# De la Géométrie de l'Interaction à la Syntaxe Transcendantale (Version étendue, révision 2)

Sambo Boris ENG, supervisé par Thomas Seiller (LoVe, LIPN)

Rapport de stage de M2 MPRI

Ceci n'est pas une référence complète/rigoureuse sur la Syntaxe Transcendantale.

## Le contexte général

La correspondance de Curry-Howard [Cur34 ; How80] établit l'existence d'une intersection entre logique et calcul en remettant en question leurs fondements autant d'un point de vue technique que philosophique. Logique et calcul sont tout d'abord reliés de façon structurelle par une correspondance formule/type et preuve/programme mais aussi de façon dynamique car l'évaluation des programmes se comporte comme une procédure de la logique appelée *élimination des coupures* [Gen35a ; Gen35b].

Dans cette optique, la logique linéaire [Gir87a], principalement initiée par Jean-Yves Girard nous permet d'étudier le contenu opérationnel de la logique : les preuves, traditionnellement définies comme des arbres d'applications de règles de déduction deviennent des *réseaux de preuve* exhibant une structure plus précise des preuves. Le programme de « Géométrie de l'Interaction » [Gir89b ; Gir89a ; Gir88 ; Gir95 ; Gir06 ; Gir11 ; Gir13a] succédant à la logique linéaire propose de se libérer du format des réseaux pour étudier ce qui les caractérise mathématiquement. C'est un modèle de la logique linéaire qui diffère des autres par la considération d'une dynamique d'exécution alors que les modèles dénotationnels vont se concentrer sur l'aspect observationnel.

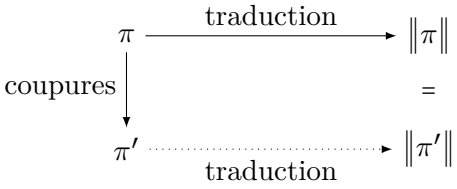



Figure 1 – Modèle dénotationnel

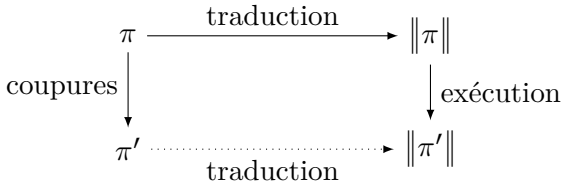



Figure 2 – Géométrie de l'Interaction

Les modèles dénotationnels (figure 1) aplatissent l'interprétation d'une preuve et son évaluation par élimination des coupures car ils représentent en un certain sens la même entité (du point de vue subjectif). Le modèle de géométrie de l'interaction (figure 2), en prenant en compte la présence d'une notion d'exécution permet une analyse fine des propriétés calculatoires de la logique qui sont précieuses pour les études de complexité.

En partant de ces travaux, Girard propose de reconstruire la logique en plaçant l'exécution comme primitive, ce qui débouche sur le programme de « Syntaxe Transcendantale » [Gir13a ; Gir13b ; Gir17 ; Gir16b ; Gir16c] qui étend la géométrie de l'interaction.

## Le problème étudié

Les principaux articles écrits par Girard ne posent que le plan architectural du projet sans développement technique précis. Le sujet du stage était de formaliser ce programme de Syntaxe Transcendantale. Il a fallu établir clairement les définitions, rédiger les théorèmes et les prouver.

Son intérêt technique repose sur une inversion méthodologique : au lieu d'explorer les propriétés de la logique définie comme telle, on introduit un vaste espace calculatoire d'où la logique émerge. Cette émergence repose sur une dualité entre les entités de notre espace qui leur permettent d'interagir par une procédure calculatoire. Cette approche pourrait entre autre permettre de donner un contenu calculatoire à la logique du premier ordre [Gir16c] (qui n'est pas capturée par la correspondance de Curry-Howard et qui a donc un statut purement logique) et donner une relecture des résultats de complexité descriptive en plus de d'autres applications à la complexité à rapprocher à ceux des flots [Bag14 ; BP01 ; AB14 ; ABS16], des graphes d'interaction [Sei18 ; Sei13].

## La contribution proposée

Le stage a apporté un premier pas dans la formalisation de la Syntaxe Transcendantale. Le modèle de calcul introduit par le programme (étoiles et constellations) a été décrit formellement et rapproché des modèles de flots [Gir95 ; Bag14] et des graphes d'interaction [Sei12b]. En particulier, le cheminement de la géométrie de l'interaction à la conception de la syntaxe transcendantale est décrit. Comme premier pas, un modèle du fragment multiplicatif de la logique linéaire (MLL) est donné.

La démarche scientifique avait un fort penchant expérimental : il était nécessaire de construire nous-même les outils à utiliser, produisant parfois des échecs dûs à une formalisation mal fondée. Nous sommes partis d'une généralisation du modèle de Girard et avons étudié ses propriétés calculatoires afin de déterminer les contraintes nécessaires à imposer pour avoir un calcul cohérent et adéquat pour interpréter les phénomènes logiques.

## Les arguments en faveur de sa validité

La formalisation repose sur une architecture générale donnée par le programme de géométrie de l'interaction et reste donc libre mais guidée par la volonté d'avoir certaines propriétés majeures postulant une certaine forme de cohérence calculatoire (confluence, associativité de l'élimination des coupures) et les propriétés nécessaires pour pouvoir reconstruire le fragment MLL de la logique linéaire.

## Le bilan et les perspectives

Ce travail apporte une ouverture aux derniers développements de la géométrie de l'interaction qui disposent de très peu d'écrits mais constitue aussi une première introduction technique au programme de Syntaxe Transcendantale.

L'objectif suivant serait de modéliser des fragments plus riches de la logique linéaire (notamment le fragment additif MALL de la logique linéaire qui est sujet à des problèmes techniques) dans l'espoir de mieux les comprendre. Une contribution intéressante serait de donner un modèle de la logique du premier ordre (comme esquissée par Girard [Gir16c]) avec la complexité en ligne de mire. Il serait aussi intéressant de trouver un modèle mathématique des étoiles et des constellations qu'on pourrait comparer aux graphages [Sei17].

**Remerciements**

Je tiens à remercier mon encadrant Thomas Seiller pour avoir supervisé mon stage et pour tout le temps qu'il m'a accordé. Je remercie également Tito (Nguyễn Lê Thành Dũng) pour l'aide qu'il m'a apporté et Damiano Mazza pour m'avoir aidé pour ma soutenance de stage.

# 1 Architecture

Nous commençons par présenter la base et la position du projet de Syntaxe Transcendantale dans l'évolution du domaine de la logique mathématique. Nous prenons racine dans la *théorie de la démonstration* qui est une branche de la logique plaçant la preuve mathématique à la position d'objet d'étude à part entière.

## 1.1 Traditions

Traditionellement, on considère que le raisonnement est régit par des *lois logiques* dont les formulations sont multiples. On fait le choix d'utiliser le *calcul des séquents* de Gentzen [Gen35a ; Gen35a].

Un séquent est de la forme $A_1, ..., A_n \vdash B_1, ..., B_k$ et énonce que l'*une* (disjonction $\vee$) des conclusions $B_{1\leq i\leq k}$ est prouvable sous *toutes* (conjonction $\wedge$) les hypothèses $A_1, ..., A_n$. Les entités $A_1, ..., A_n, B_1, ..., B_k$ sont des *formules* qui encodent des faits qui sont qualifiés de *vrais* ou *faux*. Une règle d'inférence est de la forme $\frac{\Gamma'\vdash\Delta'}{\Gamma\vdash\Delta}$ où $\Gamma, \Gamma', \Delta, \Delta'$ sont des multi-ensembles de formules.

La théorie de la démonstration se concentre sur le mécanisme calculatoire des *systèmes de déduction* c'est-à-dire la production de preuves mathématiques par application successive de règles fixées. Par exemple, on a les règles suivantes :

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ \wedge_L \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'}\ \wedge_R \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ \Rightarrow_R$$

permettent de construire la preuve suivante de l'implication $\vdash (A \wedge B) \Rightarrow (B \wedge A)$ :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\ \text{ax} \quad \dfrac{}{A \vdash A}\ \text{ax}}{A, B \vdash B \wedge A}\ \wedge_R}{A \wedge B \vdash B \wedge A}\ \wedge_L}{\vdash (A \wedge B) \Rightarrow (B \wedge A)}\ \Rightarrow_R$$

La règle $(ax)$ pour "axiome" connecte une occurrence d'une formule $A$ en hypothèse et une autre en conclusion. La règle $(\Rightarrow)$ introduit une hypothèse et la règle $(\wedge_R)$ nous demande de prouver $A$ et $B$ en distribuant les hypothèses et conclusions.

## 1.2 Vers une morphologie de la preuve

En 1986, Jean-Yves Girard introduit la *logique linéaire* [Gir87a] non pas comme un nouveau système logique mais comme un affinement de la logique traditionnelle. L'appelation « linéaire » provient du fait que ce système consomme exactement *une fois* chaque hypothèse ce qui nous permet de l'intérpréter comme une logique de gestion des ressources, des actions ou encore des communications [Gir89b].

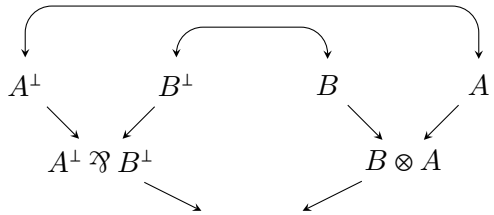
En particulier, il révèle une opérationnalité interne à la logique par une décomposition des connecteurs usuels :

| $\wedge$ | $\vee$ | $A \Rightarrow B$ |
|---|---|---|
| $\otimes, \&$ | $\oplus, \parr$ | $!A \multimap B$ |

où $A \multimap B$ est une implication linéaire plus primitive qui consomme son hypothèse et $!A$ représente un stock arbitraire d'hypothèses $A$, c'est-à-dire qu'on spécifie explicitement que $A \Rightarrow B$ peut potentiellement utiliser plusieurs fois son hypothèse (ou pas du tout) : on obtient une décomposition de $A \Rightarrow B$ en $!A \multimap B$.

En travaillant avec la logique linéaire, Girard remarque que la linéarité dévoile une structure interne des preuves. En particulier, la négation linéaire étant involutive ($A^{\perp\perp} = A$), les règles $\frac{\Gamma, A^\perp \vdash \Delta}{\Gamma \vdash A, \Delta}$ et $\frac{\Gamma \vdash A^\perp, \Delta}{\Gamma A, \vdash \Delta}$ nous permettent de transformer toutes les hypothèses en conclusion et les règles sont vues comme des moyens de connecter les formules. De plus, l'ordre d'application des règles est souvent inessentiel.

Pour exhiber cette structure interne des preuves, Girard introduit une syntaxe alternative : les *réseaux de preuve* prenant la forme d'une sorte de câblage entre des formules. Par exemple, on pourrait avoir le réseau de preuve suivant pour $(A \otimes B) \multimap (B \otimes A)$ qui est traduit en $(A^\perp \parr B^\perp) \parr (B \otimes A)$ par la traduction de l'implication $A \multimap B = A^\perp \parr B$ et la loi de Morgan $(A \otimes B)^\perp = A^\perp \parr B^\perp$ :




Les réseaux de preuve forment un support pour une étude de la logique, de ses mécanismes et de ses phénomènes à travers un point de vue particulièrement riche et fin.

### 1.3 La dynamique de la logique

Les travaux de Gentzen [Gen35a ; Gen35b] révèlent l'existence d'une dynamique dans la logique par le « théorème d'élimination des coupures ».

La règle de *coupure* formulée par :

version classique : $\dfrac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$ cut  version unilatérale : $\dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}$ cut

représente l'usage d'un lemme $A$ en mathématique (qu'on prouve puis utilise) afin de continuer notre preuve. De façon plus terre-à-terre, on a simplement connecté deux preuves par anihilation de deux occurrences opposées/duales de $A$.

Le théorème d'élimination des coupures nous dit qu'il est possible d'éliminer l'usage des règles de coupure et qu'on peut donc démontrer sans usage de lemmes. Ce n'est pas si étonnant si on considère que l'on peut faire de la programmation sans fonctions auxiliaires, de façon irréfléchie et explicite.

Ces travaux ayant été introduits pour des soucis de cohérence de l'arithmétique, le théorème d'élimination des coupures a été complètement sous-estimé au temps de Gentzen : il donne une procédure d'évaluation des preuves qu'on peut apparenter à l'exécution des programmes en informatique et prend un statut fondamental à travers la correspondance de Curry-Howard.

C'est dans l'étude de cette procédure d'élimination des coupures que se fonde le programme de « géométrie de l'interaction » pour étudier ce qu'on appellera maintenant la *dynamique de la logique*. La logique n'est plus primitive mais sujet d'étude : on parle de mathématique de la logique.

# 2 Géométrie de l'Interaction

Le programme de Géométrie de l'Interaction propose une reconstruction de la logique avec les ingrédients suivants : (1) une entité généralisant les preuves (2) une notion de dualité entre ces entités (3) une notion d'exécution permettant de faire interagir deux entités duales. Une telle reconstruction nous permet de faire émerger les opérations logiques à travers leur potentiel d'interaction calculatoire. Ce programme porte aussi la volonté de libérer la logique des contraintes des formats qu'on peut lui imposer : on se place entre syntaxe et sémantique.

## 2.1 Calcul des séquents de MLL

Pour simplifier les choses au maximum, nous travaillerons sur le fragment multiplicatif[1] de la logique linéaire appelé MLL [Gir87b ; DR89] (pour *multiplicative linear logic*) avec les formules définies par :

$$A, B \quad := \quad a \mid A^\perp \mid A \otimes B \mid A \parr B$$

où $\otimes$ est appelé *tenseur*, $\parr$ est appelé *par* et $a$ est une formule minimale appelée *atome*. On peut reconstituer l'implication linéaire par $A \multimap B := A^\perp \parr B$. On dispose des règles suivantes du calcul des séquents linéaire :

$$\frac{}{\vdash A, A^\perp}\ ax \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr$$

où l'ordre des formules n'importe pas dans un séquent (car considéré comme un multi-ensemble). Conformément aux lois de De Morgan qui sont aussi prouvables dans le cadre linéaire, on a

$$(A \otimes B)^\perp = A^\perp \parr B^\perp \qquad (A \parr B)^\perp = A^\perp \otimes B^\perp \qquad A = A^{\perp\perp}$$

qui nous permettent de ne considérer la négation que sur les atomes et de pouvoir travailler sur une grammaire plus simple des formules :

$$A, B \quad := \quad a \mid a^\perp \mid A \otimes B \mid A \parr B$$

où $a$ est un atome dans un ensemble dénombrable d'atomes qu'on a fixé.

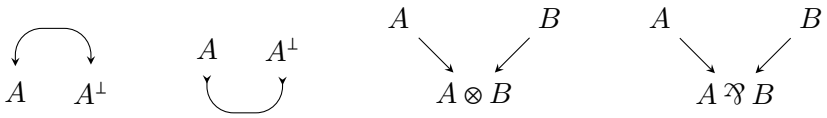
## 2.2 Réseaux de preuve de MLL

En se libérant de la syntaxe arborescente, on obtient des objets plus généraux que les preuves, qu'on peut voir comme des entités calculatoires (du même genre que les programmes) : les structures de preuve.

**Definition 1 (Structure de preuve).**

> Une structure de preuve est un hypergraphe orienté avec des formules comme sommets et les hyperarêtes (arêtes avec plusieurs sources et cibles) suivantes :

1. Note aux spécialistes : on fait le choix d'éviter de parler des unités $\perp$ et 1

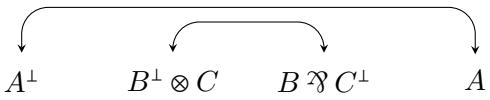



où chaque sommet (formule) est but d'exactement une hyperarête. La première et la seconde hyperarête représentent respectivement l'axiome et la coupure. Les cibles d'hyperarête qui ne sont pas sources sont appelées *conclusions*. Les sources sont les *hypothèses*.

**Definition 2 (Véhicule).**

Le **véhicule** $V(\pi)$ d'une structure de preuve $\pi$ est l'ensemble de ses axiomes.

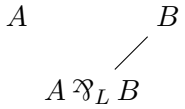
**Exemple.**

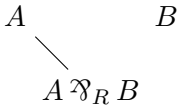



Une conséquence de cette métamorphose des entités logiques est qu'on perd la distinction comportementale entre $\otimes$ et $\parr$ que l'on doit retrouver par un critère. Un tel critère est appelé *critère de correction* et nous dit qu'on a bien formé une preuve "correcte" dans le sens qu'elle peut être rapportée à une preuve du calcul des séquent de MLL.

**Definition 3 (Test).**

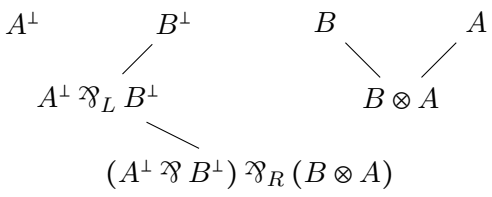
Soit $\pi$ une structure de preuve et $\varphi : \parr(\pi) \longrightarrow \{\mathtt{l}, \mathtt{r}\}$ une sélection gauche/droite pour chaque sommet $\parr$ de $\pi$. Le *test* $\mathtt{test}_\varphi(\pi)$ induit par $\pi$ et $\varphi$ est défini de la façon suivante où $p$ désigne le sommet $\parr$ concerné :

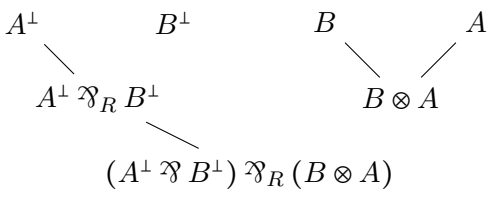
Si $\phi(p) = \mathtt{l}$ alors 


Si $\phi(p) = \mathtt{r}$ alors 


où dans les autres cas, les arêtes deviennent non-orientées et $\parr(\pi)$ est l'ensemble des sommets $\parr$ dans la preuve $\pi$. Les tests, destinés à être connectés à un véhicule, permettent de définir une preuve par un ensemble véhicule+test qui définit un arbre (pour se ramener à la forme arborescente du calcul des séquents).

**Exemple.** On a les deux tests suivants pour le réseau de preuve de $(A \otimes B) \multimap (B \otimes A)$ introduit précédemment (mais il en existe $2^2 = 4$) :




**Definition 4 (Gabarit).**

Le **gabarit** $G(\pi)$ d'une structure de preuve $\pi$ est l'ensemble de tous les tests $\texttt{test}_\varphi(\pi)$ pour tous les choix possibles de $\varphi$. Il y en a exactement $|\{l, r\}|^{|\parr(\pi)|} = 2^{|\parr(\pi)|}$.

**Definition 5 (Critère de correction de Danos-Regnier).**

Soit $\pi$ une structure de preuve de MLL. Pour tout test $t \in G(\pi), V(\pi)+t$ est un arbre où $V(\pi)+t$ est $t$ que l'on a complété avec les axiomes de $\pi$.

**Théorème 1 (Séquentialisation).**

Si le critère de correction de Danos-Regnier est satisfait par une structure de preuve $\pi$ de MLL alors $\pi$ est la traduction d'une preuve du calcul des séquents de MLL sachant une définition naturelle du calcul des séquents vers les structures de preuve (qu'on omet ici).

**Preuve.**

On se réfèrera à la thèse de Laurent Regnier [Reg92] qui propose une preuve de séquentialisation avec l'introduction d'un critère de correction à base d'interrupteurs correspondant à ce qu'on appelle *test* dans ce document.

□

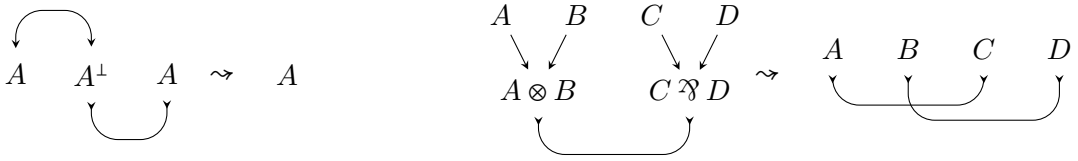
**Definition 6 (Réseau de preuve).**

Un *réseau de preuve* (ou simplement *réseau*) est une structure de preuve qui satisfait le critère de correction de Danos-Regnier.

Une structure de preuve est une entité qui obtient le titre (subjectif) de "réseau de preuve" seulement si elle passe un ensemble de tests avec lesquels elle intéragit. De cette idée, Girard introduit une terminologie d'industrie automobile qu'on a choisi d'adopter : on certifie l'usage d'un véhicule s'il passe un ensemble de tests proposés par une usine. C'est une analogie qui rappelle aussi les *tests unitaires* en programmation.

**Definition 7 (Elimination des coupures).**

On définit une procédure d'élimination des coupures qui va permettre de réduire/évaluer les structures de preuves vers d'autres structures de preuves équivalentes :




## 2.3 Caractérisation algébrique

Nous faisons le choix de partir de l'algèbre $\Lambda^*$ de Girard [Gir89a] pour proposer une caractérisation algébrique des réseaux qui nous mènera aux flots que nous généraliserons avec les étoiles et constellations.

**Definition 8 (Chemin).**

Comme les structures de preuve sont des hypergraphes, on a quelques difficultés pour définir une notion de chemin.

Un **chemin** $\varphi$ dans une structure de preuve de MLL $\pi$ est une séquence $\varphi = v_0 e_1 v_1 ... e_n v_n$ où les $v_i$ sont des sommets de $\pi$ (formules) et les $e_i$ sont soit des hyperarêtes ou des sources/cibles d'hyperarêtes. C'est-à-dire qu'on décrit comment on passe d'un sommet à un autre (soit par un axiome ou une coupure soit une prémisse/conclusion de connecteur binaire). Pour suivre les conventions de la littérature, on écrit les chemins de droite à gauche.

**Definition 9 (Monoïde involutif $L^*$).**

On définit $L^*$ ($\Lambda^*$ chez Girard) comme étant le monoïde involutif généré par $\{\mathtt{l}, \mathtt{r}, 0\}$ sujet aux relations suivantes :

$$0x = x0 = 0 \qquad \mathtt{l}^*\mathtt{r} = \mathtt{r}^*\mathtt{l} = 0 \qquad \mathtt{l}^*\mathtt{l} = \mathtt{r}^*\mathtt{r} = 1$$

L'involution est une opération unaire (plus précisément, un antimorphisme) définie par $0^* = 0$, $(w^*)^* = w$ et $(vw)^* = w^* v^*$.

**Exemple.** on peut avoir $\mathtt{l}\mathtt{r}^*\mathtt{r}\mathtt{r}^* = \mathtt{l}\mathtt{r}^*$ par la relation $\mathtt{r}^*\mathtt{r} = 1$.

**Definition 10 (Pondération).**

On définit une fonction de **poids** $\omega(\varphi)$ sur un chemin $\varphi = v_0 e_1 v_1 ... e_n v_n$ d'une structure de preuve de MLL avec $\omega(v_0 e_1 v_1 ... e_n v_n) = \omega(v_0)\omega(e_1)\omega(v_1)...\omega(e_n)\omega(v_n)$ et :

- $\omega(e_i) = \mathtt{l}$ si $e_i$ est une prémisse gauche d'un connecteur binaire
- $\omega(e_i) = \mathtt{r}$ si $e_i$ est une prémisse droite d'un connecteur binaire
- $\omega(e_i) = 1$ si $e_i$ est un axiome ou une coupure et $v_i \neq v_{i+1}$ (pas de demi-tour)
- $\omega(x) = 0$ sinon

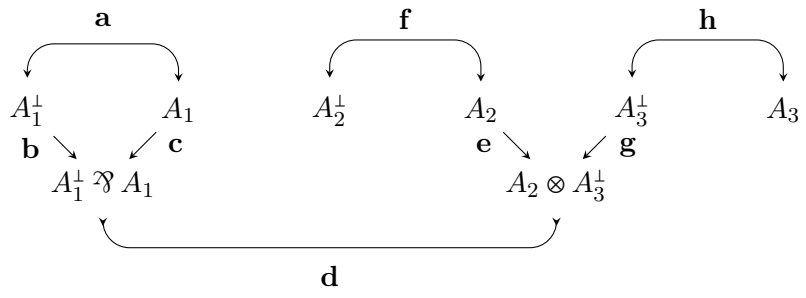
On peut étendre la fonction de poids sur :

- les arêtes inverses par $w(e^R) = \omega(e)^*$
- les chemins par $\omega(e_1 \cdot ... \cdot e_n) = \omega(e_1) \cdot ... \cdot \omega(e_n)$

**Definition 11 (Chemin régulier).**

Un chemin $\varphi$ est **régulier** si $\omega(\varphi) \neq 0$.

L'élément absorbant 0 va représenter la destruction de chemins par l'élimination des coupures. Danos et Regnier [DR95] ont montré une équivalence entre la notion de chemin régulier et de chemin droit *persistant* qui est un chemin qui ne fait pas de demi-tour, ne passe pas d'une prémisse à l'autre et survit à l'élimination des coupures. De tels chemins caractérisent la forme normale du réseau. Illustrons en quoi $L^*$ décrit l'élimination des coupures :



Supposons qu'on écrive les chemins de droite à gauche et que $A_1 = A_2 = A_3 = A$. Pour simplifier on représente les chemins par une séquence d'étiquettes :

— En partant de $A_3$, on a les chemins suivants :

$\omega(fe^R dcab^R dgh) = \mathtt{l}^*\mathtt{rl}^*\mathtt{r} = 0$ $\quad$ $\omega(fe^R dbac^R dgh) = \mathtt{l}^*\mathtt{lr}^*\mathtt{r} = 1$ $\quad$ (vers $a_2^\perp$)

$\omega(hg^R dcab^R dgh) = \mathtt{r}^*\mathtt{rl}^*\mathtt{r} = 0$ $\quad$ $\omega(hg^R dbac^R dgh) = \mathtt{r}^*\mathtt{lr}^*\mathtt{r} = 0$ $\quad$ (vers $a_3$)

— Si on part de $A_2^\perp$, on a les chemins suivants :

$\omega(hg^R dcab^R def) = \mathtt{r}^*\mathtt{rl}^*\mathtt{l} = 1$ $\quad$ $\omega(hg^R dbac^R def) = \mathtt{lr}^*\mathtt{l} = 0$ $\quad$ (vers $a_3$)

$\omega(fe^R dcab^R def) = \mathtt{l}^*\mathtt{rl}^*\mathtt{l} = 0$ $\quad$ $\omega(fe^R dbac^R def) = \mathtt{l}^*\mathtt{lr}^*\mathtt{l} = 0$ $\quad$ (vers $A_2^\perp$)

Comme chemins persistants, on obtient un chemin de $A_3$ à $A_2^\perp$ et un autre de $A_2^\perp$ à $A_3$. Cette pondération est équivalente à celle d'un axiome entre $A_3$ et $A_2^\perp$. Conclusion : on a procédé à une élimination des coupures sans réécriture, simplement par *analyse statique* du réseau et simplifications d'expressions algébriques. C'est un point essentiel pour parler de complexité (en particulier en espace).

**Definition 12 (Exécution).**

On définit **l'exécution** d'un réseau $\pi$ comme la somme du poids de ses chemins persistants :

$$\mathtt{Ex}(\pi) = \sum_{\varphi \in \mathtt{persistent}(\pi)} \omega(\varphi)$$

où $\mathtt{persistent}(\pi)$ est l'ensemble des chemins persistants de $\pi$ c'est-à-dire que si $\varphi \in \mathtt{persistent}(\pi)$ et $\pi$ se réduit en $\pi'$ alors $\varphi \in \mathtt{persistent}(\mathcal{R}')$. Cette exécution représente l'évaluation de $\pi$ par élimination des coupures c'est-à-dire que si $\pi$ se réduit en $\pi'$ alors $\mathtt{Ex}(\pi) = \mathtt{Ex}(\pi')$.

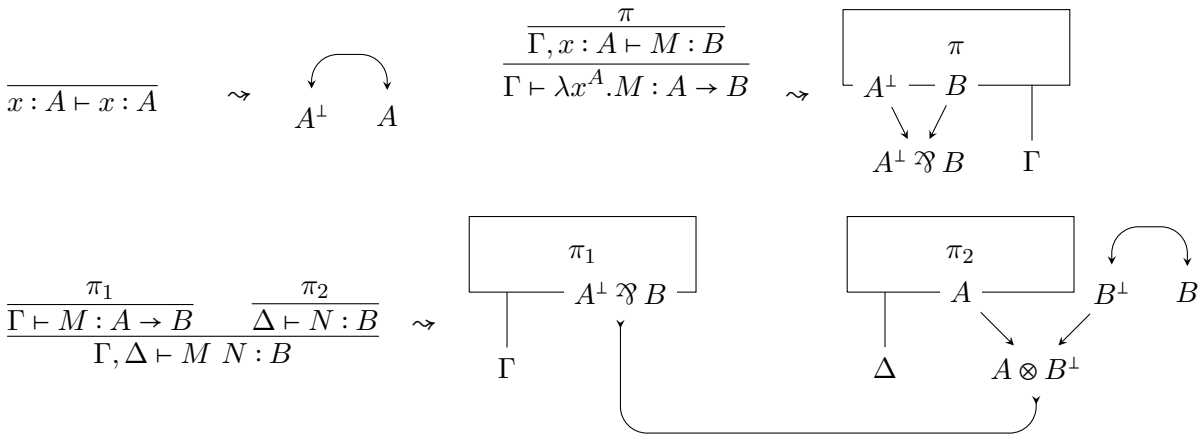
## 2.4 Interlude : réseaux et lambda-calcul

Pour revenir à des notions familières de l'informatique, on peut modéliser le $\lambda$-calcul linéaire et son execution par les réseaux et la géométrie de l'interaction dans MLL. Si on fixe un ensemble de variables et de types de base, la grammaire du $\lambda$-calcul simplement typé est donnée par :

$$M, N \quad ::= \quad x \mid \lambda x^A.M \mid M\ N \qquad\qquad A, B \quad ::= \quad a \mid A \to B$$

Le $\lambda$-calcul linéaire est un modèle de calcul fonctionnel où les arguments sont utilisés exactement une fois et sont donc en quelque sorte consommés à l'usage. L'exécution est donnée par la $\beta$-réduction définie par $(\lambda x.M)N \to_\beta M[x := N]$ où $M[x := N]$ est $M$ où on a remplacé l'unique occurrence de $x$ par $N$.

On peut traduire les règles de typage du $\lambda$-calcul simplement typé dans les réseaux [Reg92] :



La géométrie de l'interaction fournit en fait un moyen de lire statiquement un réseau représentant un $\lambda$-terme pour en cacluler la forme normale correspondante. En particulier, cela a un intérêt précieux en complexité [Maz17 ; Maz15 ; MT15].

Le réseau précédent sur lequel nous avons calculé l'élimination des coupures par géométrie de l'interaction représente le terme linéaire $y : a \vdash (\lambda x^a.x)\ y$ qui se réduit en $x : a$ qui est bien traduit en un axiome entre $a$ et $a^\perp$. Ce que nous venons d'illustrer s'étend au cadre non-linéaire et donc au $\lambda$-calcul simplement typé et même pur [Dan90] en interprétant le fragment exponentiel de la logique linéaire.

## 2.5 Eléments d'unification

Durant l'étude algébrique de Girard sur la géométrie de l'interaction, une des versions s'est stabilisée vers une algèbre d'unification [Gir95]. Girard montre que l'unification, qu'on peut attribuer à Jacques Herbrand [Her30] permet de donner un modèle de géométrie de l'interaction pour la logique linéaire complète.

On définit une signature $\mathcal{S} = \texttt{Vars} \uplus \texttt{Func}$ où
- $\texttt{Vars} = \{w, x, y, z, ...\}$ est un ensemble dénombrable de symboles de variables.
- $\texttt{Func} = \{f^{(n_1)}, g^{(n_2)}, h^{(n_3)}, ...\}$ est un ensemble dénombrable de symboles de fonctions accompagnés d'une arité.

Tous ces ensembles peuvent être considérés finis puisqu'on peut se restreindre aux symboles dont on a besoin pour un contexte donné. Les $\mathcal{S}$-termes (ou simplement *termes* en l'absence d'ambiguïté) sont générés par la grammaire suivante :

$$t, u, v, w \quad ::= x \quad | \quad f(t_1, ..., t_n) \qquad\qquad (\mathcal{S}\text{-termes})$$

où $x \in \texttt{Vars}$ et $f^{(n)} \in \texttt{Func}$ pour un certain $n \in \mathbb{N}$. On note $\texttt{vars}(t)$ l'ensemble des variables d'un $\mathcal{S}$-terme $t$ et un terme $t$ est clos si $\texttt{vars}(t) = \varnothing$. Exemples : $x, f(x), h(f(x), z)$. Non-exemples : $f(g), x(f), y(f(x))$.

**Definition 13 (Substitution).**

Une **substitution** est une fonction $\theta = \{\!|x_1 \mapsto t_1, ..., x_n \mapsto t_n|\!\}$ associant une valeur aux variables et définie par son action sur les termes. Si on définit les ensembles suivants pour $\theta$ :
- $\texttt{dom}(\theta) = \{x_1, ..., x_n\}$
- $\texttt{img}(\theta) = \{t_1, ..., t_n\}$

alors on peut définir l'action d'une substitution par :

$$\theta(x_i) = t_i \qquad \theta(y)_{y \in \texttt{Vars} \setminus dom(\theta)} = y \qquad \theta(f(u_1, ..., u_k)) = f(\theta u_1, ..., \theta u_k)$$

**Definition 14 (Composition de substitutions).**

La **composition** de deux substitutions $\theta_1$ et $\theta_2$ est définie par $(\theta_1 \circ \theta_2)t = \theta_1(\theta_2 t)$. On la considère associative à droite par défaut c'est-à-dire que $\theta_1 \circ \theta_2 \circ \theta_3$ est une notation pour $\theta_1 \circ (\theta_2 \circ \theta_3)$.

**Definition 15 (Unification).**

Deux termes $t$ et $u$ sont **unifiables** quand il existe une substitution $\theta$ telle que $\theta t = \theta u$. Une telle substitution est aussi appelée **unificateur** de $t$ et $u$.

**Definition 16 (Renommage).**

Un **renommage** $\alpha$ est une permutation sur l'ensemble des variables qu'on représente sous la forme d'une substitution $\alpha$ telle que $\texttt{dom}(\alpha) = \texttt{img}(\alpha) = \texttt{Vars}$. Pour tout terme $t$ et renommage $\alpha$, $\alpha t$ est appelé **renommage** de $t$.

**Definition 17 (Equivalence par renommage).**

Deux substitutions $\theta_1, \theta_2$ sont $\alpha$-**équivalents** noté $\theta_1 \approx_\alpha \theta_2$ si et seulement si $\theta_1 = \alpha\theta_2$ ou $\theta_2 = \alpha\theta_1$ pour un certain renommage $\alpha$.

Comme il est toujours possible de renommer, on considère les substitutions à renommage près et on considère que $=$ est confondu avec $\approx_\alpha$.

**Definition 18 (Unificateur principal).**

La composition induit un pré-ordre sur les substitutions : $\theta \preceq \psi \iff \exists\theta'.\psi = \theta' \circ \theta$.

Un **unificateur principal** $\theta$ pour deux termes $t$ et $u$ est un unificateur de $t$ et $u$ minimal pour l'ordre défini ci-dessus. On note $\theta \sim \psi$ si et seulement si $\theta \preceq \psi$ et $\psi \preceq \theta$.

**Definition 19 (Problème d'unification).**

Un **problème d'unification** est un ensemble $P = \{t_1 \doteq u_1, ..., t_n \doteq u_n\}$ de paires de termes. Le problème $P$ est **soluble** s'il existe un unificateur $\theta$ appelé **solution** de $P$ tel que $\texttt{dom}(\theta) \subseteq \texttt{vars}(P)$ qui est un unificateur pour toutes les équations de $P$. La solution est **principale** si c'est un unificateur principal pour chacune des équations de $P$.

On peut appliquer une substitution à un problème d'unification :

$$\theta\{x_1 \doteq t_1, ..., x_n \doteq t_n\} = \{\theta x_1 \doteq \theta t_1, ..., \theta x_n \doteq \theta t_n\}$$

**Definition 20 (Forme résolue).**

Un problème $P$ est en **forme résolue** s'il est de la forme $\{x_1 \doteq t_1, ..., x_n \doteq t_n\}$ où $\{x_1, ..., x_n\} \cap \bigcup_{j=1}^{n} \texttt{fv}(t_j) = \varnothing$. Sa **substitution associée** est définie par $\vec{P} = \{\!|x_1 \mapsto t_1, ..., x_n \mapsto t_n|\!\}$.

On suppose qu'on utilise un algorithme d'unification non-déterministe comme celui décrit par Martelli et Montanari [MM82] pour produire un unificateur principal noté $\mathcal{S}\{\}t \doteq u$. Il est égal à $\bot$ s'il n'est pas défini et peut être étendu à des problèmes d'unification : $\mathcal{S}\{\}\{t_1 \doteq u_1, ..., t_n \doteq u_n\}$ et un unificateur pour $t_1 \doteq u_1, ..., t_n \doteq u_n$.

**Corollaire 2 (Existence d'un unificateur principal).**

Tout problème d'unification $P$ soluble a une solution principale. [Preuve page 28]

**Corollaire 3 (Unicité de l'unificateur principal).**

Si un problème $P$ est soluble avec un unificateur principal $\theta$ en forme résolue, alors $\theta$ est unique à renommage près. [Preuve page 29]

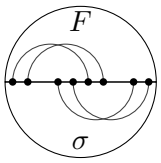



FIGURE 3

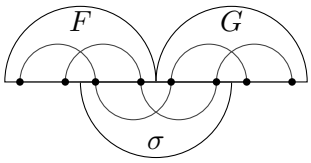



FIGURE 4

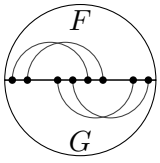



FIGURE 5

## 2.6 Flots et câblages

Faisons des remarques préliminaires sur l'essence de l'élimination des coupures par calcul des chemins persistants. Il y a plusieurs points de vues possibles.

- On remarque que les chemins dans un réseau changent de direction lors du passage sur une coupure ou un axiome. De plus, les conclusions des axiomes peuvent être vus comme des sortes de zones d'interactions. Si on représente un axiome par un demi-cercle $F$ et les coupures par un demi-cercle $\sigma$ (figure 3), on peut voir le calcul des chemins persistants comme le calcul des chemins alternants entre $F$ et $\sigma$ qui sont maximaux (pour avoir une exécution complète). On peut aussi voir $F$ et $\sigma$ comme des permutations sur les conclusions des atomes dont on fait le produit.
- On pourrait avoir un point de vue où les coupures relient deux sous-parties disjointes du véhicule qu'on note $F + G$ (figure 4). Une arête de coupure dans $\sigma$ identifie en fait deux points d'interaction (conclusion des axiomes) de $F$ et $G$. On peut simplifier ce modèle en connectant directement les points d'interactions de $F$ et $G$ en suivant $\sigma$ (figure 5). L'élimination des coupures revient donc à calculer l'ensemble des chemins alternants entre $F$ et $G$. On peut maintenant limiter l'essence calculatoire d'une preuve à son véhicule qu'on connecte à d'autres véhicules.

Les flots nous donnent un modèle permettant, en quelque sorte, de désynchroniser l'élimination des coupures en partant du modèle de la figure 5 en plus de pouvoir être étendu à des fragments plus riches de la logique linéaire (additif et exponentiel). On va représenter les *points d'interaction* ou *lieux* par des termes ouverts sur une signature $\mathcal{S}$ fixée et leur connectivité par l'unification de termes. Ce sont des sortes d'adresses (similairement aux poids dans $L^*$) décrivant des positions dans le réseau. L'élimination des coupures sera une sorte de résolution d'adresse.

C'est un modèle de calcul général qui peut être vu comme une forme de résolution du premier ordre sur des clauses unaires.

**Definition 21 (Flots).**

> Un **flot** est une paire ordonnée $t \leftharpoonup u$ de termes du premier ordre telle que $\mathtt{var}(t) = \mathtt{var}(u)$. On considère les flots égaux à renommage près. Exemples : $x \leftharpoonup x$ et $f(g(y), y) \leftharpoonup g(y)$. Le flot $id = x \leftharpoonup x$ est appelé **flot identité**.

**Definition 22 (Symétrisation).**

> La **symétrisation** d'un flot $t \leftharpoonup u$ est le flot $(t \leftharpoonup u) + (u \leftharpoonup t)$ noté $t \rightleftharpoons u$ ou $u \rightleftharpoons t$.

**Definition 23 (Produit).**

> Le **produit** de deux flots $f = u \leftharpoonup v$ et $g = t \leftharpoonup w$ est défini par
>
> $$(u \leftharpoonup v)(t \leftharpoonup w) := \theta u \leftharpoonup \theta w$$

où $f$ et $g$ sont renommés de sorte à ne pas partager de variable et $\theta$ est l'unificateur principal de $v$ et $t$. S'il n'existe pas de tel unificateur, le produit n'est pas défini et $fg = \bot$.

**Exemple.** On a le produit suivant : $(f(x) \leftharpoonup x)(g(x) \leftharpoonup x) = (f(x) \leftharpoonup x)(g(y) \leftharpoonup y) = f(g(y)) \leftharpoonup y$. Le flot de droite est renommé puis on applique $\theta = \{x \mapsto g(y)\}$ sur les extrémités formant le flot résultat.

**Definition 24 (Câblage de flots).**

Un **câblage de flots** ou simplement *câblage* quand il n'y a pas d'ambiguité est un ensemble de flots. On peut étendre le produit de flots à celui de produit de deux câblages $F$ et $G$ par :

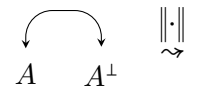
$$FG = \{fg \mid f \in F, g \in G, fg \neq \bot\}$$

Par définition on peut définir la puissance $F^n$ avec $n \in \mathbb{N}$ comme :

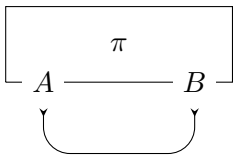
$$F^0 = \{id\} \qquad\qquad F^{n+1} = F^n F$$

**Definition 25 (Traduction vers les flots).**

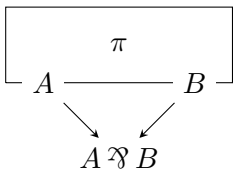
On reconstruit le véhicule des *réseaux de preuve* (et non des structures de preuves) avec des symétrisations de flots où chaque côté représente un lieu.



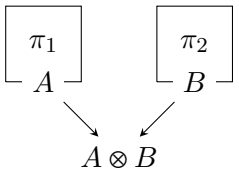

$\overset{\|\cdot\|}{\rightsquigarrow}$ $(p_A \cdot x \rightleftharpoons p_{A^\perp} \cdot x)$




$\overset{\|\cdot\|}{\rightsquigarrow}$ $\|\pi\| + (p_A \cdot x \rightleftharpoons p_B \cdot x)$




$\overset{\|\cdot\|}{\rightsquigarrow}$ $\|\pi\|\,[p_A(x) := p_{A⅋B} \cdot \mathtt{l} \cdot x, p_B(x) := p_{A⅋B} \cdot \mathtt{r} \cdot x]$




$\overset{\|\cdot\|}{\rightsquigarrow}$ $\|\pi_1\|\,[p_A(x) := p_{A\otimes B} \cdot \mathtt{l} \cdot x] + \|\pi_2\|\,[p_B(x) := p_{A\otimes B} \cdot \mathtt{r} \cdot x]$

L'expression $[t := u]$ met à jour le terme $t$ en le remplaçant par $u$. On utilise l'opérateur $f + g$ comme raccourci pour $\{f\} \cup \{g\}$. Le symbole $\cdot$ est un symbole binaire de $\mathcal{S}$ associatif à droite c'est-à-dire qu'on a $x \cdot y \cdot z = x \cdot (y \cdot z)$.

Il existe trois points de vues de l'exécution selon les figures 3, 4 et 5.

**Definition 26 (Exécution I).**

L'**exécution** (figure 3) calcul des chemins alternants entre un véhicule $F$ et une coupure $\sigma$. Elle est définie par :

$$\mathtt{Ex}_f^1(F,\sigma) = (1-\sigma^2)F(\sum_{k=0}^{\infty}(\sigma F)^k)(1-\sigma^2) = (1-\sigma^2)F(1-\sigma F)^{-1}(1-\sigma^2)$$

qui représente les chemins de compositions de flots alternants entre $F$ et l'ensemble des coupures $\sigma$.

Les termes $(1-\sigma^2)$ suppriment les compositions non maximales, c'est-à-dire celles dont le terme source et/ou le terme but appartient à la coupure. La notation est empruntée à la formulation de l'execution dans les algèbres d'opérateurs, où $\sigma$ est une symétrie partielle représentant la coupure. Dans ce cas, $\sigma^2$ est donc une projection sur un sous-espace (le lieu de la coupure, ici l'intersection des lieux de $F$ et $G$), et $1-\sigma^2$ est une projection sur le sous-espace complémentaire. La composition à droite et à gauche par cette projection a donc cet effet de restreindre le domaine et le codomaine de l'opérateur calculé au sous-espace complémentaire de la coupure, ce qui correspond ici à supprimer les compositions non maximales.

**Definition 27 (Exécution II).**

L'**exécution** (figure 4) peut aussi être vu comme la connexion de deux véhicules $F$ et $G$ et le calcul de leurs chemins de compositions alternants intercalés par la coupure $\sigma$. Dans ce cas elle est définie par :

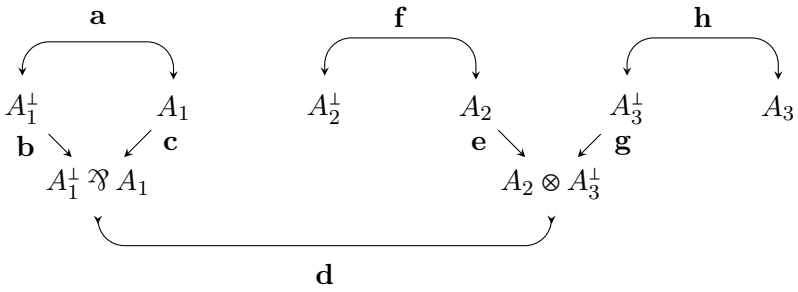
$$\mathtt{Ex}_f^2(F,G,\sigma) = (1-\sigma^2)(E_{F,F}+E_{F,G}+E_{G,G}+E_{G,F})(1-\sigma^2)$$

où $E_{F,F} = \sum_{k=0}^{\infty} F(GF)^k, E_{F,G} = \sum_{k=0}^{\infty} G(GF)^k, E_{G,G} = \sum_{k=0}^{\infty}(GF)^k G, E_{G,F} = \sum_{k=0}^{\infty} F(GF)^k G$ qui représentent des chemins alternants commençant et finissant par $F$ ou $G$.

**Definition 28 (Exécution III).**

L'**exécution** $\mathtt{Ex}_f(F,G)$ (figure 5) identifie des points d'interactions par $\sigma$ et connecte directement les véhicules $F$ et $G$ en calculant leurs chemins de compositions maximaux alternants. La définition est similaire à $\mathtt{Ex}_f^2(F,G,\sigma)$ mais $\sigma$ est considéré implicite.

Prenons le réseau de preuve suivant :




Le premier point de vue de l'exécution de la figure 3 ou figure 4 va produire l'adressage suivant :

$$(p_{A_1^\perp \parr A_1}\cdot l\cdot x) \rightleftharpoons (p_{A_1^\perp \parr A_1}\cdot r\cdot x) \quad (p_{A_2^\perp}\cdot x) \rightleftharpoons (p_{A_2\otimes A_3^\perp}\cdot l\cdot x) \qquad (p_{A_2\otimes A_3^\perp}\cdot r\cdot x) \rightleftharpoons (p_{A_3}\cdot x)$$

$$(p_{A_1^\perp \parr A_1}\cdot x) \rightleftharpoons (p_{A_2\otimes A_3^\perp}\cdot x)$$

On remarque que chaque terme du flot inférieur, représentant la coupure peut se connecter à deux termes des axiomes. Cela représente le fait qu'il y a quatre chemins possibles entre le tenseur et le par. Cependant, si par exemple on connecte $p_{A_1^\perp \parr A_1} \cdot x$ à la prémisse gauche de $A_1^\perp \parr A_1$ c'est-à-dire $p_{A_1^\perp \parr A_1} \cdot l \cdot x$, la variable $x$ de la coupure est remplacée par $l \cdot x$, forçant le terme $p_{A_2 \otimes A_3^\perp} \cdot x$ de la coupure à se connecter à la prémisse gauche de $A_2 \otimes A_3^\perp$ c'est-à-dire $p_{A_2 \otimes A_3^\perp} \cdot l \cdot x$.

Cet exemple montre que l'usage de variable et de l'unification est essentiel pour représenter le comportement de l'élimination des coupures sans parler de réécriture d'hypergraphe.

# 3 Modèle stellaire

Girard introduit la syntaxe transcendantale comme successeur de la géométrie de l'interaction avec un espace d'entités formé des *étoiles et constellations* [Gir13a ; Gir13b ; Gir17]. Ce modèle part du constat que les flots sont bidirectionnels $t \rightleftharpoons u$ dans l'interprétation de la logique et ne sont adéquats que pour modéliser la partie calculatoire des preuves (véhicule).

Ce nouveau programme propose de généraliser le modèle de flots pour prendre en considération le gabarit (modélisation des tests) vu comme typage afin d'étendre l'espace des entités logique, avoir des constructions riches de types/formules et reconstituer d'autres phénomènes logiques (entre autre une explication des individus du premier ordre). De plus, par sa portée philosophique, ce programme propose de reconstruire la logique sans référence nécessaire à l'infini afin de donner une explication finitaire de la logique.

## 3.1 Etoiles et constellations

On introduit le modèle de calcul d'étoiles et de constellation qui va nous permettre de reconstruire la plupart des phénomènes et entités logiques comme les structures de preuve, l'élimination des coupures, les formules mais aussi le critère de correction de Danos-Regnier.

**Definition 29 (Couleur).**

> Pour une signature $\mathcal{S}$ fixée, on fixe un ensemble de symboles de fonctions unaires appelés **couleurs** utilisés comme préfixe des $\mathcal{S}$-termes.

**Exemple.** Le terme $\alpha(t)$ est de couleur $\alpha$ si $\alpha$ est une couleur. On utilisera la notation $\alpha.t$. Les couleurs existent par paires duales. On notera $\overline{\alpha}$, la couleur duale de $\alpha$.

**Definition 30 (Rayons).**

> Pour une signature $\mathcal{S}$ fixée, un **rayon** est un $\mathcal{S}$-terme préfixé par une couleur. Si un rayon n'est pas préfixé par une couleur, il est qualifié d'*incolore*. Deux rayons sont **duaux** s'ils sont de couleur duales.

**Definition 31 (Étoiles).**

> Une **étoile** désignée par $\sigma$ est un multi-ensemble non vide et fini $[\![t_1, ..., t_n]\!]$ de rayons.
> - pour toute substitution $\theta$, on a $\theta\sigma = [\![\theta t_1, ..., \theta t_n]\!]$ comme l'application de $\theta$ à tous les rayons de $\sigma$.
> - $\texttt{colors}(\sigma)$ comme l'ensemble des couleurs qui apparaissent dans $\sigma$.
> - La coloration de $\sigma = [\![t_1, ..., t_n]\!]$ par $\alpha$ notée $\alpha.\sigma$ représente $\sigma$ où tous les rayons incolores sont préfixés par la couleur $\alpha$.
>
> Ce sont des regroupements de termes avec des variables connectées entre elles. Exemples :

$[\![\alpha.x, \overline{\alpha}.x, \beta.x]\!]$ et $[\![\alpha.f(x), h(f(x)), \beta.g(x)]\!]$

**Definition 32 ($\alpha$-équivalence).**

Deux étoiles $\sigma_1$ et $\sigma_2$ sont $\alpha$-**équivalentes** noté $\sigma_1 \approx_\alpha \sigma_2$ quand il existe un renommage $\alpha$ tel que $\sigma_1 = \alpha\sigma_2$ ou $\sigma_2 = \alpha\sigma_1$. Par commodité, on considérera toujours que l'égalité entre étoiles et $\approx_\alpha$ coïncident.

**Definition 33 (Constellations).**

Une **constellation** est un multi-ensemble d'étoiles $\sigma_1 + ... + \sigma_n$ qu'on désigne par les symboles $\Gamma, \Delta$.

- La coloration de $\Sigma$ par $\alpha$ est notée $\alpha.\Sigma = \alpha.\sigma_1 + ... + \alpha.\sigma_n$.
- Chaque étoile doit avoir un ensemble de variables disjoint des autres (en d'autres termes, les variables sont locales aux étoiles). Lorsque que ce n'est plus le cas, on peut toujours renommer les variables de sorte à former une constellation.

## 3.2 Diagrammes et normalisation

L'évaluation des constellations consiste à former toutes les possibilités de connexions entre les rayons duaux et à les réduire par fusion des étoiles et unification des rayons connectés. Chaque réduction produit une étoile qui fera partie d'une constellation résultat. Cette idée sera décrite formellement par la suite.

Ce modèle a la spécificité d'avoir un mécanisme presque identique à la *méthode de résolution* [Rob+65] utilisée en programmation logique. Une clause $A_1, ..., A_n \vdash B$ peut être traduite en une étoile $[\![\overline{\alpha}.A_1, ..., \overline{\alpha}.A_n, \alpha.B]\!]$ (avec $B$ incolore si c'est un but qui n'est pas destiné à disparaître). L'évaluation que nous considérons est une variante de la méthode de résolution.

**Definition 34 (Graphe d'unifiabilité).**

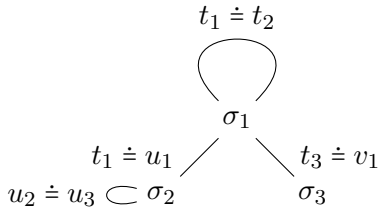
Le **graphe d'unifiabilité** $\mathcal{U}(\Sigma)$ d'une constellation $\Sigma = \sigma_1 + ... + \sigma_n$ est un multigraphe orienté où :

- l'ensemble des sommets est l'ensemble induit par $\Sigma$. On les étiquete avec les étoiles correspondantes.
- on a une arête entre deux étoiles pour chacun de leurs rayons duaux unifiables. On étiquete les arêtes par l'équation $t \doteq u$ si elle relie deux rayons $\alpha.t$ et $\overline{\alpha}.u$ pour une couleur $\alpha$.

Cela représente l'ensemble des connexions potentielles dans une constellation.

**Exemple.** Si on considère tous les rayons duaux comme étant unifiables, on a :

$$\begin{array}{lcll} \sigma_1 & = & [\![\alpha.t_1, \overline{\alpha}.t_2, \gamma.t_3]\!] & + \\ \sigma_2 & = & [\![\overline{\alpha}.u_1, \beta.u_2, \overline{\beta}.u_3]\!] & + \\ \sigma_3 & = & [\![\overline{\gamma}.v_1]\!] & \end{array} \qquad \text{produit}$$




**Definition 35 (Diagramme).**

Soit $\Sigma$ une constellation. Un **diagramme** de base $\Sigma$ est un arbre $D$ tel qu'il existe un morphisme de multigraphe $\varphi : D \longrightarrow \mathcal{U}(\Sigma)$.

On note $\texttt{links}(D)$ l'ensemble des équations associées aux arêtes de $D$ qu'on appellera **liaisons** et $\texttt{free}(D)$ l'ensemble des **rayons libres** : les rayons de $\Sigma$ qui ne sont pas dans $\texttt{links}(D)$ qu'on notera `links` et `free` lorsqu'il n'y a pas d'ambiguïtés.

**Exemple.** On peut avoir les diagrammes suivants pour $\Sigma_1 = [\![\alpha.x, \beta.x]\!] + [\![\overline{\alpha}.f(y), \overline{\beta}.g(y)]\!]$ :

$$D_{\Sigma_1} = [\![\overbrace{\alpha.x, \beta.x}]\!] + [\![\overline{\alpha}.f(y), \overline{\beta}.g(y)]\!] \qquad \text{et} \qquad D'_{\Sigma_1} = [\![\alpha.x, \overbrace{\beta.x}]\!] + [\![\overline{\alpha}.f(y), \overline{\beta}.g(y)]\!]$$

Connecter tous les rayons duaux ne forme pas un arbre et n'est donc pas un diagramme valide. On a $\texttt{links}(D_{\Sigma_1}) = \{x \doteq f(y)\}$, $\texttt{free}(D_{\Sigma_1}) = \{\beta.x, \overline{\beta}.g(y)\}$, $\texttt{links}(D'_{\Sigma_1}) = \{x \doteq g(y)\}$ et $\texttt{free}(D'_{\Sigma_1}) = \{\alpha.x, \overline{\alpha}.f(y)\}$.

**Definition 36 (Diagramme saturé).**

Soit $\Sigma$ une constellation. Un **diagramme saturé** de base $\Sigma$ est un diagramme $D$ de base $\Sigma$ tel que pour tout arbre $D' \supset D$ avec pour sommets les éléments de $\Sigma$, il n'existe pas de morphisme $\varphi' : D' \longrightarrow \mathcal{U}(\Sigma)$ tel que $\varphi'_{|D} = \varphi$ où $\varphi'_{|D}$ est $\varphi'$ dont le domaine est restreint à $D$.

C'est l'analogue des chemins maximaux dans un réseau. Il faut tout connecter jusqu'à qu'il ne reste plus de connexions possibles.

**Exemple.** Sur la constellation $\Sigma$ précédente on a pas de diagramme saturé car on peut toujours utiliser d'autres occurrences pour créer des diagrammes plus grands. Cependant, on peut avoir le diagramme saturé suivant :

$$D_{\Sigma_2} = [\![\overbrace{\alpha.x, \underbrace{\beta.x}}]\!] + [\![\overline{\alpha}.f(y), y]\!] + [\![\underbrace{\overline{\beta}}.f(z), \gamma.g(z), z]\!]$$

Il n'y a pas d'autres liaisons possibles même en ajoutant de nouvelles occurrences d'étoiles.

On évalue un diagramme par la notion *de fusion* qui consiste à fusionner/contracter les étoiles par unification et anihilation des rayons connectés.

**Definition 37 (Réduction par fusion).**

Soit $D$ un diagramme de liaisons `links` et de rayons libres `free`. La **réduction par fusion** est une contraction d'arbre dans $D$ notée $\rightsquigarrow$. Deux sommets $\sigma, \sigma'$ reliés par $t \doteq t'$ sont contractés de la manière suivante :

1. On calcule un unificateur principal $\theta := \mathcal{S}\{t \doteq t'\}$ pour $t$ et $t'$
2. On efface les rayons $t$ et $t'$ pour obtenir deux étoiles $\delta$ et $\delta'$
3. La nouvelle étoile formée est $\theta(\delta \cup \delta')$

La **procédure de réduction par fusion** une contraction d'arbre appliquée jusqu'à l'obtention d'une étoile qu'on peut noter $D \rightsquigarrow^* \sigma$ où $\sigma$ est une étoile.

**Exemple.** La réduction par fusion sur $D_{\Sigma_2}$ se déroule ainsi :

$$[\![\overbrace{\alpha.x, \underbrace{\beta.x}}]\!] + [\![\overline{\alpha}.f(y), y]\!] + [\![\underbrace{\overline{\beta}}.f(z), \gamma.g(z), z]\!]$$

$$\rightsquigarrow_{x \doteq f(y)} [\![\beta.f(y), y]\!] + [\![\overline{\beta}.f(z), \gamma.g(z), z]\!] \rightsquigarrow_{f(y) \doteq f(z)} [\![z, \gamma.g(z), z]\!]$$

On peut imaginer une forme alternative de réduction des diagrammes qui fournit une résolution de toutes les équations :

Remarques : la procédure de réduction par fusion termine nécessairement sur une étoile.

**Definition 38 (Actualisation).**

> Soit $D$ un diagramme de liaisons `links` et de `free`. L'**actualisation** de $D$ est $\Downarrow D := \mathcal{S}\{\texttt{links}\}(\texttt{free})$. On résout les équations sous-jacentes et on applique l'unificateur résultant sur l'ensemble des rayons libres (non connectés).

Remarque : l'actualisation est entièrement définie par l'ensemble des liaisons et l'ensemble des rayons libres.

**Exemple.** La constellation $D_{\Sigma_2}$ produit le problème $P = \{x \doteq f(y), x \doteq f(z)\}$ avec $\texttt{free}(D_{\Sigma_2}) = [\![y, \gamma.g(z), z]\!]$. La résolution de $P$ donne $\theta = \{x \mapsto f(y), y \mapsto z\}$ et on obtient l'actualisation $\theta\texttt{free} = [\![z, \gamma.g(z), z]\!]$.

Ces deux procédures de réduction sont équivalentes. On travaillera avec l'actualisation par simplicité.

**Théorème 4 (Équivalence entre actualisation et réduction par fusion).**

> Pour tout diagramme correct $D$, on a $D \rightsquigarrow^* \Downarrow D$. [Preuve page **??**]

**Definition 39 (Diagramme correct et incorrect).**

> L'actualisation d'un diagramme peut échouer ou non selon la présence de conflits d'unification.
> - si $\Downarrow D$ n'est pas défini (une unification échoue durant l'actualisation) alors $D$ est qualifié de diagramme **incorrect**.
> - sinon $\Downarrow D$ produit une étoile appelée **étoile résiduelle** et $D$ est qualifié de diagramme **correct**.

**Exemple.** Pour la constellation $\Sigma_1$ définie plus haut, les diagrammes à 1 liaison sont corrects mais l'unique diagramme de 2 liaisons est incorrect.

On cherche maintenant à évaluer une constellation entière en calculant tous les diagrammes saturés possibles chacun produisant une étoile faisant partie d'une constellation résultat.

**Definition 40 (Normalisation).**

> La **normalisation** ou l'**exécution** de $\Sigma$ est définie par :
>
> $$\texttt{Ex}^\star(\Sigma) := \Downarrow \{D \mid D \text{ est correct de base } \Sigma \text{ et saturé}\}$$

**Definition 41 (Propriété de normalisation forte).**

> La propriété de **normalisation forte** garantissant la terminaison de la normalisation et la production d'un résultat est donnée par les conditions suivantes : (1) Le nombre de diagrammes saturés corrects doit être fini (2) Aucun diagramme clos sans rayons libre n'est correct (pour éviter la production de l'étoile vide). Une constellation qui satisfait cette propriété est qualifée

de **fortement normalisable**.

**Théorème 5 (Confluence de la normalisation).**

Les deux procédures de normalisation suivantes sont équivalentes :
— On normalise les couleurs une par une
— On normalise toutes les couleurs

Ce résultat nous assure qu'on a un espace de calcul cohérent où l'ordre d'évaluation n'a pas de conséquence sur le résultat. C'est point majeur qui nous assurera aussi l'associativité de l'élimination des coupures permettant l'existence de modèles catégoriques.

# 4 Reconstruction de MLL

Maintenant qu'on a défini un modèle de calcul, on montre que l'on peut reconstruire les entités et les phénomènes logiques de façon assez naturelle.

## 4.1 Reconstruction des preuves (usage)

La syntaxe transcendantale prend comme véhicule le contenu calculatoire des preuves (axiomes et coupures).

**Definition 42 (Adressage des axiomes).**

Soit $\pi$ une structure de preuve. Considérons la forêt $\pi'$ de conclusions $\vdash C_1, ..., C_n$ obtenir en retirant les axiomes et les coupures. On définit une fonction $\mathtt{loc}_A(\pi)$ qui calcule l'adresse de l'axiome $A$ dans $\pi$ :

$$\mathtt{loc}_X\begin{pmatrix}\pi_1 & & \pi_n\\ \vdots & \cdots & \vdots\\ C_1 & & C_n\end{pmatrix} \quad \rightsquigarrow \quad p_{C_k}\big(\mathtt{loc}'_A(\pi_k)\big) \qquad \text{si } X \in C_k$$

$$\mathtt{loc}'_X(Y) \quad \rightsquigarrow \quad x \in \mathtt{Vars} \qquad \text{si } Y = X$$

$$\mathtt{loc}'_X\left(\begin{array}{ccc}\overset{\pi_1}{\vdots} & & \overset{\pi_2}{\vdots}\\ A & & B\\ & \searrow\quad\swarrow & \\ & A * B & \end{array}\right) \quad \rightsquigarrow \quad \begin{cases}\mathtt{l}\cdot\mathtt{loc}'_X(\pi_1) & \text{si } X \in A\\ \mathtt{r}\cdot\mathtt{loc}'_X(\pi_2) & \text{si } X \in B\end{cases} \qquad \text{où } * \in \{\otimes, \parr\}$$

**Definition 43 (Véhicule stellaire).**

Soit $\pi$ une structure de preuve de MLL de véhicule $V = \{(A_1, B_1), ..., (A_n, B_n)\}$. Le véhicule stellaire $V^\star$ de $\pi$ est :

$$V^\star := \sum_{k=1}^{n} [\![\mathtt{loc}_{A_k}(\pi), \mathtt{loc}_{B_k}(\pi)]\!]$$

C'est-à-dire qu'on traduit $\pi$ par des étoiles binaires qui représentent l'adresse des formules conclusions des axiomes.

**Definition 44 (Traduction des coupures).**

Soit $\pi$ une structure de preuve de MLL de coupures $C = \{(A_1, B_1), ..., (A_n, B_n)\}$. L'ensemble est coupures est traduit par :

$$C^\star = \sum_{k=1}^{n} [\![ \frac{\overline{\alpha}.p_{A_k}(x), \overline{\alpha}.p_{B_k}(x)}{} ]\!]$$

où la notation de fraction est purement esthétique.

**Definition 45 (Traduction des structures de preuves).**

Soit $\pi$ une structure de preuve de MLL de véhicule $V$ et de coupures $C$. Cette structure est traduite par :

$$\pi^\star = \alpha.(V^\star) \cup C^\star$$

On remarque que la paire de couleur $(\alpha, \overline{\alpha})$ permet de simuler une distinction entrée/sortie.

**Definition 46 (Élimination des coupures).**

Soit $\pi$ une structure de preuve de MLL. L'élimination des coupures de $\pi$ est donnée par l'exécution de sa traduction dans les étoiles c'est-à-dire $\mathtt{Ex}^\star(\pi^\star)$.

## 4.2 Reconstruction des formules (usage)

Les constellations permettent de placer de nombreuses entités logiques dans le même espace. Il existe une dualité entre les constellations que l'on définit par la convergence de leur interaction :

**Definition 47 (Orthogonalité).**

Soient $\Sigma_1$ et $\Sigma_2$ deux constellations. On a $\Sigma_1 \perp \Sigma_2$ si $\mathtt{Ex}^\star(\Sigma_1 \cup \Sigma_2)$ a la propriété de normalisation forte. On définit l'**orthogonal** d'un ensemble de constellations par : $A^\perp = \{\Sigma \mid \forall \Sigma' \in A. \Sigma \perp \Sigma'\}$.

On peut aussi reconstruire la notion de formule logique ou type en restant dans le même espace (étoiles et constellations). C'est une construction qui voit les types comme une classification ou certification d'objets calculatoires de la même manière qu'un type peut représenter une classe de preuves et une formule une classe de programmes.

**Definition 48 (Type).**

Un **type** est un ensemble de constellations $\mathbf{A}$ tel que il existe un ensemble de constellations $B$ tel que $\mathbf{A} = B^\perp$.

C'est-à-dire qu'un ensemble de constellations est un type s'il a toujours un ensemble dual avec qui il peut interagir correctement. Cela fait apparaître l'idée de *test d'usage* : une constellation fait partie d'un type si elle passe tous les tests de convergence par interaction avec les constellations du type dual. Autrement dit, un type est constitué d'entités testables par un type dual. On peut avoir une définition alternative par une double orthogonalité (biorthogonal dans la littérature) qui explicite l'idée d'un espace "clos par dualité".

**Théorème 6 (Clôture par biorthogonal).**

Soit $\mathbf{A}$ un ensemble de constellation. On a $\mathbf{A} = \mathbf{A}^{\perp\perp}$ si et seulement si $\mathbf{A}$ est un type. [Preuve page 35]

Les connecteurs comme le tenseur $\otimes$ et le par $\parr$ sont vus comme de simples constructions sur les types en se basant sur les équivalences habituelles de la logique linéaire par lois de De Morgan (par exemple $A \parr B = (A^\perp \otimes B^\perp)^\perp$). On retrouve de telles constructions dans la définition d'un modèle dénotationel de MLL à partir des étoiles et des constellations.

**Théorème 7 (Modèle dénotationnel de MLL).**

Le modèle stellaire forme un modèle dénotationnel de MLL.

## 4.3 Reconstruction du critère de correction (usine)

Deux paires de couleur $\alpha$ et $\beta$ sont suffisantes pour reconstruire le critère de correction de Danos-Regnier. On a travaillé avec la paire de couleur $(\alpha, \overline{\alpha})$ représentant la partie calculatoire d'une preuve. On introduit maintenant la paire de couleur $(\beta, \overline{\beta})$ représentant la partie logique d'une preuve permettant de juger sa correction.

**Definition 49 (Test).**

Soit $\pi$ une structure de preuve de véhicule $V$ et de gabarit $G$ et $t \in G$ un de ses tests. Le **test stellaire** $\|t\|$ associé est donné par la constellation produite par la traduction de tous ses sommets :

$$A, A^\perp : [\![\frac{\overline{\alpha}.\mathtt{loc}_A(\pi)}{\beta.q_A(x)}]\!] + [\![\frac{\overline{\alpha}.\mathtt{loc}_{A^\perp}(\pi)}{\beta.q_{A^\perp}(x)}]\!]$$

$$A \otimes B : [\![\frac{\overline{\beta}.q_A(x) \quad \overline{\beta}.q_B(x)}{\beta.q_{A\otimes B}(x)}]\!] \quad \text{On fait un usage essentiel des étoiles : on a 3 rayons liés.}$$

$$A \parr_L B : [\![\frac{\overline{\beta}.q_A(x)}{}]\!] + [\![\frac{\overline{\beta}.q_B(x)}{\beta.q_{A\parr B}(x)}]\!] \qquad A \parr_R B : [\![\frac{\overline{\beta}.q_A(x)}{\beta.q_{A\parr B}(x)}]\!] + [\![\frac{\overline{\beta}.q_B(x)}{}]\!]$$

$$\text{Coupure} : [\![\frac{\overline{\beta}.q_A(x), \overline{\beta}.q_{A^\perp}(x)}{}]\!] \qquad \text{Conclusion} : [\![\frac{\overline{\beta}.q_A(x)}{p_A(x)}]\!] \text{ où } A \text{ est une conclusion}$$

**Definition 50 (Gabarit stellaire).**

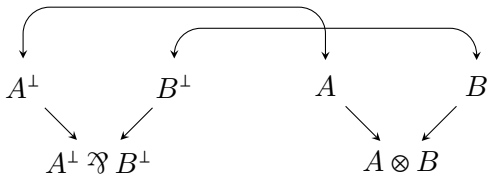
Soit $\pi$ une structure de preuve de véhicule $G$. Le **gabarit stellaire** est défini par l'ensemble de tous les tests stellaires pour $\pi$ c'est-à-dire $G^\star = \{\|t\| \mid t \in G\}$.

On reconstruit le critère de correction de Danos-Regnier grâce aux étoiles et constellations.

**Definition 51 (Critère de correction stellaire).**

Soit $\pi$ une structure de preuve de MLL de véhicule $V$ et de conclusions $\{A_1, ..., A_n\}$. Pour tout test stellaire $t \in G$, on a $\mathtt{Ex}^\star(\alpha.V^\star \cup \|t\|) = [\![p_{A_1}(x_1), ..., p_{A_n}(x_n)]\!]$ pour certains $x_1, ..., x_n$.

Illustrons avec le réseau de preuve suivant et sa traduction associée avec le test $\parr_L$ :

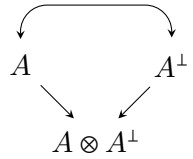



$$\alpha.\mathcal{V} = \begin{array}{l}[\![\alpha.p_{A\otimes B}(l\cdot x), \alpha.p_{A^\perp \parr B^\perp}(l\cdot x)]\!] + \\ [\![\alpha.p_{A\otimes B}(r\cdot x), \alpha.p_{A^\perp \parr B^\perp}(r\cdot x)]\!]\end{array}$$

$$\mathcal{T}_L = \begin{array}{llll}[\![\frac{\overline{\alpha}.p_{A^\perp \parr B^\perp}(\mathtt{l}\cdot x)}{\beta.q_{A^\perp}(x)}]\!] + [\![\frac{\overline{\alpha}.p_{A^\perp \parr B^\perp}(\mathtt{r}\cdot x)}{\beta.q_{B^\perp}(x)}]\!] + [\![\frac{\overline{\alpha}.p_{A\otimes B}(x)}{\beta.q_A(\mathtt{l}\cdot x)}]\!] + [\![\frac{\overline{\alpha}.p_{A\otimes B}(x)}{\beta.q_B(\mathtt{r}\cdot x)}]\!] + & \text{(ax)} & [\![\frac{\overline{\beta}.q_A(x), \overline{\beta}.q_B(x)}{\beta.q_{A\otimes B}(x)}]\!] + & (\otimes) \\ [\![\frac{\overline{\beta}.q_{A^\perp}(x)}{\beta.q_{A^\perp \parr B^\perp}(x)}]\!] + [\![\frac{\overline{\beta}.q_{A^\perp}(x)}{}]\!] + & (\parr_L) & [\![\frac{\overline{\beta}.q_{A^\perp \parr B^\perp}(x)}{p_{A^\perp \parr B^\perp}(x)}]\!] + [\![\frac{\overline{\beta}.q_{A\otimes B}(x)}{p_{A\otimes B}(x)}]\!] & \text{(conclusions)}\end{array}$$

On remarque que si on connecte toutes les étoiles de façon évidente, toutes les couleurs duales s'annulent pour ne laisser que les conclusions. On peut aussi essayer avec l'autre test ($\parr_R$). Les graphes d'unifiabilités produits étant acycliques (grâce à l'étoile $\parr_L$), le nombre de diagrammes est fini (donc le nombre de diagrammes corrects aussi).

Cependant, si on veut simuler une structure qui ne satisfait pas le critère de correction, il faut libéraliser la définition de véhicule (on l'omet ici). Un tenseur sur les conclusions d'un axiome ne relie pas deux réseaux disjoints. On devrait donc avoir le véhicule $[\![p_{A\otimes A^\perp}(l\cdot x), \alpha.p_{A\otimes A^\perp}(r\cdot x)]\!]$.




$$\alpha.\mathcal{V} = [\![\alpha.p_{A\otimes A^\perp}(l\cdot x), \alpha.p_{A\otimes A^\perp}(r\cdot x)]\!]$$
$$\mathcal{T} = [\![\frac{\overline{\alpha}.p_{A\otimes A^\perp}(\mathtt{l}\cdot x)}{\beta.q_A(x)}]\!] + [\![\frac{\overline{\alpha}.p_{A\otimes A^\perp}(\mathtt{r}\cdot x)}{\beta.q_{A^\perp}(x)}]\!] + [\![\frac{\overline{\beta}.q_A(x),\overline{\beta}.q_{A^\perp}(x)}{\beta.q_{A\otimes A^\perp}(x)}]\!] + [\![\frac{\overline{\beta}.q_{A\otimes A^\perp}(x)}{p_{A\otimes A^\perp}(x)}]\!]$$

On a un cycle entre l'unique étoile du véhicule $[\![\alpha.p_{A\otimes A^\perp}(l\cdot x), \alpha.p_{A\otimes A^\perp}(r\cdot x)]\!]$ et l'étoile du tenseur $[\![\frac{\overline{\beta}.q_A(x),\overline{\beta}.q_{A^\perp}(x)}{\beta.q_{A\otimes A^\perp}(x)}]\!]$. Par le théorème de cycle parfait, on peut produire une infinité de diagrammes corrects. De plus, les diagrammes formés sont saturés puisque le cycle concerne des étoiles binaires à l'exception de l'étoile du tenseur où le rayon conclusion ne peut être lié à aucune autre étoile du cycle.

**Corollaire 8 (Simulation de correction).**

> Soit $\pi$ une structure de preuve de MLL. On a $\pi$ est correct si et seulement si $\pi$ satisfait le critère de correction stellaire.

On peut énoncer la correction de notre simulation par le théorème suivant que l'on ne prouve pas dans ce document :

**Théorème 9 (Simulation de correction).**

> Soit $\pi$ une structure de preuve de MLL. On a $\pi$ est correct si et seulement si $\pi$ satisfait le critère de correction stellaire.

Nous avons obtenu une reconstruction des preuves de MLL par la juxtaposition d'un véhicule et d'un gabarit $\mathcal{V}+\mathcal{G}$ où chacun peut posséder des coupures (et donc possède une notion d'évaluation). Le véhicule représente la partie calculatoire de la preuve et le gabarit la cohérence (typage) du véhicule.

# Références


[AB14] Clément Aubert et Marc Bagnol. "Unification and logarithmic space". In : *Rewriting and Typed Lambda Calculi*. Springer, 2014, p. 77-92.

[ABS16] Clément Aubert, Marc Bagnol et Thomas Seiller. "Unary resolution : Characterizing ptime". In : *International Conference on Foundations of Software Science and Computation Structures*. Springer. 2016, p. 373-389.

[Bag14] Marc Bagnol. "On the resolution semiring". Thèse de doct. Aix-Marseille Université, 2014.

[BP01] Patrick Baillot et Marco Pedicini. "Elementary complexity and geometry of interaction". In : *Fundamenta Informaticae* 45.1-2 (2001), p. 1-31.

[Cur34] Haskell B Curry. "Functionality in combinatory logic". In : *Proceedings of the National Academy of Sciences of the United States of America* 20.11 (1934), p. 584.

[Dan90] Vincent Danos. “La Logique Linéaire appliquée à l’étude de divers processus de normalisation (principalement du Lambda-calcul)”. Thèse de doct. Paris 7, 1990.

[DR89] Vincent Danos et Laurent Regnier. “The structure of multiplicatives”. In : *Archive for Mathematical logic* 28.3 (1989), p. 181-203.

[DR95] Vincent Danos et Laurent Regnier. “Proof-nets and the Hilbert space”. In : *London Mathematical Society Lecture Note Series* (1995), p. 307-328.

[Gen35a] Gerhard Gentzen. “Untersuchungen über das logische Schließen. I”. In : *Mathematische zeitschrift* 39.1 (1935), p. 176-210.

[Gen35b] Gerhard Gentzen. “Untersuchungen über das logische Schließen. II”. In : *Mathematische Zeitschrift* 39.1 (1935), p. 405-431.

[Gir01] Jean-Yves Girard. “Locus solum : From the rules of logic to the logic of rules”. In : *Mathematical structures in computer science* 11.3 (2001), p. 301-506.

[Gir06] Jean-Yves Girard. “Geometry of interaction IV : the feedback equation”. In : *Logic Colloquium*. T. 3. 2006, p. 76-117.

[Gir11] Jean-Yves Girard. “Geometry of interaction V : logic in the hyperfinite factor”. In : *Theoretical Computer Science* 412.20 (2011), p. 1860-1883.

[Gir13a] Jean-Yves Girard. “Geometry of interaction VI : a blueprint for transcendental syntax”. In : *preprint* (2013).

[Gir13b] Jean-Yves Girard. “Three lightings of logic (Invited Talk)”. In : *Computer Science Logic 2013 (CSL 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2013.

[Gir16a] Jean-Yves Girard. *Le fantôme de la transparence*. Éditions Allia, 2016.

[Gir16b] Jean-Yves Girard. “Transcendental syntax II : Non-deterministic case”. In : *Logical Methods in Computer Science, to appear* (2016).

[Gir16c] Jean-Yves Girard. “Transcendental syntax III : equality”. In : *Logical Methods in Computer Science* (2016).

[Gir17] Jean-Yves Girard. “Transcendental syntax I : deterministic case”. In : *Mathematical Structures in Computer Science* 27.5 (2017), p. 827-849.

[Gir87a] Jean-Yves Girard. “Linear logic”. In : *Theoretical computer science* 50.1 (1987), p. 1-101.

[Gir87b] Jean-Yves Girard. “Multiplicatives”. In : (1987).

[Gir88] Jean-Yves Girard. “Geometry of interaction 2 : Deadlock-free algorithms”. In : *International Conference on Computer Logic*. Springer. 1988, p. 76-93.

[Gir89a] Jean-Yves Girard. “Geometry of interaction 1 : Interpretation of System F”. In : *Studies in Logic and the Foundations of Mathematics*. T. 127. Elsevier, 1989, p. 221-260.

[Gir89b] Jean-Yves Girard. “Towards a geometry of interaction”. In : *Contemporary Mathematics* 92.69-108 (1989), p. 6.

[Gir95] Jean-Yves Girard. “Geometry of interaction III : accommodating the additives”. In : *London Mathematical Society Lecture Note Series* (1995), p. 329-389.

[Her30] Jacques Herbrand. “Recherches sur la théorie de la démonstration”. Thèse de doct. Université de Paris, 1930.

[How80] William A Howard. “The formulae-as-types notion of construction”. In : *To HB Curry : essays on combinatory logic, lambda calculus and formalism* 44 (1980), p. 479-490.

[HS03] Martin Hyland et Andrea Schalk. “Glueing and orthogonality for models of linear logic”. In : *Theoretical computer science* 294.1-2 (2003), p. 183-231.

[Maz15] Damiano Mazza. “Simple parsimonious types and logarithmic space”. In : *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2015.

[Maz17] Damiano Mazza. “Polyadic Approximations in Logic and Computation”. In : (2017).

[MM82] Alberto Martelli et Ugo Montanari. “An efficient unification algorithm”. In : *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.2 (1982), p. 258-282.

[MT15] Damiano Mazza et Kazushige Terui. “Parsimonious types and non-uniform computation”. In : *International Colloquium on Automata, Languages, and Programming*. Springer. 2015, p. 350-361.

[Reg92] Laurent Regnier. “Lambda-calcul et réseaux”. Thèse de doct. Paris 7, 1992.

[Rob+65] John Alan Robinson et al. “A machine-oriented logic based on the resolution principle”. In : *Journal of the ACM* 12.1 (1965), p. 23-41.

[Sei12a] Thomas Seiller. “Interaction graphs : multiplicatives”. In : *Annals of Pure and Applied Logic* 163.12 (2012), p. 1808-1837.

[Sei12b] Thomas Seiller. “Logique dans le facteur hyperfini : géometrie de l’interaction et complexité”. Thèse de doct. Aix-Marseille Université, 2012.

[Sei13] Thomas Seiller. “Interaction graphs : Exponentials”. In : *arXiv preprint arXiv :1312.1094* (2013).

[Sei17] Thomas Seiller. “Interaction graphs : Graphings”. In : *Annals of Pure and Applied Logic* 168.2 (2017), p. 278-320.

[Sei18] Thomas Seiller. “Interaction Graphs : Non-Deterministic Automata”. In : *ACM Transactions on Computational Logic (TOCL)* 19.3 (2018), p. 21.

[Wit09] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 2009.

# Annexes

Cette partie contient des informations qui ne sont pas strictement nécessaires à la lecture du document. On y inclut entre autre des détails de preuves de lemmes et théorèmes ainsi que des réflexions conceptuelles afin de prendre du recul sur le travail effectué.

# Annexe 1 - Théorie de l'unification

## 5 Théorie de l'unification

On définit une signature $\mathcal{S} = \texttt{Vars} \uplus \texttt{Func}$ où
— $\texttt{Vars} = \{w, x, y, z, ...\}$ est un ensemble dénombrable de symboles de variables.
— $\texttt{Func} = \{f^{(n_1)}, g^{(n_2)}, h^{(n_3)}, ...\}$ est un ensemble dénombrable de symboles de fonctions accompagnés d'une arité.

Tous ces ensembles peuvent être considérés finis puisqu'on peut se restreindre aux symboles dont on a besoin pour un contexte donné. Les $\mathcal{S}$-termes (ou simplement *termes* en l'absence d'ambiguïté) sont générés par la grammaire suivante :

$$t, u, v, w \quad := x \quad | \quad f(t_1, ..., t_n) \qquad\qquad (\mathcal{S}\text{-termes})$$

où $x \in \texttt{Vars}$ et $f^{(n)} \in \texttt{Func}$ pour un certain $n \in \mathbb{N}$. On note $\texttt{vars}(t)$ l'ensemble des variables d'un $\mathcal{S}$-terme $t$ et un terme $t$ est clos si $\texttt{vars}(t) = \varnothing$. Exemples : $x, f(x), h(f(x), z)$. Non-exemples : $f(g), x(f), y(f(x))$.

**Definition 52 (Substitution).**

Une **substitution** est une fonction $\theta = \{\!|x_1 \mapsto t_1, ..., x_n \mapsto t_n|\!\}$ associant une valeur aux variables et définie par son action sur les termes. Si on définit les ensembles suivants pour $\theta$ :
— $\texttt{dom}(\theta) = \{x_1, ..., x_n\}$
— $\texttt{img}(\theta) = \{t_1, ..., t_n\}$
alors on peut définir l'action d'une substitution par :

$$\theta(x_i) = t_i \qquad \theta(y)_{y \in \texttt{Vars} \setminus dom(\theta)} = y \qquad \theta(f(u_1, ..., u_k)) = f(\theta u_1, ..., \theta u_k)$$

**Definition 53 (Composition de substitutions).**

La **composition** de deux substitutions $\theta_1$ et $\theta_2$ est définie par $(\theta_1 \circ \theta_2)t = \theta_1(\theta_2 t)$. On la considère associative à droite par défaut c'est-à-dire que $\theta_1 \circ \theta_2 \circ \theta_3$ est une notation pour $\theta_1 \circ (\theta_2 \circ \theta_3)$.

**Definition 54 (Unification).**

Deux termes $t$ et $u$ sont **unifiables** quand il existe une substitution $\theta$ telle que $\theta t = \theta u$. Une telle substitution est aussi appelée **unificateur** de $t$ et $u$.

**Definition 55 (Renommage).**

Un **renommage** $\alpha$ est une permutation sur l'ensemble des variables qu'on représente sous la forme d'une substitution $\alpha$ telle que $\texttt{dom}(\alpha) = \texttt{img}(\alpha) = \texttt{Vars}$. Pour tout terme $t$ et renommage $\alpha$, $\alpha t$ est appelé **renommage** de $t$.

**Definition 56 (Equivalence par renommage).**

Deux substitutions $\theta_1, \theta_2$ sont $\alpha$-**équivalents** noté $\theta_1 \approx_\alpha \theta_2$ si et seulement si $\theta_1 = \alpha\theta_2$ ou $\theta_2 = \alpha\theta_1$ pour un certain renommage $\alpha$.

Comme il est toujours possible de renommer, on considère les substitutions à renommage près et on considère que $=$ est confondu avec $\approx_\alpha$.

**Definition 57 (Unificateur principal).**

La composition induit un pré-ordre sur les substitutions : $\theta \preceq \psi \iff \exists\theta'.\psi = \theta' \circ \theta$.

Un **unificateur principal** $\theta$ pour deux termes $t$ et $u$ est un unificateur de $t$ et $u$ minimal pour l'ordre défini ci-dessus. On note $\theta \sim \psi$ si et seulement si $\theta \preceq \psi$ et $\psi \preceq \theta$.

**Definition 58 (Problème d'unification).**

Un **problème d'unification** est un ensemble $P = \{t_1 \doteq u_1, ..., t_n \doteq u_n\}$ de paires de termes. Le problème $P$ est **soluble** s'il existe un unificateur $\theta$ appelé **solution** de $P$ tel que $\texttt{dom}(\theta) \subseteq \texttt{vars}(P)$ qui est un unificateur pour toutes les équations de $P$. La solution est **principale** si c'est un unificateur principal pour chacune des équations de $P$.

On peut appliquer une substitution à un problème d'unification :

$$\theta\{x_1 \doteq t_1, ..., x_n \doteq t_n\} = \{\theta x_1 \doteq \theta t_1, ..., \theta x_n \doteq \theta t_n\}$$

**Definition 59 (Forme résolue).**

Un problème $P$ est en **forme résolue** s'il est de la forme $\{x_1 \doteq t_1, ..., x_n \doteq t_n\}$ où $\{x_1, ..., x_n\} \cap \bigcup_{j=1}^{n} \texttt{fv}(t_j) = \varnothing$. Sa **substitution associée** est définie par $\vec{P} = \{\!|x_1 \mapsto t_1, ..., x_n \mapsto t_n|\!\}$.

**Théorème 10 (Idempotence de la forme résolue).**

Si $\theta$ est en forme résolue alors $\theta \circ \theta = \theta$.

**Preuve.**

Par définition, $\theta = \{\!|x_1 \mapsto t_1, ..., x_n \mapsto t_n|\!\}$ tel que tous les $x_i$ sont distincts et n'apparaissent dans aucun $t_j$. Donc une application de $\theta$ sur un terme $t$ fait disparaître toutes les occurrences de $x_1, ..., x_n$. Une seconde application de $\theta$ n'a donc aucun effet.

□

**Definition 60 (Algorithme d'unification).**

On se donne les règles suivantes correspondant à l'algorithme d'unification de *Martelli* et *Montanari* [MM82] qui progressent de haut en bas :

$$\frac{P}{P \cup \{t \doteq t\}}\ \text{effacer} \qquad \frac{P \cup \{t_1 \doteq u_1, ..., t_n \doteq u_n\}}{P \cup \{f(t_1, ..., t_n) \doteq f(u_1, ..., u_n)\}}\ \text{ouvrir}$$

$$\frac{P \cup \{x \doteq t\}}{P \cup \{t \doteq x\} \text{ avec } t \notin \texttt{Vars}}\ \text{diriger} \qquad \frac{\{\!|x \mapsto t|\!\}P \cup \{x \doteq t\}}{P \cup \{x \doteq t\} \text{ avec } x \in \texttt{vars}(P) \text{ et } x \notin \texttt{fv}(t)}\ \text{remplacer}$$

$$\frac{}{P \text{ (en forme résolue)}}\ \text{succès} \qquad \frac{\bot}{P \text{ (en forme non résolue)}}\ \text{échec}$$

Un arbre de dérivation formé à partir de ces règles et se terminant par la règle "succès" ou "échec" lorsque plus aucune règle n'est applicable est appelé **exécution** de l'algorithme

d'unification. La conclusion de l'exécution est notée $\mathcal{S}\{P\}$ pour un problème $P$ de départ. S'il peut rester des règles applicables, on parle **d'exécution partielle**.

On remarque que la condition $x \in \texttt{Vars}(P)$ permet de "fixer" les substitutions qui sont terminales c'est-à-dire qu'on ne traitera plus.

**Théorème 11 (Correction).**

Si l'algorithme sur $P$ termine avec une erreur, alors $P$ n'a pas d'unificateur. Si l'algorithme termine avec succès alors on obtient un problème $S$ équivalent en forme résolue.

**Preuve.**

La preuve se trouve dans [MM82] et procède par analyse de chaque règle. Ce théorème permet d'extraire un unificateur $\vec{S}$ à partir du problème final résultant de l'algorithme d'unification. On peut considérer par commodité que cet algorithme renvoie $\vec{S}$.

□

**Corollaire 12 (Existence d'un unificateur principal).**

Tout problème d'unification $P$ soluble a une solution principale.

**Preuve.**

L'algorithme d'unification permet de calculer un unificateur $\theta = \{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ en forme résolue pour un problème $P$. Par la correction de l'algorithme, on en déduit l'existence d'un unificateur pour $P$. Comme il est en forme résolue alors on a $\theta(x_i) = t_i$ et $\theta(t_i) = t_i$. Soit $\psi$ une solution arbitraire du problème $P$. On a nécessairement $\psi(x_i) = \psi(t_i)$ car c'est une solution de $P$. Donc on a $\psi(x_i) = \psi(t_i) = (\psi \circ \theta)(x_i)$ (par forme résolue) impliquant $\psi = \psi \circ \theta$. Par conséquent, $\theta \preceq \psi$ et $\theta$ est bien principal.

□

**Théorème 13 (Équivalence par renommage).**

Soient $\theta_1$ et $\theta_2$ deux substitutions en forme résolue. Si $\theta_1 \sim \theta_2$ alors $\theta_1 \approx_\alpha \theta_2$.

**Preuve.**

Supposons que $\theta_1 \sim \theta_2$. Par définition, on a donc $\theta_1 \preceq \theta_2$ et $\theta_2 \preceq \theta_1$. En particulier, il existe $\psi$ tel que $\theta_1 = \psi \circ \theta_2$. On raisonne sur l'action de $\psi$ sur une variable $x$. Supposons que $\psi(x) = f(t_1, ..., t_n)$ avec $f$ un symbole d'arité $n$ quelconque. On a donc altéré une certaine substitution $y \mapsto u \in \theta_2$ (car on a des formes résolues) qui nous donne $(\psi \circ \theta_2)(y) = \{\!|x \mapsto f(t_1, ..., t_n)|\!\}u = \theta_1(y)$ (par hypothèse). Mais cela contredit $\theta_1 \preceq \theta_2$ car $\theta_2(y) = u$ et $\theta_1(y) = \{\!|x \mapsto f(t_1, ..., t_n)|\!\}u$. Il est impossible de trouver un $\psi'$ tel que $\theta_2 = \psi' \circ \theta_1$.

Donc, toutes les variables $x$ doivent forcément être renvoyées sur une autre variable $y$. La substitution $\psi$ est donc un renommage.

□

**Corollaire 14 (Unicité de l'unificateur principal).**

Si un problème $P$ est soluble avec un unificateur principal $\theta$ en forme résolue, alors $\theta$ est unique à renommage près.

**Preuve.**

Soient $\theta_1, \theta_2$ deux unificateurs principaux en forme résolue pour $P$. Comme ils sont tous deux principaux on a $\theta_1 \sim \theta_2$. Par le théorème d'équivalence par renommage, $\theta_1 \approx_\alpha \theta_2$.

□

**Corollaire 15 (Confluence de l'algorithme d'unification).**

Deux ordres d'exécutions sans échec de l'algorithme d'unification sur un problème $P$ induisent la même solution à renommage près.

**Preuve.**

Supposons deux ordres d'exécution quelconques de l'algorithme d'unification sur le problème $P$ produisant les solutions principales $\vec{S}, \vec{S'}$. Par le théorème d'unicité de l'unificateur principal, on a $\vec{S} \approx_\alpha \vec{S'}$.

□

**Théorème 16 (Terminaison).**

L'algorithme d'unification se termine.

**Preuve.**

La preuve est décrite dans [MM82] et se base sur la bonne-fondation de $\mathbb{N}^3$ en montrant la décroissance d'une mesure $(n_1, n_2, n_3)$ sur un problème d'unification au fil de l'exécution de l'algorithme où :

- $n_1$ est le nombre de variables qui apparaissent plusieurs fois à gauche d'une équation.
- $n_2$ est le nombre total de symboles dans le problème.
- $n_3$ est la somme du nombre d'équations de la forme $x \doteq x$ et $t \doteq x$ où $x \in \texttt{Vars}$ et $t \notin \texttt{Vars}$.

□

**Lemme 17 (Décomposition de solution).**

Pour tous problèmes $P_1, P_2$, on a $\mathcal{S}\{P_1 \cup P_2\} = \mathcal{S}\{\mathcal{S}\{P_1\}(P_2) \circ \mathcal{S}\{P_1\}\}$

**Preuve.**

Dans cette preuve on considère implicite le renommage de deux termes de sorte à ce qu'ils soient égaux lorsqu'ils sont équivalents à renommage près. Comme l'ordre d'exécution de l'algorithme d'unification est arbitraire, on peut résoudre $P_1 \cup P_2$ par une stratégie particulière en se concentrant sur les équations de $P_1$ permettant de construire $\mathcal{S}\{P_1\}$ (dont le domaine reste fixé car la règle "remplacer" efface les occurrences des variables dans le reste du problème). Il reste $\mathcal{S}\{P_1\}(P_2)$ que l'on résoud pour obtenir $\mathcal{S}\{\mathcal{S}\{P_1\}(P_2)\}$. Par correction, même après altération du codomaine pendant cette dernière résolution, on garde une solution de $P_1$ équivalente à renommage près à $\mathcal{S}\{P_1\}$.

Comme $\mathcal{S}\{\mathcal{S}\{P_1\}(P_2) \circ \mathcal{S}\{P_1\}\}$ est une exécution complète de l'algorithme d'unification sur $P_1 \cup P_2$ on obtient $\mathcal{S}\{P_1 \cup P_2\}$.

□

# Annexe 0 - Théorème de confluence (incomplet)

**Corollaire 18 (Confluence de la fusion).**

La réduction par fusion est confluente c'est-à-dire que si $D_1 \rightsquigarrow^* D'_1$ et $D_2 \rightsquigarrow^* D'_2$, alors il existe $D$ tel que $D'_1 \rightsquigarrow^* D$ et $D'_2 \rightsquigarrow^* D$.

**Preuve.**

L'algorithme d'unification qu'on a choisi est tel que l'ordre d'exécution n'a pas d'influence sur le résultat. Donc, tous les choix mènent vers les mêmes substitutions à renommage près pour un problème $P$ posé par un diagramme. Comme la réduction par fusion est équivalente à l'actualisation qui est confluente, on peut en déduire qu'elle est aussi confluente.

□

On définit une normalisation pour une unique paire de couleur afin de pouvoir exprimer la propriété de confluence dans notre formalisme.

**Definition 61 (Actualisation ciblée).**

L'**actualisation ciblée** d'un diagramme $D$ notée $\Downarrow_\alpha D$ applique une fusion pour toutes les liaisons entre $\alpha$ et $\overline{\alpha}$. L'actualisation ciblée retourne un autre diagramme.

**Definition 62 (Normalisation ciblée).**

La **normalisation ciblée** $\mathtt{Ex}^\star_\alpha(\Sigma)$ sur une couleur $\alpha$ pour constellation $\Sigma$ correspond à effectuer une normalisation où on ne construit que des diagrammes contenant uniquement les couleurs $\alpha$ et $\overline{\alpha}$.

**Lemme 19 (Préservation de saturation).**

Un diagramme $D$ est saturé sur deux couleurs $\alpha$ et $\beta$ si et seulement il est saturé sur $\alpha$ et sur $\beta$.

**Preuve.**

La saturation implique qu'il n'est pas possible de dupliquer un sommet de sorte à pouvoir le connecter au rayon d'une étoile. Un rayon étant affecté à une unique couleur, en étendant un diagramme en général, on étend une seule couleur. La saturation sur deux couleurs $\alpha$ et $\beta$ forme donc deux opérations indépendantes.

□

**Théorème 20 (Confluence de la normalisation ciblée).**

$\mathtt{Ex}^\star_\beta(\mathtt{Ex}^\star_\alpha(\Sigma_A)) = \mathtt{Ex}^\star_\alpha(\mathtt{Ex}^\star_\beta(\Sigma_A))$

**Preuve.**

On fixe les notations suivantes :

$$\Sigma_B := \mathtt{Ex}^\star_\alpha(\Sigma_A) \qquad \Sigma_{B'} := \mathtt{Ex}^\star_\beta(\Sigma_A) \qquad \Sigma_C := \mathtt{Ex}^\star_\beta(\mathtt{Ex}^\star_\alpha(\Sigma_A)) \qquad \Sigma_{C'} := \mathtt{Ex}^\star_\alpha(\mathtt{Ex}^\star_\beta(\Sigma_A))$$

On a donc :

$$\Sigma_C \Leftarrow^\beta \Sigma_B \Leftarrow^\alpha \Sigma_A \Rightarrow^\beta \Sigma_{B'} \Rightarrow^\alpha \Sigma_{C'}$$

et on souhaite montrer $\Sigma_C \approx_\alpha \Sigma_{C'}$. Considérons un diagramme $D$ de $\Sigma_B$ (destiné à devenir une étoile de $\Sigma_C$ par normalisation de la couleur $\beta$). Le diagramme $D$ est composé d'étoiles de $\Sigma_B$ dont certaines proviennent d'actualisations de diagrammes de $\Sigma_A$. Supposons qu'on inverse cette opération en remplaçant ces étoiles par les diagrammes correspondants. On obtient un diagramme bicolore sur $\Sigma_A$ avec les liaisons $\beta$ déjà présentes mais aussi les liaisons $\alpha$ par inversion de l'étape de normalisation.

On sait que la fusion est confluente (et que l'actualisation est une séquence de fusions dans un ordre quelconque), donc on peut choisir de réduire la couleur $\alpha$ pour avoir un diagramme de $\Sigma_B$ ou la couleur $\beta$ pour avoir un diagramme de $\Sigma_{B'}$. Ce diagramme devient finalement une étoile commune de $\Sigma_C$ et $\Sigma_{C'}$. On peut utiliser ce raisonnement pour montrer que $\Sigma_C \subseteq \Sigma_{C'}$. On peut utiliser le raisonnement symétrique pour montrer $\Sigma_{C'} \subseteq \Sigma_C$. On obtient finalement $\Sigma_C = \Sigma_{C'}$, en particulier, les deux ensembles ont le même nombre d'occurrences d'étoiles.

Il reste à prendre en compte la saturation : on sait que le diagramme $D$ est saturé et les diagrammes associés aux étoiles provenant de $\Sigma_A$ sont aussi saturés. Par le lemme de préservation de saturation, remplacer les étoiles par ces diagrammes forme donc un diagramme saturé.

□

**Corollaire 21 (Confluence de la normalisation).**

Soit $\Sigma$ possédant deux paires de couleurs $(\alpha, \overline{\alpha})$ et $(\beta, \overline{\beta})$. On a

$$\mathtt{Ex}^\star_\beta(\mathtt{Ex}^\star_\alpha(\Sigma_A)) = \mathtt{Ex}^\star_\alpha(\mathtt{Ex}^\star_\beta(\Sigma_A)) = \mathtt{Ex}^\star(\Sigma_A)$$

**Preuve.**

La première égalité est donnée par le théorème de confluence de la normalisation ciblée. Ce lemme nous dit que l'on peut effectuer la normalisation à partir des couleurs que l'on veut (pour une paire de deux couleurs). On peut supposer (sans impact sur le résultat) que toutes les paires de $\Sigma_A$ sont de couleurs distinctes ce qui nous permet de normaliser les paires de couleur deux à deux pour obtenir le même résultat.

□

# Annexe 1 - Une étude stellaire de la normalisation

On montre quelques résultats concernant la forme des graphes d'unifiabilité et leur impact sur la normalisation. Un premier résultat est qu'une constellation avec un graphe d'unifiabilité acyclique est nécessairement fortement normalisable. On ne prend pas en compte la saturation. On parle de *multigraphe* mais on utilisera le terme graphe.

**Definition 63 (Acyclicité des constellations).**

Une constellation $\Sigma$ est **acyclique** si $\mathcal{U}(\Sigma)$ est acyclique.

**Definition 64 (Chemin régulier).**

Un **chemin régulier** dans un graphe est un chemin $\pi$ sans boucle, c'est-à-dire qu'il n'y a pas d'arête $e = (s, s)$ pour un certain sommet $s$.

**Lemme 22 (Génération).**

Si $\pi$ est un chemin régulier dans $\mathcal{U}(\Sigma)$ pour une constellation $\Sigma$ alors on peut définir un diagramme $D(\pi)$ de $\mathcal{U}(\Sigma)$.

**Preuve.**

Par induction sur le nombre d'arêtes de $\pi$ :

**Cas de base** On a une arête $e$. On a bien formé un arbre et donc un diagramme (car $\pi$ est régulier).

**Récurrence** On a $\pi = \rho \cdot e_{n+1}$. Par hypothèse d'induction on peut construire $D(\rho)$ à partir duquel on ajoute l'arête $e_{n+1}$ qui a sa source dans $D(\rho)$ et on choisit pour cible une nouvelle occurrence d'étoile étiquetée par la cible de $e_{n+1}$. On obtient bien un arbre (car $D(\rho)$ est un arbre et qu'on a ni déconnecté le graphe ni formé de cycle).

□

**Lemme 23 (Simulation de chemins).**

Soit $D$ un diagramme de $\mathcal{U}(\Sigma)$ pour une constellation acyclique $\Sigma$ tel qu'on ait un morphisme de graphe $D \overset{f}{\mapsto} \mathcal{U}(\Sigma)$. Si $\pi_n = e_1 \cdot ... \cdot e_n$ est un chemin $n$ dans $D$ alors $f(\pi_n) = f(e_1) \cdot ... \cdot f(e_n)$ est un chemin de taille $n$ dans $\mathcal{U}(\Sigma)$.

**Preuve.**

Par induction sur la taille d'un chemin $\pi$ dans $G$ :

**Cas de base** On a $\pi = v$ où $v$ est un sommet. Par définition des diagrammes, ce sommet est aussi dans $\mathcal{U}(\Sigma)$.

**Récurrence** Supposons, par hypothèse d'induction, qu'un chemin $\pi_n = e_1 \cdot ... \cdot e_n$ de taille $n$ dans $D$ induise un chemin $f(\pi_n) = f(e_1) \cdot ... \cdot f(e_n)$ de taille $n$ dans $\mathcal{U}(\Sigma)$. Montrons que c'est le cas pour un chemin $\pi_{n+1} = \pi_n \cdot e_{n+1} = e_1 \cdot ... \cdot e_n \cdot e_{n+1}$ de taille $n+1$ dans $D$. On a donc un certain nombre d'arêtes entre $e_n$ et $e_{n+1}$.

- Soit $e_{n+1}$ a le même label qu'un certain $e_k$ avec $1 \leq k \leq n$ et dans ce cas cela induit un cycle $f(e_k), ..., f(e_{n+1})$ dans $\mathcal{U}(\Sigma)$ ce qui est contradictoire (car $\mathcal{U}(\Sigma)$ est acyclique) donc ce cas ne peut pas se produire.
- Sinon, tous les labels de $\pi_{n+1}$ sont disjoints et donc cela forme bien un chemin élémentaire $f(\pi_{n+1}) = f(\pi_n) \cdot f(e_{n+1})$ de taille $n+1$ en utilisant le chemin de taille $n$ de l'hypothèse d'induction.

□

**Théorème 24 (Finitude).**

Si $\mathcal{U}(\Sigma)$ est acyclique alors il induit un nombre fini de diagrammes.

**Preuve.**

Par le lemme de simulation de chemin, la taille des chemins dans un diagramme de $\mathcal{U}(\Sigma)$ est bornée par celle des chemins de $\mathcal{U}(\Sigma)$. Comme $\mathcal{U}(\Sigma)$ est acyclique, il a un nombre fini de chemin. Donc c'est aussi le cas de tout diagramme $D$ induit. On ne peut donc formé qu'un nombre fini de diagrammes car chaque arbre de $\mathcal{U}(\Sigma)$ est caractérisé par un ensemble d'arêtes de $\mathcal{U}(\Sigma)$.

□

**Corollaire 25 (Normalisation forte).**

Si une constellation $\Sigma$ est acyclique alors elle est fortement normalisable.

**Preuve.**

Par le théorème de finitude, comme $\Sigma$ est acyclique, son graphe d'unifiabilité $\mathcal{U}(\Sigma)$ induit un nombre fini de diagrammes et donc en particulier un nombre fini de diagrammes corrects.

□

Une remarque intéressante est que la présence de cycle est liée au potentiel infini du nombre de diagramme mais ne l'induit pas nécessairement.

**Corollaire 26 (Infinitude).**

Si une constellation $\Sigma$ est cyclique alors elle engendre une infinité de diagrammes $D$.

**Preuve.**

Une constellation cyclique possède un graphe d'unifiabilité avec un nombre de chemin arbitraire donc on peut former une infinité de diagrammes grâce au lemme de génération.

□

**Théorème 27 (Infinitude annulée).**

Il existe au moins une constellation $\Sigma$ cyclique et fortement normalisable.

**Preuve.**

On peut choisir la constellation suivante : $[\![\alpha.x, \beta.x]\!] + [\![\overline{\alpha}.f(y), \overline{\beta}.g(y)]\!]$. La constellation possède bien un cycle de taille 2 mais seuls les diagrammes de taille 1 sont corrects. Tous les diagrammes de taille 2 (il n'y en a qu'un) sont incorrects. Donc la constellation est fortement normalisable. On peut aussi remarquer que si on considère des diagrammes saturés, on en obtient aucun avec cette constellation.

□

**Théorème 28 (Cycle parfait).**

Soit $\Sigma$ une constellation. S'il y a un cycle $\pi = v_1 \cdot ... \cdot v_n \cdot v_1$ dans $\mathcal{U}(\Sigma)$ tel que toutes les arêtes du cycle portent des équations de la forme $t \doteq t$ pour un certain $t$, alors on peut construire une infinité de diagrammes corrects sur $\mathcal{U}(\Sigma)$.

**Preuve.**

Les sommets $v_1 ... v_n$ induisent un diagramme linéaire sur $\mathcal{U}(\Sigma)$. On peut construire un diagramme linéaire de taille arbitraire de la forme $(v_1 ... v_n)^N$ pour un $N > 1$ quelconque par le lemme d'infinitude. Comme on a un "matching parfait", c'est-à-dire que tous les rayons connectés sont deux à deux unifiables, tous les diagrammes formés sont corrects.

□

## Annexe 2 - Modèles de MLL par biorthogonalité

On peut définir un modèle de MLL par biorthogonalité en nous inspirant de l'orthogonalité des entités en ludique [Gir01] et des constructions catégoriques par *tight double glueing* [HS03] avec les éléments suivants :

**Dessein** Une entité actrice de l'interaction associé à des adresses (aussi appelés *lieux*).

**Interaction** Une notion d'interaction notée :: (qui agit sur un certain lieu) qui va connecter deux desseins pour en produire un autre.

**Orthogonalité** Une notion d'orthogonalité symétrique qui oppose les desseins et les sépare en deux classes selon un point de vue subjectif/arbitraire.

**Cohérence** Des propriétés qui assurent la cohérence de l'interaction comme *l'associativité* de l'interaction, et une *propriété d'adjonction* qui va nous permettre de dissocier le comportement du connecteur ⅋ et ⊗.

**Definition 65 (Orthogonalité).**

On note $\perp$ la relation binaire d'**orthogonalité** entre deux desseins et

$$\mathbf{A}^{\perp} = \{b \mid \forall a \in \mathbf{A}. a \perp b\}$$

**Definition 66 (Comportement).**

Un **comportement** est un ensemble $\mathbf{A}$ de desseins tel que $\mathbf{A} = \mathbf{A}^{\perp\perp}$

**Definition 67 (Tenseur).**

Soient $\mathbf{A}, \mathbf{B}$ deux comportements. On définit le tenseur par :

$$\mathbf{A} \otimes \mathbf{B} = \{a \oslash b \mid a \in \mathbf{A}, b \in \mathbf{B}\}^{\perp\perp}$$

où $a \oslash b$ est une opération fixée d'avance.

**Definition 68 (Flèche linéaire).**

Soient $\mathbf{A}, \mathbf{B}$ deux comportements. On définit la flèche linéaire par :

$$\mathbf{A} \multimap \mathbf{B} = \{f \mid \forall a \in \mathbf{A}, f :: a \in \mathbf{B} \text{ et } f \perp a\}^{\perp\perp}$$

**Axiome 29 (Adjonction).**

Soient $a, b, c$ des desseins. On a $a \perp (b \otimes c)$ si et seulement si $a \perp b$ et $a :: b \perp c$.

**Théorème 30 (Traduction de flèche linéaire).**

Soient $\mathbf{A}, \mathbf{B}$ deux comportements. L'adjonction implique :

$$\mathbf{A} \multimap \mathbf{B} = (\mathbf{A} \otimes \mathbf{B}^{\perp})^{\perp}$$

**Preuve.**

Par définition, $f \in \mathbf{A} \multimap \mathbf{B}$ si et seulement si pour tout $A \in \mathbf{A}, f :: a \in \mathbf{B}$ avec $f \perp a$. Comme tout comportement est orthogonal à un autre on a pour tout $a \in \mathbf{A}$ et pour tout $b' \in \mathbf{B}^{\perp}, f :: a \perp b'$. Par adjonction, $f \perp a \otimes b'$. Par conséquent, $f \in (\mathbf{A} \otimes \mathbf{B}^{\perp})^{\perp}$.

□

En nous aidant de ces modèles par biorthogonalité on peut ensuite construire des catégories $*$-autonomes nous fournissant un modèle de MLL. On peut retrouver une telle reconstruction dans les graphes d'interaction de Seiller [Sei12a].

# Annexe 3 - Modèle d'étoiles (incomplet)

**Definition 69 (Execution).**

Soient $\Sigma_1$ et $\Sigma_2$ deux constellations. Leur exécution est définie par $\Sigma_1 :: \Sigma_2 = \texttt{Ex}^\star(\Sigma_1 \cup \Sigma_2)$.

**Definition 70 (Type).**

Un **type** est un ensemble de constellations $\mathbf{A}$ tel que il existe un ensemble de constellations $B$ tel que $\mathbf{A} = B^\perp$.

**Lemme 31 (Inclusion).**

Pour tout ensemble de constellations $A$, on a $A \subseteq A^{\perp\perp}$.

**Preuve.**

Soit $\Sigma \in A$. Par définition, pour tout $\Sigma' \in A^\perp$, $\Sigma \perp \Sigma'$. Par conséquent $\Sigma \in A^{\perp\perp}$ par définition de $A^{\perp\perp}$.

□

**Lemme 32 (Triple orthogonalité).**

Pour tout ensemble de constellations $A$, on a $A^{\perp\perp\perp} = A^\perp$.

**Preuve.**

($A^\perp \subseteq A^{\perp\perp\perp}$) Conséquence du lemme d'inclusion.

($A^{\perp\perp\perp} \subseteq A^\perp$) Soit $\Sigma \in A^{\perp\perp\perp}$. Par définition, il est orthogonal à tout élément de $A^{\perp\perp}$. Or, on a $A \subseteq A^{\perp\perp}$ par le lemme d'inclusion. Donc $\Sigma$ est orthogonal à tout élément de $A$ et par conséquent $\Sigma \in A^\perp$ par définition de $A^\perp$.

□

**Théorème 33 (Clôture par biorthogonal).**

Soit $\mathbf{A}$ un ensemble de constellations. On a $\mathbf{A} = \mathbf{A}^{\perp\perp}$ si et seulement si $\mathbf{A}$ est un type.

**Preuve.**

($\Rightarrow$) Supposons qu'on a $\mathbf{A} = \mathbf{A}^{\perp\perp}$. Donc on a $B = \mathbf{A}^\perp$ tel que $\mathbf{A} = (\mathbf{A}^\perp)^\perp = B^\perp$.

($\Leftarrow$) Soit $B^\perp$ tel que $\mathbf{A} = B^\perp$. Montrons qu'on a $\mathbf{A} = \mathbf{A}^{\perp\perp}$. Par le lemme de triple orthogonalité sur $B$, on a $B^{\perp\perp\perp} = B^\perp$, donc $\mathbf{A} = \mathbf{A}^{\perp\perp}$.

□

On définit un modèle de MLL par orthognalité avec les constellations comme desseins, l'exécution comme interaction et la terminaison de l'exécution comme orthogonalité.

**Definition 71 (Orthogonalité).**

Soient $\Sigma_1$ et $\Sigma_2$ deux constellations. On a $\Sigma_1 \perp \Sigma_2$ si $\Sigma_1 :: \Sigma_2$ est fortement normalisable.

**Théorème 34 (Associativité de l'exécution).**

Soient $\Sigma_1, \Sigma_2, \Sigma_3$ des constellations. On a $(\Sigma_1 :: \Sigma_2) :: \Sigma_3 = \Sigma_1 :: (\Sigma_2 :: \Sigma_3)$.

**Preuve.**

La preuve découle naturellement du théorème de confluence de la normalisation.

□

**Théorème 35 (Adjonction).**

Soient $\Sigma_1, \Sigma_2, \Sigma_3$ des constellations. On a $\Sigma_1 \perp (\Sigma_2 \otimes \Sigma_3)$ si et seulement si $\Sigma_1 \perp \Sigma_2$ et $(\Sigma_1 :: \Sigma_2) \perp \Sigma_3$.

**Preuve.**

Preuve non abordée dans ce document. Il faut aussi définir le tenseur.

□

Il reste à montrer qu'on peut former une catégorie $*$-autonome $\mathbf{Star}_{MLL}$. On peut, pour cela, s'inspirer d'construction plus détaillée où on construit une catégorie $*$-autonome à travers un modèle de MLL par orthogonalité pour les graphes d'interaction [Sei12a].

## Annexe 4 - Des étoiles aux flots (expérimental)

On cherche à simuler les flots à partir des étoiles. Les flots ont une polarité, c'est-à-dire qu'on ne connecte que les têtes et les queues. On peut représenter un flot par une étoile à deux couleurs $(t \leftharpoonup u)^\star = [\![-t, +u]\!]$ où $+$ et $-$ sont deux couleurs duales. On peut étendre cette traduction aux câblages.

Cependant, la dynamique des flots est différente de celle des étoiles. Les flots sont plus libres que les étoiles car on ne leur impose pas de symétrie. En particulier le produit de flot est orienté. On peut, par contre, montrer qu'on peut simuler la notion d'exécution qu'on utilise en logique.

**Théorème 36 (Simulation).**

Soit $F$ et $G$ deux câblages. On a $\mathtt{Ex}_f(F, G) = \mathtt{Ex}^\star(F^\star \cup G^\star)$.

On peut, à l'inverse, imaginer s'il est possible de simuler les étoiles avec une extension des flots. En particulier, il serait idéal d'avoir un formalisme aussi expressif sans être soumis à une symétrie. On peut par exemple imaginer une idée de lieur de variable entre des flots.

## Annexe 5 - Modèle stellaire et résolution du premier ordre

On peut simuler la programmation logique pour la résolution du premier ordre avec les étoiles et constellations en utilisant une couleur pour représenter les entrées et une autre pour représenter les sorties. On travaille avec des clauses disjonctives.

**Definition 72 (Traduction).**

On considère la traduction suivante de la résolution vers les constellations :

$$\text{Clause} : (t_1 \vee ... \vee t_n \vee u)^\star = [\![-t_1, ..., -t_n, +u_1]\!] \quad \text{où } u \text{ est un littéral positif}$$

$$\text{Programme} : (P_1, ..., P_n)^\star = P_1^\star + ... + P_n^\star$$

**Exemple.** On peut considérer le programme suivant :

$$[\![+add(0,y,y)]\!] + [\![-add(s(x),y,s(z)), +add(x,y,z)]\!]$$

**Exemple.** Si on prend le programme précédent auquel on ajoute une autre étoile :

$$\sigma_0[\![+add(0,y,y), end]\!] + \sigma_r[\![-add(s(x),y,s(z)), +add(x,y,z)]\!] + \sigma_s[\![+add(ss(0),ss(0),ssss(0))]\!]$$

On aura un unique diagramme saturé correct donné par la chaîne suivante de liaison :

$$\sigma_s\sigma_r\sigma_r\sigma_s$$

qui va nous donner :

$$[\![+add(ss(0),ss(0),ssss(0))]\!][\![-add(s(x),y,s(z)), +add(x,y,z)]\!]\sigma_r\sigma_s$$
$$= [\![+add(s(0),ss(0),sss(0))]\!][\![-add(s(x),y,s(z)), +add(x,y,z)]\!]\sigma_s$$
$$= [\![+add(0,ss(0),ss(0))]\!]\sigma_s$$
$$= [\![end]\!]$$

L'obtention de la constellation $[\![end]\!]$ nous assure que le calcul s'est terminé correctement. On ne l'obtient pas si on remplace l'étoile $[\![+add(ss(0),ss(0),ssss(0))]\!]$ par $[\![+add(ss(0),ss(0),0)]\!]$, par exemple.

**Definition 73 (Résolution).**

> La **résolution** est définie par les règles suivantes :
>
> $$\frac{\Gamma \vee t \quad \Delta \vee \neg u}{\theta\Gamma \vee \theta\Delta}\ res$$
>
> où $\theta$ est un unificateur principal de $t \doteq u$ et $\Gamma$ et $\Delta$ ne partagent pas de variables.

**Definition 74 (Arbre de dérivation).**

> Un **arbre de dérivation** est un arbre formé à partir d'applications de la règle de résolution de telle sorte à ce qu'il ne puisse pas être étendu par une application de règle supplémentaire.

On peut ensuite imaginer une traduction des arbres de résolution vers des diagrammes partiels (en prenant en compte la réutilisation d'étoiles pour former un arbre). Un diagramme saturé représente un arbre de résolution qu'on ne peut plus étendre, c'est-à-dire une inférence qui termine. L'exécution des constellations permet de calculer toutes les clauses que l'on peut inférer à partir d'un programme et ainsi simuler un algorithme de résolution.

## Annexe 6 - Le maillage conceptuel de Girard (très expérimental)

Le programme repose sur un maillage conceptuel [Gir16a] qui réorganise la relation entre logique et calcul qui sont couramment entremêlés.

| | Analytique/Réponses | Synthétique/Questions |
|---|---|---|
| **Explicite** | Constat | Usine |
| **Implicite** | Performance | Usage |

Ce modèle conceptuel tire son inspiration de l'epistémologie de Kant et en propose une mise à jour tenant compte de la compréhension moderne de la logique et de l'introduction du calcul par l'informatique.

### Le point aveugle

Une raison d'être de cette organisation provient de l'idée de « point aveugle », l'idée que l'acquisition de la connaissance est obstruée par les observations premières et les préconceptions que l'on a d'un sujet. C'est une idée qui émerge du concept de *rupture epistémologique* (qu'on peut associer à Bachelard puis Althusser). Ainsi, pour accéder à un contenu plus "réel" de la logique, il faut identifier des obstacles et procéder à un passage des préjugés au réel. C'est dans cette perpétuelle remise en question qu'évoluerait un domaine scientifique.

En particulier, la logique a la particularité d'être un domaine fortement biaisé par notre conception du monde, un biais dont il faut (et dont on pourrait) se défaire. Ainsi, la logique qui émerge de nos observations du réel (vérité, formules, connecteurs logiques) trouve une morphologie précise à travers les preuves (arborescentes) puis les réseaux de preuves (topologiques) et enfin les flots/étoiles (asynchrones).

Il semblerait que nous n'ayons accès à rien de plus tangible que la syntaxe (la sémantique étant elle-même une syntaxe dissociée). L'idée de la Syntaxe Transcendantale est donc de trouver un espace syntaxique qui ne se réfère à rien d'autre que lui-même. Cette internalisation nous est permise par une interaction calculatoire et symétrique entre différentes entités cohabitant dans un même espace (dans notre cas, l'espace des étoiles et constellations). Ce changement de paradigme nous permet entre d'autre d'étudier la logique sous un nouveau point de vue plus riche et transparent.

### Analytique / Réponses

**Transparence et opacité** Ce que Girard appelle *analytique* correspond à un espace calculatoire libre où résultat (constat) et programme (performance) cohabitent : il ne doit pas y avoir d'opérations cachées. Par exemple l'évaluation du $\lambda$-caclul pur (avec le couple forme normale / rédex) cache des opérations complexes comme le montre son étude par la logique linéaire. Il est donc important de trouver un espace analytique satisfaisant utilisé comme matériau brut pour reconstruire la logique avec le maximum de transparence. Girard appelle *épistate* une généralisation calculatoire des preuves vivant dans un espace purement analytique.

**Subjectivité et objectivité** L'informatique nous montre que les programmes peuvent être des résultats. Les résultats mais aussi les performances (procédures nous menant aux résultats) sont considérés comme des réponses, sans signification, sans engagement. Cela nous amène à deux faits : (1) les réponses portent une part de subjectivité : on décide de voir un programme comme résultat statique ou comme procédure dynamique, (2) la "réponse" est indépendante de toute conception de signification ce qui la rend objective (la subjectivité est effacée si on se place dans un espace où constat et performance sont distingués ce qui nous est donné par les couleurs des rayons d'étoiles).

**Unification** Il existe une multitude d'espaces analytiques. Par exemple, l'espace analytique standard de l'informatique est donné par le codage binaire codant à la fois les résultats et

les programmes qui y mènent [Gir13b]. L'étude de la géométrie de l'interaction s'est stabilisé sur un espace analytique basé sur l'unification de termes qui a donné le modèle des flots [Gir95] puis qui a été mis à jour vers le modèle d'étoiles de la Syntaxe Transcendantale pour permettre des considérations logiques plus ambitieuses (reconstruction des types/formules et du critère de correction de Danos-Regnier).

**La nature de l'analyticité** Ce que Girard entend par un "bon espace analytique" n'est pas parfaitement clair. Tous les modèles de calcul possèdent une part d'analyticité mais ne se valent pas. De plus, il n'est pas évident que l'unifcation fournit un espace "purement" analytique malgré qu'on en ait une intuition vague par le fait qu'il calcule par interaction entre des entités duales (termes unifiables). Une compréhension plus mature (topologique ?) pourrait probablement fournir un éclaircissement (par exemple, la présentation de l'unification par graphe de terme décrit l'unification comme une manipulation de pointeurs). Les graphes d'interaction [Sei12b] semblent fournir une réponse plus satisfaisante puisque l'évaluation est immédiatement induite par la juxtaposition de deux graphes.

### Synthétique / Questions

**Emergence du sens** Le sens émerge par les conventions et l'usage d'entités : Girard propose une dichotomie entre sens d'usine et sens d'usage qui sont deux facettes de la signification. Le synthétique est à la logique ce que l'analytique est au calcul : il type le calcul. Dans les étoiles et constellations cela correspond à connecter le gabarit à un véhicule. La connexion elle-même est analytique (calculatoire) mais leur réunion devient synthétique i.e subjectivement engagée puisqu'une épistate est considérée comme réponse à une formule logique que Girard appelle *dichologie*. On a un engagement dans le sens où une entité calculatoire (véhicule) est étiquettée par une formule certifiant son comportement. La logique classique répond aux questions dans un espace booléen et la logique intuitionniste dans un espace de preuves. Il y a des réponses plus satisfaisantes que d'autres mais c'est principalement le problème de la *certitude* qui les sépare.

**Usine** Le sens d'usine est celui qui pose les labels et l'usage prévu. Girard se réfère à l'industrie automobile. Un véhicule porte une appelation après avoir passé un certain nombre de tests certifiant son usage. Pour les réseaux de preuve cela correspond aux graphes de test que l'on connecte aux axiomes pour vérifier le critère de correction de Danos-Regnier. Ce sont des tests qui sont explicites et qui correspondent au synthétique à posteriori de Kant (la certification par test est empirique). Dans l'idéal, on souhaite un nombre fini de tests d'usine.

**Usage** Le sens comme usage est celui auquel Wittgenstein se réfère [Wit09]. Le sens d'usage d'une entité est défini par le potentiel d'interaction avec d'autres entités du même espace. Chaque entité est donc vu comme un "test d'usage" permettant de définir le sens d'autres entités de façon mutuelle. Le test d'une entité impose une opposition sujet/objet mais qui est interchangeable (le sujet peut être vu comme objet et vice-versa). Prenons un programme informatique, on peut le définir par son comportement lorsqu'il interagit avec un certain environnement qui à son tour, peut être défini par son comportement selon les programmes qui peuvent y être interptés. En logique, les connecteurs sont reconstruits selon leur potentiel d'interaction (type comme ensemble de constellations avec convergence d'interaction certifiée par l'ensemble dual). L'usage d'une entité est souvent imprévisible, dû à l'infinité des interactions potentielles.

**Typage** Le synthétique est fondamentalement une affaire de typage. Un type ou une formule (conformément à la correspondance de Curry-Howard) est un ensemble de tests (qui sont

en fait des entités calculatoires comme les autres). Si le type est clos par biorthogonal c'est-à-dire qu'on a $\mathbf{A} = \mathbf{A}^{\perp\perp}$ alors c'est un test d'usage qui peut interagir mutuellement avec les autres. Les tests d'usines, en général, ne sont pas clos par biorthogonal ce qui créer une asymétrie des tests (on teste les graphes de Danos-Regnier sur un véhicule pour définir un réseau de preuve).

**Certitude** Le problème de la certitude est le suivant : à quel point l'usine certifie l'usage ? [Gir13b] C'est-à-dire, est-ce que par nos définitions (tests d'usine) on capture suffisamment bien l'usage qu'on peut en faire. Comment être sûr qu'une preuve est bien une preuve de ce qu'elle affirme ? Pour les réseaux, la certitude est absolue (*apodictique* pour Girard) pour MLL (car le nombre de tests de Danos-Regnier est fini et l'élimination des coupures plutôt limitée) mais serait seulement raisonnable (*épidictique* pour Girard) pour des fragments plus riches comme la logique du second ordre puisque le nombre de tests est infini mais il semblerait que cette certitude raisonnable soit donnée par un échantillon fini. Ce n'est pas quelque chose que nous avons abordé dans notre travail.

### Les liaisons conceptuelles

**Constat/Performance** La chose qui assure le passage de la performance (l'évaluable) au constat (l'évalué) est la *normalisation forte* c'est-à-dire la terminaison du calcul et l'assurance d'un résultat. Cela correspond à la finitude ou l'infinitude du nombre de diagrammes corrects que l'on peut construire à partir d'une constellation.

**Usine/Usage** La liaison entre usine et usage peut être qualifiée de *déontique*. Il s'agit d'un équilibre entre les définitions (devoirs) et l'utilisation (droits). Les deux ne concordent pas nécessairement. La *normalisation* en déduction naturelle nous assure qu'on peut annihiler les séquences introduction/élimination correspondant à une dualité définition/usage. En calcul des séquents, l'usage prend la forme de coupure (usage de lemme) qui permet de connecter deux preuves. L'*élimination des coupures* produit des preuves équivalentes sans coupures démontrant une clôture des droits par les devoirs.

**Performance/Usage** L'usage possède lui-aussi un contenu calculatoire. En particulier, l'élimination des coupures est une procédure calculatoire. Il s'agit donc de performance. Ce qui assure la cohérence calculatoire de l'élimination des coupure est son *associativité* qui prend la forme de la propriété de Church-Rosser ou de confluence. Dans le formalisme des étoiles et constellations c'est la confluence de la normalisation qui assure une cohérence de l'élimination des coupures.

### Les séparations conceptuelles

**Constat/Performance** Ce qui sépare le constat de la performance est *l'indécidabilité* (qu'on attribue à Church et Turing). C'est-à-dire qu'on ne peut pas réduire l'évaluable à l'évalué. On ne peut pas prédire la terminaison d'un calcul de façon statique/observationnelle.

**Usine/Usage** Ce qui sépare l'usine et l'usage est *l'indécidabilité* (qu'on attribue à Gödel).

Les deux séparations posent chacunes une forme d'incertitude, l'une dans l'analytique et l'autre dans le synthétique.