Colimites et structure de modèle en théorie des types homotopique

Simon Boulier,
sous la direction de Nicolas Tabareau et Kevin Quirin

Juillet 2015*

RAPPORT DE STAGE DE M2, EFFECTUÉ DU 16 MARS AU 1ER AOÛT 2015 DANS L'ÉQUIPE ASCOLA, ÉCOLE DES MINES DE NANTES

Synthèse

Le contexte général

La théorie des types homotopique (HoTT, pour Homotopy Type Theory) est une variante de la théorie des types de Martin-Löf intentionnelle dans laquelle on traite précautionneusement l'égalité, de sorte à ce qu'elle interprète au mieux la notion "d'égalité à isomorphisme près" utilisée par les mathématiciens. Elle émerge du modèle groupoïde de Hofmann et Streicher et du modèle simplicial de Voevodsky. Dans ce dernier, un type est interprété par un espace topologique (un ensemble simplicial en fait), et l'égalité entre deux objets $a, b : A$ par un chemin entre a et b dans l'espace topologique correspondant à A. En première approximation, on peut penser à la théorie des types homotopique comme la théorie de Martin-Löf à laquelle on ajoute l'axiome d'univalence et la construction de HIT (Higher Inductive Types).
Aujourd'hui, la description des équivalences faibles (les isomorphismes de type) et de l'axiome d'univalence sont bien connues. On en trouve une présentation formelle dans le livre collectif [Uni13] qui est la principale synthèse des connaissances de base du domaine. En revanche, l'univalence y est posée en axiome et n'a pas de contenu calculatoire. Des travaux récents autour de la théorie des types cubique [BCH14] commencent à donner un contenu calculatoire à l'univalence. L'on espère qu'à terme, cela mène à une meilleur syntaxe de HoTT, internalisant les lemmes de transport d'égalités. En outre, la théorie complète des HIT n'est pas encore tout à fait claire (les avancées les plus récentes dans ce domaine sont celles de Sojakova [Soj15]).
Un des enjeux majeurs du domaine est de bien comprendre la structure d'homotopie de l'univers **Type**. Par exemple, un des problème ouvert important est de donner une définition des types (semi-)simpliciaux en HoTT. Cela permettrait de reproduire, en théorie des types, le modèle simplicial qui est un modèle ensembliste.
HoTT-Coq [1] est une bibliothèque Coq formalisant la quasi totalité du livre [Uni13] ainsi que quelques autres résultats. Il existe d'autres bibliothèques HoTT : HoTT-Agda [2], UniMath [3], The Lean HoTT Library [4], ... Nous utilisons celle-ci car c'est la plus complète disponible pour Coq.

*version mise à jour le 31 août 2015
1. `https://github.com/HoTT/HoTT`
2. `https://github.com/HoTT/HoTT-Agda`
3. `https://github.com/UniMath/UniMath`
4. `https://github.com/leanprover/lean/blob/master/hott/hott.md`

Le problème étudié

La question principale que nous avons abordée est celle de la formulation, en HoTT, d'un analogue à l'axiome de Giraud. Bien que nous nous intéressions à cette question pour des raisons précises (cf. partie 2), celle-ci est en fait assez profonde puisque qu'elle s'inscrit dans le cadre du lien entre HoTT et la théorie des ∞-topos. Des recherches autour de ce lien on lieu depuis que l'on s'intéresse à la sémantique de HoTT, nous n'avons donc pas la prétention d'y répondre intégralement. En outre, elle est aussi très liée à celle de la définition des ensembles simpliciaux en HoTT, nous en reparlerons.
Nos investigations autour de l'axiome de Giraud nous ont ensuite amenés à nous interroger sur la définition et sur les propriétés des colimites en HoTT. Nous ne savons définir que les colimites sur des graphes, c'est donc les propriétés de telles colimites que nous proposons. On peut montrer l'existence de ces colimites grâce aux HIT, cette idée est présente dans le livre [Uni13]. Par contre, à notre connaissance, les autres propriétés des colimites en HoTT n'ont pas encore été exposées ni formalisées. Cela dit, celles-ci sont peu surprenantes puisqu'elles ont leur analogue en théorie des catégories (exception faite, peut-être, de la commutation aux sigmas, cf. 1.4).
Enfin, la question de la définition des colimites en HoTT, nous a amené à réfléchir à la structure d'homotopie de l'univers **Type** puisque les colimites considérées sont en fait des colimites homotopiques.
Nous avons travaillé sur ces trois questions en parallèle, l'une nourrissant l'autre à chaque fois.

La contribution proposée

Nous avons développé une petite bibliothèque Coq formalisant des propriétés des colimites (existence, fonctorialité, unicité et commutation aux sigmas), ce qui correspond à une formalisation de la première partie de ce rapport.
Grâce à celle-ci, nous pouvons définir la notion de nerf de Čech d'une fonction. En nous basant sur un résultat de Floris van Doorn (cf. 2.1), nous montrons que la colimite du nerf de Čech d'une fonction est équivalente à l'image de cette fonction. Ce théorème pourrait être un analogue de l'axiome de Giraud en HoTT, ce qui était le but de ce stage. Nous présentons ce résultat dans la partie 2.
Et enfin, dans la partie 3, qui est un peu orthogonale aux deux premières, nous proposons une structure de modèle sur l'univers **Type**. Nous reprenons pour cela des résultats existants de Gambino et Garner [GG08] et Lumsdaine [Lum11] mais nous allons plus loin puisque nous proposons une caractérisation descriptive des cofibrations.

Les arguments en faveur de sa validité

Dans la partie 2, nous utilisons la bibliothèque sur les colimites que nous avons créée pour démontrer des résultats non triviaux. Ceci apporte une première validation de cette bibliothèque. En outre, les propriétés de fonctorialité donnent une preuve particulièrement élégante de l'unicité.
La définition du nerf de Čech d'une fonction est assez élégante et c'est le théorème 7 qui justifie le mieux cette définition. Cela dit, nous n'avons, pour l'ins-

tant, pas assez d'arguments pour affirmer que ce que nous proposons est la bonne notion de nerf.
La structure de modèle que nous proposons dans la partie 3 est la même que celle proposée par Lumsdaine, cependant nos définitions de cofibrations et fibrations acycliques sont beaucoup plus descriptives, ce qui est assurément une bonne chose.
Les parties 2 et 3 sont aussi formalisées en Coq. Le développement entier est accessible ici :

`https://github.com/SimonBoulier/hott-colimits`

Le bilan et les perspectives

Notre bibliothèque sur les colimites continuera de s'enrichir au fur et à mesure des utilisations, mais nous pouvons imaginer que le plus gros est fait. Nous réfléchissons à une éventuelle intégration dans la librairie HoTT-Coq.
Nous prévoyons aussi de poursuivre nos investigations autour du nerf de Čech, Kevin Quirin est actuellement en train d'utiliser ce résultat pour développer l'analogue de la faisceautisation d'un ∞-topos en HoTT (cf. plus en détail au 2).
En outre, en théorie des topos, le nerf de Čech d'une fonction est un ensemble simplicial. Nous avons donc espoir que notre définition de nerf de Čech nous suggère des avancées dans la définition des types simpliciaux.
Et enfin, nous avons commencé à utiliser notre structure de modèle pour justifier certains calculs de colimites (cf. 3.4) et nous comptons poursuivre cela dès la rentrée prochaine.

Prélude

Notre travail s'inscrit dans le cadre théorique développé dans le livre [Uni13]. À savoir, la théorie des types de Martin-Löf (types dépendants, types inductifs, égalité propositionnelle). On rappelle ici quelques notations et résultats mais l'on reste volontairement très succin. On rappelle aussi ce que sont l'univalence et les HIT puisqu'ils sont au cœur de la théorie des types homotopique.

Notations

— si $P : A \to \mathbf{Type}$ est une famille de types, $\Pi_a P(a)$ est le produit dépendant de A et P, c'est le type des fonctions $(a : A) \to P(a)$
— si $P : A \to \mathbf{Type}$ est une famille de types, $\Sigma_a P(a)$ est la somme dépendante de A et P, c'est le type des paires $(a; b)$ pour $a : A$ et $b : P(a)$
— **Type** est le type de tous les types (on ne s'intéresse pas aux problèmes d'univers ici)
— si $a, b : A$, $a =_A b$ est le type des preuves d'égalité de a et b, $\mathbf{refl}_a$ est l'habitant canonique de $a =_A a$
— si $P : A \to \mathbf{Type}$ est une famille de types et $p : a =_A b$ et $u : P(a)$, alors $\mathrm{transport}_P(p, u) : P(b)$ est le transport de u le long de p
— **1** est le type *Unit*, ne contenant qu'un habitant
— pour tout type X, $\mathbf{id}_X$ est la fonction identité $X \to X$
— si $f : X \to Y$ et $y : Y$, $\mathbf{fib}_f\, y$ est la fibre de f en y : $\Sigma_x f(x) = y$

On omettra régulièrement les différents indices lorsque le contexte est clair.

Équivalences Une notion cruciale en HoTT est celle d'*équivalence faible*, ce qui correspond à une notion d'isomorphisme de type.
Soit $f : X \to Y$ une fonction. On dit que f est une équivalence faible si le type $\mathbf{IsEquiv}\, f$ est habité. Il existe plusieurs définition du type $\mathbf{IsEquiv}\, f$, nous ne les préciserons pas. Il importe seulement de connaître le résultat suivant : s'il existe g tel que $g \circ f = \mathbf{id}$ et $f \circ g = \mathbf{id}$ alors $\mathbf{IsEquiv}\, f$.
On notera $X \simeq Y$ quand il existe une équivalence entre X et Y. On a par exemple le résultat suivant que l'on utilisera plusieurs fois :

$$X \simeq \Sigma_y \mathbf{fib}_f\, y$$

via $\lambda x.\, (f x; (x; 1))$ et π_2 (la projection canonique $\Sigma_y \mathbf{fib}_f\, y \to Y$).

Univalence L'axiome d'univalence décrit quand est-ce que deux types sont égaux. Il assure que deux types A et B sont égaux si et seulement si ils sont équivalents :

$$(A = B) \simeq (A \simeq B)$$

La façon naturelle de postuler cela est la suivante :
Axiome d'Univalence : pour tous types A et B, $\mathbf{id\_to\_iso} : (A = B) \to (A \simeq B)$ est une équivalence.
Une conséquence non triviale de l'axiome d'univalence est l'extensionnalité fonctionnelle : pour toutes fonctions $f, g : A \to B$, on a :

$$f = g \quad \text{si et seulement si} \quad \text{pour tout } x : A,\ f(x) = g(x)$$

Dans les travaux qui suivent nous n'utiliserons jamais l'univalence mais seulement cette conséquence (qui est à priori bien plus faible).

HIT Les types inductifs d'ordre supérieur (HIT, pour Higher Inductive Types) sont une généralisation des types inductifs dans laquelle un type n'est pas seulement engendré par des constructeurs mais aussi par des constructeurs d'égalité entre les habitants de ce type. Donnons un exemple pour illustrer cela.
Le cercle $\mathbb{S}^1$ est le HIT suivant :

```
Inductive S1 :=
  | base : S1
  | loop : base = base.
```

Ici, **base** est un constructeur de point, et **loop** un constructeur d'égalité. Ce qui est important c'est que **loop** est une preuve d'égalité *librement engendrée*. Ainsi, dans le cas du cercle, on peut montrer que $\mathbf{loop} \neq \mathbf{refl}_{\mathbf{base}}$. Ce qui est toujours un peu difficile avec les HIT, c'est d'énoncer le principe d'élimination. Dans le cas du cercle c'est le suivant. Pour toute famille de type $P : \mathbb{S}^1 \to \mathit{Type}$
— si $\mathbf{base}' : P(\mathbf{base})$
— et si $\mathbf{loop}' : \quad \mathrm{transport}_P(\mathbf{loop}, \mathbf{base}') \;=\; \mathbf{base}'$
alors $\mathbf{rec}_{\mathbb{S}^1}(\mathbf{base}', \mathbf{loop}') : \Pi_w P(w)$.

Niveaux d'homotopie Une dernière notion très importante en HoTT est celle de niveau d'homotopie. Intuitivement un type est dit de niveau n si tous les types d'égalités de niveau n sont triviaux [5].
Nous n'utiliserons que la notion de proposition qui correspond à celle de (-1)-niveau. Un type A est une *proposition* si l'on a :

$$\text{pour tout } a, b : \; A \;,\; a =_A b$$

Il existe une notion de troncation propositionnelle. Soit A un type de niveau quelconque. On dit que le type T_A est une troncation propositionnelle de A si c'est une proposition et que l'on a la propriété universelle suivante :

$$\text{pour toute proposition } B, \quad T_A \to B \;\simeq\; A \to B$$

On peut définir une troncation propositionnelle grâce aux HIT (cf. 2.1).

1 Colimites en HoTT

Type, le type de tous les types, forme une catégorie dont les objets sont les types et les flèches les fonctions entre ces types. On cherche à calculer des limites et colimites de diagrammes simples.
Par exemple une limite du diagramme

$$A \qquad B$$

où A et B sont des types, sera le produit $A \times B$, et une colimite le coproduit $A + B$.

5. plus précisément contractibles : habités par exactement un seul élément

L'article [AKL15] explique comment calculer les limites de diagrammes sur des graphes. Les limites sont relativement simples à calculer car elles représentent des sous-ensemble, ce que l'on peut encoder grâce à des types Σ. Par exemple, une limite du diagramme suivant :

$$A \underset{g}{\overset{f}{\rightrightarrows}} B$$

est $\Sigma_{x:A}\, f(x) = g(x)$, on appelle *égaliseur* de f et g ce type.
Pour les colimites, cela est un peu plus compliqué : les colimites représentent des quotients, que l'on ne peut facilement encoder en théorie des types de Martin-Löf. Par contre, on peut facilement les encoder grâce aux types inductifs d'ordre supérieur (HIT). Par exemple, une colimite du diagramme précédent — qui sera le *coégaliseur* de f et g — est donné par le HIT suivant (cf. [Uni13]) :

```
Inductive Coeq : Type :=
  | coeq : B -> Coeq
  | cp : forall x:A, coeq (f x) = coeq (g x).
```

Dans cette partie nous allons définir ce qu'est une colimite (1.1) puis nous montrerons des propriétés sur celles-ci : existence (1.2) ; fonctorialité et unicité (1.3) ; commutation aux sigmas et produits (1.4).

1.1 Définition

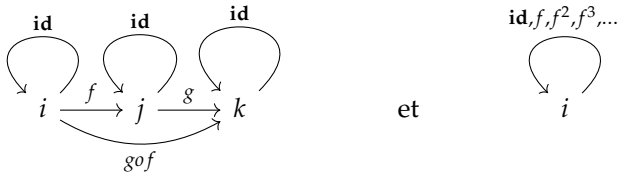
On ne sait pas calculer les limites et les colimites de tous les diagrammes mais seulement de certains diagrammes particulièrement simples : les diagrammes sur les graphes. Ce sont les diagrammes où les seules compositions de flèches sont des flèches librement engendrées. On suit ici la démarche de [AKL15].

Définition 1. *Un* graphe *G consiste en la donnée :*
— *d'un type* G_0 *de* sommets
— *pour chaque* $i, j : G_0$ *, d'un type* $G_1(i, j)$ *d'*arrêtes *entre i et j.*

Les graphes engendrent une catégorie libre, et c'est cette catégorie libre qui servira de forme aux diagrammes que nous considérerons. Par exemple les graphes

$$i \xrightarrow{f} j \xrightarrow{g} k \qquad \text{et} \qquad \circlearrowleft_{f} \; i$$

engendrent respectivement les catégories



et

.

On fixe un graphe G pour la suite de cette partie. Tous les diagrammes sont pris sur ce graphe.
Un diagramme est un foncteur entre la catégorie libre engendrée par G et la catégorie **Type**. Concrètement, on définit un diagramme comme suit.

Définition 2. *Un* diagramme *D sur G consiste en la donnée :*
— *pour chaque $i : G_0$, d'un type $D_0(i)$*
— *pour chaque arrête $g : G_1(i, j)$, d'une fonction $D_1(g) : D_0(i) \to D_0(j)$*

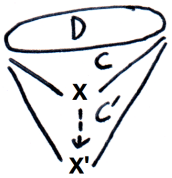
Les preuves de commutation du foncteur sont "gratuites" car la forme du diagramme est une catégorie libre.
Un cocône en X sur un diagramme D est une transformation naturelle entre D et le diagramme constant en X :

Définition 3. *Soit D un diagramme et X un type. Un* cocône *en X consiste en la donnée :*
— *pour chaque $i : G_0$, d'une fonction $D_0(i) \to X$*
— *pour chaque arrête $g : G_1(i, j)$, d'une preuve de $q_j \circ D_1(g) = q_i$*
On note $\mathbf{cocone}_D(X)$ le type des cocônes en X sur D.

Soit $C : \mathbf{cocone}_D(X)$ un cocône et f une fonction de $X \to Y$. On peut alors étendre C en un cocône en Y en postcomposant par f de façon canonique. Cela nous donne une fonction

$$\mathbf{postcompose} : (\mathbf{cocone}_D\, X) \to (X \to Y) \to (\mathbf{cocone}_D\, Y)$$

Grâce à cette notion d'extension de cocône, on peut formuler la propriété universelle des colimites. Un cocône C est dit universel si pour tout autre cocône sur le même diagramme, ce dernier peut être obtenu de façon unique par extension de C.



Une façon élégante de le formuler cela en théorie des types est la suivante.

Définition 4. *Un cocône $C : \mathbf{cocone}_D(X)$ est dit* universel *si la fonction $\mathbf{postcompose}_C : (X \to Y) \to (\mathbf{cocone}_D\, Y)$ est une équivalence pour tout Y.*

Définition 5. *Soit D un diagramme et Q un type. On dit que Q est* une colimite *de D s'il existe un cocône universel en Q sur D.*

Nous montrons ci-dessous que les colimites de graphes existent et sont uniques à équivalence près.

1.2 Existence

Comme cela est présenté dans [AKL15] les limites de diagrammes en théorie des types sont simplement des types existentiels Σ. Les limites existent donc dans n'importe quelle théorie des types avec Σ. Pour les colimites, l'existence des colimites est donnée par les HIT. En cela, on peut voir les HIT comme un dual des types Σ.
On a déjà donné le HIT du coégaliseur de deux fonctions. Cela se généralise à un diagramme D par :

```
Inductive colimit (D: diagram) : Type :=
  | colim : forall (i: G_0), D_0(i) -> colimit
  | eq : forall (i, j: G_0) (g: G_1(i,j)) (x: D_0(i)),
                                  colim j (D_1(g) x) = colim i x
```

On montre que **colimit** D est bien une colimite de D au sens défini précédemment (vérification directe). Ainsi les colimites de diagrammes sur des graphes existent.

1.3 Fonctorialité et Unicité

Morphisme de diagrammes Tout d'abord, il nous faut nous donner une notion de morphisme entre diagrammes. Un diagramme étant un foncteur, un morphisme de diagrammes est naturellement ... une transformation naturelle !

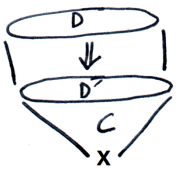
Définition 6. *Soient D et D' deux diagrammes sur le même graphe G. Un* morphisme de diagrammes $m : D \Rightarrow D'$ *entre D et D' consiste en la donnée :*
- *pour tout $i : G_0$, d'une fonction $m_i : D_0(i) \to D'_0(i)$*
- *pour tout $g : G_1(i, j)$, d'une preuve de $D'_1(g) \circ m_i = m_j \circ D_1(g)$*

On a une notion de composition et de morphisme identité évidente sur les morphismes de diagramme.
On dit qu'un morphisme de diagrammes m est une *équivalence de diagramme* si toutes les fonctions m_i sont des équivalences. On peut alors définir l'inverse de ce morphisme (en renversant les preuves de commutation des diagrammes). On vérifie que l'inverse d'un morphisme est bien l'inverse pour la composition de morphismes de diagrammes.

Précomposition On a déjà défini la postcomposition d'un cocône par une fonction (extension d'un cocône par l'avant). On peut maintenant définir la précomposition par un morphisme de diagrammes (extension par l'arrière).

Définition 7. *Étant donné un morphisme de diagrammes $m : D \Rightarrow D'$. On peut transformer tout cocône sur D' en un cocône sur D en le précomposant par m. Cela nous donne une fonction*

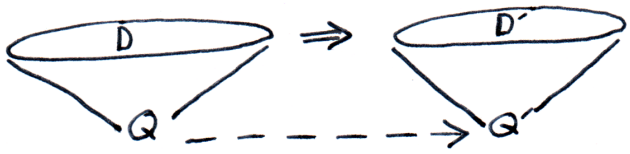
$$\textbf{precompose} : (D \Rightarrow D') \to (\textbf{cocone}_{D'}\, X) \to (\textbf{cocone}_{D}\, X)$$



On vérifie que la précomposition et la postcomposition respectent l'identité et la composition (4 lemmes). On peut alors montrer que les notions d'universalité et de colimite sont stables par postcomposition et précomposition par équivalence.

Fonctorialité de la colimite Soit $m : D \Rightarrow D'$ un morphisme de diagrammes et soient Q et Q' deux colimites de D et D' et C et C' les cocônes universels respectifs. On obtient alors une fonction de $Q \to Q'$ donnée par :

$$\mathbf{postcompose}_C{}^{-1}\,(\mathbf{precompose}_m\ C')$$



On vérifie que si m est une équivalence de diagramme alors la fonction de $Q' \to Q$ donnée par m^{-1} est bien un inverse de $Q \to Q'$. Par conséquent l'on a :

Lemme 1. *Les colimites de deux diagrammes équivalents sont équivalentes.*

Unicité En particulier, en considérant le morphisme de diagrammes identité $D \Rightarrow D$ on obtient :

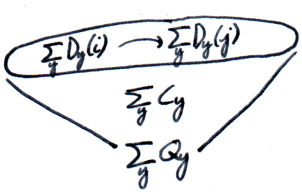
Théorème 1. *Soient Q_1 et Q_2 deux colimites d'un même diagramme, alors :*

$$Q_1 \simeq Q_2\,.$$

Autrement dit, la colimite d'un diagramme est unique à équivalence près.
Remarque : Avec l'axiome d'univalence, la colimite d'un diagramme est réellement unique.

1.4 Commutation aux sigmas

Dans cette sous partie on énonce un théorème non trivial de commutation des colimites avec les sigma.
Soient Y un type et, pour tout $y : Y$, D^y un diagramme sur un graphe G. On peut alors construire un diagramme sur G dont les objets sont les $\Sigma_y D_0^y(i)$ et dont les fonctions $\Sigma_y D_0^y(i) \to \Sigma_y D_0^y(j)$ sont les fonctions induites par l'identité sur la première composante et par les fonctions $D_1^y(g) : D_0^y(i) \to D_0^y(j)$ sur la seconde. Notons ΣD ce diagramme (on dira que ΣD est un Σ-*diagramme*).
De même on peut transformer une famille de cocône $C : \Pi_y\, \mathbf{cocone}_{D^y}(Q_y)$ en un cocône sur ΣD en $\Sigma_y Q_y$.



On a alors le résultat suivant :

Théorème 2. *Si pour tout $y : Y$, Q_y est une colimite de D_y, alors $\Sigma_y Q_y$ est une colimite de ΣD.*

On ne détaille pas la preuve ici : celle-ci est relativement directe. Se reporter à la formalisation pour les détails.
Ce résultat se révélera particulièrement utile (cf. 2.2).

Commutation aux produits Les produits étant des sommes dépendantes constantes on a la conséquence directe suivante.

Théorème 3. *Soient D un diagramme sur G et A un type. On a un diagramme $A \times D$ dont les objets sont les $A \times D_0(i)$ et les flèches, celles induites par D avec l'identité sur la première composante.*
Alors pour toute colimite Q de D, $A \times Q$ est une colimite de $A \times D$.

1.5 Formalisation en Coq

Tous les résultats de cette partie ont été formalisés en Coq dans le répertoire `Colimits`. Nous avons emprunté les définitions de diagramme et de morphisme de diagrammes à [AKL15]. Aussi notre fichier `Diagram.v` vient entièrement de leur développement. Les cocônes et colimites sont définies dans le fichier `Colimit.v`. C'est aussi dans ce fichier que l'on définit le HIT et que l'on montre les propriétés de fonctorialité et d'unicité.
Le fichier `Colimit_Sigma.v` contient la formalisation de la commutation aux sigmas, et `Colimit_Prod.v` celle aux produits. Enfin, les fichiers `CoEqualizer.v`, `Loop.v` et `MappingTelescope.v` contiennent des exemples. On montre en particulier que le HIT `Coeq` défini dans la librairie est bien un coégaliseur au sens des colimites sur un graphe.
Remarque : Les définitions de cocône et de morphisme de diagrammes de notre développement sont très légèrement différentes que celles présentées ici : on remplace par exemple $q_j \circ D_1(g) = q_i$ par $\Pi_x\, q_j(D_1(g)(x)) = q_i(x)$. Par extensionnalité fonctionnelle, ces deux types sont équivalents mais ce dernier est plus pratique à utiliser en Coq.

2 Nerf de Čech et axiome de Giraud

Dans cette partie, nous allons exposer notre contribution à la question principale de mon stage, celle de la formulation d'un analogue à l'axiome de Giraud en HoTT. On commence, par expliquer le problème un peu plus en détail.

En théorie des topos (cf. par exemple [MM12]) on a le résultat élémentaire suivant :

Théorème 4. *Soit $f : X \to Y$ un épimorphisme, alors Y est la colimite du diagramme suivant (la* kernel pair *de f) :*

$$X \times_Y X \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} X$$

La théorie des topos a été généralisée aux ∞-topos, on en trouve un présentation dans le livre de Lurie [Lur09]. Cette généralisation est loin d'être triviale. Dans le cas de notre résultat, il faut remplacer la kernel pair par le nerf de Čech.
Le nerf de Čech de $f : X \to Y$ est un objet simplicial U de la forme :

$$\ldots \qquad U_2 \;\substack{\longrightarrow\\ \longrightarrow\\ \longrightarrow}\; U_1 \;\substack{\longrightarrow\\ \longrightarrow}\; X \times_Y X \;\substack{\xrightarrow{\pi_1}\\ \xrightarrow[\pi_2]{}}\; X$$

qui vérifie des conditions supplémentaires sur les U_n ($n \geq 1$) (cf. [Lur09] prop. 6.1.2.11) [6].
L'on a alors :

Théorème 5 ([Lur09] cor. 6.2.3.5)**.** *Pour tout morphisme $f : X \to Y$ surjectif (en un sens que nous ne détaillons pas), Y est la colimite de son nerf de Čech.*

Or, bien que l'on n'en n'ait pas la preuve formelle, la théorie des types homotopiques est pressentie pour être le langage interne des ∞-topos (au même titre que le lambda calcul simplement typé est le langage interne des catégories cartésiennes fermées). On doit donc pouvoir démontrer le théorème 5 en HoTT. Malheureusement, la définition d'un objet simplicial en HoTT est un problème ouvert sur lequel beaucoup de chercheurs ont travaillé et que l'on peut donc considérer comme très difficile. Notre approche consiste à prendre une définition de nerf différente : on remplace l'objet simplicial et les pullback par une application itérée de la kernel pair. Puis l'on montre qu'elle satisfait le résultat souhaité (partie 2.2).
Pourquoi fait-on cela ? Le but à long terme est, comme cela a été présenté par Kevin Quirin [Qui15], d'étendre la faisceautisation à HoTT. La faisceautisation est une technique qui permet, à partir d'un topos existant, de construire un nouveau topos (le topos des faisceaux) vérifiant de nouvelles propriétés. On peut par exemple construire un topos booléen qui valide la loi du tiers exclu ou encore construire un topos qui invalide l'hypothèse du continu (le travail initial de Cohen utilise la méthode du forcing mais il peut être reformulé en terme de faisceautisation). Cette technique est présentée dans le livre de Mac-Lane et Moerdijk [MM12] et nous cherchons à l'adapter à HoTT. La preuve utilise le théorème 4, c'est donc pour cela que nous avons besoin d'un analogue en HoTT.

2.1 La troncation comme une colimite

Dans cette sous-partie nous décrivons un résultat de Floris van Doorn [vD15] sur lequel nous nous basons.
Une tentative de définition de la troncation propositionnelle d'un type X est :

```
Inductive T X :=
   | alpha : X -> T X
   | beta : forall x x': X, alpha(x) = alpha(x').
```

Malheureusement, $\mathbf{T}(X)$ n'est en général pas une proposition, le constructeur de chemins **beta** n'est pas assez fort. Pour rappel, la définition de troncation qui fonctionne est :

6. Remarque : on n'a pas dessiné les dégénérescences $U_n \to U_{n+1}$ sur le diagramme mais elles sont bien présentes.

```
Inductive Tr X :=
   | tr : X -> Tr X
   | tr_eq : forall x x': Tr X , x = x'.
```

Le constructeur **tr_eq** est bien plus fort que **beta** ce qui force le type engendré à être une proposition.
Floris van Doorn a exploré la différence entre ces deux HIT et a réussi à exprimer **Tr** en fonction de **T**. Plus précisément, il a montré le théorème suivant :

Théorème 6. *Soit X un type. Considérons le diagramme suivant :*

$$X \xrightarrow{\alpha} \mathbf{T}\,X \xrightarrow{\alpha} \mathbf{T}\,(\mathbf{T}\,X) \xrightarrow{\alpha} \mathbf{T}^3\,X \longrightarrow \dots$$

Alors la colimite de ce diagramme est $\mathbf{Tr}(X)$.

2.2 Définition

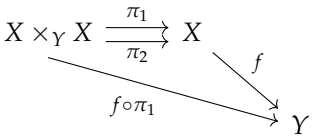
Nous allons généraliser le résultat précédent à une fonction $f : X \to Y$ quelconque. On retrouvera le théorème 6 en considérant l'unique fonction $X \to \mathbf{1}$ (le type *Unit* **1** est un objet terminal).
Soit $f : X \to Y$. On note $\mathbf{KP}(f)$ la colimite de la kernel pair de f :

$$X \times_Y X \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} X$$

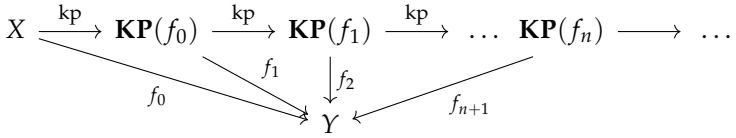
où le pullback $X \times_Y X$ est donné par $\Sigma_{x,x'} f(x) = f(x')$. $\mathbf{KP}(f)$ est donc équivalent au HIT suivant :

```
Inductive KP f :=
  | kp : X -> KP f
  | kp_eq : forall x x', f(x) = f(x') -> kp(x) = kp(x').
```

En considérant le cocône suivant :



on obtient une fonction $\mathbf{lift}_f : \mathbf{KP}(f) \to Y$ par universalité de la colimite.
On peut alors construire le diagramme suivant par induction :



où $f_0 := f : X \to Y$ et $f_{n+1} := \mathbf{lift}_{f_n} : \mathbf{KP}(f_n) \to Y$.

Nous allons montrer le résultat suivant :

Théorème 7. *La colimite de ce diagramme est* $\Sigma_{y:Y} \mathbf{Tr}(\mathbf{fib}_f\, y)$, l'image *de* f.

2.3 Preuve

On va montrer que le diagramme que l'on considère est équivalent à un autre diagramme dont la colimite est visiblement l'image de f. On passe par un diagramme intermédiaire :

$$\begin{array}{ccccccccc}
X & \xrightarrow{\mathrm{kp}} & \mathbf{KP}(f_0) & \xrightarrow{\mathrm{kp}} & \mathbf{KP}(f_1) & \xrightarrow{\mathrm{kp}} & \dots & \mathbf{KP}(f_n) & \dots \\
\wr\downarrow{\scriptstyle e_0} & & \wr\downarrow{\scriptstyle e_1} & & \wr\downarrow{\scriptstyle e_2} & & & \wr\downarrow{\scriptstyle e_{n+1}} & \\
\Sigma_y \mathbf{fib}_{f_0}\, y & \xrightarrow{\Sigma\overline{\mathrm{kp}}} & \Sigma_y \mathbf{fib}_{f_1}\, y & \xrightarrow{\Sigma\overline{\mathrm{kp}}} & \Sigma_y \mathbf{fib}_{f_2}\, y & \xrightarrow{\Sigma\overline{\mathrm{kp}}} & \dots & \Sigma_y \mathbf{fib}_{f_{n+1}}\, y & \dots \\
\wr\downarrow{\scriptstyle \Sigma h_0} & & \wr\downarrow{\scriptstyle \Sigma h_1} & & \wr\downarrow{\scriptstyle \Sigma h_2} & & & \wr\downarrow{\scriptstyle \Sigma h_{n+1}} & \\
\Sigma_y \mathbf{fib}_{f_0}\, y & \xrightarrow{\Sigma\alpha} & \Sigma_y\, \mathbf{T}(\mathbf{fib}_{f_0}\, y) & \xrightarrow{\Sigma\alpha} & \Sigma_y\, \mathbf{T}^2(\mathbf{fib}_{f_0}\, y) & \xrightarrow{\Sigma\alpha} & \dots & \Sigma_y\, \mathbf{T}^{n+1}(\mathbf{fib}_{f_0}\, y) & \dots
\end{array}$$

où $\overline{\mathrm{kp}} : \mathbf{fib}_{f_n} \to \mathbf{fib}_{f_{n+1}}$ est défini par $\lambda(x;\, p).\, (\mathrm{kp}x;\, p)$ et où $\Sigma\overline{\mathrm{kp}}$ et $\Sigma\alpha$ sont les fonctions induites par $\overline{\mathrm{kp}}$ et α sur les secondes composantes des types Σ.

1. Tout d'abord, l'image de f est bien la colimite de la troisième ligne. En effet, d'après le résultat de Floris (théorème 6), $\mathbf{Tr}(\mathbf{fib}_f\, y)$ est la colimite du diagramme

$$\mathbf{fib}_{f_0}\, y \xrightarrow{\alpha} \mathbf{T}(\mathbf{fib}_{f_0}\, y) \xrightarrow{\alpha} \mathbf{T}^2(\mathbf{fib}_{f_0}\, y) \xrightarrow{\alpha} \dots\ \mathbf{T}^{n+1}(\mathbf{fib}_{f_0}\, y)\ \dots$$

pour tout $y : Y$. D'où le résultat par la commutation des colimites aux sigmas (théorème 2).

2. Montrons maintenant que les deux premières lignes du diagramme sont équivalentes. Cette équivalence est relativement facile, elle est vraie pour n'importe quel diagramme de cette forme. Il s'agit en fait d'appliquer l'équivalence $A \simeq \Sigma_{b:B}\, \mathbf{fib}_g\ b$ pour $g : A \to B$ (cf. prélude) à tous les types du diagramme. On définit donc les e_n par $e_n := \lambda x.\, (f_n(x);\, (x;\, 1))$, ce sont clairement des équivalences. Le n^{e} carré commute car $e_{n+1} \circ \mathrm{kp}$ et $\Sigma\overline{\mathrm{kp}} \circ e_n$ sont égales sur la deuxième composante de $\Sigma_y\, \mathbf{fib}_{f_{n+1}}\, y$, ce qui suffit pour qu'elles soient égales.

3. Pour finir montrons l'équivalence entre la deuxième et la troisième ligne. On remarque tout d'abord que, comme ce sont deux Σ-diagrammes, il suffit de montrer qu'ils sont équivalents point à point. Fixons $y : Y$ et montrons l'équivalence :

$$\begin{array}{ccccccccc}
\mathbf{fib}_{f_0}\, y & \xrightarrow{\overline{\mathrm{kp}}} & \mathbf{fib}_{f_1}\, y & \xrightarrow{\overline{\mathrm{kp}}} & \mathbf{fib}_{f_2}\, y & \xrightarrow{\overline{\mathrm{kp}}} & \dots & \mathbf{fib}_{f_{n+1}}\, y & \dots \\
\wr\downarrow{\scriptstyle h_0} & & \wr\downarrow{\scriptstyle h_1} & & \wr\downarrow{\scriptstyle h_2} & & & \wr\downarrow{\scriptstyle h_{n+1}} & \\
\mathbf{fib}_{f_0}\, y & \xrightarrow{\alpha} & \mathbf{T}(\mathbf{fib}_{f_0}\, y) & \xrightarrow{\alpha} & \mathbf{T}^2(\mathbf{fib}_{f_0}\, y) & \xrightarrow{\alpha} & \dots & \mathbf{T}^{n+1}(\mathbf{fib}_{f_0}\, y) & \dots
\end{array}$$

On pose $h_0 := \mathbf{id}$. On remarque que l'on a ensuite l'équivalence suivante :

Lemme 2. *Pour $f : X \to Y$ on a :*

$$\mathbf{KP}\ f\ \simeq\ \Sigma_y\, \mathbf{T}(\mathbf{fib}_f\, y)$$

.

En effet **KP** f est une colimite du diagramme $X \times_Y X \rightrightarrows X$ et $\Sigma_y \mathbf{T}(\mathbf{fib}_f\, y)$ est une colimite de $\Sigma_y(\mathbf{fib}_f\, y) \times (\mathbf{fib}_f\, y) \rightrightarrows \Sigma_y\, \mathbf{fib}_f\, y$ (par commutation aux sigmas). Or ces deux diagrammes sont équivalents, d'où l'équivalence (lemme 1).

On peut alors montrer le lemme suivant qui permet construire les h_n par récurrence :

Lemme 3. *Soient $f : X \to Y$ une fonction, A un type, $y : Y$ et $\epsilon : (\mathbf{fib}_f\, y) \simeq A$ une équivalence. Alors il existe une équivalence $\epsilon' : (\mathbf{fib}_{\mathbf{lift}_f}\, y) \simeq \mathbf{T}(A)$ telle que*

$$\begin{array}{ccc} \mathbf{fib}_f\, y & \xrightarrow{\overline{\text{kp}}} & \mathbf{fib}_{\mathbf{lift}_f}\, y \\ {\scriptstyle\epsilon}\downarrow\wr & & \wr\downarrow{\scriptstyle\epsilon'} \\ A & \xrightarrow{\alpha} & \mathbf{T}(A) \end{array}$$

commute.

Si l'on note s l'équivalence donnée par le lemme précédent, alors ϵ' est donnée par la suite d'équivalences :

$$\begin{aligned} \mathbf{fib}_{\mathbf{lift}_f}\, y &\equiv \Sigma_{x:\mathbf{KP}\, f}\, \mathbf{lift}_f(x) = y && (1)\\ &\simeq \Sigma_{x:\, \Sigma_y\, \mathbf{T}(\mathbf{fib}_f\, y)}\ \mathbf{lift}_f \circ s^{-1}(x) \ =\ y && (2)\\ &\simeq \Sigma_{x:\, \Sigma_y\, \mathbf{T}(\mathbf{fib}_f\, y)}\ \pi_1(x) \ =\ y && (3)\\ &\simeq \mathbf{T}(\mathbf{fib}_f\, y) && (4)\\ &\simeq \mathbf{T}(A) && (5) \end{aligned}$$

$(1) \simeq (2)$: par s (lemme 2).
$(2) \simeq (3)$: car l'on peut montrer que $\pi_1 \circ s = \mathbf{lift}_f$
$(4) \simeq (5)$: par fonctorialité de la colimite (si $A \simeq B$ alors $\mathbf{T}(A) \simeq \mathbf{T}(B)$).

2.4 Formalisation en Coq

C'est en formalisant cette partie que nous nous sommes rendu compte que nous avions besoin de toutes les propriétés de fonctorialité des colimites. Ce qui nous a poussé à réécrire presque entièrement notre bibliothèque.
La formalisation de cette partie se trouve dans les fichiers `KernelPair.v` et `CechNerve.v` du répertoire `Colimits`. Le lemme 3 a été le plus difficile à formaliser.

3 Structure de modèle

Cette partie est un peu orthogonale aux deux autres. Dans celle-ci nous cherchons à comprendre la structure d'homotopie de **Type**. Pour cela, on définit les fibrations et les cofibrations en HoTT de façon à munir **Type** d'une structure de modèle. Une structure de modèle est un moyen de munir une catégorie d'une structure d'homotopie. Cela permet par exemple de définir les colimites homotopiques, qui sont en fait les colimites que nous considérons en HoTT.

3.1 Homotopy Type System

Pour définir une structure de modèle nous avons besoin d'utiliser une égalité définitionnelle. Nous allons donc travailler dans le système de type Homotopy Type System (HTS) proposé par Voevodsky [Voe13]. Dans ce système, deux égalités cohabitent :

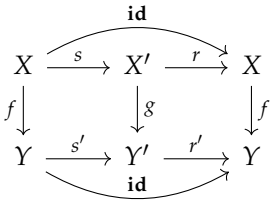
1. une égalité définitionnelle, que nous noterons $\equiv$, qui vérifie l'extensionnalité fonctionnelle et UIP ("Uniqueness of identity proof")
2. une égalité propositionnelle, que nous noterons $=$, qui vérifie l'univalence.

Dans ce système, il faut un mécanisme pour empêcher que les deux égalités coïncident car l'univalence et UIP sont contradictoires. Pour cela, on introduit un nouveau jugement indiquant qu'un type est fibrant et l'on n'autorise à détruire une égalité propositionnelle que sur un type fibrant. Ainsi on a toujours $a \equiv b \to a = b$ mais pas la réciproque en général. On renvoie à [Voe13] pour les détails (bien que cette note soit un peu cryptique).
Ce qui est important pour nous, c'est que **Type** forme une catégorie pour l'égalité définitionnelle : les objets de **Type** sont les types qui l'habitent et les flèches sont les fonctions entre ces types. On a bien l'associativité et les identités pour l'égalité $\equiv$ (par exemple $f \circ \mathbf{id} \equiv \mathbf{id} \circ f \equiv f$).
Maintenant que l'on voit **Type** comme une catégorie, on s'intéresse à la structure d'homotopie sur elle qui est induite par l'égalité $=$, puis aux conséquences que cela a sur les colimites en HoTT.

3.2 Catégories de modèle

Une structure de modèle est une façon classique de donner une structure d'homotopie sur une catégorie.
Dans la suite $\mathcal{C}$ est une catégorie. On a besoin de quelques définitions avant de pouvoir définir une structure de modèle.

Définition 8. *Soit $g : X' \to Y'$ une flèche de $\mathcal{C}$. On dit que $f : X \to Y$ est un* rétract *de g s'il existe des flèches s, r, s' et r' telles que le diagramme suivant commute (pour l'égalité $\equiv$) :*



Petit point de terminologie : Dans une catégorie, un objet B est un *rétract* d'un objet A s'il existe deux flèches $r : A \to B$ et $s : B \to A$ telles que $r \circ s = \mathbf{id}$ (on peut penser à r comme une surjection). On dit alors que r est une *rétraction* de s et s une *section* de r[7] .
Dans le cas de notre définition, X (resp. Y) est un rétract de X' (resp. Y') dans la catégorie **Type**. Et f est un rétract de g dans la catégorie $\mathbf{Type}^{\to}$ où les objets sont les fonctions entre deux types et les flèches les carrés commutatifs entre deux fonctions.

7. http://ncatlab.org/nlab/show/retract

Définition 9. *Une classe S de flèches de $\mathcal{C}$ vérifie* 2-out-of-3 *si, pour toutes flèches $X \xrightarrow{f} Y \xrightarrow{g} Z$ telles que deux des flèches f, g et $g \circ f$ sont dans S, la troisième l'est aussi.*

Définition 10. *Étant données deux flèches $f : X \to Y$ et $g : X' \to Y'$, on dit que f possède la "left lifting property"* [8] *(LLP) par rapport à g et que g possède la"right lifting property" (RLP) par rapport à f si pour toutes flèches $F : X \to X'$ et $G : Y \to Y'$ telles que $G \circ f = g \circ F$ il existe γ tel que le diagramme suivant commute (pour $\equiv$) :*

$$\begin{array}{ccc} X & \xrightarrow{F} & X' \\ {\scriptstyle f}\downarrow & \overset{\gamma}{\nearrow} & \downarrow{\scriptstyle g} \\ Y & \xrightarrow{G} & Y' \end{array}$$

On dit qu'une fonction f possède la LLP (resp. RLP) par rapport à une classe de fonction S si elle la possède par rapport à toutes les flèches de cette classe. On note LLP(S) (resp. RLP(S)) la classe des telles flèches.

Définition 11. *Un* système de factorisation faible *sur $\mathcal{C}$ consiste en la donnée de deux classes de flèches L et R telles que :*

1. *toute flèche f de $\mathcal{C}$ peut être factorisée comme $f = r \circ l$ avec $l \in L$ et $r \in R$*
2. *L est exactement la classe de toutes les flèches de $\mathcal{C}$ ayant LLP par rapport à R : $L = LLP(R)$*
3. *R est exactement la classe de toutes les flèches de $\mathcal{C}$ ayant RLP par rapport à L : $R = RLP(L)$*

Définition 12. *Une structure de modèle sur $\mathcal{C}$ consiste en la donnée de trois classes de morphismes F, C et W (les fibrations, les cofibrations et les équivalences faibles) telles que :*

1. *W vérifie 2-out-of-3*
2. *(AC, F) et (C, AF) sont des systèmes de factorisation faibles, où $AC = C \cap W$ et $AF = F \cap W$.*

3.3 Une structure de modèle sur Type

On a vu au 3.1 que **Type** forme une catégorie (pour $\equiv$), le but est de donner une structure de modèle sur cette catégorie.
Dans l'article [GG08], Gambino et Garner proposent un système de factorisation faible (cofibrations acycliques, fibrations) pour la théorie des types. Dans notre formalisme c'est le suivant :

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ & {\scriptstyle \lambda x.\,(f(x);\,(x;\mathbf{refl}_{f(x)}))} \searrow \quad \nearrow {\scriptstyle \pi_1} & \\ & \Sigma_y\, \mathbf{fib}_f\, y & \end{array} \tag{6}$$

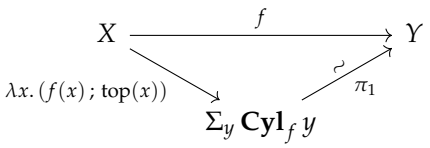
Gambino et Garner proposent aussi une caractérisation simple des fibrations et des cofibrations acycliques. Par contre, ils ne donnent pas l'autre système de

8. on se garde de toute traduction hasardeuse…

factorisation car celui-ci nécessite la présence de HIT. Leur travail a été poursuivi par Lumsdaine dans [Lum11] où il introduit le système de factorisation dual grâce à un HIT. Malheureusement les définitions de cofibrations et fibrations acycliques qu'il propose ne sont pas du tout descriptives, il est même précisé : « A [...] characterisation of cofibrations would be very nice, but seems elusive ! ». Nous proposons dans la suite une telle description des cofibrations et des fibrations acycliques.
On définit le *cylindre* d'une fonction f comme le type inductif suivant :

```
Inductive Cyl (f: X -> Y) : Y -> Type :=
  | top : forall x, Cyl f (f x)
  | base : forall y, Cyl f y
  | cyl_eq : forall x, top x = base (f x).
```

On a alors une autre factorisation :

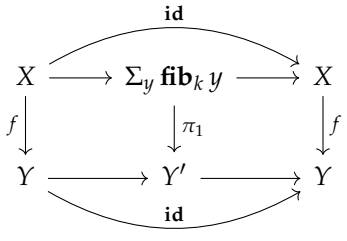


(7)

Le cylindre est l'analogue des "mapping cylinder" [9] en théorie de l'homotopie. C'est un dual de la fibre dans le sens suivant : $\Sigma_y \mathbf{fib}_f\, y$ est un pullback de f et $\mathbf{id}_Y$ et $\Sigma_y \mathbf{Cyl}_f\, y$ est un pushout [10] de f et $\mathbf{id}_X$.

Définition 13. *On définit les classes W, F, C, AC et AF de la manière suivante.*

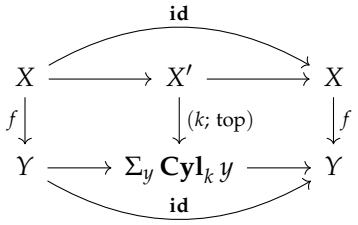
1. *On dit que* $f : X \to Y$ *est une* équivalence faible *si l'on a* $\mathbf{IsEquiv}\, f$.
2. *On dit que* $f : X \to Y$ *est une* fibration *s'il existe* $k : X' \to Y'$ *telle que* f *soit un rétract de* $\pi_1 : \Sigma_y \mathbf{fib}_k\, y \to Y'$.



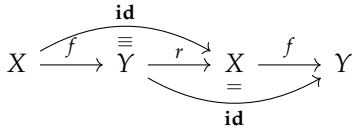
3. *On dit que* $f : X \to Y$ *est une* cofibration *s'il existe* $k : X' \to Y'$ *telle que*

9. https://en.wikipedia.org/wiki/Mapping_cylinder
10. cf. lemme sig_cyl_rec dans le fichier FibCofib.v

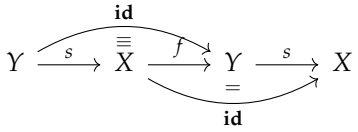
f soit un rétract de $\lambda x.\,(k(x);\ \mathrm{top}(x)) : X' \to \Sigma_y\, \mathbf{Cyl}_k\, y$.



4. *On dit que $f : X \to Y$ est une* cofibration acyclique *si c'est une* équivalence injective, *c'est-à-dire s'il existe $r : Y \to X$ telle que l'on ait :*
   — *pour tout x, $r(f(x)) \equiv x$*
   — *$\epsilon : \Pi_y,\ f(r(y)) = y$*
   — *pour tout x, $\epsilon_{f(x)} \equiv \mathbf{refl}_{f(x)}$.*



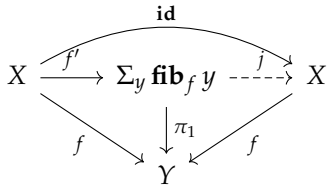
5. *On dit que $f : X \to Y$ est une* fibration acyclique *si c'est une* équivalence surjective, *c'est-à-dire s'il existe $s : Y \to X$ telle que s l'on ait :*
   — *pour tout x, $f(s(x)) \equiv x$*
   — *pour tout y, $s(f(y)) = y$.*



Remarque : contrairement aux équivalences injectives, on ne requiert pas de compatibilité des égalités dans la définition des équivalences surjectives. Nous ne comprenons pas encore très bien cette dissymétrie flagrante.

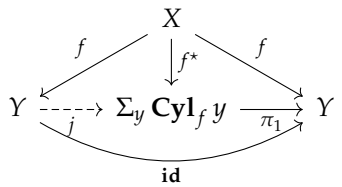
On peut donner une caractérisation des fibrations et des cofibrations qui est plus pratique dans certains cas :

Lemme 4. *Une fonction $f : X \to Y$ est une fibration si et seulement si il existe j faisant commuter (pour $\equiv$) :*



(le triangle de gauche commute toujours) où f' représente $\lambda x.\,(f(x);\ (x; \mathbf{refl}_{f(x)}))$.

Une fonction $f : X \to Y$ est une cofibration si et seulement si il existe j faisant commuter (pour $\equiv$) :



(le triangle de droite commute toujours) où $f^\star$ représente $\lambda x.\,(f(x);\ \mathrm{top}(x))$.

Autrement dit (cas des fibrations), si une fonction est un rétract d'un π_1 alors c'est un rétract de "son" π_1.
Remarque (cas des fibrations) : pour une fonction f quelconque on a toujours un candidat pour j : la fonction π_2 . Celle-ci fait bien commuter le triangle du haut mais ne fait en général pas commuter définitionnellement le triangle de droite (seulement propositionnellement).

Théorème 8. *On a $AF = F \cap W$ et $AC = C \cap W$. Et les classes (W, F, C) forment une structure de modèle sur Type.*

On ne détaille pas la preuve ici, la formalisation est disponible dans le fichier `FibCoFib.v`.

3.4 Justification de certaines colimites

Une des utilités de la structure de modèle pourrait être de justifier certains calculs de colimite. En effet, nous avons vu que notre définition de colimite n'est valable que sur des graphes. Mais même sur des graphes, comment justifier que l'on a bien la bonne définition de colimite ? Cela sont des travaux en cours, nous n'en présentons que les grandes idées.
Les colimites que nous souhaitons considérer en HoTT sont en fait des colimites homotopiques. En effet, les colimites "strictes" ne se comportent pas bien vis à vis de l'égalité propositionnelle $=$: par exemple, elles ne sont pas invariantes par équivalence.
En théorie des catégories homotopique, la définition de colimites est assez compliquée : un foncteur **hocolim** : $R \to$ **Type** est bien le foncteur "colimite homotopique" si c'est un foncteur dérivé à gauche du foncteur "colimite stricte" **colim** : $R \to$ **Type**. On retrouvera les détails dans [Hir03] par exemple.
Un des cas très simple est celui où la forme R du diagramme considérée est une catégorie munie d'une structure de Reedy (il s'agit en gros d'un ordre bien fondé sur les objets de la catégorie qui permet de faire des inductions) et où le foncteur **colim** est Quillen à droite. Dans ce cas, le foncteur **hocolim** est le simplement le foncteur **colim** appliqué aux remplacement cofibrant du diagramme. On peut calculer ce remplacement cofibrant (grâce à la structure de modèle), ce qui nous donne alors une formule pour la colimite homotopique en fonction de la colimite stricte.
Pour que le foncteur **colim** soit Quillen à droite, il semble qu'il suffise que la forme du diagramme soit un graphe dirigé sans cycle. Nous espérons donc appliquer cette méthode aux coégaliseurs, au pushout, au mapping telescope (la forme de diagramme que nous avons utilisé au 2.2), ...

3.5 Formalisation en Coq

Pour formaliser cette partie nous aurions besoin d'un assistant de preuve prenant en charge le système de type HTS. Bien que des prototypes soient en cours d'expérimentation [11], il n'en n'existe pas, à notre connaissance, de suffisamment mature pour nos travaux. Nous avons donc décidé d'expérimenter ce système de type avec Coq. Pour cela nous définissons une nouvelle égalité $\equiv$ (avec une famille inductive, comme dans la librairie standard de Coq par exemple) et supposons par axiome qu'elle vérifie UIP. Il nous faudrait alors un mécanisme de types fibrants, sans lequel le système est inconsistant. Pour l'instant nous nous passons d'un tel mécanisme et vérifions "à la main" que nous n'éliminons les égalités propositionnelles que sur des types fibrants, bien sûr ce n'est pas durable mais cela nous permet tout de même d'expérimenter un peu ce nouveau système de type.
La nouvelle égalité est définie dans le répertoire `StrictEq`, ainsi que les premiers lemmes de transports (nous avons repris la structure de la bibliothèque HoTT). Les parties 3.2 et 3.3 sont formalisées dans le fichier `FibCoFib.v`.

Conclusion

Comme nous l'avons déjà précisé dans synthèse nous avons : d'une part, développé un petit peu de théorie en HoTT, en formalisant les colimites sur des graphes. Cela s'inscrit dans la continuité directe du programme de fondation univalente des mathématiques proposé par Voevodsky. Et d'autre part, cherché à comprendre toujours un peu mieux la structure d'homotopie de **Type** en donnant une description des fibrations et cofibrations en HoTT et en explorant la notion de nerf de Čech.
Ce stage a aussi été l'occasion pour moi de commencer à rencontrer la communauté travaillant sur HoTT : j'ai pu participer au workshop HoTT/UF – RDP 2015 [12] et j'ai pu discuter plusieurs fois avec Egbert Rijke, venu collaborer avec l'équipe à Nantes. C'est notamment lui qui nous a indiqué l'existence du résultat de Floris van Doorn et c'est avec lui que nous avons défini les cofibrations, nous le remercions chaleureusement.
C'est avec grand plaisir que je poursuivrai ces investigations en thèse. Nous commencerons par finir nos travaux sur la justification des colimites. Puis nous nous intéresserons aux modèles, en théorie des types, de la théorie des types.

11. cf. par exemple Andromeda `https://github.com/andrejbauer/andromeda/`
12. `http://hott-uf.gforge.inria.fr/`

Références

[AKL15] Jeremy Avigad, Krzysztof Kapulkin, and Peter LeFanu Lumsdaine. Homotopy limits in type theory. *Mathematical Structures in Computer Science*, 25(5) :1040–1070, 2015.

[BCH14] Marc Bezem, Thierry Coquand, and Simon Huber. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26, pages 107–128, 2014.

[GG08] Nicola Gambino and Richard Garner. The identity type weak factorisation system. *Theor. Comput. Sci.*, 409(1) :94–109, 2008.

[Hir03] Philip S. Hirschhorn. *Model categories and their localizations*, volume 99 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2003.

[Lum11] Peter LeFanu Lumsdaine. *Model Structures from Higher Inductive Types*. Notes non publié, `http://peterlefanulumsdaine.com/research/Lumsdaine-Model-strux-from-HITs.pdf`, 2011.

[Lur09] Jacob Lurie. *Higher topos theory*. Number 170. Princeton University Press, 2009.

[MM12] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic : A first introduction to topos theory*. Springer Science & Business Media, 2012.

[Qui15] Kevin Quirin. *Lawvere-Tierney Sheafification in Homotopy Type Theory*. Workshop on HoTT / UF, collocated with RDP/TRA/TLCA 2015, `http://hott-uf.gforge.inria.fr/complete_HoTTUF15.pdf`, 2015.

[Soj15] Kristina Sojakova. Higher inductive types as homotopy-initial algebras. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 31–42. ACM, 2015.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory : Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[vD15] Floris van Doorn. *Constructing the Propositional Truncation using Nonrecursive HITs*. Blog Homotopy Type Theory, `http://homotopytypetheory.org/2015/07/28/constructing-the-propositional-truncation-using-nonrecursive-hits/`, 2015.

[Voe13] Vladimir Voevodsky. *A simple type system with two identity types*. Notes non publiées, `http://uf-ias-2012.wikispaces.com/file/view/HTS.pdf`, 2013.