linéaire)
Une *théorie linéaire* (PRO) $\mathbb{L}$ est la donnée d'une catégorie monoïdale dont les objets sont les entiers naturels : 0, 1, 2, etc. et dont le produit tensoriel de k objets $m_1, \ldots, m_k$ est donné par la somme arithmétique $m_1 + \ldots + m_k$.

Un $\mathbb{L}$-modèle A dans un catégorie monoïdale $\mathcal{C}$ est alors défini par un foncteur monoïdal fort (voir la Définition 1.32)

$$\mathbb{L} \to \mathcal{C}.$$

Comme pour les théories algébriques, toute théorie linéaire $\mathbb{L}$ et toute catégorie monoïdale $\mathcal{C}$ induisent une catégorie $\mathbf{MonCat_{str}}(\mathbb{L}, \mathcal{C})$ dont les objets sont les $\mathbb{L}$-modèles dans $\mathcal{C}$, et dont les morphismes $\theta : A \to B$ sont les transformations naturelles monoïdales (voir la Définition 1.34) de A vers B, vues comme des foncteurs monoïdaux.

Il est important de remarquer que la définition de la catégorie $\mathbf{MonCat_{str}}(\mathbb{L}, \mathcal{C})$ que nous venons de donner coïncide avec la catégorie Modèle$(\mathbb{L}, \mathcal{C})$ définie plus tôt pour les théories algébriques – lorsque le produit tensoriel des deux catégories $\mathbb{L}$ et $\mathcal{C}$ se trouve être des produits finis. En effet, (a) tout foncteur préservant les produits finis définit un foncteur monoïdal fort et (b) toute transformation naturelle θ entre deux foncteurs préservant les produits finis satisfait automatiquement les cohérences requises pour une transformation naturelle monoïdale.

Afin d'exhiber les différences avec les théories algébriques, nous introduisons maintenant deux théories linéaires $\mathbb{N}$ et Δ jouant le rôle de deux théories algébriques $\mathbb{F}^{op}$ et $\mathbb{M}$ définies plus haut.

- La théorie linéaire *triviale* est la catégorie monoïdale *libre* $\mathbb{N}$ générée par un objet : cela donne la théorie linéaire dans laquelle les seuls morphismes sont les identités sur chaque objet $0, 1, 2, \ldots$
- La théorie linéaire des *monoïdes* est la catégorie *simpliciale augmentée* Δ dans laquelle chaque morphisme $m \to n$ est une fonction monotone de l'ensemble ordonné $[m] = \{1, ..., m\}$ vers l'ensemble ordonnée $[n] = \{1, ..., n\}$.

Comme pour les théories algébriques, la catégorie $\mathbf{MonCat_{str}}(\mathbb{N}, \mathcal{C})$ est équivalente à la catégorie $\mathcal{C}$, et la catégorie $\mathbf{MonCat_{str}}(\mathbb{N}, \mathcal{C})$ est équivalente à la catégorie des monoïdes et homomorphismes de monoïdes dans $\mathcal{C}$. Le lecteur attentif remarquera par exemple que la notion d'homomorphisme de monoïdes entre deux monoïdes est capturée par la notion de transformation naturelle monoïdale – pas seulement naturelle – entre deux Δ-modèles.

1.4.3 Théories symétriques (PROPs).

Définition 1.58 (théorie symétrique)
Une *théorie symétrique* (PROP) $\mathbb{L}$ est la donnée d'une catégorie monoïdale symétrique dont les objet sont les entiers naturels : 0, 1, 2, etc. et dont le produit tensoriel de k objets $m_1, \ldots, m_k$ est donné par la somme arithmétique $m_1 + \ldots + m_k$.

Afin d'exhiber les différences avec les théories algébriques, nous introduisons maintenant deux théories linéaires $\mathbb{N}$ et Δ jouant le rôle de deux théories algébriques $\mathbb{F}^{op}$ et $\mathbb{M}$ définies plus haut.

- La théorie symétrique *triviale* est la catégorie monoïdale symétrique *libre* Bij générée par un objet : c'est la catégorie des ensembles finis et des bijections.
- La théorie symétrique des *monoïdes commutatif* est la catégorie des ensembles finis FinSet et fonctions ensemblistes.
- dualement, la théorie symétrique des *comonoïdes commutatifs* est la théorie algébrique triviale $\mathbb{F}^{op}$.

Le foncteur d'inclusion de Bij dans $\mathbb{F}^{op}$ induit un foncteur d'oubli

$$\mathbf{SMonCat_{fort}}(\mathbb{F}^{op}, \mathcal{C}) \xrightarrow{U_{\mathbb{F}^{op}}} \mathcal{C}$$

qui associe à tout comonoïde commutatif son objet sous-jacent. La proposition et le corollaire suivants ont été démontrés dans [Mel08].

Proposition 1.59 Une catégorie monoïdale symétrique $\mathcal{C}$ est cartésienne si et seulement si le foncteur d'oubli $U_{\mathbb{F}^{op}}$ est un isomorphisme.

Autrement dit, les catégories cartésiennes sont exactement les catégories monoïdales symétriques dont tous les objets sont (naturellement) des comonoïdes commutatifs.

Corollaire 1.60 La catégorie des comonoïdes commutatifs et morphismes de comonoïdes sur une catégorie monoïdale symétrique est cartésienne.

1.5 Logique et modèles catégoriques

L'isomorphisme de Curry-Howard, entre les termes du λ-calcul simplement typé et les preuves de la logique intuitionniste minimale, est devenu un résultat phare de l'informatique théorique. Un résultat connexe, peut-être moins connu, est que le λ-calcul simplement typé (avec appariement surjectif) est le langage interne des catégories cartésiennes fermées. Cela veut dire en essence qu'il peut être interprété dans toute catégorie cartésienne fermée et que cette interprétation n'est pas dégénérée ; au sens où elle n'introduit pas d'égalité structurelle superflue.

De notre point de vue, ce sont les catégories cartésiennes fermées qui donnent lieu à la logique intuitionniste minimale au sens où son calcul des séquents provient naturellement des lois catégoriques. En effet, la structure cartésienne permet de décrire des séquents de la forme

$$\Gamma \vdash A \stackrel{\text{def}}{=} A_1, \ldots, A_n \vdash A$$

où la virgule représente le produit. Ce séquent se lit « à partir des hypothèses $A_1, \ldots, A_n$, je peux prouver la formule A ». L'identité et la composition de la catégorie donnent lieu aux deux séquents

$$\frac{}{A \vdash A}\ \text{Axiome} \qquad \text{et} \qquad \frac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\ \text{Coupure.}$$

Enfin, l'adjonction qui décrit la clôture donne lieu aux deux séquents

$$\frac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A \Rightarrow B \vdash C}\ \Rightarrow\text{-Gauche} \qquad \text{et} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-Droit.}$$

Ces deux séquents viennent respectivement du morphisme d'évaluation donné par la counité de l'adjonction, et de la bijection liée à l'adjonction. L'aspect un peu singulier du séquent de gauche – vis-à-vis du point de vue catégorique – vient du fait que le calcul des séquents refuse l'élimination des constructeurs. En déduction naturelle, on aurait eu une règle plus proche de l'évaluation

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B}\ \Rightarrow\text{-Élimination,}$$

mais en calcul des séquents, chaque règle d'élimination est interprétée par une règle duale de création à gauche du symbole $\vdash$. En un sens, le calcul des séquents interprète de façon contravariante les règles d'élimination. Notons que le caractère cartésien de la catégorie correspond au fait que le contexte Γ, dans la règle de coupure ou la règle ($\Rightarrow$-Gauche), n'est pas distingué. Par exemple, si on avait voulu une version monoïdale de la règle ($\Rightarrow$-Gauche), il aurait plutôt fallu écrire le séquent

$$\frac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\ \multimap\text{-Gauche.}$$

Ceci indique que l'on peut faire un travail similaire en obtenant la logique linéaire intuitionniste en lieu et place de la logique intuitionniste minimale.

1.5.1 Logique linéaire et catégories monoïdales symétriques fermées

Le fragment multiplicatif de la logique linéaire intuitionniste correspond aux catégories monoïdales symétriques fermées. Nous expliquons maintenant comment les séquents de la logique linéaire intuitionniste peuvent être obtenu à partir de leur sémantique catégorique. Le lecteur peut se référer à la monographie de Paul-André Melliès [Mel08] pour un description plus complète.

À nouveau, la structure monoïdale permet de décrire des séquents de la forme

$$\Gamma \vdash A \stackrel{\text{def}}{=} A_1, \ldots, A_n \vdash A$$

où la virgule représente le produit tensoriel. L'identité et la composition de la catégorie induisent les deux séquents

$$\frac{}{A \vdash A}\ \text{Axiome} \qquad \text{et} \qquad \frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ \text{Coupure.}$$

Remarquons la différence avec le cas cartésien, car dorénavant, le contexte est géré de façon linéaire (ou monoïdale). On ne peut pas identifier les contextes Γ et Δ car on distingue les hypothèses qui servent à prouver A, des hypothèses qui servent à prouver B. La bifonctorialité du produit tensoriel donne lieu aux deux séquents

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ \otimes\text{-Gauche} \qquad \text{et} \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{-Droit}$$

et l'existence d'un élément neutre est assuré par les séquents

$$\frac{\Gamma \vdash A}{\Gamma, 1 \vdash A}\ 1\text{-Gauche} \qquad \text{et} \qquad \frac{}{\vdash 1}\ 1\text{-Droit.}$$

Notons que la loi d'associativité est automatiquement satisfaite par la structure de séquent ; tout comme la loi d'effacement à gauche et la loi d'effacement à droite de l'élément neutre. La symétrie du produit tensoriel induit un séquent correspondant à la loi d'échange :

$$\frac{\Gamma, A_1, A_2, \Delta \vdash B}{\Gamma, A_2, A_1, \Delta \vdash B}\ \text{Échange.}$$

À l'image du cas cartésien, la fermeture est donnée par deux séquents similaires aux séquents (⇒-Gauche) et (⇒-Droit). La différence réside dans une gestion linéaire des contextes :

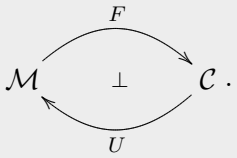
$$\frac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\ \multimap\text{-Gauche} \qquad \text{et} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap\text{-Droit.}$$

On vient donc de retrouver le fragment multiplicatif de la logique linéaire intuitionniste à travers sa sémantique catégorique. Nous voulons maintenant traiter le fragment exponentiel.

1.5.2 Logique linéaire et adjonctions monoïdales

La littérature présente de nombreuses variantes catégoriques qui donnent toutes lieu à un modèle du fragment multiplicatif et exponentiel de la logique linéaire intuitionniste (ILL). On note parmi ces modèles les catégories de Seely, les catégories de Lafont, les catégories linéaires et les catégorie Linéaire/Non Linéaire. Nous choisissons ici de considérer ces dernières car elles seront à la base de notre définition des modalités de ressource au Chapitre 2.

Définition 1.61 (catégorie Linéaire/Non Linéaire)
Une catégorie Linéaire/Non Linéaire [Ben95] est la donnée :
- d'une catégorie monoïdale symétrique fermée $(\mathcal{C}, \otimes, I)$;
- d'une catégorie cartésienne $(\mathcal{M}, \times, u)$;
- d'une adjonction monoïdale symétrique



.

L'adjonction monoïdale est exactement ce qu'il manquait à la catégorie monoïdale symétrique fermée pour pouvoir interpréter ILL. En effet, la comonade induite par l'adjonction construit une exponentielle sur la catégorie $\mathcal{C}$.

Proposition 1.62 ([BBHdP92],[Ben95]) Toute catégorie Linéaire/Non Linéaire donne lieu à un modèle de ILL.

Nous montrons maintenant comment déduire les nouveau séquents induits par cette adjonction monoïdale. Tout d'abord, cette adjonction construit une comonade $F \circ U$ sur $\mathcal{C}$ avec une comultiplication – interprétée par la règle (Promotion) – et une counité – interprétée par la règle (Déréliction). On note $!A$ l'image d'un objet A par cette comonade et on note $!Gamma$ l'application de ! à chaque formule du contexte Γ.

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash\ !A}\ \text{Promotion} \qquad \text{et} \qquad \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}\ \text{Déréliction.}$$

Remarquons que la règle (Promotion) n'est pas complètement canonique par rapport à sa sémantique catégorique. D'autres formulations plus fonctorielle sont possibles, mais elles ne vérifient pas la propriété de la « sous formule » : toutes les formules apparaissant dans les prémisses sont des sous formules des formules apparaissant dans la conclusion. C'est la formulation usuelle du système (S4) de la logique modale en calcul des séquents. Comme la catégorie $\mathcal{M}$ est cartésienne et que l'adjonction est monoïdale, on a une structure de comonoïde commutatif sur les objets de la forme $!A$ dans $\mathcal{C}$. La counité du comonoïde donne la règle (Affaiblissement) et la duplication donne la règle (Contraction).

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B}\ \text{Affaiblissement} \qquad \text{et} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\ \text{Contraction}$$

Il apparaît que ces quatre séquents sont suffisants pour capturer l'essence des catégories Linéaire/Non Linéaire. Le lecteur ne sera pas surpris de voir qu'on vient de retrouver la totalité du calcul des séquents pour ILL.

Nous utiliserons, au Chapitre 2, cette interprétation des modèles catégoriques vers le calcul des séquents. Cela nous permettra d'extraire la structure logique qui se dégage de la sémantique catégorique de la logique tensorielle.

Chapitre 2

Dualité et modalités de ressource

Festival mondial pour la paix,
Picasso (Berlin, 1951)

Sommaire

Malgré son apparente efficacité pour modéliser le fragment multiplicatif de la logique linéaire, la sémantique des jeux a longtemps résisté aux sollicitations pour qu'elle interprète la logique linéaire toute entière. Ceci est en particulier lié au problème de Blass, mis en avant par Samson Abramsky et Radha Jagadeesan [AJ94], qui indique que la sémantique des jeux introduite par Andrea Blass [Bla92] ne définit pas un modèle catégorique. Il a fallut attendre le début des années 2000 et les travaux de Paul-André Melliès [Mel05a, Mel05b] sur les jeux asynchrones pour obtenir un premier résultat de complétude. Curieusement, la solution qui y est proposée s'articule autour d'un quotient sur les stratégies – quotient qui, à la lumière des travaux de Masahito Hasegawa, revient à transformer la monade de continuation forte en monade commutative. Il est donc naturel de se demander ce que serait une logique linéaire où la monade de continuation serait non commutative ; ce qui nous amène dans ce chapitre à étudier la logique tensorielle – logique plus primitive et plus directement liée à la sémantique des jeux.

Il apparaît alors que la plupart des modèles de la logique linéaire sont en fait obtenus à partir de modèles de logique tensorielle, dotés d'une monade commutative. C'est le cas par exemple pour les espaces de cohérence, les espaces de phases, les espaces de finitude [Ehr05], etc. Un des intérêt de cette approche est la possibilité de définir différentes négations sur la même catégorie monoidale – ceci permettant par exemple d'intégrer de manière harmonieuse différents modèles de logique linéaire (espaces de cohérence, espace de finitude, etc.).

Cet éclairage de la logique linéaire par la logique tensorielle ressemble à s'y méprendre à celui apporté par les logiques polarisés [Lau02a]. La différence majeure réside dans notre volonté de ne pas confondre les modalités de ressources, comme l'exponentielle,

avec la négation ; et donc de se dégager du paradigme cartésien qui sous-tend les logiques polarisées.

Nous présentons en fin de chapitre un modèle de la logique tensorielle basé sur les jeux de Conway [Joy77], agrémentés d'une notion de gain qui permet de définir différentes modalités de ressources.

2.1 Dualité

Dans cette section, nous introduisons une logique de tenseur et de négation – appelé logique tensorielle. Nous procédons à l'inverse de l'ordre habituel en théorie de la preuve : au lieu de définir en premier la logique, nous commençons par décrire sa sémantique catégorique. La raison de cette entorse à la tradition tient à ce que la situation algébrique est basique et concise. Une fois que nous aurons donné le cadre catégorique, nous formulerons le calcul des séquents de la logique induite en utilisant le dictionnaire donné à la Section 1.5 – ce qui nous mènera vers la logique tensorielle.

2.1.1 Négation tensorielle

Définition 2.1 (négation tensorielle)
Une *négation tensorielle* sur un catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I)$ est la donnée d'un foncteur

$$\neg \ : \ \mathcal{C} \longrightarrow \mathcal{C}^{\mathrm{op}}$$

et d'une famille de bijections

$$\varphi_{A,B,C} \ : \ \mathcal{C}(A \otimes B, \neg C) \cong \mathcal{C}(A, \neg(B \otimes C))$$

naturelle en A, B et C de telle sorte que le diagramme

$$\begin{array}{ccc}
\mathcal{C}(A \otimes (B \otimes C), \neg D) & \xrightarrow{\mathcal{C}(\alpha_{A,B,C}, \neg C)} & \mathcal{C}((A \otimes B) \otimes C, \neg D) \\
\Big\downarrow{\scriptstyle \varphi_{A,B\otimes C,D}} & & \Big\downarrow{\scriptstyle \varphi_{A\otimes B,C,D}} \\
 & & \mathcal{C}(A \otimes B, \neg (C \otimes D)) \\
 & & \Big\downarrow{\scriptstyle \varphi_{A,B,C\otimes D}} \\
\mathcal{C}(A, \neg ((B \otimes C) \otimes D)) & \xrightarrow[\mathcal{C}(A, \neg \alpha^{-1}_{B,C,D})]{} & \mathcal{C}(A, \neg (B \otimes (C \otimes D)))
\end{array}$$

commute. Une catégorie monoïdale symétrique munie d'une négation tensorielle est appelée une *catégorie de dialogue*.

Remarquons que nous n'imposons de cohérence qu'avec l'associativité de la catégorie monoïdale, la cohérence avec les effacements λ et ρ de l'unité étant déduite par naturalité.

La Figure 2.1 donne une représentation intuitive de la négation tensorielle. Ce diagramme de cordes trouve son sens et sa rigueur dans une bicatégorie de modules. Notons aussi que la négation est placée à cheval sur l'ensemble des morphismes et non d'un coté ou de l'autre comme c'est le cas habituellement pour un foncteur. Ceci est justifié comme nous le verrons à la Section 2.1.2 car la négation est autoduale.

Étant donné un négation, il est d'usage de définir la formule *faux* comme l'objet $\bot \stackrel{\mathrm{def}}{=} \neg I$ obtenu en « niant » l'objet unité I de la catégorie monoïdale.

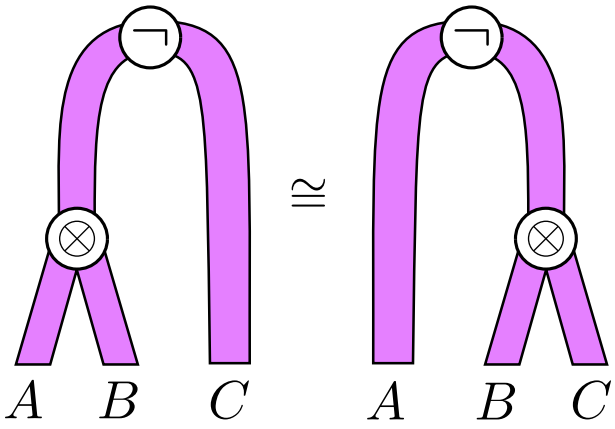


FIG. 2.1 – Dessin de la négation tensorielle

Remarquons que la bijection $\varphi_{A,B,I}$ équipe la catégorie $\mathcal{C}$ d'un bijection $\exp(\varphi)_{A,B}$, en posant $\perp^B = \neg B$

$$\exp(\varphi)_{A,B} = \mathcal{C}(A, \neg(\rho^{-1})) \circ \varphi_{A,B,I} \quad : \quad \mathcal{C}(A \otimes B, \perp) \quad \cong \quad \mathcal{C}(A, \perp^B) \tag{2.1}$$

pour tous objets A et B. On dit dans ce que l'objet $\perp$ est *exponentiable*.

Proposition 2.2 L'existence d'un objet $\perp$ exponentiable est équivalente à l'existence d'une négation tensorielle.

Démonstration : On sait déjà qu'une négation définit l'objet exponentiel $\neg I$. Toute objet exponentiable définit à sont tour une négation par

$$\mathrm{neg}(\psi)_{A,B,C} = \psi_{A,B\otimes C} \circ \mathcal{C}(\alpha^{-1}_{A,B,C}, \perp) \circ \psi^{-1}_{A\otimes B,C}$$

La cohérence de la négation ainsi que la naturalité des bijections permettent de déduire que ces deux constructions sont inverses l'une de l'autre. ■

C'est pour cette raison que la définition de la négation $\neg$ est souvent remplacée par l'énoncé – quelque peu informel – que « l'objet $\perp$ est exponentiable » dans la catégorie symétrique monoïdale $\mathcal{C}$, ce qui donne lieu à la négation $\neg A$ notée $\perp^A$.

Exemple 2.3
Soit $\perp$ un objet d'une catégorie monoïdale symétrique fermée $\mathcal{C}$ ayant pour fermeture $\multimap$. Le foncteur qui à tout objet A de $\mathcal{C}$ associe l'objet $A \multimap \perp$ définit une négation tensorielle sur $\mathcal{C}$.

2.1.2 Auto-adjonction

Dans sa thèse, Hayo Thielecke [Thi97] observe pour la première fois le phénomène fondamental d'« auto-adjonction », relié à la négation. Cette observation joue un rôle prépondérant dans un travail non publié de Paul-André Melliès et Peter Selinger [MS] sur les catégories polaires, une sémantique catégorique de la logique linéaire polarisée, des continuations et des jeux. La même idée réapparaît plus récemment dans une étude très complète des catégories polarisées Robin Cockett et Robert Seely [CS07]. En effet, une

catégorie symétrique monoïdale avec une négation tensorielle est une instance particulière de catégorie polarisée $\mathcal{C} \times \mathcal{C}^{op} \to \mathbf{Ens}$ induite par une adjonction. Dans notre cas, le phénomène d'« auto-adjonction » se ramène au fait que toute négation tensorielle est adjointe à gauche à son foncteur opposé.

$$\neg \ : \ \mathcal{C}^{op} \longrightarrow \mathcal{C} \tag{2.2}$$

à cause de la bijection naturelle

$$\mathcal{C}^{op}(\neg A, B) \quad \cong \quad \mathcal{C}(A, \neg B).$$

Remarquons que nous notons indifféremment la négation et son dual par $\neg$, ceci est inexact mais habituel en théorie de la continuation, où l'on ne se soucie guère de catégories.

2.1.3 Monade de continuation

Toute négation tensorielle $\neg$ induit une adjonction et donc une monade

$$\neg\neg \quad : \quad \mathcal{C} \longrightarrow \mathcal{C}$$

Cette monade est appelée la *monade de continuation* associée à la négation. Eugenio Moggi [Mog91] a fait la remarque fondamentale que la monade de continuation est *forte* mais pas commutative en général (voir la Section 1.3.6 pour des détails sur les définitions).

Proposition 2.4 La négation tensorielle induit une monade forte $\neg\neg$

Démonstration : Pour rendre la démonstration plus lisible, nous notons L la négation sur $\mathcal{C}$ et R son dual. D'abord, remarquons que la définition de négation nous donne deux morphismes

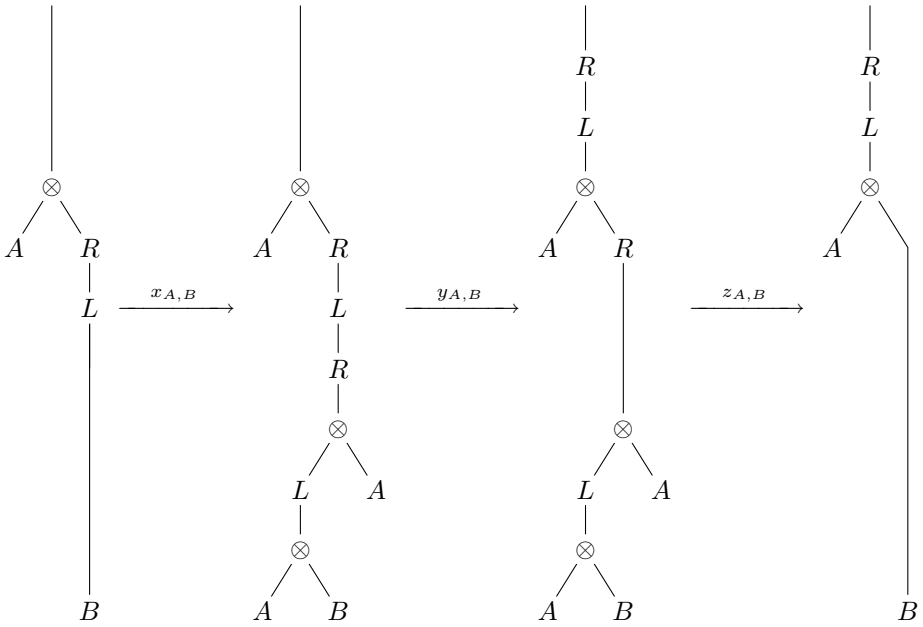
$$B \xrightarrow{\eta_{A,B}} R(L(A \otimes B) \otimes A) \qquad L(A \otimes R(B \otimes A)) \xrightarrow{\varepsilon_{A,B}} B$$

Ces deux morphismes sont appelés respectivement unité et counité de la négation. En particulier, $\eta_{I,-}$ et $\varepsilon_{I,-}$ sont l'unité et la counité de la monade de continuation. Il vérifier la loi de zigzag :

$$\begin{array}{ccc} L(A \otimes B) & = & L(A \otimes B) \\ & {\scriptstyle L\eta_{A,B}} \searrow \quad \nearrow {\scriptstyle \varepsilon_{A,L(A\otimes B)}} & \\ & L(A \otimes R(L(A \otimes B) \otimes A)) & \end{array} \tag{2.3}$$

Il vont maintenant nous permettre de construire la force. Nous utilisons la dualité de

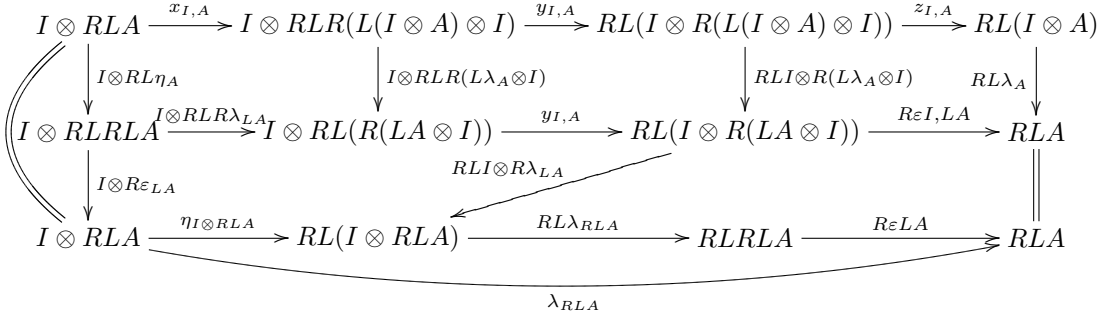
Poincaré pour améliorer la lisibilité.



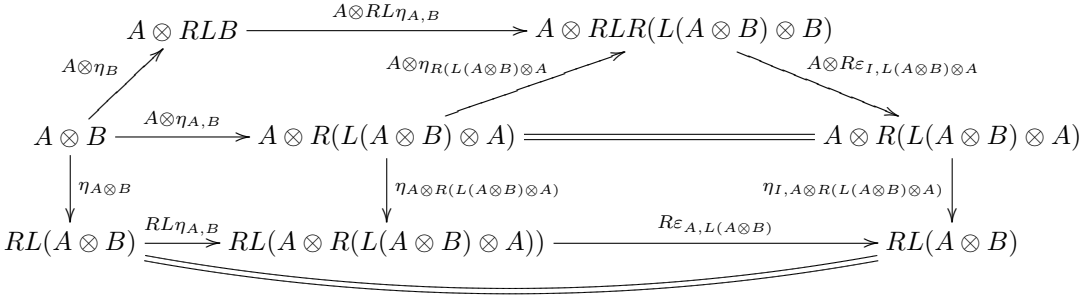
où

- $x_{A,B} = A \otimes RL\eta_{A,B}$;
- $y_{A,B} = \eta_{I,A\otimes R(L(A\otimes B)\otimes A)} \circ (A \otimes R\varepsilon_{I,L(A\otimes B)\otimes A})$;
- $z_{A,B} = R\varepsilon_{A,L(A\otimes B)}$.

La vérification des lois de cohérence nécessaire pour avoir est assez délicate. Le Diagramme 1.47 de cohérence avec l'unité du produit tensoriel commute car on peut le paver avec les tuiles de naturalité et d'identité de zigzag suivantes :



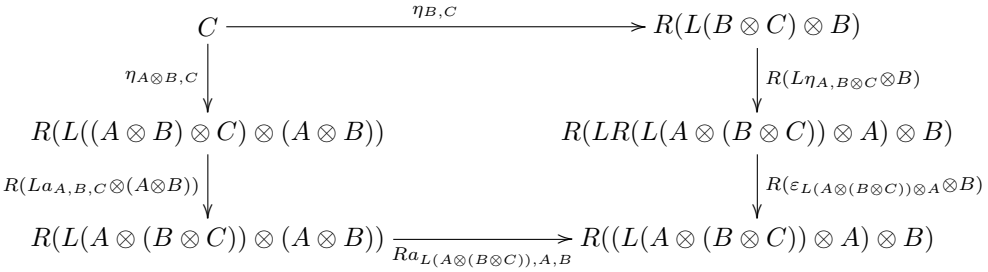
Le Diagramme 1.12 de cohérence avec l'unité de la monade commute car on peut le paver avec les tuiles de naturalité et d'identité de zigzag suivantes :



Le Diagramme 1.10 de cohérence avec la multiplication de la monade commute facilement en remarquant que

$$\mu_A \stackrel{\text{def}}{=} R\epsilon_{LA}$$

Enfin, on remarque que la cohérence de la négation avec l'associativité du produit tensoriel implique que le diagramme



commute. Ce diagramme constitue la clef d'arc permettant de paver le Diagramme 1.9 de cohérence entre la force et l'associativité du produit tensoriel. ■

Une négation tensorielle est dite *commutative* lorsque la monade de continuation associée dans $\mathcal{C}$ est commutative – ou de manière équivalente lorsque la monade est monoïdale au sens relâché.

2.1.4 Implication linéaire

Une catégorie monoïdale symétrique $\mathcal{C}$ équipée d'une négation n'est pas très loin d'être monoïdale *fermée*. Il est en effet possible de définir une *implication linéaire* $\multimap$ lorsque sa cible $\neg B$ est un objet nié :

$$A \multimap \neg B \quad \stackrel{\text{def}}{=} \quad \neg (A \otimes B).$$

De cette façon, le foncteur (2.2) définit ce que nous appelons un *idéal exponentiel* sur la catégorie $\mathcal{C}$ (voir la Définition 1.39). Lorsque le foncteur est injective sur les objets et les morphismes, il est alors possible d'identifier cet idéal exponentiel avec la sous-catégorie de $\mathcal{C}$ constituée des *objets niés*. La notion d'idéal exponentiel discutée dans la thèse de Guy McCusker [McC96] apparaît précisément de cette manière. Remarquons que la négation $\neg$ n'a pas besoin d'être fidèle sur les objets car nous avons pris soin de ne pas définir un idéal exponentiel comme une sous-catégorie de $\mathcal{C}$.

2.1.5 Catégories de continuation

Toute catégorie de dialogue $\mathcal{C}$ avec pour négation $\neg$ donne lieu à une *catégorie de continuation* $\mathcal{C}^{\neg}$.

Définition 2.5 (catégorie de continuation)
Étant donnée une catégorie de dialogue $\mathcal{C}$ avec pour négation $\neg$, on construit la *catégorie de continuation* $\mathcal{C}^{\neg}$ dont les objets sont les mêmes que ceux de $\mathcal{C}$, et dont les morphismes sont définis par

$$\mathcal{C}^{\neg}(A, B) \quad \stackrel{\text{def}}{=} \quad \mathcal{C}(\neg A, \neg B).$$

Notons que la catégorie $\mathcal{C}^{\neg}$ est la catégorie de Kleisli associée à la comonade dans $\mathcal{C}^{op}$ induite par l'auto-adjonction. Celle-ci étant aussi la catégorie opposée de la catégorie de Kleisli associée à la monade de continuation sur $\mathcal{C}$. Comme la monade de continuation est forte, la catégorie $\mathcal{C}^{\neg}$ est *prémonoïdale* au sens de John Power et Edmund Robinson [PR97] (voir le Chapitre 1 pour la définition et le Chapitre 5 pour de plus amples explications).

2.1.6 Exemple des espaces de phase

Jean-Yves Girard a introduit le modèle des espaces de phase pour donner une version linéaire de la sémantique basée sur une notion de valeurs de vérité de Tarski. Nous pouvons recomprendre ce modèle à la lumière de la logique tensorielle.

Soit $\mathbb{M}$ un comonoïde commutatif dans **Ens**. Pour tous sous-ensembles $X, Y \subseteq \mathbb{M}$, on définit les connecteurs :

$$X \otimes Y \stackrel{\text{def}}{=} \{xy \mid x \in X \text{ et } y \in Y\} \quad \text{et} \quad X \multimap Y \stackrel{\text{def}}{=} \{f \mid \forall x \in X, fx \in Y\}$$

On peut montrer l'équivalence

$$X \otimes Y \subseteq Z \quad \text{ssi} \quad X \subseteq Y \multimap Z$$

La catégorie des espaces de phase $\mathcal{C}$ est le préordre dont les éléments sont les sous-ensembles de $\mathbb{M}$ munis de la relation d'ordre induite par l'inclusion. L'équivalence du dessus nous montre que $(\mathcal{C}, \otimes, \emptyset, \multimap)$ définit une catégorie monoïdale symétrique fermée.

On se donne alors un sous-ensemble particulier $\perp$ de $\mathbb{M}$ que l'on appelle un *pôle*. Comme indiqué dans l'Exemple 2.3, l'opérateur qui a X associe $X \multimap \perp$ définit une négation tensorielle. On construit ensuite un modèle de MLL en restreignant la catégorie des espaces de phase aux sous-ensembles égaux à leur double négation appelés *faits*, c'est-à-dire vérifiant

$$\neg\neg X \subseteq X \quad \text{ou de manière équivalente} \quad X = \neg\neg X$$

Cela revient à considérer la catégorie (d'Eilenberg-Moore) des algèbres induite par la (co)monade de continuation, qui dans ce cas particulier est équivalente à la catégorie de continuation car

$$X \subseteq \neg\neg Y \quad \text{ssi} \quad \neg\neg X \subseteq \neg\neg Y.$$

En fait, la catégorie de Kleisli coïncide avec la catégorie d'Eilenberg-Moore dès que la monade est idempotente (voir la Définition 1.19) ; ce qui est le cas ici. On en déduit que la catégorie de continuation donne un modèle de MLL, ce qui n'est pas surprenant – comme nous le verrons en Section 2.3.4 – car la monade de continuation des espaces de phase est commutative.

Remarquons d'ailleurs que beaucoup de modèles de logique linéaire apparaissent de cette façon. C'est la cas par exemple du modèle historique des espaces de cohérence ou de modèles plus récents comme les espaces de finitude [Ehr05], ou les espace de Köthe [Ehr02]. Nous verrons plus en détail ces constructions à la Section 2.3.6. Il est intéressant de noter que cette construction adapte à notre langage linéaire le fait bien connu que les objets niés (de la forme $\neg A$) d'une algèbre de Heyting définissent une algèbre de Boole.

2.2 Modalités de ressource

2.2.1 Une description catégorique des ressources

Lorsque l'on réfléchit à la notion de modalité de ressource – guidé en particulier par le préalable offert par l'exponentielle de la logique linéaire définie par Jean-Yves Girard – il semble naturel de la voir comme la description d'une certaine classe d'objets d'une catégorie monoïdale $\mathcal{C}$ pour lesquels le tenseur possède des propriétés supplémentaires. Par exemple, un objet de la forme $!A$ en logique linéaire est un objet que l'on peut dupliquer et effacer à sa guise. Comment cette structure intrinsèque se marie-t-elle avec à

Modalité	Catégorie $(\mathcal{M}, \bullet, u)$
Affine	l'unité u est terminale
Dupliquante	existence d'une duplication naturelle pour tout objet
Exponentielle	la structure de $\mathcal{M}$ est cartésienne

TAB. 2.1 – Les modalités de ressource affine, dupliquante et exponentielle

la structure déjà présente dans la catégorie monoïdale sous-jacente ? ... via une adjonction bien entendu. En effet, du point de vue catégorique, il est naturel de voir cette interaction comme une adjonction entre la catégorie $\mathcal{C}$ de départ et une catégorie monoïdale $\mathcal{M}$ plus riche symbolisant la classe d'objets possédant des propriétés supplémentaires.

Définition 2.6 (modalité de ressource)
Une *modalité de ressource* sur une catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I)$ est la donnée d'une adjonction

$$\mathcal{M} \underset{F}{\overset{U}{\rightleftarrows}} \mathcal{C} \quad (\perp) \tag{2.4}$$

où
- $(\mathcal{M}, \bullet, u)$ est un catégorie monoïdale symétrique ;
- U est un foncteur symétrique monoïdal fort.

Comme introduit dans les préliminaires, un foncteur symétrique monoïdal fort transporte la structure de $(\mathcal{M}, \bullet, u)$ vers la structure de $(\mathcal{C}, \otimes, I)$, à isomorphismes cohérents près. Ainsi, toute propriété sur le tenseur $\bullet$ valide dans $\mathcal{M}$ donne lieu à une propriété similaire pour les objets de la forme UA de la catégorie $\mathcal{C}$.

Cette définition de modalité de ressource est très largement inspirée de la notion de modèle Linéaire/Non Linéaire (Définition 1.61) introduit par Nick Benton [Ben95] – qui peut être reformulée maintenant comme une catégorie monoïdale symétrique $\mathcal{C}$ munie d'une modalité de ressource de type exponentielle (voir la Section 2.2.2). Par la suite, nous identifierons très souvent la modalité de ressource et la comonade $! = U \circ F$ induite sur la catégorie $\mathcal{C}$.

Nous présentons brièvement une définition plus conceptuelle de la modalité de ressource dans un langage 2-catégorique.

Définition bis 2.6 (modalité de ressource)
Une *modalité de ressource* est une adjonction dans la 2-catégorie des catégories symétriques monoïdales, foncteurs symétriques monoïdaux relâchés et transformations monoïdales.

2.2.2 Différents types de modalité

Nous allons maintenant décrire différents types de ressources déjà étudiées dans la littérature en spécialisant la catégorie $\mathcal{M}$ comme ceci est résume dans le Tableau 2.1.

Affine. On dit qu'une modalité de ressource est *affine* lorsque l'unité $\mathcal{M}$ de la catégorie monoïdale $\mathcal{M}$ possède un objet terminal.

Dupliquante. On dit qu'une modalité de ressource est *dupliquante* lorsque tout objet de la catégorie $\mathcal{M}$ est dupliquable, i.e. lorsqu'il existe une diagonale

$$\delta_A \quad : \quad A \longrightarrow A \otimes A$$

naturelle en A, compatible avec la symétrie et satisfaisant les diagrammes de cohérence.

$$\begin{array}{ccc} A & \xrightarrow{\delta_A} & A \otimes A \\ & \searrow^{\delta_A} & \downarrow{\scriptstyle \sigma_{A,A}} \\ & & A \otimes A \end{array} \qquad (2.5)$$

$$\begin{array}{ccc} A & \xrightarrow{\delta_A} & A \otimes A \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle A \otimes \delta_A} \\ A \otimes A & \xrightarrow[\delta_A \otimes A]{} & A \otimes A \otimes A \end{array} \qquad (2.6)$$

Exponentielle. On dit qu'une modalité de ressource est *exponentielle* lorsque le produit tensoriel $\bullet$ est un produit cartésien, et l'unité u est l'objet terminal de $\mathcal{M}$. Ainsi, tout objet de $\mathcal{M}$ est un comonoïde commutatif et la modalité exponentielle est à la fois affine et dupliquante.

Remarque 2.7
Le travail de Bart Jacobs sur les modalités affine et dupliquante [Jac94] est basé sur les notions éponymes pour une monade commutative sur une catégorie cartésienne fermée. Il considère ensuite la catégorie d'Eilenberg-Moore de cette monade (affine ou dupliquante) pour obtenir un modèle de logique linéaire intuitionniste (ILL) avec une modalité (affine ou dupliquante). Le problème de cette construction est qu'elle se restreint à certains modèles très spécifiques de ILL qui sont obtenus comme des catégories d'algèbres.

2.3 Logique tensorielle

De notre point de vue, la logique tensorielle est décrite par sa sémantique catégorique. Nous donnons donc en premier lieu la structure catégorique décrivant cette logique. Nous donnerons ensuite une présentation plus traditionnelle à base de calcul des séquents, et ceci de deux manières différentes mais équivalentes : bilatérale ou monolatérale. Le point de contact entre la description catégorique et la description à base de calcul des séquents a été étudié plus en détail dans [Mel].

Dans un premier temps, toute catégorie de dialogue $\mathcal{C}$ définit un modèle de la logique tensorielle *multiplicative*.

Lorsque cette catégorie $\mathcal{C}$ est de plus munie de coproduits finis (notés $\oplus$) qui commutent avec le produit tensoriel, elle définit un modèle de la logique tensorielle *multiplicative et additive*. La commutation avec le produit tensoriel signifie que les morphisme canoniques :

$$\begin{array}{rcl} (A \otimes B) \oplus (A \otimes C) & \to & A \otimes (B \oplus C) \\ 0 & \to & A \otimes 0 \end{array}$$

sont des isomorphismes.

Enfin, un modèle de *logique tensorielle* (complète) est défini comme un modèle de la logique tensorielle multiplicative et additive munie d'une modalité de ressource affine (dont la comonade est notée $!_w$), d'une modalité dupliquante (dont la comonade est notée $!_c$) et d'une modalité exponentielle (dont la comonade est notée $!_e$). En résumé,

Définition 2.8 (modèle de logique tensorielle)
Un modèle catégorique de *logique tensorielle* est la donnée d'une catégorie de dialogue $\mathcal{C}$, des coproduits finis, d'une modalité affine, dupliquante et exponentielle. On demande de plus que les coproduits finis commutent avec le produit tensoriel.

Logique	Catégorie
Multiplicative	$\otimes$, $\neg$
Multiplicative Additive	$\otimes$, $\neg$, $\oplus$ et commutation avec $\otimes$
Complète	$\otimes$, $\neg$, $\oplus$ et commutation avec $\otimes$ + modalités affine, dupliquante et exponentielle.

Tab. 2.2 – Les différents fragments de la logique tensorielle

2.3.1 Présentation bilatérale

Les formules $A, B, \ldots$ de la logique tensorielle (dans sa présentation bilatérale) sont construites de la manière suivante :

multiplicatives	$1 \mid \neg A \mid A \otimes B$
additives	$0 \mid A \oplus B$
modalités de ressource	$!_w A \mid ?_c A \mid !_e A$

Il y a deux types de séquents : $\Gamma \vdash A$ où Γ est un contexte, et A est une formule ; $\Gamma \vdash$ où Γ est un contexte. La Figure 2.2 présente le calcul des séquents du fragment multiplicatif. Les quatre premières règles expriment la structure monoïdale de $\mathcal{C}$, les deux suivantes définissent une négation tensorielle et les deux dernières représentent simplement l'identité et la composition de la catégorie $\mathcal{C}$. La Figure 2.3 présente les règles pour gérer les coproduits finis. La Figure 2.4 introduit les règles de la modalité exponentielle qui sont bien évidemment les mêmes que celles du ! de la logique linéaire. Les règles de la modalité affine données en Figure 2.5 sont les mêmes que celles de la modalité exponentielle, mais sans la contraction. Les règles de la modalité dupliquante données en Figure 2.6 sont les mêmes que celles de la modalité exponentielle, mais sans affaiblissement.

2.3.2 Présentation monolatérale

Pour passer de la présentation bilatérale à la présentation monolatérale, nous devons introduire la notion de polarité. Les formules A qui étaient sur la droite dans la présentation bilatérale reste à droite, et sont appelées *positives*. De façon duale, les formules qui étaient sur la gauche se retrouve sur la droite, et sont appelées *négatives*. Cette transformation est schématisée par le tableau suivant

Présentation Bilatérale		Présentation Monolatérale
$\Gamma \vdash$	$\rightsquigarrow$	$\vdash \Gamma^*$
$\Gamma \vdash A$	$\rightsquigarrow$	$\vdash \Gamma^*, A$

Ainsi, il y a deux types de séquents dans cette formulation : les séquents de la forme $\vdash \Gamma$ où Γ ne contient que des formules négatives, et les séquents de la forme $\vdash \Gamma, P$ contenant

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{-Droit} \qquad \frac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, A \otimes B, \Gamma_2 \vdash C}\ \otimes\text{-Gauche}$$

$$\frac{}{\vdash 1}\ 1\text{-Droit} \qquad \frac{\Gamma \vdash A}{\Gamma, 1 \vdash A}\ 1\text{-Gauche}$$

$$\frac{\Gamma, A \vdash}{\Gamma \vdash \neg A}\ \neg\text{-Droit} \qquad \frac{\Gamma \vdash A}{\Gamma, \neg A \vdash}\ \neg\text{-Gauche}$$

$$\frac{}{A \vdash A}\ \text{Axiome} \qquad \frac{\Gamma \vdash A \qquad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ \text{Coupure}$$

FIG. 2.2 – Logique tensorielle multiplicative : présentation bilatérale

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\text{-Droit-1} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus\text{-Gauche}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\text{-Droit-2}$$

Pas de règle d'introduction à droite pour le zéro

$$\frac{}{\Gamma, 0 \vdash A}\ (0\text{-Gauche})$$

FIG. 2.3 – Logique tensorielle additive : présentation bilatérale

$$\frac{!_e\Gamma \vdash A}{!_e\Gamma \vdash\ !_eA}\ \text{Promotion} \qquad \frac{\Gamma, A \vdash}{\Gamma, !_eA \vdash}\ \text{Déréliction}$$

$$\frac{\Gamma \vdash}{\Gamma, !_eA \vdash}\ \text{Affaiblissement} \qquad \frac{\Gamma, !_eA, !_eA \vdash}{\Gamma, !_eA \vdash}\ \text{Contraction}$$

FIG. 2.4 – Modalité exponentielle : présentation bilatérale

$$\frac{!_w\Gamma \vdash A}{!_w\Gamma \vdash\ !_wA}\ \text{Promotion} \qquad \frac{\Gamma, A \vdash}{\Gamma, !_wA \vdash}\ \text{Déréliction}$$

$$\frac{\Gamma \vdash B}{\Gamma, !_wA \vdash}\ \text{Affaiblissement}$$

FIG. 2.5 – Modalité affine : présentation bilatérale

$$\frac{!_c\Gamma \vdash A}{!_c\Gamma \vdash\ !_cA}\ \text{Promotion} \qquad \frac{\Gamma, A \vdash}{\Gamma, !_cA \vdash}\ \text{Déréliction}$$

$$\frac{\Gamma, !_cA, !_cA \vdash}{\Gamma, !_cA \vdash}\ \text{Contraction}$$

FIG. 2.6 – Modalité dupliquante : présentation bilatérale

exactement une formule positive P. Pour distinguer les formules négatives des formules positives, nous allons cloner chaque constructeur de la formulation bilatérale $0, 1, \oplus, \otimes, !_w, !_c, !_e$ en lui-même : $0, 1, \oplus, \otimes, !_w, !_c, !_e$ et son dual : $\top, \bot, \&, ⅋, ?_w, ?_c, ?_e$. La négation est elle-même clonée en deux opérations $\uparrow$ et $\downarrow$ ayant chacune un effet spécifique :

– $\uparrow$ transporte les formules positives vers des formules négatives ;

– $\downarrow$ transporte les formules négatives vers des formules positives.

Il est notable que les modalités affine, dupliquante et exponentielle ne changent pas la polarité d'une formule : c'est la différence principale avec la logique polarisée. Nous utilisons les lettres P et Q pour les formules positives, les lettres L et M pour les formules négatives et les lettres Γ et Δ pour les contextes constitués uniquement de formules négatives. Les formules sont construites sur la grammaire suivante

$$\begin{array}{lccccccccccc}
\text{Positifs :} & 0 & | & 1 & | & \downarrow L & | & P \otimes Q & | & P \oplus Q & | \\
 & & & & & !_w P & | & !_c P & | & !_e P & \\
\text{Négatifs :} & \bot & | & \top & | & \uparrow P & | & L ⅋ M & | & L \,\&\, M & | \\
 & & & & & ?_w L & | & ?_c L & | & ?_e L &
\end{array}$$

À toute formule positive P, on peut associer une formule duale négative $P^\perp$, obtenue en dualisant tous les constructeurs logiques apparaissant dans la formule P. Le calcul des séquents pour le fragment multiplicatif de la Figure 2.7 est une adaptation de celui de la Figure 2.2 ; il en est de même pour le fragment additif de la Figure 2.8 qui est une adaptation de la Figure 2.3. Les Figures 2.9, 2.10 et 2.11 qui traitent des modalités de ressources sont une adaptation des Figures 2.4, 2.5 et 2.6.

2.3.3 Jeux d'arène et logique classique

En s'appuyant sur les travaux de Hayo Thielecke, Peter Selinger [Sel01] a développé la notion de *catégorie de contrôle* afin d'axiomatiser soigneusement la sémantique de la logique classique. Ensuite, aiguillé par un résultat de complétude établi par Martin Hofmann et Thomas Streicher [HS02], il a démontré un très joli *théorème de structure* énonçant que toute catégorie de contrôle est équivalente à la catégorie de continuation $A^\neg$ d'une *catégorie de réponse* $\mathcal{C}$. Nous reprendrons cette théorie dans un cadre linéaire (et donc monoïdal) dans le Chapitre 5 de ce manuscrit.

Remarquons maintenant qu'une catégorie de dialogue $\mathcal{C}$ n'est rien d'autre qu'une catégorie de réponse $\mathcal{C}$ – pour laquelle les unités de la monade de continuation n'ont pas besoin d'être des monomorphismes – où le tenseur $\otimes$ est *cartésien* et où l'unité I est *terminale*. Une analyse uniquement basée sur la théorie de la preuve mène exactement à la même conclusion. En partant d'un travail de Jean-Yves Girard sur les polarités dans LC [Gir91], Olivier Laurent a développé une analyse complète et perspicace de la polarité en logique, incorporant la logique classique [Lau02a], les jeux d'arène (non parenthésés) [Lau02b] et les catégories de contrôle [Lau02b]. Mais il s'avère que la logique polarisée d'Olivier Laurent coïncide avec le fragment multiplicatif et additif (et donc, sans modalité exponentielle) de la logique tensorielle – où la structure monoïdale est *cartésienne*. Ceci apparaît très clairement dans la présentation monolatérale de la logique tensorielle. Nous résumons ici les différences entre la logique classique et la logique tensorielle dans un tableau très schématique.

Logique tensorielle	$\otimes$ est monoïdal $\neg$ est tensorielle
Logique classique	$\otimes$ est cartésien $\neg$ est tensorielle

$$\frac{\vdash \Gamma, P \qquad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}\otimes \qquad \frac{\vdash \Gamma_1, L, M, \Gamma_2, P}{\vdash \Gamma_1, L \parr M, \Gamma_2, P}\parr$$

$$\frac{}{\vdash 1}\,1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\,\bot$$

$$\frac{\vdash \Gamma, L}{\vdash \Gamma, \downarrow L}\downarrow \qquad \frac{\vdash \Gamma, P}{\vdash \Gamma, \uparrow P}\uparrow$$

$$\frac{}{\vdash P^\bot, P}\text{ Axiome} \qquad \frac{\vdash \Gamma, P \qquad \vdash P^\bot, \Delta, Q}{\vdash \Gamma, \Delta, Q}\text{ Coupure}$$

Fig. 2.7 – Logique tensorielle multiplicative : présentation monolatérale

$$\frac{\vdash \Gamma, P}{\vdash \Gamma, P \oplus Q}\oplus\text{-Gauche} \qquad \frac{\vdash \Gamma, L \qquad \vdash \Gamma, M}{\vdash \Gamma, L \,\&\, M}\text{ With}$$

$$\frac{\vdash \Gamma, Q}{\vdash \Gamma, P \oplus Q}\oplus\text{-Droit}$$

Pas de règle d'introduction pour le zéro

$$\frac{}{\vdash \Gamma, \top}\text{ Top}$$

Fig. 2.8 – Logique tensorielle additive : présentation monolatérale

$$\frac{\vdash\ ?_e\Gamma, P}{\vdash\ ?_e\Gamma, !_e P}\text{ Promotion} \qquad \frac{\vdash \Gamma, L}{\vdash \Gamma,\ ?_e L}\text{ Déréliction}$$

$$\frac{\vdash \Gamma}{\vdash\ !_e\Gamma,\ ?_e L}\text{ Affaiblissement} \qquad \frac{\vdash \Gamma,\ ?_e L,\ ?_e L}{\vdash \Gamma,\ ?_e L}\text{ Contraction}$$

Fig. 2.9 – Modalité exponentielle : présentation monolatérale

$$\frac{\vdash\ ?_w\Gamma, P}{\vdash\ ?_w\Gamma, !_w P}\text{ Promotion} \qquad \frac{\vdash \Gamma, L}{\vdash \Gamma,\ ?_w L}\text{ Déréliction}$$

$$\frac{\vdash \Gamma, L}{\vdash \Gamma,\ ?_w L}\text{ Affaiblissement}$$

Fig. 2.10 – Modalité affine : présentation monolatérale

$$\frac{\vdash\ ?_c\Gamma, P}{\vdash\ ?_c\Gamma, !_c P}\text{ Promotion} \qquad \frac{\vdash \Gamma, L}{\vdash \Gamma,\ ?_c L}\text{ Déréliction}$$

$$\frac{\vdash \Gamma,\ ?_c L,\ ?_c L}{\vdash \Gamma,\ ?_c L}\text{ Contraction}$$

Fig. 2.11 – Modalité dupliquante : présentation monolatérale

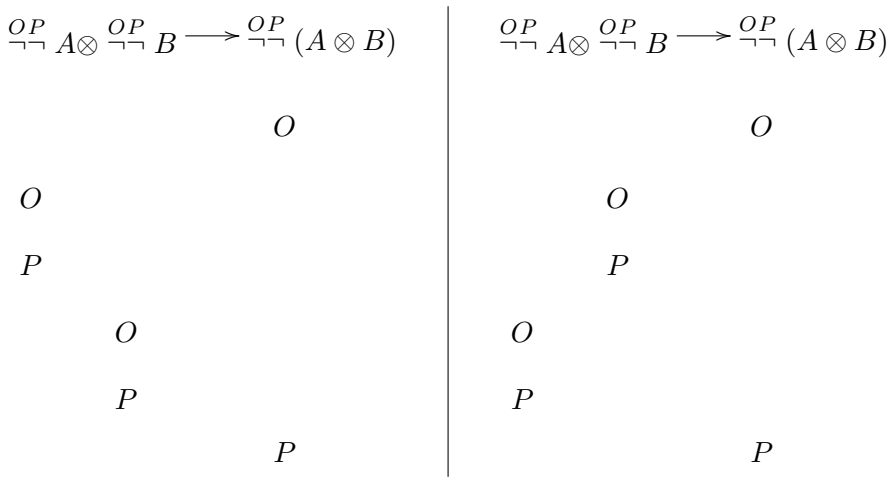


FIG. 2.12 – Description des deux stratégies canoniques de type $\overset{OP}{\neg\neg} A \otimes \overset{OP}{\neg\neg} B \to \overset{OP}{\neg\neg} (A \otimes B)$: celle qui interroge en premier à gauche et celle qui interroge en premier à droite.

2.3.4 Sémantique des jeux et logique linéaire

La monade de continuation $A \mapsto \overset{OP}{\neg\neg} A$ de la sémantique des jeux soulève un jeu A commençant par opposant avec un coup Opposant $\neg_O$ suivi d'un coup Joueur $\neg_P$. Seulement, il s'avère que le problème de Blass décrit par Samson Abramsky dans [Abr03] apparaît précisément car la monade est forte, mais pas commutative [MS, Mel05a]. La Figure 2.12 décrit les deux stratégies canoniques ayant pour type

$$\overset{OP}{\neg\neg} A \otimes \overset{OP}{\neg\neg} B \to \overset{OP}{\neg\neg} (A \otimes B).$$

En revanche, si on identifie les deux stratégies canoniques, on obtient un modèle de logique linéaire propositionnelle – ce qui donne lieu ensuite à un modèle pleinement adéquat (*fully complete* en anglais) de la logique linéaire comme décrit dans [Mel05b].

Cette construction à un joli pendant catégorique. Nous avons déjà mentionné que la catégorie de continuation $\mathcal{C}^\neg$ hérite d'une structure prémonoïdale venant de la structure symétrique monoïdale de la catégorie $\mathcal{C}$. Or, Masahito Hasegawa a montré dans une note non publiée [Has05] la proposition suivante dans le cadre des catégories monoïdales symétriques fermées. Nous donnons ici une extension mineure au cadre des catégories de dialogue.

Proposition 2.9 (Masahito Hasegawa) La catégorie de continuation $\mathcal{C}^\neg$ équipée de cette structure prémonoïdale est $*$-autonome avec pour foncteur de dualisation $\neg$ si et seulement si la monade de continuation est commutative.

Lorsque la monade de continuation est commutative, elle est aussi idempotente. De manière équivalente, on a l'égalité

$$\eta_{TA} = T\eta_A$$

Démonstration : D'après la Proposition 1.51, la catégorie de Kleisli est symétrique monoïdale si et seulement si la monade de continuation est commutative. Il reste à montrer que, dans ce cas, on a gratuitement l'$*$-autonomie. Maintenant, pour tout objet A de $\mathcal{C}$ on peut définir les morphismes

$$A \xrightarrow{\eta} TA \xrightarrow{\eta_{TA}} T^2A \qquad \text{et} \qquad TA \xrightarrow{\mathrm{id}_{TA}} TA$$

qui définissent un isomorphisme entre A et TA dans $\mathcal{C}^\neg$. En effet, on déduit les égalités

$$\begin{array}{rcl} A \xrightarrow{\eta} TA \xrightarrow{\eta_{TA}} T^2A \xrightarrow{T\mathrm{id}_{TA}} T^2A \xrightarrow{\mu_A} TA & = & A \xrightarrow{\eta_A} TA \\ TA \xrightarrow{\mathrm{id}_{TA}} TA \xrightarrow{T\eta} T^2A \xrightarrow{T\eta_{TA}} T^3A \xrightarrow{\mu_{TA}} T^2A & = & TA \xrightarrow{\eta_{TA}} T^2A \end{array}$$

de fait que la monade est idempotente. Comme la monade est idempotente, on sait que $\neg\neg\neg A \cong \neg A$ et on a donc la bijection idempotente

$$\begin{array}{ccccc} A \otimes B & \xrightarrow{f} & T\neg C & \cong & \neg C \\ \hline A & \xrightarrow{\varphi f} & T\neg(B \otimes C) & \cong & \neg(B \otimes C) \end{array}$$

Masahito Hasegawa a montré que lorsque la monade de continuation est commutative, elle est aussi idempotente . ■

Ainsi, $\mathcal{C}^\neg$ définit un modèle de logique linéaire multiplicative. Dans ce cas, $\mathcal{C}^\neg$ définit un modèle de logique linéaire multiplicative. Le lecteur averti reconnaîtra ici une catégorification de la sémantique des phases de Jean-Yves Girard [Gir87]. Quoiqu'il en soit, cela montre que la logique linéaire est essentiellement la logique tensorielle pour laquelle la négation tensorielle est commutative.

Logique linéaire	$\otimes$	est monoïdal
	$\neg$	est commutative

2.3.5 Cas commutatif : Un modèle de logique linéaire

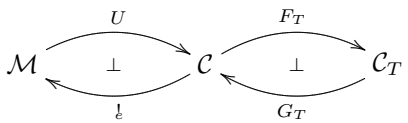
Nous allons maintenant plus loin en montrant qu'un modèle de logique tensorielle – pour lequel la monade de continuation est commutative – induit un modèle de logique linéaire.

Pour obtenir ce résultat, nous allons travailler majoritairement sur la catégorie de Kleisli de la monade de continuation. Ensuite, nous transporterons les structures sur la catégorie de continuation qui n'est, après tout, que sa catégorie opposée.

Nous montrons dans un premier temps que la modalité exponentielle se relève à la catégorie de Kleisli.

Lemme 2.10 La modalité exponentielle $\underset{e}{!}$ se relève en la modalité ! sur la catégorie de Kleisli $\mathcal{C}_T$.

Démonstration : D'après la Proposition 1.51, l'adjoint à gauche F_T entre $\mathcal{C}$ et $\mathcal{C}_T$ est symétrique monoïdal fort. On en déduit grâce à la Proposition 1.54 que l'adjonction est symétrique monoïdale. Comme les adjonctions symétriques monoïdales composent, on obtient une adjonction symétrique monoïdale :



$\mathcal{M}$ étant cartésienne, cette adjonction construit un $\otimes$-comonoïde commutatif

$$!A = \overset{\text{def}}{=} \underset{e}{!}\neg\neg A$$

sur $\mathcal{C}_T$. ■

Nous rappelons maintenant un résultat faisant parti du folklore catégorique : la catégorie de Kleisli préserve les coproduits finis. Plus formellement,

Proposition 2.11 Soit T une monade sur une catégorie $\mathcal{C}$ équipée de coproduits finis. Alors, la catégorie de Kleisli $\mathcal{C}_T$ associée a aussi les coproduits finis. De plus, si les coproduits commutent au tenseur, ils le font toujours dans la catégorie de Kleisli.

Démonstration : Notons $A + B$ le coproduit de A et B. Les injections sont définies par

$$A \xrightarrow{\text{inj}_1} A + B \xrightarrow{\eta_{A+B}} T(A + B) \qquad\qquad B \xrightarrow{\text{inj}_2} A + B \xrightarrow{\eta_{A+B}} T(A + B)$$

Étant donné un objet X deux flèches $f : A \to TX$ et $g : B \to TX$, on définit la somme par

$$A + B \xrightarrow{f+g} (TX$$

On vérifie aisément que cette flèche convient et est universelle.

En ce qui concerne la commutation au tenseur, il suffit de remarquer que la définition du tenseur et des coproduits dans la catégorie $\mathcal{C}_T$ sont donnés par l'image de ceux de $\mathcal{C}$ par le foncteur F_T. Enfin, le morphisme canonique de distributivité

$$(A \otimes B) \oplus (A \otimes C) \;\cong\; A \otimes (B \oplus C)$$

dans $\mathcal{C}_T$ est l'image du morphisme de distributivité dans $\mathcal{C}$ par le foncteur F_T. Il est isomorphisme dans $\mathcal{C}$ donc il est isomorphisme dans $\mathcal{C}_T$ car tout foncteur préserve les isomorphismes. ■

Par dualité, on en déduit que la catégorie de Kleisli $\mathcal{C}_T$ est munie des produits finis définis par

$$A \,\&\, B \;\stackrel{\text{def}}{=}\; \neg(\neg A \oplus \neg B).$$

On peut maintenant exprimer le dernier lemme qui nous manque.

Lemme 2.12 L'isomorphisme

$$!(A \,\&\, B) \;\cong\; !A \;\otimes\; !B$$

est valide dans la catégorie $\mathcal{C}_T$

Démonstration : L'isomorphisme vient du fait que les adjoints à droite (ici $\underset{e}{!}$ et G_T) préservent les produits et que les adjoints à gauche F_T et U sont monoïdaux forts. ■

$\mathcal{C}_T$ définit donc bien un modèle de logique linéaire. C'est évidement aussi le cas pour sa catégorie opposée $\mathcal{C}^\neg$, la seule différence est que dans ce cas, la modalité exponentielle est donnée par

$$!A \;\stackrel{\text{def}}{=}\; \neg\neg\underset{e}{!}A.$$

Théorème 2.13

Soit $\mathcal{C}$ un modèle de logique tensorielle (multiplicative, additive, exponentielle) pour lequel la monade de continuation $\neg\neg$ est commutative. La catégorie de continuation $\mathcal{C}^\neg$ donne un modèle de logique linéaire (multiplicative, additive, exponentielle).

2.3.6 Espaces de cohérence, de finitude et de Köthe

Nous allons maintenant montrer que les espaces de cohérence, de finitude et de Köthe sont obtenus à partir d'une logique tensorielle pour laquelle la monade de continuation est commutative. Il fait parti du folklore dans le milieu de la logique linéaire que les espaces cohérents peuvent être décrit comme certains objets plus primitifs qui sont clos

par biorthogonal. Il s'avère que c'est aussi comme ça que sont construits les espaces de finitude et de Köthe. Pourtant, à notre connaissance, personne n'a décrit en détail la structure catégorique qui régie ces objets plus primitifs. Nous proposons d'y remédier en définissant un modèle de logique tensorielle $\mathbf{G}(\mathbf{Mat})$ obtenue par *recollement* le long du foncteur $\mathbf{Mat}(\mathbf{1}, -)$, où $\mathbf{Mat}$ est la catégorie des matrices à valeurs dans $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$. Cette utilisation du recollement pour les modèles de la logique tensorielle est inspirée du travail de Martin Hyland et Andrea Schalk sur pour la logique linéaire [HS03]. Nous définirons alors deux notions d'orthogonalité (et donc de négation) ; l'une donnant lieu aux espaces de cohérence, l'autre aux espaces de finitude. Les espaces de Köthe sont obtenus en remplaçant $\overline{\mathbb{N}}$ par $\overline{\mathbb{K}}$ pour $\mathbb{K} = \mathbb{R}$ ou C.

Notre construction part d'une structure catégorique importante en informatique ; celle des distributeurs **Dist** (ou modules) à valeurs dans **Ens**. Gian Luca Cattani et Glynn Winskel [CW05] ont récemment montré que cette structure permettait de modéliser finement les processus et les bisimulations entre processus. Elle donne aussi un modèle de logique tensorielle. Un objet $\mathcal{C}$ de **Dist** est une petite catégorie et un morphisme M entre $\mathcal{C}$ et $\mathcal{D}$ est un foncteur

$$M \quad : \quad \mathcal{C} \times \mathcal{D}^{\mathrm{op}} \longrightarrow \mathbf{Ens},$$

noté

$$M \quad : \quad \mathcal{C} \quad \nrightarrow \quad \mathcal{D},$$

et appelé *distributeur* ou *module*. Ceci ne définit pas exactement une catégorie mais plutôt une bicatégorie. Comme nous allons rapidement nous restreindre à la catégorie **Mat**, nous reportons la construction complète de cette bicatégorie au Chapitre 3.

Cette bicatégorie définit un modèle (bicatégorique) de logique tensorielle. Le produit tensoriel est donné par le produit cartésien de catégories et la somme est donnée par le coproduit de catégories. La négation tensorielle est définie par le foncteur $(-)^{\mathrm{op}}$ qui à toute catégorie associe sa catégorie opposée. Les isomorphismes suivants assurent que le foncteur définit bien une négation tensorielle

$$\mathbf{Dist}(\mathcal{C} \times \mathcal{D}, \mathcal{E}^{\mathrm{op}}) \;\cong\; \mathbf{Cat}(\mathcal{C} \times \mathcal{D} \times \mathcal{E}, \mathbf{Ens}) \;\cong\; \mathbf{Dist}(\mathcal{C}, (\mathcal{D} \times \mathcal{E})^{\mathrm{op}}).$$

Il nous reste à donner la modalité exponentielle. Idéalement, nous souhaiterions qu'elle viennent de la construction libre d'une catégorie monoïdale symétrique $\widetilde{S}(\mathcal{C})$ à partir d'une catégorie $\mathcal{C}$. Malheureusement, un monoïde commutatif dans **Cat** est une catégorie monoïdale strictement symétrique, ce qui est un peu moins courant. En utilisant le théorème démontré au Chapitre 3 sur la construction du monoïde commutatif libre, on peut construire un 2-foncteur S sur **Cat** qui à toute catégorie $\mathcal{C}$ associe la catégorie monoïdale strictement symétrique $S(\mathcal{C})$. Cette catégorie est obtenue intuitivement à partir de $\widetilde{S}(\mathcal{C})$ en identifiant les objets $A \otimes B$ et $B \otimes A$. On a d'ailleurs une transformation naturelle α de composante

$$\alpha_{\mathcal{C}} \quad : \quad \widetilde{S}(\mathcal{C}) \longrightarrow S(\mathcal{C}).$$

Cette construction se relève en un bifoncteur sur la bicatégorie **Dist** qui, par dualité dans **Dist**, donne la modalité exponentielle de notre modèle.

Proposition 2.14 La bicatégorie **Dist** définit un modèle (bicatégorique) de la logique tensorielle.

Notre but est de retrouver petit à petit les modèles des espaces de cohérence, espaces de finitude et espaces de Köthe à partir de la bicatégorie **Dist**. On considère dans un premier temps la bicatégorie $\mathbf{Mod}(\mathbf{Ens})$ obtenue en ne gardant de *Dist* que les objets qui sont des ensembles dénombrables (vus comme des catégories discrètes) et les modules

à valeur dans les ensembles dénombrables. Cette sous-bicatégorie est toujours un modèle de la logique tensorielle car elle est close par toutes les constructions susmentionnées. La négation tensorielle devient alors l'identité et la modalité exponentielle devient la construction multiensemble. L'intérêt de cette bicatégorie est qu'on peut la projeter dans la catégorie **Mat** des matrices à valeurs dans $\overline{\mathbb{N}}$ que nous décrivons maintenant.

Définition 2.15 (matrice à valeurs dans $\overline{\mathbb{N}}$)
La catégorie des matrice à valeurs dans $\overline{\mathbb{N}}$ a pour objets X, Y les ensembles dénombrables et pour morphismes $M : X \nrightarrow Y$ les fonctions

$$M \quad : \quad X \times Y \to \overline{\mathbb{N}}.$$

L'identité est définie par

$$\mathrm{id}_X \quad : \quad (x,y) \mapsto \begin{cases} 1 & \text{si } x = y \\ 0 & \text{sinon.} \end{cases}$$

La composition de $M : X \nrightarrow Y$ avec $N : Y \nrightarrow Z$ est obtenue par la formule habituelle de composition des matrices

$$N \circ M = \sum_{y \in Y} M(x,y) * N(y,z)$$

où $+$ et $*$ dénotent l'addition et la multiplication usuelles dans l'anneau $\overline{\mathbb{N}}$ (en particulier $0 * \infty = 0$).

On a un bifoncteur

$$\begin{array}{rcl} \mathbf{Mod}(\mathbf{Ens}) & \longrightarrow & \mathbf{Mat} \\ X & \mapsto & X \\ M : X \nrightarrow Y & \mapsto & (x,y) \mapsto \#(M(x,y)) \end{array}$$

qui est l'identité sur les objets et qui à un module associe une matrice représentant la cardinalité de chaque ensemble définissant le module. Ce foncteur va nous permettre d'importer les structures de **Mod**(**Ens**) à **Mat**.

La multiplication de $\overline{\mathbb{N}}$ induit une structure monoïdale $\otimes$ sur **Mat** comme suit

– Le produit tensoriel de deux objets X et Y de **Mat** est donné par leur produit ensembliste

$$X \otimes Y \stackrel{\text{def}}{=} X \times Y.$$

– Le produit tensoriel de deux matrices $M : X \nrightarrow X'$ et $N : Y \nrightarrow Y'$ est obtenu par

$$M \otimes N(x,y,x',y) \stackrel{\text{def}}{=} M(x,x') * N(y,y').$$

– L'unité du produit tensoriel est donné par l'ensemble singleton $\mathbf{1} = \{\star\}$.

La négation tensorielle est maintenant donnée par le foncteur identité, la somme est toujours donnée par le coproduit. On va maintenant décrire la modalité exponentielle en s'appuyant sur le travail de Ryu Hasegawa [Has02b] dans lequel il définit une exponentielle sur une catégorie de préfaisceaux.

Définition 2.16 (modalité exponentielle dans Mat)
Étant donné un ensemble X de **Mat**, l'exponentiel $!X$ est donné par l'ensemble des multiensembles finis d'éléments de X. La comultiplication $\delta : !X \nrightarrow !X \otimes !X$ correspond à la matrice

$$\delta : !X \times !X \times !X \to \overline{\mathbb{N}} \quad : \quad (w, w', w'') \mapsto \begin{cases} 1 & \text{si } w = w' \cup w'' \\ 0 & \text{sinon.} \end{cases}$$

Étant donnée une matrice $M : X \nrightarrow Y$, la matrice $!M : !X \nrightarrow !Y$ est définie par

$$!M \quad : \quad ([x_1, \cdots, x_n], [y_1, \cdots, y_m]) \mapsto \begin{cases} \displaystyle\sum_{\sigma \in L([x_1, \cdots, x_n])} \prod_{1 \leq i \leq n} M(x_{\sigma(i)}, y_i) & \text{si } n = m \\ 0 \quad \text{sinon.} \end{cases}$$

où $[x_1, \cdots, x_n]$ représente un multiensemble à n éléments de X. La sommation est faite sur le sous-ensemble

$$L([x_1, \cdots, x_n] \stackrel{\text{def}}{=} S_n / \sim$$

des permutations sur $[1, \cdots, n]$ où $\sim$ est la relation d'équivalence

$$\sigma \sim \rho \quad \stackrel{\text{def}}{\iff} \quad \forall\, i,\ x_{\sigma(i)} = x_{\rho(i)}.$$

Proposition 2.17 La catégorie **Mat** définit un modèle de la logique tensorielle.

Pour retrouver les espaces de cohérence et de finitude, on va maintenant construire la catégorie $\mathbf{G}(\mathbf{Mat})$ obtenue par *recollement* le long du foncteur $\mathbf{Mat}(\mathbf{1}, -)$. Martin Hyland et Andrea Schalk [HS03] ont montré que cette construction préserve les modèles de logique linéaire. C'est aussi le cas pour les modèles de logique tensorielle. La preuve donnée par Hyland et Schalk ne nécessite que des adaptations mineures.

Définition 2.18 (recollement le long de Mat(1, −))
Le recollement le long de $\mathbf{Mat}(\mathbf{1}, -)$ est la catégorie $\mathbf{G}(\mathbf{Mat})$ pour laquelle

- un objet est un triplet $(X, X_\bullet, X_\bullet \to \mathbf{Mat}(\mathbf{1}, X))$. Chaque triplet peut être vu comme un ensemble X et un sous-ensemble $X_\bullet$ de matrices de $\mathbf{1}$ dans X, c'est à dire de multiensembles (pas forcément finis) sur X. Nous noterons donc ce triplet simplement $(X, X_\bullet)$ par la suite ;
- un morphisme entre $(X, X_\bullet)$ et $(Y, Y_\bullet)$ est une matrice $M : X \nrightarrow Y$ telle que pour tout u de $X_\bullet$,

$$\mathbf{1} \stackrel{u}{\nrightarrow} X \stackrel{M}{\nrightarrow} Y \in Y_\bullet$$

Remarque 2.19
Pour les espaces de cohérence, l'ensemble $X_\bullet$ d'un objet $(X, X_\bullet)$ représente l'ensemble des cliques. Nous aurions aussi pu définir la catégorie $\mathbf{G}(\mathbf{Mat})$ par double recollement le long des foncteurs $\mathbf{Mat}(\mathbf{1}, -)$ et $\mathbf{Mat}(-, \mathbf{1})$. On travaillerait alors avec des objets de la forme $(X, X_\bullet, {}_\bullet X)$ où $X_\bullet$ représente l'ensemble des cliques et ${}_\bullet X$ représente l'ensemble des anticliques. Comme la catégorie de départ **Mat** est autoduale, nous pouvons nous passer de cette complication.

Remarquons que la catégorie $\mathbf{G}(\mathbf{Mat})$ peut être définie abstraitement comme la catégorie tranche (*comma category* en anglais) sur le foncteur $\mathbf{Mat}(\mathbf{1}, -)$

$$\mathbf{G}(\mathbf{Mat}) \stackrel{\text{def}}{=} (\mathbf{Mat} \downarrow \mathbf{Mat}(\mathbf{1}, -)).$$

On va maintenant utiliser le relèvement automatique du produit tensoriel, du coproduit et de la modalité exponentielle décrit dans [HS03].

Proposition 2.20 La catégorie $\mathbf{G}(\mathbf{Mat})$ est monoïdale symétrique et possède les coproduits finis. L'unité est donnée par le couple $(\mathbf{1}, \{\mathrm{id}_{\mathbf{1}}\})$, que nous noterons abusivement encore $\mathbf{1}$. Le produit tensoriel de deux objets $A = (X, X_\bullet)$ et $B = (Y, Y_\bullet)$ est défini par

$$A \otimes B = (X \times Y, X_\bullet \otimes Y_\bullet)$$

où

$$X_\bullet \otimes Y_\bullet = \{\mathbf{1} \cong \mathbf{1} \otimes \mathbf{1} \stackrel{u \otimes v}{\longmapsto} X \times Y \mid u \in X_\bullet, v \in Y_\bullet\}.$$

Le coproduit de $(X, X_\bullet)$ et $(Y, Y_\bullet)$ est donné par

$$(X, X_\bullet) \oplus (Y, Y_\bullet) \stackrel{\text{def}}{=} (X \uplus Y, \{1 \stackrel{u}{\longmapsto} X \mapsto X + Y \mid u \in X_\bullet\} \cup \{1 \stackrel{v}{\longmapsto} Y \mapsto X + Y \mid v \in Y_\bullet\}).$$

La définition de l'exponentielle est plus délicate et requiert que le foncteur $\mathbf{Mat}(\mathbf{1}, -)$ soit linéairement distributif (avec la fonction identité comme modalité exponentielle sur $\mathbf{Ens}$). Sans rentrer dans les détails, notons que cette propriété repose sur l'existence d'une transformation naturelle

$$\kappa : \mathbf{Mat}(1, -) \longrightarrow \mathbf{Mat}(1, !(-))$$

vérifiant des lois de cohérence avec la structure comonoïdal et comonadique de $!(-)$. Dans notre cadre, chaque composante de cette transformation naturelle est définie par

$$\begin{array}{rcl} \kappa_X \cong \mathbf{Mat}(1, X) & \to & \mathbf{Mat}(1, !X) \\ u & \mapsto & \left(\mu \mapsto \prod\limits_{x \in X} u(x)^{\mu(x)}\right) \end{array}$$

où μ dénote un multiensemble fini sur X vu comme une fonction de X dans $\mathbb{N}$. Cette formule peut être rapprochée du multi-exposant utilisé en théorie des polynômes à plusieurs variables. Martin Hyland et Andrea Schalk ont montré que l'existence de cette transformation naturelle suffit pour définir une modalité exponentielle sur la catégorie $\mathbf{G}(\mathbf{Mat})$.

Définition 2.21 (modalité exponentielle sur G(Mat))
On peut définir une modalité de ressource sur la catégorie $\mathbf{G}(\mathbf{Mat})$ par

$$!(X, X_\bullet, f : X_\bullet \to \mathbf{Mat}(1, X)) = (!X, X_\bullet, X_\bullet \stackrel{f}{\to} \mathbf{Mat}(1, X) \stackrel{\kappa_X}{\longrightarrow} \mathbf{Mat}(1, !X))$$

Nous allons maintenant définir une notion générale d'orthogonalité induite par un ensemble $S \subset \overline{\mathbb{N}}$. Cette notion est appelée *orthogonalité focalisée* dans [HS03]. Lorsqu'on prendra $S = \{0, 1\}$, les objets clos par double négation seront les espaces de cohérence, lorsqu'on prendra $S = \mathbb{N}$, les objets clos par double négation seront les espaces de finitude.

Définition 2.22 (négation par rapport à S)
Soit $S \subset \overline{\mathbb{N}}$. On définit l'orthogonalité focalisée selon S par

$$(\mathbf{1} \overset{u}{\multimap} X) \perp_S (\mathbf{1} \overset{v}{\multimap} X) \overset{\text{def}}{\iff} \mathbf{1} \overset{u}{\multimap} X \overset{v^{\text{op}}}{\multimap} \mathbf{1} \in S$$

$$\overset{\text{def}}{\iff} \sum_{x \in X} u(x) * v(x) \in S.$$

On définit ensuite la *négation focalisée selon S* d'un objet $(X, X_\bullet)$ de $\mathbf{G}(\mathbf{Mat})$ par

$$\neg_S(X, X_\bullet) = (X, X_\bullet^{\perp_S})$$

avec

$$X_\bullet^{\perp_S} \overset{\text{def}}{=} \{v \in \mathbf{Mat}(1, X) \mid \forall u \in X_\bullet, u \perp_S v\}.$$

Proposition 2.23 Quelque soit l'ensemble S choisi, $\neg_S$ défini une négation tensorielle. De plus, la monade de continuation associée est commutative car idempotente.

Démonstration : Un morphisme M de $(X, X_\bullet) \otimes (Y, Y_\bullet) \to \neg_S(Z, Z_\bullet)$ est une matrice

$$M \quad : \quad X \times Y \multimap Z$$

vérifiant

$$\forall (u, v) \in X_\bullet \otimes Y_\bullet,\ \forall w \in Z_\bullet, \quad \sum_z \big(\sum_{x,y} M(x, y, z) * u(x) * v(y) \big) * w(z) \in S$$

tandis qu'un morphisme M' de $(X, X_\bullet) \to \neg_S\big((Y, Y_\bullet) \otimes (Z, Z_\bullet)\big)$ est un module

$$M' \quad : \quad X \times Y \multimap Z$$

vérifiant

$$\forall u \in X_\bullet,\ \forall (v, w) \in Y_\bullet \otimes Z_\bullet, \quad \sum_{y,z} \big(\sum_x M'(x, y, z) * u(x) \big) * v(y) * w(z) \in S.$$

Ces deux conditions sont clairement les mêmes après réorganisation des sommations. L'idempotence de la monade de continuation se montre de la même manière. ■

On peut alors rassembler ces résultats pour obtenir la proposition suivante.

Proposition 2.24 La catégorie $\mathbf{G}(\mathbf{Mat})$ donne un modèle de logique tensorielle équipé de plusieurs négations.

Nous allons maintenant voir qu'en instanciant l'ensemble S définissant la négation focalisée de façon bien choisie, on peut retrouver le modèle des espaces de cohérence et celui des espaces de finitude. Avant cela, il faut modifier un peu la définition de l'exponentielle car dans le cas des espaces de cohérence, elle donne les multicliques finis et non les multiensembles finis.

Définition 2.25 (support)
Le support d'un objet $(X, X_\bullet)$, noté $\text{supp}\,(X, X_\bullet)$, est obtenu en restreignant X aux éléments x qui apparaissent dans au moins un des multiensembles de $X_\bullet$. Cette construction s'étend en un foncteur sur $\mathbf{G}(\mathbf{Mat})$ qui définit une comonade.

La comonade $\mathrm{supp}\,(-)$ définit un foncteur monoïdal strict qui distribue avec l'exponentielle

$$\mathrm{supp}\,(!A) \overset{d_A}{\nrightarrow} !\,\mathrm{supp}\,(A).$$

On déduit la proposition suivante :

Proposition 2.26 La comonade $!^{\mathrm{supp}}$, donnée par

$$!^{\mathrm{supp}}(-) \overset{\mathrm{def}}{=} \mathrm{supp}\,(!-),$$

définit une modalité exponentielle sur $\mathbf{G}(\mathbf{Mat})$.

Espaces de cohérence. Prenons $S = \{0, 1\}$. La sous catégorie de **Mat** des objets clos par $\{0, 1\}$-biorthogonal

$$A = \neg_S \neg_S A$$

définit un modèle de logique linéaire d'après le Théorème 2.13 car la monade de continuation associée à la négation est commutative. Ce modèle donne une version pondérée des espaces de cohérences.

Si on se restreint à la sous catégorie (pleine) des objets $(X, X_\bullet)$ pour lesquels tous les morphismes u de $X_\bullet$ sont à valeurs dans $\{0, 1\}$ (c'est à dire sont des ensembles), on retrouve le modèle habituel des espaces de cohérence. La modalité exponentielle induite redonne bien la construction usuelle des multicliques finies car la restriction du support retire les multiensembles qui ne sont pas des multicliques.

La construction d'espaces de cohérence non uniforme nécessite de distinguer cliques et anticliques en faisant du double recollement. La construction de l'exponentielle devient alors plus délicate.

Espaces de finitude. Prenons maintenant $S = \mathbb{N}$. À nouveau, on obtient un modèle de logique linéaire en considérant la catégorie de continuation associée à la négation $\neg_{\mathbb{N}}$. On retrouve donc les espaces de finitude et R-module (ici $rR = \mathbb{N}$) introduit par Thomas Ehrhard [Ehr05]. Si on veut retrouver le modèle relationnel des espaces de finitude, on doit se restreindre à la sous catégorie (pleine) des objets $(X, X_\bullet$ pour lesquels tous les morphismes u de $X_\bullet$ sont à valeurs dans $\{0, 1\}$ (c'est à dire sont des ensembles), la relation d'orthogonalité s'écrit

$$u \perp_S v \quad \overset{\mathrm{def}}{\iff} \quad u \cap v \text{ est fini.}$$

où on voit une relation $u : \mathbf{1} \nrightarrow X$ comme un sous-ensemble de X. On obtient donc le modèle relationnel des espaces de finitude mais avec une notion de morphisme plus générale que les relations car on compte le nombre de « témoins » lors de la composition. Mais l'orthogonalité garantit justement que ce nombre de témoins est fini. On retrouve donc exactement le modèle des espaces de finitude en plongeant $\overline{\mathbb{N}}$ dans $\{0, 1, +\infty\}$.

Notons que la restriction du support pour l'exponentielle n'a pas d'incidence dans ce modèle car l'ensemble $X_\bullet$ d'un espace de finitude $(X, X_\bullet)$ contient au moins toutes les parties finies de X.

Espaces de Köthe. Pour les espaces de Köthe, la situation est bien évidement un peu différente. Il faut passer de l'anneau $\mathbb{N}$ au corps $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$. On considère alors la catégorie $\mathbf{Mat}_{\mathbb{K}}$ des matrices à valeurs dans $\overline{\mathbb{K}}$. Un objet $(X, X_\bullet)$ de $\mathbf{G}(\mathbf{Mat}_{\mathbb{K}})$ est alors un ensemble au plus dénombrable X et un sous-ensemble $X_\bullet$ de $\mathbb{K}^X$. On peut alors calquer des constructions similaires sur cette nouvelle catégorie de matrices. La négation qui donne lieu aux espaces de Köthe est obtenue en prenant $S = \mathbb{K}$. Le lecteur intéressé pourra trouver tous les détails dans le travail de Thomas Ehrhard [Ehr02].

2.4 Un modèle de jeu avec gain

Dans cette section, nous présentons un modèle sémantique simple de la logique tensorielle. Nous utiliserons pour cela le modèle des jeux de Conway, introduit dans un papier pionnier d'André Joyal [Joy77], que nous allons ensuite enrichir d'une notion de gain afin de pourvoir contrôler la linéarité d'un jeu.

2.4.1 Jeux de Conway

Les jeux de Conway donne un formalisme très intuitif, qui a le mérite de relier directement la sémantique des jeux à des notions algorithmiques standards en utilisant explicitement la structure de graphes.

Définition 2.27 (Jeu de Conway)
Un jeu de Conway A est un graphe orienté enraciné $(V_A\ , E_A\ , \lambda_A\ )$ composé
- d'un ensemble V_A de sommets représentant les *positions* du jeu ;
- d'un ensemble $E_A \subseteq V_A \times V_A$ d'arêtes représentant les *coups* du jeu ;
- d'une fonction $\lambda_A : E_A \to \{-1, +1\}$ indiquant si un coup appartient à Opposant (-1) ou à Joueur $(+1)$.

Nous notons $\star_A$ la racine du graphe sous-jacent. Un jeu de Conway est dit *négatif* (resp. *positif*) lorsque tous les coups partant de la racine sont Opposant (resp. Joueur).

Chemins et Parties. Un *chemin* $m_1 \cdot m_2 \cdot \ldots \cdot m_{k-1} \cdot m_k$ d'un jeu de Conway A est une suite de coups de la forme

$$x_0 \xrightarrow{m_1} x_1 \xrightarrow{m_2} \ldots \xrightarrow{m_{k-1}} x_{k-1} \xrightarrow{m_k} x_k \tag{2.7}$$

On note $s : x \twoheadrightarrow y$ pour indiquer que s a pour position initiale x et position finale y. Deux chemins sont dits *parallèles* lorsqu'ils ont la même position initiale et finale.

Un chemin (2.7) est *alternant* lorsque :

$$\forall i \in \{1, \ldots, k-1\}, \qquad \lambda_A(m_{i+1}) = -\lambda_A(m_i).$$

Une *partie* s est un chemin commençant à la racine $\star_A$ du jeu de Conway A, i.e. $s : \star_A \twoheadrightarrow x$. On note Play_A l'ensemble des parties d'un jeu A.

On dit que $t : x \twoheadrightarrow x'$ est un *sous-chemin* d'une partie $s : \star \twoheadrightarrow y$ s'il existe $u : \star \twoheadrightarrow x$ et $v : x' \twoheadrightarrow y$ tel que $s = u \cdot t \cdot v$. Lorsque u est vide, on dit que t est un *préfixe* de s, noté $t \prec s$.

Stratégies. Le lecteur remarquera que la définition d'un jeu de Conway n'implique pas que les parties qu'il décrit seront alternées. La notion d'alternance entre Joueur et Opposant n'apparaît donc pas sur les jeux (i.e. sur les types), mais plutôt sur les stratégies (i.e. sur les programmes). Ceci correspond à l'intuition qu'un jeu décrit un espace de calcul à priori assez libre tandis qu'une stratégie décrit des exécutions plus réglementées.

Définition 2.28 (stratégie)
Une *stratégie* σ d'un jeu de Conway A est définie par un ensemble non vide de *partie alternante* de longueur paire tel que :

- toute partie non vide commence par un coup Opposant ;
- σ est close par préfixe pair : pour toute partie s et pour tous coups m, n,

$$s \cdot m \cdot n \in \sigma \quad \text{implique} \quad s \in \sigma;$$

- σ est déterministe : pour toute partie s, et pour tous coups m, n, n',

$$s \cdot m \cdot n \in \sigma \quad \text{et} \quad s \cdot m \cdot n' \in \sigma \quad \text{implique} \quad n = n'.$$

Le déterminisme est crucial dès lors que l'on veut calculer la réponse d'une stratégie. En effet, la réponse d'une stratégie à une partie $s \cdot m$ est l'unique coup n – s'il existe – pour lequel $s \cdot m \cdot n \in \sigma$.

Notons au passage que notre notion de stratégie est *partielle* car une stratégie ne doit pas nécessairement répondre à un coup Opposant. Nous écrivons $\sigma : A$ pour indiquer que σ est une stratégie sur le jeu A. Enfin, comme indiqué un peu plus haut, une partie d'un jeu de Conway n'est pas en générale alternée alors qu'une partie jouée par une stratégie l'est toujours.

Dual. Tout jeu de Conway A induit un jeu *dual* A^* obtenu tout simplement en renversant la polarité des coups du jeu A. Plus formellement, on définit $A^* = (V_{A^*}, E_{A^*}, \lambda_{A^*})$ par

- $V_{A^*} = V_A$;
- $E_{A^*} = E_A$;
- $\lambda_{A^*} = -\lambda_A$.

Remarquons que comme les coups du jeu A et de son dual A^* sont identiques, on peut voir un chemin de A comme un chemin de A^* et réciproquement. Nous utiliserons ceci lorsque nous définirons la composition de deux stratégies $\sigma : A^* \otimes B$ et $\tau : B^* \otimes C$ car nous verrons les coups de B joués par σ comme des coups de B^* joués par τ.

Tenseur. Le produit tensoriel $A \otimes B$ de deux jeux de Conway A et B est essentiellement le produit asynchrone de leurs deux graphes sous-jacents. Plus formellement, il peut être défini comme suit :

- ces positions sont les paires (x, y) notées $x \otimes y$ vérifiant $\star_{A\otimes B} = \star_A \otimes \star_B$, c'est à dire

$$V_{A\otimes B} = V_A \times V_B;$$

- ces coups sont de deux sortes différentes

$$x \otimes y \to \begin{cases} z \otimes y \text{ si } x \to z \text{ dans le jeu } A, \\ x \otimes z \text{ si } y \to z \text{ dans le jeu } B. \end{cases}$$

- Les polarités des coups du jeu $A \otimes B$ sont directement héritées de celles des jeux A et B.

Le jeu de Conway 1 composé d'une unique position $\star$ et d'aucun coup est l'élément neutre du produit tensoriel.

Comme on en a l'habitude en sémantique des jeux, toute partie s du jeu $A \otimes B$ peut être vue comme l'entrelacement d'une partie $s_{|A}$ du jeu A et d'une partie $s_{|B}$ du jeu B.

Composition. Nous allons maintenant définir la composition de deux stratégies en utilisant le concept de « parallel and hiding » qui fonctionne aussi bien pour les jeux que pour

des notions plus abstraites de catégories monoïdales tracées libres ou de catégories compactes fermées libres. Dans le cadre de la sémantique des jeux, cette méthode s'implémente en introduisant la notion d'interaction développée dans [McC96, Har00].

On dit qu'une suite de coups u de trois jeux A, B, C est une *interaction* sur A, B, C, ce qui est noté $u \in int_{ABC}$, lorsque la projection de u sur chacun des jeux $A^* \otimes B$, $B^* \otimes C$ et $A^* \otimes C$ est une partie. La notion d'interaction décrit la partie « parallel », maintenant, définissons la composition de stratégies à proprement dit.

Définition 2.29 (composition)
Étant données deux stratégies $\sigma : A^* \otimes B, \tau : B^* \otimes C$, on définit la composition $\tau \circ \sigma$ de ces deux stratégies par l'ensemble des projections sur $A^* \otimes C$ d'interactions sur A, B, C dont les projections sur $A^* \otimes B$ et $B^* \otimes C$ appartiennent à la stratégie σ ou τ. En langage ensembliste, cela donne

$$\tau \circ \sigma = \{u_{|A^* \otimes C} \mid u \in int_{ABC}, u_{|A^* \otimes B} \in \sigma, u_{|B^* \otimes C} \in \tau\}$$

Comme souvent en sémantique des jeux, on montre que la composition de deux stratégies est elle-même une stratégie en passant par le lemme de « témoin unique » qui stipule que la projection $u_{|A^* \otimes C}$ provient d'une unique interaction u.

Lemme 2.30 (Témoin unique) Si σ et τ sont des stratégies de $A^* \otimes B$ et $B^* \otimes C$ respectivement, alors pour tout $s \in \tau \circ \sigma$, il existe un unique $u \in int(A, B, C)$ tel que $s = u_{|A,C}$, $u_{|A,B} \in \sigma$ et $u_{|B,C} \in \tau$.

De plus, si $s \in \tau \circ \sigma$ est un préfixe de $t \in \tau \circ \sigma$, alors le témoin unique de s est préfixe du témoin unique de t.

Démonstration : Si $s \in \tau \circ \sigma$ a deux témoins distincts u_1 et u_2, alors le premier coup m où ils différent est dans B. Si ce coup est Opposant (resp. Joueur) alors la stratégie τ (resp. σ) a violé la condition de déterminisme. ■

La stratégie identité. La *stratégie identité* id_A sur le jeu A est définie comme une variation de la *stratégie d'imitation* (usuellement appelée *copycat* en anglais) du jeu $A^* \otimes A$ décrite par André Joyal dans [Joy77].

Rapidement, la stratégie d'imitation répond à tout coup Opposant dans l'une des deux composantes A^* ou A par le même coup dans la composante duale. Voici maintenant une définition plus formelle.

Définition 2.31 (stratégie identité)
La *stratégie identité* sur un jeu de Conway A est définie par l'ensemble de parties suivant :

$$\mathrm{id}_A \stackrel{\text{def}}{=} \{s \in \mathrm{Play}^{even}_{A_1^* \otimes A_2} \mid \forall t \prec^{even} s \ , \ t_{|A_1} = t_{|A_2}\}$$

où $\mathrm{Play}_{A_1^* \otimes A_2}$ décrit l'ensemble des parties alternantes de $A_1^* \otimes A_2$ commençant par un coup Opposant. Nous utilisons les marqueurs 1 et 2 pour distinguer les deux occurrences du jeu A (i.e. A et A^*), et l'exposant *even* permet de restreindre un opérateur sur des chemins en un opérateur sur des chemins de longueur paire.

La catégorie Conway des jeux Conway. La catégorie **Conway** a pour objets les jeux de Conway, et a pour morphismes $\sigma : A \to B$ les stratégies σ sur $A^* \otimes B$.

Remarquons que cette catégorie est *compacte fermée* (voir la Définition 4.4) avec l'unité $\eta_A : 1 \to A \otimes A^*$ et la counité $\varepsilon_A : A^* \otimes A \to 1$ définies par des stratégies d'imitation. Nous n'insistons pas sur ce point pour le moment mais il sera traité bien plus en détail au Chapitre 4 car il sera au cœur de la définition d'une trace nécessaire à l'interprétation des références. La seule chose à retenir pour le moment est que de cette structure compacte fermée découle automatiquement un opérateur de clôture qui est défini par $A^* \otimes B$.

2.4.2 Jeux de Conway à gain

Avant d'introduire la notion de gain, nous devons restreindre la catégorie aux jeux négatifs. En effet, si nous restons dans un cadre entièrement autodual, les contraintes à imposer pour avoir la compositionalité des stratégies en présence de gain sont assez subtiles. En particulier, elles ne peuvent être entièrement positionnelles. Nous reportons cette étude au Chapitre 4, où une analyse des politiques de ressource nous mènera à une axiomatique de la notion de gain basée sur des contraintes satisfaites pour tous sous-chemins d'une partie et non plus seulement sur les positions atteintes. Nous préférons pour le moment donner un modèle positionnel où le parenthésage est absent mais qui est suffisant pour interpréter les différentes modalités de la logique tensorielle.

La sous-catégorie pleine des jeux de Conway négatifs $\mathcal{N}$ n'est pas compacte fermée mais hérite de la clôture de **Conway**. Étant donné un jeu de Conway A, on note A^- le jeu de Conway négatif obtenu en enlevant tous les coups Joueur partant de la racine.

La construction A^- s'étend à un foncteur plein et fidèle de **Conway** vers $\mathcal{N}$ qui est un rétracte du foncteur d'inclusion. Cela suffit pour exporter la fermeture de la catégorie **Conway** à un fermeture dans la catégorie $\mathcal{N}$ en définissant

$$A \multimap B \quad \stackrel{\text{def}}{=} \quad (A^* \otimes B)^-$$

comme le montre la proposition suivante.

Proposition 2.32 (transport de fermeture) Soient $(\mathcal{C}, \otimes, \multimap)$ une catégorie monoïdale symétrique fermée et $(\mathcal{D}, \bullet)$ une catégorie monoïdale symétrique. Supposons qu'il existe une adjonction monoïdale $U \dashv F : \mathcal{D} \to \mathcal{C}$ où U est à la fois plein et fidèle. Alors, on peut exporter la fermeture sur $\mathcal{C}$ en une fermeture sur $\mathcal{D}$ en définissant pour A, B dans $\mathcal{D}$:

$$A \twoheadrightarrow B = F(U(A) \multimap U(B))$$

Notons que la plupart du temps, le foncteur U sera le foncteur d'inclusion et F forcera le fermeture de C à vivre dans la sous-catégorie D.

Démonstration : Comme l'adjonction est monoïdale, le foncteur U est nécessairement monoïdal fort. La fermeture se déduit de la séquence de bijections suivante :

$$\begin{array}{lcll}
\mathcal{D}(B, A \twoheadrightarrow C) & \cong & \mathcal{D}(B, F(U(A) \multimap U(C))) & \\
& \cong & \mathcal{C}(U(B), U(A) \multimap U(C)) & \text{adjonction } U \dashv F \\
& \cong & \mathcal{C}(U(A) \otimes U(B), U(C)) & \text{fermeture de } \mathcal{C} \\
& \cong & \mathcal{C}(U(A \otimes B), U(C)) & U \text{ monoïdal fort} \\
& \cong & \mathcal{D}(A \bullet B, C) & U \text{ plein et fidèle}
\end{array}$$

■

Proposition 2.33 La catégorie $\mathcal{N}$ est symétrique monoïdale fermée.

Nous utilisons cette catégorie comme nouvelle base pour définir la notion de gain.

Définition 2.34 (Jeu de Conway à gain)
Un *jeu de Conway à gain* est un jeu de négatif Conway $A = (V_A, E_A, \lambda_A)$ muni d'une fonction de gain positionnelle

$$\gamma_A : V_A \to \{-1, 0, +1\}.$$

Une position x est dite gagnante si $\gamma_A(x) \in \{0, +1\}$.

Intuitivement, la valeur -1 désigne une position gagnante pour Opposant, la valeur $+1$ désigne une position gagnante pour Joueur, et 0 une position « neutre ».

Produit tensoriel et clôture. Il nous faut à présent étendre la fonction de gain au produit tensoriel $\otimes$ et à l'opérateur de clôture $\multimap$. Comme le gain est positionnel, il suffit pour cela de donner leur « table de vérité » (voir Table 2.3). Ces tables de vérité

$\otimes$	-1	0	$+1$
-1	-1	-1	-1
0	-1	0	$+1$
$+1$	-1	$+1$	$+1$

$\multimap$	-1	0	$+1$
-1	$+1$	$+1$	$+1$
0	-1	0	$+1$
$+1$	-1	-1	$+1$

TAB. 2.3 – « Tables de vérité » du produit tensoriel et de la clôture

sont obtenues en ayant à l'esprit que $\otimes$ est moralement la conjonction booléenne $\wedge$, $\multimap$ correspond à l'implication booléenne $\Rightarrow$, -1 signifie faux, $+1$ signifie vrai et 0 ne change jamais la polarité. Ainsi, le jeu de Conway à gain $A \otimes B$ est défini comme le jeu de Conway $A \otimes B$ sous-jacent, muni de la fonction de gain

$$\gamma_{A\otimes B}(x \otimes y) = \gamma_A(x) \otimes \gamma_B(y);$$

et le jeu de Conway à gain $A \multimap B$ est défini comme le jeu de Conway $A \multimap B$ sous-jacent, muni de la fonction de gain

$$\gamma_{A\multimap B}(x \multimap y) = \gamma_A(x) \multimap \gamma_B(y).$$

On donne 0 à la polarité de l'unique position du jeu 1.

Stratégies gagnantes. Avec la notion de stratégie générale, tous les jeux négatifs ont une unique flèche dans 1. Nous allons utiliser le gain pour définir des stratégies gagnantes, ce qui nous permettra ensuite de distinguer les *jeux affines* – dont la polarité de la racine est 0 –, des *jeux linéaires* – dont la polarité de la racine est $+1$.

Définition 2.35 (stratégie gagnante)
Une stratégie σ sur le jeu de Conway à gain A est dite *gagnante* si toutes les parties qu'elle joue mènent à des position gagnantes, c'est-à-dire dont la polarité vaut 0 ou $+1$:

$$\text{pour tout } s : x \twoheadrightarrow y, \quad s \in \sigma \quad \text{implique} \quad \gamma_A(s) \in \{0, +1\}$$

Notre but est de définir une catégorie des jeux de Conway à gain dont les morphismes du jeu A vers le jeu B sont les stratégies gagnantes de $A \multimap B$. Remarquons que la définition du gain implique qu'il n'y a aucune stratégie gagnante de $A \multimap B$ lorsque $\gamma_A(\star_A) \multimap \gamma_B(\star_B) = -1$. Ceci est le cas en particulier lorsque le jeu A est linéaire et que le jeu B est affine. Avant de définir une catégorie, il nous faut maintenant nous assurer que les stratégies gagnantes composent.

Proposition 2.36 Si $\sigma : A \multimap B$ et $\tau : B \multimap C$ sont des stratégies gagnantes, alors $\tau \circ \sigma : A \multimap C$ est une stratégie gagnante.

Démonstration : On sait déjà que les stratégies composent, il faut uniquement vérifier la condition de gain. Comme celui-ci est défini de manière positionnelle, il suffit de s'assurer par une étude de cas que la composée de deux positions gagnantes sur $\multimap$ est gagnante, au sens où

$$\frac{\begin{array}{ll}\gamma_A(x) \multimap \gamma_B(y) \in \{0,+1\} & (x \multimap y : \text{gagnante}) \\ \gamma_B(y) \multimap \gamma_C(z) \in \{0,+1\} & (y \multimap z : \text{gagnante})\end{array}}{\gamma_A(x) \multimap \gamma_C(z) \in \{0,+1\} \qquad (x \multimap z : \text{gagnante})}$$

Ceci marche bien car la définition du gain sur $\multimap$ vient de l'implication booléenne $\Rightarrow$ qui est elle-même stable par composition. ■

Proposition 2.37 (catégorie des jeux de Conway à gain) La catégorie ayant pour objet les jeux de Conway à gain et pour morphismes de A vers B les stratégies gagnantes de $A \multimap B$ est monoïdale symétrique fermée.

Démonstration : Nous savons déjà que les jeux de Conway négatifs avec les stratégies de $A \multimap B$ comme morphismes définissent une catégorie monoïdale symétrique fermée. Il reste à vérifier que

$$(\gamma_A(x) \otimes \gamma_B(y)) \multimap \gamma_C(z) = \gamma_A(x) \multimap (\gamma_B(y) \multimap \gamma_C(z))$$

pour toutes positions $x \in V_A$, $y \in V_B$ et $z \in V_C$. Cette équation se déduit en partie de la validité de la formule booléenne $(A \wedge B) \Rightarrow C \equiv A \Rightarrow (B \Rightarrow C)$ ■

Afin de pouvoir définir une modalité exponentielle sur notre catégorie, nous devons nous restreindre aux jeux dont la racine est gagnante. À présent, nous travaillerons avec la catégorie $\mathcal{G}$ qui a pour objets les jeux de Conway à gain dont la racine est gagnante et qui a pour morphismes les stratégies gagnantes entre de tels jeux. Nous appellerons ces jeux simplement des *jeux à gain*. Remarquons que la catégorie $\mathcal{G}$ est monoïdale symétrique mais n'est plus close car $\multimap$ ne préserve pas la propriété d'avoir une racine gagnante. Nous allons néanmoins pouvoir définir une négation tensorielle.

Négation tensorielle. Soit $\perp$ le jeu linéaire définit par
- deux positions $\star_\perp$ et x ;,
- un unique coup Opposant $m_\perp : \star_\perp \twoheadrightarrow x$;
- $\gamma_\perp(\star_\perp) = +1$ et $\gamma_\perp(x) = 0$.

Comme indiqué dans l'Exemple 2.3, toute catégorie monoïdale symétrique fermée engendre une négation tensorielle. La catégorie $\mathcal{G}$ n'est pas fermée mais elle hérite néanmoins des propriétés de fermeture de $\multimap$ lorsque celle-ci est définie. Nous pouvons donc en déduire une négation.

Définition 2.38 (négation tensorielle)
La négation tensorielle d'un jeu à gain A est définie par

$$\neg A \stackrel{\text{def}}{=} A \multimap \bot.$$

2.4.3 Modalités affine, dupliquante et exponentielle

Modalité affine. Nous avons mentionné qu'un jeu de Conway à gain A est dit *affine* lorsque sa racine à un gain nul. L'unique stratégie de A dans 1, que nous noterons t_A, est alors gagnante. On note $\mathcal{G}_w$ la sous-catégorie des jeux affines.

Le jeu affine $!_w A$ associé à un jeu à gain A est défini en forçant la racine du jeu A à être neutre. Cette modalité s'étend à une stratégie $\sigma : A \multimap B$ en posant $!_w\sigma = \sigma$. Cette définition est valide car le fait de rendre la racine neutre pour A et B ne fait qu'augmenter les nombres de positions gagnantes de $A \multimap B$.

Lemme 2.39 Soient A un jeu à gain affine et B un jeu à gain. Toute position $x \multimap y$ de $A \multimap B$ (ou indifféremment de $A \multimap !_w B$) vérifie

$$\gamma_A(x) \multimap \gamma_B(y) \in \{0, +1\} \quad \text{ssi} \quad \gamma_A(x) \multimap \gamma_{!B}(y) \in \{0, +1\}$$

Démonstration : Si y n'est pas la racine de B, y a le même gain dans les deux jeux B et $!_w B$. Il suffit donc de regarder le cas où $y = \star_B$. Dans ce cas, $x = \star_A$ car Opposant doit jouer son premier coup dans B (les jeux sont négatifs). Comme A est un jeu affine, on en déduit que

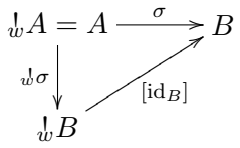
$$\begin{cases} \gamma_A(x) \multimap \gamma_B(y) = \gamma_B(\star_B) \in \{0, +1\} & \text{car la racine est gagnante} \\ \gamma_A(x) \multimap \gamma_{!B}(y) = \gamma_{!B}(\star_B) = 0 & \text{car } !_w B \text{ est affine.} \end{cases}$$

■

Proposition 2.40 Le foncteur $!_w : \mathcal{G} \to \mathcal{G}_w$ qui à A associe $!_w A$ définit une *modalité de ressource affine* sur $\mathcal{G}$.

Démonstration : La catégorie des jeux affines $\mathcal{G}_w$ est une sous-catégorie monoïdale symétrique de $\mathcal{G}$ ayant 1 pour objet terminal. Le foncteur de retour de $\mathcal{G}_w$ vers $\mathcal{G}$ est simplement le foncteur d'inclusion. Reste à définir la bijection entre $\mathcal{G}_w(A, !_w B)$ et $\mathcal{G}(A, B)$ pour un jeu affine A et un jeu à gain B. Or, le Lemme 2.39 indique qu'une stratégie est gagnante sur $A^* \otimes B$ si et seulement elle l'est sur $A^* \otimes !_w B$. Il suffit donc de prendre l'identité pour la bijection. ■

Remarquons au passage que cette modalité est libre. En effet, en notant $[\mathrm{id}_B] : !_w B \to B$ la stratégie d'imitation qui possède les mêmes parties que l'identité sur B, on sait que pour toute stratégie $\sigma : A \to B$ où A est un jeu affine, le diagramme suivant commute de manière unique pour la stratégie $!_w\sigma$:



Modalité dupliquante. Tout jeu à gain A induit un jeu dupliquable $!_c A$ – qui peut être regardé comme le tenseur infini de A – comme suit

- ses positions sont les mots $w = x_1 \cdots x_k$ dont les lettres sont des positions x_i du jeu A différentes de la racine ; l'intuition est que la lettre x_i décrit la position courante dans la i-ème copie de A ;
- sa racine $\star_{!_c A}$ est le mot vide ;
- ses coups $m : w \to w'$ sont ou bien des coups joués dans une copie :

$$w_1 \cdot x \cdot w_2 \xrightarrow{m} w_1 \cdot y \cdot w_2$$

où $\underline{m} : x \to y$ est un coup du jeu A ; ou bien des coups où Opposant ouvre une nouvelle copie :

$$w \xrightarrow{m} w \cdot x$$

où $\underline{m} : \star_A \to x$ est un coup initial de A. En particulier, si m est un coup de $!_c A$, on note $\underline{m}$ le coup sous-jacent dans A ;

- sa fonction de gain sur une position $w = x_1 \cdots x_k$ est obtenue par

$$\gamma_{!_c A}(w) = \bigotimes_{1 \leq i \leq n} \gamma_A(x_i)$$

La polarité des coups est directement héritée de celle des coups du jeu A de la même manière que pour le tenseur.

Nous allons maintenant décrire la stratégie diagonale

$$\delta_A \quad : \quad !_c A \longrightarrow !_c A \otimes !_c A$$

qui fait de $!_c$ une modalité dupliquante sur $\mathcal{G}$. En première approximation, δ_A est une stratégie d'imitation avec une gestion des copies. Lorsque Opposant ouvre une nouvelle copie dans $!_c A \otimes !_c A$, la stratégie δ_A répond en ouvrant une nouvelle copie dans $!_c A$. Cette copie est ensuite reliée à la copie ouverte par Opposant, et cela jusqu'à la fin de l'interaction. Par conséquent, lorsque Opposant joue un coup dans une copie déjà ouverte, δ_A répond par le même coup dans la copie qui lui est reliée. Plus formellement, on définit, par induction sur le nombre de coups joués, l'*entrelacement* d'une partie s de $!_c A \otimes !_c A$ comme la partie $\langle s \rangle$ de $!_c A$ qui imite les coups de s. La partie vide de $!_c A \otimes !_c A$ est traduit sans surprise par la partie vide de $!_c A$, et une partie $s \cdot m$ est traduite par

$$\langle s \cdot m \rangle \stackrel{\text{def}}{=} \langle s \rangle \cdot m'$$

où m' est un coup de $!_c A$ ayant le même coup sous-jacent que m dans A

$$\underline{m'} \stackrel{\text{def}}{=} \underline{m}$$

et joué dans la copie correspondante si $\underline{m}$ n'est pas initial dans A, ou dans une nouvelle copie si $\underline{m}$ est initial dans A. En utilisant cette fonction d'entrelacement, on peut isoler aisément l'ensemble des parties décrivant la stratégie δ_A :

$$\delta_A \stackrel{\text{def}}{=} \{s \in \mathrm{Play}^{even}_{!_c A_1 \multimap (!_c A_2 \otimes !_c A_3)} \mid \forall t \prec^{even} s \ , \ t_{|!_c A_1} = \langle t_{|!_c A_2 \otimes !_c A_3} \rangle\}.$$

À nouveau, les marqueurs 1, 2 et 3 servent à distinguer les trois occurrences du jeu A et l'exposant *even* permet de restreindre un opérateur sur des chemins en un opérateur sur des chemins de longueur paire. Cette stratégie est gagnante car les positions à gauche et à droite du $\multimap$ à toujours la même polarité. Il est facile de voir que δ_A satisfait les deux

diagrammes de cohérence 2.5 et 2.6 et définie ainsi une diagonale sur $!_c A$. Cette modalité s'étend à une stratégie gagnante $\sigma : A \multimap B$ en posant

$$!_c\, \sigma \stackrel{\text{def}}{=} \{(x_1 \cdots x_k) \multimap (y_1 \cdots y_k) \mid \forall 1 \leq i \leq k, x_i \multimap y_i \in \sigma\}$$

qui vit bien dans la catégorie $\mathcal{G}$ comme l'indique le lemme suivant :

Lemme 2.41 Si σ est une stratégie gagnante, alors la stratégie $!_c\, \sigma$ est gagnante.

Démonstration : Il suffit d'observer que pour toutes positions $x \in V_A$, $y \in V_B$, $z \in V_C$ et $t \in V_D$ telles que

$$\gamma_{(A \multimap B) \otimes (C \multimap D)}((x \multimap y) \otimes (z \multimap t)) \in \{0, +1\},$$

on a bien

$$\gamma_{(A \otimes C) \multimap (B \otimes D)}((x \otimes z) \multimap (y \otimes t)) \in \{0, +1\}.$$

■

Nous soutenons maintenant que $!_c$ défini une modalité dupliquante sur $\mathcal{G}$. On note $\mathcal{G}_c$ la catégorie des objets munis d'une diagonale et des stratégies gagnantes préservant cette diagonale.

Proposition 2.42 Le foncteur $!_c : \mathcal{G} \to \mathcal{G}_c$ qui à A associe le jeu dupliquable $!_c A$ défini une *modalité de ressource dupliquante* sur $\mathcal{G}$.

Démonstration : Pour définir une adjonction entre $\mathcal{G}_c$ et $\mathcal{G}$, il suffit de montrer que $!_c$ définit une comonade sur $\mathcal{G}$ car le foncteur adjoint à $!_c$ est le foncteur d'inclusion. La counité ε_A et la comultiplication δ_A de la comonade :

$$\varepsilon_A \;:\; !_c A \longrightarrow A \qquad\qquad \delta_A \;:\; !_c A \longrightarrow !_c !_c A$$

sont définies comme pour la diagonale – en définissant en premier lieu deux fonction d'entrelacement des parties de $!_c A$ vers les parties de A et $!_c !_c A$, puis en définissant une stratégie d'imitation. Étant donnée une partie $s \cdot m$ de $!_c A$, on définit

$$\langle s \cdot m \rangle_\epsilon = \langle s \rangle_\epsilon \cdot \underline{m} \qquad\qquad \langle s \cdot m \rangle_\delta = \langle s \rangle_\delta \cdot m'$$

où m' est un coup de $!_c !_c A$ ayant le même coup sous-jacent $\underline{m}$, joué dans la copie correspondante si $\underline{m}$ n'est pas initial dans A, et joué dans une nouvelle copie si $\underline{m}$ est initial dans A.

$$\begin{array}{rcll} \varepsilon_A & \stackrel{\text{def}}{=} & \{s \in \text{Play}^{even}_{!_c A_1 \multimap A_2} & \mid \forall t \prec^{even} s\ ,\ t_{|!_c A_1} = \langle t_{|A_2} \rangle_\epsilon\} \\ \delta_A & \stackrel{\text{def}}{=} & \{s \in \text{Play}^{even}_{!_c A_1 \multimap !_c !_c A_2} & \mid \forall t \prec^{even} s\ ,\ t_{|!_c A_1} = \langle t_{|!_c !_c A_2} \rangle_\delta\} \end{array}$$

Il est aisé de vérifier que les diagrammes de cohérence d'une comonade sont vérifiés. ■

On peut maintenant se demander si cette comonade définit l'objet dupliquable libre sur la catégorie $\mathcal{G}$ au sens où toute flèche $D \xrightarrow{f} A$ d'un jeu dupliquable D vers un jeu A peut être factorisée de manière unique comme $D \xrightarrow{f^\dagger} !_c A \xrightarrow{\varepsilon_A} A$, avec $f^\dagger$ une stratégie préservant la diagonale. Malheureusement, ceci n'est pas possible en général comme on peut le voir en considérant le jeu

$$\text{Unit} \stackrel{\text{def}}{=} !_w(\bot \multimap \bot).$$

Ce jeu est affine et on peut définir une stratégie $com : 1 \to \text{Unit}$ qui répond à l'unique coup Opposant par l'unique coup Joueur. Ainsi,

$$\text{Unit} \xrightarrow{t_{\text{Unit}}} 1 \cong 1 \otimes 1 \xrightarrow{com \otimes com} \text{Unit} \otimes \text{Unit}$$

rend le jeu Unit dupliquable. Pourtant, toute stratégie σ : Unit $\to$ $!_c A$ qui préserve la diagonale ne peut pas jouer dans Unit comme l'exprime le diagramme suivant

$$\begin{array}{ccccc} \text{Unit} & \xrightarrow{t_A} & 1 \cong 1 \otimes 1 & \xrightarrow{com \otimes com} & \text{Unit} \otimes \text{Unit} \\ {\scriptstyle\sigma}\downarrow & & & & \downarrow{\scriptstyle\sigma\otimes\sigma} \\ !_c A & & \xrightarrow{\Delta_A} & & !_c A \otimes !_c A \end{array}$$

On en déduit qu'il est uniquement possible de factoriser des stratégies de la forme

$$\text{Unit} \xrightarrow{t_{\text{Unit}}} 1 \xrightarrow{f} A$$

Par exemple, il est impossible de factoriser l'identité sur Unit. La modalité de duplication n'est donc pas libre. Mais ceci est dû uniquement à l'absence de cohérence entre la duplication et l'effacement du jeu Unit. Ceci permet à tout jeu affine de se comporter comme un jeu dupliquable sans implémenter une « vraie » duplication. Ce type de duplication feinte va être rejeté en considérant les comonoïdes commutatifs au lieu des objets dupliquables.

Modalité exponentielle. Le jeu comonoïdal $!_e A$ associé à un jeu à gain A est obtenu en appliquant successivement les modalités affine et dupliquante

$$!_e A \stackrel{\text{def}}{=} !_c !_w A = !_w !_c A.$$

L'ordre d'application n'importe pas car les deux comonades commutent. On vérifie facilement que le jeu $!_e A$ vit dans la catégorie cartésienne $\mathcal{G}_e$ des comonoïdes commutatifs sur $\mathcal{G}$.

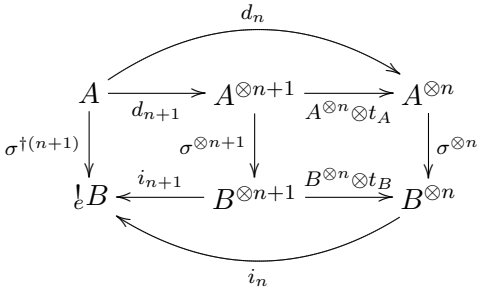
Proposition 2.43 Le foncteur $!_e : \mathcal{G} \to \mathcal{G}_e$ qui à A associe le jeu comonoïdal $!_e A$ définit une *modalité de ressource exponentielle* sur $\mathcal{G}$.

Démonstration : La commutation de $!_w$ et $!_c$ induit une loi distributive entre ces deux comonades, faisant automatiquement de $!_e$ une comonade sur $\mathcal{G}$. ■

Nous allons montrer que cette comonade définit le comonoïde commutatif libre sur $\mathcal{G}$. On suppose – sans perte de généralité car la modalité affine est libre – que le jeu B est affine. Introduisons la stratégie $i_n : B^{\otimes n} \to !_e B$ qui imite Opposant sur $B^{\otimes n}$ pour les n premières copies de $!_e B$, et qui ne répond pas lorsque Opposant ouvre la $n+1$-ème copie de $!_e B$. Soit $\sigma : A \to B$ une stratégie d'un comonoïde commutatif A – dont la counité est notée t_A et la comultiplication d – vers un jeu à gain B. On définit pour tout n la stratégie $\sigma^{\dagger(n)}$ par le diagramme commutatif suivant :

$$\begin{array}{ccc} A & \xrightarrow{d_n} & A^{\otimes n} \\ {\scriptstyle\sigma^{\dagger(n)}}\downarrow & & \downarrow{\scriptstyle\sigma^{\otimes n}} \\ !_e B & \xleftarrow{i_n} & B^{\otimes n} \end{array}$$

où $d_0 = t_A$, d_1 est l'identité sur A et d_n correspond à la comultiplication d composée $n-1$ fois avec elle-même. Considérons maintenant le diagramme suivant :



Il commute sur toutes ces faces excepté pour celle du bas qui vérifie $i_n \circ B^{\otimes n} \otimes t_B \subseteq i_{n+1}$. Le chemin extérieur dans le sens horaire étant égal à $\sigma^{\dagger(n)}$, on en déduit que

$$\sigma^{\dagger(n)} \subseteq \sigma^{\dagger(n+1)}.$$

On peut donc définir le soulèvement comonoïdal $\sigma^\dagger$ par la limite croissante des $\sigma^{\dagger(n)}$:

$$\sigma^\dagger \quad \stackrel{\text{def}}{=} \quad \bigcup_n \sigma^{\dagger(n)}$$

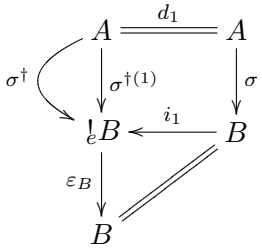
Nous allons maintenant montrer que $\sigma^\dagger$ est comonoïdale. On a la chaîne d'inclusion

$$\delta_B \circ \sigma^{\dagger(n)} \ \subseteq \ (\sigma^{\dagger(n)} \otimes \sigma^{\dagger(n)}) \circ d \ \subseteq \ \delta_B \circ \sigma^{\dagger(2n)}.$$

En prenant la limite de cette chaîne d'inclusion, on obtient :

$$\delta_B \circ \sigma^\dagger \ \subseteq \ (\sigma^\dagger \otimes \sigma^\dagger) \circ d \ \subseteq \ \delta_B \circ \sigma^\dagger,$$

ce qui donne l'égalité requise. Remarquons que $!_e B \xrightarrow{\varepsilon_B^{\otimes n} \circ \delta_B^n} B^{\otimes n}$ coégalise les deux stratégies $\sigma^\dagger$ et $\sigma^{\dagger(n)}$, où – comme pour d_n – on pose $\delta_B^0 = t_{!_e B}$, δ_B^1 est l'identité et δ_B^n correspond à la comultiplication δ_B composée $n-1$ fois avec elle-même. Utiliser cette remarque pour $n = 1$ donne le diagramme

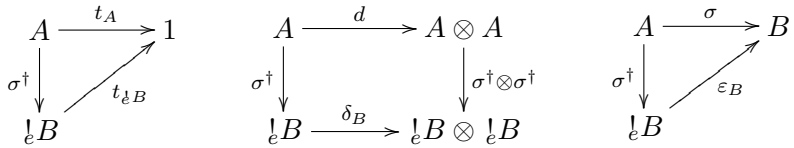


indiquant que l'on a bien factorisé σ par $\sigma^\dagger$:

$$\varepsilon_B \circ \sigma^\dagger = \sigma.$$

Il reste à montrer l'unicité de ce soulèvement.

Proposition 2.44 Étant donnée une stratégie $\sigma : A \to B$ d'un comonoïde commutatif A vers un jeu à gain B, $\sigma^\dagger : A \to !_e B$ est l'unique stratégie satisfaisant :



Démonstration : Nous avons déjà vu que $\sigma^\dagger$ était un bon candidat. Il reste à montrer que c'est le seul. Prenons pour cela un autre candidat τ. On définit $\tau^{(n)}$ comme la sous-stratégie de τ qui joue seulement sur les n premières copies de $!_e B$, c'est-à-dire

$$\tau^{(n)} = i_n \circ (\varepsilon_B^{\otimes n} \circ \delta_B^n) \circ \tau \quad \text{et} \quad \tau = \bigcup_n \tau^{(n)}.$$

Considérons maintenant le diagramme commutatif

$$\begin{array}{ccccc} A & \xrightarrow{d_n} & A^{\otimes n} & = & A^{\otimes n} \\ \downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau^{\otimes n}} & & \downarrow{\scriptstyle \sigma^{\otimes n}} \\ !_e B & \xrightarrow[\delta_B^n]{} & !_e B^{\otimes n} & \xrightarrow[\varepsilon_B^{\otimes n}]{} & B^{\otimes n} \end{array}$$

Ajouter i_n à la fin de ce diagramme donne directement l'égalité

$$\tau^{(n)} = \sigma^{\dagger(n)}$$

ce qui signifie que

$$\tau = \sigma^\dagger.$$

■

2.4.4 Un modèle de logique tensorielle

Pour avoir un modèle de la logique tensorielle, il ne nous manque que les additifs. Malheureusement, la catégorie $\mathcal{G}$ n'a pas les coproduits. Nous allons donc utiliser la *construction des familles* pour les ajouter librement à notre catégorie.

Produits Finis. Pour que la construction de famille se marie bien avec la négation tensorielle, on a besoin que notre catégorie ait les produits finis. Étant donnée une famille $(A_i)_{i\in I}$ de jeux à gain indexé sur un ensemble fini I, le produit $\&_{i\in I} A_i$ est défini par

- son graphe sous-jacent est obtenu prenant l'union disjointe des graphes sous-jacents aux A_i en fusionnant les racines ;
- la polarité des coups est héritée directement de celle des coups des A_i ;
- le gain des positions est hérité directement de celui des positions des A_i, sauf pour la racine qui a pour gain $+1$ si toutes les racines $\star_{A_i}$ ont pour gain $+1$ et 0 sinon. En particulier, le jeu $\&_{i\in I} A_i$ est affine dès que l'un des jeux A_i est affine.

La i-ème projection est donnée par la stratégie d'imitation sur le jeux A_i.

Coproduits finis libres. Construisons maintenant la complétion libre par coproduits finis – notée $Fam(\mathcal{G})$ – de la catégorie $\mathcal{G}$ [AM98].

Définition 2.45 (complétion libre par coproduits finis)
Étant donnée une catégorie $\mathcal{C}$, sa complétion libre par coproduits finis $Fam(\mathcal{C})$ a comme objets les familles $\{A_i | i \in I\}$, où I est un ensemble fini. Un morphisme de $\{A_i \mid i \in I\}$ vers $\{B_j \mid j \in J\}$ consiste en une fonction de réindexation $f : I \to J$ ainsi qu'en une famille de morphismes $\{f_i : A_i \to B_{f(i)} | i \in I\}$ de $\mathcal{C}$.

Le coproduit de $Fam(\mathcal{C})$ est simplement donné par l'union disjointe de deux familles. Cette construction s'étend à une 2-monade. Martin Hyland et John Power donne cette

2-monade comme exemple de 2-monade pseudo-commutative symétrique dans leur papier [HP02]. Ils en déduisent qu'elle distribue avec la 2-monade des catégories monoïdales symétriques. Par conséquent, (a) la catégorie $Fam(\mathcal{C})$ hérite de la structure monoïdale symétrique de $\mathcal{C}$, (b) le coproduit de $Fam(\mathcal{C})$ distribue avec le produit tensoriel, et (c) Fam préserve les adjonctions monoïdales.

Samson Abramsky et Guy McCusker [AM98] ont montré que la construction des familles préservait aussi les produits et l'existence d'un objet terminal. On en déduit qu'elle préserve les modalités affine, dupliquante et exponentielle et que $Fam(\mathcal{C})$ hérite des produits de $\mathcal{C}$. Plus prosaïquement, le produit tensoriel et le produit de deux familles $A = \{A_i | i \in I\}$ et $B = \{B_j | j \in J\}$ sont définis par

$$\begin{array}{rcl} A \otimes B & \stackrel{\text{def}}{=} & \{A_i \otimes B_j \mid (i,j) \in I \times J\} \\ A \,\&\, B & \stackrel{\text{def}}{=} & \{A_i \,\&\, B_j \mid (i,j) \in I \times J\} \end{array}$$

De plus, lorsque la catégorie $\mathcal{C}$ est munie d'une négation tensorielle, on peut l'exporter à $Fam(\mathcal{C})$.

Proposition 2.46 L'opération qui à une famille $A = \{A_i | i \in I\}$ de $Fam(\mathcal{C})$ associe

$$\neg A = \{\&_i(\neg A_i)\}$$

définit une négation tensorielle sur $Fam(\mathcal{C})$

Démonstration : Un morphisme $A \otimes B \to \neg C$ de $Fam(C)$ est donné par des morphismes $f_{i,j} : A_i \otimes B_j \to \&_k(\neg A_k)$ (la fonction de réindexation est ici triviale). Ces morphismes peuvent être projetés par $\pi_k \circ f_{i,j} : A_i \otimes B_j \to \neg C_k$. On peut maintenant utiliser la négation de $\mathcal{C}$ pour avoir des morphismes $g_{i,j,k} : A_i \to \neg(B_j \otimes C_k)$ dont on peut prendre le produit sur i et k

$$g_i \stackrel{\text{def}}{=} \&_{j,k}\, g_{i,j,k} \quad : \quad A_i \to \&_{j,k} \neg(B_j \otimes C_k).$$

On obtient ainsi un morphisme $g : A \to \neg(B \otimes C)$ de la catégorie $Fam(\mathcal{C})$. Il est aisé de vérifier que la transformation définie ci-dessus induit une bijection entre $Fam(C)(A \otimes B, \neg C)$ et $Fam(C)(A, \neg(B \otimes C))$. ■

En rassemblant toutes ces remarques, on peut énoncer le théorème suivant :

Théorème 2.47

$Fam(\mathcal{G})$ est un modèle de la logique tensorielle

Nous avons vu que les preuves sur les modalités de ressource sont un peu périlleuses. Nous présentons au chapitre suivant un moyen pour calculer automatiquement ces modalités à partir d'extensions de Kan. Cette méthode présente le double avantage de fournir un candidat canonique pour définir une modalité de ressource et de fournir une preuve abstraite que cette définition est bien valide.

Chapitre 3

Algèbre libre d'une T-théorie enrichie

La gerbe, Matisse (1953)

Sommaire

L'étude des modalités exponentielles dans les modèles de logique linéaire a souvent été source de malentendus. On sait par exemple que les espaces de cohérence munis de l'exponentielle des cliques finies , donnée originellement par Jean-Yves Girard dans son papier fondateur [Gir87], ne définissent pas un modèle dénotationnel de la logique linéaire. En effet, Thomas Ehrhard a remarqué que cette construction d'un comonoïde commutatif est un peu malade au sens où la déréliction n'est pas naturelle [Mel08]. Il est néanmoins possible de la soigner en considérant l'exponentielle des multicliques finies. Ce malentendu vient du fait qu'il est très fastidieux de vérifier tous les diagrammes commutatifs requis. Nous souhaiterions donc de manière générale pouvoir calculer plus systématiquement les modalités de ressources en logique tensorielle et avoir par la même occasion une preuve automatique que tous les diagrammes de cohérence sont vérifiés. À cette fin, nous abordons maintenant le calcul des algèbres libres d'une T-théorie enrichie. Par exemple, la modalité exponentielle peut se calculer en envoyant tout objet vers son comonoïde commutatif libre. Ainsi, d'un point de vue plus abstrait, la définition d'une modalité exponentielle peut se ramener à l'étude des algèbres libres de la théorie des comonoïdes commutatifs. De la même manière, la modalité affine peut se ramener à l'étude des algèbres libres de la théorie des objets pointés. Ainsi, nous souhaiterions remplacer des preuves directes par des arguments plus abstraits.

Nous exposons dans un premier temps le cheminement qui nous a amené à étudier les T-algèbres libres. Nous introduisons ensuite le cadre bicatégorique des équipements en distributeurs [Woo82, Woo85] qui nous a permis de définir deux conditions pour lesquelles le calcul est possible : une première condition d'aspect combinatoire appelée *opéradicité* et un deuxième condition de caractère algébrique appelée *complétude algébrique*. Ces deux conditions nous permettent d'établir un résultat général d'existence de T-algèbres libres

que nous utilisons pour retrouver les résultats déjà connus d'Eduardo Dubuc [Dub74], et plus récemment de Bruno Vallette [Val04] et Steve Lack [Lac08], sur le calcul des monoïdes libres. Enfin, nous concluons ce chapitre par la description du calcul du comonoïde commutatif libre dans les espaces de cohérence et les jeux de Conway.

3.1 Prolégomènes

L'exemple de base où le calcul d'une algèbre libre est possible nous vient de la théorie des modules. Soit k un anneau commutatif. On associe une k-algèbre TA à chaque k-module A, appelée son *algèbre tensorielle*, et définie comme la somme infinie de produits tensoriels

$$TA \;=\; \bigoplus_{n\in\mathbb{N}} A^{\otimes n} \tag{3.1}$$

où nous notons $A \otimes_k B$ le produit tensoriel de deux k-modules A et B (nous l'avons déjà défini à la Section 1.3 pour le cas particulier des groupes abéliens, i.e. lorsque $k = \mathbb{Z}$). Nous notons $A^{\otimes n}$ le produit $A \otimes_k \cdots \otimes_k A$ pris n fois. Afin de simplifier cette introduction, nous prenons la liberté ici et par la suite de considérer ce produit tensoriel comme étant *strictement* associatif.

En termes catégoriques, on dit que cette construction T est *fonctorielle*. pour dire qu'elle définit un foncteur

$$T \;:\; k\text{-Mod} \;\to\; k\text{-Alg}$$

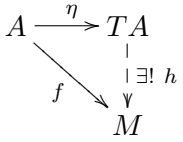
de la catégorie k-Mod de k-modules et morphismes de k-modules à la catégorie k-Alg des k-algèbres et morphismes de k-algèbres. De plus, ce foncteur est adjoint à gauche au foncteur d'oubli

$$U_{alg} \;:\; k\text{-Alg} \;\to\; k\text{-Mod} \tag{3.2}$$

qui transporte toute k-algèbre vers son k-module sous-jacent. Comme nous l'avons déjà présenté dans l'Équation (1.1) des rappels catégoriques du Chapitre 2, cette adjonction

$$T \dashv U_{alg}$$

indique que tout morphisme $f : A \to M$ d'un k-module A vers une k-algèbre M se factorise de manière unique par

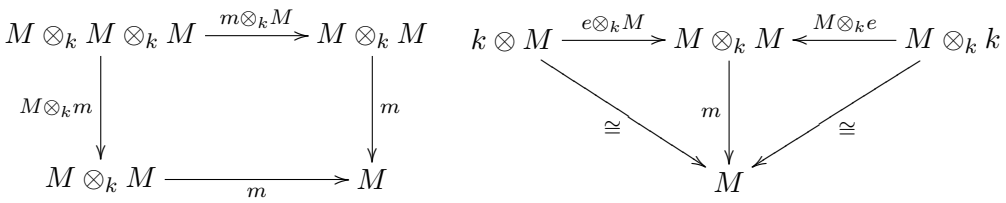


où h est un morphisme de k-algèbres et η est l'injection canonique induite par l'Équation (3.1). Cette propriété exprime que TA est la k-algèbre *libre* générée par le k-module A.

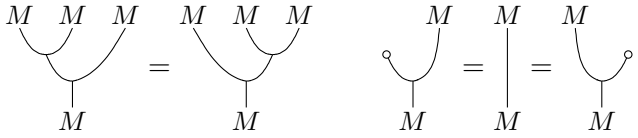
Rappelons qu'une k-algèbre M est définie comme un k-module équipé de deux morphismes,

$$k \;\xrightarrow{\;e\;}\; M \;\xleftarrow{\;m\;}\; M \otimes_k M,$$

appelés respectivement *unité* et *multiplication*, et faisant commuter le diagramme suivant :



Pour une représentation plus visuelle, nous indiquons les diagrammes de cordes correspondants



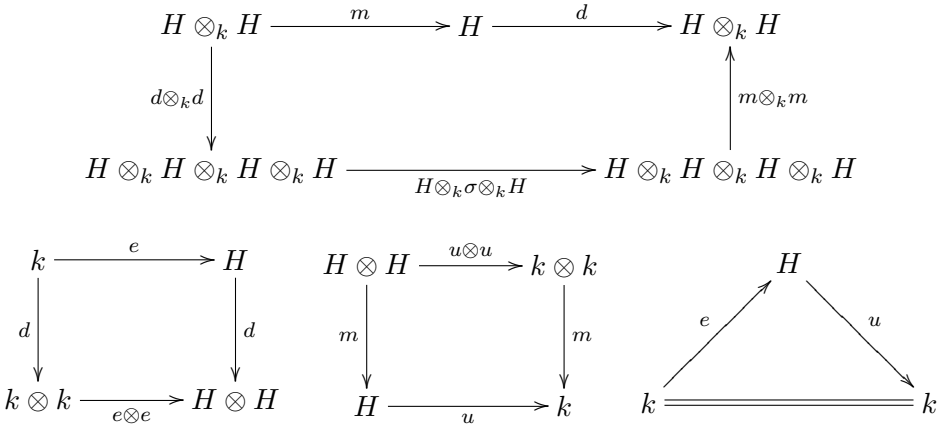
Ainsi, une k-algèbre est un monoïde (voir la Définition 1.41) dans la catégorie des k-modules k-Mod munie du tenseur $\otimes_k$. Autrement dit, la k-algèbre TA est le monoïde *libre* sur A dans la catégorie k-Mod.

Un problème élémentaire en algèbre. L'existence d'une k-algèbre libre TA pour tout k-module A doit être contrastée par le fait, établi par Jean-Louis Loday, qu'il n'existe (en général) pas de k-bigèbre libre sur un k-module. Ceci mérite quelques explications. Premièrement, rappelons qu'une k-cogèbre K est un k-module muni de deux morphismes

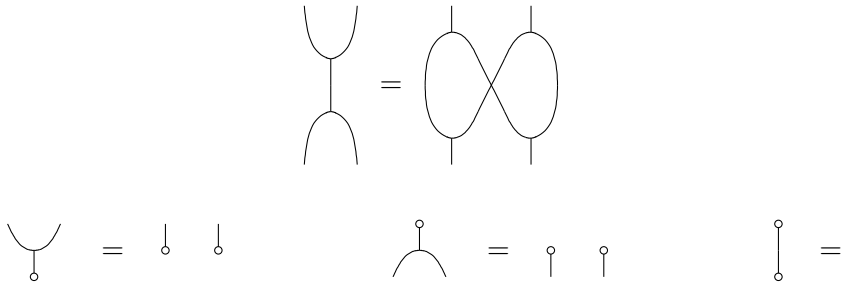
$$k \xleftarrow{u} K \xrightarrow{d} K \otimes_k K$$

appelés respectivement *counité* et *comultiplication*, faisant de K un comonoïde (voir la Définition 1.41) dans k-Mod. On note k-Cog la catégorie des k-cogèbres et morphismes de k-cogèbres.

Ensuite, une k-bigèbre H est un k-module qui a à la fois une structure de k-algèbre et de k-cogèbre, et faisant commuter les diagrammes bien connus de *compatibilité de Hopf* :



Ici, σ correspond à la symétrie de k-Mod. À nouveau, nous donnons une représentation en diagrammes de corde de ces compatibilités



Il est intéressant de constater que k-Alg et k-Cog héritent toutes les deux de la structure monoïdale de k-Mod, et qu'une k-bigèbre peut être vue alternativement ou bien comme un monoïde de k-Cog, ou bien comme un comonoïde de k-Alg.

Maintenant, notons k-Big la catégorie des k-bigèbres avec pour morphismes, les morphismes à la fois de k-algèbres et de k-cogèbres. Jean-Louis Loday a montré que le foncteur d'oubli

$$U_{bij} \quad : \quad k\text{-Big} \longrightarrow k\text{-Mod}$$

n'a pas d'adjoint à gauche, pour un anneau commutatif k arbitraire. Ce résultat établit précisément qu'il n'y a pas de k-bigèbre libre.

Dans ce chapitre, nous voulons comprendre plus conceptuellement ce qui distingue le foncteur d'oubli U_{alg} ayant un adjoint à gauche de cet autre foncteur d'oubli U_{big} n'en ayant pas. De plus, nous voulons extraire, dans les cas où tout se passe bien, une formule à l'image de l'Équation (3.1) pour calculer l'adjoint à gauche – c'est-à-dire la construction libre associée au foncteur d'oubli. Nous orientons maintenant notre étude vers la sémantique fonctorielle de Lawvere, exploitant l'extraordinaire boîte à outils fournie par l'algèbre catégorique, et plus particulièrement les notions de théories algébriques (introduite à la Section 1.4.1) et d'extensions de Kan – que nous allons expliquer maintenant.

Extensions de Kan et théories algébriques. Un *morphisme algébrique* entre deux théories algébriques (voir le Section 1.4.1)

$$j \quad : \quad \mathbb{L}_1 \longrightarrow \mathbb{L}_2$$

est un foncteur préservant les produits, tel que $f(1) = 1$. Tout morphisme algébrique induit un foncteur

$$U_j \quad : \quad \text{Modèle}(\mathbb{L}_2, \mathcal{C}) \longrightarrow \text{Modèle}(\mathbb{L}_1, \mathcal{C})$$

appelé *foncteur d'oubli* associé, qui transporte chaque $\mathbb{L}_2$-modèle

$$\mathbb{L}_2 \xrightarrow{B} \mathcal{C}$$

vers le $\mathbb{L}_1$-modèle

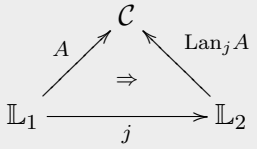
$$\mathbb{L}_1 \xrightarrow{j} \mathbb{L}_2 \xrightarrow{B} \mathcal{C}$$

obtenu en précomposant par le foncteur j.

En guise d'exemple, considérons la théorie triviale $\mathbb{F}^{op}$ et la théorie des monoïdes $\mathbb{M}$ de l'Exemple 1.56. Il existe un (unique) morphisme algébrique $j : \mathbb{F}^{op} \to \mathbb{M}$, qui transporte la i-ème projection $\pi_i \in \mathbb{F}^{op}(n, 1)$ vers le mot à une lettre $i \in \mathbb{M}(n, 1)$. Il est aisé de vérifier que le foncteur d'oubli associé U_j transporte chaque monoïde de la catégorie $\mathcal{C}$ vers son objet sous-jacent.

Cela nous amène à la formulation élégante introduite par William Lawvere d'une « construction libre » dans un cadre fonctoriel, en utilisant des extensions de Kan. Rappelons ici la définition d'une extension de Kan.

Définition 3.1 (extension de Kan)
Une *extension de Kan à gauche* d'un foncteur $A : \mathbb{L}_1 \to \mathcal{C}$ le long d'un foncteur $j : \mathbb{L}_1 \to \mathbb{L}_2$ est donnée par un foncteur $\text{Lan}_j A$, et une transformation naturelle



induisant une bijection pour chaque foncteur $B : \mathbb{L}_2 \to \mathcal{C}$

$$[\text{Lan}_j A, B] \xrightarrow{\cong} [A, B \circ j] \tag{3.3}$$

entre les ensembles de transformations naturelles.

Supposons maintenant que $\mathcal{C}$ soit la catégorie **Ens** des ensembles et fonctions ensemblistes, ou plus généralement n'importe quelle catégorie cartésienne fermée (cf Définition 1.36) avec toutes les petites colimites (voir la Définition 3.6 dans le cadre enrichi). Alors, l'extension de Kan à gauche le long de j existe pour n'importe quel foncteur $\mathbb{L}_1 \to \mathcal{C}$, et il se produit quelque chose d'assez miraculeux : lorsque f est un morphisme algébrique, l'extension de Kan à gauche transporte un foncteur préservant les produits finis A en un foncteur préservant les produits finis $\mathrm{Lan}_j A$. Ainsi, tout $\mathbb{L}_1$-modèle A est transporté vers un $\mathbb{L}_2$-modèle $\mathrm{Lan}_j A$, et la bijection (3.3) se spécialise en la bijection

$$\text{Modèle}(\mathbb{L}_2, \mathcal{C})(\mathrm{Lan}_j A, B) \xrightarrow{\cong} \text{Modèle}(\mathbb{L}_1, \mathcal{C})(A, U_j B)$$

entre des ensembles de morphismes, pour chaque $\mathbb{L}_2$-modèle B. Cette bijection est naturelle en B, ce qui montre que l'extension de Kan à gauche $\mathrm{Lan}_j A$ est le $\mathbb{L}_2$-modèle libre généré par le $\mathbb{L}_1$-modèle A, le long du morphisme algébrique j.

La construction libre est fonctorielle : l'extension de Kan à gauche $\mathrm{Lan}_j A$ définit un foncteur adjoint à gauche au foncteur d'oubli :

$$\mathrm{Lan}_j \dashv U_j : \text{Modèle}(\mathbb{L}_1, \mathcal{C}) \to \text{Modèle}(\mathbb{L}_2, \mathcal{C}).$$

C'est exactement ce qui arrive lorsque $j : \mathbb{F}^{op} \to \mathbb{M}$ est le morphisme algébrique de la théorie algébrique triviale $\mathbb{F}^{op}$ vers la théorie algébrique des monoïdes $\mathbb{M}$. Dans ce cas, le foncteur Lan_j transporte tout ensemble A (identifié ici à un $\mathbb{F}^{op}$-modèle) vers le monoïde libre $\mathrm{Lan}_j A$ dont l'ensemble sous-jacent $A^* = (\mathrm{Lan}_j A)[1]$ est l'ensemble des mots finis sur l'alphabet A

$$A^* \quad = \quad \coprod_{n \in \mathbb{N}} A^{\times n}; \tag{3.4}$$

le produit étant donné par la concaténation de mots. Cette équation bien connue peut être obtenue par un raisonnement direct, ou déduite d'une formule générale utilisant les « cofins » pour calculer les extensions de Kan à gauche dans la catégorie **Ens** – une formule apparaissant dans [ML71] et que nous déduirons de la théorie de « distributeurs » de Jean Bénabou.

Comparons les deux formules (3.1) et (3.4). L'analogie est saisissante : les deux formules calculent *de la même façon* le monoïde libre TA ou A^* généré par un objet A dans leur catégorie respective k-Mod ou **Ens**. Il y a cependant un différence fondamentale : le produit fini $A^{\times n}$ apparaissant dans la définition du monoïde libre A^* est remplacé par un produit tensoriel $A^{\otimes n}$ dans la définition de l'algèbre tensorielle TA. Cela semble indiquer qu'il faille se tourner vers les théories linéaires introduites à la Section 1.4.2.

Extensions de Kan monoïdales et PRO. Comme pour les théories algébriques, un morphisme *linéaire* $j : \mathbb{L}_1 \to \mathbb{L}_2$ entre deux théories linéaires (voir la Section 1.4.2) est donné par un foncteur monoïdal strict tel que $j(1) = 1$. À nouveau, tout morphisme linéaire induit un foncteur d'oubli

$$U_j \quad : \quad \text{Modèle}(\mathbb{L}_2, \mathcal{C}) \longrightarrow \text{Modèle}(\mathbb{L}_1, \mathcal{C})$$

qui transporte chaque $\mathbb{L}_2$-modèle B vers le $\mathbb{L}_1$-modèle $B \circ j$ obtenu en précomposant avec le foncteur j.

Afin de poursuivre notre exemple axé sur les monoïdes, nous considérons maintenant les deux théories linéaires $\mathbb{N}$ et Δ jouant le rôle de deux théories algébriques $\mathbb{F}^{op}$ et $\mathbb{M}$

définies plus haut. À présent, nous sommes *presque* prêts à raconter la même histoire que pour les théories algébriques – au moins pour ce qui est de calculer les monoïdes libres dans la catégorie $\mathcal{C} = k$-Mod des k-modules. Comme $\mathbb{N}$ est la catégorie monoïdale libre sur la catégorie monoïdale à une seul objet, il existe une *unique* morphisme linéaire

$$j_{\mathbb{N}} \quad : \quad \mathbb{N} \longrightarrow \Delta \tag{3.5}$$

envoyant 1 sur 1. Ensuite, chaque k-module $A : \mathbb{N} \to k$-Mod (vu comme un modèle de la théorie triviale $\mathbb{N}$) est transporté par l'extension de Kan à gauche le long de $j_{\mathbb{N}}$ vers le foncteur défini par la formule générale tirée de [ML71] et mentionnée plus haut :

$$\mathrm{Lan}_{j_{\mathbb{N}}} A \quad : \quad p \ \mapsto \ \bigoplus_{n \in \mathbb{N}} \Delta(n,p) \otimes A^{\otimes n}$$

où le k-module $\Delta(n,p) \otimes A^{\otimes n}$ dénote la somme directe d'autant de copies du k-module $A^{\otimes n}$ qu'il y a d'éléments dans le ensemble $\Delta(n,p)$. Il se trouve si on y regarde de plus près que le foncteur $\mathrm{Lan}_{j_{\mathbb{N}}} A$ est *monoïdal*, et définit donc un Δ-modèle qui coïncide avec l'algèbre tensorielle TA générée par un objet A de la catégorie k-Mod. Remarquons que la monoïdalité du foncteur $\mathrm{Lan}_{j_{\mathbb{N}}} A$ se ramène essentiellement à l'isomorphisme

$$\mathrm{Lan}_{j_{\mathbb{N}}} A(p+q) \ \cong \ \mathrm{Lan}_{j_{\mathbb{N}}} A(p) \otimes \mathrm{Lan}_{j_{\mathbb{N}}} A(q)$$

pour toute paire d'entiers naturels p et q. L'isomorphisme lui-même provient de la bijection

$$\Delta(n, p+q) \ \cong \ \coprod_{k=0}^{n} \quad \Delta(k,p) \times \Delta(n-k, q)$$

qui exprime que toute fonction monotone $f : [n] \to [p+q]$ se décompose en une paire de fonctions monotones $f_1 : [k] \to [p]$ et $f_2 : [n-k] \to [q]$ pour un certain entier naturel $0 \leq k \leq n$.

Cette réussite avec les k-modules est un peu miraculeuse, et ne doit pas nous duper. Il semble même que nous rencontrions une sérieuse difficulté : contrairement à ce qui se passe avec les théories algébriques, le fait qu'un $\mathbb{L}_1$-modèle A donne lieu à un extension de Kan à gauche le long du foncteur $j : \mathbb{L}_1 \to \mathbb{L}_2$ ne nous dit pas que le foncteur résultant $\mathrm{Lan}_j A$ est le $\mathbb{L}_2$-modèle libre généré par A ... Il y a deux causes à cette imperfection : (a) rien n'assure que le foncteur $\mathrm{Lan}_j A$ soit monoïdal et donc définisse un $\mathbb{L}_2$-modèle ; (b) même si le foncteur $\mathrm{Lan}_j A$ se trouve être monoïdal, rien n'assure que ce soit le bon – le $\mathbb{L}_2$-modèle libre généré par le $\mathbb{L}_1$-modèle A le long de f. En effet, l'extension de Kan nous donne une bijection entre les transformations naturelles $\mathrm{Lan}_j A \to B$ et les transformations naturelles $A \to U_j B$, pour chaque $\mathbb{L}_2$-modèle B – alors qu'on a besoin d'une bijection entre les transformations naturelles *monoïdales*.

Ces imperfections sont trop fondamentales pour qu'elles puissent être évincées par des arguments pédestres. Il est incontournable de raisonner plus conceptuellement et de se rappeler que la notion d'extension de Kan est avant tout bicatégorique. Nous présenterons dans un premier temps les notions dont nous avons besoin pour réaliser notre raisonnement abstrait avant de développer notre théorie à proprement dite. Nous terminerons enfin par de nombreux exemples, certains provenant directement de l'univers algébrique d'autres provenant de l'univers informatique.

3.2 Bicatégories et pseudoalgèbres enrichies

3.2.1 Catégories enrichies

Comme nous ne nous intéressons pas aux hypothèses minimales sur $\mathcal{V}$ pour que les notions ci-dessous soient définies, nous nous plaçons tout de suite dans le cas confortable

d'une catégorie monoïdale symétrique fermée $(\mathcal{V}, \circledast, I)$ étant complète et cocomplète. Pour une analyse plus fine des catégories enrichies, nous renvoyons le lecteur à l'intarissable monographie de Max Kelly [Kel82].

Définition 3.2 (catégorie enrichie)
Soit $(\mathcal{V}, \circledast, I)$ une catégorie monoïdale. Une *catégorie enrichie sur* $\mathcal{V}$ ($\mathcal{V}$-catégorie) $\mathcal{C}$ est constituée :

- d'une classe $\mathcal{C}_0$ dont les éléments sont appelés *objets* ;
- d'un objet $\mathcal{C}_1(A, B)$ de $\mathcal{V}$ pour chaque paire d'objets A et B appelé le $\mathcal{V}$-espace entre A et B (ou aussi le Hom-objet) ;
- d'une opération $\circ : \mathcal{C}_1(B, C) \circledast \mathcal{C}_1(A, B) \to \mathcal{C}_1(A, C)$ appelé *composition* ;
- d'un morphisme $\mathrm{id}_A : I \to \mathcal{C}_1(A, A)$ pour tout objet A, appelé *identité* ;

qui satisfont les axiomes suivants :

- $\circ$ est associative : le diagramme

$$\begin{array}{ccc}
(\mathcal{C}_1(C,D) \circledast \mathcal{C}_1(B,C)) \circledast \mathcal{C}_1(A,B) & \xrightarrow{a} & \mathcal{C}_1(C,D) \circledast (\mathcal{C}_1(B,C) \circledast \mathcal{C}_1(A,B)) \\
\downarrow{\scriptstyle \circ \circledast \mathcal{C}_1(A,B)} & & \downarrow{\scriptstyle \mathcal{C}_1(C,D) \circledast \circ} \\
\mathcal{C}_1(C,D) \circledast \mathcal{C}_1(A,C) & & \mathcal{C}_1(B,D) \circledast \mathcal{C}_1(A,B) \\
& {\scriptstyle \circ} \searrow \quad \swarrow {\scriptstyle \circ} & \\
& \mathcal{C}_1(A,D) &
\end{array} \tag{3.6}$$

  commute ;
- id_A est le neutre de la composition $\circ$: le diagramme

$$\begin{array}{ccccc}
I \circledast \mathcal{C}_1(A,B) & & & & \mathcal{C}_1(A,B) \circledast I \\
\downarrow{\scriptstyle \mathrm{id}_B \circledast \mathcal{C}_1(A,B)} & \searrow^{\lambda} & & \swarrow_{\rho} & \downarrow{\scriptstyle \mathcal{C}_1(A,B) \circledast \mathrm{id}_A} \\
\mathcal{C}_1(B,B) \circledast \mathcal{C}_1(A,B) & \xrightarrow[\circ]{} & \mathcal{C}_1(A,B) & \xleftarrow[\circ]{} & \mathcal{C}_1(A,B) \circledast \mathcal{C}_1(A,A)
\end{array} \tag{3.7}$$

  commute.

Toute $\mathcal{V}$-catégorie possède une (**Ens**-)catégorie sous-jacente obtenue en appliquant le « changement de base » $\mathcal{V}(I, -) : \mathcal{V} \to \mathbf{Ens}$. Cette opération permet de révéler le **Ens**-espace à partir du $\mathcal{V}$-espace.

Définition 3.3 (foncteur enrichi)
Un *foncteur enrichi sur* $\mathcal{V}$ ($\mathcal{V}$-foncteur) $F : \mathcal{C} \to \mathcal{D}$ entre deux $\mathcal{V}$-catégories $\mathcal{C}$ et $\mathcal{D}$ est la donnée :

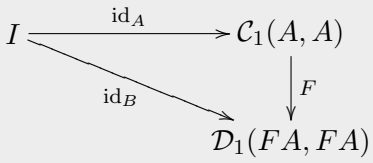
- d'une fonction $F : \mathcal{C}_0 \to \mathcal{D}_0$ qui à tout objet A de $\mathcal{C}$ associe un objet FA de $\mathcal{D}$;
- d'une fonction $F_{A,B} : \mathcal{C}_1(A, B) \to \mathcal{D}_1(FA, FB)$ (souvent simplement notée F) reliant les $\mathcal{V}$-espaces de $\mathcal{C}$ aux $\mathcal{V}$-espaces de $\mathcal{D}$;

qui satisfont les axiomes suivants :

- *préservation de la composition* : pour tous objets A, B et C, on a

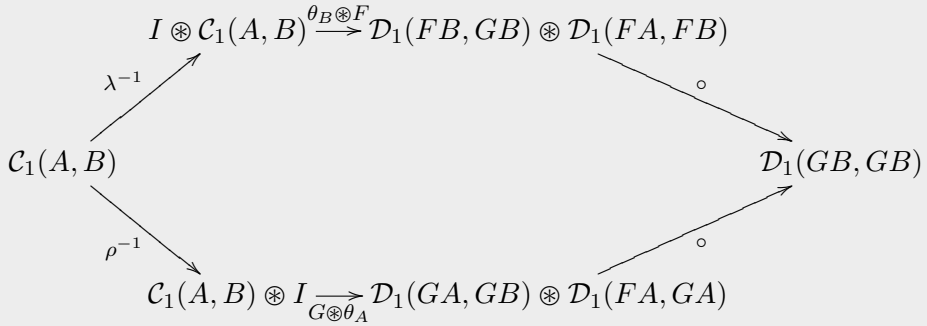
$$\begin{array}{ccc}
\mathcal{C}_1(B,C) \circledast \mathcal{C}_1(A,B) & \xrightarrow{\quad\circ\quad} & \mathcal{C}_1(A,C) \\
\downarrow{\scriptstyle F \circledast F} & & \downarrow{\scriptstyle F} \\
\mathcal{D}_1(FB,FC) \circledast \mathcal{D}_1(FA,FB) & \xrightarrow{\quad\circ\quad} & \mathcal{D}_1(FA,FB)
\end{array}$$

– *préservation des identités* : pour tout objet A de $\mathcal{C}$, on a



Un foncteur F est dit *plein* (resp. *fidèle*) lorsque la fonction $F_{A,B}$ est injective (resp. *surjective*) pour toute paire d'objets A et B de $\mathcal{C}$. Lorsqu'il est les deux à la fois, on dit qu'il est pleinement fidèle

Définition 3.4 (transformation naturelle enrichie)
Soit deux $\mathcal{V}$-catégories $\mathcal{C}$ et $\mathcal{D}$ et deux $\mathcal{V}$-foncteurs $F : \mathcal{C} \to \mathcal{D}$ et $G : \mathcal{C} \to \mathcal{D}$, une *transformation naturelle enrichie sur* $\mathcal{V}$ ($\mathcal{V}$-transformation naturelle) $\theta : F \to G$ entre les $\mathcal{V}$-foncteurs F et G est une famille $\theta_A : I \to \mathcal{D}_1(FA, GA)$ de morphismes de $\mathcal{V}$, indexée sur les objets A de $\mathcal{C}$, telle que pour tous objets A et B et tout morphisme $f : A \to B$ de $\mathcal{C}$ le diagramme



commute dans $\mathcal{V}$.

Comme c'est le cas pour les catégories ordinaires, on peut définir la 2-catégorie $\mathcal{V}$-$\mathbf{Cat}_2$ des $\mathcal{V}$-catégories, $\mathcal{V}$-foncteurs et $\mathcal{V}$-transformations naturelles. Bien sûr, lorsque $\mathcal{V} = \mathbf{Ens}$, on retrouve exactement la 2-catégorie des catégories.

Comme nous avons supposé que $\mathcal{V}$ était symétrique, on peut définir un produit tensoriel sur les $\mathcal{V}$-catégories qui fait de la catégorie des $\mathcal{V}$-catégories et $\mathcal{V}$-foncteurs la catégorie monoïdale symétrique $\mathcal{V}$-**Cat** . L'élément neutre est donné par la $\mathcal{V}$-catégorie à un objet $\star$ et un espace I, on note abusivement cette $\mathcal{V}$-catégorie encore I. Le produit tensoriel de deux $\mathcal{V}$-catégories $\mathcal{C}$ et $\mathcal{D}$ est donné par

– $(\mathcal{C} \otimes \mathcal{D})_0 = \mathcal{C}_0 \times \mathcal{D}_0$;
– $(\mathcal{C} \otimes \mathcal{D})_1((A, B), (A', B')) = \mathcal{C}_1(A, A') \circledast \mathcal{D}_1(B, B')$.

On peut définir la $\mathcal{V}$-catégorie duale $\mathcal{C}^{\mathrm{op}}$ pour toute $\mathcal{V}$-catégorie $\mathcal{C}$ qui a les mêmes objets que $\mathcal{C}$ mais dont les flèches sont inversées

$$\mathcal{C}_1^{\mathrm{op}}(A, B) = \mathcal{C}_1(B, A).$$

À nouveau, la symétrie de $\mathcal{V}$ nous permet de définir les lois de $\mathcal{C}^{\mathrm{op}}$ à partir des lois de $\mathcal{C}$.

En nous inspirant du travail de Max Kelly et Ross Street [KS72], nous voulons définir une notion de pseudomonade et de bicatégorie enrichie des algèbres relâchées associées, le

tout dans un cadre bicatégorique. Avant de commencer à définir ce qu'est une bicatégorie enrichie, regardons le cas plus simple des 2-catégories enrichies. On peut voir ça comme une 2-catégorie au sens de la Section 1.2 mais où l'ensemble de 2-cellules entre deux 1-cellules a été replacé par un objet de $\mathcal{V}$. Étant données deux 1-cellules $f, g : A \to B$ de la 2-catégorie enrichie, on note $[A, B](f, g)$ le $\mathcal{V}$-espace entre elles. Plus formellement, on peut utiliser la définition usuelle d'une 2-catégorie comme une catégorie enrichie sur **Cat** pour donner une définition concise d'une 2-catégorie enrichie.

Définition 3.5 (2-catégorie enrichie)
Une *2-catégorie enrichie sur* $\mathcal{V}$ est une catégorie enrichie sur la catégorie monoïdale symétrique $\mathcal{V}$-**Cat**.

Lorsque la catégorie $\mathcal{C}$ est petite (au sens où sa collection d'objets $\mathcal{C}_0$ est un ensemble), l'ensemble des $\mathcal{V}$-transformations naturelles $[\mathcal{C}, \mathcal{D}](F, G)$ entre deux $\mathcal{V}$-foncteurs $F, G : \mathcal{C} \to \mathcal{D}$ peut être étendu à l'objet de $\mathcal{V}$

$$[\mathcal{C}, \mathcal{D}](F, G) = \int_{C \in \mathcal{C}} \mathcal{D}_1(FC, GC)$$

Grâce à cette construction, on peut définir la 2-catégorie enrichie $\mathcal{V}$-$\mathbf{Cat}_2$ dont les 0-cellules sont les $\mathcal{V}$-catégories, les 1-cellules sont les $\mathcal{V}$-foncteurs et le $\mathcal{V}$-espace des 2-cellules est donné par les $\mathcal{V}$-transformations naturelles.

On suppose à partir de maintenant que $[\mathcal{C}, \mathcal{D}](F, G)$ existe à chaque fois qu'on en aura besoin. Cela n'engendre pas de restriction car Max Kelly a montré que l'on peut toujours « élargir l'univers » afin que l'objet $[\mathcal{C}, \mathcal{D}](F, G)$ existe toujours – voir la section 2.6 de [Kel82]. Cet élargissement consiste à définir une catégorie monoïdale $\mathcal{V} \subseteq \mathcal{V}'$ par complétion afin de récupérer tous les objets $[\mathcal{C}, \mathcal{D}](F, G)$.

Le problème d'inadéquation des limites canoniques Voir $\mathcal{V}$-$\mathbf{Cat}_2$ comme une 2-catégorie enrichie est important dès que l'on s'intéresse aux limites indexées et aux extensions de Kan. En effet, ces deux notions font un usage intensif de la possibilité d'exprimer un isomorphisme entre un espace de flèches dans $\mathcal{C}$ entre deux objets $\mathcal{C}_1(C, C')$ et un espace de $\mathcal{V}$-transformations naturelles dans $\mathcal{V}$-$\mathbf{Cat}_2$ entre deux $\mathcal{V}$-foncteurs $[\mathcal{C}, \mathcal{D}](F, G)$. Bien que cet isomorphisme induise un isomorphisme entre les ensembles sous-jacents, il est assez rare que la réciproque soit vraie – à moins que la catégorie $\mathcal{V}$ soit *conservative* au sens où elle reflète les isomorphismes. Plus généralement, une limite indexée dans une $\mathcal{V}$-catégorie ne peut pas toujours être exprimée par un cône canonique. Ce phénomène est parfois résumé par la phrase « les diagrammes ne disent pas tout ».

Après ces quelques avertissements, nous pouvons maintenant décrire la notion de limite indexée dans un cadre enrichi.

Définition 3.6 (limite indexée)
Soit $F : \mathcal{K} \to \mathcal{V}$ un $\mathcal{V}$-foncteur – vu comme un type d'index – et $G : \mathcal{K} \to \mathcal{C}$ un $\mathcal{V}$-foncteur – vu comme un diagramme dans $\mathcal{C}$ de type F. On définit lorsqu'elle existe la *limite de* G *indexée par* F, notée $\{F, G\}$, comme l'objet de $\mathcal{C}$ vérifiant

$$\mathcal{C}(C, \{F, G\}) \cong [\mathcal{K}, \mathcal{V}](F, \mathcal{C}(C, G-)) \qquad \forall C \in \mathcal{C}$$

De la même manière, on définit la colimite indexée $F \star G$ comme l'objet de $\mathcal{C}$ vérifiant

$$\mathcal{C}(F \star G, C) \cong [\mathcal{K}, \mathcal{V}](F, \mathcal{C}(G-, C)) \qquad \forall C \in \mathcal{C}$$

Remarquons que la généralisation de la notion de limite et de colimite permet de définir les fins et les cofins.

Définition 3.7 (fin et cofin)
La *fin* et la *cofin* d'un $\mathcal{V}$-foncteur $G : \mathcal{C}^{\mathrm{op}} \otimes \mathcal{C} \to \mathcal{D}$ sont définies respectivement par la limite et la colimite indexées suivantes

$$\int_A G(A,A) \ = \{\mathcal{C}_1(-,-), G\} \qquad \int^A G(A,A) \ = \mathcal{C}_1(-,-) \star G$$

Cela permet aussi de définir de manière concise la notion de tenseur et de cotenseur.

Définition 3.8 (tenseur et cotenseur)
Étant donnés deux objets X de $\mathcal{V}$ et C de $\mathcal{C}$, on définit les scalaires associés $\widetilde{X} : I \to \mathcal{V}$, $\widetilde{X}^{op} : I \to \mathcal{V}^{op}$ et $\widetilde{C} : I \to \mathcal{C}$. Ces flèches permettent de définir le tenseur et le cotenseur de C par X respectivement par la colimite et la limite indexées suivantes

$$X \otimes C \ \cong \widetilde{X}^{op} \star \widetilde{C} \qquad \text{et} \qquad X \pitchfork C \ \cong \{\widetilde{X}, \widetilde{C}\}.$$

Nous voyons donc qu'il faut être très attentif lorsqu'on étend les notions classiques de limite et de colimite au cadre enrichi. On peut donc penser que la généralisation des notions catégoriques au cadre enrichi est assez fastidieuse. Heureusement, la plupart des notions catégoriques dont nous avons besoin – comme celle de pseudomonade ou d'algèbre relâchée– sont purement algébriques et peuvent donc être étendues au cadre enrichi directement sans avoir à en modifier les définitions.

Définitions algébriques et enrichissement Nous appelons définition algébrique une définition dans laquelle on demande l'existence de certains morphismes satisfaisant des égalités mais dans laquelle on ne demande pas de choses non structurelles comme l'unicité ou l'universalité d'une construction. Dans ce cas, le passage au cadre enrichi se fait sans heurt. Ou bien la notion que l'on définit est un morphisme de la forme $I \to C$ auquel cas on passe au cadre enrichi sans changer un mot – la seule différence réside dans l'interprétation des diagrammes décrivant les égalités demandées. Ou bien la notion définit un objet de $\mathcal{V}$ auquel cas l'objet que l'on définit doit être donné par un égaliseur. Pour ce deuxième cas, nous reportons le lecteur à la définition d'une 2-cellule algébrique de la Section 3.2.3.

Par exemple, la définition d'un $\mathcal{V}$-foncteur n'est rien d'autre que la définition d'un foncteur où la notion de Hom-set a été remplacée par la notion de $\mathcal{V}$-espace. C'est simplement parce que les $\mathcal{V}$-foncteurs forment un ensemble et non un objet de $\mathcal{V}$. En revanche, les $\mathcal{V}$-transformations naturelles forment les 2-cellules de $\mathcal{V}$-$\mathbf{Cat}_2$ et doivent donc être définies par des égaliseurs. Nous avons donné plus haut une définition par fin mais Max Kelly [Kel82] a montré que cette fin peut être vue comme l'égaliseur suivant

$$\textstyle\int_{A\in A} G(A,A) \longrightarrow \prod_{A\in A} G(A,A) \rightrightarrows \prod_{A,B\in A} [A(A,B), G(A,B)]$$

Nous allons maintenant utiliser ces remarques pour définir les notions enrichies de bicatégories, pseudomonades et algèbres relâchées.

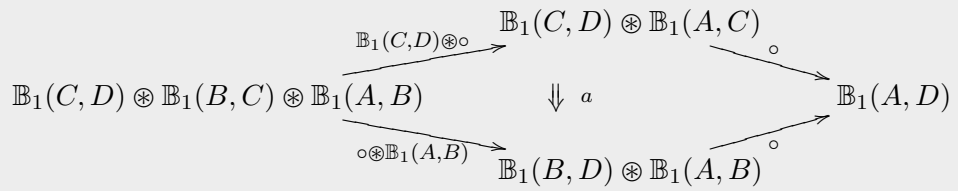
3.2.2 Bicatégories enrichies

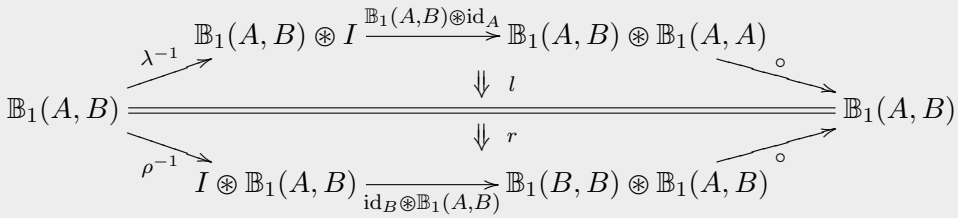
Plus tard dans ce chapitre, nous allons considérer la notion de distributeurs enrichis. Comme la composition des deux distributeurs n'est pas strictement associative, nous devons travailler dans une cadre bicatégorique. Ceci est un affaiblissement de la notion de 2-catégorie définie à la Section 1.2 où la composition des 1-cellules n'est associative qu'à isomorphismes 2-dimensionnels près.

Bien que la définition de bicatégories soit apparue en 1967 au cours du travail de Jean Bénabou [Bé67], la littérature sur ce sujet est assez clairsemée. Nous pouvons tout de même référer le lecteur aux travaux récents de Steve Lack et Robin Houston [Lac07, Hou07] pour une étude des bicatégories dans un cadre non enrichi. La situation est encore plus fragile dans le cadre enrichi où l'écart entre ce que connaissent les spécialistes et ce qui a été écrit est conséquent. Voilà pourquoi nous prenons le temps ici de poser les définitions de base.

Définition 3.9 (bicatégorie enrichie)
Une *bicatégorie enrichie sur* $\mathcal{V}$ ($\mathcal{V}$-bicatégorie) $\mathbb{B}$ est composée des données suivantes :

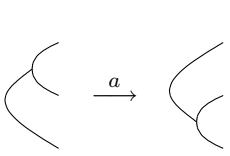
- d'une classe $\mathbb{B}_0$ dont les éléments sont appelés *0-cellules* ;
- d'une $\mathcal{V}$-catégorie $\mathbb{B}_1(A,B)$ pour chaque paire de 0-cellules A et B – les objets de la catégorie enrichie deviennent les *1-cellules* ou *morphismes* de la bicatégorie et les espaces deviennent les *espaces de 2-cellules* ;
- d'un $\mathcal{V}$-foncteur $\mathrm{id}_A : I \to \mathbb{B}_1(A,A)$ pour chaque 0-cellule A appelé *identité* sur A ;
- d'un $\mathcal{V}$-foncteur $\circ : \mathbb{B}_1(B,C) \circledast \mathbb{B}_1(A,B) \to \mathbb{B}_1(A,C)$ pour chaque triplet A, B et C de 0-cellules appelé *composition* ;
- de trois $\mathcal{V}$-transformations naturelles inversibles a, l et r pour chaque quadruplet A, B, C et D de 0-cellules.

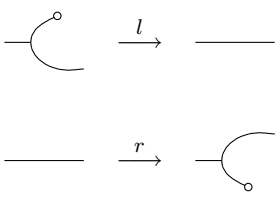




satisfaisant les lois de cohérence de la Figure 3.1.

Nous avons jugé plus lisible de présenter les lois de cohérence sous la forme de diagrammes de cordes. L'associativité de la composition a, l'identité à gauche l et l'identité à droite r vont maintenant être représentées par les diagrammes de cordes





Cette représentation par diagrammes de cordes nous permet de décrire les lois de cohérence par des petits « films » décrivant la déformation des morphismes structuraux. Par exemple, le film de gauche de la Figure 3.1 indique que les deux façons que l'on a de passer d'une association à gauche à une association à droite sont identiques.

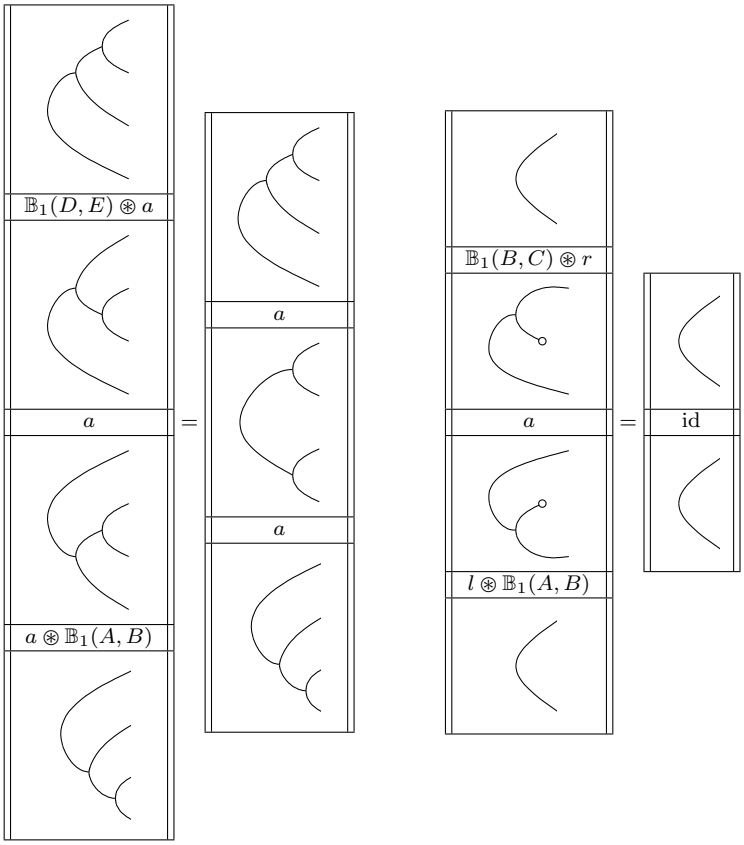
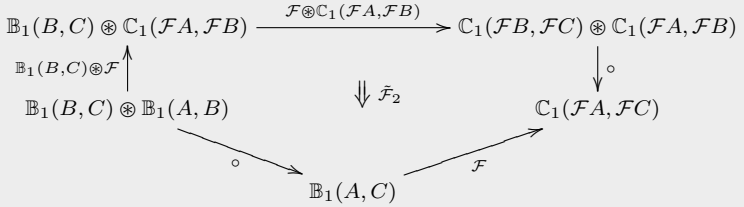
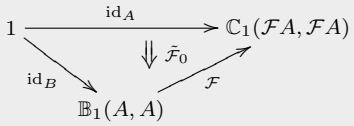


FIG. 3.1 – Lois de cohérence pour l'associativité et l'identité dans une $\mathcal{V}$-bicatégorie

Définition 3.10 (foncteur relâché enrichie)
Un *morphisme de $\mathcal{V}$-bicatégorie* aussi appelé *foncteur relâché enrichi sur $\mathcal{V}$* ($\mathcal{V}$-foncteur relâché) $\mathcal{F} : \mathbb{B} \to \mathbb{C}$ entre deux $\mathcal{V}$-bicatégories est la donnée :

- d'une fonction $\mathcal{F} : \mathbb{B}_0 \to \mathbb{C}_0$ qui associe pour chaque 0-cellule A de la bicatégorie $\mathbb{B}$ une 0-cellule $\mathcal{F}A$ de la bicatégorie $\mathbb{C}$;
- d'un $\mathcal{V}$-foncteur $\mathcal{F} : \mathbb{B}_1(A,B) \to \mathbb{C}_1(\mathcal{F}A, \mathcal{F}B)$ pour chaque paire de 0-cellules A et B ;
- de deux $\mathcal{V}$-transformations naturelles $\tilde{\mathcal{F}}_2$ et $\tilde{\mathcal{F}}_0$ pour chaque triplet A, B et C de la bicatégorie $\mathbb{B}$





satisfaisant les axiomes de la Figure 3.2.

Lorsque les deux $\mathcal{V}$-transformations naturelles sont des isomorphismes, on parle de $\mathcal{V}$-foncteur fort. On représente les deux $\mathcal{V}$-transformations naturelles $\tilde{\mathcal{F}}_2$ et $\tilde{\mathcal{F}}_0$ par les diagrammes de cordes suivants

Encore une fois, ils permettent d'exprimer les lois de cohérence par les petits films de la Figure 3.2.

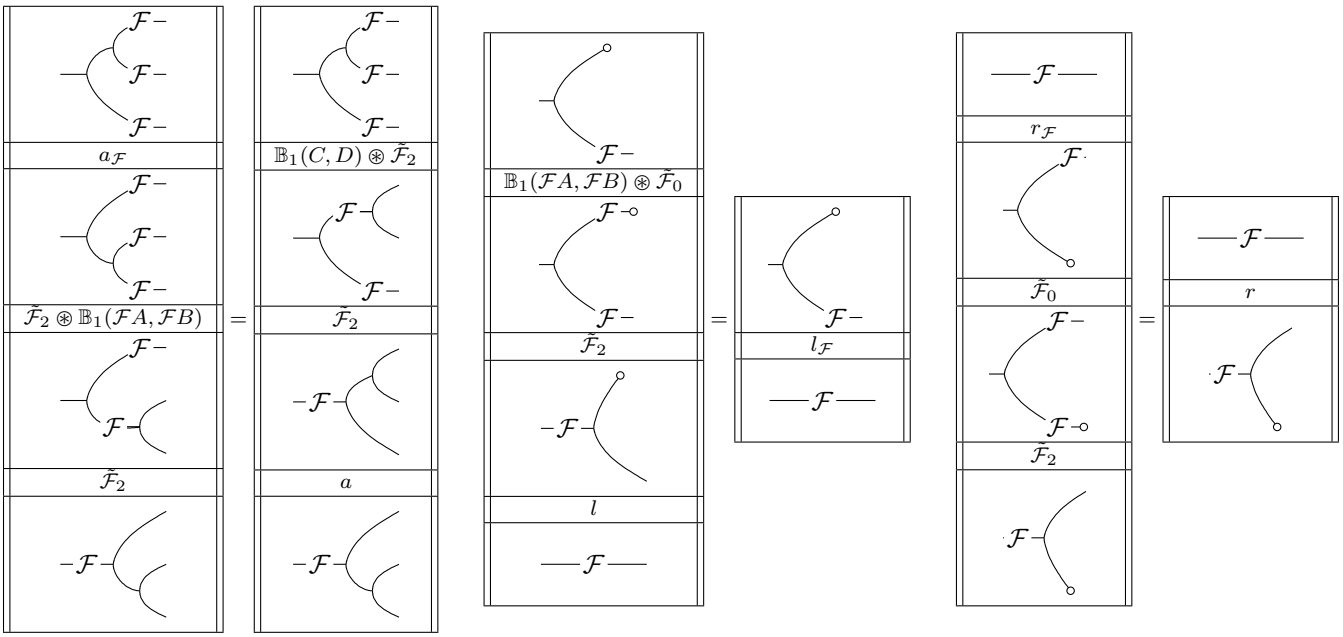
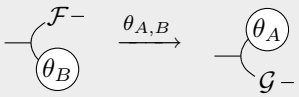


Fig. 3.2 – Lois de cohérence pour les $\mathcal{V}$-foncteurs relâchés

Remarque 3.11 (monade et foncteur relâché)
À la Section 1.2.3, nous avons défini la notion de monade dans une 2-catégorie. Cette notion peut être étendue à une bicatégorie. On s'aperçoit alors qu'un foncteur relâché de $1 \to \mathbb{B}$ est la même chose qu'une monade dans $\mathbb{B}$.

Définition 3.12 (transformation naturelle relâchée enrichie)
Une transformation naturelle relâchée enrichie sur $\mathcal{V}$ ($\mathcal{V}$-transformation naturelle relâchée) $\theta : \mathcal{F} \to \mathcal{G} : \mathbb{B} \to \mathbb{C}$ entre deux $\mathcal{V}$-foncteurs relâchés enrichis est la donnée :
- d'une famille de morphismes $\theta_A : 1 \to \mathbb{B}_1(\mathcal{F}A, \mathcal{G}A)$;

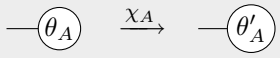
– d'une 2-cellule $\theta_{A,B}$ pour toute paire de 0-cellules A et B de la bicatégorie $\mathbb{B}$



satisfaisant les axiomes de la Figure 3.3.

Lorsque les 2-cellules $\theta_{A,B}$ sont inversibles,on parle de $\mathcal{V}$-transformation naturelle forte.

Définition 3.13 (modification enrichie)
Une *modification enrichie sur* $\mathcal{V}$ ($\mathcal{V}$-modification) $\chi : \theta \to \theta'$ entre deux $\mathcal{V}$-transformations naturelles relâchées est constituée d'une 2-cellule χ_A pour chaque 0-cellule A de $\mathbb{B}$:



satisfaisant la propriété de cohérence dépeinte en Figure 3.4.

Nous introduisons maintenant la notion de 1-cellule pseudomonique dont nous aurons besoin par la suite. Cette notion a été définie dans [CJSV94].

Définition 3.14 (pseudomonique)
Une 1-cellule $f : A \to B$ d'une bicatégorie enrichie $\mathbb{B}$ est dite *pseudomonique* lorsque le $\mathcal{V}$-foncteur de postcomposition par f

$$f \circ (-) : \mathbb{B}_1(A', A) \to \mathbb{B}_1(A', B)$$

est pleinement fidèle pour tout objet A' de $\mathbb{B}$.

3.2.3 Pseudo-monade et algèbres relâchées

Afin de ne pas trop alourdir le texte, nous n'annoterons pas toujours explicitement les notations avec la catégorie monoïdale sous-jacente $\mathcal{V}$ bien que toutes nos définitions soient enrichies. Lorsqu'on écrit des diagrammes bicatégoriques, il est courant de prendre la liberté d'omettre les isomorphismes structurels provenant de la définition même de bicatégorie, foncteur relâché et transformation naturelle relâchée : les conditions de cohérence sont suffisantes pour forcer à ce qu'il y ait à chaque fois un choix unique pour remplir le diagramme. C'est pourquoi nous retenons cette convention ici pour éviter un encombrement parasite autour des concepts de pseudomonade et algèbre relâchée.

Définition 3.15 ($\mathcal{V}$-pseudomonade)
Une $\mathcal{V}$-*pseudomonade* sur une $\mathcal{V}$-bicatégorie $\mathbb{B}$ est la donnée
– d'un $\mathcal{V}$-foncteur relâché $T : \mathbb{B} \to \mathbb{B}$;
– d'une $\mathcal{V}$-transformation naturelle forte $\mu : T^2 \to T$;

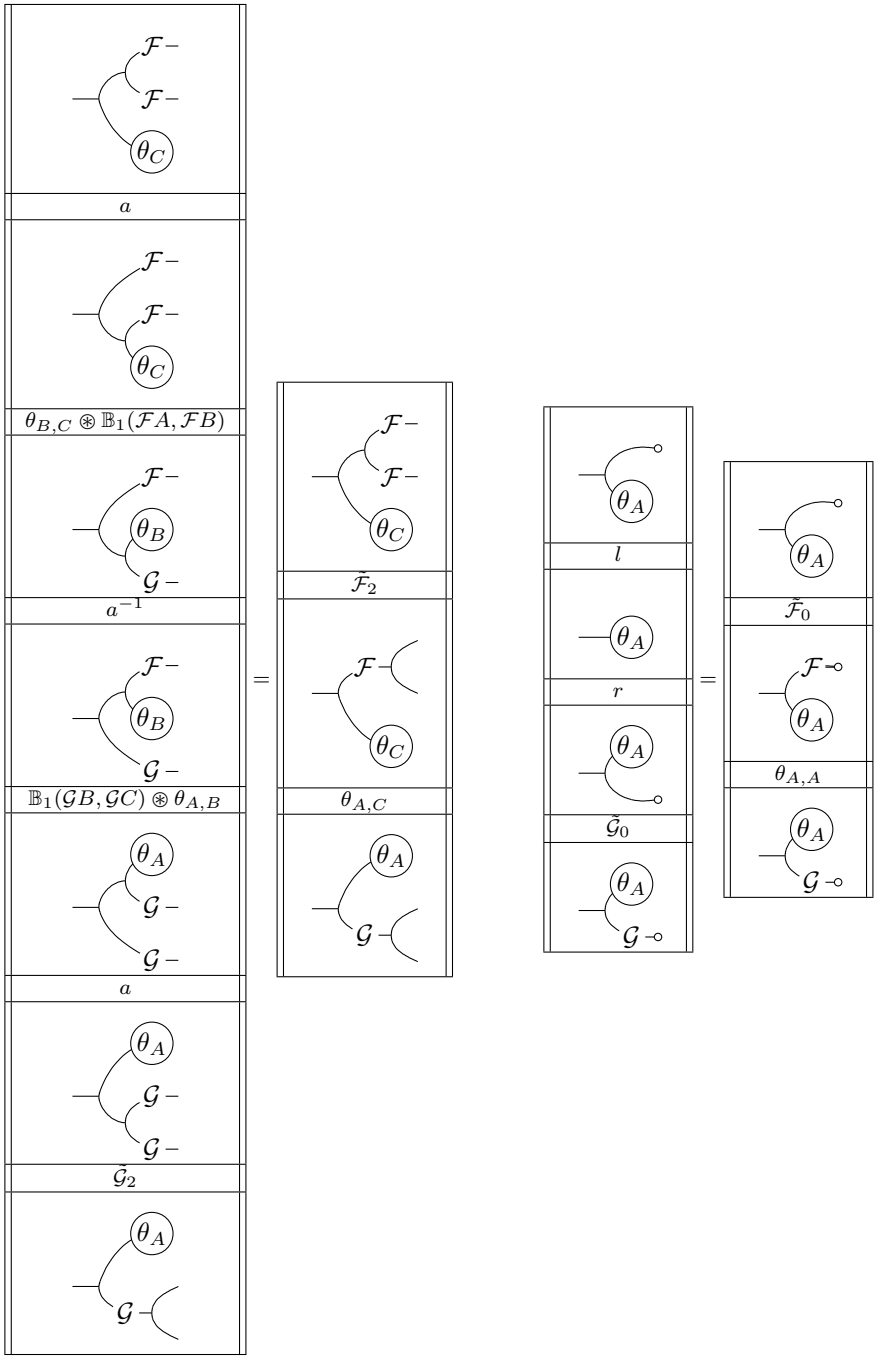


FIG. 3.3 – Lois de cohérence pour les transformations naturelles

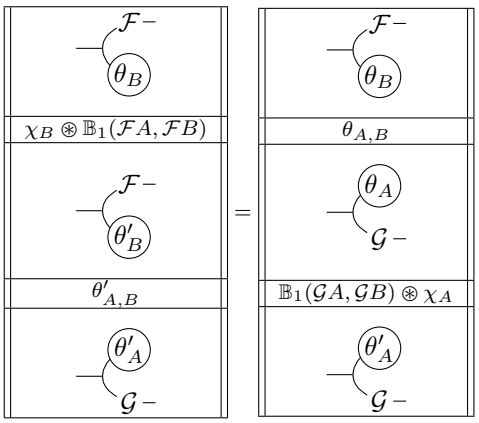
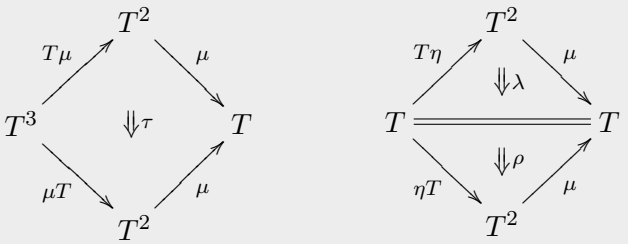
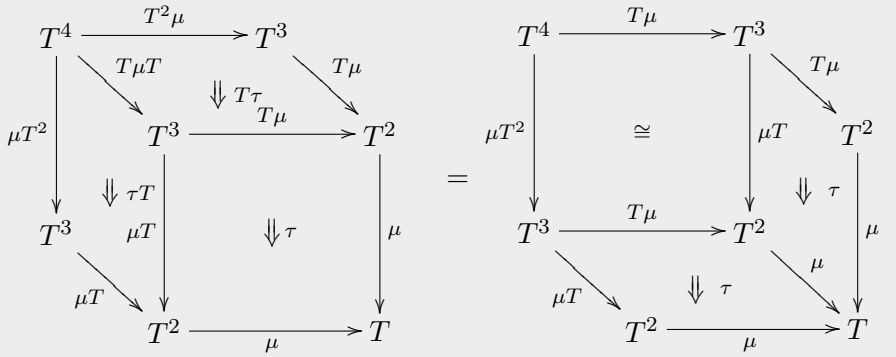


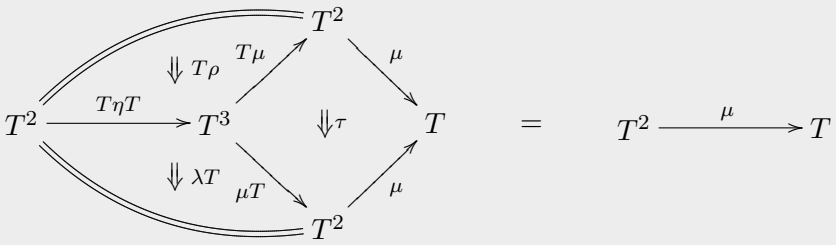
FIG. 3.4 – Lois de cohérence pour les modifications

– d'une $\mathcal{V}$-transformation naturelle forte $\eta : 1 \to T$;
– d'une $\mathcal{V}$-modification inversible



qui satisfont les deux axiomes de cohérence suivant :

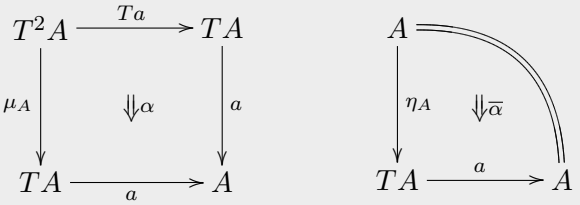




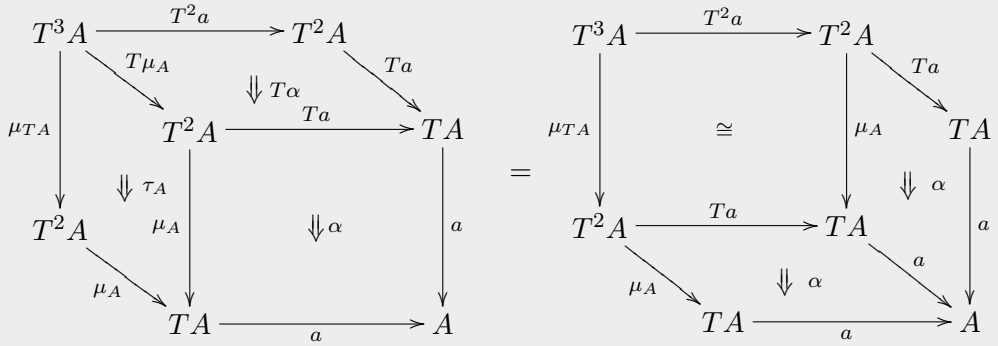
Remarquons que contrairement à ce qu'on peut trouver dans la littérature sur les

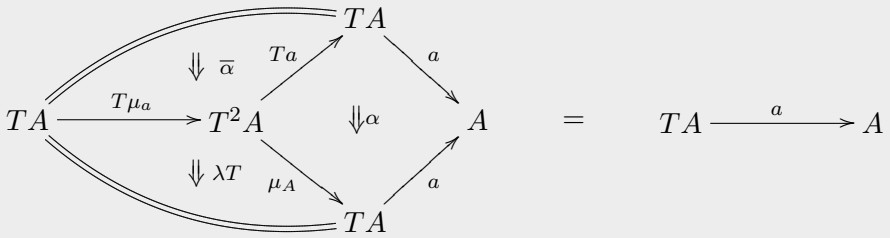
pseudomonades [Mar99, CHP04], nous avons choisi de suivre Nicola Gambino [Gam06] pour le sens des modifications. Ces choix de direction sont en accord avec la notion plus classique d'algèbre relâchée pour une 2-monade [Str74] ; au sens où la pseudoalgèbre libre déterminée par une pseudomonade peut être vue comme une algèbre relâchée sans inverser aucune 2-cellule.

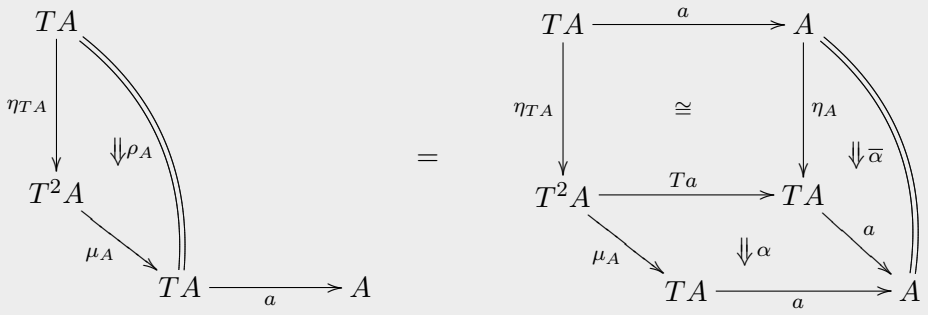
Définition 3.16 (T-algèbre relâchée)
Une *T-algèbre relâchée* pour une $\mathcal{V}$-pseudomonade T est la donnée
- d'un objet A de $\mathbb{B}$;
- d'une 1-cellule $a : TA \to A$;
- de deux 2-cellules



qui satisfont aux conditions de cohérence suivantes :





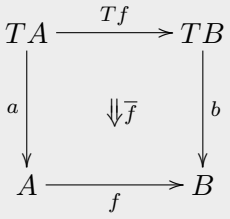


Lorsque les 2-cellules α et $\overline{\alpha}$ sont inversibles, on parle alors d'une pseudoalgèbre ; si les

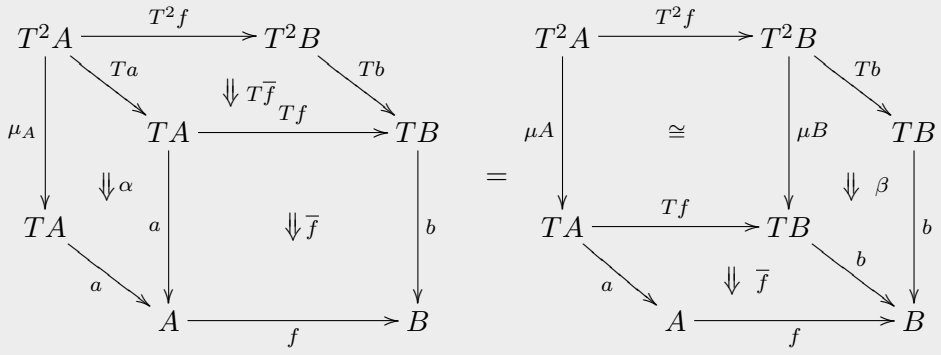
2-cellules vont dans les directions opposées, on parle alors d'algèbre corelâchée ; et si les 2-cellules sont des égalités, on parle alors d'algèbre stricte.

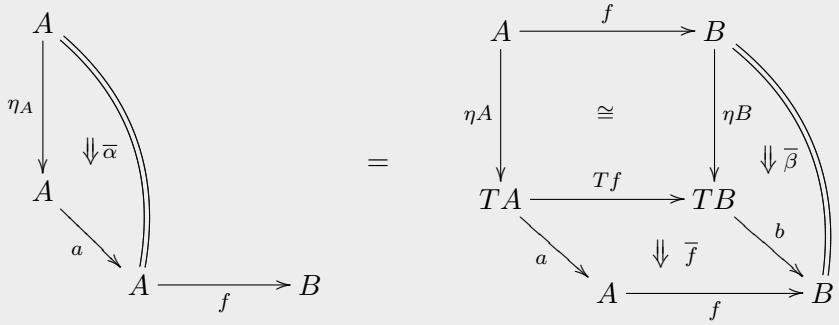
Définition 3.17 (morphisme relâché)
Un *morphisme relâché* entre deux T-algèbres relâchées $(A, a, \alpha, \overline{\alpha})$ et $(B, b, \beta, \overline{\beta})$ est la donnée

– d'une 1-cellule $f : A \to B$;
– d'une 2-cellule



qui satisfait les conditions de cohérence suivantes :



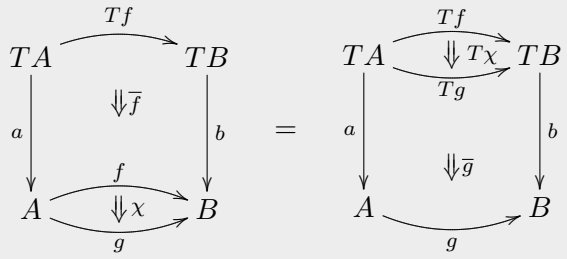


Lorsque la 2-cellule $\overline{f}$ est inversible, on parle alors de pseudomorphisme de T-algèbres. Si elle va dans la direction opposée, on parle alors de morphisme corelâché ; et si la 2-cellule est une égalité, on parle alors de morphisme strict.

Nous introduisons maintenant la notion de 2-cellule algébrique qui va nous permettre d'obtenir une bicatégorie (enrichie) des T-algèbres relâchées.

Définition 3.18 (2-cellule algébrique)
Une *2-cellule algébrique* entre deux morphismes relâchés de T-algèbres $(f, \overline{f})$ et $(g, \overline{g})$ est

une 2-cellule $\chi : f \Rightarrow g$ satisfaisant la loi de cohérence suivante :



Ces données définissent une bicatégorie des T-algèbres relâchées qui coïncide avec la définition habituelle dans le cas $\mathcal{V} = \mathbf{Ens}$. Conformément à la discussion sur l'enrichissement des structures algébriques donnée plus haut, l'ensemble des 2-cellules algébriques peut être enrichi sur $\mathcal{V}$. On définit l'espace représentant les 2-cellules algébriques, noté $[(A,a),(B,b)](f,g)$, par l'égaliseur dans $\mathcal{V}$ des deux flèches $\lambda_1^f = (\_ \circ \overline{f}) \cdot (\_ \circ a)$ et $\lambda_2^g = (\overline{g} \circ \_) \cdot (b \circ \_) \cdot T_{f,g}$:

$$[(A,a),(B,b)](f,g) \rightarrowtail [A,B](f,g) \underset{(\overline{g}\circ\_)\cdot(b\circ\_)\cdot T_{f,g}}{\overset{(\_\circ\overline{f})\cdot(\_\circ a)}{\rightrightarrows}} [TA,B](b \circ Tf, g \circ a)$$

Nous notons $Eq_{f,g}$ le morphisme associé à cette égaliseur. Comme nous nous sommes donné une catégorie $\mathcal{V}$ cocomplète, l'égaliseur $[(A,a),(B,b)](f,g)$ existe pour tous $(f,\overline{f})$ et $(g,\overline{g})$. Nous affirmons qu'il définit le $\mathcal{V}$-espace d'une $\mathcal{V}$-catégorie $[(A,a),(B,b)]$ dont les objets sont les morphismes relâchés de T-algèbres relâchées. Dans le cas non enrichi, le fait que cela définisse une catégorie est assez triviale car il suffit de remarquer que la composée de deux 2-cellules algébriques est encore une 2-cellule algébrique en appliquant successivement les deux lois de cohérence des 2-cellules algébriques. Dans le cas enrichi en revanche, on ne possède pas de description directe de l'égaliseur. On doit donc définir la composition de deux 2-cellules algébriques à l'aide de l'universalité de la construction d'égaliseur. En effet, la composition dans la $\mathcal{V}$-catégorie $[(A,a),(B,b)]$ est déterminée de façon unique par le fait que

$$\bullet \cdot (Eq_{f,g} \otimes Eq_{g,h})$$

égalise les deux flèches λ_1^f et λ_2^h. Pour voir cela, décrivons dans un premier temps la situation de manière géométrique. Les quatre flèches λ_1^f, λ_2^g, λ_1^g et λ_2^h donnent lieu à quatre façons de déformer l'espace $[(A,a),(B,b)](f,g) \otimes [(A,a),(B,b)](g,h)$ vers l'espace $[TA,B](b \circ Tf, h \circ a)$ comme le montre la Figure 3.5.

Il suffit maintenant de remarquer que la flèche $(Eq_{f,g} \otimes Eq_{g,h})$ égalise c'est quatre flèches et que

$$\underset{4}{\overset{1}{\rightrightarrows}} \;=\; \bullet \longrightarrow \underset{\lambda_2^h}{\overset{\lambda_1^f}{\rightrightarrows}} .$$

Plus formellement, la situation est entièrement décrite algébriquement par l'ensemble d'équations suivant :

$$\begin{aligned}
(\lambda_1^f \otimes (_ \circ a)) \cdot (Eq_{f,g} \otimes Eq_{g,h}) &= (\lambda_2^g \otimes (_ \circ a)) \cdot (Eq_{f,g} \otimes Eq_{g,h}) \\
(((b \circ _) \cdot T_{f,g}) \otimes \lambda_1^g) \cdot (Eq_{f,g} \otimes Eq_{g,h}) &= (((b \circ _) \cdot T_{f,g}) \otimes \lambda_2^h) \cdot (Eq_{f,g} \otimes Eq_{g,h}) \\
\lambda_2^g \otimes (_ \circ a) &= ((b \circ _) \cdot T_{f,g}) \otimes \lambda_1^g \\
\bullet \cdot (\lambda_1^f \otimes (_ \circ a)) &= \lambda_1^f \cdot \bullet \\
\bullet \cdot (((b \circ _) \cdot T_{f,g}) \otimes \lambda_2^h) &= \lambda_2^h \cdot \bullet
\end{aligned}$$

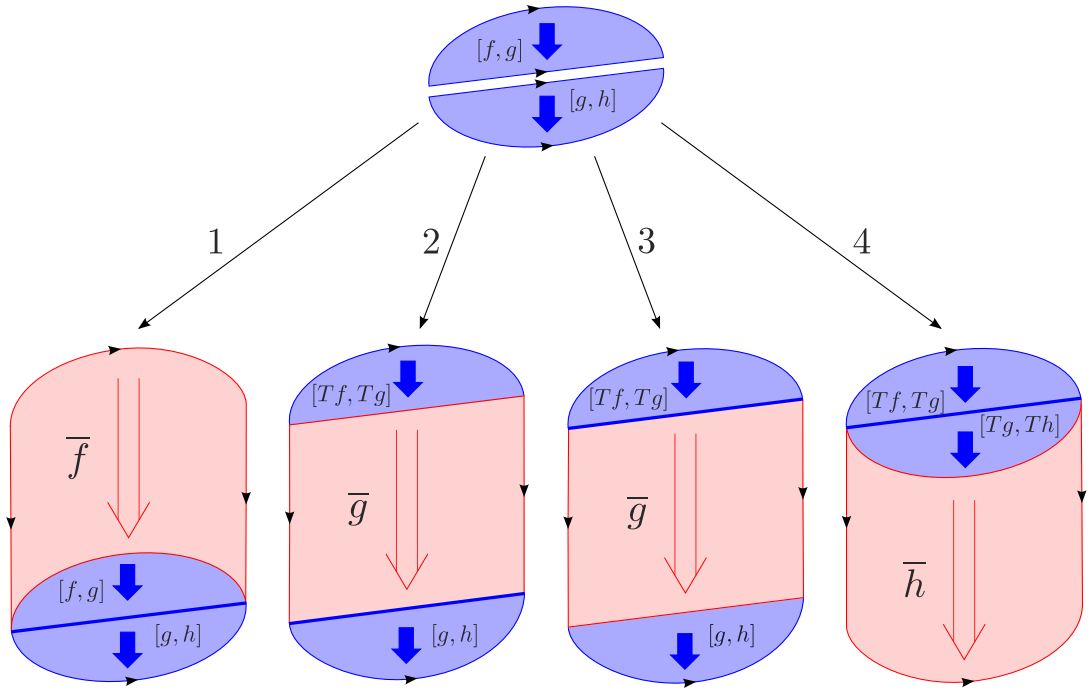


FIG. 3.5 – Les quatre façons de déformer l'espace $[(A,a),(B,b)](f,g) \otimes [(A,a),(B,b)](g,h)$ vers l'espace $[TA,B](b \circ Tf, h \circ a)$. Le trait épais symbolise la composition des deux espaces faite par $\bullet$.

De la même manière, on peut définir les identités de la $\mathcal{V}$-catégorie $[(A,a),(B,b)]$ en remarquant que id_f fait commuter les flèches λ_1^f et λ_2^f. Ainsi, les identités de $[(A,a),(B,b)]$ sont les identités de $[A,B]$. De manière plus formelle, on remarque que les propriétés de l'identité permettent de tirer les équations suivantes :

$$\begin{aligned} (\_ \circ a) \cdot id_f &= id_{f \circ a} \\ (b \circ \_) \cdot T_{f,g} \cdot id_f &= id_{b \circ Tf} \\ (\_ \circ \overline{f}) \cdot id_{f \circ a} &= f \\ (\overline{f} \circ \_) \cdot id_{b \circ Tf} &= f \end{aligned}$$

Les lois d'associativité et de l'unité d'une $\mathcal{V}$-catégorie sont satisfaites par $[(A,a),(B,b)]$ car elles le sont par $[A,B]$ et λ est donné par un égaliseur. Cela finit de montrer notre affirmation que l'on peut donner une structure de $\mathcal{V}$-catégorie à $[(A,a),(B,b)]$. Mais nous voulons aller un peu plus loin en montrant que cette $\mathcal{V}$-catégorie définit un $\mathcal{V}$-espace de la $\mathcal{V}$-bicatégorie des algèbres relâchées d'une $\mathcal{V}$-pseudomonade T.

La $\mathcal{V}$-foncteur de composition entre $[(A,a),(B,b)]$ et $[(B,b),(C,c)]$ est défini par la composition évidente sur les objets (qui sont des morphismes relâchés d'algèbres relâchées). Au niveau des flèches, la composition est donnée par l'unique flèche déterminée par le fait que $\circ \cdot (Eq_{f',g'} \otimes Eq_{f,g})$ égalise les deux flèches $\lambda_1^{f' \circ f}$ et $\lambda_2^{g' \circ g}$. À nouveau, la situation est entièrement décrite par les équations suivantes :

$$\begin{aligned} \bullet \cdot (((g' \circ \_) \cdot \lambda_2^g) \otimes ((\_ \circ Tf) \cdot \lambda_2^{g'}))) &= \lambda_2^{g' \circ g} \\ \bullet \cdot (((g' \circ \_) \cdot \lambda_1^f) \otimes ((\_ \circ Tf) \cdot \lambda_1^{f'}))) &= \lambda_1^{f' \circ f} \end{aligned}$$

Mais la situation peut être plus facilement appréhendée par la description géométrique de la Figure 3.6.

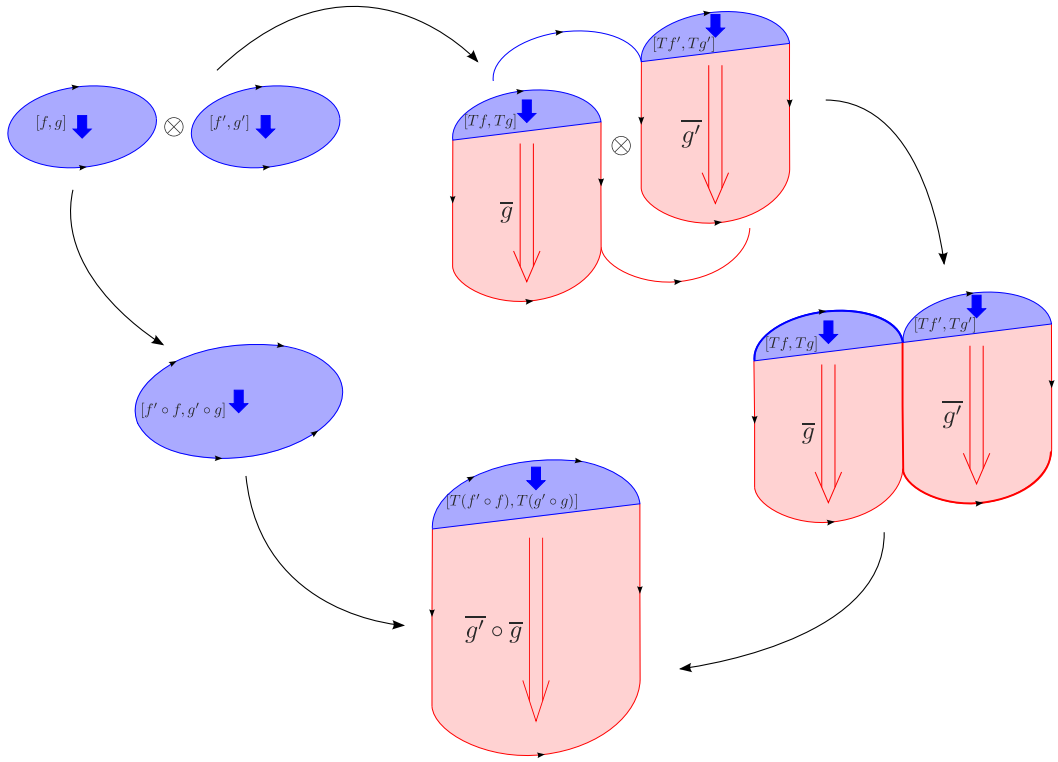


FIG. 3.6 – Une description géométrique de l'équation $\bullet\cdot(((g'\circ\_)\cdot\lambda_2^g)\otimes((\_\circ Tf)\cdot\lambda_2^{g'}))) = \lambda_2^{g'\circ g}$.

Proposition 3.19 Soit T une $\mathcal{V}$-pseudomonade. Les $\mathcal{V}$-algèbres relâchées sur T, morphismes relâchés et 2-cellules algébriques définissent une $\mathcal{V}$-bicatégorie **Lax-T-Alg** .

Si on se restreint aux algèbres fortes, morphismes forts et 2-cellules algébriques, on obtient une autre $\mathcal{V}$-bicatégorie notée **Ps-T-Alg** .

3.3 Équipements en distributeurs

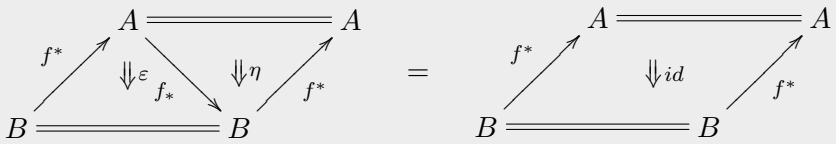
Comme nous l'avons vu en introduction, nous voulons calculer des algèbres enrichies libres par extension de Kan le long d'un $\mathcal{V}$-foncteur. La théorie des distributeurs introduite par Jean Bénabou [Bé73] dans les années 70 nous permet de nous plonger dans un cadre où tout $\mathcal{V}$-foncteur admet un adjoint à droite. Or les extensions de Kan à gauche se calculent très facilement lorsque le morphisme sur lequel on les calcule se trouve être un adjoint à gauche. Nous présentons aussi la notion d'équipement en distributeurs introduite par Richard Wood [Woo82, Woo85] qui permet de donner un cadre abstrait à notre construction.

3.3.1 Adjonctions dans une bicatégorie

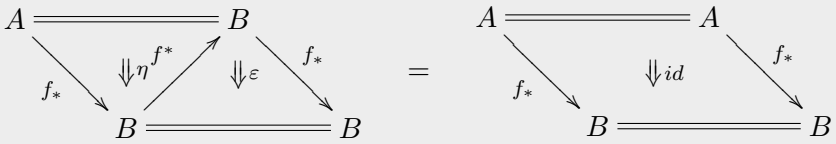
Nous donnons ici la notion d'adjonction dans une bicatégorie enrichie. Comme cette notion est purement algébrique, la définition est essentiellement la même qu'à la Section 1.2.2. On utilise ici la remarque introduite à la Section 3.2.1 sur la transcription au cadre enrichi des notions algébriques.

Définition 3.20 (adjonction dans une bicatégorie enrichie)
Une *adjonction* $f_* \dashv f^* : \eta \to \varepsilon$ entre deux 1-cellules $f_* : A \to B$ et $f^* : B \to A$ d'une bicatégorie enrichie $\mathbb{B}$ est la donnée :
- d'un morphisme $\eta : I \to \mathcal{C}_1(\mathrm{id}_A, f^* \circ f_*)$ appelé *unité* ;
- d'un morphisme $\varepsilon : I \to \mathcal{C}_1(f_* \circ f^*, \mathrm{id}_B)$ appelé *counité* ;

satisfaisant les égalités triangulaires (aussi appelées « identités de zigzag »)



(3.8)



(3.9)

Comme la définition d'une adjonction dans une bicatégorie enrichie épouse parfaitement la définition classique dans une 2-catégorie, la Proposition 1.17 et son Corollaire 1.18 sont toujours valides.

3.3.2 Extension de Kan

Nous allons maintenant définir la notion d'extension de Kan (faible) dans une bicatégorie enrichie en nous inspirant du travail de Ross Street pour le cadre bicatégorique [Str74], et du travail de Max Kelly pour le cadre enrichi [Kel82]. Remarquons que Kelly insiste sur le fait qu'il soit préférable d'utiliser une définition plus forte d'extension de Kan basée sur la notion de limites indexées que certains appellent point à point. Comme nous utilisons les extensions de Kan pour calculer des algèbres libres, la notion qui nous intéresse ici est la première, car elle est plus structurelle.

Définition 3.21 (extension de Kan)
Soit $\mathbb{B}$ une bicatégorie enrichie et $j : A \to B$ et $f : A \to C$ deux 1-cellules. Lorsqu'elle existe, *l'extension de Kan à droite* de f le long de j, notée $\mathrm{Ran}_J F$, est la 1-cellule satisfaisant

$$\mathbb{B}_1(s, \mathrm{Ran}_j f) \cong \mathbb{B}_1(s \circ j, f) \qquad \forall s : B \to C$$

De la même manière, on définit *l'extension de Kan à gauche*, notée $\mathrm{Lan}_J F$, comme étant la 1-cellule satisfaisant

$$\mathbb{B}_1(\mathrm{Lan}_j f, s) \cong \mathbb{B}_1(f, s \circ j) \qquad \forall s : B \to C$$

Lorsque l'extension de Kan à gauche le long d'une 1-cellule j existe pour tout f, alors Lan_j définit un $\mathcal{V}$-foncteur adjoint à gauche au $\mathcal{V}$-foncteur de composition à gauche par j

$$\mathrm{Lan}_j \dashv - \circ j : \mathbb{B}_1(A, C) \to \mathbb{B}_1(B, C)$$

Lorsque $\mathbb{B}$ a tous les tenseurs (resp. cotenseurs), l'extension à gauche (resp. droite) peut être calculée par la cofin (resp. fin) suivante

$$\begin{array}{rcl} \mathrm{Lan}_j f & \cong & \int^a B(ja, -) \otimes fa \\ \mathrm{Ran}_j f & \cong & \int_a B(-, ja) \pitchfork fa \end{array} \tag{3.10}$$

Mais lorsque j à un adjoint à droite, l'extension de Kan à gauche peut être calculée directement avec l'adjoint à droite.

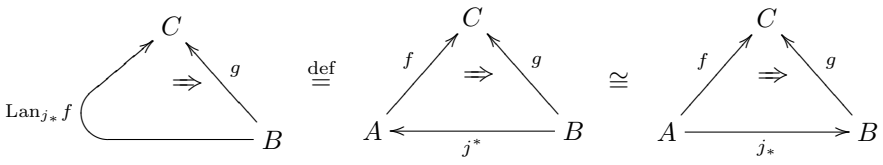
Proposition 3.22 Soit $j_* \dashv j^* : A \to B$ une adjonction dans une bicatégorie enrichie $\mathbb{B}$. Pour toute 1-cellule $f : A \to C$, l'extension de Kan à gauche $\mathrm{Lan}_j f$ existe et est définie par

$$\mathrm{Lan}_{j_*} f = f \circ j^*.$$

De plus, cette construction est fonctorielle au sens où elle définit un $\mathcal{V}$-foncteur

$$\mathrm{Lan}_{j_*} : \mathbb{B}_1(A, C) \to \mathbb{B}_1(B, C).$$

Démonstration : En utilisant le Corollaire 1.18, on a pour tout $g : B \to C$



■

3.3.3 Distributeurs enrichis

Nous présentons ici la notion de distributeurs introduite par Jean Bénabou [Bé73, Bé00] puis remaniée dans le langage des catégories enrichies par l'école de Sydney, autour de Max Kelly et de Ross Street. Plus précisément, Ross Street [Str80] a montré que la notion de distributeur enrichi peut se définir uniquement lorsque la catégorie $\mathcal{V}$ est complète, cocomplète, et les $\mathcal{V}$-foncteurs $- \circledast X$ et $X \circledast -$ préservent les colimites. Cette construction utilise la notion de fibrations dans une bicatégorie et sort donc du cadre de notre étude. Comme nous avons plus d'hypothèses sur le catégorie $\mathcal{V}$, nous pouvons donner ici une construction plus basique qui repose sur l'existence d'un dual et d'un produit tensoriel dans la catégorie $\mathcal{V}$-**Cat** ainsi que sur le calcul des cofins.

Définition 3.23 (distributeur enrichi)
Un *distributeur enrichi* ($\mathcal{V}$-distributeur) $f : \mathcal{C} \nrightarrow \mathcal{D}$ entre deux $\mathcal{V}$-catégories $\mathcal{C}$ et $\mathcal{D}$ est un $\mathcal{V}$-foncteur

$$f : \mathcal{C} \otimes \mathcal{D}^{\mathrm{op}} \ \to \ \mathcal{V}.$$

Une $\mathcal{V}$-transformation naturelle entre deux distributeurs est une $\mathcal{V}$-transformation naturelle entre les $\mathcal{V}$-foncteurs sous-jacents.

La composition de deux $\mathcal{V}$-distributeurs $f : \mathcal{C} \nrightarrow \mathcal{D}$ et $g : \mathcal{D} \nrightarrow \mathcal{E}$ est donnée par la cofin

$$g \circ f(c, e) = \int^{d \in \mathcal{D}} f(c, d) \circledast g(d, e).$$

Comme la composition est obtenue de manière universelle, elle n'est pas strictement associative. On obtient une bicatégorie enrichie $\mathcal{V}$-**Dist** des $\mathcal{V}$-catégories, $\mathcal{V}$-distributeurs et $\mathcal{V}$-transformations naturelles.

Il y a au moins deux façons de voir un $\mathcal{V}$-distributeur :
– comme un $\mathcal{V}$-foncteur généralisé ;
– comme un $\mathcal{V}$-module à plusieurs objets.

Suivant les points de vue, un distributeur est appelé un *profoncteur*, ou un *module*. Nous adoptons ici la terminologie de Jean Bénabou, qui avait en tête une analogie entre les distributeurs et les *distributions* en analyse fonctionnelle.

Nous avons donné plus haut à travers l'Équation (3.10) la formule classique qui permet de calculer une extension de Kan à gauche d'un $\mathcal{V}$-foncteur f le long d'une $\mathcal{V}$-foncteur j dans la plupart des situations intéressantes. Cette formule apparaît par exemple au Chapitre 10 de la monographie de Saunders Mac Lane [ML71].

Le fait que la théorie des distributeurs explique cette formule d'un point de vue conceptuel fait partie du folklore des théoriciens des catégories. L'idée principale de cette reconstruction est que chaque $\mathcal{V}$-foncteur

$$j : \mathcal{C} \ \to \ \mathcal{D}$$

induit une paire de $\mathcal{V}$-distributeurs adjoints $j_* \dashv j^*$.

Définition 3.24
Soit $j : \mathcal{C} \to \mathcal{D}$ un $\mathcal{V}$-foncteur. On peut définir deux $\mathcal{V}$-distributeurs $j_* : \mathcal{C} \nrightarrow \mathcal{D}$ et $j^* : \mathcal{D} \nrightarrow \mathcal{C}$ par

$$j_*(c,d) \ = \ \mathcal{D}(d, jc) \qquad\qquad j^*(c,d) \ = \ \mathcal{D}(jc, d)$$

Ces deux $\mathcal{V}$-distributeurs forment une adjonction

$$j_* \dashv j^* : \mathcal{C} \to \mathcal{D}$$

L'extension de Kan à gauche du $\mathcal{V}$-foncteur

$$f : \mathcal{C} \ \to \ \mathcal{E}$$

le long du $\mathcal{V}$-foncteur j est alors calculer en « composant » les deux distributeurs $f^* : \mathcal{D} \nrightarrow \mathcal{C}$ et $g_* : \mathcal{C} \nrightarrow \mathcal{E}$, puis en « prenant le représentant » du distributeur résultant – c'est à dire en calculant le $\mathcal{V}$-foncteur $\mathrm{Lan}_j f : \mathcal{D} \to \mathcal{E}$ satisfaisant

$$\mathbf{Dist}(f_* \circ j^*, g_*) \ \cong \ \mathbf{Cat}(\mathrm{Lan}_j f, g)$$

pour tout $\mathcal{V}$-foncteur $g : \mathcal{D} \to \mathcal{E}$. La terminologie « représentant » vient de la situation classique d'un objet représentant un foncteur comme le décrivent Michael Barr et Charles Wells [BW85].

Comme nous voulons extraire les ingrédients minimaux dont nous avons besoin pour construire les algèbres libres, nous présentons maintenant la notion d'équipement en distributeurs qui donne un cadre abstrait au traitement du lien entre $\mathcal{V}$-foncteurs et $\mathcal{V}$-distributeurs. Une approche axiomatique préserve les ingrédients de la construction tout en évitant la complicité des détails.

3.3.4 Équipement en distributeur

La notion d'équipement en distributeurs est une formalisation introduite par Richard Wood (dans la cadre non enrichie) [Woo82, Woo85] de l'homomorphisme de bicatégories entre $\mathbf{Cat}_2$ et **Dist** . On utilise ici les notations originelles de Wood $\mathcal{K}$ et $\mathcal{M}$ pour les deux bicatégories enrichies en question.

Définition 3.25 (équipement en distributeurs)
Un *équipement en distributeurs* est un morphisme de $\mathcal{V}$-bicatégories

$$(-)_* : \mathcal{K} \ \to \ \mathcal{M}$$

qui satisfaisait les trois axiomes :

1. les objets de $\mathcal{M}$ sont ceux de $\mathcal{K}$ et $(-)_*$ est l'identité sur les objets ;
2. $(-)_*$ est localement pleinement fidèle, i.e.

$$\mathcal{K}(f, g) \cong \mathcal{M}(f_*, g_*);$$

3. pour toute 1-cellule f de $\mathcal{K}$, f_* a un adjoint à droite f^*.

Exemple 3.26
Mentionnons quelques exemples d'équipement en distributeurs :
- $\mathcal{K} = \mathcal{V}$-**Cat** et $\mathcal{M} = \mathcal{V}$-**Dist** dont nous avons parlé en Section 3.3.3 ;
- $\mathcal{K} = \mathrm{Map}\mathbb{B}$ et $\mathcal{M} = \mathbb{B}$ pour toute bicatégorie enrichie $\mathbb{B}$, où $\mathrm{Map}\mathbb{B}$ est la sous bicatégorie pleine de $\mathbb{B}$ restreinte aux morphismes qui ont des adjoints à droite. $(-)_*$ est alors simplement le morphisme d'inclusion ;
- $\mathcal{K} =$ **Ens** et $\mathcal{M} =$ **Rel** où $(-)_*$ envoie toute fonction $f : A \to B$ vers son graphe f_*

$$a f_* b \quad \text{ssi} \quad fa = b.$$

Dans l'équipement $\mathbf{Cat}_2 \to$ **Dist**, le fait qu'un foncteur $f : \mathcal{C} \to \mathcal{D}$ vérifie l'assertion « f^* est un rétracte de f_* » indique que

$$B(fc, fc') \cong A(c, c')$$

c'est à dire que f est pleinement fidèle. Cette propriété devient une définition ici.

Définition 3.27 (pleinement fidèle)
On dit qu'un morphisme $f : A \to B$ de $\mathcal{K}$ est *pleinement fidèle* lorsque f^* est un rétracte de f_*.

La notion de représentabilité dans un équipement en distributeurs vient de la notion correspondante pour un foncteur. Usuellement, étant donné un foncteur $R : \mathcal{D} \to \mathcal{C}$, on dit d'un objet Lc qu'il représente le foncteur $\mathcal{C}(c, R(-))$ dès que

$$\mathcal{C}(c, R(-)) \cong \mathcal{D}(Lc, -)$$

Il est bien connu que lorsque le foncteur $\mathcal{C}(c, R(-))$ est représentable pour tout objet c de $\mathcal{C}$, le foncteur R a un adjoint à gauche [BW85]. Nous nous inspirons de cette notion pour définir la notion de représentation dans un équipement en distributeurs.

Définition 3.28 (représentant)
Un morphisme $g : B \to C$ de $\mathcal{K}$ est un *représentant* d'un morphisme $f : B \to C$ de $\mathcal{M}$ lorsqu'il y a une bijection

$$\mathcal{M}(f, h_*) \;\cong\; \mathcal{K}(g, h)$$

pour tout $h : B \to C$ dans $\mathcal{K}$.

La notion de représentant permet de trouver le morphisme de $\mathcal{K}$ le « plus proche » d'un morphisme de $\mathcal{M}$. L'idée est alors de calculer une extension de Kan dans $\mathcal{K}$ en passant par $\mathcal{M}$ via l'équipement, puis de la ramener à $\mathcal{K}$ en prenant le représentant.

Nous donnons maintenant une recette pour calculer les représentants. Pour cela, nous allons introduire le concept de *situation de Yoneda* dans un équipement en distributeurs. Cette situation permet de décrire abstraitement les morphismes de $\mathcal{V}$-**Cat** d'une $\mathcal{V}$-catégorie C vers une $\mathcal{V}$-catégorie $\overline{C}$ qui peuvent être vus comme des restrictions du plongement de Yoneda de C dans $\mathcal{V}$-$\mathbf{Cat}_1(C^{\mathrm{op}}, \mathcal{V})$.

Définition 3.29 (situation de Yoneda)
Un morphisme $y : C \to \overline{C}$ de $\mathcal{K}$ est en *situation de Yoneda* lorsque
- y est pleinement fidèle ;
- y^* est pseudomonique par rapport à $\mathcal{K}$, c'est à dire que le $\mathcal{V}$-foncteur

$$y^* \circ (-)_* : \mathcal{K}_1(A, \overline{C}) \;\to\; \mathcal{M}_1(A, C)$$

est pleinement fidèle pour tout objet A de $\mathcal{K}$.

Cette définition abstraite d'une situation de Yoneda permet à son tour de donner une notion générale de $\overline{C}$-cocomplétude sur une équipement en distributeurs.

Définition 3.30 ($\overline{C}$-cocomplétude)
Un objet C de $\mathcal{K}$ est dit *$\overline{C}$-cocomplet* lorsqu'il existe un morphisme $y : C \to \overline{C}$ en situation de Yoneda ayant un adjoint à gauche

$$\mathrm{colim} \dashv y : \overline{C} \to C.$$

Exemple 3.31 (catégorie totale)
Un cas connu de $\overline{C}$-cocomplétude pour l'équipement $\mathcal{V}$-$\mathbf{Cat}_2 \to \mathcal{V}$-**Dist** est obtenu lorsque $\overline{C}$ est la $\mathcal{V}$-catégorie des préfaisceaux sur C et y est le plongement de Yoneda. On dit alors que la $\mathcal{V}$-catégorie C est *totale*. Dans ce cas, la $\mathcal{V}$-catégorie est non seulement cocomplète pour toutes les petites colimites mais aussi complète pour toutes les petites limites [Kel86]. De plus, elle admet certaines limites et colimites de grande taille.

Prenons un morphisme $f : B \to C$ dans la $\mathcal{V}$-bicatégorie $\mathcal{M}$. Supposons que C soit $\overline{C}$-cocomplet pour la situation de Yoneda $y : C \to \overline{C}$ et que f se factorise par y^* au sens où

$$B \xrightarrow{\;f\;} C \;=\; B \xrightarrow{\;\overline{f}_*\;} \overline{C} \xrightarrow{\;y^*\;} C.$$

Proposition 3.32 Le morphisme $\mathrm{colim} \circ \overline{f}$ est un représentant de f.

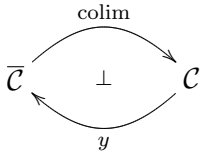
Démonstration : La preuve découle de la cascade d'équivalences suivante

$$\begin{array}{rcll} \mathcal{M}(f, g_*) & = & \mathcal{M}(y^* \circ \overline{f}_*, g_*) & f = y^* \circ \overline{f}_* \\ & \cong & \mathcal{M}(y^* \circ \overline{f}_*, y^* \circ y_* \circ g_*) & y \text{ est pleinement fidèle} \\ & \cong & \mathcal{K}(\overline{f}, y \circ g) & y^* \text{ est pseudomonique par rapport à } \mathcal{K} \\ & \cong & \mathcal{K}(\mathrm{colim} \circ \overline{f}, g) & \mathrm{colim} \text{ est adjoint à gauche de } y \end{array}$$ ■

Expliquons un peu la construction dans l'équipement (non enrichi) paradigmatique $\mathbf{Cat}_2 \to \mathbf{Dist}$. Nous utilisons ici le système de factorisation sur **Cat** par foncteurs finaux et fibrations discrètes décrit à la Section 1.2.5. Ainsi, tout diagramme $f : J \to \mathcal{C}$ peut être vu comme un préfaisceau φ donné par la décomposition

$$J \xrightarrow{f} \mathcal{C} \;=\; J \xrightarrow{f_1} \mathrm{Elt}(\varphi) \xrightarrow{f_2} \mathcal{C}$$

où f_1 est un foncteur final et f_2 est un fibration discrète. Nous suivons maintenant la notion introduite par Max Kelly de $\mathcal{F}$-cocomplétion et considérons un catégorie $\mathcal{C}$ cocomplète pour une certaine classe $\mathcal{F}$ de catégories, appelées indices, contenant la catégorie $\mathbf{1}$. Cela veut dire qu'il y a une adjonction



entre la catégorie $\overline{\mathcal{C}}$ des diagrammes dont la base est dans $\mathcal{F}$ et la catégorie $\mathcal{C}$. Le foncteur y envoie un objet c de $\mathcal{C}$ vers le diagramme constant $y : C \to (\mathbf{1} \to \mathcal{C})$. D'après le lemme de Yoneda, on sait que y est pleinement fidèle et que le foncteur $y^* \circ (-)_*$ est égale à l'identité. On en déduit directement que y^* est pseudomonique par rapport à $\mathbf{Cat}_2$. Ainsi, le foncteur y est bien en situation de Yoneda.

Nous allons maintenant montrer que la notion de représentation permet de calculer les extensions de Kan à gauche dans la $\mathcal{V}$-bicatégorie $\mathcal{K}$.

Proposition 3.33 Soit $f : A \to C$ et $j : A \to B$ deux morphismes de $\mathcal{K}$. Le représentant du morphisme $f_* \circ j^*$ (s'il existe) est l'extension de Kan à gauche de f le long de j dans $\mathcal{K}$.

Démonstration : D'après la Proposition 3.22, le morphisme $f_* \circ j^*$ est l'extension de Kan à gauche de f_* le long de j_* dans $\mathcal{M}$. Notons $\mathrm{Lan}_j f$ son représentant. Pour tout $g : B \to C$ dans $\mathcal{K}$, on a la cascade de bijections suivante

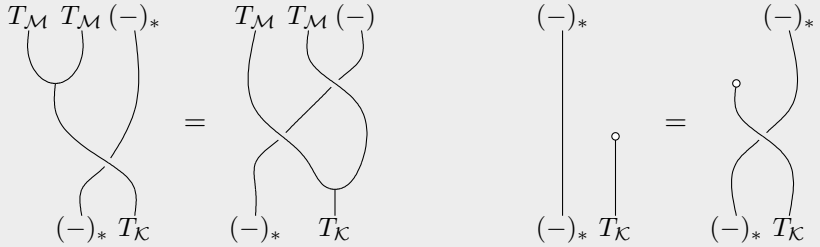
$$\begin{array}{rcll} \mathcal{K}(\mathrm{Lan}_j f, g) & \cong & \mathcal{M}(f_* \circ j^*, g_*) & \mathrm{Lan}_j f \text{ représente } f_* \circ j^* \\ & \cong & \mathcal{M}(f_*, g_* \circ j_*) & f_* \circ j^* \text{ extension de Kan dans } \mathcal{M} \\ & \cong & \mathcal{M}(f_*, (g \circ j)_*) & \text{fonctorialité de } (-)_* \\ & \cong & \mathcal{K}(f, g \circ j) & (-)_* \text{ est pleinement fidèle} \end{array}$$

qui établit que $\mathrm{Lan}_j f$ est bien l'extension de Kan à gauche de f le long de j dans $\mathcal{K}$. ■

Pseudomonade sur un équipement en distributeurs. Pour décrire la notion d'algèbres relâchées dans notre équipement, nous avons besoin d'étendre la notion de pseudomonade à ce cadre. Nous suivons ici les idées sur les lois distributives et en particulier le travail de Nicola Gambino [Gam06] où il définit les morphismes entre pseudomonades.

Définition 3.34 (pseudomonade sur un équipement)
Une *$\mathcal{V}$-pseudomonade sur un équipement en distributeurs* $(-)_* : \mathcal{K} \to \mathcal{M}$ est la donnée
- d'une $\mathcal{V}$-pseudomonade $T_\mathcal{K}$ sur la $\mathcal{V}$-bicatégorie $\mathcal{K}$;
- d'une $\mathcal{V}$-pseudomonade $T_\mathcal{M}$ sur la $\mathcal{V}$-bicatégorie $\mathcal{M}$;
- d'une $\mathcal{V}$-transformation naturelle forte $h : T_\mathcal{M} \circ (-)_* \to (-)_* \circ T_\mathcal{K}$ notée

faisant de $((-)_*, h)$ un morphisme de $\mathcal{V}$-pseudomonades de $T_\mathcal{K}$ vers $T_\mathcal{M}$ au sens où les égalités suivantes sont satisfaites



Exemple 3.35
L'exemple typique de pseudomonade sur un équipement en distributeurs est donnée par les pseudomonades qui distribuent avec Yoneda. Ces pseudomonades induisent naturellement une pseudomonade sur l'équipement en distributeurs $\mathbf{Cat}_2 \to \mathbf{Dist}$. Cette situation a été étudiée en détail par Francisco Marmolejo [Mar99], puis par Eugenia Cheng, Martin Hyland et John Power [CHP04]. Un travail similaire peut être fait dans le cadre enrichi pour donner lieu à des pseudomonades sur l'équipement $\mathcal{V}$-$\mathbf{Cat}_2 \to \mathcal{V}$-$\mathbf{Dist}$.

Toutes les pseudomonades dont nous allons parler plus bas dans le cadre des théories T-algébriques partagent la propriété qu'elles distribuent avec Yoneda.

La notion de pseudomonade sur un équipement permet d'établir que les deux pseudomonades $T_\mathcal{K}$ et $T_\mathcal{M}$ sont compatibles avec le transport par $(-)_*$ au sens suivant.

Proposition 3.36 Le morphisme $(-)_* : \mathcal{K} \to \mathcal{M}$ se relève en un morphisme $(-)_*^T : \mathbf{Lax\text{-}T}_\mathcal{K}\mathbf{\text{-}Alg} \to \mathbf{Lax\text{-}T}_\mathcal{M}\mathbf{\text{-}Alg}$ (que nous noterons aussi abusivement $(-)_*$ par la suite) qui se restreint à travers l'inclusion des pseudoalgèbres vers les algèbres relâchées à un morphisme $(-)_*^T : \mathbf{Ps\text{-}T}_\mathcal{K}\mathbf{\text{-}Alg} \to \mathbf{Ps\text{-}T}_\mathcal{M}\mathbf{\text{-}Alg}$. Ces morphismes font commuter le diagramme suivant

$$\begin{array}{ccc}
\mathbf{Ps\text{-}T}_\mathcal{K}\mathbf{\text{-}Alg} & \xrightarrow{(-)_*^T} & \mathbf{Ps\text{-}T}_\mathcal{M}\mathbf{\text{-}Alg} \\
\downarrow & & \downarrow \\
\mathbf{Lax\text{-}T}_\mathcal{K}\mathbf{\text{-}Alg} & \xrightarrow{(-)_*^T} & \mathbf{Lax\text{-}T}_\mathcal{M}\mathbf{\text{-}Alg} \\
{\scriptstyle U_\mathcal{K}}\downarrow & & \downarrow{\scriptstyle U_\mathcal{M}} \\
\mathcal{K} & \xrightarrow{(-)_*} & \mathcal{M}
\end{array}$$

où $U_\mathcal{K}$ et $U_\mathcal{M}$ sont les morphismes d'oubli usuels.

Démonstration : Les diagrammes sont assez faciles à vérifier et l'ont tous été dans le travail de Nicola Gambino [Gam06]. ■

3.4 Cadre général

Comme nous l'avons esquissé en introduction de ce chapitre, nous voulons calculer des modèles libres par extension de Kan sur des théories de Lawvere, des PROs, ou encore des PROPs. Cela nous amène à définir un cadre commun à toutes ces notions de théories que nous appelons théories T-algébriques. Nous présentons ensuite le cadre abstrait des équipements en distributeurs qui permet de calculer les modèles libres.

3.4.1 Théories T-algébriques

Nous suivons les idées développées par William Lawvere dans son travail sur les *doctrines équationnelles* [Law67] et formulons la notion de *théorie T-algébrique* pour une $\mathcal{V}$-pseudomonade T sur la 2-catégorie enrichie $\mathcal{V}$-$\mathbf{Cat}_2$. Cette notion décrit de manière générique toutes les sortes de théories algébrique, linéaire, symétrique ou même tressée – chaque type de théorie décrivant une pseudomonade particulière T. La seule propriété dont nous aurons besoin par la suite sur la pseudomonade T est qu'elle s'étende à une pseudomonade sur l'équipement $\mathcal{V}$-$\mathbf{Cat}_2 \to \mathcal{V}$-$\mathbf{Dist}$. Nous ne le supposons pas dans la définition générale mais c'est le cas pour les pseudomonades suivantes car elles distribuent avec Yoneda.

- théories algébriques = catégorie avec produits finis libre ;
- théories linéaires = catégorie monoïdale libre ;
- théories symétriques = catégorie monoïdale symétrique libre ;
- théories tressées = catégorie monoïdale tressée libre ;
- esquisses projectives = catégorie avec limites finies libres.

Chacune de ces pseudomonades T induit une 2-catégorie enrichie $\mathbf{Cat}^T$ que nous définissons de la manière suivante.

Définition 3.37 (théorie T-algébrique)
Soit T une pseudomonade sur $\mathcal{V}$-$\mathbf{Cat}_2$. On considère la 2-catégorie enrichie $\mathbf{Cat}^T$ dont les 0-cellules sont les catégories T-algébriques, les 1-cellules sont les foncteurs T-algébriques et les 2-cellules sont les transformations naturelles T-algébriques – où nous avons utilisé le dictionnaire suivant :

- catégorie T-algébrique = pseudoalgèbre de la pseudomonade T ;
- foncteur T-algébrique = pseudomorphisme de pseudoalgèbres ;
- transformation naturelle T-algébrique = 2-cellule algébrique inversible.

Une théorie T-algébrique est définie comme une catégorie T-algébrique *petite* – c'est à dire dont la classe d'objets est un ensemble.

Remarquons que notre définition de théorie T-algébrique est assez généreuse, et n'essaye pas de caractériser quelles théories algébriques doivent être acceptées ou rejetées suivant le rang de la pseudomonade T – contrairement à l'approche développée par Max Kelly et John Power dans leur travail élaboré sur les théories de Lawvere enrichies [KP93, Pow99].

Définition 3.38 (modèle d'une théorie T-algébrique)
Un *modèle* M d'une théorie T-algébrique $\mathbb{L}$ ($\mathbb{L}$-modèle) dans une catégorie T-algébrique $\mathcal{C}$ est un foncteur T-algébrique

$$A : \mathbb{L} \to \mathcal{C}$$

On obtient la $\mathcal{V}$-catégorie Modèle$(\mathbb{L}, \mathcal{C})$ des $\mathbb{L}$-modèles et transformations naturelles T-algébriques.

On peut maintenant reformuler notre problème de calcul de modèle libre par la question suivante :

Quand une extension de Kan à gauche d'un foncteur T-algébrique A le long d'un foncteur T-algébrique j est-elle une extension de Kan T-algébrique ?

3.4.2 Notre résultat principal

Nous allons maintenant établir le résultat principal de ce chapitre au niveau des équipements en distributeurs, avant d'expliquer plus en détail toutes les hypothèses et leurs instanciations. Dans ce qui suit, nous disons qu'un morphisme $f : A \to B$ est T-algébrique pour indiquer que A et B sont des T-pseudoalgèbres et que f est un pseudomorphisme de pseudoalgèbres. Nous travaillons dorénavant avec une pseudomonade $(T_{\mathcal{K}}, T_{\mathcal{M}}, h)$ sur un équipement en distributeurs $(-)_* : \mathcal{K} \to \mathcal{M}$. Comme on l'a vu à la Proposition 3.33, l'extension de Kan à gauche d'un morphisme $T_{\mathcal{K}}$-algébrique $f : A \to C$ le long d'un morphisme $T_{\mathcal{K}}$-algébrique $j : A \to B$ dans $\mathcal{K}$ s'obtient en précomposant f_* avec l'adjoint à droite j^* puis en prenant le représentant du morphisme résultant. Il nous reste maintenant à clarifier dans quelle situation l'extension de Kan ainsi obtenue est-elle aussi $T_{\mathcal{K}}$-algébrique. Nous annonçons brièvement les deux ingrédients de cette construction

l'adjonction	le morphisme $T_{\mathcal{M}}$-algébrique
$j_* \dashv j^*$	$f_* \circ j^* : B \nrightarrow C$
est $T_{\mathcal{K}}$-algébrique	est représenté par un morphisme $T_{\mathcal{K}}$-algébrique.

Pour capturer ces deux situations, nous introduisons la terminologie suivante.

Définition 3.39 (opéradicité)
Un morphisme f de la $\mathcal{V}$-bicatégorie $\mathcal{K}$ est dit *$T_{\mathcal{K}}$-opéradique* lorsqu'il est $T_{\mathcal{K}}$-algébrique et que son adjoint à droite f^* dans $\mathcal{M}$ est $T_{\mathcal{M}}$-algébrique.

Comme nous le verrons par la suite, la notion opéradicité est de nature *combinatoire*. Elle indique une sorte de propriété de décomposition en arbres. La seconde propriété est de nature *algébrique*. Elle donne des contraintes sur les ingrédients de la recette donnée à la Section 3.3.4 pour que le représentant calculé soit aussi algébrique. Cela revient essentiellement à dire que le morphisme colim – qui intuitivement calcule les colimites qui nous intéressent – commute au produit tensoriel.

Définition 3.40 (cocomplétude algébrique)
Un objet $\overline{C}$-cocomplet C de la $\mathcal{V}$-bicatégorie $\mathcal{K}$ est dit *$\overline{C}$-cocomplet T-algébriquement* lorsque C et $\overline{C}$ sont des T-pseudoalgèbres et que les morphismes y, y^* et colim impliqués dans la définition de $\overline{C}$-cocomplétude sont tous des morphismes de T-pseudoalgèbres.

Grâce à ces deux nouvelles notions, nous pouvons maintenant énoncer et démontrer le théorème qui a motivé tout ce chapitre.

Théorème 3.41

Soit $j : A \to B$ un morphisme $T_{\mathcal{K}}$-opéradique, C un objet $\overline{C}$-cocomplet algébriquement de la bicatégorie enrichie $\mathcal{K}$ et $f : A \to C$ un morphisme $T_{\mathcal{K}}$-algébrique. Si le morphisme $f_* \circ j^*$ se factorise par

$$B \xrightarrow{f_* \circ j^*} C \;=\; B \xrightarrow{g_*} \overline{C} \xrightarrow{y^*} C$$

pour tout f, alors le foncteur d'oubli

$$U_f \quad : \quad \mathbf{Ps\text{-}T_{\mathcal{K}}\text{-}Alg}(B, C) \;\to\; \mathbf{Ps\text{-}T_{\mathcal{K}}\text{-}Alg}(A, C)$$

a un adjoint à gauche calculé par extension de Kan à gauche

$$Lan_j \quad : \quad \mathbf{Ps\text{-}T_{\mathcal{K}}\text{-}Alg}(A, C) \;\to\; \mathbf{Ps\text{-}T_{\mathcal{K}}\text{-}Alg}(B, C)$$

Démonstration : D'après la Proposition 3.22, le morphisme $f_* \circ j^*$ est l'extension de Kan à gauche de f_* le long de j_* dans $\mathcal{M}$. D'après le Proposition 3.32, le morphisme colim $\circ g$ est le représentant de $f_* \circ j^*$ dans $\mathcal{K}$. Or d'après la Proposition 3.33, ce représentant est bien l'extension de Kan à gauche $\mathrm{Lan}_j f$ de f le long de j dans $\mathcal{K}$.

Il reste maintenant à vérifier que $\mathrm{Lan}_j f$ soit bien $\mathrm{T}_{\mathcal{K}}$-algébrique. Comme j est opéradique, le morphisme $f_* \circ j^*$ est $T_{\mathcal{M}}$-algébrique. Mais le morphisme y^* est pseudomonique par rapport à $\mathcal{K}$ donc g est $T_{\mathcal{K}}$-algébrique. Comme colim est $T_{\mathcal{K}}$-algébrique, on en déduit que $\mathrm{Lan}_j f$ est $T_{\mathcal{K}}$-algébrique.

La fonctorialité de la construction s'obtient en utilisant encore une fois la pseudomonicité de y^*. ■

Pour le cas particulier des théories T-algébriques, on déduit du Théorème 3.41 que le $\mathbb{L}_2$-modèle libre sur un $\mathbb{L}_1$-modèle A le long d'un morphisme $j : \mathbb{L}_1 \to \mathbb{L}_2$ se calcule par la cofin

$$\mathrm{Lan}_j A \;=\; \int^{m \in \mathbb{L}_1} \mathbb{L}_2(fm, n) \otimes A(m)$$

Comme nous avons déjà beaucoup discuté les conclusions de ce théorème, nous allons directement commenter les deux hypothèses d'opéradicité et de complétude algébrique.

3.4.3 Une hypothèse combinatoire : l'opéradicité

Nous avons dit que la notion d'opéradicité était de nature combinatoire. Nous allons maintenant expliquer cette intuition pour les théories T-algébriques. Dans cette situation, un morphisme T-algébrique $j : \mathbb{L}_1 \to \mathbb{L}_2$ est T-opéradique lorsque le morphisme canonique

$$\int^{p \in T(\mathbb{L}_1)} \mathbb{L}_1(m, [p]) \otimes T(\mathbb{L}_2)(Tj(p), n) \;\longrightarrow\; \mathbb{L}_2(jm, [n]) \tag{3.11}$$

est un isomorphisme dans la catégorie $\mathcal{V}$, pour tout objet m de la catégorie enrichie $\mathbb{L}_1$, et pour tout objet n de la catégorie enrichie $T(\mathbb{L}_2)$.

Cette définition est purement algébrique mais sa signification est fondamentalement combinatoire.

Opéradicité = Propriété de décomposition en arbres.

Pour comprendre pourquoi, cela vaut la peine d'instancier la définition de foncteur opéradique dans le cas particulier (non enrichi) des théories linéaires – en gardant en tête qu'un objet m de la théorie linéaire $\mathbb{L}_1$ est un entier naturel, et qu'un objet n de la catégorie $T(\mathbb{L}_2)$ est une suite finie $(n_1, \cdots, n_k)$ d'entiers naturels.

Comme la catégorie sous-jacente $\mathcal{V}$ est la catégorie **Ens**, le morphisme canonique (3.11) est ici une fonction dans le domaine est l'ensemble

$$\int^{p_1 \in \mathbb{L}_1} \cdots \int^{p_k \in \mathbb{L}_1} \mathbb{L}_1(m, p_1 + \cdots + p_k) \times \mathbb{L}_2(p_1, n_1) \times \cdots \times \mathbb{L}_2(p_k, n_k)$$

des paires $(g, h_1, \cdots, h_k)$ constituée d'un morphisme

$$g \quad : \quad m \ \to \ p_1 + \cdots + p_k$$

de la théorie linéaire $\mathbb{L}_1$, et d'une famille de k-morphismes

$$h_i \quad : \quad p_i \ \to \ n_i \qquad\qquad (1 \leq i \leq k)$$

de la théorie linéaire $\mathbb{L}_2$. Ces paires sont considérées modulo la plus petite relation d'équivalence $\sim$ satisfaisant :

$$(g, h_1 \circ j(h'_1), \cdots, h_k \circ j(h'_k)) \ \sim \ ((h'_1 \otimes \cdots \otimes h'_k) \circ g, h_1, \cdots, h_k)$$

à chaque fois que

$$h'_i \quad : \quad p_i \ \to \ q_i \qquad\qquad (1 \leq i \leq k)$$

est une famille de morphismes dans la théorie linéaire $\mathbb{L}_1$.

Maintenant, une fonction (3.11) transporte toute famille $(g, h_1, \cdots, h_k)$ vers le morphisme

$$(h_1 \otimes \cdots \otimes h_k) \circ j(g) \quad : \quad m \to p_1 + \cdots + p_k \to n_1 + \cdots + n_k$$

de la théorie linéaire $\mathbb{L}_2$. Par opéradique, nous voulons dire que la fonction (3.11) est une bijection, pour tout entier naturel m et toute séquence d'entiers naturels $(n_1, \cdots, n_k)$. Cela doit être vu comme une propriété de décomposition en arbres, qui stipule que tout morphisme

$$h \quad : \quad m \ \to \ n_1 + \cdots + n_k$$

de la théorie linéaire $\mathbb{L}_2$ se décompose de manière unique par un morphisme

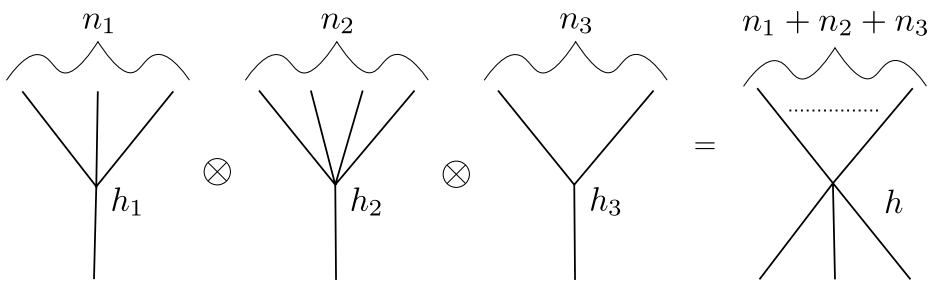
$$j(g) \quad : \quad m \to p_1 + \cdots + p_k$$

suivi d'un morphisme

$$h_1 \otimes \cdots \otimes h_k \quad : \quad p_1 + \cdots + p_k \ \to \ n_1 + \cdots + n_k,$$

bien évidement tout ça modulo la relation d'équivalence $\sim$ définie ci-dessus.

La terminologie « opéradique » est justifiée par le fait que cette propriété est vraie pour tout foncteur linéaire $j : \mathbb{L}_1 \to \mathbb{L}_2$ entre deux opérades – c'est à dire deux théories linéaires $\mathbb{L}_1$ et $\mathbb{L}_2$ générées par une théorie équationnelle sur des opérations n-aires $n \to 1$. Dans ce cas, on peut prendre l'identité pour le morphisme g, avec $m = p_1 + \cdots + p_k$, et chaque morphisme h_i décrit un « composant » particulier avec n_i entrées du morphisme h vu comme une « forêt » d'opérations avec $n_1 + \cdots + n_k$ entrées.



En particulier, le foncteur linéaire $\mathbb{N} \to \Delta$ mentionné dans (3.5) est opéradique : cela explique pourquoi la formule (3.1) pour l'algèbre tensorielle, obtenue en prenant l'extension de Kan à gauche le long de j, calcule effectivement le monoïde libre dans k-Mod, et définit un adjoint à gauche au foncteur d'oubli $U : k\text{-Alg} \to k\text{-Mod}$.

Un autre exemple est donné par le foncteur linéaire de la théorie linéaire des k-algèbres vers la théorie linéaire des algèbres de Lie. Cet exemple induit la notion de k-algèbre enveloppante. Un exemple moins immédiat est donné par la théorie symétrique des comonoïdes vers la théorie symétrique des bimonoïdes.

Enfin, nous terminerons en donnant un contre-exemple, celui de la théorie linéaire triviale $\mathbb{N}$ vers celle des bimonoïdes. Cela explique l'observation de Jean-Louis Loday rappelée dans l'introduction qu'il n'existe pas de k-bigèbre libre en général.

Cas des théories algébriques et des esquisses projectives. Pour les théories algébriques et les esquisses projectives, on sait depuis longtemps que le calcul d'un $\mathbb{L}_2$-modèle libre sur un $\mathbb{L}_1$-modèle est toujours possible. Il semble donc que l'hypothèse d'opéradicité soit toujours valide dans ces deux cas. Regardons dans un premier temps ce que cela veut dire dans le cas cartésien. Le domaine du morphisme canonique 3.11 s'écrit alors

$$\int^{p_1 \in \mathbb{L}_1} \cdots \int^{p_k \in \mathbb{L}_1} \mathbb{L}_1(m, p_1 \times \cdots \times p_k) \times \mathbb{L}_2(jp_1, n_1) \times \cdots \times \mathbb{L}_2(jp_k, n_k). \tag{3.12}$$

On peut utiliser la propriété d'appariement surjectif (*surjective pairing* en anglais) pour en déduire

$$\mathbb{L}_1(m, p_1 \times \cdots \times p_k) \;\cong\; \mathbb{L}_1(m, p_1) \times \cdots \times \mathbb{L}_1(m, p_k).$$

Ensuite, en utilisant le lemme de Yoneda, on en déduit que l'objet 3.12 est isomorphe à

$$\mathbb{L}_2(jm, n_1) \times \cdots \times \mathbb{L}_2(jm, n_k)$$

et on conclut en utilisant une nouvelle fois l'appariement surjectif. Il apparaît qu'on peut faire un raisonnement plus abstrait qui capture à la fois le cas cartésien et le cas esquissable. Les deux monades associées à ces types de théories – la monade des catégories cartésienne et celle des catégories avec limites finies – partagent une même propriété. En effet, dans ces deux cas, des KZ-doctrines, ce qui indique que la structure T-algébrique $a : T\mathbb{L}_1 \to \mathbb{L}_1$ d'une $\mathcal{V}$-catégorie $\mathbb{L}_1$ est l'adjoint à droite de l'unité $\eta_{\mathbb{L}_1}$ de la pseudomonade T. Cela signifie que l'on a un isomorphisme

$$\mathbb{L}_1(m, a(p)) \;\cong\; T\mathbb{L}_1(\eta_A(m), p)$$

On peut donc utiliser cette version abstraite de l'appariement surjectif pour montrer que la propriété d'opéradicité est toujours valide :

$$\begin{array}{lll}
 & \int^{p \in T(\mathbb{L}_1)} \mathbb{L}_1(m, [p]) \otimes T(\mathbb{L}_2)(Tj(p), n) & \\
\cong & \int^{p \in T(\mathbb{L}_1)} T\mathbb{L}_1(\eta_{\mathbb{L}_1}(m), p) \otimes T(\mathbb{L}_2)(Tj(p), n) & (a \vdash \eta_{\mathbb{L}_1}) \\
\cong & T\mathbb{L}_2(Tj(\eta_{\mathbb{L}_1}(m))n) & (\text{lemme de Yoneda}) \\
\cong & T\mathbb{L}_2(\eta_{\mathbb{L}_2}(jm), n) & (\text{naturalité de } \eta) \\
\cong & \mathbb{L}_2(jm, [n]) & (a \vdash \eta_{\mathbb{L}_1})
\end{array}$$

On en déduit donc un résultat bien connu, qui est pour nous un corollaire du Théorème 3.41, et qui stipule que le calcul de modèles libres sur **Ens** est toujours possible pour les théories de Lawvere et les esquisses projectives.

Corollaire 3.42 Soit j un morphisme algébrique entre deux théories algébriques ou deux esquisses projectives $\mathbb{L}_1$ et $\mathbb{L}_2$. Tout $\mathbb{L}_1$-modèle A induit un $\mathbb{L}_2$-modèle $\mathrm{Lan}_j A$, et la bijection (3.3) se spécialise en la bijection

$$\text{Modèle}(\mathbb{L}_2, \mathbf{Ens})(\mathrm{Lan}_j A, B) \xrightarrow{\cong} \text{Modèle}(\mathbb{L}_1, \mathbf{Ens})(A, U_j B)$$

Plus précisément, l'extension de Kan à gauche $\mathrm{Lan}_j A$ définit un foncteur adjoint à gauche au foncteur d'oubli :

$$\mathrm{Lan}_j \dashv U_j : \text{Modèle}(\mathbb{L}_1, \mathbf{Ens}) \to \text{Modèle}(\mathbb{L}_2, \mathbf{Ens}).$$

3.4.4 Une hypothèse algébrique : la complétude algébrique

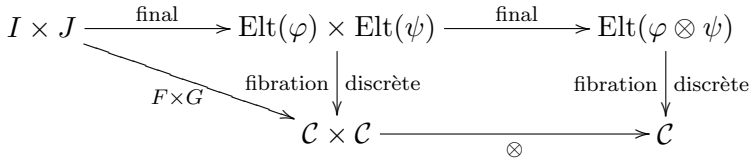
Une catégorie est dite cocomplète lorsqu'elle a toutes les petites colimites. Lorsque la catégorie est de surcroît monoïdale, on demande en général que « le tenseur commute aux colimites ». Très bien... Mais qu'est-ce que cela veut dire ? Nous avons clarifié ce concept en le reformulant dans le langage des équipements en distributeurs, à savoir par la propriété de *complétude algébrique*. Regardons ce qu'il signifie dans le cas de l'équipement 2-**Cat** $\to$ **Dist**. Rappelons la situation décrite à la Section 3.3.4. On suppose qu'une catégorie $\mathcal{C}$ à toutes les colimites indexées par un classe $\mathcal{F}$ de catégories. Si on reste informel, on peut énoncer le slogan suivant

$\overline{C}$-cocomplétude T-algébrique = les colimites calculées sur des diagrammes indexés par un élément de $\mathcal{F}$ commutent avec la structure T-algébrique.

À nouveau, regardons plus précisément ce qui se passe dans le cas des théories linéaires. La catégorie $\mathcal{C}$ est $\overline{C}$-cocomplète T-algébriquement lorsque la classe d'indices $\mathcal{F}$ est close pour le produit de catégories et que le foncteur colim est T-algébrique. Étant donnés deux foncteurs $F : I \to \mathcal{C}$ et $G : J \to \mathcal{C}$ et leur préfaisceau correspondant φ et ψ avec $I, J \in \mathcal{F}$, on peut former le produit tensoriel de Day de φ et ψ comme le décrit la formule de cofin

$$\varphi \otimes \psi : c \mapsto \int^{c_1 c_2} \mathcal{C}(c, c_1 \otimes c_2) \times \varphi(c_1) \times \psi(c_2).$$

Le système de factorisation sur **Cat** assure que le diagramme correspondant à ce préfaisceau est le foncteur $\otimes \circ (F \times G)$ indexé par $I \times J$ comme le montre le diagramme suivant



On voit donc dans ce cas là que si les colimites de F et de G existent et commutent au tenseur, alors la colimite de $\otimes \circ (F \times G)$ existe et commute au tenseur. Il est donc naturel de considérer la catégorie $\overline{\mathcal{C}}$ des préfaisceaux ayant une colimite qui commute au tenseur. La seule chose qui reste à faire dans les cas concrets est de vérifier que le distributeur sur lequel on calcule le représentant pour obtenir l'extension de Kan se factorise bien par cette catégorie.

3.5 Calcul du monoïde libre

Nous allons maintenant appliquer notre théorie aux cas du calcul du monoïde libre. Comme nous l'avons indiqué en introduction de ce chapitre, cela revient à calculer le Δ-modèle libre sur un $\mathbb{N}$-modèle dans le cadre des théories linéaires. Les deux théories $\mathbb{N}$ et

Δ proviennent d'opérades et le morphisme d'inclusion

$$j_{\mathbb{N}} \quad : \quad \mathbb{N} \to \Delta$$

est évidemment opéradique. Tout ce qu'il reste à faire est de s'assurer que les colimites que l'on calcule dans la catégorie monoïdale $\mathcal{C}$ qui nous intéresse existent et commutent au tenseur (nous dirons colimite monoïdale par la suite).

3.5.1 Une vérification simplifiée

Soit c un objet d'un catégorie monoïdale $\mathcal{C}$ et $F : \mathbb{N} \to \mathcal{C}$ le modèle associé. Rappelons que Fn vaut $c^{\otimes n}$ pour tout n de $\mathbb{N}$. Il faut vérifier que le distributeur $F_* \circ j_{\mathbb{N}}^*$ se factorise par la catégorie $\overline{C}$ des préfaisceaux ayant une colimite monoïdale. À première vue, on doit regarder si le préfaisceau

$$F_* \circ j_{\mathbb{N}}^*(-, n) = \int^{k \in \mathbb{N}} \Delta(Jk, n) \times \mathcal{C}(-, c^{\otimes k})$$

a une colimite monoïdale, et ce pour tout $n \in \Delta$. Comme le distributeur $F_* \circ j_{\mathbb{N}}^*$ est monoïdal fort, il suffit d'étudier le cas $n = 1$ car ensuite l'isomorphisme

$$F_* \circ j_{\mathbb{N}}^*(-, n) \cong F_* \circ j_{\mathbb{N}}^*(-, 1)^{\otimes n}$$

nous assure que les autres préfaisceaux ont aussi une colimite monoïdale. Mais l'objet 1 est terminal dans la catégorie Δ, donc l'expression du préfaisceau $F_* \circ j_{\mathbb{N}}^*(-, 1)$ se simplifie drastiquement en

$$F_* \circ j_{\mathbb{N}}^*(-, 1) = \operatorname*{colim}_{k \in \mathbb{N}} \ \mathcal{C}(-, c^{\otimes k}).$$

On a vu à la Section 1.2.5 que le diagramme correspondant à un préfaisceau de ce type est précisément le foncteur $F : \mathbb{N} \to \mathcal{C}$. On déduit de tout ce raisonnement la proposition suivante.

Proposition 3.43 Soit $\mathcal{C}$ une catégorie monoïdale et c un objet de $\mathcal{C}$. Si la colimite du diagramme

$$F : \begin{array}{ccc} \mathbb{N} & \to & \mathcal{C} \\ n & \mapsto & c^{\otimes n} \end{array}$$

existe et commute au tenseur, alors elle définit le monoïde libre sur c.

On vient donc de redémontrer en passant par les équipements en distributeurs que les Équations (3.1) et (3.4) définissent bien respectivement l'algèbre libre dans k-Mod et le monoïde libre dans **Ens**. Mais maintenant que nous avons la théorie générale, nous pouvons aussi recomprendre des cas plus compliqués et moins connus.

3.5.2 Un diagramme comme colimite de diagrammes

Divers constructions de monoïdes libres [Dub74, Val04, Lac08] font apparaître le besoin de savoir que la colimite d'un diagramme est monoïdale à partir de propriétés générales de commutation de certains types de colimites. Par exemple, une hypothèse courante est que la catégorie considérée à les coégaliseurs et que ceux-ci commutent aux tenseurs. Nous présentons ici un moyen de s'assurer qu'un diagramme à une colimite monoïdale en montrant que c'est une colimite monoïdale de diagrammes qui ont eux-mêmes une colimite monoïdale. Plus formellement, on a la proposition suivante.

Proposition 3.44 Soit C une catégorie monoïdale et $\overline{C}$ la catégorie des diagrammes sur C ayant une colimite monoïdale. Supposons que les diagrammes ayant pour base la catégorie J ont tous une colimite monoïdale. Alors la catégorie $\overline{C}$ est close pour les colimites indexées sur J.

Démonstration : Considérons un diagramme $F : J \to \overline{C}$ indexé par J. Nous devons montrer que la colimite de ce diagramme existe dans $\overline{C}$.

Soit $\mathrm{colim} : \overline{C} \to C$ le foncteur qui associe à tout diagramme de $\overline{C}$ sa colimite dans C et $\iota : \overline{C} \to \widehat{C}$ l'injection de $\overline{C}$ dans la catégorie des préfaisceaux sur C. Par hypothèse sur J, le diagramme $\mathrm{colim} \circ F$ à une colimite monoïdale dans C, notée c. Comme $\widehat{C}$ est la complétion libre de C par colimite, le diagramme $\iota \circ F$ à une colimite φ dans $\widehat{C}$. Il suffit donc de montrer que le préfaisceau φ est représenté par c. Soit d un objet de C et $y : C \to \overline{C}$ la plongement de Yoneda restreint à $\overline{C}$. La chaîne d'équivalence suivante

$$\begin{array}{rcll} \widehat{C}(\varphi, \iota \circ y(d)) & \cong & \widehat{C}(\iota \circ F, \iota \circ y(d)) & \varphi \text{ est la colimite de } \iota \circ F \\ & \cong & \overline{C}(F, y(d)) & \iota \text{ est pseudomonique} \\ & \cong & C(\mathrm{colim} \circ F, d) & \mathrm{colim} \vdash y \\ & \cong & C(c, d) & c \text{ est la colimite de } \mathrm{colim} \circ F \end{array}$$

montre que c est un représentant de φ. On en déduit que la colimite de φ existe et est monoïdale, autrement dit $\varphi \in \overline{C}$. ■

Cette proposition nous donne un nouveau point de vue sur les constructions d'Eduardo Dubuc et de Bruno Vallette (raffinée par Steve Lack).

3.5.3 La construction de Dubuc

Dans un papier de 1974 [Dub74], Eduardo Dubuc donne une construction du monoïde libre sur un objet pointé pour une catégorie monoïdale $\mathcal{C}$. Cette construction fait intervenir un processus transfini qui sort du cadre que nous nous sommes fixé. Nous nous contentons de donner ici notre interprétation des idées que Dubuc avait développées.

Tout d'abord, nous devons introduire la théorie linéaire des objets pointés. Celle-ci n'est autre que la catégorie Δ_{face} des ensembles simpliciaux augmentés et fonctions injectives. C'est en particulier la sous-catégorie de Δ dans laquelle on n'a gardé que les morphismes non dégénérés. Nous notons $d_i^n : n-1 \to n$ le morphisme de face correspondant

$$d_i^n : \{1, \ldots, n-1\} \mapsto \{1, \ldots, i-1, i+1, \ldots, n\}.$$

Ainsi, on peut voir tout objet pointé $p : 1 \to A$ de C comme une foncteur fort $A : \Delta_{\mathrm{face}} \to \Delta$ qui à n associe $A^{\otimes n}$ et

$$A(d_i^n) = \mathrm{id}_A \otimes \cdots \otimes p \otimes \ldots \mathrm{id}_A.$$

Encore une fois, cette théorie linéaire est décrite par une opérade et le morphisme

$$j_\Delta \ : \ \Delta_{\mathrm{face}} \ \to \ \Delta$$

est un morphisme d'opérades. Comme à la Proposition 3.43, pour obtenir le monoïde libre sur un objet pointé $A : \Delta_{\mathrm{face}} \to C$, il suffit donc de vérifier que la colimite de A existe et commute au tenseur. C'est le cas dans le cas général suivant.

Proposition 3.45 Soit C une catégorie monoïdale telle que
- les coégaliseurs existent et commutent au tenseur ;
- les colimites séquentielles (au sens de colimites de diagrammes indexés sur $\mathbb{N}$) existent et commutent au tenseur.

Alors le monoïde libre sur un objet pointé $A : \Delta_{\text{face}} \to C$ existe et est calculé par l'image en 1 de l'extension de Kan à gauche de A sur j_Δ :

$$\mathrm{Lan}_{j_\Delta} A(1) = \operatorname*{colim}_{n \in \Delta_{\text{face}}} A(n)$$

Démonstration : Il suffit de vérifier que le diagramme A vit dans $\overline{C}$. Posons $\Delta_{\text{face}}(n)$ la sous-catégorie pleine de Δ_{face} des entiers inférieurs ou égaux à n. Nous nous assurons dans un premier temps que la colimite du diagramme de la restriction de A à $\Delta_{\text{face}}(n)$

$$A_n \quad : \quad 1 \longrightarrow A \underset{A(d_2^2)}{\overset{A(d_1^2)}{\rightrightarrows}} A^{\otimes 2} \underset{A(d_3^3)}{\overset{A(d_1^3)}{\rightrightarrows}} \cdots \underset{A(d_n^n)}{\overset{A(d_1^n)}{\rightrightarrows}} A^{\otimes n}$$

vit dans $\overline{C}$. Remarquons que ce diagramme à la même colimite que le diagramme :

$$D_n \quad : \quad A^{\otimes n-1} \underset{A(d_n^n)}{\overset{A(d_1^n)}{\rightrightarrows}} A^{\otimes n}$$

Remarquons aussi que ce diagramme se calcule en coégalisant les morphismes deux-à-deux. En effet, la colimite des deux flèches du haut est donnée par l'égaliseur $e_1 : A^{\otimes 2} \to A_2$ des deux faces d_1^2 et d_2^2 tensorisé avec $A^{\otimes n-2}$

$$A^{\otimes n-1} \underset{A(d_2^n)}{\overset{A(d_1^n)}{\rightrightarrows}} A^{\otimes n} \xrightarrow{e_1 \otimes A^{\otimes n-2}} A_2 \otimes A^{\otimes n-2}$$

car les coégaliseurs sont monoïdaux. Ensuite, on calcule le coégaliseur du morphisme $m_1 = (e_1 \otimes A^{\otimes n-2}) d_1^n = (e_1 \otimes A^{\otimes n-2}) \circ d_2^n$ et du morphisme $m_3 = (e_1 \otimes A^{\otimes n-2}) d_3^n$

$$A^{\otimes n-1} \underset{m_3}{\overset{m_1}{\rightrightarrows}} A_2 \otimes A^{\otimes n-2} \xrightarrow{e_2 \otimes A^{\otimes n-3}} A_3 \otimes A^{\otimes n-3} \ .$$

Et on recommence jusqu'à obtenir la colimite monoïdale du diagramme D_n ou de manière équivalente du diagramme A_n. Donc A_n est dans $\overline{C}$ pour tout n. On conclut en remarquant que A est la colimite séquentielle du foncteur

$$\mathbb{N} \to \overline{C} : n \mapsto A_n.$$

et en utilisant la Proposition 3.44. ■

On peut raffiner ce théorème lorsque la catégorie $\mathcal{C}$ est munie des coproduits finis car il est alors possible de calculer l'objet pointé libre sur un objet. L'objet pointé libre est simplement défini par

$$A \oplus 1.$$

Proposition 3.46 Soit C une catégorie monoïdale munie des coproduits finis et telle que
- les coégaliseurs existent et commutent au tenseur ;
- les colimites séquentielles (au sens de colimites de diagrammes indexés sur $\mathbb{N}$) existent et commutent au tenseur.

Alors le monoïde libre sur un objet $A : \mathbb{N} \to C$ existe et est calculé par l'image en 1 de l'extension de Kan à gauche de l'objet pointé $A \oplus 1$ sur j_Δ.

Démonstration : Il existe un foncteur d'inclusion $j'_{\mathbb{N}} : \mathbb{N} \to \Delta_{\text{face}}$ qui factorise $j_{\mathbb{N}}$ par j_Δ. Comme les extensions de Kan composent, on en déduit que le monoïde libre sur un objet pointé libre sur un objet A est aussi le monoïde libre sur cet objet A. D'après la Proposition 3.45, il suffit de montrer que $A \oplus 1$ est l'objet pointé libre sur A.

Soit $f : A \to P$ un morphisme de $\mathcal{C}$ vers un objet pointé (P, p). On définit le morphisme d'objet pointé $f \oplus p : (A \oplus 1, i_2) \to (P, p)$. Ce morphisme convient et est unique par universalité de la construction du coproduit. ■

Remarquons que lorsque la catégorie $\mathcal{C}$ à les coproduits finis mais que ceux-ci ne commutent pas au tenseur, la construction de l'objet pointé libre n'est pas obtenu par extension de Kan à gauche dans $\mathbf{Cat}_2$. En effet, par extension de Kan à gauche, on obtient le foncteur $\mathrm{Lan}_{j'_{\mathbb{N}}} A$ qui n'est pas monoïdal comme le montre l'inéquation ci-dessous

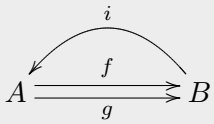
$$\mathrm{Lan}_{j'_{\mathbb{N}}} A(2) = 1 \oplus A \oplus A \oplus A^2 \ncong (A \oplus 1) \otimes (A \oplus 1) = \mathrm{Lan}_{j'_{\mathbb{N}}} A(1) \otimes \mathrm{Lan}_{j'_{\mathbb{N}}} A(1).$$

Ainsi, la Proposition 3.46 combine la « construction à la main » d'une extension de Kan à gauche monoïdale sur $j'_{\mathbb{N}} : \mathbb{N} \to \Delta_{\mathrm{face}}$ et le calcul d'une extension de Kan à gauche sur $j_\Delta : \Delta_{\mathrm{face}} \to \Delta$ dont on sait pour des raisons abstraites qu'elle est aussi monoïdale.

3.5.4 La construction de Vallette/Lack

Bruno Vallette remarque dans son papier [Val04] que l'hypothèse que les coégaliseurs commutent au tenseur est souvent trop forte. Il indique des situations où l'on a uniquement les coégaliseurs réflexifs qui commutent au tenseur et montre que l'on peut toujours calculer le monoïde libre. Rappelons d'abord rapidement ce qu'est un coégaliseur réflexif.

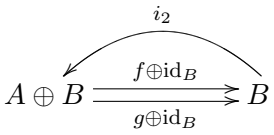
Définition 3.47 (coégaliseur réflexif)
Une paire de morphismes $f, g : A \to B$ est dite *réflexive* lorsqu'il existe un morphisme $i : B \to A$ qui fait commuter le diagramme suivant



Un *coégaliseur réflexif* est un coégaliseur calculé sur une paire réflexive.

Lorsque la catégorie que l'on considère est munie des coproduits finis, il est bien connu que l'on peut remplacer toute paire par une paire réflexive calculant le même coégaliseur.

Proposition 3.48 Soit $\mathcal{C}$ une catégorie ayant les coproduits finis (notés $\oplus$) et $f, g : A \to B$ une paire de morphismes de $\mathcal{C}$. Le coégaliseur de f et g existe si et seulement si le coégaliseur réflexif de la paire réflexive



existe (i_2 est la deuxième injection).

On voit donc se dessiner une variante de la Proposition 3.45 où le calcul de A_n par coégaliseurs est remplacé par un calcul par coégaliseurs réflexifs. C'est précisément le résultat de Bruno Vallette ensuite simplifié par Steve Lack.

Proposition 3.49 Soit C une catégorie monoïdale munie des coproduits finis et telle que
- les coégaliseurs réflexifs existent et commutent au tenseur ;
- les colimites séquentielles (au sens de colimites de diagrammes indexés sur $\mathbb{N}$) existent et commutent au tenseur.

Alors le monoïde libre sur un objet $A : \mathbb{N} \to C$ existe et est calculé par l'image en 1 de l'extension de Kan à gauche de l'objet pointé $A \oplus 1$ sur j_Δ.

Démonstration : On reprend la preuve des Propositions 3.45 et 3.46 avec les mêmes notations. La seule différence est qu'on ne peut pas utiliser de coégaliseurs généraux pour calculer la colimite du diagramme A_n et de manière équivalente D_n. Qu'à cela ne tienne, nous allons utiliser la Proposition 3.48 pour calculer la colimite de D_n avec des coégaliseurs réflexifs. Le premier coégaliseur calculé peut être remplacé par le coégaliseur réflexif monoïdal suivant

$$A^{n-1} \xrightarrow{i_1} (A \oplus A^2) \otimes A^{n-2} \underset{(A(d_2^2)\oplus \mathrm{id}_{A^2})\otimes \mathrm{id}_{A^{n-2}}}{\overset{(A(d_1^2)\oplus \mathrm{id}_{A^2})\otimes \mathrm{id}_{A^{n-2}}}{\rightrightarrows}} A^2 \otimes A^{n-2} \xrightarrow{e_1 \otimes A^{n-2}} A_2 \otimes A^{n-2}$$

avec $i_2 \otimes \mathrm{id}_{A^{n-2}}$: $A^{n-2} \to (A \oplus A^2) \otimes A^{n-2}$.

Et ainsi de suite. Le reste de la preuve est inchangé. ■

3.6 Calcul du comonoïde commutatif libre

3.6.1 Passer d'une théorie à sa théorie duale

Comme nous l'avons indiqué au Chapitre 2, nous sommes avant tout intéressé par le calcul des comonoïdes commutatifs libres qui correspondent à la modalité exponentielle de la logique linéaire. Pour cela, nous devons résoudre dans un premier temps un petit problème : depuis le début de ce chapitre, nous traitons des monoïdes mais jamais des comonoïdes. Cette difficulté apparente n'est pas très sérieuse car un comonoïde dans un catégorie monoïdale $\mathcal{C}$ n'est autre qu'un monoïde dans la catégorie duale $\mathcal{C}^{\mathrm{op}}$ munie du produit tensoriel induit.

On peut donc réutiliser tout ce qui a été fait pour le calcul des monoïdes juste en dualisant tous les diagrammes obtenus et en calculant des limites à la place de colimites. Énonçons en guise d'exemple un corollaire direct de la Proposition 3.45.

Corollaire 3.50 Soit C une catégorie monoïdale telle que
- les égaliseurs existent et commutent au tenseur ;
- les limites séquentielles (au sens de limites de diagrammes indexés sur $\mathbb{N}$) existent et commutent au tenseur.

Alors le comonoïde libre sur un objet copointé $A : {\Delta_{\mathrm{face}}}^{\mathrm{op}} \to C$ existe et est calculé par l'image en 1 de l'extension de Kan à droite de A sur ${j_\Delta}^{\mathrm{op}}$:

$$\mathrm{Ran}_{{j_\Delta}^{\mathrm{op}}} A(1) = \lim_{n \in {\Delta_{\mathrm{face}}}^{\mathrm{op}}} A(n)$$

Comme nous nous intéressons au cas commutatif, nous devons changer de cadre et considérer maintenant les théories symétriques (PROPs). Nous avons mentionné à la Section 1.4.3 que la la catégorie Bij des ensembles finis et des bijections est la théorie triviale des PROPs. De même, la catégorie FinSet des ensembles finis et fonctions ensemblistes est la théorie des monoïdes commutatifs.

L'injection de Bij dans FinSet étant opéradique, on obtient directement la proposition suivante

Proposition 3.51 Soit $\mathcal{C}$ une catégorie monoïdale symétrique algébriquement cocomplète. Le comonoïde commutatif libre $\underset{e}{!}A$ sur un objet A de $\mathcal{C}$ vu comme un foncteur monoïdale symétrique de Bij dans $\mathcal{C}$ est alors obtenu par le formule suivante

$$\underset{e}{!}A = \lim_{n \in \mathrm{Bij}} A^n$$

Exemple 3.52
Cette construction s'applique à la catégorie **Rel** des ensembles et relations. On obtient bien évidemment la construction de l'ensemble des multiensembles finis sur un ensemble

$$!_e A = \coprod_{n \in \mathrm{Bij}} A^n / \sim_{\mathrm{Bij}},$$

ce qui donne l'exponentiel usuel du modèle relation de la logique linéaire.

Nous allons maintenant calculer des comonoïdes commutatifs dans des catégories où les limites – et en particulier les produits – ne commutent pas au tenseur. Comme pour les constructions de Dubuc, Vallette et Lack présentées ci-dessus, nous avons aussi besoin de la théorie des objets pointés dans ce cadre. Elle correspond ici à la catégorie Inj des ensembles finis et des injections.

3.6.2 Calcul dans les espaces de cohérence

Reprenons le modèle des espaces cohérents considéré à la Section 2.3.6. Le lecteur qui souhaiterait une définition des structures plus directe que celle donnée à cette section est prié de se rapporter au papier fondateur [Gir87]. Il est bien connu que le produit de deux espaces cohérents $X \,\&\, Y$ ne distribue pas avec le tenseur. Pour calculer le comonoïde commutatif sur espace cohérent X, il faut d'abord passer à son objet copointé libre. Nous allons utiliser une propriété similaire à la Proposition 3.46 où nous avions calculé le monoïde libre en calculant l'objet pointé libre puis le monoïde libre sur cet objet pointé. Dans le cas monoïdal, l'objet pointé libre sur A est $A \oplus 1$. Dualement dans le cas comonoïdal, l'objet copointé libre sur A est donné par $A \,\&\, 1$. L'intuition dans les espaces de cohérence est que $A \,\&\, 1$ représente les multicliques de A à au plus 1 élément, la multiclique vide étant donnée par l'unique élément de 1. Il suffit alors de montrer que la limite du diagramme

$$1 \xleftarrow[i_2]{} (A \,\&\, 1) \underset{i_2 \otimes \mathrm{id}_{(A\&1)}}{\overset{\mathrm{id}_{(A\&1)} \otimes i_2}{\lessdot\!\!\lessdot}} (A \,\&\, 1)^2 \overset{\sigma}{\circlearrowright} \underset{i_2 \otimes \mathrm{id}_{(A\&1)^2}}{\overset{\mathrm{id}_{(A\&1)^2} \otimes i_2}{\lessdot\!\!\lessdot}} \cdots (A \,\&\, 1)^n \overset{\sigma}{\circlearrowright} \cdots$$

existe et commute au tenseur. Comme nous l'avons montré à la Section 3.5.2, nous pouvons scinder le travail en deux : d'abord en égalisant les morphismes provenant de la symétrie du produit tensoriel, ensuite en calculant la limite du diagramme séquentiel résultant.

La limite A_n du diagramme

$$(A \,\&\, 1)^n \underset{\mathrm{id}_{(A\&1)^{n-2}} \otimes \sigma}{\overset{\sigma \otimes \mathrm{id}_{(A\&1)^{n-2}}}{\lessdot\!\!\lessdot}} (A \,\&\, 1)^n$$

est donnée par l'ensemble des multicliques à au plus n éléments, deux multicliques étant cohérentes si leur union est encore une multiclique. Il est aisé de vérifier que cette limite est monoïdale.

On obtient alors un nouveau diagramme

$$A_0 = 1 \xleftarrow[i_2]{} A_1 = (A \,\&\, 1) \longleftarrow A_2 \longleftarrow \cdots A_n \cdots$$

dont toutes les flèches sont des relations d'inclusion (dualisée) de A_{n-1} dans A_n. La limite $!_e A$ de ce diagramme est simplement donnée par l'ensemble des multicliques finies, deux multicliques étant cohérentes si leur union est encore une multiclique. Encore une fois il n'est pas difficile de voir que cette limite est monoïdale. On en déduit la proposition suivante.

Proposition 3.53 Le comonoïde commutatif libre $!_e A$ sur un espace cohérent A est donné par l'ensemble des multicliques finies, deux multicliques étant cohérentes si leur union est encore une multiclique.

3.6.3 Calcul dans les jeux de Conway

Nous allons maintenant calculer le comonoïde commutatif libre sur les jeux de Conway négatifs (voir la Définition 2.27) grâce à notre théorie générale. Cette catégorie est munie des produits finis mais ceux ci ne commutent pas au tenseur. Là encore, nous allons plutôt regarder le comonoïde commutatif libre sur un objet copointé. Mais comme tout jeu de Conway négatif est automatiquement pointé, avec comme stratégie $t_A : A \to 1$ la stratégie vide qui ne répond pas ; on aura du même coup le comonoïde commutatif libre sur un objet quelconque.

En résumé, il nous suffit de calculer la limite du diagramme

$$\mathcal{A} \quad : \quad 1 \xleftarrow[t_A]{} A \underset{t_A \otimes \mathrm{id}_A}{\overset{\mathrm{id}_A \otimes t_A}{\leftleftarrows}} A^2 \overset{\sigma}{\circlearrowright} \underset{t_A \otimes \mathrm{id}_{A^2}}{\overset{\mathrm{id}_{A^2} \otimes t_A}{\leftleftarrows}} \cdots A^n \overset{\sigma}{\circlearrowright} \cdots$$

et de montrer que cette limite est monoïdale. Nous prétendons que la limite de $\mathcal{A}$ est le jeu $!_e A$ – qui peut être regardé comme le tenseur infini de A – défini comme suit

- ses positions sont les mots $w = x_1 \cdots x_k$ dont les lettres sont des positions x_i du jeu A différentes de la racine ; l'intuition est que la lettre x_i décrit la position courante dans la i-ème copie de A ;
- sa racine $\star_{!A}$ est le mot vide ;
- ses coups $m : w \to w'$ sont ou bien des coups joués dans une copie :

$$w_1 \cdot x \cdot w_2 \xrightarrow{m} w_1 \cdot y \cdot w_2$$

où $\underline{m} : x \to y$ est un coup du jeu A ; ou bien des coups où Opposant ouvre une nouvelle copie :

$$w \xrightarrow{m} w \cdot x$$

où $\underline{m} : \star_A \to x$ est un coup initial de A. Si m est un coup de $!_e A$, on note $\underline{m}$ le coup sous-jacent dans A.

La polarité des coups est directement héritée de celle des coups du jeu A de la même manière que pour le tenseur.

Au lieu de montrer en deux étapes que $!_e A$ est la limite du diagramme $\mathcal{A}$ et que cette limite commute au tenseur, on va directement montrer que $X \otimes !_e A$ est la limite du diagramme

$$X \otimes \mathcal{A} \quad : \quad X \xleftarrow[t_A]{} X \otimes A \underset{\mathrm{id}_X \otimes t_A \otimes \mathrm{id}_A}{\overset{\mathrm{id}_X \otimes \mathrm{id}_A \otimes t_A}{\leftleftarrows}} X \otimes A^2 \overset{\mathrm{id}_X \otimes \sigma}{\circlearrowright} \underset{\mathrm{id}_X \otimes t_A \otimes \mathrm{id}_{A^2}}{\overset{\mathrm{id}_X \otimes \mathrm{id}_{A^2} \otimes t_A}{\leftleftarrows}} \cdots X \otimes A^n \overset{\mathrm{id}_X \otimes \sigma}{\circlearrowright} \cdots$$

On obtiendra ensuite la propriété sur $!_e A$ en faisant $X = 1$.

Proposition 3.54 $X \otimes !_e A$ est la limite du diagramme $X \otimes \mathcal{A}$

Démonstration : Nous devons dans un premier temps définir un cône d'origine $X \otimes !_e A$ sur le diagramme $X \otimes \mathcal{A}$. On procède comme à la Section 2.4.3 en définissant une fonction

d'entrelacement des parties de $!_e A$ vers les parties de A^n, puis en définissant une stratégie d'imitation. Étant donnée une partie $s \cdot m$ de $!_e A$, on définit

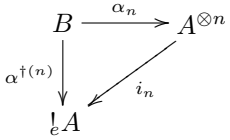
$$\langle s \cdot m \rangle = \langle s \rangle \cdot \underline{m}$$

où $\underline{m}$ est le coup sous-jacent dans A. On définit alors la stratégie $\varepsilon_n : !_e A \to A^n$ par l'ensemble de parties

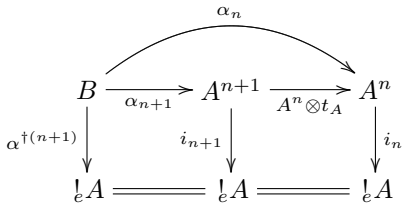
$$\varepsilon_n \quad \stackrel{\text{def}}{=} \quad \{s \in \mathrm{Play}^{even}_{!_e A_1 \multimap A_2^n} \mid \forall t \prec^{even} s \ , \ t_{|!_e A_1} = \langle t_{|A_2^n} \rangle\}$$

Le cône de $X \otimes !_e A$ sur $X \otimes \mathcal{A}$ est alors donné par les morphismes $\mathrm{id}_X \otimes \varepsilon_n$. Remarquons que c'est la présence des symétries qui force l'ordonnancement des ouvertures copies dans le jeu $!_e A$. Voilà pourquoi $!_e A$ n'est pas simplement le produit tensoriel infini du jeu A. Soit maintenant $(B, \alpha : B \to \mathcal{A})$ un cône sur $X \otimes \mathcal{A}$. Nous devons définir une flèche de B dans $X \otimes !_e A$.

Introduisons la stratégie $i_n : X \otimes A^n \to X \otimes !_e A$ qui imite Opposant sur $X \otimes A^{\otimes n}$ pour X et les n premières copies de $!_e A$, et qui ne répond pas lorsque Opposant ouvre la $n+1$-ème copie de $!_e A$. On définit pour tout n la stratégie $\alpha^{\dagger(n)}$ par le diagramme commutatif suivant :



Considérons maintenant le diagramme suivant :



Il commute sur toutes ces faces excepté pour celle en bas à droite qui vérifie $i_n \circ (A^n \otimes t_B) \subseteq i_{n+1}$. Le chemin extérieur dans le sens horaire étant égal à $\alpha^{\dagger(n)}$, on en déduit que

$$\alpha^{\dagger(n)} \subseteq \alpha^{\dagger(n+1)}.$$

On peut donc définir le soulèvement comonoïdal $\alpha^\dagger$ par la limite croissante des $\alpha^{\dagger(n)}$:

$$\alpha^\dagger \quad \stackrel{\text{def}}{=} \quad \bigcup_n \alpha^{\dagger(n)}$$

On doit montrer que cette stratégie est un morphisme de cônes. Pour cela, remarquons que $\varepsilon_n \circ i_n = \mathrm{id}_{A^n}$ et donc que

$$\varepsilon_n \circ \alpha^{\dagger(n)} = \varepsilon_n \circ i_n \circ \alpha_n = \alpha_n.$$

Il reste à montrer que ce morphisme de cônes est unique. Soit β un autre morphisme de cônes de B vers $!_e A$. On définit

$$\beta^{(n)} = i_n \circ \varepsilon_n \circ \beta$$

et on remarque les deux choses suivantes

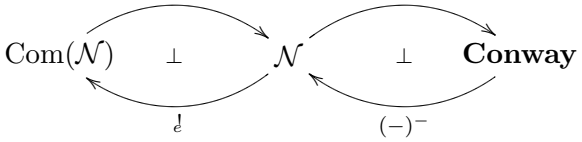
$$\beta^{(n)} = \alpha^{\dagger(n)} \quad \text{et} \quad \beta = \bigcup_n \beta^{(n)}.$$

On en déduit que $\alpha^\dagger = \tau$, ce qui termine la preuve. ∎

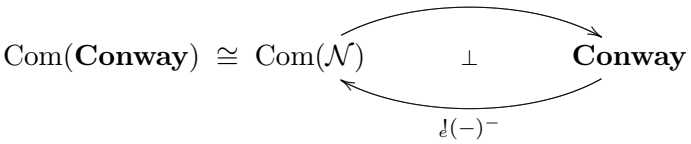
Clairement, on a une propriété symétrique avec le diagramme $\mathcal{A} \otimes X$. En utilisant cette propriété avec $X = 1$, on en déduit que $!_{\mathcal{C}}A$ est la limite du diagramme $\mathcal{A}$. Nous pouvons maintenant énoncer la proposition annoncée plus haut.

Proposition 3.55 Le jeu $!_{\mathcal{C}}A$ est le comonoïde commutatif libre dans $\mathcal{N}$ sur le jeu de Conway négatif A.

Remarquons au passage que l'on obtient le comonoïde commutatif libre sur la catégorie des jeux de Conway toute entière. En effet, la Proposition 3.55 nous indique que $\mathcal{N}$ est en adjonction avec sa sous catégorie des comonoïdes $\mathrm{Com}(\mathcal{N})$. Comme la catégorie $\mathcal{N}$ est aussi en adjonction avec la catégorie des jeux de Conway **Conway**, on obtient la chaîne d'adjonctions suivante :



Maintenant, remarquons que tout jeu de Conway qui est aussi un comonoïde commutatif est négative (l'argument s'obtient en étudiant le diagramme de compatibilité avec la symétrie). En d'autres termes, on a



On en déduit la proposition suivante

Proposition 3.56 Le jeu $!_{\mathcal{C}}A^{-}$ est le comonoïde commutatif libre dans **Conway** sur le jeu de Conway A.

On se rend compte ici de la simplification dans la preuve apportée par la théorie générale par rapport à la preuve donnée à la Section 2.4.3 par les jeux de Conway à gain. Cela vient principalement du fait qu'on n'a pas besoin de définir toute la structure comonoïdale car elle est automatiquement déduite par universalité du calcul des limites.

Chapitre 4

Une sémantique des jeux avec références

La persistance de la mémoire,
Dali (1931)

Sommaire

Les catégories monoïdales tracées [JSV96] ont été introduites par André Joyal, Ross Street et Dominic Verity afin de fournir une description uniforme de divers constructions mathématiques ayant un comportement cyclique. Parmi les constructions les plus notables décrite par les catégories monoïdales tressées, nous citerons la fermeture des tresses en théorie des nœuds et l'opérateur de trace en algèbre linéaire. Elles devinrent rapidement populaires dans la communauté informatique comme un moyen élégant pour exprimer la notion de boucles dans un cadre catégorique. Elles ont été extrêmement fructueuses dans ce champ, que ce soit pour formaliser la formule d'exécution de la Géométrie des Interactions [Abr96, AHS02], pour analyser l'opérateur de point fixe en théorie des domaines [Has02a], ou pour offrir un modèle catégorique en concurrence et plus récemment en physique quantique [AC04].

L'une des premières apparitions de l'opérateur de trace pour modéliser la rétroaction (feedback en anglais) dans un cadre sémantique est due à Robin Milner [Mil94]. Dans ce papier, il présente une façon abstraite de modéliser la rétroaction dans les *calculs de processus* qu'il nomme *réflexion* (le terme trace n'était pas encore à l'ordre du jour). La réflexion lui permet de décrire des opérations compliquées comme le célèbre opérateur de restriction $\nu : \epsilon \to p$ qui devient simplement la trace de la diagonale $\delta_p : p \to p \otimes p$. Rappelons que cet opérateur modélise le fait qu'un canal de communication public peut être soudain restreint pour devenir un canal de communication privée entre les processus qui l'utilisaient déjà. On peut comprendre ce mécanisme comme le passage d'une mémoire globale à une mémoire locale via un phénomène de localisation. Il est à noter que bien que Milner n'avait pas connaissance des travaux récents sur les traces à l'époque où il a défini les réflexions, tous les axiomes qu'il donne pour que son opérateur de restriction conserve les propriétés habituelles (comme le fameux "scope extrusion") coïncident exactement avec l'axiomatique de l'opérateur de trace. Ceci fait de la trace un objet canonique qui semble destiné à interpréter les références.

Ici, nous nous intéressons à la trace comme moyen de description des variables locales dans les langages de programmation. Traditionnellement en sémantique, on interprète un langage de programmation dans une catégorie en distinguant les objets A décrivant les valeurs, des objets TA décrivant les calculs de type A, où T est une monade. Dans le cas des références, la monade considérée est la monade d'état $S \multimap (S \otimes \_)$ qui permet d'interpréter un programme de type $A \to B$ comme un programme prenant une valeur A et renvoyant un calcul $S \multimap (S \otimes B)$, ce qui, via la clôture monoïdale, correspond à un morphisme de $S \otimes A \to S \otimes B$.

Il faut penser cette interprétation comme la description d'un système avec entrée/sortie et mémoire accessible à l'utilisateur ; les deux notions réunies dans le morphisme $f : S \otimes A \longrightarrow S \otimes B$. Dès lors, si on est capable de prendre la trace sur S de f, on obtient un morphisme avec mémoire interne de type $A \longrightarrow B$. C'est l'analogue de la restriction (ou localisation) chez Milner pour un langage où les canaux sont remplacés par des adresses mémoires. Remarquons tout de suite qu'une catégorie monoïdale symétrique fermée $\mathcal{C}$ munie d'une trace est automatiquement coreflexive dans sa catégorie compacte fermée libre $\mathrm{Int}(\mathcal{C})$. Il est donc naturel de travailler directement avec la catégorie compacte fermée.

Par la suite, cette approche va être utilisée pour décrire un modèle d'un langage de programmation avec des références d'ordre supérieur dans les styles des références de ML. Pour cela, il nous faut allier le pouvoir de la logique tensorielle pour décrire l'aspect fonctionnel du langage avec le pouvoir des traces pour décrire l'aspect mémoriel du langage. Comme la logique tensorielle repose sur une gestion linéaire des ressources, nous allons aussi devoir comprendre comment la modalité exponentielle permet de créer des cellules mémoires dans lesquelles on peut écrire et lire autant de fois que l'on veut.

Nous voulons donner un modèle de notre langage dans une catégorie de jeux proche de la catégorie des familles sur les jeux de Conway à gain présentée à la Section 2.4. Il apparaît que cette catégorie est étroitement liée au modèle de jeux introduit par Samson Abramsky, Kohei Honda et Guy McCusker [AHM98] pour donner un modèle pleinement adéquat (*fully abstract* en anglais) d'un langage avec références d'ordre supérieur. Il nous manque seulement une notion de parenthésage afin de contraindre le déroulement d'une interaction. Mais avoir une notion de parenthésage, sans pour autant briser la structure compacte fermée de la catégorie des jeux de Conway, n'est pas une mince affaire. Cela nous a amené à une refonte totale du parenthésage pour aboutir à la notion de *multiparenthésage*. Cette notion repose sur la gestion de requêtes initiées par Joueur et Opposant permettant de préserver la structure autoduale des jeux de Conway.

La catégorie de jeux introduite par Abramsky et al. est isomorphe à la catégorie de Kleisli induite par la modalité de ressource exponentielle sur notre catégorie de jeux multiparenthésés. Nous utiliserons donc directement leur résultat pour obtenir la pleine adéquation de notre modèle. La différence majeure de notre travail est que la stratégie $\mathtt{cell}_A$ d'interprétation d'une cellule mémoire de type A est obtenue graduellement en utilisant la logique tensorielle, la structure compacte fermée, et la présence d'un accès mémoire pour la modalité exponentielle. Cet accès mémoire réduit la gestion des copies réalisée par une cellule mémoire à la présence d'une transformation naturelle de composante

$$\xi_{A,B} \quad : \quad !(A \otimes !B) \longrightarrow !A \otimes !B$$

satisfaisant deux lois de cohérence. Grâce à cette notion, on va construire pas à pas trois types de cellules :

- la *cellule mémoire linéaire* dans laquelle on écrit une seule fois et on lit une seule fois (cette construction utilise la négation tensorielle et la structure compacte fermée) ;
- la *cellule mémoire constante* dans laquelle on écrit une fois et on lit autant de fois que l'on veut (cette construction utilise la modalité exponentielle) ;

– la *cellule mémoire* dans laquelle on écrit et on lit autant de fois que l'on veut (cette construction utilise l'accès mémoire de la modalité exponentielle).

4.1 Préliminaires

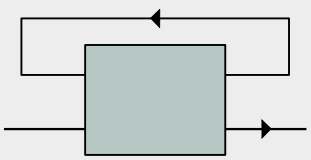
4.1.1 Catégories monoïdales tracées

Nous présentons ici les catégories monoïdales symétriques tracées, les catégories compactes fermées et les liens qui les unissent.

Définition 4.1 (catégorie monoïdale tracée [JSV96])
Une *catégorie monoïdale symétrique tracée* est une catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I, c)$ munie d'une famille naturelle de fonctions, appelée un *opérateur de trace*,

$$\mathrm{Tr}^X_{A,B} \quad : \quad \mathcal{C}_1(X \otimes A, X \otimes B) \;\to\; \mathcal{C}_1(A, B)$$

que nous représenterons graphiquement par



vérifiant les axiomes suivants :

– **Dissipation :**

$$\mathrm{Tr}^I_{A,B}(f) = f : A \to B$$

avec $f : A \to B$, et

$$\mathrm{Tr}^{X\otimes Y}_{A,B}(f) = \mathrm{Tr}^X_{A,B}(\mathrm{Tr}^Y_{A\otimes X, B\otimes X}(f)) : A \to B$$

avec $f : A \otimes X \otimes Y \to B \otimes X \otimes Y$;

– **Superposition :**

$$\mathrm{Tr}^X_{C\otimes A, C\otimes B}(\mathrm{id}_C \otimes f) = \mathrm{id}_C \otimes \mathrm{Tr}^X_{A,B}(f) : C \otimes A \to C \otimes B$$

avec $f : A \otimes X \to B \otimes X$;

– **Étirement :**

$$\mathrm{Tr}^X_{X,X}(c_{XX}) = \mathrm{id}_X : X \to X.$$

Nous omettrons parfois les indices lorsqu'aucune confusion n'est possible. Pour aider le lecteur à se créer une intuition sur les différents axiomes requis pour une trace, et aussi pour insister sur le lien entre trace et rétroaction, nous présentons Figure 4.1 une version graphique de ces axiomes.

Nous mentionnons un exemple important en informatique théorique de catégorie symétrique monoïdale tracée car c'est un modèle de base de la logique linéaire.

Exemple 4.2
Soit la catégorie **Rel** des ensembles avec relations, munie du produit tensoriel défini sur les objets comme le produit cartésien des ensembles et sur les relations par $\langle x, y\rangle(R \times$

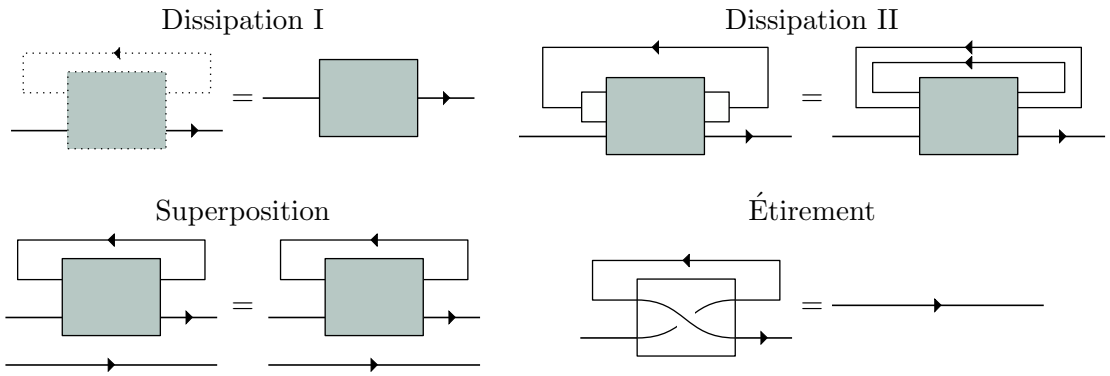


FIG. 4.1 – Les axiomes définissant l'opérateur de trace

$R')\langle x', y'\rangle$ si et seulement si xRy et $x'Ry'$. Remarquons que ceci ne définit pas un produit au sens catégorique. Pour $R : X \times A \to X \times B$, on définit $Tr_X R : A \to B$ via :

$$a(Tr_X R)b \quad \text{ssi} \quad \exists x \in X.\ \langle x, a\rangle R\langle x, b\rangle$$

On en déduit que $(\mathbf{Rel}, \times, Tr)$ est une catégorie tracée.

4.1.2 Catégories compactes fermées

Si l'on voit la notion de trace comme la généralisation d'un monoïde simplifiable (à gauche), la notion de catégorie compacte [KL80] fermée est alors la généralisation d'un groupe.

Pour illustrer ce concept dans l'univers mathématique, citons comme exemple frappant la catégorie des espaces vectoriels avec le produit tensoriel usuel. Dans cette catégorie monoïdale symétrique fermée, l'opérateur de clôture possède la propriété remarquable d'avoir lui aussi une structure tensorielle. Un autre exemple naturel nous est donné par les catégories modèles de logique linéaire dans lesquelles $\otimes = \parr$. Dans ces catégories, on sait directement que le tenseur est autodual car

$$(A \otimes B)^* = A^* \parr B^* = A^* \otimes B^*$$

Pour définir les catégories compactes fermées, nous avons besoin de la notion d'objet dual.

Définition 4.3 (objet dual)
Un objet A^* dans une catégorie monoïdale symétrique $(\mathcal{C}, \otimes, I, c)$ est le *dual* d'un objet A s'il est adjoint à droite à A dans la suspension bicatégorique de $\mathcal{C}$ (i.e. $\mathcal{C}$ vu comme une bicatégorie à un point).

Dans ce cas, il est équipé de deux morphismes appelés *unité* $\eta_A : I \to A^* \otimes A$ et *counité* $\varepsilon_A : A \otimes A^* \to I$ tels que les composées

$$A \xrightarrow{\rho_A^{-1}} A \otimes I \xrightarrow{A\otimes\varepsilon_A} A \otimes (A^* \otimes A) \xrightarrow{\alpha^{-1}_{A,A^*,A}} (A \otimes A^*) \otimes A \xrightarrow{\eta_A\otimes A} I \otimes A \xrightarrow{\lambda_A} A$$

$$A^* \xrightarrow{\lambda^{-1}_{A^*}} I \otimes A^* \xrightarrow{\varepsilon_A\otimes A^*} (A^* \otimes A) \otimes A^* \xrightarrow{\alpha_{A^*,A,A^*}} A^* \otimes (A \otimes A^*) \xrightarrow{A^*\otimes\eta_A} A^* \otimes A \xrightarrow{\rho_{A^*}} A^*$$

sont des identités.

Définition 4.4 (catégorie compacte fermée [KL80])
Une *catégorie compacte fermée* est une catégorie monoïdale symétrique $\mathcal{C}$ dans laquelle chaque objet A est muni d'un *objet dual*. Cet objet unique à isomorphisme près est noté A^*.

Il est facile de vérifier que toute catégorie compacte fermée est automatiquement monoïdale fermée, la fermeture étant donnée par

$$A \multimap B \cong A^* \otimes B$$

De façon plus subtile, toute catégorie compacte fermée admet un opérateur de trace.

Proposition 4.5 ([JSV96]) Soit $(\mathcal{C}, \otimes, I)$ une catégorie compacte fermée. À tout morphisme $f : X \otimes A \to X \otimes B$, on peut associer un morphisme

$$\mathrm{Tr}^X_{A,B} : A \xrightarrow{\alpha^{-1} \circ (A \otimes \eta_{C^*}) \circ \rho_B^{-1}} X^* \otimes (X \otimes A) \xrightarrow{f \otimes C^*} X^* \otimes (X \otimes B) \xrightarrow{\rho_B \circ (B \otimes \varepsilon_C) \circ \alpha} B$$

qui fait de $\mathcal{C}$ une catégorie monoïdale tracée.

Il est bien plus simple de définir un catégorie compacte fermée et d'en déduire ensuite l'opérateur de trace. Mais n'oublions pas que nous voulons utiliser l'opérateur de trace pour modéliser les références et l'on doit se demander si se restreindre n'est pas trop fort au sens où l'on oublierait au passage certaines catégories primordiales pour la description des références.

Un première réponse à cette question est la construction *Int* due à [JSV96] de la catégorie compacte libre engendrée par un catégorie tracée. Cela dit que l'on peut toujours voir une catégorie tracée comme une version relâchée d'une catégorie compacte fermée. Cette construction est à la base de la notion de polarité en logique.

Définition 4.6 (construction Int)
Soit $\mathcal{C}$ une catégorie tracée. On définit $\mathrm{Int}(\mathcal{C})$ comme la catégorie compacte fermée ayant pour objet les couples (A^+, A^-) d'objets de $\mathcal{C}$ et pour morphismes entre (A^+, A^-) et (B^+, B^-) les flèches $A^+ \otimes B^- \longrightarrow A^- \otimes B^+$ dans $\mathcal{C}$. La composition est définie à l'aide de la trace (voir le Diagramme 4.2). Le dual est juste l'inversion de la polarité $(A^+, A^-)^* = (A^-, A^+)$et le produit tensoriel est défini point à point

$$(A^+, A^-) \otimes (B^+, B^-) = (A^+ \otimes B^+, A^- \otimes B^-)$$

Du point de vue des groupes et des monoïdes simplifiables, cette construction correspond simplement au groupe libre sur un monoïde, l'objet $(x, 1)$ représentant x et l'objet $(1, x)$ représentant son inverse x^{-1}.

Proposition 4.7 ([JSV96]) Le foncteur $J : \mathcal{C} \to \mathrm{Int}(\mathcal{C})$ qui à A associe (A, I) est monoïdal fort, plein et fidèle et préserve la trace. Ce foncteur est l'unité de l'adjonction entre la 2-catégorie des catégories monoïdales symétriques tracées et la 2-catégorie des catégories compactes fermées.

Remarquons que cette construction est au cœur de la catégorisation des modèles de GoI donnée par Samson Abramsky [AHS02]. La composition dans la catégorie $\mathrm{Int}(C)$ représentant la formule d'exécution.

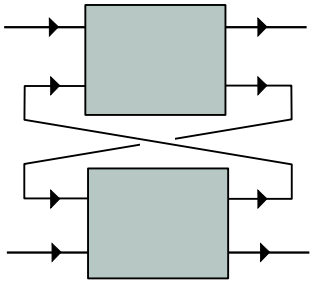


FIG. 4.2 – Composition dans la catégorie Int($\mathcal{C}$)

4.1.3 Trace et fermeture

Dans ce chapitre, nous souhaitons modéliser les références d'un langage de programmation à l'aide de trace. Comme nous voulons de l'ordre supérieur, la catégorie dans laquelle nous allons modéliser notre langage sera donc à la fois fermée et tracée. La Proposition 2.32 établit qu'une catégorie monoïdale symétrique tracée $\mathcal{C}$ hérite directement la fermeture de Int($\mathcal{C}$). C'est un premier résultat en faveur de l'étude des catégories compactes fermées, mais il manque une sorte de réciproque à cette propriété, qui assurerait que si une catégorie tracée $\mathcal{C}$ est fermée, alors cette fermeture est étroitement liée à Int($\mathcal{C}$). Nos discussions avec Masahito Hasegawa [Has05] sur les relations qu'entretiennent la catégorie des jeux de Conway négatifs $\mathcal{N}$ et sa "compactifiée" Int($\mathcal{N}$) l'ont amené à préciser la situation dans le cadre des catégories tracées.

Proposition 4.8 Soit $\mathcal{C}$ une catégorie monoïdale symétrique tracée. $\mathcal{C}$ est fermée si et seulement si $J : \mathcal{C} \to \mathrm{Int}(\mathcal{C})$ admet un adjoint à droite.

Il est donc clair que si une catégorie tracée est fermée, sa fermeture vient directement de la clôture de Int($\mathcal{C}$), c'est à dire de la catégorie compacte fermée sous-jacente. Dès lors, il n'y a plus de raison de s'empêcher de travailler avec une catégorie compacte fermée, quitte à se restreindre légèrement à posteriori.

4.2 IdeaML : un langage avec références locales

4.2.1 Types et termes

Nous présentons ici un langage de programmation avec des références d'ordre supérieur dans les styles des références de ML. Nous reprenons exactement le langage utilisé par Samson Abramsky et al. dans [AHM98]. La grammaire de type est donnée par

$$A, B \quad ::= \quad \mathrm{Unit} \mid \mathrm{Nat} \mid A \to B \mid A \times B \mid \mathrm{ref}[A]$$

Remarque 4.9
Cette restriction des références (qui ne peuvent pas pointer sur d'autres références) permet tout de même de typer des programmes d'ordre supérieur arbitraire. En revanche, on ne rentre pas dans le cadre de l'aliasing, ce qui ferait entrer le langage dans une classe de complexité bien plus grande.

Les termes de **IdeaML** sont inspirés de ML :

$$\begin{aligned} M \quad ::= \quad & \texttt{skip} \mid n \mid x \mid \lambda x.M \mid MM \mid \texttt{ifzero } M \texttt{ then } M \texttt{ else } M \mid \texttt{succ } M \mid \texttt{prec } M \\ & \mid \langle M, M \rangle \mid \pi_1(M) \mid \pi_2(M) \mid M := M \mid !M \mid \texttt{new}_A \mid \texttt{mkvar } MM \end{aligned}$$

$$[Var]\ \frac{}{\Gamma, x : A \vdash x : A} \quad [Bool]\ \frac{}{\Gamma \vdash b : \text{Bool}} \quad [Nat]\ \frac{}{\Gamma \vdash n : \text{Nat}} \quad [Unit]\ \frac{}{\Gamma \vdash \texttt{skip} : \text{Unit}}$$

$$[Abs]\ \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta}\,(x \notin \Gamma) \quad [App]\ \frac{\Gamma \vdash M : \alpha \to \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}$$

$$[Succ]\ \frac{\Gamma \vdash M : \text{Nat}}{\Gamma \vdash \texttt{succ}\ M : \text{Nat}} \quad [Prec]\ \frac{\Gamma \vdash M : \text{Nat}}{\Gamma \vdash \texttt{prec}\ M : \text{Nat}}$$

$$[If]\ \frac{\Gamma \vdash M : \text{Bool} \quad \Gamma \vdash M_i : \alpha (i = 1, 2)}{\Gamma \vdash \texttt{ifzero}\ M\ \texttt{then}\ M_1\ \texttt{else}\ M_2 : \alpha}$$

$$[Pair]\ \frac{\Gamma \vdash M_i : \alpha_i (i = 1, 2)}{\Gamma \vdash \langle M_1, M_2 \rangle : \alpha_1 \times \alpha_2} \quad [Proj]\ \frac{\Gamma \vdash M : \alpha_1 \times \alpha_2}{\Gamma \vdash \pi_i(M) : \alpha_i (i = 1, 2)}$$

$$[New]\ \frac{}{\Gamma \vdash \texttt{new}_A : \text{ref}[A]} \quad [MkVar]\ \frac{\Gamma \vdash M : A \to \text{Unit} \quad \Gamma \vdash M : \text{Unit} \to A}{\Gamma \vdash \texttt{mkvar}\ MN : \text{ref}[A]}$$

$$[Assign]\ \frac{\Gamma \vdash M : \text{ref}[A] \quad \Gamma \vdash N : A}{\Gamma \vdash M := N : \text{Unit}} \quad [Deref]\ \frac{\Gamma \vdash M : \text{ref}[A]}{\Gamma \vdash\ !x : A}$$

FIG. 4.3 – Règles de typage dans **IdeaML**

où n représente un entier et x est une variable. Les règles de typage sont données à la Figure 4.3 où Γ est une contexte de typage qui associe à chaque variable un type.

Remarque 4.10 (Le constructeur `mkvar` et les mauvaises variables)
Le constructeur `mkvar` est là pour palier au problème des *mauvaises variables*. Ce problème, qui a été identifié par John Reynolds [Rey78], vient des valeurs de type ref[A] mais qui n'ont pas un emplacement mémoire. Le constructeur `mkvar` permet alors de créer de mauvaises variables à partir d'une valeur de type $A \to$ Unit (la méthode d'écriture) et d'une valeur de type Unit $\to A$ (la méthode de lecture). Nous ne rentrons pas plus dans les détails ici mais le lecteur pourra trouver de plus amples informations dans [AHM98].

4.2.2 Sémantique à grands pas

Nous allons maintenant donner la sémantique opérationnelle de notre langage. Celle-ci sera donnée pour des triplets (M, L, σ) appelés configurations, où M est un terme, L est un ensemble d'emplacements mémoires et σ est une fonction partielle, que nous appellerons état de la mémoire, qui va des emplacements vers des valeurs du type approprié. Nous faisons le choix de présenter une sémantique *à grands pas*.

Pour cela, il faut une notion de forme canonique, qui sont les termes de la forme :

$$V ::= n \mid b \mid x \mid \texttt{skip} \mid \lambda x.V \mid \langle V_1, V_2 \rangle$$

Lorsque qu'une configuration (M, L, σ) se réduit en une configuration canonique (V, L', σ'), nous notons

$$(M, L, \sigma) \Downarrow (V, L', \sigma')$$

Pour plus de légèreté dans la notation, nous adoptons la convention que

$$\frac{M \Downarrow V \quad M' \Downarrow V'}{M'' \Downarrow V''}$$

est une abréviation pour

$$\frac{(M, L, \sigma) \Downarrow (V, L', \sigma') \quad (M', L', \sigma') \Downarrow (V', L'', \sigma'')}{(M'', L, \sigma) \Downarrow (V'', L'', \sigma'')}$$

La Figure 4.4 décrit inductivement la sémantique opérationnelle de notre langage.

$$\frac{}{(V, L, \sigma) \Downarrow (V, L, \sigma)} \qquad \frac{N \Downarrow V \quad M \Downarrow \lambda x.V'}{MN \Downarrow V'[V/x]} \qquad \frac{M_1 \Downarrow V_1 \quad M_2 \Downarrow V_2}{\langle M_1, M_2 \rangle \Downarrow \langle V_1, V_2 \rangle}$$

$$\frac{M \Downarrow \langle V_1, V_2 \rangle}{\pi_i(M) \Downarrow V_i (i = 1, 2)} \qquad \frac{M \Downarrow V}{\texttt{succ}\ M \Downarrow V + 1} \qquad \frac{M \Downarrow V}{\texttt{prec}\ M \Downarrow V - 1}$$

$$\frac{M \Downarrow 0 \quad M_1 \Downarrow V}{\texttt{ifzero}\ M\ \texttt{then}\ M_1\ \texttt{else}\ M_2 \Downarrow V} \qquad \frac{M \Downarrow n \quad M_2 \Downarrow V \quad (n \neq 0)}{\texttt{ifzero}\ M\ \texttt{then}\ M_1\ \texttt{else}\ M_2 \Downarrow V}$$

$$\frac{}{(\texttt{new}_A, L, \sigma) \Downarrow (l, L \cup (l : A), \sigma)}$$

$$\frac{(M, L, \sigma) \Downarrow (l, L', \sigma') \quad (N, L', \sigma') \Downarrow (V, L'', \sigma'')}{(M := N, L, \sigma) \Downarrow (\texttt{skip}, L'', \sigma''(x \mapsto V))} \qquad \frac{(M, L, \sigma) \Downarrow (l, L', \sigma') \quad \sigma'(l) = V}{(!M, L, \sigma) \Downarrow (V, L', \sigma')}$$

$$\frac{M \Downarrow \texttt{mkvar}\ V_1\ V_2 \quad N \Downarrow V \quad V_1(V) \Downarrow \texttt{skip}}{x := M \Downarrow \texttt{skip}} \qquad \frac{M \Downarrow \texttt{mkvar}\ V_1\ V_2 \quad V_2(\texttt{skip}) \Downarrow V}{!M \Downarrow V}$$

FIG. 4.4 – Sémantique opérationnelle de **IdeaML**

4.3 Un modèle de jeu avec ressource

Nous voulons définir un modèle basé sur les jeux de Conway car ils définissent une catégorie compacte fermée. Mais nous voulons aussi avoir une notion de parenthésage afin de nous restreindre à des stratégies interprétables dans notre calcul. Malheureusement, l'extension aux jeux de Conway à gain proposée à la Section 2.4 ne satisfait pas ces deux conditions car elle brise la structure compacte fermée. Il est donc important de bien comprendre le parenthésage habituellement présenté dans les jeux d'arènes afin de pouvoir l'importer dans nos jeux de Conway sans pour autant perdre la structure compacte fermée et ainsi la notion de trace.

4.3.1 Jeux d'arène et parenthésage

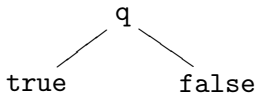
Un *jeu d'arène* est défini comme une forêt d'arbres enracinés dont les nœuds sont appelés les *coups* du jeu. On peut aussi le définir comme un graphe orienté acyclique avec des racines distinguées comme l'a fait Russ Harmer dans sa thèse [Har00]. On écrit

$$m \vdash n$$

et dit que le coup m *justifie* le coup n lorsque le coup m est un ancêtre immédiat du coup n dans l'arène. À chaque coup m, on assigne une polarité $\lambda^{OP}(m) \in \{-1, +1\}$. Par convention, $\lambda^{OP}(m) = +1$ lorsque le coup est un coup Joueur, et $\lambda^{OP}(m) = -1$ lorsque le coup est Opposant. Finalement, on demande à ce que l'arène soit alternante :

$$m \vdash n \quad \text{implique} \quad \lambda^{OP}(m) = -\lambda^{OP}(n)$$

et à ce que toutes les racines (appelées coups *initiaux*) de l'arène aient la même polarité. Un exemple classique d'arène est l'arène booléenne $\mathbb{B}$:

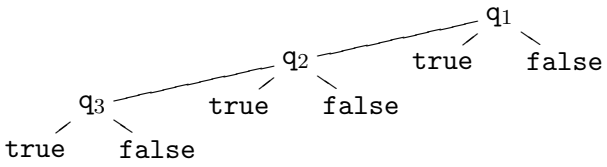


(4.1)

où le coup Opposant `q` justifie les deux coups Joueur `true` et `false`. Chaque arène A induit un ensemble de *parties justifiées*, qui sont essentiellement des suites de coups (pour éviter des complications techniques qui n'auront pas de suite dans ce manuscrit, nous ne parlerons volontairement pas de pointeurs). Typiquement, le type PCF

$$(\mathbb{B}_3 \Rightarrow \mathbb{B}_2) \Rightarrow \mathbb{B}_1$$

définit l'arène



où les indices $1, 2, 3$ sont ici pour distinguer les trois occurrences de l'arène booléenne $\mathbb{B}$. Cette arène contient la partie justifiée

$$\mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_3 \cdot \mathtt{true}_2 \cdot \mathtt{true}_1 \tag{4.2}$$

aussi décrite en utilisant la convention ci-dessous :

$$\begin{array}{ccccc} (\mathbb{B} & \Rightarrow & \mathbb{B}) & \Rightarrow & \mathbb{B} \\ & & & & \mathtt{q} \\ & & \mathtt{q} & & \\ \mathtt{q} & & & & \\ \mathtt{true} & & & & \\ & & \mathtt{true} & & \\ & & & & \mathtt{true} \end{array} \tag{4.3}$$

Notons que la partie (4.2-4.3) fait partie de la stratégie implémentée par le programme PCF $\lambda f.f(\mathtt{true})$.

Martin Hyland et Luke Ong ont démontré [HO00] que les stratégies (finies) peuvent être implémentées dans PCF si et seulement si elles satisfont deux conditions fondamentales, appelées *innocence* et *bon parenthésage*. Nous nous concentrons ici sur la condition de parenthésage, qui est très proche d'une discipline de pile, mais que nous reformulons ici comme une discipline de ressource qui va nous permettre d'interpréter les modalités affine, dupliquante et exponentielle de la logique tensorielle. Cette condition est habituellement exprimée de la façon suivante. Les arènes sont sophistiquées à l'aide d'un *mode* $\lambda^{QA}(m) \in \{Q, A\}$ pour chaque coup de l'arène. On dit qu'un coup est une *question* si

$\lambda^{QA}(m) = Q$ et que c'est une *réponse* si $\lambda^{QA}(m) = A$. On demande à ce qu'aucun coup réponse m ne justifie un coup réponse n :

$$m \vdash n \quad \text{implique} \quad \lambda^{QA}(m) = Q \quad \text{ou} \quad \lambda^{QA}(n) = Q.$$

Cette condition formalise l'intuition qu'une réponse n répond à une question m qui la justifie au cours de la partie. Remarquons que l'alternance des coups d'une partie assure que les réponses Joueur répondent à des questions Opposant et vice versa : ainsi un joueur ne répond jamais à ses propres questions. Par exemple, l'arène $\mathbb{B}$ est raffinée en stipulant que le coup Opposant `q` est une question, et que les deux coups Joueur `true` et `false` sont des réponses.

Une partie justifiée s est dite *bien parenthésée* lorsque toute réponse n apparaissant au cours de la partie répond à la question en cours m, appelée question *pendante*. Cette terminologie est portée par l'intuition que (1) toute question « ouvre » une parenthèse et (2) toute réponse « ferme » une parenthèse, qui doit correspondre à la parenthèse ouverte par la question répondue. Typiquement, une partie (4.2-4.3) est bien parenthésée car toute réponse répond convenablement à la dernière question posée, ce qui donne la séquence bien parenthésée :

$$\begin{array}{l} \mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_3 \cdot \mathtt{true}_2 \cdot \mathtt{true}_1 \\ (_1 \text{———————————————} {}_1) \\ \quad (_2 \text{———————} {}_2) \\ \quad\quad (_3 \text{——} {}_3) \end{array}$$

Inversement, la partie

$$\begin{array}{ccccc} (\mathbb{B} & \Rightarrow & \mathbb{B}) & \Rightarrow & \mathbb{B} \\ & & & & \mathtt{q} \\ & & \mathtt{q} & & \\ \mathtt{q} & & & & \\ & & & & \mathtt{true} \end{array} \tag{4.4}$$

n'est pas bien parenthésée car le coup `true` répond à la première question de la partie, alors qu'il aurait dû répondre à la troisième question (c'est la question pendante). Cela peut être décrit de la manière suivante :

$$\begin{array}{l} \mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_1 \\ (_1 \text{———————} {}_1) \\ \quad (_2 \cdots\cdots\cdots\cdots) \\ \quad\quad (_3 \cdots\cdots) \end{array} \tag{4.5}$$

En fait, la partie (4.4-4.5) appartient à la stratégie qui teste si la fonction $f : \mathbb{B} \Rightarrow \mathbb{B}$ est stricte, au sens où elle interroge son argument : ce test ne peut pas être implémenté avec le langage PCF– bien qu'il puisse être décrit avec PCF étendu avec l'opérateur de contrôle `call-cc`.

4.3.2 Le parenthésage comme gestion des ressources

Nous souhaiterions maintenant comprendre le parenthésage comme une discipline de ressource, plutôt qu'une simple discipline de pile. Une avancée majeure dans cette direction est l'observation qu'une partie bien parenthésée peut être détectée simplement en comptant deux nombres caractéristiques sur un chemin

– le nombre κ_A^+ de questions Joueur ouvertes mais laissées sans réponse ;

– le nombre κ_A^- de questions Opposant ouvertes mais laissées sans réponse.

Bien évidemment, il n'est pas suffisant de compter les nombres κ_A^+ et κ_A^- sur la partie s pour savoir si la partie est bien parenthésée. En effet, la partie bien parenthésée (a) et la partie mal parenthésée (b) introduites dans (4.4-4.5) induisent les mêmes nombres κ_a^+ et κ_A^- :

$$\begin{array}{lllll} (a) & \mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_3 & \longmapsto & \kappa_A^+ = 1, & \kappa_A^- = 1 \\ (b) & \mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_1 & \longmapsto & \kappa_A^+ = 1, & \kappa_A^- = 1 \end{array}$$

Afin de détecter le bon parenthésage, il faut effectuer ce comptage sur les sous-chemins (c) et (d) de ces parties. Cela révèle la différence cruciale :

$$\begin{array}{lllll} (c) & \mathtt{q}_3 \cdot \mathtt{true}_3 & \longmapsto & \kappa_A^+ = 0, & \kappa_A^- = 0 \\ (d) & \mathtt{q}_3 \cdot \mathtt{true}_1 & \longmapsto & \kappa_A^+ = 0, & \kappa_A^- = 1 \end{array}$$

On en déduit une caractérisation fondamentale bien qu'élémentaire :

Proposition 4.11 (bon parenthésage) Une partie s est bien parenthésée si et seulement si tout sous-chemin $m \cdot t \cdot n$ de la partie s satisfait

$$\kappa_A^+(m \cdot t \cdot n) = 0 \quad \text{implique} \quad \kappa_A^-(m \cdot t \cdot n) = 0$$

où m est un coup Opposant et n un coup Joueur ; et dualement

$$\kappa_A^-(m \cdot t \cdot n) = 0 \quad \text{implique} \quad \kappa_A^+(m \cdot t \cdot n) = 0$$

où m est un coup Joueur et n est un coup Opposant

Arrêtons nous un moment pour expliquer cette propriété. Prenons un sous-chemin $m \cdot t \cdot n$ d'une partie bien parenthésée s, où m est un coup Opposant et n est un coup Joueur. La première condition indique que s'il existe une question Opposant à laquelle Joueur n'a pas encore répondu dans $m \cdot t$, alors ou bien Joueur y répond – auquel cas $\kappa_A^-(m \cdot t \cdot n) = 0$ – ou bien il existe une question Joueur à laquelle Opposant n'a pas encore répondu dans $m \cdot t \cdot n$ – auquel cas $\kappa_A^+(m \cdot t \cdot n) \neq 0$. L'autre condition est duale ; il suffit d'inverser les rôles de Joueur et Opposant.

Reformulé de cette façon, le parenthésage ressemble à s'y méprendre à une politique de ressource. L'intuition de base étant que chaque question m *initie* une *requête* pour une *session linéaire*. Cette requête est notée avec une parenthèse ouvrante $({}_i$ et est comptabilisée par $\kappa^{\pm}$ où $\pm$ est la polarité du coup m. Une *réponse* n *accède* ensuite à cette requête, on la note donc ${}_i)$. Dans notre exemple, le coup $\mathtt{q}_3$ initie une requête $({}_3$ à laquelle on accède plus tard dans la partie (4.2-4.3) par la réponse ${}_3)$ émise par le coup $\mathtt{true}$; alors qu'elle reste pendante dans la partie (4.4-4.5). On en conclut qu'une partie comme (4.4-4.5) n'est pas bien parenthésée car elle viole la contrainte de linéarité imposée par les requêtes. Notre modèle de jeu va relier cette politique de linéarité au fait que la formule booléenne associée est définie par

$$\mathbb{B} \;=\; {}^{O}\neg \quad ({}^{P}\neg \oplus {}^{P}\neg) \tag{4.6}$$

en logique tensorielle. Les balises O et P sont utilisées ici comme mnémoniques indiquant que la négation externe ${}^{O}\neg$ est interprétée par un coup Opposant, tandis que la négation ${}^{P}\neg$ est interprétée par un coup Joueur. Voici l'histoire attachée à l'Équation (4.6) : Opposant joue la négation externe, suivi par Joueur, qui joue la négation interne et *au même moment* résout le choix $1 \oplus 1$ entre $\mathtt{true}$ et $\mathtt{false}$. Ceci raffine l'image véhiculée par l'arène booléenne (4.1) en décomposant les deux coups Joueur $\mathtt{true}$ et $\mathtt{false}$ en deux

étapes enchaînées : négation et choix – où la négation encapsule donc les deux coup `true` et `false`. Par conséquent, il devient possible de relâcher la discipline de parenthésage en interprétant la formule booléenne comme

$$\mathbb{B} \;=\; \overset{O}{\neg} \quad \underset{w}{!} \;\; (\overset{P}{\neg} \oplus \overset{P}{\neg}) \tag{4.7}$$

où la modalité affine $\underset{w}{!}$ de la logique tensorielle est insérée entre les deux négations. La hiérarchie intuitionniste sur la formule booléenne (4.6) coïncide avec le modèle de jeux d'arène bien parenthésés de PCF introduit par Martin Hyland et Luke Ong [HO00] alors que la hiérarchie intuitionniste de la formule booléenne 4.7 coïncide avec le modèle de jeux d'arène non parenthésés de PCF avec contrôle décrit par Jim Laird dans [Lai97].

Cette analyse nous amène maintenant à la notion de *multiparenthésage* dans les jeux d'arène. En logique linéaire, toute preuve de la formule

$$(\mathbb{B} \otimes \mathbb{B}) \multimap \mathbb{B}$$

demande la valeur de ces deux arguments booléens, et nous aimerions comprendre cela comme un sorte de condition de parenthésage. Ainsi, la partie

$$\begin{array}{cccccc}
(\mathbb{B} & \otimes & \mathbb{B}) & \multimap & \mathbb{B} \\
 & & & & \mathtt{q} \\
 & & \mathtt{q} & & \\
 & & \mathtt{true} & & \\
\mathtt{q} & & & & \\
\mathtt{true} & & & & \\
 & & & & \mathtt{true}
\end{array} \tag{4.8}$$

va être « bien parenthésée » dans ce nouveau cadre, tandis que la partie

$$\begin{array}{cccccc}
(\mathbb{B} & \otimes & \mathbb{B}) & \multimap & \mathbb{B} \\
 & & & & \mathtt{q} \\
\mathtt{q} & & & & \\
\mathtt{true} & & & & \\
 & & & & \mathtt{true}
\end{array} \tag{4.9}$$

ne va pas être « bien parenthésée » car elle n'explore pas le deuxième argument de la fonction. Ce parenthésage généralisé est capturé par l'idée que la première question initie *trois* requêtes $(_1$, $(_a$ et $(_b$ en même temps. Alors, la partie (4.8) apparaît comme étant bien parenthésée si l'on décrit la situation de la façon suivante :

$$\begin{array}{l}
\mathtt{q}_1 \cdot \mathtt{q}_2 \cdot \mathtt{true}_2 \cdot \mathtt{q}_3 \cdot \mathtt{true}_3 \cdot \mathtt{true}_1 \\
(_1 \text{————————————————} _1) \\
(_a - _a)(_2 \text{——} _2) \\
(_b \text{—————————} _b)(_3 \text{——} _3)
\end{array}$$

tandis que la partie (4.9) n'est pas bien parenthésée car on accède jamais à la requête $(_a$, comme on peut s'en rendre compte sur l'illustration suivante :

$$\begin{array}{l}
\mathtt{q}_1 \cdot \mathtt{q}_3 \cdot \mathtt{true}_3 \cdot \mathtt{true}_1 \\
(_1 \text{——————————} _1) \\
(_a \cdots\cdots\cdots\cdots\cdots\cdots \\
(_b - _b)(_3 \text{——} _3)
\end{array}$$

4.3.3 Jeux de Conway multiparenthésés

Nous allons maintenant formaliser la notion intuitive de requêtes introduite plus haut.

Définition 4.12 (jeu de Conway multiparenthésé)
Un *jeu de Conway multiparenthésé* A est un jeu de Conway (V_A, E_A, λ_A) muni
- d'un ensemble fini $Q_A(x)$ de *requêtes* pour chaque position $x \in V_A$ du jeu de Conway ;
- d'une fonction $\lambda_A(x) : Q_A(x) \to \{-1, +1\}$ assignant à chaque requête de $Q_A(x)$ une polarité. Cette polarité indique si la requête a été faite par Opposant (-1) ou par Joueur $(+1)$;
- pour chaque coup $x \xrightarrow{m} y$, d'une *relation de résidus* $[m]$ pour tout coup m

$$[m] \subseteq Q_A(x) \times Q_A(y)$$

satisfaisant

$$\begin{array}{llll} r[m]r_1 & \text{et} & r[m]r_2 & \text{implique} \quad r_1 = r_2 \\ r_1[m]r & \text{et} & r_2[m]r & \text{implique} \quad r_1 = r_2. \end{array}$$

La définition de résidu est ensuite étendue aux chemins $s : x \twoheadrightarrow y$ de la manière habituelle – par composition des relations. On définit alors

$$r[s] \stackrel{\text{def}}{=} \{r' \mid r[s]r'\} \quad \text{et} \quad [s]r \stackrel{\text{def}}{=} \{r' \mid r'[s]r\}.$$

On dit d'un chemin $s : x \twoheadrightarrow y$ qu'il :
- *accède à une requête* $r \in Q_A(x)$ lorsque r n'a pas de résidu après s — c'est à dire $r[s] = \emptyset$;
- *initie une requête* $r \in Q_A(x)$ lorsque r n'a pas d'ancêtre avant s — c'est à dire $[s]r = \emptyset$.

Nous imposons qu'un coup m initie uniquement des requêtes de sa propre polarité, et accède uniquement à des requêtes de polarité opposé. Afin de formaliser l'intuition que le résidu d'une requête est la requête elle-même, nous imposons aussi que deux chemins parallèles s et t induisent la même relation de résidus :

$$s, t : x \twoheadrightarrow y \quad \text{implique} \quad [s] = [t].$$

Enfin, nous demandons qu'il n'y ait pas de requêtes au niveau de la racine :

$$Q_A(\star) = \emptyset.$$

Opérateur de ressource. Étendre les jeux de Conway avec des requêtes permet de définir un opérateur de ressource κ_A pour tout jeu A

$$\kappa_A = (\kappa_A^+, \kappa_A^-) : \mathrm{Play}_A \to \mathbb{N} \times \mathbb{N}$$

qui compte, pour tout chemin $s : x \twoheadrightarrow y$, le nombre $\kappa_A^+(s)$ (resp. $\kappa_A^-(s)$) de requêtes (potentiellement infini) Joueur (resp. Opposant) dans $r \in Q_A(y)$ initiées par le chemin s — i.e. tel que $[s]r = \emptyset$. La définition de jeu multiparenthésé entraîne trois propriétés capitales sur $\kappa_A^{\pm}$, qui remplaceront la définition initiale de κ_A, et joueront le rôle d'axiomes dans toutes nos preuves – en particulier celle que la composée de deux stratégies bien parenthésées est bien parenthésée.

Propriété 1 : compatibilité. Pour tout chemin $s : x \twoheadrightarrow y$ et pour tout coup Joueur $m : y \rightarrow z$,

$$\kappa_A^-(m) = 0 \quad \text{et} \quad \kappa_A^+(s \cdot m) = \kappa_A^+(s) + \kappa_A^+(m).$$

On a les égalités duales pour les coups Opposant.

Propriété 2 : domination par suffixe. Pour tous chemins $s : x \twoheadrightarrow y$ et $t : y \twoheadrightarrow z$,

$$\kappa_A(t) \leq \kappa_A(s \cdot t).$$

Propriété 3 : sous-additivité. Pour tous chemins $s : x \twoheadrightarrow y$ et $t : y \twoheadrightarrow z$,

$$\kappa_A(s \cdot t) \leq \kappa_A(s) + \kappa_A(t).$$

L'opérateur de ressource vérifie l'hypothèse de compatibilité car Joueur n'initie pas de requêtes Opposant et n'accède pas à des requêtes Joueur. La domination par suffixe stipule qu'une requête ne peut être accédée avant d'être initiée. La sous-additivité indique que la composition de deux chemins ne peut pas accroître le nombre de requêtes.

Nous allons maintenant étendre le fragment multiplicatif de la logique tensorielle aux jeux multiparenthésés.

Dual. Le dual A^* d'un jeu multiparenthésé A est obtenu en inversant la polarité des coups et des requêtes. Ainsi, cela induit un opérateur de ressource

$$(\kappa_{A^*}^+, \kappa_{A^*}^-) = (\kappa_A^-, \kappa_A^+).$$

Produit tensoriel. Le produit tensoriel $A \otimes B$ de deux jeux multiparenthésés A et B a déjà été défini sur les jeux de Conway sous-jacents. Reste à définir les requêtes et la relation de résidus.

- Les requêtes à la position $x \otimes y$ sont données par l'union disjointe des requêtes à la position x du jeu A et des requêtes à la position y du jeu B

$$Q_{A\otimes B}(x \otimes y) = Q_A(x) \uplus Q_B(y).$$

- La polarité des requêtes est héritée de la polarité des requêtes de A et B
- La relation de résidus est définie point-à-point. Par exemple, un coup $x \otimes y \xrightarrow{m \otimes y} z \otimes y$ provenant d'un coup m de A induit la relation

$$\begin{cases} \forall (r, r') \in Q_A(x) \times Q_A(z) & r\ [m \otimes y]_{A\otimes B}\ r' \quad \text{ssi} \quad r\ [m]_A\ r' \\ \forall r \in Q_A(y) & r\ [m \otimes y]_{A\otimes B}\ r \end{cases}$$

Il est remarquable que la prolongation de la notion de requête pour le produit tensoriel implique que l'opérateur de ressource est monoïdal sur les jeux multiparenthésés au sens où

$$\kappa_{A\otimes B}(s) = \kappa_A(s_{|A}) + \kappa_B(s_{|B})$$

L'unique jeu multiparenthésé ayant 1 comme jeu de Conway sous-jacent est l'élément neutre du produit tensoriel. Il sera aussi noté 1.

Parties bien parenthésée et stratégies. Une fois que l'on a défini un opérateur de ressource κ_A sur les chemins, il devient possible d'exprimer ce qu'est une *partie bien parenthésée* comme une partie que satisfait la Proposition 4.11. Ainsi, la proposition devient une définition dans notre modèle.

Définition 4.13 (partie bien parenthésée)
Une partie s est *bien parenthésée* si et seulement si tout sous-chemin $m \cdot t \cdot n$ de s vérifie

$$\kappa_A^+(m \cdot t \cdot n) = 0 \quad \text{implique} \quad \kappa_A^-(m \cdot t \cdot n) = 0$$

où m est un coup Opposant et n un coup Joueur – on dit dans ce cas que s est bien parenthésée pour Joueur – et dualement

$$\kappa_A^-(m \cdot t \cdot n) = 0 \quad \text{implique} \quad \kappa_A^+(m \cdot t \cdot n) = 0$$

où m est un coup Joueur et n un coup Opposant – on dit dans ce cas que s est bien parenthésée pour Opposant.

Il est alors naturel de dire qu'une stratégie est bien parenthésée lorsque tout sous-chemin qu'elle joue est bien parenthésée pour Joueur.

Définition 4.14 (stratégie bien parenthésée)
Une stratégie σ est *bien parenthésée* si et seulement si toute partie $s \cdot m \cdot t \cdot n$ de σ vérifie

$$\kappa_A^+(m \cdot t \cdot n) = 0 \quad \text{implique} \quad \kappa_A^-(m \cdot t \cdot n) = 0$$

où m est un coup Opposant et n est (nécessairement) un coup Joueur.

Toute stratégie σ préserve le parenthésage au sens où :

Lemme 4.15 Si $s \cdot m \cdot n \in \sigma$ et $s \cdot m$ est bien parenthésée, alors $s \cdot m \cdot n$ est bien parenthésée

Démonstration : Comme $s \cdot m$ est bien parenthésée, il suffit de vérifier les sous-chemins de la forme $m' \cdot t \cdot n$ où m' est nécessairement un coup Opposant et n un coup Joueur. Tous ces chemins sont gérés par la définition de stratégie bien parenthésée. ■

Ainsi, deux stratégies bien parenthésées qui interagissent sur un jeu multiparenthésé produisent des parties bien parenthésées.

Il reste maintenant à vérifier que cette notion de parenthésage pour les stratégies est stable par composition. Pour cela, nous allons utiliser toute la puissance de l'axiomatique extraite de la définition de l'opérateur de ressource κ_A.

Proposition 4.16 La composée de deux stratégies bien parenthésées est aussi bien parenthésée.

Démonstration : Nous procédons par l'absurde en supposant qu'il existe un (plus petit) chemin s joué par la stratégie $\tau \circ \sigma$ tel que

$$\kappa^+_{A^* \otimes C}(s) = 0 \quad \text{et} \quad \kappa^-_{A^* \otimes C}(s) > 0.$$

On montre alors que σ ou τ a triché.

Dans un premier temps, en utilisant le lemme du témoin unique sur s, on obtient une interaction $u \in int(A, B, C)$ telle que $u_{|A,B}$ est jouée par σ, $u_{|B,C}$ est jouée par τ. En appliquant la *domination par suffixe*, on sait que pour tout préfixe u' de u, $\kappa^+_{A^* \otimes C}(u') = 0$, ce qui entraîne en utilisant la monoïdalité du produit tensoriel

$$\kappa^+_{A^*}(u'_{|A}) = \kappa^+_C(u'_{|C}) = 0$$

Maintenant, deux cas doivent être considérés :

1. $\kappa_B(u_{|B}) = (0, 0)$.
   La monoïdalité du produit tensoriel donne $\kappa_{A^* \otimes B}(u) = \kappa_{A^*}(u)$ et $\kappa_{B^* \otimes C}(u) = \kappa_C(u)$. Mais comme $\kappa_{A^* \otimes C}(u)^- = \kappa^-_{A^*}(u) + \kappa_C(u)^- > 0$ par hypothèse, on en déduit que $\kappa^-_{A^*}(u) > 0$, ou bien $\kappa^-_{C^*}(u) > 0$.
   Ainsi, soit $\kappa_{A^* \otimes B}(u) \in 0 \times (\overline{\mathbb{N}} \setminus \{0\})$, soit $\kappa_{B^* \otimes C}(u) \in 0 \times \mathbb{N}^*$, ce qui entraîne qu'au moins une des deux stratégies a triché.
2. $\kappa_B(u_{|B}) \neq (0, 0)$.
   Considérons maintenant le plus petit suffixe v de u pour lequel $\kappa_B(v_{|B}) \neq (0, 0)$. Posons $v = m \cdot v'$. Comme v est le plus suffixe à gain non nul, nécessairement $\kappa_B(v'_{|B}) = (0, 0)$ et alors $\kappa_B(v_{|B}) \leq \kappa_B(m)$ (par *sous-additivité*). Mais comme u est la plus petite interaction donnant lieu à un mauvais gain dans $A^* \otimes C$, m est nécessairement dans E_B.
   Traitons le cas où m est un coup Opposant ; le cas où m est un coup Joueur étant similaire. En utilisant la *compatibilité* de κ_A, on sait que $\kappa^-_B(m) = 0$, ce qui implique $\kappa^-_B(v_{|B}) = \kappa^+_{B^*}(v_{|B}) = 0$ et donc $\kappa^-_{B^*}(v_{|B}) > 0$. Considérons maintenant la partie jouée par τ. Les précédentes considérations montrent que $\kappa^+_{B^* \otimes C}(v_{|B^* \otimes C}) = 0$. Mais $\kappa^-_{B^*}(v_{|B}) > 0$ implique $\kappa^-_{B^* \otimes C}(v_{|B^* \otimes C}) > 0$ par monoïdalité de l'opérateur de ressource sur le produit tensoriel. Alors la stratégie τ à joué un chemin interdit. ■

Proposition 4.17 (catégorie des jeux multiparenthésés) Les jeux multiparenthésés et stratégies bien parenthésées forment une catégorie notée $\mathcal{B}$. Comme le catégorie des jeux de Conway, $\mathcal{B}$ est compacte fermée.

4.3.4 Modalités affine et exponentielle

Motivé à la fois par la définition de modalités de ressource mais aussi par l'intuition que la négation tensorielle se matérialise par le soulèvement par un coup Opposant en sémantique des jeux, nous souhaiterions construire un modèle de la logique tensorielle basé sur la catégorie $\mathcal{B}_O$ des jeux multiparenthésés négatifs.

La propriété 2.32 de transport de fermeture permet d'obtenir une catégorie monoïdale symétrique à partir de la structure compacte fermée de $\mathcal{B}$. On perd la structure compacte fermée mais on conserve quand même la notion de trace.

Proposition 4.18 La catégorie $\mathcal{B}$ est une catégorie monoïdale symétrique tracée et fermée.

Négation tensorielle. La négation $\neg A$ d'un jeu négatif est le jeu pointé obtenu en soulevant le jeu dual A^* par un coup Joueur $m_\perp$ qui n'initie pas de requête.

Modalité affine. Comme aucune requête n'est initiée à la racine d'un jeu multiparenthésé, le jeu 1 est terminal dans $\mathcal{B}$. Tous les jeux de $\mathcal{B}_O$ sont donc affines. Pour pouvoir distinguer les jeux affines des jeux linéaires, il faut passer à la notion de *jeux pointés*. Un jeu pointé peut être vu de deux manières différentes : (1) comme un jeu multiparenthésé négatif soulevé par un unique coup Joueur, (2) comme un jeu multiparenthésé négatif en

relâchant l'hypothèse qu'aucune requête n'est initiée à la racine pour les requêtes Joueur. Comme l'interprétation des types de **IdeaML** nécessite seulement des jeux affines, nous pensons qu'il est inutile de compliquer le modèle en considérant les jeux pointés. Aussi, nous nous satisfaisons ici d'une modalité affine est égale à l'identité.

Modalité exponentielle. Tout jeu négatif A induit un jeu dupliquable $!_e A$ – qui peut être regardé comme le tenseur infini de A. On définit $!_e A$ comme suit

- ses positions sont les mots finis $w = x_1 \cdots x_k$ dont les lettres sont des positions x_i du jeu A différentes de la racine ; l'intuition est que la lettre x_i décrit la position courante dans la i-ème copie de A.
- sa racine est le mot vide $\star_{!_e A} = \varepsilon$;
- un coup $w \to w'$ est
  - ou bien un coup joué dans une copie :

$$w_1\ y\ w_2 \ \to\ \ w_1\ y'\ w_2$$

    où $y \to y'$ est un coup du jeu A et $y \neq \star_A$;
  - ou bien l'ouverture d'une nouvelle copie :

$$w \to w \cdot y$$

    où $\star_A \to y$ est un coup du jeu A partant de la racine $\star_A$.

  Si m est un coup de $!_e A$, on note $\underline{m}$ le coup sous-jacent dans A ;
- ses requêtes à la position $w = x_1 \cdots x_n$ sont les paires (i, r) constituées d'un index $1 \leq i \leq n$ et d'une requête r à la position x_i du jeu A. En particulier, il n'y pas de requête à la position vide ε.

Les polarités des coups et des requêtes sont héritées de celles du jeu $!_w A$ de manière évidente ; et la relation de résidus est définie de la même manière que pour le produit tensoriel.

Nous allons maintenant utiliser le résultat général du Théorème 3.41 – et plus particulièrement l'instanciation que nous en avons fait à la Section 3.6.3 – pour montrer que $!_e A$ définit le comonoïde commutatif libre sur le jeu A. On note $t_A : A \to 1$ l'unique flèche de A vers l'objet terminal 1. On va montrer que $!_e A$ est la limite du diagramme

$$\mathcal{A} \quad : \quad 1 \xleftarrow[t_A]{} A \underset{t_A \otimes \mathrm{id}_A}{\overset{\mathrm{id}_A \otimes t_A}{\rightleftarrows}} A^2 \overset{\sigma}{\circlearrowright} \underset{t_A \otimes \mathrm{id}_{A^2}}{\overset{\mathrm{id}_{A^2} \otimes t_A}{\rightleftarrows}} \cdots A^n \overset{\sigma}{\circlearrowright} \cdots$$

dans la catégorie $\mathcal{B}_O$ et que cette limite est monoïdale. La preuve va être très similaire à celle de la Section 3.6.3. La différence majeure est qu'on doit maintenant vérifier que les stratégies que l'on définit sont toutes bien parenthésées. À nouveau, on va directement montrer que $X \otimes !_e A$ est la limite du diagramme

$$X \otimes \mathcal{A} \quad : \quad X \xleftarrow[t_A]{} X \otimes A \underset{\mathrm{id}_X \otimes t_A \otimes \mathrm{id}_A}{\overset{\mathrm{id}_X \otimes \mathrm{id}_A \otimes t_A}{\rightleftarrows}} X \otimes A^2 \overset{\mathrm{id}_X \otimes \sigma}{\circlearrowright} \underset{\mathrm{id}_X \otimes t_A \otimes \mathrm{id}_{A^2}}{\overset{\mathrm{id}_X \otimes \mathrm{id}_{A^2} \otimes t_A}{\rightleftarrows}} \cdots X \otimes A^n \overset{\mathrm{id}_X \otimes \sigma}{\circlearrowright} \cdots$$

On obtiendra ensuite la propriété sur $!_e A$ en faisant $X = 1$.

Proposition 4.19 $X \otimes !_e A$ est la limite du diagramme $X \otimes \mathcal{A}$

Démonstration : Nous devons dans un premier temps définir un cône d'origine $X \otimes !_e A$ sur le diagramme $X \otimes \mathcal{A}$. On procède comme à la Section 2.4.3 en définissant une fonction d'entrelacement des parties de $!_e A$ vers les parties de A^n, puis en définissant une stratégie d'imitation. Étant donnée une partie $s \cdot m$ de $!_e A$, on définit

$$\langle s \cdot m \rangle = \langle s \rangle \cdot \underline{m}$$

où $\underline{m}$ est le coup sous-jacent dans A. On définit alors la stratégie $\varepsilon_n : !_e A \to A^n$ par l'ensemble de parties

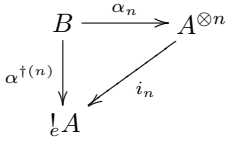
$$\varepsilon_n \quad \stackrel{\text{def}}{=} \quad \{s \in \text{Play}^{even}_{!_e A_1 \multimap A_2^n} \mid \forall t \prec^{even} s \ , \ t_{|!_e A_1} = \langle t_{|A_2^n} \rangle\}$$

Comme nous avons défini des variantes de la stratégie d'imitation, on s'assure que toutes ces stratégies sont bien parenthésées en remarquant que toute partie $s \cdot m \cdot t \cdot n$ jouée par ε_n vérifie

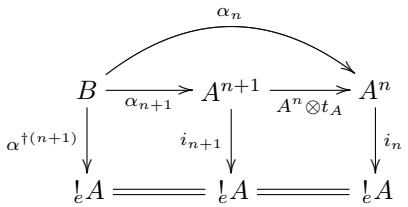
$$\kappa^+_{!_e A \multimap A^n}(m \cdot t \cdot n) = \kappa^-_{!_e A \multimap A^n}(m \cdot t \cdot n)$$

Le cône de $X \otimes !_e A$ sur $X \otimes \mathcal{A}$ est alors donné par les morphismes $\text{id}_X \otimes \varepsilon_n$. Soit maintenant $(B, \alpha : B \to \mathcal{A})$ un cône sur $X \otimes \mathcal{A}$. Nous devons définir une flèche de B dans $X \otimes !_e A$.

Introduisons la stratégie $i_n : X \otimes A^n \to X \otimes !_e A$ qui imite Opposant sur $X \otimes A^{\otimes n}$ pour X et les n première copies de $!_e A$, et qui ne répond pas lorsque Opposant ouvre la $n+1$-ème copie de $!_e A$. Nous avons défini ces stratégies par des restrictions de la stratégie d'imitation sur $X \otimes !_e A$, elles sont donc naturellement bien parenthésées. On définit pour tout n la stratégie $\alpha^{\dagger(n)}$ par le diagramme commutatif suivant :



Considérons maintenant le diagramme suivant :



Il commute sur toutes ces faces excepté pour celle en bas à droite qui vérifie $i_n \circ (A^n \otimes t_B) \subseteq i_{n+1}$. Le chemin extérieur dans le sens horaire étant égal à $\alpha^{\dagger(n)}$, on en déduit que

$$\alpha^{\dagger(n)} \subseteq \alpha^{\dagger(n+1)}.$$

On peut donc définir le soulèvement comonoïdal $\alpha^\dagger$ par la limite croissante des $\alpha^{\dagger(n)}$:

$$\alpha^\dagger \quad \stackrel{\text{def}}{=} \quad \bigcup_n \alpha^{\dagger(n)}$$

Cette stratégie $\alpha^\dagger$ est bien parenthésée car chaque partie qu'elle joue et aussi jouée par la stratégie bien parenthésée $\alpha^{\dagger(n)}$ pour un certain n. On doit aussi montrer que cette stratégie est un morphisme de cônes. Pour cela, remarquons que $\varepsilon_n \circ i_n = \text{id}_{A^n}$ et donc que

$$\varepsilon_n \circ \alpha^{\dagger(n)} = \varepsilon_n \circ i_n \circ \alpha_n = \alpha_n.$$

Il reste à montrer que ce morphisme de cônes est unique. Soit β un autre morphisme de cônes de B vers $!_e A$. On définit

$$\beta^{(n)} = i_n \circ \varepsilon_n \circ \beta$$

et on remarque les deux choses suivantes

$$\beta^{(n)} = \alpha^{\dagger(n)} \quad \text{et} \quad \beta = \bigcup_n \beta^{(n)}.$$

On en déduit que $\alpha^\dagger = \beta$ ce qui termine la preuve. ■

Clairement, on a une propriété symétrique avec le diagramme $\mathcal{A} \otimes X$. En utilisant cette propriété avec $X = 1$, on en déduit que $!_e A$ est la limite du diagramme $\mathcal{A}$. Nous pouvons maintenant énoncer la proposition annoncée plus haut.

Proposition 4.20 Le jeu $!_e A$ est le comonoïde commutatif libre sur le jeu A.

On vient donc de définir un modèle de la logique tensorielle (sans les coproduits) dans une catégorie tracée de jeux équipés d'une notion de parenthésage. La modalité affine y est dégénérée.

4.3.5 Ajouter les additifs

Il nous reste maintenant à ajouter les coproduits à la catégorie $\mathcal{B}_O$. Nous utilisons pour cela la construction de famille Fam introduite à la Section 2.4.4. Comme les résultats ont été obtenus dans un cadre général, nous nous contentons ici de rappeler les constructions.

Produits Finis. Il nous faut tout d'abord définir les produits dans la catégorie $\mathcal{B}_O$ pour que les résultats de la Section 2.4.4 s'appliquent.

Étant donnée une famille $(A_i)_{i\in I}$ de jeux multiparenthésés indexée sur un ensemble fini I, le produit $\&_{i\in I} A_i$ est défini par

- son graphe sous-jacent est obtenu en prenant l'union disjointe des graphes sous-jacents aux A_i en fusionnant les racines ;
- la polarité des coups est héritée directement de celle des coups des A_i ;
- la relation de résidu est directement héritée de celle de chacun des jeux A_i.

La i-ème projection est donnée par la stratégie d'imitation sur le jeux A_i.

Construction de familles. Nous pouvons maintenant définir la construction de famille et la notion induite de produit tensoriel et de négation.

Définition 4.21 (la catégorie $Fam(\mathcal{B}_O)$)
La catégorie $Fam(\mathcal{B}_O)$ a pour objets les familles $\{A_i | i \in I\}$ de jeux multiparenthésés, où I est un ensemble fini. Un morphisme de $\{A_i \mid i \in I\}$ vers $\{B_j \mid j \in J\}$ consiste en une fonction de réindexation $f : I \to J$ ainsi qu'en une famille de stratégies bien parenthésées transverses $\{\sigma_i : A_i \to B_{f(i)} | i \in I\}$ de $\mathcal{C}$.
Le produit tensoriel de deux familles $A = \{A_i | i \in I\}$ et $B = \{B_j | j \in J\}$ est défini par

$$A \otimes B \quad \stackrel{\text{def}}{=} \quad \{A_i \otimes B_j \mid (i,j) \in I \times J\}$$

L'opération qui à une famille $A = \{A_i | i \in I\}$ de $Fam(\mathcal{B}_O)$ associe

$$\neg A = \{\&_i(\neg A_i)\}$$

définit une négation tensorielle sur $Fam(\mathcal{B}_O)$.

On a montré à la Section 2.4.4 la proposition suivante (dans un cadre général)

Théorème 4.22

$Fam(\mathcal{B}_O)$ est un modèle de la logique tensorielle.

Notation Avant de passer à l'interprétation de **IdeaML** à proprement dite, nous rappelons le lexique utilisé pour la présentation monolatérale de la logique tensorielle. Celui-ci va servir à distinguer les coups de Joueur des coups de Opposant. Comme la catégorie $\mathcal{B}_O$ est celle des jeux négatifs, elle est de la polarité négative, c'est à dire Opposant. Dualement, sa catégorie opposée $\mathcal{B}_P \stackrel{\text{def}}{=} (\mathcal{B}_O)^{\text{op}}$ est positive, c'est à dire Joueur.

	$\mathcal{B}_O$	$\mathcal{B}_P$
négation tensorielle	$\overset{O}{\downarrow} A$	$\overset{P}{\uparrow} A$
produit tensoriel	$\otimes$	$\parr$
produit	&	$\oplus$
modalité de ressource	!	?

4.4 Interprétation des références dans les jeux multiparenthésés

Dans cette section, nous présentons un résultat de pleine adéquation de notre modèle vis-à-vis du langage **IdeaML**. Nous obtenons ce résultat en montrant que le modèle d'Abramsky et al. [AHM98] est la catégorie de Kleisli associé à la comonade $!_e$ sur notre catégorie de jeux multiparenthésés négatifs. Nous présentons ensuite l'intérêt de notre modèle : il permet de définir la stratégie correspondant à une cellule mémoire de manière interne et graduelle. On va construire les trois types de cellules suivants, grâce à la négation tensorielle, la structure compacte fermée et la modalité exponentielle munie d'une notion d'accès mémoire.

- la *cellule mémoire linéaire* dans laquelle on écrit une seule fois et on lit une seule fois (cette construction utilise la négation tensorielle et la structure compacte fermée) ;
- la *cellule mémoire constante* dans laquelle on écrit une fois et on lit autant de fois que l'on veut (cette construction utilise la modalité exponentielle) ;
- la *cellule mémoire* dans laquelle on écrit et on lit autant de fois que l'on veut (cette construction utilise l'accès mémoire de la modalité exponentielle).

4.4.1 Complétude par rapport à IdeaML

Nous allons dans un premier temps montrer que le modèle donné dans [AHM98] est une description de la catégorie de Kleisli $(\mathcal{B}_O)_{!_e}$ associé à la comonade $!_e$ sur $\mathcal{B}_O$, vue à travers le prisme de la construction de famille.

Nous rappelons d'abord la catégorie des jeux d'arènes avec question/réponse définie dans [AHM98]. Elle correspond essentiellement aux jeux décrits à la Section 4.3.1.

Définition 4.23 (jeu d'arène avec question/réponse)
Une arène avec question/réponse est un triplet $A = (M_A, \lambda_A, \vdash_A)$ où

- M_A est un ensemble de *coups*.

- $\lambda_A = (\lambda_A^{OP}, \lambda_A^{QA}) : M_A \to \{O,P\}\times\{Q,A\}$ est une fonction d'*étiquetage* qui indique si un coup est Joueur (P) ou Opposant (O), et si c'est une question (Q) ou une réponse (A).
- $\vdash_A$ est une relation de justification entre les coups $M_A + \{\star\}$ et les coups de M_A telle que : un coup initial $(\star \vdash m)$ est de polarité OQ, une réponse est toujours justifiée par une question et l'arène est alternante

$$m \vdash n \;\Rightarrow\; \lambda_A^{OP}(m) \neq \lambda_A^{OP}(n).$$

On dit qu'une réponse n répond à un question m lorsque $m \vdash n$.
L'arène $A \Rightarrow B$ est définie par

- $M_{A\Rightarrow B} = M_A + M_B$
- $\lambda_{A\Rightarrow B} = \lambda_A^* + \lambda_B$ où λ_A^* applique les polarités duales de la fonction λ_A
- $m \vdash_{A\Rightarrow B} n \quad \overset{\text{def}}{\iff} \quad m \vdash_A n \quad \text{ou} \quad m \vdash_A n \quad \text{ou} \quad (\star \vdash_B m \wedge \star \vdash_A n)$.

Une stratégie est alors un sous-ensemble de parties

- *justifiées* : une partie s est justifiée, ce que l'on note $s \in L_A$, si pour tout coup non initial n de s, il existe un coup m de s qui justifie n ;
- *alternées* : une partie s est alternée si chaque coup est suivi d'un coup de polarité opposée ;
- *bien parenthésées* : nous ne revenons pas ici sur la longue discussion donnée à la Section 4.3.1.

Abramsky et al. ajoutent aussi une notion de stratégie *filaire*. Techniquement, cette condition indique qu'une stratégie $\sigma : A \to B$ joue uniquement selon sa composante connexe, c'est à dire en fonction des coups qui ont été justifiés par le même coup initial. En fait, cela revient à dire que la stratégie joue indépendamment dans chaque fibre de B et donc est de la forme $!A \multimap B$ dans la catégorie $\mathcal{B}_O$. Nous allons maintenant formaliser cette intuition. On note **AHM** la catégorie dont les objets sont les arènes avec question/réponse et les morphismes de A vers B sont les stratégies bien parenthésées sur l'arène $A \Rightarrow B$ et on note $\mathbf{AHM_{fil}}$ la restriction de **AHM** aux stratégies filaires.

On définit, dans un premier temps, un foncteur $\mathcal{U}$ de **AHM** vers $\mathcal{B}_O$.

Définition 4.24 (AHM → $\mathcal{B}_O$)
Soit $A = (M_A, \lambda_A, \vdash_A)$ une arène avec question/réponse, on définit le jeu multiparenthésé négatif $\mathcal{U}(A)$ par

- ses positions sont les parties justifiées

$$V_{\mathcal{U}(A)} \overset{\text{def}}{=} L_A;$$

- les arêtes entre deux parties justifiées s et s' sont données par

$$s \xrightarrow{m} s' \quad \overset{\text{def}}{\iff} \quad s' = s \cdot m;$$

- la polarité des coups est héritée de celle du jeu A ;
- un coup $s \xrightarrow{m} s'$ initie une requête lorsque $\lambda_A^{QA}(m) = Q$ dans le jeu A ;
- un coup n accède à une requête initiée par un coup m lorsque

$$\lambda_A^{QA}(n) = A \quad \text{et} \quad m \vdash_A n.$$

À chaque stratégie $\sigma : A$ dans **AHM** correspond la même stratégie $\sigma : \mathcal{U}(A)$ dans $\mathcal{B}_O$.

Russ Harmer a montré dans sa thèse [Har00] que la catégorie $\mathbf{AHM_{fil}}$ est la sous catégorie de **AHM** des morphismes respectant la structure comonoïdale des arènes. De notre point de vue, ce résultat s'exprime par un isomorphisme entre la catégorie $\mathbf{AHM_{fil}}$ et la catégorie de Kleisli de la comonade $!_e$ sur $\mathcal{B}_O$.

Proposition 4.25 Le foncteur $\mathcal{U} : \mathbf{AHM} \to \mathcal{B}_O$ se relève en un isomorphisme

$$\widetilde{\mathcal{U}} \quad : \quad \mathbf{AHM_{fil}} \xrightarrow{\cong} (\mathcal{B}_O)_{!_e}$$

faisant commuter le diagramme

$$\begin{array}{ccc} \mathbf{AHM_{fil}} & \xrightarrow{\widetilde{\mathcal{U}}} & (\mathcal{B}_O)_{!_e} \\ \downarrow & & \downarrow \\ \mathbf{AHM} & \xrightarrow{\mathcal{U}} & \mathcal{B}_O \end{array}$$

Démonstration : Tout d'abord, remarquons que tout jeu multiparenthésé $\mathcal{U}(A)$ est de la forme $!_e\widetilde{\mathcal{U}}(A)$, où $\widetilde{\mathcal{U}}(A)$ est le jeu $\mathcal{U}(A)$ où on ne peut jouer qu'un seul coup initial dans une partie. Et réciproquement, tout jeu multiparenthésé négatif de la forme $!_eA'$ est l'image par $\mathcal{U}$ d'un jeu d'arène A. Ensuite, en utilisant la caractérisation de la Proposition 4.11, on en déduit que les notions de parenthésage coïncident. Enfin, la définition d'une stratégie filaire dans les jeux d'arènes nous permet de dire que toute stratégie

$$\mathcal{U}(\sigma) : \mathcal{U}(A) \multimap \mathcal{U}(B) = !_e\widetilde{\mathcal{U}}(A) \multimap !_e\widetilde{\mathcal{U}}(A)$$

est en fait le relèvement unique d'une stratégie

$$\widetilde{\mathcal{U}}(\sigma) : !_e\widetilde{\mathcal{U}}(A) \multimap \widetilde{\mathcal{U}}(B).$$

On en déduit que les stratégie bien parenthésées et filaires sur un jeu $A \to B$ de **AHM** sont en bijection avec les stratégies bien parenthésées sur le jeu $!_e\widetilde{\mathcal{U}}(A) \multimap \widetilde{\mathcal{U}}(B)$ dans $\mathcal{B}_O$. ■

Abramsky et al. modifient ensuite leur modèle en considérant la construction de famille, exactement comme nous l'avons fait pour obtenir un modèle de la logique tensorielle (cf. Théorème 4.22). On en déduit la proposition suivante et son corollaire immédiat.

Proposition 4.26 La catégorie utilisée dans [AHM98] pour donner un modèle pleinement adéquat (*fully abstract* en anglais) de **IdeaML** est isomorphe à la catégorie $Fam((\mathcal{B}_O)_{!_e})$.

Corollaire 4.27 La catégorie $Fam((\mathcal{B}_O)_{!_e})$ offre un modèle pleinement adéquat de **IdeaML**.

4.4.2 Une analyse de la stratégie $\texttt{cell}_A$

Avant de pouvoir expliquer la construction des différentes stratégies correspondant à chaque type de cellule, il nous semble nécessaire de rappeler plus en détail ce que donne, dans notre modèle, l'interprétation du langage **IdeaML** faite dans [AHM98].

Abramsky et al. ont utilisé une interprétation du langage en appel par valeur **IdeaML** basée sur une catégorie cartésienne fermée munie d'une monade forte. Cette monade, construite sur la catégorie de famille, se décrit dans notre modèle par

$$T(A_i)_{i\in I} \stackrel{\text{def}}{=} (\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i).$$

L'unité de cette monade est obtenue à partir de l'unité de la monade de continuation et des injections

$$A_i \longrightarrow \overset{OP}{\downarrow\uparrow} A_i \longrightarrow \overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i.$$

La force de la monade est une conséquence directe de la force de la monade de continuation. L'interprétation d'un jugement de typage

$$x_1 : A_1, \dots x_n : A_n \vdash M : A$$

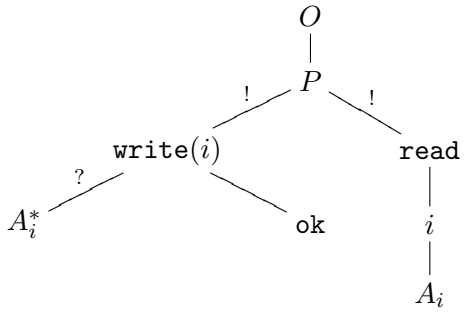
est donnée par un morphisme

$$!\{A_1\} \otimes \cdots \otimes !\{A_n\} \longrightarrow T!\{A\}.$$

La traduction de l'arène correspondant au type ref[A] telle que l'ont donnée Abramsky et al. produit le jeu de Conway

$$\{\mathrm{ref}[A]\} \stackrel{\text{def}}{=} T(!(\&_i \overset{O}{\downarrow} (\overset{P}{\uparrow} 1 \parr ?A_i^*)) \otimes !(\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i)) \stackrel{\text{def}}{=} T(!\mathtt{Write}_A \otimes !\mathtt{Read}_A) \qquad (4.10)$$

où $\{A\} = (A_i)_{i\in I}$ est la traduction du type A dans la catégorie $Fam(\mathcal{B}_O)$. $\mathtt{Write}_A$ correspond au type du module d'écriture et $\mathtt{Read}_A$ correspond au type du module de lecture. Ce jeu peut être vu comme l'arène



De manière informelle, la stratégie $\mathtt{cell}_A$
- répond au premier coup O par le coup P ;
- si le coup suivant joué par Opposant est `read`, la stratégie ne répond pas ;
- la stratégie $\mathtt{cell}_A$ répond à $\mathtt{write}(i)$ par le coup `ok` ;
- si Opposant joue le coup `read` et que le dernier coup d'écriture à avoir été joué est $\mathtt{write}(i)$, la stratégie $\mathtt{cell}_A$ répond i ;
- lorsque Opposant joue un coup initial de A_i dans la composante de lecture, la stratégie $\mathtt{cell}_A$ répond par le coup initial dans une nouvelle copie du jeu $?A_i^*$ qui correspond au dernier coup $\mathtt{write}(i)$ joué ;
- ensuite, la stratégie $\mathtt{cell}_A$ imite les coups de Opposant dans A_i (resp. A_i^*) par les coups correspondants dans A_i^* (resp. A_i).

Nous allons maintenant construire formellement la stratégie $\mathtt{cell}_A$ en plusieurs étapes.

Cellule mémoire linéaire et objet dual. Premièrement, nous allons construire la stratégie $\mathtt{lin\_cell}_A$ pour interpréter la cellule mémoire linéaire, dans laquelle on ne peut écrire et lire qu'une seule fois. Cette stratégie a pour type

$$1 \xrightarrow{\mathtt{lin\_cell}_A} T((\&_i \overset{O}{\downarrow} (\overset{P}{\uparrow} 1 \parr A_i^*)) \otimes (\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i)). \qquad (4.11)$$

Nous allons voir que cette stratégie s'obtient facilement en utilisant les propriété de la négation tensorielle. La seule chose qui nous manque est une flèche opposée à la force $t_{A,B,C}$

$$\mathfrak{t}_{A,B,C} \quad : \quad \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C)) \longrightarrow \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} B) \otimes C.$$

Pour la construire, nous allons utiliser la structure compacte fermée de la catégorie $\mathcal{B}$. La construction vient de la propriété suivante, bien connue des algébristes.

Proposition 4.28 ([May06]) Soit $(\mathcal{C}, \otimes, 1, \multimap)$ une catégorie monoïdale symétrique fermée. Si l'objet C de $\mathcal{C}$ possède un objet dual C^*, alors, pour tous objets A et B, le morphisme canonique

$$f_{A,B,C} \quad : \quad (A \multimap B) \otimes C \longrightarrow A \multimap (B \otimes C)$$

est un isomorphisme.

Démonstration : Le morphisme inverse est donné par

$$\begin{array}{rcl} A \multimap (B \otimes C) & \xrightarrow{\cdots \otimes \eta_C} & A \multimap (B \otimes C) \otimes C^* \otimes C \\ & \xrightarrow{f_{A, B\otimes C, C^* \otimes C}} & (A \multimap (B \otimes C \otimes C^*)) \otimes C \\ & \xrightarrow{(\cdots \varepsilon_C) \otimes C} & (A \multimap B) \otimes C \end{array}$$

Il est immédiat de vérifier que cela définit bien un isomorphisme. ■

Nous allons utiliser une variante de cette proposition dans le cadre de la négation tensorielle au lieu de la fermeture. Tout d'abord, remarquons que la force

$$\overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} B) \otimes C \xrightarrow{t_{A,B,C}} \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C))$$

existant dans la catégorie $\mathcal{B}_O$ s'étend à toute la catégorie $\mathcal{B}$. La seule subtilité est que, dans le cas où le jeu C est positif, si Opposant joue son premier coup dans le jeu C situé à gauche de la flèche, la stratégie ne répond pas. On peut maintenant définir, dans la catégorie $\mathcal{B}$, la stratégie qui nous intéresse par

$$\begin{array}{rcl} \mathfrak{t}_{A,B,C} \overset{\text{def}}{=} \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C)) & \xrightarrow{\cdots \otimes \eta_C} & \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C)) \otimes C^* \otimes C \\ & \xrightarrow{t_{A,B\otimes C, C^* \otimes C}} & \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C \otimes C^*)) \otimes C \\ & \xrightarrow{(\cdots \varepsilon_C) \otimes C} & \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} B)) \otimes C. \end{array}$$

Cette stratégie définit un morphisme de la catégorie $\mathcal{B}_O$, qui fournit un inverse à droite à la force.

$$t_{A,B,C} \circ \mathfrak{t}_{A,B,C} = \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C)) \xrightarrow{\text{id}} \overset{O}{\downarrow} (A \parr \overset{P}{\uparrow} (B \otimes C))$$

Cette stratégie va être au cœur de notre définition de la stratégie $\texttt{lin\_cell}_A$. Cette construction a pour point de départ l'unité de la monade T

$$A_i \longrightarrow \overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i.$$

On prend ensuite l'image de cette unité par la négation tensorielle :

$$\overset{P}{\uparrow} A_i \longrightarrow \overset{P}{\uparrow} (\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i)$$

et on utilise la loi définissant la négation tensorielle pour obtenir une stratégie $\chi_{(A_i)}$ de type

$$\chi_{(A_i)} \quad : \quad 1 \longrightarrow \overset{O}{\downarrow} (A_i^* \parr \overset{P}{\uparrow} (\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i)).$$

À ce stade, nous allons utiliser la stratégie partielle

$$\overset{O}{\downarrow} (A_i^* \parr \overset{P}{\uparrow} (1 \otimes \overset{OP}{\downarrow\uparrow} A_i)) \quad \xrightarrow{\mathfrak{t}} \quad \overset{O}{\downarrow} (A_i^* \parr \overset{P}{\uparrow} 1)) \otimes \overset{OP}{\downarrow\uparrow} A_i,$$

$$\{\Gamma \vdash \texttt{new}\ x : A, y : B\ \texttt{in}\ M\} = \{\Gamma \vdash \texttt{new}\ y : B, x : A\ \texttt{in}\ M\} \quad (4.12)$$

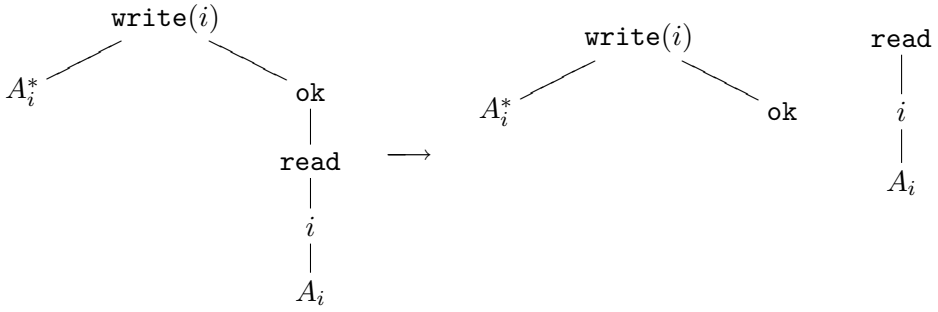
$$\{\Gamma, x : \mathrm{ref}[A] \vdash \texttt{new}\ y : B\ \texttt{in}\ x := V; M\} = \{\Gamma, x : \mathrm{ref}[A] \vdash x := V; \texttt{new}\ y : B\ \texttt{in}\ M\} \quad (4.13)$$

$$\{\Gamma \vdash \texttt{new}\ x : A, y : B\ \texttt{in}\ x := V_1; y := V_2; M\} = \{\Gamma \vdash \texttt{new}\ x : A, y : B\ \texttt{in}\ y := V_2; x := V_1; M\} \quad (4.14)$$

$$\{\Gamma \vdash \texttt{new}\ x : A\ \texttt{in}\ x := V; (\lambda y.M)(!x)\} = \{\Gamma \vdash \texttt{new}\ x : A\ \texttt{in}\ x := V; (\lambda y.M)V\} \quad (4.15)$$

FIG. 4.5 – Validité de la cellule mémoire linéaire (et de la cellule mémoire constante)

ce qui s'énonce dans la notation des jeux d'arènes par



Toutes les parties jouées par cette stratégie commence par les coups

$$\texttt{write}(i) \cdot (\texttt{write}(i))^* \cdot i^* \cdot i$$

suivis par la trace d'une stratégie d'imitation. Autrement dit, nous obtenons une stratégie d'imitation qui ne répond pas si opposant demande de lire avant d'écrire. Il suffit maintenant de faire le produit sur tous les i et de composer avec l'unité de la monade de continuation pour obtenir la stratégie $\texttt{lin\_cell}_A$ décrite par l'équation (4.11).

Nous allons maintenant montrer que cette cellule mémoire linéaire fait bien ce qu'on attend d'elle au sens où elle valide les quatre équations de la Figure 4.5. Ces quatre équations sont bien connues puisqu'elles constituent quatre des cinq équations validées par une cellule mémoire dans le travail d'Abramsky et al. [AHM98]. Elles définissent ici la validité de la cellule mémoire linéaire.

Le formalisme de la logique tensorielle nous permet de déduire la validité de ces équations de manière purement catégorique, à l'aide de diagrammes commutatifs de nature algébriques. En particulier, nous n'utilisons jamais les stratégies sous-jacentes aux morphismes canoniques.

– L'égalité (4.12) vient du fait que le diagramme

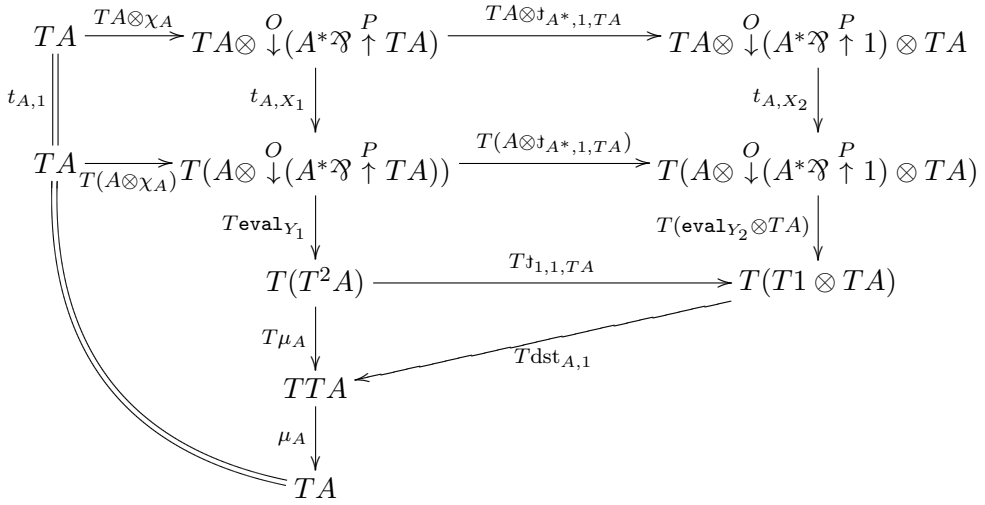
$$A \otimes B \xrightarrow{\eta_A \otimes \eta_B} TA \otimes TB \underset{\text{right}}{\overset{\text{left}}{\rightrightarrows}} T(A \otimes B)$$

commute, où les deux morphismes de droite sont les deux ordres d'évaluation induits par la monade forte T ;

– L'égalité (4.13) provient directement des lois de cohérence 1.9 (associativité) et 1.10 (multiplication) vérifiées par T ;

– L'égalité (4.14) vient de la bifonctorialité du produit tensoriel.

L'égalité (4.15) est plus difficile à obtenir, car elle fait intervenir les propriétés diagrammatiques du morphisme $\natural$ (pour simplifier, nous présentons ici les diagrammes clé dans le cas d'une famille singleton, $(A_i)_{i\in I} = (A)$, dans un catégorie monoïdale stricte). Plus précisément, cette équation se réduit au diagramme commutatif



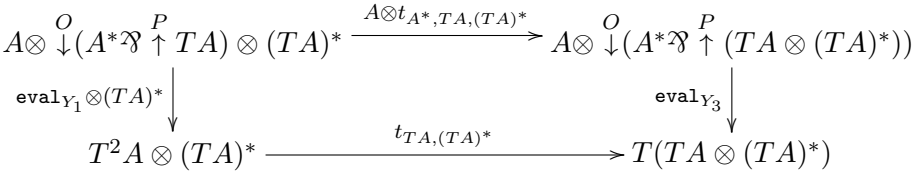
où

$$\mathtt{eval}_B : A \otimes \overset{O}{\downarrow} (A^* \parr B) \to \overset{O}{\downarrow} B$$

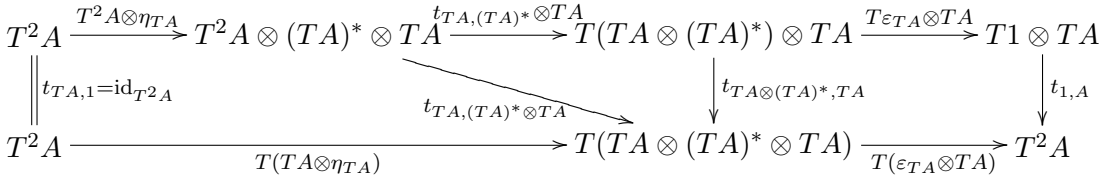
est obtenue en appliquant la loi définissant la négation tensorielle à l'identité

$$\overset{O}{\downarrow} (A^* \parr B) \xrightarrow{\mathrm{id}} \overset{O}{\downarrow} (A^* \parr B),$$

et où $X_1 = \overset{O}{\downarrow}(A^* \parr \overset{P}{\uparrow} TA)$, $X_2 = \overset{O}{\downarrow}(A^* \parr \overset{P}{\uparrow} 1) \otimes TA$, $Y_1 = \overset{P}{\uparrow} TA$, $Y_2 = \overset{P}{\uparrow} 1$. Les deux carrés du haut commutent par naturalité de la force de la monade T. Le triangle en bas à gauche commute en tant que variante de l'identité de zigzag satisfaite par la négation tensorielle. En développant la définition de $\natural$ et en utilisant la naturalité de l'unité et de la counité de la structure compacte fermée, la commutation du carré mettant en jeu le morphisme $\mathtt{eval}$ se réduit à la commutation du carré



où $y_3 = \overset{P}{\uparrow} (TA \otimes (TA)^*)$. De la même manière, pour voir que le triangle en bas à droite commute, on doit revenir à la définition de $\natural$. La commutation de ce triangle se réduit alors à la commutation du diagramme



Le carré de gauche commute par naturalité de la force, le triangle du milieu commute par associativité de la force, et le carré de droite commute par naturalité de la force.

On vient donc de montrer que la cellule mémoire linéaire satisfait les quatre équations de la Figure 4.5.

Cellule mémoire constante et modalité exponentielle. On peut aussi décrire la stratégie $\texttt{const\_cell}_A$ qui interprète la cellule dans laquelle on peut écrire une fois et lire autant de fois que l'on veut. On l'obtient en prenant l'image par ! de l'unité de la monade T

$$!A_i \longrightarrow ! \overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i.$$

On applique ensuite la même transformation que pour la cellule mémoire linéaire afin d'obtenir

$$1 \xrightarrow{\texttt{const\_cell}_A} T((\&_i \overset{O}{\downarrow} (\overset{P}{\uparrow} 1 \parr ?A_i^*)) \otimes !(\overset{O}{\downarrow} \oplus_i \overset{P}{\uparrow} A_i)) = T(\texttt{Write}_A \otimes !\texttt{Read}_A). \quad (4.16)$$

Cette stratégie à une seule phase d'écriture et lance ensuite une nouvelle copie sur le même fil à chaque fois que Opposant demande une nouvelle lecture.

En utilisant la déréliction de la modalité exponentielle, on déduit facilement que la cellule mémoire constante valide elle aussi les équations de la Figure 4.5.

Cellule mémoire et temporalité. Il nous faut maintenant décrire la cellule mémoire dans laquelle on peut faire autant de lecture et d'écriture que l'on veut. L'idée la plus naturelle est de se dire que cela revient au même qu'une infinité de cellule mémoire constante. Le problème est que cela nous donne une stratégie de type

$$1 \longrightarrow !(\texttt{Write}_A \otimes !\texttt{Read}_A)$$

où la gestion de l'ordonnancement des copies est explicite. La cellule mémoire quant à elle masque complètement cette gestion au niveau du type. Il va donc falloir utiliser l'aspect dynamique de notre catégorie de jeux afin de définir une stratégie d'accès mémoire

$$\xi_{\texttt{Write}_A,\texttt{Read}_A} \quad : \quad !(\texttt{Write}_A \otimes !\texttt{Read}_A) \longrightarrow !\texttt{Write}_A \otimes !\texttt{Read}_A$$

qui va forcer chaque lecture à lire la dernière valeur écrite. Cette stratégie n'étant pas innocente (elle n'est même pas visible), on a aucune chance de pouvoir la définir à partir des morphismes structuraux décrivant la logique tensorielle.

Définition 4.29 (stratégie d'accès mémoire)
Étant donné une partie

$$s : \star_{!A \otimes !B} \twoheadrightarrow (w, w')$$

du jeu $!A \otimes !B$ où $w \in V_A^*$ et $w' \in V_B^*$, on définit la position du jeu $!(A_1 \otimes !B_1)$ selon la partie s par

$$(w, w') \downarrow s \overset{\text{def}}{=} (a_1, w_1) \cdots (a_n, w_n)$$

où $a_i \in V_A$, $w_i \in V_B^*$ et où chaque copie, correspondant soit à la position a_i et soit aux position de w_i, a été ouverte, au cours de la partie s, avant la copie correspondant à la position a_{i+1}.

On définit la fonction d'accès mémoire $f : \text{Play}_{!A_2 \otimes !B_2} \to \text{Play}_{!(A_1 \otimes !B_1)}$ comme l'unique fonction

$$s : \star_{!A \otimes !B} \twoheadrightarrow (w, w') \quad \mapsto \quad f(s) : \star_{!(A \otimes !B)} \twoheadrightarrow (w, w') \downarrow s$$

telle que s et $f(s)$ aient les mêmes coups sous-jacents. La stratégie d'accès mémoire $\xi_{A,B} : !(A \otimes !B) \longrightarrow !A \otimes !B$ est alors définit par

$$\xi_{A,B} \overset{\text{def}}{=} \{s \in \text{Play}^{even}_{!(A_1 \otimes !B_1) \multimap (!A_2 \otimes !B_2)} \mid \forall t \prec^{even} s \ , \ t_{|!(A_1 \otimes !B_1)} = f(t_{|!A_2 \otimes !B_2})\}$$

Nous allons maintenant axiomatiser les propriétés satisfaite par cette stratégie d'accès mémoire pour qu'elle donne bien lieu à une cellule mémoire (par soucis de clarté, nous omettons le recours à l'associativité de la catégorie monoïdale).

Définition 4.30 (modalité d'accès en mémoire)
Une *modalité d'accès mémoire* est une modalité exponentielle ! équipée d'une transformation naturelle ξ, appelée *accès mémoire*, de composante

$$\xi_{A,B} \quad : \quad !(A \otimes !B) \longrightarrow !A \otimes !B$$

telle que les diagrammes

$$\begin{array}{ccc} !(A \otimes !B) & \xrightarrow{\varepsilon_{A\otimes !B}} & A \otimes !B \\ {\scriptstyle \xi_{A,B}}\downarrow & \nearrow_{\varepsilon_A \otimes !B} & \\ !A \otimes !B & & \end{array} \qquad (4.17)$$

$$\begin{array}{ccc} !(A \otimes !B) & \xrightarrow{d_{A\otimes !B}} & !(A \otimes !B) \otimes !(A \otimes !B) \\ {\scriptstyle \xi_{A,B}}\downarrow & & \downarrow{\scriptstyle !(A\otimes \varepsilon_B)\otimes \xi_{A,B}} \\ !A \otimes !B & \xrightarrow[d_A \otimes !B]{} & !A \otimes !A \otimes !B \end{array} \qquad (4.18)$$

commutent, où $\varepsilon_A : !A \to a$ représente la counité de la modalité et $d_A : !A \to !A \otimes !A$ représente la comultiplication de la modalité exponentielle.

On montre facilement que la stratégie d'accès mémoire vérifie les diagrammes (4.17) et (4.18) et équipe donc la modalité exponentielle d'un accès mémoire.

Proposition 4.31 La transformation naturelle induite par les stratégies d'accès mémoire définit une modalité d'accès mémoire sur les jeux multiparenthésés négatifs.

Nous allons maintenant montrer que cette notion d'accès mémoire d'une modalité exponentielle permet de garantir que la stratégie

$$\mathtt{cell}_A \stackrel{\text{def}}{=} \xi_{\mathtt{Write}_A,\mathtt{Read}_A} \circ !\mathtt{const\_cell}_A,$$

interprétant la cellule mémoire valident les cinq équations requises dans le travail d'Abram-

$$\{\Gamma \vdash \mathtt{new}\ x : A, y : B\ \mathtt{in}\ M\} = \{\Gamma \vdash \mathtt{new}\ y : B, x : A\ \mathtt{in}\ M\} \qquad (4.19)$$

$$\{\Gamma, x : \mathrm{ref}[A] \vdash \mathtt{new}\ y : B\ \mathtt{in}\ x := V; M\} = \{\Gamma, x : \mathrm{ref}[A] \vdash x := V; \mathtt{new}\ y : B\ \mathtt{in}\ M\} \qquad (4.20)$$

$$\{\Gamma \vdash \mathtt{new}\ x : A, y : B\ \mathtt{in}\ x := V_1; y := V_2; M\} = \{\Gamma \vdash \mathtt{new}\ x : A, y : B\ \mathtt{in}\ y := V_2; x := V_1; M\} \qquad (4.21)$$

$$\{\Gamma \vdash \mathtt{new}\ x : A\ \mathtt{in}\ x := V; (\lambda y.M)(!x)\} = \{\Gamma \vdash \mathtt{new}\ x : A\ \mathtt{in}\ x := V; (\lambda y.M)V\} \qquad (4.22)$$

$$\{\Gamma \vdash \mathtt{new}\ x : A\ \mathtt{in}\ x := V_1; x := V_2; M\} = \{\Gamma \vdash \mathtt{new}\ x : A\ \mathtt{in}\ x := V_2; M\} \qquad (4.23)$$

FIG. 4.6 – Sémantique de la cellule mémoire

sky et al. [AHM98], rappelé à la Figure 4.6. Les quatre premières équations se déduise en deux temps. D'abord, comme le diagramme (4.17) commute, on sait que si une seule écriture est faite, la cellule mémoire se comporte comme la cellule mémoire constante. Par ailleurs, on sait déjà que la cellule mémoire constante valide ces équations. Il nous reste à étudier l'égalité (4.23). Elle provient directement du fait que le diagramme (4.18) commute, ce qui indique que c'est la dernière écriture qui va être lu par la stratégie $\mathtt{cell}_A$.

4.4.3 Travaux futurs

Nous avons défini une catégorie de jeux compacte fermée, équipée d'une notion de multiparenthésage. Cette catégorie interprète un langage à la ML avec des références d'ordre supérieur en se ramenant au modèle introduit par Samson Abramsky, Kohei Honda et Guy McCusker dans [AHM98]. L'intérêt de notre modèle est que le traitement de la cellule mémoire est obtenu de manière purement diagrammatique, sans passer par des définitions et des preuves directes souvent laborieuses en sémantique des jeux.

Il nous faudrait maintenant développer un langage plus adapté à nos idées sémantiques afin de mettre en valeur la notion de trace et de multiparenthésage. Cela passe par deux modifications du langage **IdeaML** :

1. La nécessité d'un opérateur de trace serait plus apparente avec une règle de typage du style
$$\frac{\Gamma, x : \mathrm{ref}[A] \vdash M : B}{\Gamma \vdash \mathtt{new}_A\ x\ \mathtt{in}\ M : B}$$
L'idée serait alors d'interpréter cette règle dans notre modèle par
$$\frac{M : \{\Gamma\} \otimes\, !A \ \longrightarrow\ B \otimes\, !A}{\mathrm{Tr}^{!A}_{\{\Gamma\},B}(M) : \{\Gamma\} \ \longrightarrow\ B}$$

2. Il faudrait enrichir les types afin que le contrôle des ressources soit plus apparent. Le multiparenthésage permet par exemple de définir un type
$$\mathbb{B}\mathrm{ool}_1 \otimes \mathbb{B}\mathrm{ool}_2 \ \longrightarrow\ \mathrm{Bool}_3$$
où le joueur est obligé d'utiliser les deux arguments $\mathbb{B}\mathrm{ool}_1$ et $\mathbb{B}\mathrm{ool}_2$ avant de répondre dans le jeu $\mathbb{B}\mathrm{ool}_3$.

Chapitre 5

Dualité et continuations linéaires

La Reproduction Interdite,
Magritte (1937)

Sommaire

Une des motivations de la logique tensorielle est de fournir un cadre plus souple à l'interprétation des langages de bas niveau de type assembleur afin d'appréhender la sémantique de la compilation de façon plus uniforme. Les deux derniers chapitres de cette thèse étayent cette idée.

L'étude d'un langage de bas niveau que nous proposons au chapitre suivant est basée sur une sémantique de « style passage par continuation » (CPS) pour permettre l'établissement de propriétés sur les fragments de code assembleur axées sur la divergence. Nous souhaitons donc comprendre dans un premier temps la structure qui se dégage d'une catégorie de continuation construite à partir d'une catégorie de dialogue (i.e. une catégorie monoïdale symétrique munie d'une négation tensorielle). Il apparaît, après réflexion, que l'objet canonique de notre étude est plus naturellement une multicatégorie de continuation. Notre tâche est alors de définir la notion de multicatégorie de contrôle afin de décrire exhaustivement la structure présente dans une multicatégorie de continuation.

Après une présentation des travaux antérieurs qui on été réalisé dans le cadre cartésien, nous introduisons les notions de base autour des multicatégories, ainsi qu'un nouvel éclairage basé sur la notion d'empreinte d'une multicatégorie. Ce point de vue singulier nous permet de donner une définition synthétique d'une multicatégorie de contrôle comme une multicatégorie équipé d'une notion d'idéal exponentiel et d'une structure prémonoïdale (voir les Définitions 1.39 et 1.44). Nous montrons en fin de chapitre que toute multicatégorie de contrôle est équivalente à une multicatégorie de continuation.

5.1 Prolégomènes

Nous souhaitons maintenant combiner harmonieusement la logique linéaire et la théorie des continuations en utilisant la logique tensorielle. Nous avons insisté à la Section 2.1.1 sur le fait que, dans la logique tensorielle, il existe une preuve canonique

$$A \quad \vdash \quad \neg\neg A \tag{5.1}$$

mais que ces deux formules n'y sont pas identiques. Si maintenant on se penche du côté des langages de programmation, il est naturel de voir l'environnement – aussi appelé contexte d'évaluation ou continuation – d'un programme de type A comme un programme du type dual $\neg A$. Il est bien connu en théorie des continuations qu'appliquer la négation une nouvelle fois sur le type $\neg A$ ne redonne pas le type A lui-même, mais plutôt le type un peu plus libéral $\neg\neg A$.

Ce phénomène est reflété en sémantique catégorique par l'existence d'une monade de continuation $\neg\neg$ associé à toute négation tensorielle sur une catégorie monoïdale $\mathcal{C}$ (voir Section 2.1.3). La preuve canonique 5.1 est ensuite interprétée dans la catégorie $\mathcal{C}$ par l'unité de la monade.

Il est aussi bien connu en théorie des continuations que la monade $\neg\neg$ est forte (voir Définition 1.47). Ceci crée donc un lien tangible entre continuation et négation tensorielle. Nous aimerions les relier plus précisément.

D'après la Proposition 1.50, on sait que la catégorie de Kleisli associée à la monade de continuation est prémonoïdale car la monade est forte. Ceci est à mettre en parallèle avec l'observation clé qu'il est toujours possible avec une négation tensorielle de définir la disjonction $⅋$ à partir de la conjonction $\otimes$ en utilisant l'égalité de De Morgan

$$A \quad ⅋ \quad B \quad := \quad \neg \; (\; \neg A \quad \otimes \quad \neg B \;).$$

Il faut cependant faire attention car la disjonction ainsi définie n'est plus associative. En fait, la bonne façon de faire est de définir *simultanément* la famille de disjonctions n-aires :

$$[\, A_1 ⅋ \cdots ⅋ A_n \,] \quad := \quad \neg \; (\; \neg A_1 \quad \otimes \quad \cdots \quad \otimes \quad \neg A_n \;).$$

Cette famille de disjonctions n-aires s'avère être associative, mais au sens relâché : elle définit ce que Tom Leinster appelle un produit monoïdal relâché. Typiquement, il existe des morphismes canoniques :

$$[A_1 ⅋ [A_2 ⅋ A_3]] \quad \to \quad [A_1 ⅋ A_2 ⅋ A_3] \quad \longleftarrow \quad [[A_1 ⅋ A_2] ⅋ A_3]$$

qui ne sont pas forcément des isomorphismes. Il y a aussi un morphisme canonique

$$A \quad \to \quad [A]$$

qui coïncide avec l'unité de la monade de continuation.

Il semble donc important de comprendre le rapport entre la structure induite sur la catégorie de Kleisli par la monade de continuation et les structures logiques induites par la négation tensorielle. Deux cas particulièrement importants ont été étudiés en détail dans la littérature. Le cas où la monade de continuation est commutative ; et le cas où elle ne l'est pas, mais où le produit tensoriel est cartésien.

Première situation : une monade commutative. Comme nous l'avons rappelé à la Section 2.3.5, lorsque la monade de continuation est commutative, la catégorie de Kleisli associée définit un modèle de logique linéaire. Rappelons que dans ce cas, la catégorie de Kleisli $\mathbf{K}$ est la catégorie dont les objets sont ceux de $\mathbf{C}$, et les flèches sont décrites par l'égalité

$$\mathbf{K}(A, B) = \mathbf{C}(A, \neg\neg B) = \mathbf{C}(\neg B, \neg A).$$

Malheureusement, la monade de continuation $\neg\neg$ n'est pas commutative dans beaucoup de situations familières : typiquement, lorsque $\neg A$ est défini comme $A \Rightarrow \bot$ pour un certain objet $\bot$ d'une catégorie cartésienne fermée. Dans ce cas, la catégorie de Kleisli n'est pas nécessairement $*$-autonome.

Deuxième situation : un tenseur cartésien. Lorsque le produit tensoriel $\otimes$ est cartésien et que la catégorie $\mathbf{C}$ est munie des coproduits finis (notés $\oplus$), un phénomène intéressant se produit. En effet, on voit que la catégorie opposée $\mathbf{K}^{op}$ est alors cartésienne fermée. C'est la structure cocartésienne de $\mathbf{C}$ qui se transpose à la catégorie $\mathbf{K}$, et devient une structure cartésienne sur $\mathbf{K}^{op}$. Ainsi, le produit cartésien de deux objets A et B de $\mathbf{K}^{op}$ est défini comme le coproduit $A \oplus B$. La série de bijections naturelles

$$\begin{array}{rcl} \mathbf{K}^{op}(A \oplus B, C) & \cong & \mathbf{C}(\neg(A \oplus B), \neg C) \\ & \cong & \mathbf{C}(\neg A \otimes \neg B, \neg C) \\ & \cong & \mathbf{C}(\neg A, \neg(\neg B \otimes C)) \\ & \cong & \mathbf{K}^{op}(A, \neg B \otimes C) \end{array}$$

établit que $\neg B \otimes C$ définit l'implication intuitionniste $B \multimap C$ de l'objet B vers l'objet C de la catégorie $\mathbf{K}^{op}$. Cela fournit une recette de base pour construire une catégorie cartésienne fermée à partir de la catégorie cartésienne $\mathbf{C}$ équipée d'une négation.

À ce niveau, il est intéressant de clarifier quel type de catégories cartésiennes fermées sont de la forme $\mathbf{K}^{op}$ pour une catégorie cartésienne $\mathbf{C}$ munie d'une négation et de coproduits finis. En particulier, il doit rester quelque chose du produit cartésien $\otimes$ de la catégorie de départ $\mathbf{C}$ dans la catégorie induite $\mathbf{K}^{op}$. Un résultat général de John Power et Edmund Robinson (Proposition 1.50) indique que fournir une force pour la monade $\neg\neg$ est la même chose que fournir une structure prémonoïdale sur la catégorie de Kleisli $\mathbf{K}$ telle que le foncteur canonique $U_{\neg\neg} : \mathbf{C} \to \mathbf{K}$ est une foncteur prémonoïdal strict. Cela signifie en particulier que tous objets A et B de la catégorie $\mathbf{K}$ définissent deux foncteurs :

$$B \mapsto A \otimes_L B \qquad\qquad A \mapsto A \otimes_R B$$

qui coïncident sur les objet, au sens où l'objet $A \otimes_L B$ est égal à l'objet $A \otimes_R B$ – cet objet est généralement noté $A \otimes B$. Remarquons en particulier que le diagramme

$$\begin{array}{ccc} A \otimes B & \xrightarrow{f \otimes_L B} & A' \otimes B \\ {\scriptstyle A \otimes_R g}\downarrow & & \downarrow{\scriptstyle A' \otimes_R g} \\ A \otimes B' & \xrightarrow{f \otimes_L B'} & A' \otimes B' \end{array} \tag{5.2}$$

ne commute pas nécessairement pour tous $f : A \to A'$ et $g : B \to B'$. Intuitivement, chaque bord du diagramme capture un ordre particulier d'évaluation des composants f et g. Une étude plus détaillée des catégories prémonoïdales est donnée à la Section 1.3.5.

Dans son travail sur les catégories de contrôle réalisé en 2001, Peter Selinger [Sel01] caractérise les catégories cartésiennes fermées de la forme $\mathbf{K}^{op}$ pour une catégorie cartésienne fermée $\mathbf{C}$ munie d'une négation et de produits finis. À dessein, il commence par

remarquer que la structure prémonoïdale ($\otimes$) dont nous venons de parler peut être vue alternativement comme une structure prémonoïdale ($\parr$) dans la catégorie opposé $\mathbf{K}^{op}$. Selinger axiomatise les propriétés de la structure prémonoïdale $\parr$ au sein de la catégorie cartésienne close $\mathbf{K}^{op}$, en commençant par l'observation clé que l'isomorphisme

$$(\neg A \otimes B) \otimes C \cong \neg A \otimes (B \otimes C)$$

dans la catégorie $\mathbf{C}$ induit un isomorphisme

$$(A \multimap B) \parr C \cong A \multimap (B \parr C)$$

dans la catégorie $\mathbf{K}^{op}$. Cela le mène à la définition d'une *catégorie de contrôle*. Selinger établit ensuite un théorème de structure, qui stipule que toute catégorie de contrôle $\mathbf{P}$ est de la forme $\mathbf{K}^{op}$ pour une certaine catégorie cartésienne $\mathbf{C}$ munie d'une négation et de coproduits finis. Le cœur de la preuve réside dans la définition de la catégorie $\mathbf{C}$ comme étant le centre de la catégorie $\mathbf{P}$. Cette sous-catégorie du centre se trouve être cartésienne munie d'une négation et de coproduits finis. De plus, la catégorie associée $\mathbf{K}^{op}$ est équivalente à la catégorie d'origine $\mathbf{P}$ – équivalente au sein de la 2-catégorie des catégories de contrôle.

Plus tard en 2003, Selinger [Sel03] explique comment se passer de l'hypothèse que la catégorie $\mathbf{C}$ possède les coproduits finis. Rappelons que la catégorie $\mathbf{K}^{op}$ a les mêmes objets que la catégorie $\mathbf{C}$, avec pour ensemble de morphismes

$$\mathbf{K}^{op}(A, B) = \mathbf{C}(\neg A, \neg B).$$

L'idée est de remplacer $\mathbf{K}_1 = \mathbf{K}^{op}$ par la catégorie $\mathbf{K}_2$ dont les objets $(A_1, \ldots, A_m)$ sont les suites finis d'objets de $\mathbf{C}$, avec pour ensemble de morphismes

$$\mathbf{K}_2((A_1, \ldots, A_m), (B_1, \ldots, B_n)) = \mathbf{C}(\neg A_1 \otimes \cdots \otimes \neg A_m, \neg B_1 \otimes \cdots \otimes \neg B_n).$$

Les deux catégories $\mathbf{K}_1 = \mathbf{K}^{op}$ et $\mathbf{K}_2$ sont équivalentes (au sein de la 2-catégorie des catégories cartésiennes fermées) dès que la catégorie de départ $\mathbf{C}$ est munie des coproduits finis, car tout objet $(A_1, \ldots, A_m)$ est isomorphe à l'objet singleton $(A_1 \oplus \cdots \oplus A_m)$ dans la catégorie $\mathbf{K}_2$. Cependant, contrairement à la catégorie $\mathbf{K}_1 = \mathbf{K}^{op}$, la catégorie $\mathbf{K}_2$ est cartésienne avec comme produit cartésien la concaténation des objets – et ce même lorsque la catégorie $\mathbf{C}$ n'a pas de coproduits finis. On peut alors montrer que la catégorie $\mathbf{K}_2$ est cartésienne fermée en observant que tout morphisme

$$\neg A_1 \otimes \cdots \otimes \neg A_m \quad \to \quad \neg B_1 \otimes \cdots \otimes \neg B_n$$

de la catégorie $\mathbf{C}$ se décompose (car $\otimes$ est cartésien) en n morphismes

$$\neg A_1 \otimes \cdots \otimes \neg A_m \quad \to \quad \neg B_i$$

correspondant à n morphismes allant dans les objets singletons (B_i) de la catégorie $\mathbf{K}_2$.

Cela permet de définir une clôture $\multimap$ sur la catégorie $\mathbf{K}_2$ à partir de l'équation ci-dessous

$$(A_1, \ldots, A_m) \multimap B := (\neg A_1 \otimes \cdots \otimes \neg A_m \otimes B) \tag{5.3}$$

et de l'isomorphisme bien connu

$$X \multimap (B_1, \ldots, B_n) \cong (X \multimap B_1, \ldots, X \multimap B_n)$$

valide dans toute catégorie cartésienne. Concrètement, Selinger montre que la catégorie $\mathbf{K}_2$ définit une catégorie de contrôle, et étend ainsi son résultat aux catégories cartésiennes $\mathbf{C}$ munies d'une négation mais n'ayant pas forcément les coproduits finis.

Situation Générale. Nous étudions maintenant comment la théorie des catégories de contrôle peut être étendue à la situation générale d'une catégorie monoïdale $\mathbf{C}$ munie d'une négation – sans supposer que le produit tensoriel $\otimes$ de $\mathbf{C}$ soit cartésien. La situation est apparemment sans espoir : la catégorie $\mathbf{K}_1 = \mathbf{K}^{op}$ est prémonoïdale, mais pas monoïdale, et la catégorie $\mathbf{K}_2$ est monoïdale, mais pas fermée. Laquelle de ces deux catégories $\mathbf{K}_1$ ou $\mathbf{K}_2$ devons nous considérer ?

Et bien notre point de départ est de considérer les deux en même temps, ou plus précisément, le foncteur plein et fidèle

$$\mathcal{M} : \mathcal{I} \longrightarrow \mathbf{M} \tag{5.4}$$

qui transporte la catégorie prémonoïdale $\mathcal{I} = \mathbf{K}_1$ vers la catégorie monoïdale environnante $\mathbf{M} = \mathbf{K}_2$.

Comme nous l'avons déjà mentionné plus haut, la catégorie $\mathbf{M} = \mathbf{K}_2$ n'est pas fermée. Pourtant, on peut retrouver quelque chose de la fermeture présente dans la cas cartésien en affaiblissant la notion de fermeture vers celle d'idéal exponentiel (voir la Définition 1.39). En effet, il n'est pas difficile d'établir que le foncteur $\mathcal{M}$ défini par l'Équation (5.4) induit un idéal exponentiel, où $\mathcal{I}$ est la sous-catégorie pleine de $\mathbf{M}$ constituée des objets *singletons*, et où l'implication linéaire est définie de la même façon que précédemment (5.3) :

$$(A_1, \ldots, A_k) \multimap B \quad \stackrel{\text{def}}{=} \quad (\neg A_1 \otimes \cdots \otimes \neg A_m \otimes B).$$

La tâche principale de ce chapitre est donc de décrire axiomatiquement comment les différents connecteurs : $\otimes$, $\multimap$ et $\parr$ interagissent avec la structure fonctorielle $\mathcal{M}$. Cela fournit une généralisation naturelle aux catégories monoïdales du travail de Peter Selinger sur les catégories de contrôle.

Nous serons guidés dans notre recherche par l'observation que le foncteur $\mathcal{M}$ définit une notion relâchée de multicatégorie, pouvant être *normalisée* de manière fonctorielle pour retrouver une multicatégorie au sens propre (voir la Section 5.2.2 sur l'empreinte d'une multicatégorie). La multicatégorie résultante possède les mêmes objets que la catégorie $\mathcal{I}$ et ses morphismes d'un n-uplet d'objets $(A_1, \ldots, A_n)$ vers un objet B sont définis par

$$\begin{array}{rcl} \mathcal{N}(A_1, \ldots, A_n; B) & \stackrel{\text{def}}{=} & \mathbf{M}(A_1 \otimes \cdots \otimes A_n, B) \\ & = & \mathbf{C}(\neg A_1 \otimes \cdots \otimes \neg A_n, \neg B). \end{array}$$

À partir de là, on se sent à nouveau en terrain balisé. Rappelons nous que la catégorie de contrôle $\mathbf{K}^{op}$ est l'opposé de la catégorie de Kleisli $\mathbf{K}$ induite par la monade de continuation de la catégorie $\mathbf{C}$. De la même manière, la multicatégorie $\mathcal{N}$ est l'opposé de la (co-)multicatégorie induite par notre notion généralisée de monade de continuation, induite par la disjonction relâchée $[A_1 \parr \cdots \parr A_n]$ discutée au début de ces prolégomènes,

$$\begin{array}{rcl} \mathcal{N}(A_1, \ldots, A_n; B) & = & \mathbf{C}(B, \neg(\neg A_1 \otimes \cdots \otimes \neg A_n)) \\ & = & \mathbf{C}(B, [A_1 \parr \cdots \parr A_n]). \end{array}$$

Ainsi, notre analyse des continuations linéaires nécessite la généralisation simultanée de la notion de monade de continuation, et de la notion de catégorie de Kleisli. Conséquemment, notre théorème de structure de la Section 5.3.5 caractérise la multicatégorie $\mathcal{N}$ obtenue par cette construction de Kleisli généralisée.

Travaux connexes. Yves Lafont, Bernhard Reus et Thomas Streicher [LRS93] ont observé, en utilisant un traduction par continuation, que le fragment de la logique intuitionniste avec la négation au lieu de l'implication est suffisante pour interpréter le λ-calcul.

Plus tard, Martin Hofmann et Thomas Streicher [HS97] ont défini une sémantique du $\lambda\mu$-calcul en appel par nom à l'aide d'une sémantique catégorique des continuations induite par une catégorie de réponses. Ils ont établi de plus que leurs modèles des continuations sont complets parmi tous les modèles du $\lambda\mu$-calcul en appel par nom.

La première tentative de caractérisation algébrique des catégories de continuation a été faite par Hayo Thielecke [Thi97] au cours de son étude des $\otimes\neg$-catégories. Pour cela, Thielecke a introduit l'idée importante que ces catégories doivent être prémonoïdales et non monoïdales. Comme nous l'avons rappelé à la Section 2.1.2, il observa également que la négation définit un foncteur autoadjoint dont la monade associée est la monade de continuation.

Peter Selinger a ensuite introduit la notion de catégorie de contrôle [Sel01] et a établi un théorème fondamental de structure, stipulant que de telles catégories sont toutes des catégories de continuation d'une catégorie de réponses particulière. Notre travail est largement inspiré de ce résultat, et des techniques que Selinger a introduites pour le démontrer.

Une autre classe de modèle, basée sur les fibrations, a été introduite dans un premier temps par Luke Ong pour le $\lambda\mu$-calcul d'origine en appel par nom [Ong96], puis par David Pym et Eike Ritter pour le $\lambda\mu$-calcul disjonctif [PR01]. Ces modèles clarifient la nature fibrée du $\lambda\mu$-calcul par rapport aux contextes de contrôle. Ces catégories fibrées offrent une structure algébrique riche, dans laquelle le $\lambda\mu$-calcul est retrouvé comme langage interne. Il y a une traduction dans un sens comme dans l'autre entre les modèles fibrées du $\lambda\mu$-calcul et les catégories de contrôle. Cette traduction est basée sur l'idée qu'un objet A dans une fibre Δ doit être identifié avec l'objet $A \parr \Delta$ de la catégorie de contrôle. John Power et Edmund Robinson ont observé que ce schéma est une instance d'une construction générale, définissant une fibration à partir d'une structure prémonoïdale, dont les fibres sont données par les objets comonoïdaux de la catégorie [PR97].

Dans sa thèse, Olivier Laurent [Lau02a] suggéra une notion de catégorie de contrôle linéaire, à mi-chemin entre les notions de catégorie de contrôle et de catégorie linéairement distributive de Robin Cockett, et Robert Seely [CS92]. Laurent a établit que de telles catégories fournissaient toutes un modèle de MALLP, une version polarisée du fragment multiplicatif et additif de la logique linéaire. Contrairement aux catégories de contrôle, la définition d'une catégorie de contrôle linéaire n'essaye pas de capturer les propriétés précises des modèles de continuation induits par une catégorie de réponses monoïdale. Notre tâche est donc précisément de combler ce manque.

5.2 Préliminaires

5.2.1 Multicatégories

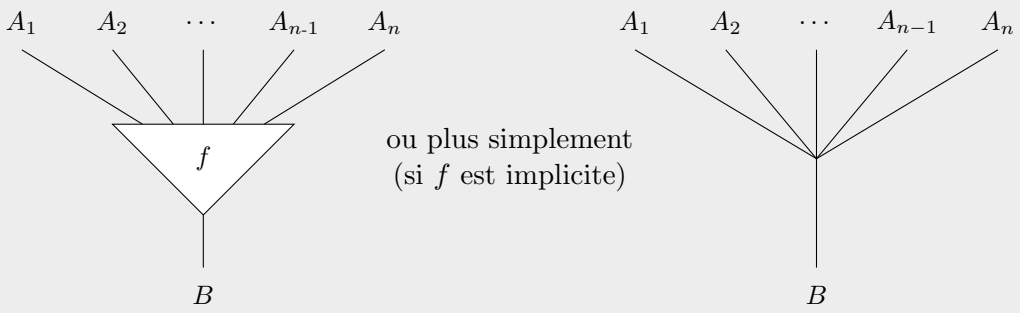
Nous rappelons ici brièvement les structures de base de la théorie des multicatégories. Le lecteur est prié de se reporter au livre de Tom Leinster [Lei04] pour de plus amples informations.

Définition 5.1 (multicatégorie)
Une *multicatégorie* $\mathcal{M}$ est la donnée

- d'une classe $\mathcal{M}_0$ dont les éléments sont appelés *objets* ;
- d'une classe $\mathcal{M}(A_1, \ldots, A_n; B)$ pour chaque $n \in \mathbb{N}$ et $A_1, \ldots, A_n, B \in \mathcal{M}_0$ dont les éléments sont appelés *morphismes* et sont souvent notés

$$f \quad : \quad A_1, \ldots, A_n \to B,$$

ce qui donne de manière graphique



– d'une fonction

$$\mathcal{M}(A_1,\dots,A_n;B)\times\mathcal{M}(A_1^1,\dots,A_1^{k_1};A_1)\times\cdots\times\mathcal{M}(A_n^1,\dots,A_n^{k_n};A_n)\longrightarrow\mathcal{M}(A_1^1,\dots,A_1^{k_1},\dots,A_n^1,\dots,A_1^{k_n};B)$$

pour chaque $n,k_1,\dots,k_n\in\mathbb{N}$ et $A_i,A_i^j,B\in\mathcal{M}_0$ appelé *composition* – notée

$$(g,f_1,\dots,f_n)\mapsto g\circ(f_1,\dots,f_n);$$

– d'un élément $\mathrm{id}_A\in\mathcal{M}(A;A)$ pour chaque $A\in\mathcal{M}_0$ appelé *identité* sur A ;

satisfaisant

– <u>associativité</u> :

$$h\circ(g_1\circ(f_1^1,\dots,f_1^{k_1}),\dots,g_n\circ(f_n^1,\dots,f_n^{k_1}))$$

à chaque fois que h, g_i, et f_i^j sont des morphismes pour lesquels ces compositions font sens (voir la Figure 5.1 pour un exemple) ;

– <u>identité</u> :

$$f\circ(\mathrm{id}_{A_1},\dots,\mathrm{id}_{A_n})=f=\mathrm{id}_B\circ f$$

pour tout morphisme $f:A_1,\dots,A_n\to B$.

Une multicatégorie P à un seul objet est appelé une *opérade*. Ses morphismes sont alors réduits à des opérations P_n de n dans 1.

Exemple 5.2

Les groupes symétriques $(\mathrm{S}_n)_{n\in\mathbb{N}}$ définissent naturellement une opérade $\mathbf{S}$ avec $\mathbf{S}_n=\mathrm{S}_n$. Nous nous contentons de donner la loi de composition sur un exemple :

$$\begin{array}{rcl}\mathrm{S}_3\times(\mathrm{S}_2\times\mathrm{S}_4\times\mathrm{S}_3) & \longrightarrow & \mathrm{S}_9\\ (\sigma,(\pi_1,\pi_2,\pi_3)) & \longmapsto & \sigma\circ(\pi_1,\pi_2,\pi_3)\end{array}$$

avec $\sigma=(2\ 1\ 3)$, $\pi_1=(2\ 1)$, $\pi_2=(3\ 1\ 2\ 4)$ et $\pi_3=(2\ 3\ 1)$ donne

$$\sigma\circ(\pi_1,\pi_2,\pi_3)=(5\ 3\ 4\ 6\ 2\ 1\ 8\ 9\ 7)$$

La Figure 5.2 décrit cette composition de manière graphique.

On dit qu'une multicatégorie est *symétrique* lorsqu'elle est munie d'une fonction

$$-\cdot\sigma\ :\ \mathcal{M}(A_1,\dots,A_n;A)\xrightarrow{\sim}\mathcal{M}(A_{\sigma(1)},\dots,A_{\sigma(n)};A)$$

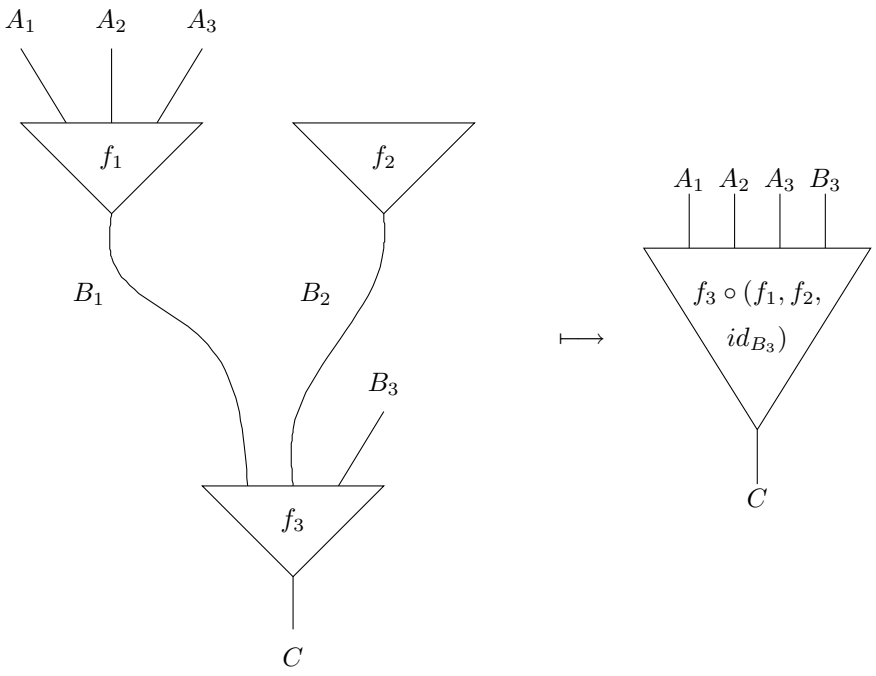


FIG. 5.1 – Exemple de morphismes composables.

pour tous objets $A_1, \ldots, A_n, A \in \mathcal{M}_0$ et pour toute permutation $\sigma \in \mathrm{S}_n$. On demande que cette fonction soit compatible avec l'identité et la composition des permutations

$$(f \cdot \sigma_1) \cdot \sigma_2 = f \cdot (\sigma_1 \sigma_2) \qquad f = f \cdot 1_{\mathrm{S}_n}$$

où f est un morphisme, et σ_1, σ_2 sont de permutations. Ceci garantit que $- \cdot \sigma$ soit une bijection. On demande aussi à ce que la symétrie soit compatible avec la composition de morphismes

$$(g \cdot \sigma_1) \circ (f_{\sigma(1)} \cdot \pi_1, \ldots, f_{\sigma(n)} \cdot \pi_n) = (g \circ (f_1, \ldots, f_n) \cdot (\sigma \circ (\pi_1, \ldots, \pi_n))$$

où $g, f_1, \ldots, f_n$ sont de morphismes, $\sigma, \pi_1, \ldots, \pi_n$ sont des permutations tels que les expressions ci-dessus ont un sens. La permutation $\sigma \circ (\pi_1, \ldots, \pi_n)$ dans le membre de droite est donnée par la composition dans l'opérade **S**.

Exemple 5.3
Toute catégorie monoïdale $\mathcal{A}$ induit une multicatégorie $\mathcal{U}(\mathcal{A})$ dont les objets sont ceux de $\mathcal{A}$ et où les virgules sont comprises comme des tenseurs, au sens où :

$$\mathcal{U}(\mathcal{A})(A_1, \ldots, A_n; B) \stackrel{\text{def}}{=} \mathcal{A}(A_1 \otimes \ldots \otimes A_n, B)$$

De la même façon, toute catégorie monoïdale symétrique induit une multicatégorie symétrique.

Lorsque la catégorie monoïdale considérée n'est pas stricte, il y a une ambiguïté dans ce que nous entendons par $A_1 \otimes \ldots \otimes A_n$. Une jolie manière de s'en extraire est de considérer

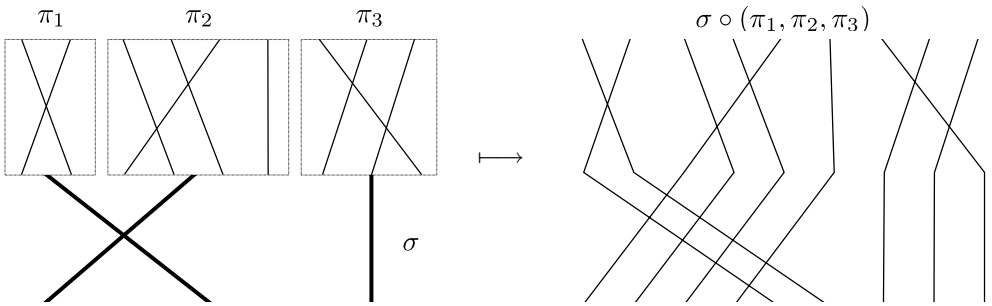


FIG. 5.2 – Composition dans l'opérade des permutations **S**.

les catégories monoïdales non biaisées comme l'a fait Tom Leinster [Lei04]. Comme ce qui nous intéresse ici est l'adjonction entre les multicatégories et les catégories monoïdales strictes, ce problème va disparaître dans la suite de ce chapitre.

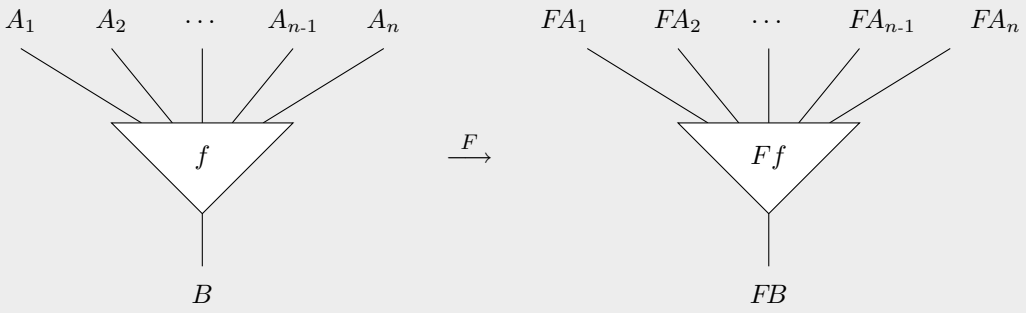
Définition 5.4 (morphisme de multicatégories)
Soit $\mathcal{M}$ et $\mathcal{N}$ deux multicatégories. Un *morphisme de multicatégories* $F : \mathcal{M} \to \mathcal{N}$ est la donnée

- d'une fonction $F_0 : \mathcal{M}_0 \to \mathcal{N}_0$ (souvent simplement notée F) qui à tout objet A de $\mathcal{A}$ associe un objet $F_0 A$ de $\mathcal{B}$;
- d'une fonction

$$F_{A_1,\dots,A_n,B} : \mathcal{M}(A_1, \dots, A_n; B) \to \mathcal{N}(F(A_1), \dots, F(A_n); F(B))$$

(souvent simplement notée F) pour chaque objet $A_1, \dots, A_n, B \in \mathcal{M}_0$;

qui préserve la composition et les identités.



Un morphisme de multicatégories $F : \mathcal{M} \to \mathcal{N}$ est dit *symétrique* lorsque $\mathcal{M}$ et $\mathcal{N}$ sont symétriques et F préserve la symétrie.

Notation 5.5
On note **MultiCat** la catégorie des multicatégories et morphismes de multicatégories. De même, on note **SMultiCat** la catégorie des multicatégories symétriques et morphismes symétriques de multicatégories symétriques.

La construction $\mathcal{U}$ donnée plus haut définit une foncteur

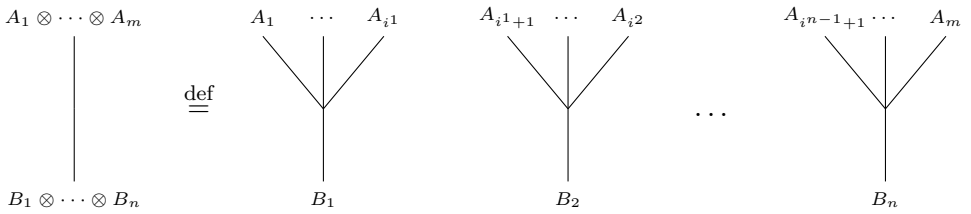
$$\mathcal{U} \quad : \quad \mathbf{MonCat_{str}} \longrightarrow \mathbf{MultiCat}$$

de la catégorie $\mathbf{MonCat_{str}}$ des catégories monoïdales et foncteurs monoïdaux stricts à la catégories **MultiCat**.

Réciproquement, toute multicatégorie $\mathcal{M}$ donne lieu à une catégorie monoïdale $\mathcal{F}(\mathcal{M})$ dont les objets sont les suites finies $(A_1, \ldots, A_n)$ d'objets de $\mathcal{M}$ concaténés par le produit tensoriel

$$A_1 \otimes \cdots \otimes A_n \quad \stackrel{\text{def}}{=} \quad (A_1, \ldots, A_n)$$

et dont les morphismes sont les suites finies de morphismes de $\mathcal{M}$ mis côte à côte comme le montre l'illustration ci-dessous :



Il est bien connu que cette construction définit un foncteur

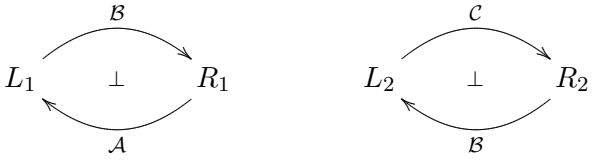
$$\mathcal{F} \quad : \quad \mathbf{MultiCat} \longrightarrow \mathbf{MonCat_{str}}.$$

Proposition 5.6 Le foncteur $\mathcal{F}$ est adjoint à gauche du foncteur d'oubli $\mathcal{U}$. Comme de plus cette adjonction est monadique, $\mathcal{FM}$ définit la catégorie monoïdale libre sur la multicatégorie $\mathcal{M}$.

Nous allons maintenant expliquer comment cette adjonction permet de voir toute multicatégorie comme un foncteur d'une catégorie « d'objets » vers une catégorie monoïdale stricte des « morphismes » – ce que nous appelons *l'empreinte d'une multicatégorie*.

5.2.2 Empreinte d'une multicatégorie

Empreintes. Supposons que nous disposons de trois catégories $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$ et de deux adjonctions



Les adjonctions $L_1 \dashv R_1$ et $L_2 \dashv R_2$ induisent une comonade $L_1 \circ R_1$ et une monade $R_2 \circ L_2$ sur la catégorie $\mathcal{B}$. Tout objet B de la catégorie $\mathcal{B}$ est ainsi équipé d'une paire de morphismes

$$L_1 \circ R_1(B) \xrightarrow{\varepsilon_B} B \xrightarrow{\eta_B} R_2 \circ L_2(B), \tag{5.5}$$

construite avec la counité ε de la comonade et l'unité η de la monade, toutes les deux appliquées à l'objet B. Imaginons un instant que la composée de ces deux morphismes

$$L_1 \circ R_1(B) \longrightarrow R_2 \circ L_2(B) \tag{5.6}$$

caractérise l'objet B, à isomorphisme près. Dans ce cas, on dit que cette composée est *empreinte* de l'objet B.

Cette situation arrive typiquement lorsque la catégorie $\mathcal{B}$ est munie d'un système de factorisation $(\mathcal{E}, \mathcal{M})$ (cf. Définition 1.26) et que de plus le morphisme counité ε_B appartient à $\mathcal{E}$ et le morphisme unité η_B appartient à $\mathcal{M}$. Dans ce cas, on peut retrouver, à isomorphisme près, la paire de morphismes (5.5) en factorisant le morphisme (5.6) comme la composée d'un morphisme de $\mathcal{E}$ et d'un morphisme de $\mathcal{M}$.

Remarquons que les adjonctions nous permettent de voir l'empreinte (5.6) ou bien comme un morphisme

$$R_1(B) \longrightarrow\!\!\!\!\!\!\rightarrow R_1 \circ R_2 \circ L_2(B)$$

de la catégorie $\mathcal{A}$, ou bien comme un morphisme

$$L_2 \circ L_1 \circ R_1(B) \longrightarrow\!\!\!\!\!\!\rightarrow L_2(B)$$

de la catégorie $\mathcal{C}$.

Empreinte d'une multicatégorie L'illustration qui nous intéresse s'obtient en instanciant $\mathcal{A}$ avec la catégorie **Cat**, $\mathcal{B}$ avec la catégorie **MultiCat** et $\mathcal{C}$ avec la catégorie $\mathbf{MonCat_{str}}$. Le système de factorisation dans **MultiCat** est donné par l'ensemble des morphismes bijectifs sur les objets pour $\mathcal{E}$ et l'ensemble des morphismes pleins et fidèles pour $\mathcal{M}$.

Le foncteur

$$L_1 \quad : \quad \mathbf{Cat} \longrightarrow \mathbf{MultiCat}$$

transporte toute catégorie A vers la multicatégorie $L_1(A)$ ayant les mêmes objets et les mêmes morphismes que A. On obtient ainsi une multicatégorie *filiforme* au sens où elle ne contient aucun morphisme avec n entrées dès que n est différent de 1. Le foncteur

$$R_1 \quad : \quad \mathbf{MultiCat} \longrightarrow \mathbf{Cat}$$

est obtenu en transportant chaque multicatégorie B vers la catégorie $R_1(B)$ obtenue en restreignant B aux morphismes ayant exactement un argument.

L'adjonction $L_2 \dashv R_2$ est l'adjonction $\mathcal{F} \dashv \mathcal{U}$ décrite plus haut.

Ceci démontre qu'une multicatégorie peut être vue comme un foncteur au sens suivant.

Définition 5.7 (empreinte d'une multicatégorie)
Soit $\mathcal{M}$ une multicatégorie. On peut la voir comme un foncteur

$$\mathcal{M} : \mathcal{I} \to \mathbf{M}$$

que nous appelons l'*empreinte* de la multicatégorie. La catégorie $\mathcal{I}$ est donnée par l'image de $\mathcal{M}$ par le foncteur d'oubli R_1 et $\mathbf{M}$ est donnée par la catégorie monoïdale stricte induite par le foncteur $\mathcal{F}$, vue comme une simple catégorie. On retrouve la multicatégorie à partir du foncteur en prenant comme objets les objets de $\mathcal{I}$ et comme morphismes de type $A_1, \ldots, A_n \to A$ les morphismes de type

$$\mathcal{M}A_1 \otimes \cdots \otimes \mathcal{M}A_n \longrightarrow \mathcal{M}A$$

de $\mathbf{M}$.

Remarquons que nous pouvons supposer sans perte de généralité – et nous le ferons par la suite – que le foncteur $\mathcal{M}$ est l'identité sur les objets.

5.3 Multicatégorie de contrôle

Nous rentrons maintenant dans le cœur de notre travail sur la continuation linéaire. Passer du cadre cartésien au cadre monoïdal dans la théorie des catégories de contrôle introduit deux nouveaux problèmes. Le premier, déjà mentionné en début de chapitre, est dû au fait que la catégorie $\mathbf{K}_2$ est monoïdale, mais pas fermée. On peut contourner ce problème en considérant la *multicatégorie de continuation* $\mathcal{N}$ induite par le foncteur

$$\mathcal{M} \quad : \quad \mathbf{K}_1 \longrightarrow \mathbf{K}_2$$

comme nous l'avons expliqué dans la Section 5.2.2. Mais il y a une autre complication provenant du fait qu'une catégorie monoïdale n'est pas nécessairement symétrique. Cela nécessite de distinguer les morphismes centraux *à gauche* des morphismes centraux *à droite* dans la catégorie $\mathbf{K}_1 = \mathbf{K}^{op}$ lorsque l'on essaye de retrouver la catégorie de départ $\mathbf{C}$ et sa négation. De manière intéressante, cela implique que l'on n'obtient pas exactement une unique catégorie $\mathbf{C}$ munie d'une négation, mais plutôt une paire de catégories $\mathbf{C}_{\text{left}}$ et $\mathbf{C}_{\text{right}}$ reliées par une paire de négations adjointes.

Nous choisissons de décrire uniquement le cas symétrique, en insistant sur le fait que passer d'un cadre cartésien à un cadre symétrique monoïdal nécessite l'emploi de multicatégories. Nous différons à des travaux ultérieurs l'étude du cas non symétrique car il nécessite de généraliser la notion même de négation introduite à la Section 2.1.1.

5.3.1 Multicatégorie de continuation

Nous allons maintenant définir formellement la structure multicatégorique induite par une négation tensorielle.

Définition 5.8 (multicatégorie de continuation)
Soit $\mathbf{C}$ une catégorie de dialogue. La *multicatégorie de continuation* associée est définie par son empreinte

$$\mathcal{M} \quad : \quad \mathbf{K}_1 \longrightarrow \mathbf{K}_2$$

où $\mathbf{K}_2$ est la catégorie dont les objets $(A_1, \ldots, A_m)$ sont les suites finies d'objets de $\mathbf{C}$, avec pour morphismes

$$\mathbf{K}_2((A_1, \ldots, A_m), (B_1, \ldots, B_n)) = \mathbf{C}(\neg A_1 \otimes \cdots \otimes \neg A_m, \neg B_1 \otimes \cdots \otimes \neg B_n),$$

$\mathbf{K}_1$ est la sous catégorie de $\mathbf{K}_2$ restreinte aux objets singletons (A) et $\mathcal{M}$ est le foncteur d'inclusion.

La notion de multicatégorie de contrôle qui nous allons maintenant introduire a pour but d'axiomatiser toute la structure présente au sein de cette multicatégorie de continuation. Cette axiomatisation sera validée par un théorème de structure.

5.3.2 Multicatégorie de contrôle

La définition de multicatégorie de contrôle repose sur la notion d'empreinte d'une multicatégorie. Cela nous permet d'importer les notions d'idéal exponentiel et de catégorie prémonoïdale dans l'univers multicatégorique sans avoir à les définir explicitement dans ce cadre.

Définition 5.9 (multicatégorie de contrôle)
Une *multicatégorie de contrôle symétrique* est une multicatégorie symétrique $\mathcal{P}$ dont l'empreinte

$$\mathcal{P} : \mathcal{I} \longrightarrow \mathbf{M}$$

plonge une catégorie prémonoïdale symétrique $(\mathcal{I}, \parr, \bot)$ à l'intérieur d'une catégorie monoïdale symétrique $(\mathbf{M}, \otimes, 1)$ et induit un idéal exponentiel

$$\multimap \quad : \quad \mathbf{M}^{op} \times \mathcal{I} \to \mathcal{I}.$$

En outre, ces structures sont reliées par une transformation

$$s_{A,B,C} : (A \multimap B) \parr C \xrightarrow{\sim} A \multimap (B \parr C)$$

naturelle en A de $\mathbf{M}$ et en B, C de $\mathcal{I}$. Cette transformation – appelée *force exponentielle* – satisfait les conditions de cohérence

$$\begin{array}{ccccc}
((A \multimap B) \parr C) \parr D & \xrightarrow{s \parr D} & (A \multimap (B \parr C)) \parr D & \xrightarrow{s} & A \multimap ((B \parr C) \parr D) \\
\downarrow{\scriptstyle \alpha} & & & & \downarrow{\scriptstyle A \multimap \alpha} \\
(A \multimap B) \parr (C \parr D) & & \xrightarrow{\qquad\qquad s \qquad\qquad} & & A \multimap (B \parr (C \parr D))
\end{array}$$

$$\begin{array}{ccc}
 & A \multimap B & \\
{\scriptstyle l}\swarrow & & \searrow{\scriptstyle A \multimap l} \\
(A \multimap B) \parr \bot & \xrightarrow{\quad s \quad} & A \multimap (B \parr \bot)
\end{array}$$

Enfin, en utilisant la symétrie σ de la catégorie prémonoïdale $\mathcal{I}$, on définit

$$s'_{A,B,C} = \sigma_{C,A \multimap B}; s_{A,B,C}; A \multimap \sigma_{B,C} : C \parr (A \multimap B) \xrightarrow{\sim} A \multimap (C \parr B)$$

et on demande à ce que la condition de cohérence suivante soit vraie pour tous objets A, B, C et D de la catégorie $\mathcal{I}$

$$\begin{array}{ccccc}
(A \multimap B) \parr (C \multimap D) & & \xrightarrow{\qquad\qquad s' \qquad\qquad} & & C \multimap ((A \multimap B) \parr D) \\
\downarrow{\scriptstyle s} & & & & \downarrow{\scriptstyle C \multimap s} \\
A \multimap (B \parr C \multimap D) & \xrightarrow{A \multimap s'} & A \multimap (C \multimap (B \parr D)) & \xrightarrow{smcc} & C \multimap (A \multimap (B \parr D))
\end{array}$$

Exemple 5.10 (multicatégorie de continuation)
Avant d'étudier plus en avant la structure d'une multicatégorie de contrôle, assurons nous qu'une multicatégorie de continuation satisfait bien tous les axiomes présentés.

Étant donnée une catégorie monoïdale symétrique $\mathbf{C}$ munie d'une négation, la catégorie $\mathbf{K}_2$ est monoïdale symétrique, avec pour produit tensoriel la concaténation des objets. Comme la catégorie $\mathbf{K}^{op}$ est équivalente à l'opposée de la catégorie de Kleisli associée à la monade de continuation, la Proposition 1.50 nous assure qu'elle est prémonoïdale symétrique.

Reste à définir l'idéal exponentiel $\multimap$ et sa force. Comme nous l'avons mentionné en introduction, l'idéal exponentiel est défini par

$$A \multimap B \stackrel{\text{def}}{=} \neg A \otimes B$$

et la force est donnée par l'associativité

$$(A \multimap B) \parr C \stackrel{\text{def}}{=} (\neg A \otimes B) \otimes C \cong \neg A \otimes (B \otimes C) \stackrel{\text{def}}{=} A \multimap (B \parr C)$$

Reste à vérifier les trois diagrammes de cohérence. Le premier est juste une instanciation particulière du pentagone de Mac Lane, le deuxième vient par naturalité de l'unité à droite et le troisième vient de la compatibilité de la symétrie avec l'associativité.

Nous allons maintenant établir que toute multicatégorie de contrôle apparaît de cette façon. Pour cela, nous allons montrer que pour toute multicatégorie de contrôle, le centre $\mathcal{Z}(\mathcal{I})$ de la catégorie prémonoïdale symétrique $\mathcal{I}$ est une catégorie monoïdale symétrique équipée d'une négation.

5.3.3 Propriété clé : La bijection du centre

Pour pouvoir étudier le centre $\mathcal{Z}(\mathcal{I})$ de la catégorie prémonoïdale symétrique $\mathcal{I}$, nous allons établir une bijection entre certains Hom-sets de $\mathcal{Z}(\mathcal{I})$ et certains Hom-sets de $\mathcal{M}$. Un premier pas vers cette bijection peut être fait en examinant ce qu'il reste de la prémonoïdalité de $\mathcal{I}$ et de l'idéal exponentiel $\multimap$ au sein de la multicatégorie $\mathcal{P}$.

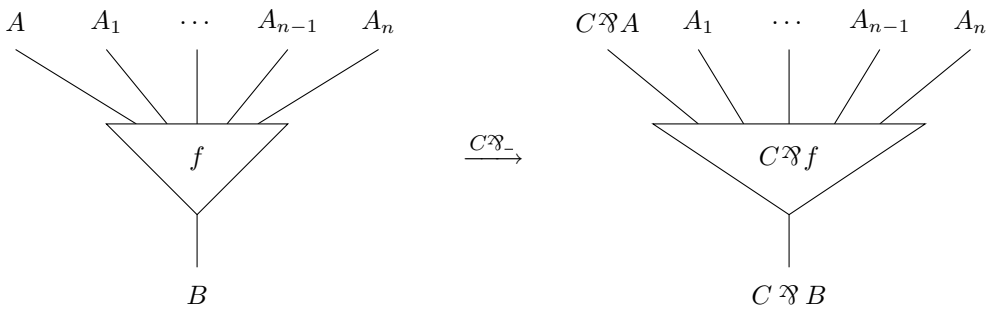
L'idéal exponentiel induit une famille de bijections

$$\varphi_{A,A_1,\ldots,A_n,B} : \frac{A, A_1, \ldots, A_n \to B}{A_1, \ldots, A_n \to A \multimap B}$$

naturelles en $A_1, \ldots A_n$, et B. On peut penser à ces bijections comme faisant partie de la définition d'une fermeture sur la multicatégorie $\mathcal{P}$. Cependant, nous ne pensons pas que ce soit nécessaire de rendre cette intuition plus précise – même si nous utiliserons cette terminologie plus loin dans le texte – car cela introduirait trop de technicité.

Notons que dans la théorie des catégories de contrôle (linéaires), le morphisme s définissant la force exponentielle est induit par une loi de distributivité (linéaire) entre le produit monoïdal et le produit prémonoïdal. Ici, on ne peut espérer avoir une telle loi de distributivité car le produit monoïdal et le produit prémonoïdal ne vivent même pas dans la même catégorie. Nous avons donc du introduire le morphisme s comme une structure additionnelle satisfaisant les lois de cohérence adéquates. Cependant, on peut avoir un point de vue plus conceptuel sur la force exponentielle.

En effet, la force exponentielle permet d'étendre la structure prémonoïdale à l'ensemble de la multicatégorie de la manière suivante. Dans toute multicatégorie de contrôle $\mathcal{P}$, il y a une flèche



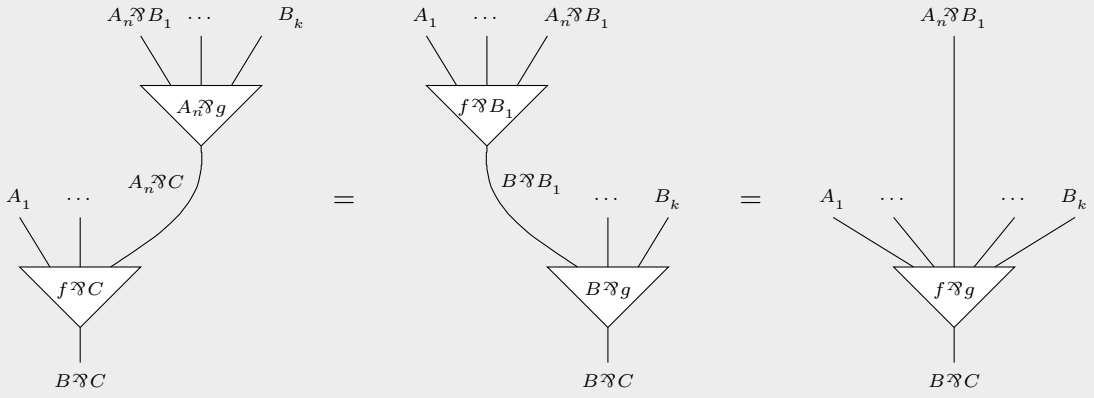
dinaturelle en C vis-à-vis des morphismes centraux, naturelle en $A_1, \ldots, A_n$ et naturelle en A, B centraux. Cette flèche est obtenue en appliquant d'abord la fermeture n fois afin d'obtenir une flèche $A \to A_1 \multimap (\ldots \multimap (A_n \multimap B))$ dans $\mathcal{I}$. Ensuite, on applique le foncteur $C \parr -$ dans $\mathcal{I}$ et on utilise la force exponentielle pour obtenir une flèche $C \parr A \to A_1 \multimap (\ldots \multimap (A_n \multimap C \parr B))$. Enfin, on ramène les A_i à gauche en utilisant à nouveau la fermeture. Remarquons que cette construction est compatible avec la fermeture au sens où

$$\varphi(f) \parr C; s_{A,B,C} = \varphi(f \parr C).$$

pour tout morphisme $f : A, A_1 \ldots, A_n \to B$. À nouveau, on peut voir cette structure comme faisant partie de la définition de ce que devrait être une multicatégorie prémonoïdale, mais nous laissons cette question épineuse à un traitement ultérieur.

Quoi qu'il en soit, la flèche $C \parr -$ et sa version symétrique $- \parr C$ permettent d'étendre la notion de centralité des morphismes de $\mathcal{I}$ aux morphismes de $\mathcal{P}$ de la manière suivante.

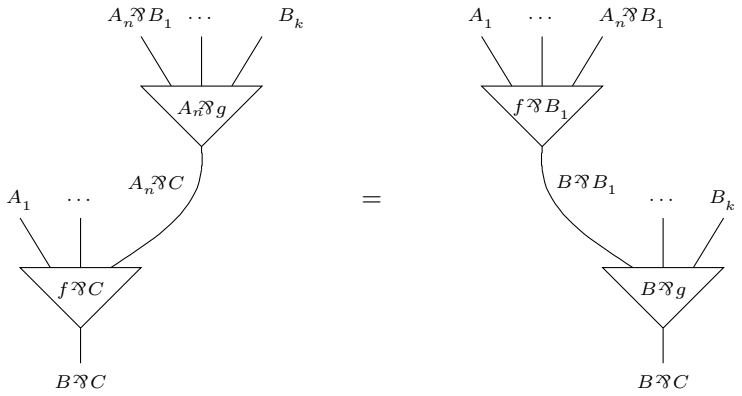
Définition 5.11 (morphisme central)
Un morphisme $f : A_1, \ldots, A_n \to B$ d'une multicatégorie de contrôle $\mathcal{P}$ est dit *central en* A_n lorsque pour tout morphisme $g : B_1, \ldots, B_k \to C$, les deux composées $(f \parr C) \circ (A_n \parr g)$ et $(B \parr g) \circ (f \parr B_1)$ sont égales. Dans ce cas, on peut utiliser la notation $f \parr g$ qui n'est pas ambiguë. Graphiquement, cette égalité devient



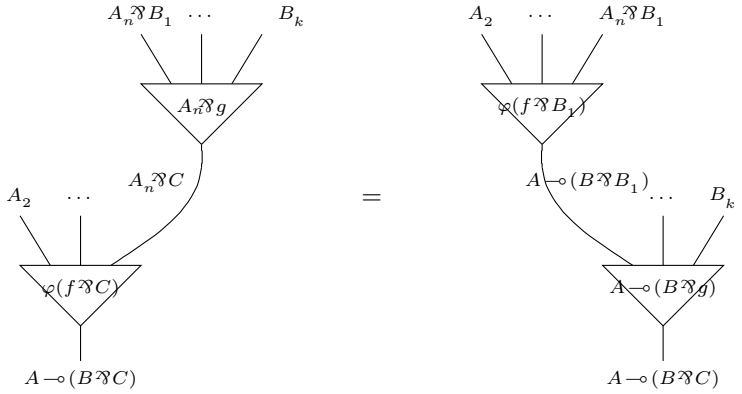
On n'a pas besoin ici de définir les morphismes centraux à gauche et à droite car les produits tensoriels considérés sont tous symétriques. Il est important de noter que cette notion étendue de centralité est préservée par la fermeture comme l'indique le lemme suivant.

Lemme 5.12 Dans une multicatégorie de contrôle $\mathcal{P}$, un morphisme $f : A, A_1, \ldots, A_n \to B$ est central en A_n si et seulement si $\varphi(f) : A_1, \ldots, A_n \to A \multimap B$ est central en A_n.

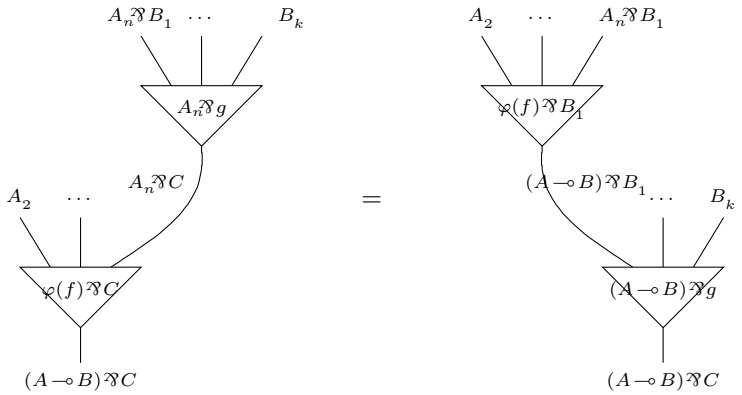
Démonstration : Soit $g : B_1, \ldots, B_k \to C$ un morphisme de $\mathcal{P}$. L'égalité



est équivalente à l'égalité



par naturalité de φ, qui est à son tour équivalente à l'égalité



par compatibilité de $\parr$ avec la fermeture. ■

On peut aussi montrer que lorsqu'un morphisme est central en A, on peut précomposer les autres arguments laissés « libres » par n'importe quel morphisme, le morphisme obtenu reste toujours central. Plus formellement, on a le lemme suivant.

Lemme 5.13 Dans une multicatégorie de contrôle $\mathcal{P}$, si un morphisme

$$f : A_1, \ldots, A_n, A \to B$$

est central en A alors le morphisme $(f_1, \ldots, f_n, \mathrm{id}_A) \circ f$ est central en A quelque soit les morphismes $f_i : C_{j_1}, \ldots, C_{j_i} \to A_i$.

Démonstration : On utilise le fait que l'opération $- \parr C$ est naturel en $A_1, \ldots, A_n$ et qu'elle est naturelle en A vis-à-vis des morphismes centraux. Comme l'identité est toujours centrale, on a

$$((f_1, \ldots, f_n, \mathrm{id}_A) \circ f) \parr C = (f_1, \ldots, f_n, \mathrm{id}_{A \parr C}) \circ (f \parr C).$$

On en déduit aisément le reste de la preuve. ■

Avant d'établir cette bijection clé entre certaines flèches centrales de $\mathcal{I}$ et certains morphismes de $\mathcal{P}$, nous allons étudier le statut du tiers exclu induit par la force exponentielle. Notons tout d'abord que tous les axiomes d'une multicatégorie de contrôle sont intuitionnistiquement valides à l'exception de la présence d'un inverse pour la force exponentielle. Cet inverse induit en effet un morphisme 0-aire *tnd* (pour *tertium non datur*) dans la multicatégorie $\mathcal{P}$ qui tient lieu de tiers exclu

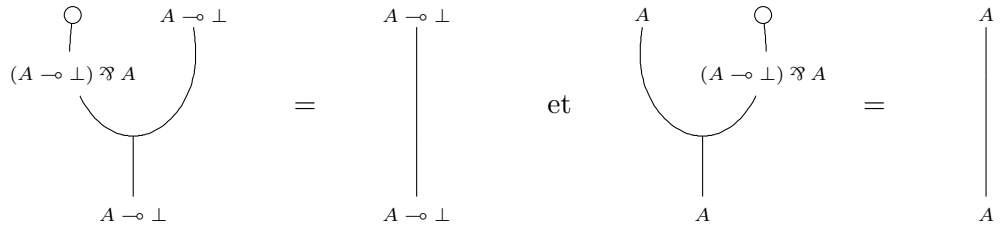
$$\xrightarrow{tnd} (A \multimap \bot) \parr A \quad \stackrel{\text{def}}{=} \quad \xrightarrow{\varphi(\mathrm{id}_A)} A \multimap A \xrightarrow{s_{A,\bot,A^{-1}}} (A \multimap \bot) \parr A$$

On le définit en appliquant la fermeture sur le morphisme identité puis en utilisant l'inverse de la force exponentielle sur $\bot$. De plus, même si cette flèche n'est pas centrale en général, elle est dinaturelle en A. Nous donnons maintenant deux égalités satisfaites par *tnd* et qui vont avoir un rôle majeur pour établir la bijection du centre.

Lemme 5.14 Les égalités

1. $[(A \multimap \bot) \parr \varphi^{-1}(\mathrm{id}_{A \multimap \bot})] \circ (tnd, \mathrm{id}_{A \multimap \bot}) = \mathrm{id}_{A \multimap \bot}$;
2. $[\varphi^{-1}(\mathrm{id}_{A \multimap \bot}) \parr A] \circ (\mathrm{id}_A, tnd) = \mathrm{id}_A$;

sont valides dans toute multicatégorie de contrôle. Graphiquement, on obtient



Remarquons que nous avons volontairement omis la mention explicite du nom des morphismes afin de rendre la représentation plus graphique. La preuve repose sur une bonne gestion des diverses naturalités.

Démonstration : Par naturalité de φ, pour tout morphisme $f : A, A \multimap \bot \to \bot$, on a

$$\varphi(f) = [(A \multimap \bot) \parr f] \circ (\varphi(\mathrm{id}_A), \mathrm{id}_{A \multimap \bot})$$

On en déduit que

$$[(A \multimap \bot) \parr \varphi^{-1}(\mathrm{id}_{A \multimap \bot})] \circ (tnd, \mathrm{id}_{A \multimap \bot}) = \varphi(\varphi^{-1}(\mathrm{id}_{A \multimap \bot}))$$

d'où la première égalité. Pour la deuxième égalité, il nous faut d'abord utiliser la compatibilité de $- \parr A$ avec φ^{-1} sur l'identité

$$\varphi^{-1}(\mathrm{id}_{A \multimap \bot}) \parr A = \varphi^{-1}((\mathrm{id}_{A \multimap \bot ; s_{A,\bot,A}}) \parr A).$$

On en déduit avec la naturalité de φ que

$$[\varphi^{-1}(\mathrm{id}_{A\multimap\bot}) \parr A] \circ (\mathrm{id}_A, tnd) = \varphi^{-1}(tnd; s^{-1}_{A,\bot,A}) = \varphi^{-1}(\varphi(\mathrm{id}_A))$$

d'où la deuxième égalité. ■

À présent, nous sommes prêts à exhiber ce que nous appelons la *bijection du centre* (à la suite de Peter Selinger). Elle permet de relier les morphismes centraux de $\mathcal{I}$ ayant pour domaine un objet nié $A \multimap \bot$ à certains morphismes 0-aires de $\mathcal{P}$.

Proposition 5.15 (bijection du centre) Dans toute multicatégorie de contrôle $\mathcal{P}$, il y a une bijection

$$\mathcal{Z}(\mathcal{I})(A \multimap \bot; B) \cong \mathcal{P}(; B \parr A)$$

naturelle en A et en B vis-à-vis des morphismes centraux.

Démonstration : Définissons les deux composantes

$$\theta_{A,B} \quad : \quad \mathcal{P}(; B \parr A) \quad \longrightarrow \quad \mathcal{Z}(\mathcal{I})(A \multimap \bot; B)$$

$$\rho_{A,B} \quad : \quad \mathcal{Z}(\mathcal{I})(A \multimap \bot; B) \quad \longrightarrow \quad \mathcal{P}(; B \parr A)$$

de la bijection.

$\theta_{A,B}$ **:** Soit f un morphisme de $\mathcal{P}(; B \parr A)$. Par le Lemme 5.12 et les propriétés du $\parr$, le morphisme h_B défini par

$$h_B \stackrel{\text{def}}{=} B \parr \varphi(\mathrm{id}_{A\multimap\bot}) \quad \in \quad \mathcal{P}(B \parr A, A \multimap \bot; B)$$

est central. Par le Lemme 5.13, le morphisme composé

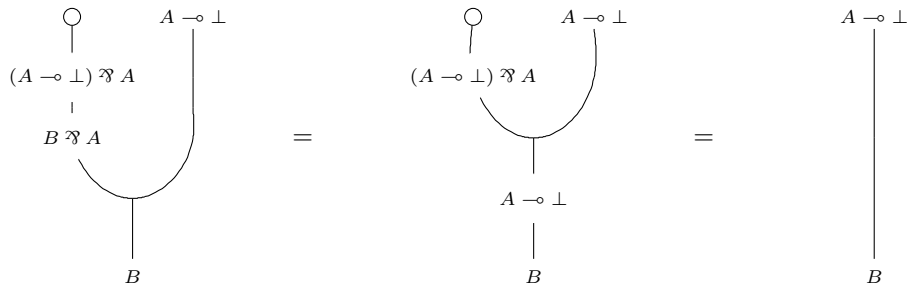
$$\theta_{A,B}(f) \quad \stackrel{\text{def}}{=} \quad h_B \circ (f, A \multimap \bot)$$

est lui aussi central.

$\rho_{A,B}$ **:** Soit g un morphisme de $\mathcal{Z}(\mathcal{I})(A \multimap \bot; B)$. Nous définissons $\rho_{A,B}$ par

$$\rho_{A,B}(g) \stackrel{\text{def}}{=} (g \parr A) \circ tnd$$

Pour s'assurer que l'égalité $\theta \circ \rho(g) = g$ est vraie pour tout g de $\mathcal{Z}(\mathcal{I})(A \multimap \bot; B)$, considérons le chaîne d'égalités suivante.



L'égalité de gauche vient de la naturalité de $B \parr -$ en B vis-à-vis des morphismes centraux. L'égalité de droite vient de la première équation du Lemme 5.14.

Pour s'assurer que l'égalité $\rho \circ \theta(f) = f$ est vraie pour tout f de $\mathcal{P}(; B \parr A)$, il suffit d'utiliser la deuxième équation du Lemme 5.14 ainsi que l'égalité $B \parr \mathrm{id}_A = \mathrm{id}_{B\parr A}$. ■

5.3.4 La 2-catégorie des multicatégories de contrôle

Afin d'établir notre théorème de structure, il nous faut une notion de morphisme et d'équivalence entre deux multicatégories de contrôle. Les définitions suivantes sont celles attendues, la seule subtilité est que l'on demande aux isomorphismes structuraux d'être centraux.

Définition 5.16
Un *morphisme de multicatégories de contrôle* entre $\mathcal{P}$ et $\mathcal{P}'$ est la donnée d'un morphisme de multicatégories $F : \mathcal{P} \to \mathcal{P}'$ ainsi que des isomorphismes centraux naturels

$$\begin{array}{rcl} FA \multimap FB & \cong & F(A \multimap B) \\ FA \parr FB & \cong & F(A \parr B) \\ \bot & \cong & F(\bot) \end{array}$$

pour chaque A et B de $\mathcal{P}$, préservant les morphismes de structures définissant une multicatégorie de contrôle de toutes les façons évidentes. Par exemple, le foncteur doit préserver la force exponentielle de la façon suivante :

$$\begin{array}{ccccc} F((A \multimap B) \parr C) & \xrightarrow{\cong} & F(A \multimap B) \parr FC & \xrightarrow{\cong} & (FA \multimap FB) \parr FC \\ {\scriptstyle Fs_{A,B,C}}\downarrow & & & & \downarrow{\scriptstyle s_{FA,FB,FC}} \\ F(A \multimap (B \parr C)) & \xrightarrow{\cong} & FA \multimap F(B \parr C) & \xrightarrow{\cong} & FA \multimap (FB \parr FC) \end{array}$$

Une *transformation naturelle* θ entre deux morphismes F et G de multicatégories de contrôle est une transformation naturelle entre les morphismes de multicatégories sous-jacents, telle que le morphisme θ_A soit central pour tout A et qui préserve les isomorphismes de structure de façon évidente. Par exemple, le diagramme

$$\begin{array}{ccc} F(A \parr B) & \xrightarrow{\cong} & FA \parr FB \\ {\scriptstyle \theta_{A \parr B}}\downarrow & & \downarrow{\scriptstyle \theta_A \parr \theta_B} \\ G(A \parr B) & \xrightarrow{\cong} & GA \parr GB \end{array}$$

commute pour tous A et B de $\mathcal{P}$.

Ces deux notions permettent de définir une 2-catégorie **ControlMult** dont les 0-cellules sont les multicatégories de contrôle, les 1-cellules sont les morphismes de multicatégories de contrôle et les 2-cellules sont les transformations naturelles. Cela permet de définir de manière 2-catégorique la notion d'équivalence entre deux multicatégories de contrôle.

Définition 5.17
Une *équivalence entre deux multicatégories de contrôle* $\mathcal{P}$ et $\mathcal{P}'$ est une équivalence dans la 2-catégorie **ControlMult**.

Nous allons maintenant montrer que toute multicatégorie de contrôle est équivalente, au sens ci-dessus, à une multicatégorie de continuation. Nous savons déjà grâce à l'Exemple

5.10 que toute multicatégorie de continuation définit une multicatégorie de contrôle. Nous allons maintenant montrer que le centre d'une multicatégorie de contrôle définit une catégorie de dialogue dont la multicatégorie de continuation associée est équivalente à la multicatégorie de contrôle de départ.

5.3.5 Théorème de structure

Soit $\mathbf{C}$ la catégorie duale de la catégorie $\mathcal{Z}(\mathcal{I})$. Comme nous l'avons rappelé en Section 1.3.5, c'est une catégorie monoïdale symétrique – nous noterons $\otimes$ son produit tensoriel. Plus intéressant, la bijection du centre nous permet de montrer que la catégorie $\mathbf{C}$ est aussi équipée d'une négation tensorielle.

Proposition 5.18 Le foncteur $\neg : \mathbf{C} \to \mathbf{C}^{\mathrm{op}}$ défini sur les objets et les flèches de $\mathbf{C}$ par

$$\neg A \stackrel{\text{def}}{=} A \multimap \bot \quad \text{et} \quad \neg f \stackrel{\text{def}}{=} f \multimap \bot$$

construit une négation tensorielle sur la catégorie $\mathbf{C}$.

Démonstration : En premier lieu, nous devons montrer que $\neg$ envoie des morphismes centraux sur des morphismes centraux. Ce fait est établi par la commutativité du diagramme suivant

$$\begin{array}{ccc} (B \multimap C) \parr D & \xrightarrow{(f \multimap C) \parr D} & (A \multimap C) \parr D \\ {\scriptstyle (B \multimap C) \parr g}\downarrow & & \downarrow{\scriptstyle (A \multimap C) \parr g} \\ (B \multimap C) \parr E & \xrightarrow[(f \multimap C) \parr E]{} & (A \multimap C) \parr E \end{array}$$

Par naturalité de la force exponentielle, cela revient à la commutativité du diagramme

$$\begin{array}{ccc} B \multimap (C \parr D) & \xrightarrow{f \multimap (C \parr D)} & A \multimap (C \parr D) \\ {\scriptstyle B \multimap (C \parr g)}\downarrow & & \downarrow{\scriptstyle A \multimap (C \parr g)} \\ B \multimap (C \parr E) & \xrightarrow[f \multimap (C \parr E)]{} & A \multimap (C \parr E) \end{array}$$

qui est obtenu par fonctorialité de l'implication $\multimap$ définissant l'idéal exponentiel. On a donc bien défini un foncteur $\neg : \mathbf{C} \to \mathbf{C}^{\mathrm{op}}$. Il reste maintenant à construire la bijection caractérisant la négation tensorielle. Nous allons utiliser la Proposition 2.2 qui nous permet de montrer uniquement que $\bot$ est exponentiable. La chaîne de bijections suivante

$$\begin{array}{rcl} \mathbf{C}(A \otimes B, \bot) & \cong & \mathcal{Z}(\mathcal{I})(\bot, A \parr B) \\ & \cong & \mathcal{P}(; A \parr B)) \\ & \cong & \mathcal{Z}(\mathcal{I})(B \multimap \bot; A) \\ & \cong & \mathbf{C}(A, \neg B) \end{array}$$

naturelle en A et B montre l'exponentiabilité de $\bot$. ∎

Maintenant que nous avons retrouvé une catégorie de dialogue au sein de notre multicatégorie de contrôle $\mathcal{P}$, il nous reste à montrer que la multicatégorie de continuation associée est équivalente à $\mathcal{P}$.

Théorème 5.19 (*théorème de structure*)

Toute multicatégorie de contrôle est équivalente à une multicatégorie de continuation associée à une catégorie de dialogue.

Démonstration : Soit $\mathcal{P} : \mathcal{I} \longrightarrow \mathbf{M}$ une multicatégorie de contrôle. Nous notons $\mathcal{N}$ la multicatégorie de continuation associée à la catégorie monoïdale symétrique $\mathbf{C} = \mathcal{Z}(\mathcal{I})^{\mathrm{op}}$ munie de la négation tensorielle

$$\neg : \mathbf{C} \to \mathbf{C}^{op}.$$

Nous allons maintenant montrer qu'il existe une équivalence de multicatégories de contrôle supportée par le morphisme F entre $\mathcal{P}$ et $\mathcal{N}$. On définit F comme étant l'identité sur les objets, la seule difficulté est alors d'établir une bijection entre les morphismes de $\mathcal{P}$ et les morphismes de $\mathcal{N}$. Cette bijection est donnée par

$$\begin{array}{rcl}
\mathcal{P}(A_1, \ldots, A_n; B) & \cong & \mathcal{P}(; A_n \multimap (\ldots \multimap (A_1 \multimap B))) \\
& \cong & \mathcal{P}(; (A_n \multimap \bot) \parr \ldots \parr (A_1 \multimap \bot) \parr B) \\
& \cong & \mathcal{Z}(\mathcal{I})(\neg B; \neg A_n \parr \cdots \parr \neg A_1) \\
& \cong & \mathbf{C}(\neg A_1 \otimes \cdots \otimes \neg A_n, \neg B) \\
& \cong & \mathcal{N}(A_1, \ldots, A_n; B)
\end{array}$$

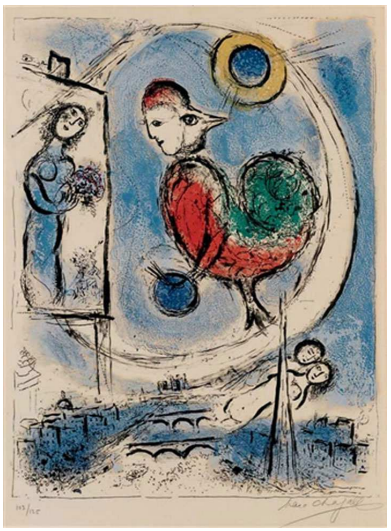
naturelle en $A_1, \ldots, A_n$. ■

Ce théorème de structure indique que nous avons complètement capturé la structure induite sur la multicatégorie de continuation par la négation tensorielle.

Ce résultat fait partie de notre programme de recherche qui a pour but de réaliser une synthèse élégante entre la logique linéaire, et la théorie des continuations. Nous pensons que cette synthèse est possible et qu'elle sera très fructueuse. Pour exemple, nous avons étendu au Chapitre 2 les aspects de cette belle et riche théorie qu'est la logique linéaire, au cadre relâché des continuations linéaires en introduisant la logique tensorielle. Typiquement, l'idée fondamentale que chaque monade de continuation est la version unaire d'une disjonction relâchée $[A_1 \parr \cdots \parr A_n]$, est le résultat de ce brassage entre les deux théories. Une des leçons de ce chapitre est qu'une hybridation réussie nécessite un enrichissement de la boîte à outils mathématiques de chaque champ – intégrant plus particulièrement les récents développements en algèbre 2-dimensionnelle. De cette façon, nous espérons qu'il émergera doucement une théorie monoïdale des effets, combinant la logique linéaire, la sémantique des jeux et l'algèbre des dimensions supérieures.

Chapitre 6

Certification d'un compilateur en Coq



Le Coq sur Paris, Chagall (1958)

Sommaire

Dans ce dernier chapitre, nous nous intéressons aux langages de bas niveau et à la compilation. Nous souhaitons avoir un compilateur dont on sait prouver la correction. En effet, on suppose généralement que les compilateurs ont une sémantique transparente : le code compilé « se comporte comme le dit la sémantique du programme source ». Malheureusement, les compilateurs – et plus particulièrement les compilateurs optimisés – sont des programmes complexes qu'on ne peut pas facilement déboguer par une simple phase de tests. La vérification d'un compilateur est un problème largement étudié comme l'atteste la bibliographie de Maulik Dave [Dav03]. Nous en mentionnons ici une partie. La correction d'un compilateur a été étudiée depuis au moins quarante ans [MP67] avec de premières formalisations notables dans un assistant de preuve à la fin des années 80 [You89]. Citons, comme exemple plus récent, le compilateur certifié en Coq par Xavier Leroy et son équipe pour un langage inspiré de C [Ler08]. Ces papiers traitent d'une notion de correction qui regarde si le terme compilé se réduit sur la même valeur que le terme du langage de haut niveau.

La correction complète du compilateur que ces travaux proposent est plus ambitieuse que l'approche par correction de typage que nous allons proposer. Mais de notre point de vue, ces projets manquent une chose essentielle car ils relient haut niveau et bas niveau sans passer par des contraintes de bas niveau indépendante du langage. Cela nous amène

à raisonner directement sur le langage non structuré de bas niveau, une idée provenant des travaux pionniers de Robert Floyd. Nous pouvons ainsi définir plus que la correction du compilateur, à savoir quel est le contrat bas niveau associé à un type haut niveau. Et ainsi nous pouvons savoir quels sont les obligations et les garanties pour avoir une interopérabilité avec le code d'autres langages.

Notre notion de correction est basée sur une propriété de sûreté déduite de la sémantique des types. Développer un système de types pour un langage bas niveau, et le préserver le typage lors de la compilation, est plus récent [MWCG99] et a particulièrement attiré l'attention dans le contexte des codes auto-certifiés [Nec97]. Mais il y a deux façon de formaliser la correction du typage : de manière syntaxique ou de manière sémantique. Nous travaillons ici avec l'approche sémantique ; c'est à dire que l'on donne un sens à chaque type indépendamment des règles qui ont été choisies pour typer les termes du langage. La sémantique d'un type est (en première approximation) donnée par un ensemble de valeurs satisfaisant une propriété ; on peut dès lors typer correctement le langage de différentes manières suivant le degré de précision que l'on cherche.

Notre approche se caractérise par l'utilisation d'une sémantique relationnelle qui permet d'interpréter un type à la fois par un ensemble de valeurs et par une égalité induite par le type sur cet ensemble de valeurs. Cette approche permet à la fois de prouver la correction du compilateur mais aussi de prouver la correction de certaines optimisations du compilateur de manière modulaire.

Comme le langage de haut niveau que nous voulons analyser contient les fonctions récursives, on ne va pas garantir la terminaison lors du typage. Il est donc insuffisant de s'intéresser uniquement à des fragments de code qui terminent. Dans ce cadre, la notion qui devient pertinente pour relier deux programmes est la *divergence*, c'est à dire le fait que l'exécution se poursuivent pour toujours. On dira que deux programmes p et p' sont reliés par la relation R aux points de programmes l et l', ce que nous noterons

$$\models p, p' \rhd l, l' : R^{\top}$$

si pour n'importe quel couple (s, s') d'états mémoire reliés par R, la divergence de p à partir de l'état s au point de programme l est équivalente à la divergence de p' à partir de l'état s' au point de programme l'. On dira dans ce cas que les deux programmes *équidivergent* pour la condition R. La propriété de correction de fragments de code assembleur reposera alors sur une version relationnelle de style « passage par continuation » (CPS) du triplet de Hoare [Hoa69]. Rappelons que le triplet de Hoare indique comment un fragment de code c modifie la mémoire en passant d'un état vérifiant le prédicat P_{pre} à un état vérifiant le P_{post}. On note usuellement ce triplet

$$\{P_{pre}\}\ c\ \{P_{post}\}.$$

La version relationnelle et CPS du triplet de Hoare est alors

$$\models p, p' \rhd l_{post}, l'_{post} : {R_{post}}^{\top} \ \Rightarrow \models p, p' \rhd l_{pre}, l'_{pre} : {R_{pre}}^{\top}.$$

Ce jugement indique que si les programmes p et p' équidivergent pour la condition R_{post} aux points l_{post} et l'_{post}, alors ils équidivergent pour la condition R_{pre} aux points l_{pre} et l'_{pre}.

Pour cette étude, nous avons choisi comme langage de haut niveau PCF_v et nous avons considéré un langage assembleur basique. Nous avons donné une sémantique du langage assembleur en nous inspirant de la théorie des continuations, de la réalisabilité et de la logique de séparation.

En utilisant nos résultats sur la négation tensorielle et les multicatégories de contrôle, nous aimerions avoir une sémantique uniforme tout au long de la compilation ; du langage de haut niveau jusqu'à l'assembleur. On sait déjà que la logique linéaire est un bon candidat pour interpréter les termes du langage de haut niveau. On sait aussi que les continuations linéaires donnent de bons signes vers l'interprétation d'un langage assembleur. Idéalement, on doit pouvoir interpréter ces deux langages dans le cadre commun de la logique tensorielle. Cette idée est d'ailleurs confortée par le développement récent d'une sémantique des jeux pour la logique de séparation (ou plus précisément de la logique BI) introduite par Guy McCusker et David Pym [MP07]. Nous n'avons pas encore réaliser ce saut conceptuel mais nous pouvons présenter ici les premiers résultats que nous avons obtenus.

Pour cette étude, nous avons choisi comme langage de haut niveau PCF_v (une version de PCF_v en appel par nom) et nous avons considéré un langage assembleur basique. Nous avons donné une sémantique du langage assembleur en nous inspirant de la théorie des continuations, de la réalisabilité et de la logique de séparation.

Nous avons ensuite écrit un compilateur en Coq du langage PCF_v vers notre assembleur, et nous avons prouvé, toujours en Coq, que le code compilé avait le comportement souhaité. La formalisation en Coq comporte un peu moins de 14000 lignes de code et repose pour l'essentiel sur la réécriture offerte par le module **Setoid**. Ce travail a été réalisé en collaboration avec Nick Benton du laboratoire Microsoft Research à Cambridge.

6.1 Définition des langages

6.1.1 Langage de haut niveau : PCF_v

On définit des types pour PCF_v un peu plus riches qu'à l'accoutumé, en particulier la présence de prédicats pour les types entiers et booléens. Ils vont nous permettre de déduire des corollaires plus intéressants du lemme de correction du compilateur. En effet, comme la mémoire va être modélisée par des registres contenant tous des entiers, pouvoir dire qu'un certain registre pointe vers un entier n'apporte rien. En revanche, si on peut dire que cet entier est pair ou égal à 22, l'information est pertinente. Voilà pourquoi nous avons enrichi le type des entiers avec un prédicat.

Définition 6.1 (type de PCF_v)
Les types de PCF_v, dont l'ensemble est noté **Type**, sont donnés par la définition inductive suivante

$$A, B ::= \mathrm{Nat}\ P_n \mid \mathrm{Bool}\ P_b \mid A \to B \mid A \times B \mid \exists n.A'n$$

où $P_n : \mathbb{N}\mathrm{at} \to \mathbb{B}\mathrm{ool}$ est un prédicat sur les entiers, $P_b : \mathbb{B}\mathrm{ool} \to \mathbb{B}\mathrm{ool}$ est un prédicat sur les booléens et A' est une fonction qui à un entier associe un type.

Les termes de PCF_v que nous allons utiliser sont quant à eux tout ce qu'il y a de plus standard.

Définition 6.2 (terme de PCF_v)
Les termes du langage sont ceux d'un λ-calcul avec point fixe et conditionnel :

$$\begin{array}{rcl} M, N, P & ::= & b \mid n \mid x \mid \lambda x.M \mid MN \mid \mathrm{Fix}\ x.M \mid \mathrm{Op}\ MN \mid M\ \mathrm{Gt}\ N \mid \\ & & \texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ P \mid \langle M, N\rangle \mid \pi_1(M) \mid \pi_2(M) \end{array}$$

où $b \in \text{Bool}$, $n \in \mathbb{N}\text{at}$ et $\text{Op} \in \{-, +, *\}$.

En présence du type simple Nat, les constructeurs Op et Gt sont typés par les deux règles d'inférence

$$[Op]\ \frac{\Gamma \vdash M : \text{Nat} \quad \Gamma \vdash N : \text{Nat}}{\Gamma \vdash \text{Op}\ MN : \text{Nat}} \qquad [Gt]\ \frac{\Gamma \vdash M : \text{Nat} \quad \Gamma \vdash N : \text{Nat}}{\Gamma \vdash \text{Op}\ MN : \text{Bool}}$$

Maintenant, nous avons étendu notre système de types avec un type entier muni d'un prédicat. Nous devons donc donner une interprétation prédicative $\{\text{Op}\}$ et $\{\text{Gt}\}$ des constructeurs Op et Gt pour pouvoir étendre les règles de typage ci-dessus aux types de la forme Nat P.

$$\begin{array}{rcccl} \{\text{Op}\}P_1P_2 & : & n & \mapsto & \exists x, \exists y, (P_1 x) \wedge (P_2 y) \wedge (\text{Op}\ xy = n) \\ \{\text{Gt}\}P_1P_2 & : & b & \mapsto & \exists x, \exists y, (P_1 x) \wedge (P_2 y) \wedge (x > y \Leftrightarrow b). \end{array}$$

Les règles $[Op]$ et $[Gt]$ de la Figure 6.1 montrent comment utiliser cette interprétation lors du typage. Plus généralement, les règles de typage sont données à la Figure 6.1. Le booléen vrai (resp. faux) est noté `v` (resp. `f`).

$$[Var]\ \frac{}{\Gamma, x : A \vdash x : A} \qquad [Bool]\ \frac{P\ b}{\Gamma \vdash b : \text{Bool}\ P} \qquad [Nat]\ \frac{P\ n}{\Gamma \vdash n : \text{Nat}\ P}$$

$$[Abs]\ \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta}\ (x \notin \Gamma) \qquad [App]\ \frac{\Gamma \vdash M : \alpha \to \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}$$

$$[Fix]\ \frac{\Gamma \vdash \lambda x.M : \alpha \to \beta}{\Gamma \vdash \text{Fix}\ \lambda x.M : \alpha \to \beta} \qquad [If]\ \frac{\Gamma \vdash M : \text{Bool} \quad \Gamma \vdash M_i : \alpha\ (i = 1, 2)}{\Gamma \vdash \texttt{if}\ M\ \texttt{then}\ M_1\ \texttt{else}\ M_2 : \alpha}$$

$$[Op]\ \frac{\Gamma \vdash M : \text{Nat}\ P_1 \quad \Gamma \vdash N : \text{Nat}\ P_2}{\Gamma \vdash \text{Op}\ MN : \text{Nat}(\{\text{Op}\}P_1P_2)} \qquad [Gt]\ \frac{\Gamma \vdash M : \text{Nat}\ P_1 \quad \Gamma \vdash N : \text{Nat}\ P_2}{\Gamma \vdash \text{Op}\ MN : \text{Bool}(\{\text{Gt}\}P_1P_2)}$$

$$[Pair]\ \frac{\Gamma \vdash M_i : \alpha_i\ (i = 1, 2)}{\Gamma \vdash \langle M_1, M_2 \rangle : \alpha_1 \times \alpha_2} \qquad [Proj]\ \frac{\Gamma \vdash M : \alpha_1 \times \alpha_2}{\Gamma \vdash \pi_i(M) : \alpha_i\ (i = 1, 2)}$$

$$[If]\ \frac{\Gamma \vdash M : \text{Bool}\ P \quad P\texttt{v} \Rightarrow \Gamma \vdash M_1 : A \quad P\texttt{f} \Rightarrow \Gamma \vdash M_2 : A}{\Gamma \vdash \texttt{if}\ M\ \texttt{then}\ M_1\ \texttt{else}\ M_2 : A} \qquad [Ex]\ \frac{\Gamma \vdash M : A'm}{\Gamma \vdash M : \exists n.A'n}$$

FIG. 6.1 – Règles de typage de PCF_v

Pour pouvoir appliquer une fonction f : Nat `True` → Nat `True` (`True` est le prédicat toujours vrai) à un entier n : Nat `Ev` (`Ev` est le prédicat des entiers pairs), nous avons besoin de sous-typage. En effet, cela nous donnera la possibilité de typer n par le type Nat `True` en utilisant la règle

$$\frac{\Gamma \vdash n : \text{Nat}\ \texttt{Ev} \qquad \text{Nat}\ \texttt{Ev} <: \text{Nat}\ \texttt{True}}{\Gamma \vdash n : \text{Nat}\ \texttt{True}}$$

On peut définir plus généralement cette relation de sous-typage.

Définition 6.3 (sous-typage)
On définit par induction la relation de sous-typage $<:$ de la manière suivante :

Nat.	$\forall P_1, P_2 : \mathbb{N}\text{at} \to \mathbb{B}\text{ool},$	$(\forall n, P_1 n \Rightarrow P_2 n) \Rightarrow (\text{Nat } P_1 <: \text{Nat } P_2)$
Bool.	$\forall P_1, P_2 : \mathbb{B}\text{ool} \to \mathbb{B}\text{ool},$	$(\forall b, P_1 b \Rightarrow P_2 b) \Rightarrow (\text{Bool } P_1 <: \text{Bool } P_2)$
Paire.	$\forall a, a', b, b' \in \mathbf{Type},$	$(a <: a') \Rightarrow (b <: b') \Rightarrow (a \times b <: a' \times b')$
Flèche.	$\forall a, a', b, b' \in \mathbf{Type},$	$(a' <: a) \Rightarrow (b <: b') \Rightarrow (a \to b <: a' \to b')$

On ajoute alors une règle de sous-typage à celles déjà décrite à la Figure 6.1

$$[Coerce]\ \frac{\Gamma \vdash M : \alpha \qquad \alpha <: \beta}{\Gamma \vdash M : \beta}$$

Nous ne donnons pas ici de sémantique à grand pas pour le langage de haut niveau car elle sera déduite de la sémantique du code assembleur obtenu par compilation.

6.1.2 Langage de bas niveau : assembleur

Nous présentons maintenant le langage assembleur vers lequel nous compilons les termes de PCF_v. Les définitions seront données dans une syntaxe proche de Coq. Il n'est pas nécessaire d'être un spécialiste de Coq pour comprendre les définitions qui suivent. Lorsque les connaissances requises pour comprendre les définitions nous apparaîtront trop importantes, nous retournerons vers une syntaxe mathématique.

Définition 6.4 (mémoire machine)
Un *état de la mémoire* est une fonction s des entiers dans les entiers qui à un entier n – une adresse mémoire – associe un entier $s(n)$ – la valeur stockée à l'adresse mémoire n.

```
Definition state := ℕat → ℕat.
```

On peut mettre à jour une adresse mémoire d'un état s à l'aide la fonction `update`

```
Definition update (s:state) (n:ℕat) (v:ℕat) : state :=
fun m ⇒ if n = m then v else s m.
```

Notons que nous utilisons ici un modèle où la mémoire est infinie. Pour définir notre ensemble d'instructions assembleur, nous devons définir des fonctions de base qui vont permettre d'accéder à ces cases mémoires. Nous appelons ces fonctions des *fonctions d'adressage*. Nous distinguons les fonctions d'adressage `src` qui décrivent comment accéder à la source d'une instruction et `dest` qui décrivent comment accéder au but d'une instruction.

```
Inductive src : Set :=
  | s_cst : ℕat → src
  | s_imm : ℕat → src
  | s_ind : ℕat → src
  | s_indo : ℕat → ℕat → src.

Inductive dest : Set :=
  | d_imm : ℕat → dest
```

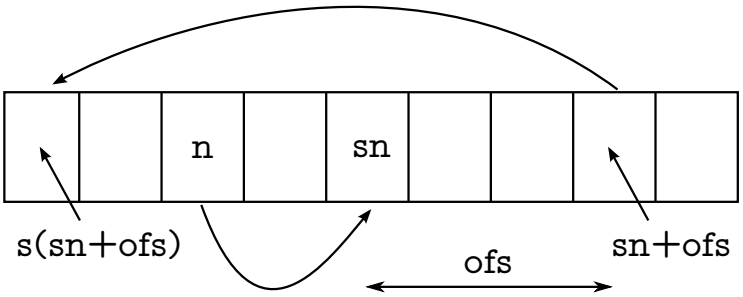


FIG. 6.2 – Sémantique de la fonction `s_indo`

```
  | d_ind : ℕat → dest
  | d_indo : ℕat → ℕat → dest.
```

Ces ensembles de fonctions d'adressage vont permettre d'accéder à l'espace mémoire de la manière suivante

```
Definition sem_src (sr:src) (s:state) :=
match sr with
 | s_cst n ⇒ n
 | s_imm n ⇒ s n
 | s_ind n ⇒ s (s n)
 | s_indo ofs n ⇒ s (s n + ofs)
end.

Definition sem_Dest (de:Dest) (s:state) : ℕat :=
match de with
 | d_imm n ⇒ n
 | d_ind n ⇒ s n
 | d_indo ofs n ⇒ s n + ofs
end.
```

Commentons par exemple la fonction la plus riche `s_indo`. Cette fonction prend en argument deux entiers `n` et `ofs` qui correspondent respectivement à une adresse mémoire et à un décalage. Elle renvoie la valeur de la case `s(s n + ofs)` comme l'indique la Figure 6.2 Ces fonctions d'adressage permettent de définir les instructions qui prennent toutes en argument des descriptions d'emplacement mémoire.

```
Inductive instruction : Set :=
  | i_halt : instruction
  | i_move : dest → src → instruction
  | i_op : (ℕat → ℕat → ℕat) → dest → src → src → instruction
  | i_branch : src → instruction
  | i_brz : src → src → instruction
  | i_brnz : src → src → instruction
  | i_eqtst : src → src → src → instruction.
```

La sémantique des instructions est donnée à partir de la sémantique de fonctions d'adressage. La fonction `sem_instr` prend en argument une instruction `ins`, un état de la mémoire `s` et un point de programme `pc`.

```
Definition sem_instr (ins:instruction) (s:state) (pc:ℕat) :
  option (state * ℕat) :=
match ins with
 | i_halt ⇒ None
 | i_move add val ⇒ Some (update s (sem_Dest add s) (sem_src val s), pc+1)
 | i_op op add val1 val2 ⇒ Some (
     update s (sem_Dest add s)(op (sem_src val1 s) (sem_src val2 s)), pc+1)
 | i_branch add ⇒ Some (s, sem_src add s)
 | i_brz val add ⇒ Some (s, match sem_src val s with
       0 ⇒ sem_src add s |
     S _ ⇒ pc+1 end)
 | i_brnz val add ⇒ Some (s, match sem_src val s with
       0 ⇒ pc+1 |
     S _ ⇒ sem_src add s end)
 | i_eqtst val1 val2 add ⇒ Some (s,
     if (sem_src val1 s) = (sem_src val2 s) then sem_src add s else pc+1)
end.
```

où `S` dénote la fonction successeur sur ℕat. On est maintenant en mesure de définir l'ensemble des programmes écrits en assembleur. Voici comment fonctionnent les instructions

- `i_halt` est l'instruction d'arrêt ;
- `i_move add val` : place la valeur correspondant à `val` à l'adresse correspondant à `add` et incrémente le point de programme ;
- `i_op op add val1 val2` : applique l'opération `op` à la valeur correspondant à `val1` et la valeur correspondant à `val2`, place la valeur obtenue à l'adresse correspondant à `add` et incrémente le point de programme ;
- `i_branch add` : place le pointeur de code à l'adresse correspondant à `add` ;
- `i_brz val add` : place le point de programme à l'adresse correspondant à `add`, si la valeur correspondant à `val` est 0 ; sinon, incrémente le point de programme ;
- `i_brnz val add` : place le point de programme à l'adresse correspondant à `add`, si la valeur correspondant à `val` n'est pas 0 ; sinon, incrémente le point de programme ;
- `i_eqtst val1 val2 add` : place le point de programme à l'adresse correspondant à `add`, si la valeur correspondant à `val1` est égale à la valeur correspondant à `val2` ; sinon, incrémente le point de programme.

Ces instructions permettent de définir un programme assembleur et une configuration d'un programme assembleur.

Définition 6.5 (programme assembleur et configuration)
Un *programme assembleur* est une fonction qui à chaque entier, vu comme une adresse mémoire, associe une instruction. On définit l'ensemble des programmes par le terme Coq

```
Definition program : Set := ℕat → instruction.
```

Une *configuration* d'un programme assembleur est donnée par un triplet $\langle s, p, l\rangle$ où s est un état, p est un programme et l est un point de programme.

On en déduit alors une sémantique déterministe évidente $\langle s, p, l\rangle \to \langle s', p', l'\rangle$ entre deux configurations, dont découle une notion de terminaison.

Définition 6.6 (terminaison)
La terminaison d'une configuration est décrite par un calcul de point fixe à partir de la fonction `sem_instr`.

```
Fixpoint kstepterm (k:ℕat) (p:program) (s:state) (l:ℕat) struct k :
Prop :=
match k with
  | 0 ⇒ False
  | (S j) ⇒
    match sem_instr (p l) s l with
      | None ⇒ True
      | Some (s', l') ⇒ kstepterm j p s' l'
    end
end.

Definition terminates p s l := ∃ k, kstepterm k p s l.
```

La fonction `kstepterm` renvoie vrai lorsque la configuration $\langle s, p, l\rangle$ peut se réduire sur `i_halt` en moins de k étapes. La fonction `terminates` indique que la configuration $\langle s, p, l\rangle$ se réduit sur `i_halt`.

6.2 Une sémantique relationnelle

La raison principale qui nous pousse à utiliser une sémantique relationnelle est que celle-ci permet d'avoir une notion d'équivalence observationnelle entre deux configurations. Ainsi, on peut montrer que deux fragments de programme qui ne sont pas rigoureusement identiques se comportent de la même manière – au sens où ils divergent en même temps (équidivergent). Cette finesse de description n'est pas présente avec un sémantique prédicative. Le lecteur trouvera plus de détail sur ces idées dans le travail de Nick Benton [Ben06].

Pour décrire la sémantique d'un fragment de code assembleur, il est impératif de pouvoir décrire l'état de la mémoire de manière précise et structurée. C'est là qu'intervient notre volonté de décrire le langage bas niveau de la même manière que le langage haut niveau. Idéalement, nous aimerions que la structure décrivant le fonctionnement du code assembleur soit munie d'une négation tensorielle. Nous n'avons pas atteint une description si uniforme. Néanmoins, les ingrédients de notre modèle sémantique reprennent ceux déjà présents dans cette thèse :

- une catégorie d'ordre **stateRel** dont les objets sont relations sur les états de la mémoire, la relation d'ordre étant induite par l'implication logique. Par exemple, le fait que s et s' soient dans la relation
$$(0 \mapsto \{25, 34\})\ s\ s'$$
indique $s(0) = 25$ et que $s'(0) = 34$;
- une structure monoïdale symétrique induite par un connecteur séparant venant de la logique de séparation ;
- une adjonction avec une catégorie « duale » **natRel** des relations sur les points de programme, l'adjonction correspondant à une notion d'orthogonalité provenant des idées introduites en réalisabilité ;

– un produit cartésien dans **stateRel** (mais n'induisant pas de modalité de ressources).

Tous ces ingrédients permettent de décrire avec précision la mémoire requise par un programme (à un point du programme donné) pour que celui-ci s'exécute sans heurt. Pour que cette description soit assez riche, il faut ajouter la possibilité d'utiliser le programme dans la description de la relation. Ainsi, on peut non seulement exprimer qu'une certaine adresse contient une certaine valeur, mais aussi qu'une adresse contient un point de programme qui vérifie certaines propriétés. Enfin, comme nous voulons interpréter les points fixes, nous introduisons un notion d'indice « de sûreté » pour chaque relation sur les états [Ahm04], qui permet de réaliser la loi de Löb [AMRV07]. Intuitivement, cet indice k indique que le programme est sûr pour cette relation mais uniquement pour k étapes de réduction.

6.2.1 La catégorie stateRel

Définition 6.7 (relation sur les états de la mémoire)
Une relation dans **stateRel** est une relation qui porte sur deux états partiels de la mémoire, deux programmes et un indice vérifiant une condition de monotonie.

```
Record stateRel : Type := mkStateRel {
  R :> (ℕat ↪ ℕat) → (ℕat ↪ ℕat) → program → program → ℕat → Prop;
  stateRel_cond : ∀ s1 s2 s1' s2' p1 p2 k1 k2,
                  (R s1 s2 p1 p2 k1 ∧ s1 ⊂ s1' ∧ s2 ⊂ s2' ∧ k2 ≤ k1) →
                  R s1' s2' p1 p2 k2
}.
```

où $A \hookrightarrow B$ dénote l'ensemble des fonctions partielles de A vers B et $s \subset s'$ l'inclusion de fonctions partielles.

Une relation R est plus petite qu'une relation R', ce que nous notons $R \preceq R'$, lorsque

```
Definition stateRelLeq (R R' : stateRel) :=
  ∀ s s' p p' k, R s s' p p' k → R' s s' p p' k.
```

La condition de monotonie indique que si on augmente l'information sur l'état de la mémoire ou si on diminue l'indice de sûreté ; on reste dans la relation. Remarquons que les états présents dans la définition de la relation sont *partiels*. Cela nous permettant d'avoir une structure de *tenseur séparant* provenant directement de la logique de séparation [Rey02]. Nous adoptons maintenant une description basée sur le langage mathématique plutôt que sur le langage de Coq afin de faciliter la lecture.

Définition 6.8 (tenseur séparant)
Le tenseur séparant $R \otimes R'$ de deux relations de **stateRel** est la relation définie par

$$(R \otimes R')\, s\, s'\, p\, p'\, k \overset{\text{def}}{\iff} \exists s_1\ s_2\ s_1'\ s_2',\ s = s_1 \# s_2 \wedge s' = s_1' \# s_2' \wedge R\, s_1\, s_1'\, p\, p'\, k \wedge R'\, s_2\, s_2'\, p\, p'\, k$$

où $s = s_1 \# s_2$ indique que les supports de s_1 et s_2 sont disjoints et que l'union de s_1 et s_2 est égale à s.

Le tenseur séparant permet de combiner des blocs de base que nous décrivons ci-dessous de manière non exhaustive.

$$\begin{aligned}
\left(\{n_1, n_2\} \mapsto R^{\top}\right)\ s\ s'\ p\ p'\ k &\overset{\text{def}}{\iff} \exists l, l'\ s(n_1) = l \wedge s'(n_2) = l' \wedge R\ l\ l'\ p\ p'\ k \\
(\{n_1, n_2\} \mapsto \{m_1, m_2\})\ s\ s'\ p\ p'\ k &\overset{\text{def}}{\iff} s(n_1) = m_1 \wedge s'(n_2) = m_2 \\
n \mapsto \{m_1, m_2\} &\overset{\text{def}}{=} \{n_1, n_2\} \mapsto \{m_1, m_2\} \\
\{n_1, n_2\} \mapsto - &\overset{\text{def}}{=} \{n_1, n_2\} \mapsto \texttt{True}^{\top} \\
\texttt{Block}\ m\ n &\overset{\text{def}}{=} \bigotimes_{0 \le i \le n-1} (m + i \mapsto -) \\
(\texttt{Topfrom}\ m\ m')\ s\ s'\ p\ p'\ k &\overset{\text{def}}{\iff} \forall n\ n', n \ge m \Rightarrow n' \ge m' \Rightarrow (\{n, n'\} \mapsto -)
\end{aligned}$$

Le tenseur séparant est crucial lorsque l'on veut faire une mise à jour de la mémoire tout en garantissant que cette mise à jour n'invalide pas d'autres parties de la relation. Pourtant, lorsqu'on veut décrire les propriétés vérifiées par une clôture de fonction, on ne peut plus garantir la séparabilité des différentes relations qui la composent. Il nous faut donc aussi un produit tensoriel non séparant qui s'avère être cartésien.

Définition 6.9 (produit cartésien)
Le produit cartésien $R \times R'$ de deux relations de **stateRel** est la relation définie par

$$(R \times R')\ s\ s'\ p\ p'\ k \quad \overset{\text{def}}{\iff} \quad R\ s\ s'\ p\ p'\ k \wedge R'\ s\ s'\ p\ p'\ k.$$

Nous avons prouvé en Coq la proposition suivante.

Proposition 6.10 La catégorie **stateRel** est une catégorie d'ordres monoïdale symétrique pour le tenseur séparant $\otimes$. Elle est munie d'un produit cartésien $\times$ qui vérifie la *loi de distributivité*

$$(R \times S) \otimes T \preceq (R \otimes T) \times (S \otimes T).$$

6.2.2 Une adjonction avec la catégorie natRel

Afin de pouvoir exprimer qu'un programme vérifie une certaine propriété à un point de programme donné, il faut aussi avoir une notion de relation sur les points de programme et non plus sur les états.

Définition 6.11
Une relation dans **natRel** est une relation qui porte sur deux points de programme, deux programmes et un indice vérifiant une condition de monotonie.

```
Record natRel : Type := mkNatRel {
  R :> ℕat → ℕat → program → program → ℕat → Prop;
  natRel_cond : ∀ l l' p p' k1 k2,
                (R l l' p p' k1 ∧ k2 ≤ k1) → R l l' p p' k2
}.
```

Comme pour **stateRel**, une relation R est plus petite qu'une relation R' lorsque

```
Definition stateRelLeq (R R' : natRel) :=
  ∀ l l' p p' k, R l l' p p' k → R' l l' p p' k.
```

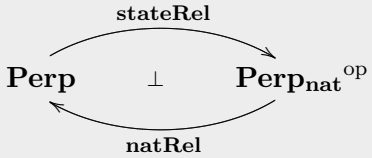
Pour relier les relations sur les états de la mémoire et les relations sur les points de programme du code assembleur, nous allons établir une adjonction entre **stateRel** et **natRel**.

Définition 6.12 (adjonction Perp ⊣ $\mathbf{Perp_{nat}}$)
Les deux foncteurs **Perp** et $\mathbf{Perp_{nat}}$ définis par

```
Definition Perp (S:stateRel) := fun p p' l l' k ⇒ ∀ s s' j,
  (j ≤ k ∧ S s s' p p' j) →
  ((kstepterm j p s l → terminates p' s' l') ∧
   (kstepterm j p' s' l' → terminates p s l)).

Definition Perp_nat (L:natRel) := fun s s' p p' k ⇒ ∀ l l' j,
  (j ≤ k ∧ L p p' l l' j) →
  ((kstepterm j p s l → terminates p' s' l') ∧
   (kstepterm j p' s' l' → terminates p s l)).
```

forment une adjonction (plus simplement une correspondance de Galois)



Quand nous utiliserons des notations mathématiques, nous noterons $R^\top$ l'image de la relation R par **Perp**. Ce foncteur, qui peut être vu comme une notion *d'orthogonalité*, nous permet de définir la validité d'une relation d'états à deux points de programme de deux programmes donnés. C'est une définition de type « passage par continuation » (CPS), au sens où l'on garantit que la continuation du programme au point l se déroulera sans encombre.

Définition 6.13 (jugement)
On dit que deux programmes p et p' *équidivergent* selon la relation $R \in$ **stateRel** aux points l et l' s'ils sont liés par la relation $R^\top$ pour tout indice k. On note alors

$$\models p, p' \rhd l, l' : R^\top \quad \stackrel{\text{def}}{=} \quad \forall k,\ R^\top\, l\, l'\, p\, p'\, k.$$

Exemple 6.14
Deux programmes p et p' contenant le fragment de code suivant

$$\left[\begin{array}{l}\texttt{i\_op plus (d\_imm 5) (s\_imm 5) (s\_cst 1)}\\ \texttt{i\_move (d\_ind 5) (s\_imm 0)}\end{array}\right]$$

aux points de programme respectifs $l, l+1$ et $l', l'+1$ valident la propriété suivante

$$\begin{array}{lll}\forall R, & & \models p, p' \rhd l+2, l'+2 : (0 \mapsto \{28,15\}) \otimes (5 \mapsto 12) \otimes (13 \mapsto -) \otimes R^\top \\ & \Longrightarrow & \models p, p' \rhd l, l' : (0 \mapsto \{28,15\}) \otimes (5 \mapsto 13) \otimes (13 \mapsto \{28,15\}) \otimes R^\top.\end{array}$$

Cette proposition décrit le fait que ce fragment de code incrémente le registre 5 de 1 puis place à l'adresse pointé par le registre 5 – en l'occurrence ici 13 – la valeur stockée dans le registre 0. Pour le programme p (resp. p'), on a pour hypothèse que le registre 0 contient la valeur 28 (resp. 15).

6.2.3 Les quantificateurs internalisés

Afin de spécifier de manière minimale l'état de la mémoire dont nous avons besoin pour que le code compilé s'exécute convenablement, nous introduisons les quantificateurs existentiel et universel dans les constructeurs de relations.

Définition 6.15 (quantificateurs)
Pour définir les quantificateurs sur les relations, nous allons directement nous servir des quantificateurs déjà présents dans Coq. Soit X un type et $h : X \to \mathbf{stateRel}$, on définit

$$\begin{aligned}(\exists h)\, s\, s'\, p\, p'\, k &\overset{\text{def}}{\iff} \exists x, hx\, s\, s'\, p\, p'\, k\\ (\forall h)\, s\, s'\, p\, p'\, k &\overset{\text{def}}{\iff} \forall x, hx\, s\, s'\, p\, p'\, k\end{aligned}$$

6.2.4 Réaliser la loi de Löb

Comme nous l'avons dit plus haut, nous avons introduit un indice dans la définition des relations sur les états afin d'appréhender les points fixes et plus particulièrement de réaliser la loi de Löb (parfois aussi appelée axiome GL pour Gödel-Lob). Cette loi décrite par la règle d'inférence

$$\frac{\Diamond\alpha \;\vdash\; \alpha}{\vdash\; \alpha}$$

se lit *"si α est valide dès que α est valide dans le futur, alors α est toujours valide"*. Cette loi fournit un schéma d'induction clair permettant de démontrer la correction de la compilation des points fixes. Pourtant, l'importance de cet indice n'est pas encore très claire à ce stade de la lecture. Cet indice est en fait présent pour pouvoir définir la modalité $\Diamond$ qui signifie « sera valable dans le futur » (la notation vient de l'opérateur « un jour » de la logique temporelle). Il nous permettra d'utiliser une variante de la loi de Löb lorsqu'on prouvera le fragment de code compilé pour le constructeur Fix.

Définition 6.16 (modalité $\Diamond$)
La modalité $\Diamond$ (prononcée « plus tard ») est définie sur les relations d'états par

$$\Diamond R\, s\, s'\, p\, p'\, k \quad \overset{\text{def}}{\iff} \quad \exists j < k,\ R\, s\, s'\, p\, p'\, j$$

Cette modalité nous permet de dériver un schéma de réduction. Il stipule que si une continuation satisfait `Post` pour k étapes aux points de programme l_1 et l_1' ; et si elle est satisfaite dès que *ptr* pointe vers un point de programme qui satisfait `Post`, alors la continuation satisfait $ptr \mapsto \{l_1, l_1'\}$ pour $(k+1)$ étapes.

Lemme 6.17 (schéma de réduction) Pour tous programmes p et p', tout registre ptr, tous points de programme l_0, l'_0, l_1 et l'_1, toutes relations $Post$ et P et pour tout indice k, on a

$$\frac{\mathtt{Post}^\top\, l_1\, l'_1\, p\, p'\, k \qquad \models p, p' \rhd l_0, l'_0 : ((ptr \mapsto \Diamond \mathtt{Post}^\top) \otimes P)^\top}{((ptr \mapsto l_1, l'_1) \otimes P)^\top\, l_0\, l'_0\, p\, p'\, (k+1)}$$

Ce schéma de réduction permet de montrer une « version CPS » de la loi de Löb.

Lemme 6.18 (loi de Löb) Pour tous programmes p et p', tout registre ptr, tous points de programme l et l', toutes relations P et pour tout indice k, on a

$$\frac{\models p, p' \rhd l, l' : ((ptr \mapsto \Diamond((ptr \mapsto l, l') \otimes P)^\top) \otimes P)^\top}{\models p, p' \rhd l, l' : ((ptr \mapsto l, l') \otimes P)^\top}$$

6.3 Le compilateur

Cette section a pour but de décrire les principes de base de notre compilateur afin de rendre compréhensible la description de l'espace mémoire qui est au cœur du lemme de correction. Le lecteur souhaitant plus de détails sur la théorie de la compilation est invité à se rapporter à la monographie d'Andrew Appel [App98] qui traite de la compilation des langages fonctionnels et plus particulièrement de ML.

6.3.1 Description des registres et de la pile

Nous donnons ici la nomenclature attachée à la gestion de registres.

- le registre 0 que nous noterons **ret** , correspond au registre de retour des fonctions ;
- le registre 1 que nous noterons **arg** , correspond à l'emplacement de l'argument avant un appel à l'allocateur ;
- les registres 2, 3 et 4 correspondent à un espace de travail. Nous noterons **wk** le registre 2 ;
- le registre 5, que nous noterons **sp** , correspond au pointeur de pile ;
- le registre 6, que nous noterons **env** , correspond au pointeur d'environnement ;
- les registres suivants sont alloués dynamiquement par l'allocateur pour stocker la pile, l'environnement, etc.

La pile courante est décrite à partir du pointeur de pile **sp** qui pointe vers un certain **ptr** qui lui-même pointe vers l'élément contenu au sommet de la pile. L'élément qui le précède dans la pile (s'il existe) est pointé par $\mathbf{ptr} - \mathbf{1}$. La Figure 6.3 schématise ce mécanisme.

La structure de donnée liée à l'environnement est différente car on ne peut pas garantir que les éléments de l'environnement se trouvent les uns après les autres. Nous allons utiliser les listes chaînées pour remédier à ce problème. La Figure 6.4 décrit plus en détail cette structure.

Le pointeur d'environnement pointe aussi indirectement vers quelques autres valeurs particulières telles que l'adresse de l'ancienne pile, l'adresse de l'ancien environnement et l'adresse de retour. Ainsi, lorsque l'on calcule l'application d'une fonction, on la place dans l'environnement souhaité avec une pile vide, puis on restaure l'ancien environnement et l'ancienne pile une fois le calcul effectué.

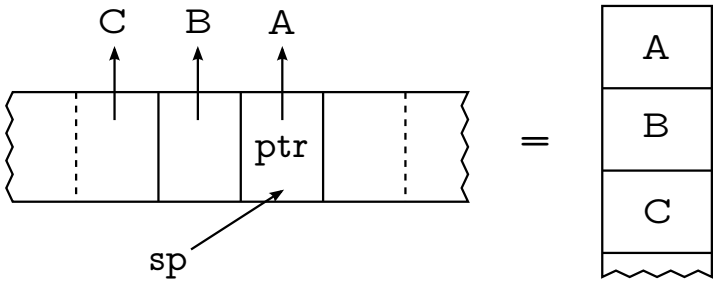


FIG. 6.3 – Description de la pile : le pointeur **sp** pointe vers **ptr** qui pointe vers un élément de type A. Les pointeurs précédents pointent vers les éléments précédents dans la pile.

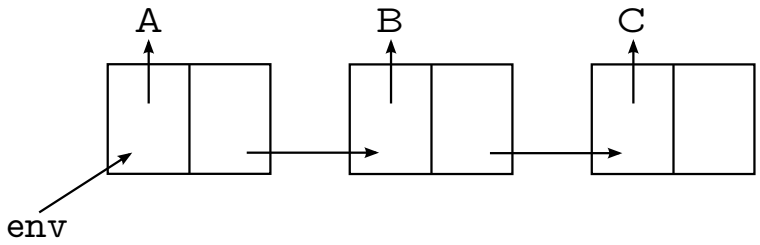


FIG. 6.4 – Description de l'environnement comme une liste chaînée : le pointeur **env** pointe vers un pointeur qui pointe vers le premier élément A de l'environnement, le pointeur suivant pointe vers un pointeur qui pointe vers le deuxième élément B de l'environnement, et ainsi de suite.

6.3.2 Compilation du code

Nous allons maintenant décrire quelques étapes de la compilation. L'algorithme particulier du compilateur n'a pas beaucoup d'importance ici car la preuve de correction est assez indépendante du compilateur. Aussi, si on améliore le compilateur, on peut réutiliser une preuve très similaire.

Le compilateur est bien évidemment défini par induction sur les termes de PCF_v. Nous nous contenterons donc d'une courte revue des principaux constructeurs du langage. Nous ne décrivons pas le code associé à l'allocateur car celui-ci n'a pas d'incidence sur le code compilé. La seule hypothèse dont on a besoin est qu'il alloue de l'espace mémoire libre à chaque fois que l'on fait appel à lui selon les conventions d'appel définies par la spécification de l'allocateur.

Nous définissons dans un premier temps la fonction `compile_rec` qui suppose que l'espace pour l'environnement et la pile a déjà été alloué. Cela implique que le code correspondant à un entier n'a pas besoin de faire appel à l'allocateur car il peut directement placer l'entier sur la pile. En revanche, pour stocker une paire par exemple, il faut faire appel à l'allocateur car la pile va pointer vers deux cases qui pointent chacune vers un des éléments de la paire.

On définira ensuite la fonction `compile` qui se chargera d'allouer de la mémoire pour l'environnement et la pile.

Compilation des entiers. La compilation d'un entier n se déroule en deux étapes. On incrémente ce vers quoi pointe le pointeur de pile **sp** de 1 et on fait pointer ce nouveau pointeur vers n. Ceci est réalisé par les deux instructions assembleur suivantes

```
 [sp]  ← [sp]+1
[[sp]] ← n
```

Compilation des variables. La compilation d'une variable de profondeur n dans l'environnement se déroule en trois étapes. On incrémente ce vers quoi pointe le pointeur de pile **sp** de 1, on parcours la liste chaînée décrivant l'environnement jusqu'à trouver la variable qui nous intéresse et on fait pointer le nouveau pointeur de pile vers cette variable. La recherche dans l'environnement est facilitée par le fait que les variables sont toutes « De Bruijnifiées » au sens où elles ne sont pas décrites par leur nom mais par leur rang dans l'environnement. Les $n+3$ instructions assembleur suivantes correspondent au code compilé pour une variable

```
 [sp]  ← [sp]+1
 [wk]  ← [env]
 [wk]  ← [[wk]+1]
 ...  n times  ...
 [wk]  ← [[wk]+1]
[[sp]] ← [[wk]]
```

Compilation des paires. La compilation d'une paire nécessite un appel à l'allocateur. On place dans le registre des arguments **arg** l'espace mémoire nécessaire (ici 2), on place dans le registre de retour **ret** le point de programme `label+4` où l'allocateur doit « sauter » et on saute vers le point de programme de l'allocateur. Ensuite, à l'instruction d'étiquette `label+4`, on peut supposer que le registre **ret** contient un emplacement où il y a deux cases libres. On décrémente le pointeur de pile, le fait pointer vers cet emplacement libre et on fait pointer les deux cases libres vers les deux éléments de la paire. Ceci est réalisé par les sept instructions assembleur suivantes

```
lab:       [sp]  ← [sp]-1
          [arg]  ← 2
          [ret]  ← lab+4
           jmp alloc
lab+4:  [[ret]]  ← [[sp]]      // store snd
       [[ret]+1] ← [[sp]+1]    // store fst
        [[sp]]   ← [ret]       // push pair
```

Compilation des fonctions. La compilation d'une fonction se déroule en trois phases. La première est la phase de préparation. Elle consiste à allouer un espace mémoire pour la pile nécessaire à l'exécution du corps de la fonction et à mettre en place l'environnement requis par le corps de la fonction. Ce nouvel environnement contient en particulier l'adresse de retour et l'adresse de l'ancien environnement. Tout ceci est réalisé par les instructions suivantes

```
lab:   [arg] ← stack_size + 5 allocate frame
       [ret] ← label + 3
       jmp alloc
lab+3: [wk] ← [[sp]]         take closure ptr
```

```
[[ret]+1] ← [[wk]+1]  store env ptr
[sp] ← [sp]-1         focus on argument
[[ret]] ← [[sp]]      store argument in env
[[ret]+2] ← [env]     store calling env
[[ret]+3] ← [sp]      store calling sp
[env] ← [ret]         switch to new env
[sp] ← [ret] + 4      switch to new stack
```

La deuxième phase est obtenue en concaténant les instructions assembleur correspondant au corps de la fonction. Enfin, la troisième phase consiste à restaurer l'ancienne pile et l'ancien environnement, à placer la valeur calculée par la fonction en tête de pile puis à désallouer l'espace qui avait été alloué pour la nouvelle pile. Tout ceci est réalisé par les instructions suivantes

```
lab:     [wk]  ← [[sp]]          save return value
         [sp]  ← [[env]+3]       restore old sp
        [[sp]] ← [wk]            push return value
         [wk]  ← [env] +2        base to free
         [env] ← [[env]+2]       restore old frame
         [arg] ← stack_size+3    size to free
         [ret] ← label + 8
         jmp dealloc
lab+8:   jmp [[env]+4]           return to caller
```

Compilation des applications. La compilation d'une application consiste simplement à stocker l'adresse de retour à l'emplacement dédié puis à sauter à l'adresse pointée par le pointeur de fonction. Ceci est réalisé par les trois instructions assembleur suivantes

```
lab:   [[env]+4] ← lab+3
       [wk] ← [[sp]]
       jmp [wk]
lab+3: ...
```

Compilation des points fixes. La compilation du point fixe est simplifiée par la règle de typage $[Fix]$ décrite à la Figure 6.1 qui force le sous-terme à avoir un lambda en tête. Ainsi, il nous faut accroître l'environnement par un « trou » qui servira à stocker le pointeur associé à la fonction sous-jacente. Ceci est réalisé par les instructions suivantes

```
lab:   [arg] ← 2
       [ret] ← lab+3
       jmp alloc
lab+3: [[ret]+1] ← [env]
       [env] ← [ret]
```

Ensuite, on place le code de la fonction, on fait pointer le trou dans l'environnement vers le pointeur de cette fonction et on restaure l'ancien environnement par les instructions

```
[[env]] ← [[sp]]    // knot
 [env]  ← [[env]+1] // restore
```

Définition 6.19 (compilation locale)
indexcompilation locale Tous ces fragments de code permettent de définir par récurrence sur le terme PCF_v la fonction `compile_rec` qui construit le fragment de code assembleur local correspondant au terme M à partir du point de programme l. On entend par local le fait que ce fragment de code s'inscrit dans un programme plus grand qui alloue la mémoire dont il a besoin.

Lorsque le terme est clos, on va construire un programme assembleur autonome qui place son résultat dans le registre **ret**.

Compilation générale. La compilation d'un terme clos se déroule en trois étapes. D'abord, on place un code d'initialisation qui initialise l'allocateur, alloue de l'espace mémoire pour l'environnement (vide) et la pile requise par le terme. Ensuite, on place le code généré par la fonction `compile_rec`. Enfin, on récupère la valeur en tête de pile pour la placer dans le registre de réponse **ret**.

```
[ret] ← [[sp]]
```

On obtient alors le code assembleur correspondant au terme PCF_v.

Définition 6.20 (programme compilé)
On note `compile` la fonction qui génère le programme assembleur correspondant au terme M, placé au point de programme l. Ce programme est constitué du code compilé décrit ci-dessus plus du code spécifique à l'allocateur. Ce programme place son résultat dans le registre **ret**.

Remarque 6.21
Nous allons maintenant montrer que le code compilé par le compilateur vérifie une spécification indépendante de l'endroit où il est compilé. Pour cela, nous allons utiliser une spécification relationnelle reliant le même code ; compilé à deux endroits différents. Comme ce code fait appel à un alloueur qui répond différemment suivant les conditions dans lequel on l'appelle, rien ne garantit qu'il alloue le même espace mémoire dans les deux versions du code. Ainsi, tous les pointeurs provenant d'une allocation mémoire doivent être distingués suivant qu'ils viennent de la première ou de la deuxième version du code. Pour ne pas trop alourdir les notations, nous omettrons cette duplication pour les spécifications qui suivent. Cette duplication est complètement orthogonale aux phénomènes de compilation que nous voulons relater et une représentation explicite alourdirait considérablement le discours.

6.3.3 Spécification de l'allocateur

Comme nous l'avons dit plus haut, l'algorithme précis de l'allocateur n'a pas d'importance. Tout ce qui compte est la spécification que celui-ci vérifie à l'initialisation, l'allocation et la déallocation. Avant de la présenter en détail, nous devons introduire une notion plus forte de jugement que celle qui nous intéressait pour le code compilé. En effet,

notre notion d'équivalence entre fragments de code est basée sur l'équidivergence ; c'est à dire sur des jugements partiels. Comme ces fragments de code font pour la plupart appel à l'allocateur, nous ne pouvons permettre de divergence pour l'allocateur car elle rendrait impossible l'établissement d'une quelconque équidivergence pour les fragments de code. Ainsi, la notion qui devient pertinente pour l'allocateur est plutôt un jugement total qui garantit que le programme se réduit toujours et vérifie une propriété.

Définition 6.22 (jugement total)
On dit que deux programmes p et p' satisfont `total_judgment` $R_{pre}\, R_{post}\, p\, p'\, l_0\, l'_0$ aux points de programme l_0 et l'_0 pour deux relations d'états R_{pre} et R_{post} lorsque

$$\begin{aligned}\texttt{total\_judgment}\, R_{pre}\, R_{post}\, p\, p'\, l_0\, l'_0 \quad \overset{\text{def}}{\iff} \quad & \forall l_1\, l'_1\, k\, s_0\, s'_0, (R_{pre} l_1 l'_1)\, s_0\, s'_0\, p\, p'\, k \Rightarrow \\ & \exists j_1\, j'_1\, s_1\, s'_1, \quad \texttt{kstepreduce}\ j_1\ p\ s_0\ s_1\ l_0\ l_1 \\ & \qquad\qquad \wedge \quad \texttt{kstepreduce}\ j'_1\ p'\ s'_0\ s'_1\ l'_0\ l'_1 \\ & \qquad\qquad \wedge \quad R_{post}\, s_1\, s'_1\, p\, p'\, k\end{aligned}$$

où `kstepreduce` $j_1\ p\ s_0\ s_1\ l_0\ l_1$ indique que le programme p se réduit en j_1 étapes de la configuration $\langle s_0, p, l_0\rangle$ vers la configuration $\langle s_1, p, l_1\rangle$.

Il nous reste maintenant à définir les préconditions et postconditions satisfaites par l'initialisation, l'allocation et la déallocation. Comme nous ne nous intéressons pas au code de l'allocateur, celles-ci sont assez abstraites et ne décrivent que l'interface minimale nécessaire au fonctionnement de l'allocateur.

Définition 6.23 (spécification de l'allocateur)
Le comportement de l'initialisation de l'allocateur est décrit par `RPre_init` et `RPost_init`.

```
Definition RPre_init (Ra:stateRel) (n5 n'5 n6 n'6
l l' : nat) :=
  (ret ↦ l,l') ⊗ (sp ↦ n5,n'5) ⊗ (env ↦ n6,n'6)
  ⊗ Topfrom 10 10.

Definition RPost_init (Ra:stateRel) (n5 n'5 n6 n'6
: nat) := Ra ⊗ (ret ↦ −) ⊗ (sp ↦ n5, n'5)
⊗ (env ↦ n6, n'6).

Definition total_init Ra p p' init init' :=
 forall n5 n'5 n6 n'6, total_judgment
   (RPre_init Ra n5 n'5 n6 n'6)
   (RPost_init Ra n5 n'5 n6 n'6) p p' init init'.
```

La précondition indique que l'allocateur se saisit de tout l'espace mémoire à partir du registre 10 pour pourvoir ensuite l'allouer et le désallouer sur demande. Le comportement de l'allocation est décrit par `RPre_alloc` et `RPost_alloc`.

```
Definition RPre_alloc (Ra Rc:stateRel) (n n' n5 n'5
n6 n'6 l l': nat) :=
  (ret ↦ l,l') ⊗ Ra ⊗ Rc ⊗ (arg ↦ n,n')
```

```
  ⊗ (sp ↦ n5, n'5) ⊗ (env ↦ n6, n'6).

Definition RPost_alloc (Ra Rc:stateRel) (n n' n5 n'5
n6 n'6: nat) :=
 (Ex fb,Ex fb',((ret ↦ fb,fb') ⊗
                          (Block fb fb' n n')))
 ⊗ Ra ⊗ Rc ⊗ (arg ↦ −)
 ⊗ (sp ↦ n5, n'5) ⊗ (env ↦ n6, n'6).

Definition total_alloc Ra p p' alloc alloc' :=
 forall Rc n n' n5 n'5 n6 n'6,
   total_judgment
       (RPre_alloc Ra Rc n n' n5 n'5 n6 n'6)
       (RPost_alloc Ra Rc n n' n5 n'5 n6 n'6)
        p p' alloc alloc'.
```

La précondition indique que l'allocateur lie la taille n du bloc à allouer dans le registre **arg** et lie l'adresse de retour dans le registre **ret**. À la sortie, le registre **ret** pointe vers un emplacement mémoire fb où un bloc libre de taille n. Le comportement de la déallocation est décrit par `RPre_dealloc` et `RPost_dealloc`.

```
Definition RPre_dealloc (Ra Rc:stateRel) (fb fb'
n n' n5 n'5 n6 n'6 l l': nat) :=
  (ret ↦ l,l') ⊗ (arg ↦ n,n') ⊗ (wk ↦ fb,fb')
  ⊗ Block fb fb' n n' ⊗ Ra ⊗ Rc
  ⊗ (sp ↦ n5, n'5) ⊗ (env ↦ n6, n'6).

Definition RPost_dealloc (Ra Rc:stateRel) (n5 n'5
n6 n'6: nat) :=
  (ret↦ −) ⊗ (arg ↦ −) ⊗ (wk ↦ −) ⊗
  Ra ⊗ Rc ⊗ (sp ↦ n5, n'5) ⊗ (env ↦ n6, n'6).

Definition total_dealloc Ra p p' dealloc dealloc':=
 forall Rc fb fb' n n' n5 n'5 n6 n'6,
  total_judgment
      (RPre_alloc Ra Rc fb fb' n n' n5 n'5 n6 n'6)
      (RPost_alloc Ra Rc n5 n'5 n6 n'6)
       p p' dealloc dealloc'.
```

La relation entre les programmes p et p' qui dit que leurs modules d'allocation sont reliés par R_a avec les points d'entrées `init`, `alloc`, `dealloc`, `init'`, `alloc'` et `dealloc'` est alors donnée par

```
Definition AllocSpec p p' Ra init init' alloc
  alloc' dealloc dealloc' :=
    (total_init Ra p p' init init')
  ∧ (total_alloc Ra p p' alloc alloc')
  ∧ (total_dealloc Ra p p' dealloc dealloc').
```

6.3.4 Spécification de l'état de la mémoire

Nous voulons montrer que le code compilé d'un terme de type A possède la propriété qu'il place dans le registre **ret** une entier satisfaisant l'interprétation $[\![A]\!]$ (voir la Définition 6.24 pour l'interprétation des types). Bien évidemment, nous devons au préalable établir une propriété sur le code généré localement par induction sur le terme avant de la relever au code complet. L'établissement d'une telle propriété nécessite d'interpréter l'ensemble des données (types, pile, environnement etc.) en terme de relations d'états.

Nous allons, dans un premier temps, donner l'interprétation des types. Elle est complètement standard excepté l'interprétation du type flèche qui à notre connaissance n'a jamais été menée dans ce cadre. Remarquons l'absence du connecteur séparant $\otimes$ dans cette spécification car plusieurs variables peuvent avoir le même type et donc la même spécification.

Définition 6.24 (interprétation des types)
La sémantique d'un type est définie par point fixe. Outre le type sur lequel on fait l'induction, le crochet sémantique prend comme argument la spécification de l'allocateur R_a (qui n'est pas censée être connue par le programme) et un entier qui décrit la valeur de type A.

```
Fixpoint semantics_of_types (t:ExpType) (Ra:stateRel) ptr ptr' struct t :=
  match t with
  | Int P ⇒ lift (P ptr ∧ (ptr = ptr'))
  | Bool P ⇒ lift (P (n2b ptr) ∧ (n2b ptr = n2b ptr'))
  | a ∗ b  ⇒ Ex value, Ex value2, Ex value', Ex va-
lue2', (ptr,ptr'↦value,value') ×
                  (ptr+1,ptr'+1↦value2,value2') × [[b]] Ra value va-
lue' × [[a]] Ra value2 value2')
  | a ⟶ b  ⇒ Ex Rprivate,
                     (ptr,ptr' ↦ Later ( Perp (Pre_arrow Rpri-
vate ptr ptr' Ra ([[a]]) ([[b]]))) × Rprivate)
  end
  where "'[[' t ']]'" := (semantics_of_types t ).
```

La fonction `Pre_arrow` qui interprète le type flèche est définie ci-dessous.

Nous allons maintenant décrire l'interprétation du type flèche. C'est l'élément clé de notre interprétation des types. Comme nous voulons ultérieurement autoriser le remplacement de certaines parties du code par des versions optimisées, il est impératif de donner la spécification le plus large et abstraite possible. L'idée étant de trouver une spécification qui décrit une *fonction pure*, c'est à dire un fragment de code qui ne comporte pas d'effet de bord. On verra à la Section 6.5.1 que notre spécification n'est pas assez complète pour n'autoriser que les fonctions pures. Elle permet néanmoins de démontrer le lemme de correction.

La spécification d'une fonction ne doit pas comporter de mention explicite à la façon qu'elle a de gérer son environnement ou sa pile. Par exemple, la fragment compilé doit satisfaire cette spécification indépendamment du compilateur qui l'a générée. Pour garantir cela, on utilise une relation R_{priv} privée à la fonction. Elle contient intuitivement la discipline de gestion de l'environnement et l'endroit où le code est stocké. Ces informations ne doivent pas être exploitées par le fragment de code qui fait appel à la fonction et, par

conséquent, doivent être abstraites dans la spécification. On peut voir notre interprétation comme une généralisation sémantique de l'utilisation des types existentiels dans la conversion de clôture typée [MMH96].

Comme le fonctionnement interne de la fonction est encapsulé par la relation R_{priv}, l'interprétation de la flèche se contente de décrire la sémantique d'appel de la fonction. Celle-ci suppose que l'argument de type est $[\![a]\!]$ ainsi que le pointeur de code sont en tête de pile. Elle garantit en contrepartie que le point de programme l pointé par $n+4$ satisfera la condition `Post_arrow`, c'est à dire que si elle termine, la fonction retournera en l avec un élément de type $[\![b]\!]$ en tête de pile.

Définition 6.25 (interprétation de la flèche)
L'interprétation du type flèche est donnée en deux temps. D'abord `Post_arrow` qui décrit l'état de la mémoire après l'exécution de la fonction, ensuite `Pre_arrow` qui décrit la mémoire nécessaire à la bonne exécution de la fonction.

```
Definition Post_arrow b (Ra Rc: stateRel) Rc_cloud
(n n' stack_ptr stack_ptr': nat):= Ex ptr_result, Ex ptr_result',
  (stack_ptr,stack_ptr' ↦ ptr_result,ptr_result') ⊗ (stack_ptr+1,stack_ptr'+1↦-) ⊗
  ((b Ra ptr_result ptr_result') × Rc_cloud) ⊗ (workreg ↦ -) ⊗ (argreg ↦ -) ⊗
  (retreg ↦ -) ⊗ Ra ⊗ Rc ⊗ (3↦-) ⊗ (4 ↦-)  ⊗ (spreg↦ stack_ptr,stack_ptr') ⊗
  (envreg↦ n,n') ⊗ unused_space.

Definition Pre_arrow R_private ptr_function ptr_function' Ra a b:=
  Ex Rc, Ex Rc_cloud, Ex n, Ex n', Ex ptr_arg, Ex ptr_arg', Ex stack_ptr, Ex stack_ptr',
  (stack_ptr,stack_ptr'↦ ptr_arg,ptr_arg') ⊗
  (stack_ptr+1,stack_ptr'+1↦ ptr_function,ptr_function') ⊗
  (R_private × a Ra ptr_arg ptr_arg' × Rc_cloud) ⊗
  (n+4,n'+4 ↦ Later Perp (Post_arrow b Ra Rc Rc_cloud n n' stack_ptr stack_ptr') × Rc)
  ⊗ (workreg ↦ -) ⊗ (argreg ↦ -) ⊗ (retreg ↦ -) ⊗ Ra ⊗ (3↦-) ⊗ (4↦-) ⊗
  (spreg↦ stack_ptr+1,stack_ptr'+1) ⊗ (envreg↦ n,n') ⊗ unused_space.
```

où *unused* est une relation qui stipule que les registres non spécifiés pointent vers des valeurs qui n'ont pas d'importance.

On utilise l'interprétation des types pour spécifier l'environnement. Il suffit juste de rajouter la description de la structure de liste chaînée.

Définition 6.26 (spécification de l'environnement)
L'environnement est décrit par une liste chaînée dont la tête est constituée d'un pointeur `current` vers un élément typé et d'un pointeur `current+1` pointant vers la suite de la liste. Il y a toujours la présence de la spécification privée R_a de l'allocateur.

```
Fixpoint semantics_of_env (Γ:EnvType) R_a current struct Γ :=
  match Γ with
  | nil ⇒ True
  | h :: t ⇒ ∃ ptr, ∃ next, (current ↦ ptr) × (current+1 ↦ next) ×
                            [[h]] R_a ptr × [[t]] R_a next
  end
where "[[Γ]]" := (semantics_of_env Γ).
```

Nous devons maintenant donner une sémantique relationnelle pour la pile. Celle-ci fait intervenir de manière subtile le connecteur $\otimes$ et le connecteur $\times$. En effet, tous les

pointeurs sur des éléments de la pile doivent être distingués et effaçables ; ils sont donc linéaires et décrits à l'aide du connecteur séparant $\otimes$. En revanche, les éléments eux-mêmes de la pile ne sont pas forcement distinguables (par exemple une variable dupliquée ou des fonctions mutuellement récursives). Ils sont donc classiques et doivent être décrits à l'aide du produit $\times$. Notons la présence de la relation R qui permet d'incorporer les termes classiques – comme celui décrivant l'environnement – à l'intérieur de la description de la pile. Ceci est important car il faut faire attention de regrouper tous les termes classiques afin qu'ils n'interfèrent pas avec le connecteur séparant.

Définition 6.27 (spécification de la pile)
La spécification de la pile décrit la structure filaire de celle-ci. Elle prend en argument une liste constituée de couples $(\mathbb{N}\text{at} \to \textbf{stateRel}) \times \mathbb{N}\text{at}$, un pointeur *ptr* et une relation d'état R.

```
Definition stacklist_type :=
  list ((nat → nat → stateRel) * (nat * nat)).

Fixpoint stack_description (stack_list:stacklist_type) ptr ptr' cloudrel
struct stack_list :=
  match stack_list with
  | nil ⇒ cloudrel
  | pair h (pair ptr_h ptr_h') :: t ⇒
    (ptr,ptr' ↦ ptr_h,ptr_h') ⊗ stack_description t (ptr-1) (ptr'-1)
                                ((h ptr_h ptr_h') × cloudrel)
  end.
```

La spécification de la mémoire utilise les relations décrites ci-dessus. Remarquons la présence de trois relations R_a, R_c et R'_c quantifiées universellement. La première correspond à la sémantique intrinsèque de l'allocateur ; que le terme compilé n'est pas censé connaître mais doit préserver. Les deux autres correspondent à la sémantique privée du client que le reste du programme doit accepter sans qu'elle soit explicite ; R_c étant la partie linéaire (ou séparable) et R'_c la partie classique (ou non séparable).

Définition 6.28 (spécification de la mémoire)
La spécification de la mémoire combine la description de la pile et de l'environnement gérés de manière classique ; à la description des registres gérés de manière linéaire.

```
Definition memory_specification Ra Rc Rc_cloud env stack_list
stack_free stack_free' stack_ptr stack_ptr' n n2 n3 n' n2' n3':=
  (spreg↦stack_ptr,stack_ptr') ⊗
  Block (stack_ptr+1) (stack_ptr'+1) stack_free stack_free' ⊗
  (envreg↦ n,n') ⊗  storing_space n n2 n3 n' n2' n3' ⊗
  stack_description stack_list stack_ptr stack_ptr' (⟦env⟧ Ra n n'
    × Rc_cloud) ⊗ (workreg ↦ -) ⊗ (argreg ↦ -) ⊗ (retreg ↦ -) ⊗
  Ra ⊗ Rc ⊗ (3↦-) ⊗ (4↦-) ⊗ unused_space.
```

où *unused* est une relation qui stipule que les registres non spécifiés pointent vers des valeurs qui n'ont pas d'importance.

```
Fixpoint extra_property a env Ra p p' e aux_code aux_code' base base'
                        alloc alloc' dealloc dealloc' struct a : Prop :=
  match a with
    | a ⟶ b ⇒ forall ptr ptr' fb fb',
      judgment (Pre_arrow (ptr+1,ptr'+1↦fb,fb' × ⟦env⟧  Ra fb fb') ptr ptr' Ra ⟦a⟧ ⟦b⟧)
        p p' (snd (compile_rec e aux_code (base+length aux_code) alloc dealloc))
             (snd (compile_rec e aux_code' (base'+length aux_code') alloc' deal-
loc'))
    | _ ⇒ False
  end.

Theorem compile_recr_correctness : forall base base' Ra alloc alloc' dealloc
dealloc' Rc Rc_cloud env a e (t : env |-- e:::a) start start' stack_free
stack_free' stack_ptr stack_ptr' n n2 n3 n' n2' n3',
  forall aux_code aux_code' code code' p p' stack_list main_code main_code'
  (hprog : main_code = compile_rec e aux_code (base+length aux_code) al-
loc dealloc)
  (hcode : code = (fst(fst main_code)) start)
  (hstack : snd(fst code) <= stack_free)
  (hprog' : main_code' = compile_rec e aux_code' (base'+length aux_code') al-
loc' dealloc')
  (hcode' : code' = (fst(fst main_code')) start')
  (hstack' : snd(fst code') <= stack_free'),
  PfunctionExtendsFunction p (fragfromprog (progfromlist (snd(fst main_code))) base)  →
  PfunctionExtendsFunction p (fragfromprog (progfromlist (fst (fst code))) start)  →
  PfunctionExtendsFunction p' (fragfromprog (progfrom-
list (snd(fst main_code'))) base')  →
  PfunctionExtendsFunction p' (fragfromprog (progfrom-
list (fst (fst code'))) start')  →
  (forall Rc_alloc size size' n5 n5' n6 n6',
      total_judgment (SPre_alloc Ra Rc_alloc size size' n5 n5' n6 n6')
           (SPost_alloc Ra Rc_alloc size size' n5 n5' n6 n6') p p' alloc al-
loc') →
  (forall Rc_alloc size size' base base' n5 n5' n6 n6',
      total_judgment (SPre_dealloc Ra Rc_alloc base base' size size' n5 n5' n6 n6')
           (SPost_dealloc Ra Rc_alloc n5 n5' n6 n6') p p' dealloc dealloc') →
  ((forall ptr ptr',
     judgment (memory_specification Ra Rc Rc_cloud env ((⟦a⟧ Ra, (ptr,ptr')) :: stack_list)
         (stack_free-1) (stack_free'-1) (stack_ptr+1) (stack_ptr'+1) n n2 n3 n' n2' n3') p p'
         (start+List.length (fst(fst code))) (start'+List.length (fst(fst code'))))  →
     judgment (memory_specification Ra Rc Rc_cloud env stack_list stack_free stack_free'
          stack_ptr stack_ptr' n n2 n3 n' n2' n3') p p' start start')
  ∧ (forall e' , e = LAMBDA e' →
          extra_property a env Ra p p' e' aux_code aux_code' base base' alloc al-
loc' dealloc dealloc').
```

FIG. 6.5 – Correction de la sémantique de type

L'énoncé en Coq du théorème de correction, bien que long et confus, est décrit à la Figure 6.5. Pour en faciliter la lecture, nous présentons ici une version simplifiée de l'hypothèse d'induction sur les termes non clos compilés localement avec la fonction `compile_rec`. Cette hypothèse nous permettra ensuite de déduire la correction du compilateur.

Lemme 6.29 La compilation avec `compile_rec` d'un terme M de type A dans l'environnement Γ ($\Gamma \vdash M : A$) vérifie

$$\begin{array}{l}\forall\ base\ base'\ R_a\ R_c\ R'_c\ l\ l'\ sk_{free}\ sk_{ptr}\ n\ n_2\ n_3\ p\ p'\ sk_{list},\\ \quad \texttt{compile\_rec}\ M\ base \subseteq p \Rightarrow \texttt{compile\_rec}\ M\ base' \subseteq p' \Rightarrow\\ \quad \texttt{total\_judgment alloc} \Rightarrow \texttt{total\_judgment dealloc} \Rightarrow\\ \quad \Big(\forall\ ptr, \models p, p' \rhd (l + \|\texttt{compile\_rec}\ M\ base\|), (l' + \|\texttt{compile\_rec}\ M\ base'\|) :\\ \qquad (\texttt{memory\_spec}\ R_a\ R_c\ R'_c\ \Gamma\ ((\llbracket A \rrbracket\ R_a, ptr) :: sk_{list})\ (sk_{free} - 1)(sk_{ptr} + 1)\ n\ n_2\ n_3)^{\top}\Big)\\ \quad \Rightarrow \models p, p' \rhd l, l' : (\texttt{memory\_spec}\ R_a\ R_c\ R'_c\ \Gamma\ sk_{list}\ sk_{free}\ sk_{ptr}\ n\ n_2\ n_3)^{\top}\end{array}$$

où `total_judgment alloc` et `total_judgment dealloc` sont des versions écourtées des jugements pour l'allocation et la déallocation comme nous les avons décrits en Section 6.3.3 et $\|\texttt{compile\_rec}\ M\ base\|$ dénote le nombre d'instructions du fragment de code compilé.

À partir de ce lemme, on en déduit un lemme de correction pour la compilation avec la fonction `compile`.

Lemme 6.30 (lemme de correction) La compilation avec `compile` d'un terme clos M de type A vérifie

$$\begin{array}{l}\forall\ base\ base'\ R_a\ l\ l'\ p\ p',\ \texttt{compile}\ M\ base \subseteq p \Rightarrow \texttt{compile}\ M\ base' \subseteq p' \Rightarrow\\ \quad \texttt{total\_judgment init} \Rightarrow \texttt{total\_judgment alloc} \Rightarrow \texttt{total\_judgment dealloc} \Rightarrow\\ \quad \models p, p' \rhd (l + \|\texttt{compile}\ M\ base\|), (l' + \|\texttt{compile}\ M\ base'\|)\ :\\ \qquad (\exists ptr, (\mathbf{ret} \mapsto ptr) \otimes ((\llbracket A \rrbracket)\ R_a\ ptr))^{\top} \Rightarrow\\ \quad \models p, p' \rhd l, l' : (\texttt{Topfrom}\ 0\ 0)^{\top}\end{array}$$

où `total_judgment init` est la version écourtée du jugement pour l'initialisation de l'allocateur comme nous l'avons décrite en Section 6.3.3.

Corollaire 6.31 Un corollaire direct du lemme de correction est que si un terme M a pour type Nat P pour un certain prédicat P, alors le code compilé va placer dans le registre **ret** un entier n tel que Pn.

Remarquons que la présence de types de base enrichis avec des prédicats est cruciale. Sinon, le corollaire deviendrait si un terme M a pour type $\mathbb{N}$at, alors après exécution du code compilé, le registre **ret** contient un entier ; or ceci est toujours le cas.

6.4 Les deux piliers de la preuve

Nous n'allons bien évidemment pas présenter la preuve entière réalisée en Coq. Néanmoins, nous allons donner ici les deux piliers qui ont permis de démontrer les Lemmes 6.29 et 6.30. Le premier pilier réside dans l'élaboration de propriétés atomiques vérifiées par les instructions de base du langage assembleur. Au cours de la preuve, nous n'aurons qu'à appliquer successivement ces propriétés atomiques pour en déduire la propriété d'un fragment complet. Le problème est que ces propriétés atomiques requièrent chacune une certaine forme pour la spécification de l'état de la mémoire. Il faut donc pouvoir réordonner l'assemblage de produits et de connecteurs séparants présents dans cette description. C'est là qu'intervient le deuxième pilier de la preuve qui consiste en une utilisation intensive de la réécriture fournie par le module **Setoid** de Coq.

6.4.1 L'interprétation des instructions de base

Nous présentons dans un premier temps la formalisation des propriétés des instructions assembleur de base. Nous avons dû établir une propriété pour chaque combinaison d'instruction et fonctions d'adressage

- `i_move (d_imm m) src`;
- `i_move (d_ind m) src`;
- `i_move (d_indo ofs m) src`;
- `i_branch (s_imm m)`;
- etc.

Nous ne donnons ici que les propriétés vérifiées par les deux dernières instructions. Le lecteur qui resterait sur sa faim est invité à regarder directement le code source en Coq pour découvrir les propriétés des autres instructions.

Lemme 6.32 L'instruction `i_move (d_indo ofs m) src` vérifie

$$
\begin{array}{l}
\forall\ p\ p'\ l\ l'\ m\ n\ ofs\ P\ src\ src'\ Pinv\ Q,\\
\quad p(l) = \texttt{i\_move (d\_indo ofs m) src} \Rightarrow\\
\quad p'(l') = \texttt{i\_move (d\_indo ofs m) src'} \Rightarrow\\
\quad (\forall\ s\ s'\ k\ p_0\ p'_0,\ ((m \mapsto n) \otimes (n + ofs \mapsto P) \otimes P_{inv})\, s\, s'\, p_0\, p'_0\, k \Rightarrow\\
\qquad \exists l_0\ l'_0, (Q\, l_0\, l'_0\, p_0\, p'_0\, k \wedge \texttt{sem\_src src s} = l_0 \wedge \texttt{sem\_src src' s'} = l'_0) \Rightarrow\\
\quad \models p, p' \rhd (l+1), (l'+1) : ((m \mapsto n) \otimes (n + ofs \mapsto Q) \otimes Pinv)^{\top} \Rightarrow\\
\quad \models p, p' \rhd l, l' : ((m \mapsto n) \otimes (n + ofs \mapsto P) \otimes Pinv)^{\top}
\end{array}
$$

La propriété dit grosso modo que si pour tous couples s, s' satisfaisant la précondition, la source de l'instruction `i_move (d_indo ofs m)` selon s et s' vérifie la propriété Q, alors après exécution de l'instruction, l'emplacement `n+ofs` pointe vers un point de programme vérifiant Q.

Lemme 6.33 L'instruction `i_branch (s_imm n)` vérifie

$$
\begin{array}{l}
\forall\ p\ p'\ l\ l'\ m\ P,\\
\quad p(l) = \texttt{i\_branch (s\_imm m)} \Rightarrow\\
\quad p'(l') = \texttt{i\_branch (s\_imm m)} \Rightarrow\\
\quad P \preceq (m \mapsto \Diamond P) \Rightarrow\\
\quad \models p, p' \rhd l, l' : P^{\top}
\end{array}
$$

Cette propriété indique que si on peut déduire de P que m pointe vers une continuation qui satisfait $\Diamond P^{\top}$, alors « brancher » sur m à partir d'un état qui satisfait P est sûr.

6.4.2 La réécriture par Setoid

Tout au long de la preuve des Lemmes 6.29 et 6.30, nous avons besoin de réorganiser la description de l'espace mémoire. Nous avons déjà vu à la Section 6.2.1 que le tenseur séparant $\otimes$ faisait de **stateRel** une catégorie monoïdale symétrique. De plus, le connecteur $\times$ définit un produit cartésien sur cette catégorie. Nous allons utiliser toutes ces propriétés pour faire de la réécriture automatique sur les descriptions d'espace mémoire données par des relations d'états. Dans un premier temps, nous allons déclarer que $\preceq$ induit une catégorie sur **stateRel** au sens où c'est un préordre. Ceci s'écrit avec le module **Setoid** de la manière suivante

```
Add Relation stateRel stateRelLeq
  reflexivity proved by stateRelLeq_refl
  transitivity proved by stateRelLeq_trans
as stateRelLeqRel.
```

Le module **Setoid** permet alors de définir des foncteurs, c'est à dire des morphismes qui vont préserver cette relation. Par exemple, le tenseur est déclaré comme un bifoncteur

```
Add Morphism stateRelTensor with signature
  stateRelLeq ++> stateRelLeq ++> stateRelLeq
as ppredtensorcompat2.
```

Un fois le tenseur séparant déclaré comme un foncteur, on peut définir des transformations naturelles qui vont être utilisées comme des règles de réécriture. Par exemple, le lemme d'associativité lié à la structure monoïdale du tenseur va devenir une règle de réécriture permettant de réécrire le but

$$(p_1 \otimes p_2) \otimes p_3$$

en

$$p_1 \otimes (p_2 \otimes p_3)$$

Mais le gros avantage de **Setoid** est qu'il permet de faire cette réécriture au sein d'un large contexte dès que celui est constitué de morphismes compatibles avec la relation considérée. On doit donc montrer que la relation d'orthogonalité est un foncteur contravariant, que la constitution d'un jugement est un foncteur covariant, etc.

Ensuite, il ne nous reste plus qu'à définir des *tactiques* de Coq pour automatiser la réécriture. Voici par exemple une tactique permettant de placer le n-ème terme en tête d'un tenseur. Elle permet de réécrire aussi bien un jugement qu'une simple inégalité en utilisant filtrage (*pattern matching* en anglais) de Coq.

```
Tactic Notation "focus" integer(n) :=
  do n setoid_rewrite stateRelTensor_assoc;
  match goal with
  | |- stateRelLeq ((?p1 ⊗ ?p2) ⊗ ?p3) ?q ⇒
    setoid_rewrite (stateRelTensor_comm p1 p2)
  | |- judgment ((?p1 ⊗ ?p2) ⊗ ?p3) ?prog ?prog' ?loc ?loc' ⇒
    setoid_rewrite (stateRelTensor_comm p1 p2)
  end;
  repeat setoid_rewrite ← stateRelTensor_assoc.
```

L'idée de cette tactique est simple. Supposons que le terme à réécrire soit associé à droite

$$p_1 \otimes (p_2 \otimes (p_3 \otimes \ldots \otimes (p_{n-1} \otimes (p_n \otimes (p_{n+1} \otimes \ldots))))).$$

Pour faire apparaître le n-ème terme en tête, il suffit d'appliquer n fois l'associativité avec la tactique `do n setoid_rewrite stateRelTensor_assoc`

$$(((((p_1 \otimes p_2) \otimes p_3) \otimes \ldots) \otimes p_{n-1}) \otimes p_n \otimes (p_{n+1} \otimes (\ldots))$$

puis de faire commuter le terme p_n avec $(p_1 \otimes \ldots \otimes p_{n-1})$ grâce à la tactique `setoid_rewrite (stateRelTensor_comm` $(p_1 \otimes \ldots \otimes p_{n-1})\ p_n$)

$$(p_n \otimes (((p_1 \otimes p_2) \otimes p_3) \otimes \ldots \otimes p_{n-1})) \otimes (p_{n+1} \otimes (\ldots))$$

et enfin de remettre à nouveau le terme sous forme associé à droite en utilisant la tactique `repeat setoid_rewrite ← stateRelTensor_assoc`

$$p_n \otimes (p_1 \otimes (p_2 \otimes (p_3 \otimes \ldots \otimes (p_{n-1} \otimes (p_{n+1} \otimes \ldots)))))$$

La tactique `focus` permet de ramener en tête – en une étape – n'importe quel terme présent dans la description de la mémoire. Elle est donc très utile lorsqu'on veut par exemple utiliser le lemme sur l'instruction `i_move (d_indo ofs m) src` qui requiert que le registre `m` soit en tête du terme. On peut bien évidemment refaire la même chose pour le produit pour pouvoir réécrire dans la description de la pile et de l'environnement.

En combinant ces deux piliers, il est alors possible de mener l'induction du Lemme 6.29 sans souffrance.

6.5 Travaux Futurs

Nous revenons maintenant sur deux aspects fondamentaux qui ne sont pas encore présents dans notre travail. Le premier est le quantificateur universel dans le système de typage de PCF_v qui permettrait une plus grande richesse dans la spécification des types. Le second est le raisonnement équationnel qui reste la motivation principale qui nous a poussés à utiliser une sémantique relationnelle plutôt que prédicative.

6.5.1 Le quantificateur universel

Un des grands absents du système de typage de PCF_v que nous avons donné est le quantificateur universel. En effet, nous aimerions par exemple typer une fonction comme `factorielle` par le type

$$\forall\, n, \text{Nat}\ n \mapsto \text{Nat}\ n! \tag{6.1}$$

qui indique que c'est une fonction qui prend un entier n et renvoie l'entier $n!$. Malheureusement, il nous faut faire encore un peu de chemin avant de pouvoir inclure le quantificateur universel dans nos types. La spécification actuelle du code compilé ne permet pas de déduire une propriété de « préservation des suprema » dont nous aurions besoin pour passer du fait qu'étant donné un entier n, `factorielle` est de type

$$\text{Nat}\ n \mapsto \text{Nat}\ n!, \tag{6.2}$$

au fait que `factorielle` a le type (6.1). La différence entre les deux paraît mineure au premier abord mais s'avère plus profonde. En effet, utiliser du code assembleur autorise de nombreuses possibilités qui ne sont pas présentes dans un langage fonctionnel comme PCF_v. Dans le code assembleur, le programme et le contexte sont mélangés. Ainsi, il n'y pas de différence structurelle entre une fonction et un client qui fait appel à cette fonction. Pire, une fonction peut contenir un client qui fait appel à elle-même !

Avec cette idée, il est en effet possible de définir une fonction « sosie » de `factorielle` qui agit comme `factorielle` si on l'utilise toujours avec le même entier mais qui agit différemment si on l'utilise avec deux entiers distincts. Pour cela, il faut que cette fonction soit donnée avec un appelant interne qui va utiliser la fonction pour deux entiers distincts. Cette fonction sosie peut alors agir comme `factorielle` sur tous les appelants possibles excepté l'appelant interne qu'elle peut détecter par son adresse de retour. Cette fonction aura alors le type (6.2) pour tout n mais pas le type (6.1).

Bien sûr, ce cas de figure ne se présente pas pour le code compilé mais la spécification actuelle ne permet pas de le certifier. Il apparaît que le problème avec la fonction sosie de `factorielle` est qu'elle ne vérifie pas une propriété de préservation des suprema du style

$$\frac{\models p, p' \rhd l_1, l_1' : X^\top \Rightarrow \models p, p' \rhd l_0, l_0' : A^\top \qquad \models p, p' \rhd l_1, l_1' : X^\top \Rightarrow \models p, p' \rhd l_0, l_0' : B^\top}{\models p, p' \rhd l_1, l_1' : X^\top \Rightarrow \models p, p' \rhd l_0, l_0' : A \times B^\top}$$

Il nous reste néanmoins à formaliser cette propriété et à montrer que tous les codes générés la vérifient. Une fois ce travail effectué, on pourra alors remplacer sereinement un fragment de code compilé par une version optimisée pour autant qu'elle satisfasse la même spécification et qu'elle « préserve les suprema ».

6.5.2 Le raisonnement équationnel

La deuxième faiblesse de notre travail est que nous ne pouvons pas pour l'instant établir d'égalité intéressante entre deux fragments de code assembleur. Ainsi, il est pour le moment impossible de montrer que le code compilé de $M + N$ se comporte de la même manière que le code compilé de $N + M$. En d'autres termes, on ne peut pas prouver la commutativité de l'addition au niveau assembleur. Le problème réside dans notre sémantique basée sur l'équidivergence qui empêche le parcours du programme de gauche tout en gelant le programme de droite. Ainsi, il est impossible d'exprimer clairement la précondition satisfaite au niveau (1) avant l'exécution du fragment de code `N` comme l'illustre la Figure 6.6. Le même problème se pose au niveau (2) pour la postcondition. Il nous

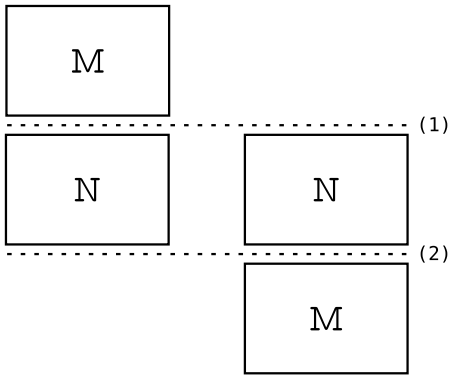


Fig. 6.6 – Raisonnement équationnel : les relations satisfaites aux points (1) et (2) ne sont pas exprimables dans la sémantique.

faut donc trouver une autre façon de gérer la divergence qui permette temporairement au programme de gauche de diverger sans que celui de droite ne diverge.

Bibliographie

Le libraire, Arcimboldo (1556)

[Abr96] Samson Abramsky : Retracting some paths in process algebra. *In 7th International Conference on Concurrency Theory*, pages 1–17, London, UK, 1996. Springer-Verlag.

[Abr03] Samson Abramsky : Sequentiality vs. concurrency in games and logic. *Mathematical Structures in Computer Science*, 13:531–565, 2003.

[AC04] Samson Abramsky et Bob Coecke : A categorical semantics of quantum protocols. *In 19th IEEE Symposium on Logic in Computer Science, July 2004, Turku, Finland*, pages 415–425. IEEE Computer Society, 2004.

[AHM98] S. Abramsky, K. Honda et G. McCusker : A fully abstract game semantics for general reference. *In 13th IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1998.

[Ahm04] Amal Ahmed : *Semantics of types for mutable state*. Thèse de doctorat, Princeton University, Princeton, NJ, USA, 2004.

[AHS02] Samson Abramsky, Esfandiar Haghverdi et Philip Scott : Geometry of interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12(5):625–665, 2002.

[AJ94] Samson Abramsky et Radha Jagadeesan : Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.

[AJM00] Samson Abramsky, Radha Jagadeesan et Pasquale Malacaria : Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.

[AM98] Samson Abramsky et Guy McCusker : Call-by-value games. *In* Mogens Nielsen et Wolfgang Thomas, éditeurs : *6th Annual Conference of the European Association for Computer Science Logic*, volume 1414 de *Lecture Notes in Computer Science*. Springer, 1998.

[AMRV07] A.W. Appel, P.A. Melliès, C.D. Richards et J. Vouillon : A very modal model of a modern, major, general type system. *34th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 109–122, 2007.

[App98] Andrew Appel : *Modern Compiler Implementation in ML*. Cambridge University Press, 1998.

[BBHdP92] Nick Benton, Gavin Bierman, Martin Hyland et Valeria de Paiva : Term assignement for intuitionistic linear logic. Rapport technique 262, Computer Laboratory, University of Cambridge, 1992.

[BC82] G. Berry et P.-L. Curien : Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[BDER97] Patrick Baillot, Vincent Danos, Thomas Ehrhard et Laurent Regnier : Believe it or not, ajm's games model is a model of classical linear logic. *In 12 th IEEE Symposium on Logic in Computer Science*, pages 68–75, 1997.

[Ben95] Nick Benton : A mixed linear and non-linear logic: Proofs, terms and models. *In 3th Annual Conference of the European Association for Computer Science Logic*, volume 933 de *LNCS*, Poland, June 1995. Springer-Verlag.

[Ben06] Nick Benton : Abstracting allocation: The new new thing. *In 15th Annual Conference of the European Association for Computer Science Logic*, volume 4207 de *LNCS*. Springer-Verlag, September 2006.

[Bla72] Andrea Blass : Degrees of indeterminacy of games. *Fundations of Mathematics*, 77:151–166, 1972.

[Bla92] Andrea Blass : A games semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–220, 1992.

[BW85] Michael Barr et Charles Wells : *Toposes, triples and theories*. Springer-Verlag, 1985.

[Bé67] Jean Bénabou : Introduction to bicategories. *In Midwest Category Seminar*, volume 42, pages 1–77. Springer, 1967.

[Bé73] Jean Bénabou : Les distributeurs. Séminaires de Mathématique Pure 33, Université Catholique de Louvain, 1973.

[Bé00] Jean Bénabou : Distributors at work. http://www.mathematik.tu-darmstadt.de/∼streicher/FIBR/DiWo.pdf.gz, June 2000.

[CHP04] Eugenia Cheng, Martin Hyland et John Power : Pseudo-distributive laws. *Electronic Notes in Theoretical Computer Science*, 89, 2004.

[CJSV94] Aurelio Carboni, Scott Johnson, Ross Street et Dominic Verity : Modulated bicategories. *Journal of Pure and Applied Algebra*, 94(3):229–282, 1994.

[CS92] Robin Cockett et Robert Seely : Weakly distributive categories. *Applications of Categories in Computer Science: Proceedings of the LMS Symposium, Durham 1991*, 1992.

[CS07] Robin Cockett et Robert Seely : Polarized category theory, modules, and game semantics. *Theory and Applications of Categories*, 18(2):4–101, 2007.

[CW05] Gian Luca Cattani et Glynn Winskel : Profunctors, open maps and bisimulation. *Mathematical Structures in Computer Science*, 15(3):553–614, 2005.

[Dav03] Maulik Dave : Compiler verification: a bibliography. *ACM SIGSOFT Software Engineering Notes*, 28(6), 2003.

[DHR96] V. Danos, H. Herbelin et L. Regnier : Game semantics and abstract machines. *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science*, 12:394–405, 1996.

[Dub74] Eduardo Dubuc : Free monoids. *Journal of Algebra*, 29:208–228, 1974.

[Ehr02] Thomas Ehrhard : On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(05):579–623, 2002.

[Ehr05] Thomas Ehrhard : Finiteness spaces. *Mathematical Structures in Computer Science*, 15(04):615–646, 2005.

[FK72] Peter Freyd et Max Kelly : Categories of continuous functors I. *Journal of Pure and Applied Algebra*, 2:169–191, 1972.

[FLK80] François Foltz, Christian Lair et Max Kelly : Algebraic categories with few monoidal biclosed structures or none. *Journal of Pure and Applied Algebra*, 17:171–177, 1980.

[Gam06] Nicola Gambino : On the coherence conditions for pseudo-distributive laws. manuscrit, 2006.

[Gir87] Jean-Yves Girard : Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir91] Jean-Yves Girard : A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[GLLN02] Jean Goubault-Larrecq, Slawomir Lasota et David Nowak : Logical relations for monadic types. *In 11th Annual Conference of the European Association for Computer Science Logic*, volume 2471 de *Lecture Notes in Computer Science*, pages 553–568. Springer-Verlag, septembre 2002.

[Har00] Russ Harmer : *Games and Full Abstraction for Nondeterministic Languages*. Thèse de doctorat, University of London, 2000.

[Has02a] Masahito Hasegawa : The uniformity principle on traced monoidal categories. *Electronic Notes in Theoretical Computer Science*, 69, 2002.

[Has02b] Ryu Hasegawa : Two applications of analytic functors. *Theoretical Computer Science*, 272(1-2):113–175, 2002.

[Has05] Masahito Hasegawa : Communication personnelle, juillet 2005.

[HO92] Martin Hyland et Luke Ong : Fair games and full completeness for multiplicative linear logic without the mix rule. Manuscrit, 1992.

[HO00] Martin Hyland et Luke Ong : On full abstraction for PCF: I, II and III. *Information and Computation*, 163(2):285–408, décembre 2000.

[Hoa69] Tony Hoare : An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[Hou07] Robin Houston : *Linear logic without units*. Thèse de doctorat, University of Manchester, 2007.

[HP02] Martin Hyland et John Power : Pseudo-commutative monads and pseudo-closed 2-categories. *Journal of Pure and Applied Algebra*, 175(1-3):141–185, November 2002.

[HS97] Martin Hofmann et Thomas Streicher : Continuation models are universal for λ-μ-calculus. *In 12th IEEE Symposium on Logic in Computer Science*, page 387, 1997.

[HS02] Martin Hofmann et Thomas Streicher : Completeness of continuation models for λ-μ-calculus. *Information and Computation*, 179(2):332–355, December 2002.

[HS03] Martin Hyland et Andrea Schalk : Glueing and orthogonality for models of linear logic. *Theoretical Computer Science*, 294:183–231, 2003.

[Jac94] Bart Jacobs : Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69(1):73–106, 1994.

[Joy77] André Joyal : Remarques sur la théorie des jeux à deux personnes. *Gazette des Sciences Mathématiques du Québec*, 1(4):46–52, 1977. English version by Robin Houston available.

[JS91] André Joyal et Ross Street : The Geometry of Tensor Calculus, I. *Advances in Mathematics*, 88:55–113, 1991.

[JSV96] André Joyal, Ross Street et Domnic Verity : Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(447-468):184, 1996.

[Kel82] Max Kelly : *Basic Concepts of Enriched Category Theory*, volume 64 de *Lecture Notes in Mathematics*. Cambridge University Press, 1982.

[Kel86] M. Kelly : A survey of totality for enriched and ordinary categories. *Cahiers Topologie Geom. Differentielle Categoriques*, 27:109–131, 1986.

[KL80] Max Kelly et Miguel Laplaza : Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.

[Koc72] Anders Kock : Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972.

[KP93] Max Kelly et John Power : Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *Journal of pure and applied algebra*, 89(1-2):163–179, 1993.

[KS72] Max Kelly et Ross Street : Review of the elements of 2-categories. *Proceedings Sydney Category Theory Seminar*, 73:75–103, 1972.

[Lac07] Steve Lack : A 2-categories companion. *Arxiv preprint math/0702535*, 2007.

[Lac08] Steve Lack : Note on the construction of free monoids. *Arxiv preprint arXiv:0802.1946*, 2008.

[Lai97] James Laird : Full abstraction for functional languages with control. *In 12th IEEE Symposium on Logic in Computer Science*, pages 58–67, 1997.

[Lam95] François Lamarche : Games semantics for full propositional linear logic. *In 10th IEEE Symposium on Logic in Computer Science*, pages 464–473. IEEE Computer Society Press, June 1995.

[Lau02a] Olivier Laurent : *Etude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, 2002.

[Lau02b] Olivier Laurent : Polarized games (extended abstract). *In 17th IEEE Symposium on Logic in Computer Science*, pages 265–274, Copenhagen, juillet 2002. IEEE Computer Society Press.

[Law63] William Lawvere : *Functorial Semantics of Algebraic Theories and Some Algebraic Problems in the context of Functorial Semantics of Algebraic Theories*. Thèse de doctorat, Columbia University, 1963.

[Law67] William Lawvere : Ordinal sums and equational doctrines. *In Lecture Notes in Mathematics*, volume 80, pages 141–155. Springer, 1967.

[Lei04] Tom Leinster : *Higher Operads, Higher Categories*. Cambridge University Press, 2004.

[Ler08] Xavier Leroy : A formally verified compiler back-end. Submitted, juillet 2008.

[LL78] Paul Lorenzen et K. Lorenz : *Dialogische Logik*. Wissenschaftliche Buchgesellschaft, 1978.

[Lor61] Paul Lorenzen : Ein dialogisches Konstruktivitatskriterium. *Infinitistic Methods*, pages 193–200, 1961.

[LRS93] Yves Lafont, Bernhard Reus et Thomas Streicher : Continuation semantics or expressing implication by negation. *Rapport Technique*, 9321, 1993.

[Mar99] Francisco Marmolejo : Distributive laws for pseudomonads. *Theory and Applications of Categories*, 5(5):91–147, 1999.

[May06] Peter May : Duality in bicategories and topological applications. Présentation donnée à la conférence en la mémoire de MacLane, avril 2006. http://www.math.uchicago.edu/ may/TALKS/SwitzerlandTalk.pdf.

[McC96] Guy McCusker : *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Thèse de doctorat, University of London, 1996.

[Mel] Paul-André Melliès : Functorial boxes in string diagrams. Invited talk in CSL'06.

[Mel05a] Paul-André Melliès : Asynchronous Games 3 An Innocent Model of Linear Logic. *Electronic Notes in Theoretical Computer Science*, 122:171–192, 2005.

[Mel05b] Paul-André Melliès : Asynchronous Games 4: A Fully Complete Model of Propositional Linear Logic. *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 386–395, 2005.

[Mel08] Paul-André Melliès : Categorical Semantics of Linear Logic. À paraître dans *Panoramas et Synthèses*, Société Mathématique de France, 2008.

[Mil94] Robin Milner : Action calculi v: reflexive molecular forms. Unpublished, third draf, June 1994.

[ML71] Saunders Mac Lane : *Categories for the working mathematician*. Springer, 1971.

[MMH96] Yasuhiko Minamide, Greg Morrisett et Robert Harper : Typed closure conversion. *In 23th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 271–283, New York, NY, USA, 1996. ACM.

[Mog91] Eugenio Moggi : Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[MP67] John McCarthy et James Painter : Correctness of a Compiler for Arithmetic Expressions. *Proceedings Symposium in Applied Mathematics*, 19:33–41, 1967.

[MP07] Guy McCusker et David Pym : A games model of bunched implications. *In 16th Annual Conference of the European Association for Computer Science Logic*, pages 573–588, 2007.

[MS] Paul-André Melliès et Peter Selinger : Games are continuation models ! Talk at Full Completeness and Full Abstraction, Satellite workshop of LICS 2001.

[MWCG99] G. Morrisett, D. Walker, K. Crary et N. Glew : From system F to typed assembly language. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 21(3):527–568, 1999.

[Nec97] George Necula : Proof-carrying code. *In 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 106–119, New York, NY, USA, 1997. ACM.

[Ong96] Luke Ong : A semantic view of classical proofs: Type-theoretic, categorical, and denotational characterizations. *In 11th IEEE Symposium on Logic in Computer Science*, pages 230–241, 1996.

[Pen71] Roger Penrose : Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications*, pages 221–244, 1971.

[Pow99] John Power : Enriched lawvere theories. *Theory and Applications of Categories*, 6(7):83–93, 1999.

[PR84] Roger Penrose et Wolfgang Rindler : *Spinors and space-time.* Cambridge University Press, 1984.

[PR97] John Power et Edmund Robinson : Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7:453–468, 1997.

[PR01] D. Pym et E. Ritter : On the semantics of classical disjunction. *Journal of Pure and Applied Algebra*, 159(2-3):315–338, 2001.

[Rey78] John Reynolds : Syntactic control of interference. *5th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 39–46, 1978.

[Rey02] J.C. Reynolds : Separation logic: A logic for shared mutable data structures. *17th IEEE Symposium on Logic in Computer Science*, pages 55–74, 2002.

[Sel01] Peter Selinger : Control categories and duality: on the categorical semantics of the λ-μ-calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.

[Sel03] Peter Selinger : Some Remarks on Control Categories. *Manuscript*, 2003.

[Str74] Ross Street : *Fibrations and Yoneda's lemma in a 2-category*, volume 420 de *Lecture Notes in Mathematics*, pages 104–133. Springer Verlag, 1974.

[Str80] Ross Street : Fibrations in bicategories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 21(2):111–160, 1980.

[SW73] Ross Street et Robert Walters : The Comprehensive Factorization Of A Functor. *American Mathematical Society*, 79(5), 1973.

[Thi97] Hayo Thielecke : *Categorical Structure of Continuation Passing Style.* Thèse de doctorat, University of Edinburgh, 1997.

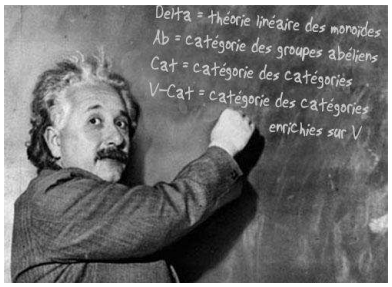
[Val04] Bruno Vallette : Free monoid in monoidal abelian categories. arXiv : math.CT/0411543, 2004.

[Woo82] Richard Wood : Abstract proarrows i. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 23(3):279–290, 1982.

[Woo85] Richard Wood : Proarrows ii. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 26(2):135–168, 1985.

[You89] William D. Young : A mechanically verified code generator. *Journal of Automated Reasoning*, 5(4):493–518, 1989.

Notations



Rêveries catégoriques.

Modalités de ressource en logique tensorielle.

$!_w$	Modalité affine, p.53
$!_c$	Modalité dupliquante, p.53
$!_e$	Modalité exponentielle, p.53

Théories algébriques.

Δ	Théorie linéaire des monoïdes, p.40
Δ_{face}	Théorie linéaire des objets pointés, p.116
$\mathbb{M}$	Théorie algébrique des monoïdes, p.39
$\mathbb{F}^{op}$	Théorie algébrique triviale, p.39
$\mathbb{N}$	Théorie linéaire triviale, p.40
Bij	Théorie symétrique triviale, p.41
FinSet	Théorie symétrique des monoïdes commutatifs, p.41
Modèle$(\mathbb{L}, \mathcal{C})$	Catégorie enrichie des $\mathbb{L}$-modèles et transformations naturelles T-algébriques, p.110

Table des catégories définies dans ce manuscrit.

$\mathcal{N}$	Catégorie des jeux de Conway négatifs, p.70
$\mathcal{G}$	Catégories des jeux de Conway à gain, p.72
Ab	Catégorie des groupes abéliens, p.25
AHM	Catégorie des arènes avec question/réponse et des stratégies bien parenthésées filaires, p.145
$\mathcal{B}$	Catégorie des jeux multiparenthésés, p.140
$\mathcal{B}_O$	Catégorie des jeux multiparenthésés négatifs, p.140
Cat	Catégorie des catégories, p.16
Cat$'$	Catégories des catégories munie du produit tensoriel asynchrone, p.34
$\mathbf{Cat}'_2$	Sesqui-catégorie des catégories, foncteurs et transformations, p.36
$\mathbf{Cat}_2$	2-catégorie des catégories, p.19
$\mathcal{V}$-**Cat**	Catégories enrichies sur $\mathcal{V}$, p.88

$\mathcal{V}$-**Cat**$_2$	2-catégorie des catégories enrichies sur $\mathcal{V}$, p.88
Cat^T	2-catégorie enrichie des catégories T-algébriques, foncteurs T-algébriques et transformations naturelles T-algébriques, p.109
ControlMult	2-catégorie des multicatégories de contrôle, morphismes de multicatégories de contrôle et transformations naturelles, p.173
Conway	Catégorie des jeux de Conway, p.70
Dist	Bicatégorie des catégories, distributeurs et transformations naturelles, p.105
$\mathcal{V}$-**Dist**	Bicatégorie enrichie des $\mathcal{V}$-catégories, $\mathcal{V}$-distributeurs et $\mathcal{V}$-transformations naturelles, p.104
G(**Mat**)	Recollement le long de **Mat**(**1**, −), p.63
Grp	Catégorie des groupes, p.15
Grph	Catégorie des graphes, p.15
k-Alg	Catégorie des k-algèbres, p.82
k-Big	Catégorie des k-bigèbres, p.84
k-Mod	Catégorie des k-modules, p.82
Lax-T-Alg	Bicatégorie enrichie des $\mathcal{V}$-algèbres relâchées sur T, morphismes relâchés et 2-cellules algébriques, p.101
Mat	Catégorie des matrices à valeurs dans $\overline{\mathbb{N}}$, p.61
MonCat	2-catégorie des catégories monoïdales foncteurs relâchés et transformations monoïdales, p.19
MonCat$_{\mathbf{fort}}$	2-catégorie des catégories monoïdales foncteurs forts et transformations monoïdales, p.19
MonCat$_{\mathbf{str}}$	2-catégorie des catégories monoïdales foncteurs stricts et transformations monoïdales, p.19
MultiCat	Catégorie des multicatégories et morphismes de multicatégories, p.163
natRel	Catégorie des relations sur les points de programme, p.186
poSet	2-catégorie des ensembles partiellement ordonnés et fonctions croissantes, p.21
Ps-T-Alg	Bicatégorie enrichie des $\mathcal{V}$-algèbres fortes sur T, morphismes forts et 2-cellules algébriques, p.101
Rel	Catégorie des relations, p.15
Ens	Catégorie des ensembles, p.15
SCat	Catégorie des catégories petites, p.16
SMonCat	2-catégorie des catégories monoïdales symétriques, foncteurs relâchés symétriques et transformations monoïdales, p.19
SMonCat$_{\mathbf{fort}}$	2-catégorie des catégories monoïdales symétriques, foncteurs forts symétriques et transformations monoïdales, p.19
SMonCat$_{\mathbf{str}}$	2-catégorie des catégories monoïdales symétriques, foncteurs stricts symétriques et transformations monoïdales, p.19
SMultiCat	Catégorie des multicatégories symétriques et morphismes symétriques de multicatégories symétriques, p.163

stateRel	Catégorie des relations sur les états de la mémoire, p.185

Notations catégoriques.

$\mathcal{C}^T$	Catégorie d'Eilenberg-Moore sur la monade T, p.23
$\mathcal{C}_T$	Catégorie de Kleisli sur la monade T, p.23
$\mathcal{C}^\neg$	Catégorie de continuation sur la négation tensoriel $\neg : \mathcal{C} \to \mathcal{C}^{\mathrm{op}}$, p.50
$\mathcal{C}^{\mathrm{op}}$	Catégorie opposée de $\mathcal{C}$, p.15
$[\mathcal{C}, \mathcal{C}]$	Catégorie des endofoncteurs sur $\mathcal{C}$ et transformations naturelles, p.27
$\int_A G(A, A)$	Fin du $\mathcal{V}$-foncteur G, p.90
$\int^A G(A, A)$	Cofin du $\mathcal{V}$-foncteur G, p.90
$\{F, G\}$	Limite de G indexée par F, p.89
$F \star G$	Colimite de G indexée par F, p.89
$X \otimes C$	Tenseur de C par X, p.90
$X \pitchfork C$	Cotenseur de C par X, p.90
$\mathrm{Elt}(\varphi)$	Catégorie des éléments d'un préfaisceau φ, p.25
$Fam(\mathcal{C})$	Construction de famille ou complétion libre de $\mathcal{C}$ par coproduits finis, p.78
$\mathrm{Int}(\mathcal{C})$	Catégorie compacte fermée libre sur une catégorie monoïdale symétrique tracée $\mathcal{C}$, p.129
$\mathrm{Lan}_J F$	Extension de Kan à gauche de F le long de J, p.102
$\mathrm{Ran}_J F$	Extension de Kan à droite de F le long de J, p.102
$\mathcal{Z}(\mathcal{C})$	Centre de la catégorie prémonoïdale $\mathcal{C}$, p.35

Sémantique des jeux.

$\mathbb{B}$	Arène booléenne, p.133
E_A	Ensemble des coups du jeu de Conway A, p.67
$\vdash_A$	Relation de justification d'un jeu d'arène A, p.145
κ_A	Opérateur de ressource associé au jeu A, p.137
λ_A	Fonction indiquant la polarité des coups du jeu de Conway A, p.67
Play_A	Ensemble des parties d'un jeu A, p.67
$Q_A(x)$	Ensemble de requêtes associées à la position x du jeu A, p.137
$[m]$	Relation de résidus au coup m, p.137
$\star_A$	Racine du jeu de Conway A, p.67
V_A	Ensemble des positions du jeu de Conway A, p.67
$\texttt{cell}_A$	Stratégie interprétant la cellule mémoire de type A, p.147
$\texttt{const\_cell}_A$	Stratégie interprétant la cellule mémoire constante de type A, p.151
$\texttt{lin\_cell}_A$	Stratégie interprétant la cellule mémoire linéaire de type A, p.147

Registres et notations utilisés pour le compilateur.

arg	Registre de l'argument avant un appel à l'allocateur, p.189
env	Registre du pointeur d'environnement, p.189
ret	Registre de retour des fonctions, p.189
sp	Registre du pointeur de pile, p.189

wk	Registre de l'espace de travail, p.189
$\models p, p' \rhd l, l' : R^{\top}$	Jugement de programmes, p.187
$\Diamond$	Modalité « sera valable dans le futur », p.188
Perp	Adjoint à gauche de **stateRel** vers **natRel**, p.187
$\mathbf{Perp}_{\mathbf{nat}}$	Adjoint à droite de **natRel** vers **stateRel**, p.187

Index

Il faut se méfier des mots,
Ben (1993)

Résumé

Modalités de ressource et contrôle en logique tensorielle

Cette thèse présente la logique tensorielle, une version primitive de la logique linéaire où la négation involutive est remplacée par une négation tensorielle. Pour illustrer ce point de vue, nous reformulons les espaces cohérents et les espaces de finitude comme deux modèles de logique linéaire obtenus à partir d'un même modèle de logique tensorielle dont on fait varier la négation. La sémantique de la logique tensorielle est pour nous avant tout catégorique, construite autour des notions de catégorie de dialogue et de modalité de ressource. Nous en donnons un modèle inspiré des jeux de Conway où tous les connecteurs, en particulier les modalités de ressource, sont interprétées de manière non dégénérée. Afin de construire ces modalités de ressource de façon plus automatique, nous développons un cadre pour le calcul des algèbres libres d'une T-théorie enrichie. Cette construction, basée sur la notion d'équipement en distributeurs, repose sur deux propriétés : l'une de nature combinatoire, l'opéradicité ; l'autre de nature algébrique, la complétude algébrique. Nous présentons ensuite un modèle de jeux équipé d'une trace et d'une notion de multiparenthésage. Le contrôle obtenu par le multiparenthésage est alors vu comme une gestion des ressources. Nous utilisons ce modèle pour interpréter une langage avec références d'ordre supérieur. Nous nous tournons enfin vers des sémantiques de plus bas niveau. Dans un premier temps, nous étudions la structure multicatégorique induite par une catégorie de dialogue. Cela nous amène à définir les multicatégories de contrôle. Dans un second temps, nous formalisons en Coq une propriété de sûreté par le typage d'un compilateur vers un langage assembleur. Cette formalisation repose sur la définition d'une sémantique relationnelle des états de la mémoire dont la structure est inspirée des catégories de dialogue.

Mots clés : Logique tensorielle – Modalités de ressource – Sémantique des jeux – Équipement en distributeur – Bicatégorie des algèbres relâchées– Multiparenthésage – Multicatégorie de contrôle – Sûreté d'un compilateur par sémantique des types

---

Abstract

Resource modalities and control in tensorial logic

This thesis presents tensorial logic, a primitive version of linear logic where involutive negation is replaced by tensorial negation. As an illustration, we reformulate coherent spaces and finiteness spaces as two different models of linear logic obtained from the same model of tensorial logic by changing the negation. Tensorial logic semantics is, from our point of view, categorical and built on the notions of dialogue category and resource modalities. We provide a mild extension of Conway games that models tensorial logic and where all connectors, and in particular resource modalities, are interpreted in non-degenerate fashion. In order to construct resource modalities more automatically, a framework for computing the free algebras of an enriched T-theory is developed. This construction, based on the notion of proarrow equipment, relies on two properties: a combinatorial one, operadicity; and an algebraic one, algebraic completeness. Next, a game model equipped with a trace operator and a notion of multibracketing is presented. The control obtained from multibracketing is seen as a resource policy. This model is used to interpret a language with higher order references. Finally, we consider lower level semantics. We begin with studying the multicategorical structure induced by a dialogue category; this leads us to define control multicategories. We then formalize a semantic type safety for a compiler (assembly code) in Coq. This formalization depends upon the definition of a relational semantics for memory states whose structure is inspired by dialogue categories.

Keywords : Tensorial logic – Resource modalities – Game semantics – Proarrow equipment – Bicategory of lax algebras – Multibracketing – Control multicategory – semantic type safety for a compiler