Rapport de stage : Groupes de cohomologie en théorie des types univalente

Elies Harington

11 août 2020

Table des matières

J'ai réalisé mon stage à distance pendant la pandémie de 2020 entre mars et juillet sous la supervision de Thierry Coquand. Je tiens à le remercier particulièrement pour nos discussions régulières en période de confinement, qui m'ont aidé à avancer malgré l'isolement et la distance. Je tiens également à remercier Anders Mörtberg pour m'avoir encouragé à participer au développement de la bibliothèque Cubical Agda.

1 Introduction

En 1966, Jean Giraud publie son ouvrage "Cohomologie non abélienne" [2]. Il y introduit la notion de champs (en anglais : stack), une généralisation des faisceaux, et montre que les groupes de cohomologie de degrés 1 et 2 peuvent être décrits par des objets appelés *torseurs* et *gerbes*.
Le langage de la *Théorie Homotopique des Types* (abrégée THoT dans la suite, HoTT en anglais pour Homotopy Type Theory), qui s'est grandement développé cette dernière décennie, offre une manière synthétique de parler d'espaces/∞-groupoïdes et d'homotopie. Le but de mon stage était d'étudier et de "traduire" certaines notions définies par Jean-Giraud dans le cadre de cette théorie, en les implémentant dans l'assistant de preuve formelle *Cubical Agda*.
Ce stage fait suite à celui de Victor Alfieri en 2019 qui portait essentiellement sur le premier groupe de cohomologie, en se basant sur un paragraphe de l'article "le Symbole Modéré" de Pierre Deligne [1]. J'ai pris comme référence les paragraphes 5.2 et 5.3 de cet article, où Deligne résume certaines définitions données dans le livre de Giraud, et ait implémenté en Agda les définitions des premier et deuxième groupes de cohomologie et prouvé différents résultats à leurs propos.
Le langage de la théorie homotopique des types semble particulièrement adapté aux notions étudiées, si ce n'est même plus que celui utilisé originellement par Jean-Giraud : là où la théorie des champs repose sur des définitions complexes à base de catégories supérieures, les mêmes définitions en THoT sont directes de par l'interprétation homotopique de la théorie. Cela suggère à l'avenir de possibles généralisations aux groupes de cohomologie supérieurs, qui ne nécessiterait pas les définitions laborieuses qui viennent avec les analogues supérieurs des faisceaux et des champs.

Dans ce rapport, je commence par donner une présentation de la théorie homotopique des types et de ses concepts fondamentaux, puis je détaille les principales constructions que j'ai étudiées lors de mon stage. Je ne détaille pas le langage des faisceaux et des champs, ce qui aurait pris beaucoup trop de place dans ce rapport déjà long. Puisque tout le contenu de mon stage est centré sur le langage de la THoT, faire une présentation de cette-dernière m'a paru nécessaire, et le dépassement de la limite du nombre de pages inévitable. J'espère néanmoins que ce niveau de détail rendra la lecture plus agréable pour les non-initiés.

2 Cadre théorique

Dans cette section, je donne une rapide présentation de la Théorie Homotopique des Types et de ses concepts essentiels, en essayant de donner un maximum d'intuitions topologiques. La THoT est une surcouche d'une théorie dite des *types dépendants*, dûe à Martin-Löf, que nous étudions donc en premier.

2.1 Théorie des types dépendants de Martin-Löf

2.1.1 Syntaxe, types dépendants

La théorie des types dépendants est une théorie logique au grand pouvoir expressif. En théorie des ensembles, les objets fondamentaux sont les ensembles, qui sont comparés par les relations d'appartenance $\in$ et d'égalité $=$. En théorie des types, les objets fondamentaux sont les types et leurs éléments. On note $a : A$ pour dire "l'objet a est de type A" ou "a est un élément de A". En théorie des types, tout objet vient accompagné d'un type : on ne peut pas dire "soit a un objet" sans préciser le type de a.
Si A est un type, et que a et b sont des éléments de A, on note $a \equiv b : A$ ou plus simplement $a \equiv b$ pour dire "a et b sont égaux par définition". Deux objets sont égaux par définition s'ils ont la même expression syntaxique modulo certaines équivalences, que je détaille plus bas. Si ϕ est une expression syntaxique, on écrit $f :\equiv \phi$ pour dire qu'on identifie par définition le symbole f à l'expression ϕ.

Étant donné deux types A et B, on peut définir un type $A \to B$ appelé *type des fonctions de A vers B*. Pour construire un élément de $A \to B$, il faut la donnée d'une expression syntaxique ϕ, pouvant contenir une variable x, telle qu'en supposant x de type A, on peut déduire que ϕ est de type B. Cela définit un objet $x \mapsto \phi$ de type $A \to B$ (parfois aussi noté $\lambda x.\phi$).
Prenons un exemple informel : on pose $\phi :\equiv x + 2$. En supposant $x : \mathbb{N}$, on sait que $x \mapsto x + 2$ est de type $\mathbb{N} \to \mathbb{N}$.
Si f est de type $A \to B$, et que a est de type A, alors l'expression $f(a)$ est de type B.
Si $x \mapsto \phi$ est de type $A \to B$, et que a est de type A, alors on identifie par définition $(x \mapsto \phi)(a)$ et $\phi[x := a]$, qui désigne l'expression ϕ où on a substitué toutes les instances de la variable x par l'expression a. Ici je ne détaille pas ce procédé de substitution, qui est en fait relativement complexe car on doit éviter des problèmes de "capture de variable". Il faut simplement retenir que ces définitions collent à l'intuition que l'on se fait de la manière dont fonctionnent les fonctions en mathématiques.
Si f est de type $A \to B$, on identifie aussi par définition f et $x \mapsto f(x)$.

Remarque 2.1. Il faut bien faire la distinction entre ce que j'appelle les *objets* et ce que j'appelle les *expressions syntaxiques*. Par exemple, si A et B sont des types, l'expression syntaxique $x \mapsto x$ peut correspondre à l'objet $x \mapsto x : A \to A$ ou à l'objet $x \mapsto x : B \to B$, qui ne sont pas le même objet dans la théorie.

Dernières remarques de notations : le symbole $\to$ est associatif à droite, donc $A \to B \to C$ se lit $A \to (B \to C)$. Si f est de type $A \to B \to C$, et que $a : A$ et $b : B$, on notera $f(a, b)$ pour $f(a)(b)$ (c'est la curryfication).

Toutes ces considérations très techniques nous amènent à une théorie logique minimaliste : on peut interpréter les types comme des propositions, leurs éléments comme des preuves, et le symbole $\to$ comme l'implication logique. Par exemple, la règle qui dit qu'à une fonction $f : A \to B$ et à un élément $a : A$ on peut associer un élément $f(a) : B$ peut se lire "si on a une preuve de $A \implies B$ et une preuve de A, alors on a une preuve de B".
On peut s'autoriser des constructions supplémentaires, comme par exemple le symbole $\times$, permettant de construire à partir de types A et B le produit cartésien $A \times B$ dont les éléments sont des paires (a, b) avec $a : A$ et $b : B$. D'un point de vue logique, le produit cartésien encode la conjonction : les preuves de $A \wedge B$ sont exactement les paires d'une preuve de A et d'une preuve de B.

Tout cela ne justifie pas encore le terme "dépendant" dans le nom de la théorie. On introduit la notion d'univers : un univers $\mathcal{U}$ est un type particulier, dont les objets sont les types. On serait tenté d'écrire $\mathcal{U} : \mathcal{U}$ car $\mathcal{U}$ est un type, mais cela amènerait à des contradictions dans la logique. Au lieu d'un univers, on se donne donc une hiérarchie infinie $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : ...$ d'univers, dont chacun est contenu dans le suivant, mais aucun dans lui-même. En pratique on écrira juste $\mathcal{U}$ sans préciser l'indice, et $\mathcal{U} : \mathcal{U}$ pour $\mathcal{U}_i : \mathcal{U}_{i+1}$.

Si A est un type, une fonction $B : A \to \mathcal{U}$ peut être vue comme une famille de types indexée par les éléments de A. On généralise alors le type des fonctions à ce qu'on appelle le type des "fonctions dépendantes" ou "produit dépendant" : si A est un type, et $B : A \to \mathcal{U}$ une famille de types indexée par A, on note

$$\prod_{x:A} B(x)$$

le type des fonctions f d'un élément $x : A$ renvoyant un élément $f(x) : B(x)$. Les règles de construction sont analogues à celles vues précédemment. Ici, le type d'arrivée de la fonction dépend bien de l'élément qu'elle prend en entrée.
En interprétant les types comme des propositions, le type du produit dépendant peut s'interpréter comme le "quelque soit" logique. Par exemple, une fonction $P : \mathbb{N} \to \mathcal{U}$ peut s'interpréter comme une proposition dépendant d'un entier, et le type $\prod_{n:\mathbb{N}} P(n)$ correspond à la proposition $\forall n, P(n)$.
On peut aussi donner une version dépendante du produit cartésien, appelée la *somme dépendante* : si A est un type et $B : A \to \mathcal{U}$ une famille de types, on note

$$\sum_{x:A} B(x)$$

le type des paires (a, b) où a est de type A et b de type $B(a)$. Il s'interprète d'un point de vue logique comme le quantificateur existentiel : $\exists a, P(a)$.

2.1.2 Types inductifs

J'ai utilisé plus tôt le type des entiers naturels, sans vraiment expliciter sa construction. Dans cette partie j'introduis la notion de *type inductif*, qui va nous permettre de définir rigoureusement le type des entiers naturels, mais aussi les types *vrai* et *faux*, et d'autres encore. Cette section n'a pas pour but de définir une toute généralité ce que sont les types inductifs, mais seulement de donner suffisamment d'exemples pour se faire une idée.

Dans l'idée, un type inductif est un type définit par des *constructeurs*, et entièrement caractérisés par ces derniers. Commençons par regarder en détail l'exemple des entiers naturels. Il est défini par deux constructeurs, *zéro* et *successeur* :

$$0 : \mathbb{N}$$
$$\mathrm{suc} : \mathbb{N} \to \mathbb{N}$$

Les entiers naturels ont un principe de *récurrence*, qui nous donne une manière de construire des fonctions définies sur les entiers. Étant donné un type A, pour construire une fonction $f : \mathbb{N} \to A$, il suffit de donner un élément $c_0 : A$ et une fonction $c_{\mathrm{suc}} : \mathbb{N} \to A \to A$. f vérifie alors :

$$f(0) :\equiv c_0$$
$$f(\mathrm{suc}(n)) :\equiv c_{\mathrm{suc}}(n, f(n))$$

Par exemple, pour définir la fonction double qui à un entier renvoie son double, il suffit de spécifier

$$c_0 :\equiv 0$$
$$c_{\mathrm{suc}}(n, x) :\equiv \mathrm{suc}(\mathrm{suc}(x))$$

On peut généraliser ce principe de récurrence en énonçant un principe dit *d'induction*, qui spécifie comment obtenir une fontion *dépendante* définie sur les entiers. Si $A : \mathbb{N} \to \mathcal{U}$ est une famille de types indexée par les entiers, pour obtenir une fonction $f : \prod_{n:\mathbb{N}} A(n)$, il suffit de spécifier un élément $c_0 : A(0)$ et une fonction $c_{\mathrm{suc}} : \prod_{n:\mathbb{N}} A(n) \to A(\mathrm{suc}(n))$. La fonction f ainsi définie vérifiera les mêmes équations que précédemment.
Ce principe d'induction caractérise le type des entiers naturels : si un autre type vérifie le même type d'induction, alors on peut construire une unique bijection entre ce dernier et le type des entiers qui soit "compatible" avec les constructeurs 0 et suc.

Les types inductifs sont donc des types spécifiés par des *constructeurs*, et ces constructeurs caractérisent le type à travers des principes de récursion et d'induction. Voyons maintenant d'autres exemples.

Le type $A \times B$ peut être vu comme le type inductif spécifié par le constructeur $(-,-) : A \to B \to A \times B$. On peut alors définir les projections $\mathrm{fst} : A \times B \to A$ et

$\mathrm{snd} : A \times B \to B$ grâce au principe de récurrence en spécifiant :

$$\mathrm{fst}((a,b)) :\equiv a$$
$$\mathrm{snd}((a,b)) :\equiv b$$

On aurait aussi pu définir le type $\sum_{x:A} B(x)$ de manière analogue, avec un constructeur $(-,-) : \prod_{x:A} B(x) \to \sum_{x:A} B(x)$. Le principe de récurrence nous permet alors de définir

$$\mathrm{fst} : \sum_{x:A} B(x) \to A \qquad \mathrm{fst}((a,b)) :\equiv a$$
$$\mathrm{snd} : \prod_{\alpha:\sum_{x:A} B(x)} B(\mathrm{fst}(\alpha)) \qquad \mathrm{snd}((a,b)) :\equiv b$$

Un autre exemple est le type $A+B$ engendré par les constructeurs $\mathrm{inr} : A \to A+B$ et $\mathrm{inl} : B \to A+B$, et qui correspond à la disjonction logique (une preuve de $A+B$ est une preuve de A *ou* une preuve de B). On peut aussi définir le type "vrai", noté $\top$ (lu "top"), ayant un unique constructeur $tt : \top$ et le type "faux", noté $\bot$ (lu "bottom"), caractérisé par le fait qu'il n'ait aucun constructeur. D'un point de vue logique, cela correspond à l'absence de preuve de $\bot$. Le principe de récurrence du type $\bot$ nous dit également que pour prouver $\bot \to A$, il n'y a rien à spécifier. Autrement dit, faux implique tout (*ex falso quodlibet*).

2.1.3 Type identité

Il nous reste encore à aborder la notion d'égalité. On a déjà vu qu'on notait $a \equiv b$ quand deux objets d'un type A était égaux par définition. Mais si l'on veut traiter l'égalité comme une proposition, il nous faut un type "égalité". On veut donc définir une famille de types $a =_A b$ indexée par a et b. Pour ce faire, on va utiliser non pas un type inductif, mais une famille de types *inductive*. Cela veut dire que le principe d'induction ne va pas concerner les différents types $a =_A b$ individuellement, mais toute la famille $b \mapsto a =_A b$ (qu'on s'autorisera à noter $a = b$ quand il n'y a pas d'ambiguïté sur le type A).
Fixons un type A, et un objet $a : A$. Alors on définit la famille de types $b \mapsto a =_A b$ par le constructeur

$$\mathrm{refl}_a : a =_A a$$

Le principe d'induction nous dit que pour toute famille de types

$$C : \prod_{b:B} a =_A b \to \mathcal{U}$$

la donnée d'un élément $c_{\mathrm{refl}_a} : C(a, refl)$ définit une fonction

$$f : \prod_{b:A} \prod_{p:a=_A b} C(b,p)$$

telle que $f(a, refl) \equiv c_{\text{refl}_a}$.

Par exemple, pour prouver $\prod_{b:A} a =_A b \to b =_A a$, il suffit de prouver $a = a \to a = a$, ce qui est évident (il suffit de prendre $x \mapsto x$). Ce principe très simple permet aisément de prouver de nombreux résultats qui seraient des axiomes en théorie des ensembles, comme par exemple

$$\text{trans} : \prod_{a:A} \prod_{b:A} \prod_{c:A} a = b \to b = c \to a = c$$
$$\text{ap} : \prod_{A:\mathcal{U}} \prod_{B:\mathcal{U}} \prod_{f:A\to B} \prod_{a:A} \prod_{b:A} a = b \to f(a) = f(b)$$

Maintenant, soit $B : A \to \mathcal{U}$ une famille de types, $a : A$ et $b : A$ et $p : a = b$. Soit $f : \prod_{x:A} B(x)$. On aimerait pouvoir écrire $f(a) = f(b)$. Cependant, $f(a)$ est un élément du type $B(a)$, et $f(b)$ vit dans $B(b)$. On a bien sûr $\text{ap}_f(p) : B(a) = B(b)$, mais cela ne veut pas dire a priori que $B(a)$ et $B(b)$ sont égaux par définition. On a envie de définir une notion d'égalité *au-dessus de p*. Pour cela, on introduit la fonction de transport :

$$\text{transport}^B(-,-) : a = b \to B(a) \to B(b)$$

définie par induction par $\text{transport}^B(refl, -) :\equiv x \mapsto x$. Une *égalité* entre $x : B(a)$ et $y : B(b)$ *au-dessus* de p est alors définie comme un élément du type

$$\text{transport}^B(p, x) =_{B(b)} y$$

Le *type égalité*, souvent aussi appelé *type identité*, bien qu'il puisse paraître trivial, est en réalité si riche qu'il est à la base de toute la théorie homotopique des types.

2.2 Théorie homotopique des types

L'intuition fondamentale derrière la théorie homotopique des types est de voir les types non pas simplement comme des ensembles, mais comme des espaces. En topologie, on étudie souvent certaines propriétés des espaces en s'intéressant à leurs chemins : un chemin dans un espace X est simplement une application continue de l'intervalle $[0; 1]$ dans X. L'ensemble de ces fonctions $[0; 1] \to X$ peut lui-même être muni de la topologie point à point, et on peut donc s'intéresser à des "chemins d'ordre 2" reliant deux chemins : ce sont les homotopies. Ces homotopies peuvent être vues comme des déformations continues entre des chemins, ou elle-même comme des chemins d'ordre supérieurs. On peut alors définir des chemins entres homotopies, des chemins entre chemins entre homotopies, et ainsi de suite. L'étude des espaces topologiques à déformation près est appelée théorie de l'homotopie, et dès qu'on limite un petit peu la classe des espaces qu'on s'autorise à étudier (en pratique on étudie des CW-complexes), on peut caractériser les espaces à homotopie près juste à partir des informations sur leurs chemins et homotopies supérieures.

La théorie homotopique des types cherche donc à voir les types comme des espaces. Pour cela, plutôt que de voir l'égalité comme une simple proposition, on décide de voir le type $a =_A b$ comme le type des *chemins* entre a et b dans le type A. On a montré par induction sur l'égalité que si $a = b$, on a $b = a$. En interprétant les égalités comme des chemins, cela veut dire qu'on a construit une fonction $-^{-1} : a = b \to b = a$. En topologie, cela correspondrait à l'application $\gamma \mapsto (t \mapsto \gamma(1-t))$. On a également montré la transitivité de l'égalité, qui s'interprète comme une fonction de concaténation $- \cdot - : a = b \to b = c \to a = c$. Le principe d'induction de l'égalité nous permet de montrer tous les résultats élémentaires qu'on s'attend à pouvoir prouver sur des chemins, comme par exemple :

$$\begin{aligned} p \cdot \mathrm{refl} &= p \\ p \cdot (q \cdot r) &= (p \cdot q) \cdot r \\ p \cdot p^{-1} &= \mathrm{refl} \\ (p^{-1})^{-1} &= p \end{aligned}$$

ou encore des résultats moins évidents, comme le théorème d'Eckmann-Hilton :

Théorème 2.2 (Eckmann-Hilton)**.** *Soit A un type, et $a : A$. Alors pour tous chemins p et q dans $\mathrm{refl}_a = \mathrm{refl}_a$, on a $p \cdot q = q \cdot p$.*

Démonstration. Voir [6, Th 2.1.6]. □

Pour rappel, les égalités entre des chemins s'interprètent comme des homotopies. Par exemple, écrire $p \cdot p^{-1} = \mathrm{refl}$ veut simplement dire qu'on peut déformer continûment le chemin $p \cdot p^{-1}$ pour obtenir le chemin refl.

Cette implication topologique de la théorie des types implique qu'il pourrait y avoir plusieurs manières différentes de considérer deux objets comme égaux, plusieurs manières de les *identifier*. Cependant, on ne peut pas exhiber des égalités non triviales (différentes de refl) sans rajouter de construction à la théorie. Nous allons maintenant voir un axiome essentiel qui nous permettra de construire de nombreuses égalités.

2.2.1 L'axiome d'univalence

Intéressons-nous à l'univers $\mathcal{U}$. Si A et B sont deux types dans $\mathcal{U}$, le transport nous donne une fonction

$$\mathrm{transport}^{X \mapsto X}(-,-) : (A = B) \to (A \to B)$$

On peut vérifier que pour tout chemin $p : A = B$, les fonctions

$$\begin{aligned} &\mathrm{transport}^{X \mapsto X}(p,-) && : A \to B \\ &\mathrm{transport}^{X \mapsto X}(p^{-1},-) && : B \to A \end{aligned}$$

sont réciproques l'une de l'autre.

Définition 2.3. On dit qu'une fonction $f : A \to B$ est une équivalence entre A et B si on a

$$\text{isEquiv}(f) :\equiv \left(\sum_{g:B\to A} \prod_{x:A} g(f(x)) = x \right) \times \left(\sum_{h:B\to A} \prod_{y:B} f(h(y)) = y \right)$$

Autrement dit, f admet une réciproque à gauche et à droite. On verra plus tard pourquoi on définit cette notion ainsi et pas en demandant simplement une réciproque.
Le type des équivalences entre A et B est donc définit comme

$$A \simeq B :\equiv \sum_{f:A\to B} \text{isEquiv}(f)$$

Si $f : A \simeq B$ et $a : A$, on notera $f(a)$ l'image de a par la fonction sous-jacente à l'équivalence f.

On vient de voir que le transport permet de définir une fonction

$$\begin{aligned} \text{pathToEquiv} : (A = B) &\to (A \simeq B) \\ p &\mapsto (\text{transport}^{X\mapsto X}(p, -), ...) \end{aligned}$$

L'axiome d'univalence énonce simplement ceci :

Axiome 2.1 (Univalence)**.** *Pour tous types A et B dans $\mathcal{U}$, la fonction* pathToEquiv : $(A = B) \to (A \simeq B)$ *est une équivalence. Autrement dit, on a une fonction*

$$\text{ua} : (A \simeq B) \to (A = B)$$

réciproque de pathToEquiv*. (ua pour univalence axiom)*

Un exemple simple consiste à considérer le type Bool à engendré par deux éléments : 0 et 1. Le type Bool $=_{\mathcal{U}}$ Bool est, par univalence, équivalent au type Bool $\simeq$ Bool, qui contient deux éléments : la fonction identité, et la fonction envoyant 0 sur 1 et 1 sur 0. Cette dernière correspond donc à une identité $swap$: Bool = Bool telle que $\text{transport}^{X\mapsto X}(swap, 0) = 1$ et $\text{transport}^{X\mapsto X}(swap, 1) = 0$.
On peut prouver par le principe d'induction de l'égalité que pour tous types A, B et C, et toutes égalités $p : A = B$, $q : B = C$, on a $ua(p \cdot q) = ua(q) \circ ua(p)$; la concaténation de chemins correspond à la composition de fonctions. De même, l'inversion de chemins correspond à la réciprocation de fonctions.

Si A et B sont deux types, et f et g deux fonctions de A à B, on peut définir par induction sur l'égalité une fonction $happly : f = g \to \prod_{x:A} f(x) = g(x)$. L'axiome d'univalence permet alors de montrer le théorème suivant :

Théorème 2.4 (Extensionnalité)**.** *happly est une équivalence. Entre autres, si f et g vérifient $\prod_{x:A} f(x) = g(x)$, alors on a $f = g$.*

Remarque 2.5. L'extensionnalité a beau être un fait évident en théorie des ensembles, elle est indécidable en théorie des types, et souvent considérée comme un axiome supplémentaire que l'on s'autorise à rajouter. Ici, on voit que l'axiome d'univalence est suffisamment puissant pour impliquer l'extensionnalité. La preuve est cependant un petit peu laborieuse, je ne la détaille donc pas ici (voir [6]).

2.2.2 n-Types

Depuis le début de cette section, on a vu qu'on peut voir les types comme des propositions. En réalité, il y a des situations où il est plus ou moins juste de faire ce parallèle. Par exemple, dire qu'un entier naturel n est une preuve de $\mathbb{N}$ aurait peu de sens. On peut donner du sens à cette intuition : une proposition est un type dont on s'intéresse seulement de savoir s'il vrai ou faux, pas des différentes manières dont il peut être vrai. On dit donc qu'un type X est une proposition s'il vérifie

$$\text{isProp}(X) :\equiv \prod_{x:X} \prod_{y:X} x = y$$

On peut vérifier que les types $\bot$ et $\top$ sont des propositions. Si B est une proposition, on peut aussi vérifier que $A \to B$ est une proposition. Si A est également une proposition, alors $A \times B$ en est également une.
Ce n'est pas le cas de $A + B$, en effet pour prouver $A + B$, on peut prouver A *ou* prouver B, ce qui donne potentiellement deux objets distincts dans $A + B$. De la même manière, si $P : A \to \mathcal{U}$ une famille de types telle que pour tout $x : A$, $P(x)$ est une proposition, on a toujours $\text{isProp}(\prod_{x:A} P(x))$, mais pas nécessairement $\text{isProp}(\sum_{x:A} P(x))$ (il pourrait y avoir plusieurs $x : A$ vérifiant P). Cela suggère que $+$ et $\sum$ ne correspondent pas réellement à l'intuition logique qui les voit comme un "ou" et un "il existe".
Pour palier à ce problème, on introduit la troncation propositionnelle : $\|-\| : \mathcal{U} \to \mathcal{U}$. Pour un type A, le type $\|A\|$ est "engendré" par les constructeurs

$$|-| : A \to \|A\|$$
$$\text{squash} : \prod_{x:\|A\|} \prod_{y:\|A\|} x = y$$

Ce type n'est pas un type inductif à proprement parler, car le constructeur squash est à valeurs non pas $\|A\|$, mais dans un type de chemins dans $\|A\|$. On parle de *type inductif supérieur* (HIT en anglais). Tout ce qu'il faut retenir est que lorsque B est une proposition et A un type quelconque, on a une équivalence entre $A \to B$ et $\|A\| \to B$. Pour définir une fonction partant de $\|A\|$, il suffit donc que le type d'arrivée B soit une proposition, et de définir une fonction de A vers B.
Si A et B sont des propositions, on peut alors définir $A \vee B :\equiv \|A + B\|$. Si A est un type quelconque et $P : A \to \mathcal{U}$ une famille de propositions, on peut définir $\exists x : A, P(x) :\equiv \|\sum_{x:A} P(x)\|$.

De manière analogue à la définition des propositions, on dit qu'un type est un ensemble si ses égalités sont propositionnelles, autrement dit :

$$\text{isSet}(X) :\equiv \prod_{x:X} \prod_{y:X} \text{isProp}(x = y)$$

Intuitivement, un ensemble est caractérisé par ses éléments, pas par les "relations supérieures" que sont les égalités et homotopies. On peut prouver que si un type

est un ensemble, alors non seulement ses égalités sont des propositions, mais aussi tous ses types d'égalités supérieures. En fait, un ensemble est ce qu'on appelle un type "0-tronqué" : il n'a pas de "contenu homotopique" au-dessus du niveau 0 : les éléments.
On peut en fait définir toute une hiérarchie de niveaux de troncations, par récurrence en commençant à -1 :

$$\begin{aligned} \text{is-}(-1)\text{-type}(X) &:\equiv \text{isProp}(X) \\ \text{is-}(n+1)\text{-type}(X) &:\equiv \prod_{x:X} \prod_{y:X} \text{is-}n\text{-type}(x = y) \end{aligned}$$

Remarque 2.6. Les -1-types sont les propositions, les 0-types sont les ensembles, 1-types sont appelés *groupoïdes*, les 2-types 2-*groupoïdes*, etc. Cette dénomination provient de la théorie des catégories supérieures.

En réalité, on peut même commencer à -2, en avec la notion de contractilité : un type X est contractile s'il vérifie

$$\text{isContr}(X) :\equiv \sum_{x:X} \prod_{y:X} x = y$$

Si $(c, p) : \text{isContr}(X)$, c est appelé le *centre de contraction* de X. Le mot "contractile" vient encore de l'intuition topologique : le type X peut être "contracté" en un point, son centre de contraction. Une proposition est alors un type dont les espaces de chemins sont contractiles.

Remarque 2.7. Ces notions, en particulier celles de contractilité et propositionnalité sont essentielles dans le reste de ce rapport. En fait, pour tout type X, "être un n-type" est une proposition. Dans mon stage, on voit souvent apparaître des hypothèses sous la forme de troncation propositionnelles. On rencontre très souvent des situations de ce style :

> On a $\|A\|$, et on aimerait construire $x : X$. Pour ça on aimerait avoir un élément $a : A$. On cherche alors un prédicat $P : X \to \mathcal{U}$ qui caractérise de manière unique l'élément $x : X$ que l'on souhaite construire. On se ramène donc à montrer $\alpha : isContr(\sum_{x:X} P(x))$, qui est une proposition, pour pouvoir utiliser le principe d'induction de la troncation propositionnelle. Il suffit enfin de prendre le centre de contraction de α.

Remarque 2.8. La notion de proposition justifie la définition précédente de isEquiv. Définit ainsi, pour toute fonction $f : A \to B$, isEquiv(f) est une proposition. Si on avait définit

$$\text{isEquiv}^{bis}(f) :\equiv \sum_{g:B\to A} (\prod_{x:A} (g(f(x)) = x)) \times (\prod_{y:B} f(g(y)) = y)$$

on n'aurait pas nécessairement cette propositionnalité.

Remarque 2.9. Là où les n-types sont des types n'ayant aucun contenu homotopique au-dessus de l'ordre n, on peut définir de manière duale les types n-connexes comme étant ceux n'ayant aucun contenu homotopique en-dessous de l'ordre n. Autrement dit, un type X est n-connexe s'il vérifie

$$\mathrm{isContr}(\|X\|_n)$$

où $\| - \|_n$ est l'analogue de la troncation propositionnelle pour les n-types.
Les types 0-connexes sont précisément les types X qui vérifient

$$\|X\| \qquad \text{(X est "localement habité")}$$

$$\prod_{x:X}\prod_{y:X}\|x = y\| \qquad \text{(X est "connexe")}$$

2.2.3 Structures

Avant d'entrer le cœur de mon sujet de stage, j'aimerais aborder la notion de *structure*. Les sommes dépendantes nous donnent une manière assez naturelle d'aborder cette notion. Commençons sur un exemple. En théorie des ensembles, un magma est un ensemble E muni d'une loi interne $- \star - : E \to E \to E$. En théorie des types, on peut donc définir

$$\mathrm{Magma} :\equiv \sum_{X:\mathcal{U}} (X \to X \to X) \times (\mathrm{isSet}(X))$$

L'axiome d'univalence nous permet de caractériser l'égalité dans les types sommes. Si A est un type et $B : A \to \mathcal{U}$ une famille de types, et si (a, b) et (a', b') sont des éléments de $\sum_{x:A} B(x)$, on peut prouver l'équivalence suivante :

$$((a, b) = (a', b')) \simeq \sum_{p:a=a'} \mathrm{transport}^B(p, b) = b'$$

Où le sens de gauche à droite de cette équivalence est défini par induction sur l'égalité. Dans le cas des magmas, cela veut dire qu'une égalité entre deux magmas (E, m, s) et (E', m', s') correspond à la donnée d'une égalité $p : E = E'$ et d'une égalité

$$q : \mathrm{transport}^{X \mapsto ((X \to X \to X) \times \mathrm{isSet}(X))}(p, (m, s)) = (m', s')$$

Or, par un raisonnement analogue pour le produit cartésien, la donnée d'une telle égalité q correspond à la donnée de deux égalités

$$r : \mathrm{transport}^{X \mapsto (X \to X \to X)}(p, m) = m'$$
$$t : \mathrm{transport}^{X \mapsto (\mathrm{isSet}(X))}(p, s) = s'$$

Puisque $\mathrm{isSet}(E')$ est une proposition, t est une preuve d'égalité dans un type propositionnel, et existe donc automatiquement. Le type $(E, m, s) = (E', m', s')$ est donc équivalent au type

$$\sum_{p:E=E'} \mathrm{transport}^{X \mapsto (X \to X \to X)}(p, m) = m'$$

qui, par extensionnalité, est équivalent au type

$$\sum_{p:E=E'} \prod_{x:E'} \prod_{y:E'} \text{transport}^{X\mapsto(X\to X\to X)}(p,m)(x,y) = m'(x,y)$$

En raisonnant par induction sur l'égalité, on peut prouver

$$\begin{aligned}&\text{transport}^{X\mapsto(X\to X\to X)}(p,m)(x,y) = \\ &\text{transport}^{X\mapsto X}(p, m(\text{transport}^{X\mapsto X}(p^{-1},x), \text{transport}^{X\mapsto X}(p^{-1},y)))\end{aligned}$$

Lorsque $p \equiv \text{ua}(f^{-1})$ avec $f : E' \simeq E$, on a donc

$$\text{transport}^{X\mapsto(X\to X\to X)}(p,m)(x,y) = f^{-1}(m(f(x),f(y)))$$

Le type $\text{transport}^{X\mapsto(X\to X\to X)}(p,m)(x,y) = m'(x,y)$ est donc équivalent au type $f^{-1}(m(f(x),f(y))) = m'(x,y)$, lui-même équivalent au type $m(f(x),f(y)) = f(m'(x,y))$. L'axiome d'univalence nous a finalement permis de montrer

$$\begin{aligned}&((E,m,s) = (E',m',s')) \simeq \\ &\sum_{f:E'\simeq E} \prod_{x:E'} \prod_{y:E'} m(f(x),f(y)) = f(m'(x,y))\end{aligned}$$

Ce dernier type est exactement ce qu'on appelle en algèbre le type des *isomorphismes* de magma. Pour des structures plus générales dont le type sous-jacent n'est pas nécessairement un ensemble, on préfère le terme d'*équivalence*. On parlera par exemple d'équivalences de torseurs et de gerbes.
L'axiome d'univalence et le principe d'induction de l'égalité ont donc permis à eux seuls de caractériser ce qu'était une égalité entre magmas, en retrouvant la bonne notion d'isomorphisme issue de l'algèbre. On aurait aussi pu définir le type des groupes, en rajoutant d'autres axiomes (tous propositionnels), et conclure de la même manière qu'une égalité entre groupes est un isomorphisme de groupes.

2.2.4 Espaces de lacets, Ω

Une dernière notation que j'utiliserai dans ce rapport concerne les *espaces de lacets*.

Définition 2.10 (Type pointé)**.** Une type pointé est une paire (X,x) où X est un type et x un élément de X. Autrement dit, le type des types pointés est $\sum_{X:\mathcal{U}} X$.

Définition 2.11 (Espace des lacets)**.** Si (X,x) est un type pointé, son espace des lacets, noté $\Omega((X,x))$ est le type pointé $(x = x, \text{refl}_x)$. On définit Ω^n par récurrence sur n comme la composée n-ième de Ω.

L'intérêt des espaces $x = x$ plutôt que $x = y$ avec y quelconque est que l'on peut concaténer autant de chemins que l'on veut sans changer le type d'arrivée. Si le type X est un ensemble, $x = x$ est donc naturellement muni d'une loi de

groupe (on l'appelle le *groupe fondamental* de X). De manière plus générale, on peut prendre la troncation ensembliste (analogue de la troncation propositionnelle, mais pour les ensembles) du n-ième espace de lacets pour obtenir ce que l'on appelle le n-ième groupe d'homotopie de X en x. L'étude des groupes d'homotopies des espaces topologiques est l'un des grands domaines de la topologie algébrique. Le théorème d'Eckmann-Hilton cité plus haut énonce précisément que tous les $\Omega^n((X, x))$ pour $n \geq 2$ sont abéliens.
En un certain sens, le but de mon stage a été de construire une "réciproque" à l'opérateur Ω, autrement dit construire un opérateur de *délacement*. Cet opérateur permet de donner une définition aux groupes de *cohomologie*.

3 Groupes de cohomologie

Les groupes de cohomologie sont des invariants algébriques que l'on peut associer aux espaces topologiques. Avec l'interprétation homotopique de la théorie des types, qui voit les types comme des espaces, on peut donc associer des groupes de cohomologie à des types. Il y a plusieurs manières de définir ces groupes. La définition que l'on va utiliser repose sur la construction de types appelés *espaces d'Eilenberg-MacLane*.

Définition 3.1 (Espaces d'Eilenberg-MacLane)**.** Soit A un groupe abélien. On note pour $n \in N$, $\mathcal{B}^n A$ un n-type $(n-1)$-connexe pointé, tel que $\Omega^n(\mathcal{B}^n A)$ est isomorphe à A en tant que groupe abélien (la loi de groupe sur $\Omega^n(\mathcal{B}^n A)$ provenant de la concaténation de chemins).
$\mathcal{B}^n A$ est appelé le n-ième espace d'Eilenberg-MacLane associé au groupe A. On le trouve souvent noté $K(A, n)$ dans la littérature.

Remarque 3.2. L'utilisation du mot "le" suppose qu'un tel espace est unique. J'ignore si cette unicité a déjà été prouvée en THoT. En topologie algébrique, leur unicité est garantie dès lors qu'on suppose que les espaces que l'on considère sont des CW-complexes [3, Prop 4.30]. Je donne une preuve de l'unicité du type pointé $\mathcal{B}^1 A$ dans le corollaire 6.3.1.

Remarque 3.3. La condition "A abélien" n'est pas nécessaire pour définir $\mathcal{B}^1 A$. Pour $n \geq 2$, l'argument d'Eckmann-Hilton (2.2) garantit que la condition "A abélien" devient nécessaire pour l'existence de $\mathcal{B}^n A$.

Définition 3.4 (Groupes de cohomologie)**.** Soit X un type, A un groupe abélien, et $n : \mathbb{N}$. Le n-ième groupe de cohomologie de X à valeurs dans A, noté $H^n(X, A)$, est défini comme $\|X \to \mathcal{B}^n A\|_0$. La loi de groupe provient de l'identification $\mathcal{B}^n A \simeq \Omega(\mathcal{B}^{n+1} A)$, qui permet de voir les éléments de $\mathcal{B}^n A$ comme des chemins dans $\mathcal{B}^{n+1} A$. Cela définit une opération

$$- \cdot - : \mathcal{B}^n A \to \mathcal{B}^n A \to \mathcal{B}^n A$$

On en déduit donc donc une loi sur $X \to \mathcal{B}^n A$ en appliquant l'opération point par point. Enfin, cette opération passe à la troncation $\| - \|_0$.

L'existence des types $\mathcal{B}^n A$ est déjà connue en THoT [5]. Le but de mon stage n'était donc pas de construire ces types, mais de les re-construire, cette fois-ci par l'approche exposée par Deligne. L'approche des auteurs de [5] repose sur une intuition que je qualifierai de plus "topologique", reposant sur des types inductifs supérieurs ; tandis que l'approche de Deligne est plus "algébrique" : on va définir des structures (torseurs, gerbes), et vérifier que les espaces sous-jacents à ces structures (type des A-torseurs, type des gerbes liées par A) sont des types $\mathcal{B}^n A$ ($n = 1$ pour les torseurs, $n = 2$ pour les gerbes).
La construction de $\mathcal{B}^1 A$ comme groupoïde des A-torseurs a déjà été étudiée lors du stage de Victor. Dans un premier temps, je ré-explore certains résultats autour de cette construction. Ensuite, je m'intéresse à la construction de $\mathcal{B}^2$.
Cette exposition n'a pas pour but prouver un résultat véritablement nouveau, mais plus de donner un panorama de la manière dont le langage des torseurs et gerbes se traduit naturellement en THoT.

4 Torseurs, $\mathcal{B}^1$

Dans cette section, G dénote un groupe, non nécessairement abélien. Pour rappel, G est donc un tuple $(X, - \times -, 1_G, -^{-1}, ...)$ où $- \times - : X \to X \to X$ est une loi interne, $1_G : X$, $-^{-1} : X \to X$, et où "..." désigne une liste d'axiomes vérifiant par exemple que 1_G est neutre à gauche et à droite pour $- \times -$, que X est un ensemble, etc. On peut prouver que tous ces axiomes sont propositionnels, on ne prendra donc pas la peine de les nommer.
On écrira par abus de notation $g : G$ pour $g : X$.
Commençons par définir les G-torseurs :

Définition 4.1. Un G-torseur est la donnée de :
- Un type X
- Qui soit un ensemble : isSet(X)
- Qui soit "habité" : $\|X\|$. Prendre la troncation propositionnelle ici permet de demander que X soit non-vide, sans spécifier pour autant un point de X. Entre autres, X n'est pas un type pointé, et on va voir par la suite que se donner un point dans X permet d'identifier X à G.
- Une action de groupe $- \star - : X \to G \to X$
- Qui vérifie donc les axiomes d'action de groupe : $(x \star g) \star g' = x \star (g \times g')$ et $x \star 1_G = x$ pour tout $x : X$, $g, g' : G$.
- Qui soit transitive : $\prod_{x,y:X} \sum_{g:G} x \star g = y$
- Et qui soit libre : si $x \star g = x$, alors $g = 1_G$.

Les deux derniers points reviennent à dire que pour tout $x, y : X$, le type $\sum_{g:G} x \star g = y$ est contractile, ce qui prouve que ces derniers axiomes sont bien propositionnels.
Le type des G-torseurs est alors défini comme :

$$G-\text{torsors} :\equiv \sum_{X:Type} \sum_{-\star-:X\to G\to X} \text{isTorsor}(G, X, - \star -)$$

Le groupe G peut lui-même être muni d'une structure de G-torseur, prenant pour X l'ensemble des éléments de G, et pour $- \star -$ la loi $- \times -$ sur G. On appelle ce torseur le "G-torseur principal", noté T_G. On a le théorème suivant :

Théorème 4.2. *Le type des G-torseurs, pointé en T_G, est un $\mathcal{B}^1 G$.*

Démonstration. La preuve repose sur deux arguments :
Tout d'abord, des raisonnements à base de transport et d'univalence permettent de caractériser les types identité dans le type des G-torseurs : si T_1 et T_2 sont deux G-torseurs, le type $T_1 = T_2$ est équivalent au type

$$T_1 \simeq T_2 :\equiv \sum_{f:T_1 \to T_2} \prod_{x:T_1} \prod_{g:G} f(x \star^1 g) = f(x) \star^2 g$$

et cette équivalence (notée F_1) envoie la concaténation de chemins sur la composition de fonctions. Ensuite, en prenant $T_1 \equiv T_2 \equiv T_G$, on peut alors définir $F_2 : (T_G \simeq T_G) \to G$ par $F_2(f, !) :\equiv f(1_G)$. On peut prouver que F_2 est une équivalence, et que pour tout $p, q : T_G = T_G$, $F_2(F_1(p \cdot q)) = F_2(F_1(q)) \times F_2(F_1(p))$.
Lorsque G est abélien, $F_2 \circ F_1$ est donc un isomorphisme de groupes entre $\Omega(G-\text{torsors}, T_G)$ et G. Si G n'est pas abélien, alors $F_2 \circ F_1$ donne un isomorphisme avec le groupe dual de G, lui-même isomorphe à G par $g \mapsto g^{-1}$. □

Ici, j'aimerais partager une intuition sur ce que sont les torseurs, et notamment sur l'hypothèse $\|X\|$, que l'on énoncera "X est localement habité", par analogie avec le langage utilisé par Deligne. Cette hypothèse est probablement la moins intuitive quand on découvre la THoT. On peut se demander "pourquoi ne pas simplement demander la donnée d'un objet $x : X$?". Le résultat suivant nous donne un début de réponse.

Théorème 4.3 (Trivialisation des torseurs)**.** *Soit T un G-torseur. Si $x : T$, alors on peut construire une équivalence $T \simeq T_G$. Autrement dit, on a une fonction* $\text{triv}_T : T \to (T \simeq T_G)$.

Idée de preuve. Étant donné $x_0 : T$, $\text{triv}_T(x_0)$ est une équivalence de torseurs dont la fonction sous-jacente associe à un élément $x : T$ l'élement $g : G$ tel que $x_0 \star g = x$. Cette équivalence "identifie" x_0 à l'élément neutre de G. □

De plus, étant donné une équivalence $f : T \simeq T_G$, on peut constuire un élément de T en prenant $f^{-1}(1_G)$. On peut prouver que la fonction ainsi définie est réciproque de triv_T, on a donc en fait

$$\text{triv}_T : T \simeq (T \simeq T_G)$$

(attention à la manière de lire cette ligne, le premier symbole $\simeq$ est une équivalence de types, tandis que le second est une équivalence de torseurs). Puis, en composant avec F_1, on peut définir

$$\text{Triv}_T : T \simeq (T = T_G)$$

.

Remarque 4.4. Lorsque $T \equiv T_G$, on retrouve le théorème 4.2.

Dans l'idée, on peut donc voir un G-torseur "non-trivial" comme une version "tordue" de G où l'élément neutre resterait à préciser. Pour l'exemple, prenons $G :\equiv \mathbb{Z}$ le groupe des entiers relatifs. Le type $\mathcal{B}^1\mathbb{Z}$ est le cercle S^1, dont une définition possible en THoT est d'utiliser un type inductif supérieur, ayant deux constructeurs :

$$\begin{aligned} &\text{base} : S^1 \\ &\text{loop} : base = base \end{aligned}$$

En acceptant l'unicité de $\mathcal{B}^1\mathbb{Z}$, étant donné un point $x : S^1$, le type $x = base$ peut être muni d'une structure de $\mathbb{Z}$-torseur, mais on ne peut pas simplement donner un élément de $x = base$ sans plus d'hypothèses sur x. Dans le cas contraire, on aurait $\prod_{x:S^1} x = base$, ce qui impliquerait que le cercle est contractile.

En généralisant la définition de $\mathcal{B}^n$, on peut "définir" $\mathcal{B}^0 G$ comme étant simplement le groupe G, pointé en son élément neutre. Intuitivement, un torseur est donc un espace $\mathcal{B}^0$, modulo le choix du neutre. Cette intuition va naturellement amener la notion de gerbes, qui seront définies comme des espaces $\mathcal{B}^1$, modulo le choix de leur point.

5 Gerbes, liens

Pour préciser l'intuition précédente, on aurait pu définir la notion de "torseur" comme étant un type qui est un ensemble et qui est localement habité. Un G-torseur est alors un "torseur" muni d'une "action" qui le "lie" au groupe G. C'est par cette approche en deux étapes que nous allons définir les gerbes.

Définition 5.1. Une gerbe $\mathcal{G}$ est la donnée d'un type X vérifiant :
- X est localement habité ($\|X\|$)
- X est un groupoïde
- X est connexe, c'est-à-dire $\prod_{x,y:X} \|x = y\|$. On sait que tous les éléments peuvent être reliés par des chemins, mais un tel chemin n'est pas spécifié en toute généralité (cela impliquerait que X est contractile).

On dit de plus qu'une gerbe $\mathcal{G}$ est abélienne si on a

$$\prod_{x:X} \prod_{p,q:x=x} p \cdot q = q \cdot p$$

.

Remarque 5.2. Les 4 axiomes ci-dessus sont tous propositionnels. Une égalité entre gerbes est donc simplement une égalité entre les types sous-jacents.

Comment alors définir ce qui lie une gerbe à un groupe ? Dans [1], Deligne écrit :

> Si $\mathcal{G}$ est une gerbe sur X et que, pour tout objet P de $\mathcal{G}$ sur un ouvert U, le faisceau $\mathbf{Aut}(P)$ sur U des automorphismes de P [...] est

commutatif, ce faisceau $\mathbf{Aut}(P)$ ne dépend pas du choix de P et les $\mathbf{Aut}(P)$ se recollent en un faisceau de groupes abéliens sur X, appelé le *lien* de $\mathcal{G}$.

Dans le langage de la THoT, cela se traduit par : si $\mathcal{G}$ est une gerbe, et que pour tout $x : \mathcal{G}$, $x = x$ est commutatif, alors les $x = x$ "ne dépendent pas du choix de $x : \mathcal{G}$" et se "recollent" en un groupe abélien $\pi(\mathcal{G})$.

Commençons par essayer de comprendre la phrase "$x = x$ ne dépend pas du choix de x". Soit $\mathcal{G}$ une gerbe et $x, y : \mathcal{G}$, on aimerait prouver $(x = x) \simeq (y = y)$. L'idée est la suivante : plutôt que de construire directement un élément de $(x = x) \simeq (y = y)$, on va essayer de caractériser de manière unique le morphisme que l'on veut construire. En s'y prenant correctement, on essaiera donc de construire un élément d'un type de la forme

$$T(x, y) :\equiv \sum_{s:(x=x)\simeq(y=y)} \mathrm{carac}(s)$$

avec pour espoir que ce type soit contractile. On pourra donc utiliser le principe d'induction de la troncation propositionnelle appliqué à l'hypothèse de connexité de $\mathcal{G}$ afin de déduire un élément $p : x = y$.

Remarque 5.3. Comme expliqué dans la remarque 2.7, les raisonnements permettant de construire des objets en se basant sur des propriétés qui les caractérisent de manière unique ont été omniprésents pendant mon stage. Ils m'ont paru si fondamentaux et incontournables que j'ai décidé de détailler au moins celui-ci dans le corps du rapport.

Voyons donc comment trouver une caractérisation unique. Étant donné un chemin $p : x = y$, on peut définir un isomorphisme de groupe

$$\begin{aligned} f_p : (x = x) &\simeq (y = y) \\ q &\mapsto p^{-1} \cdot q \cdot p \end{aligned}$$

On peut espérer que f_p ne dépende pas du choix de p, et on pose donc

$$\mathrm{carac}(s) :\equiv \prod_{p:x=y} s = f_p$$

.

On cherche à montrer $\mathrm{isContr}(T(x, y))$. Or pour tout type X, $\mathrm{isContr}(X)$ est une proposition. Donc, par le principe d'induction de la troncation propositionnelle, prouver $\|x = y\| \to \mathrm{isContr}(T(x, y))$ est à équivalent à prouver $(x = y) \to \mathrm{isContr}(T(x, y))$. On se donne donc $p_0 : x = y$. Par transport au-dessus de p_0 dans la famille $y \mapsto \mathrm{isContr}(T(x, y))$, il suffit de prouver $\mathrm{isContr}(T(x, x))$. Pour rappel, $\mathrm{isContr}(X) \equiv \sum_{a:X} \prod_{b:X} a = b$.

Commençons par construire $(s_0, c) : T(x, x)$. On prend pour s_0 l'isomorphisme identité : $(x = x) \simeq (x = x)$. On cherche à construire $c : \prod_{p:x=x} s_0 = f_p$. Soit $p : x = x$. Une égalité entre isomorphismes de groupes est simplement une égalité

entre les fonctions sous-jacentes, qui par extensionnalité est juste l'égalité point à point. Soit donc $q : x = x$. On veut montrer $s_0(q) = f_p(q)$. Autrement dit, on veut prouver que pour tout $p, q : x = x$,

$$q = p^{-1} \cdot q \cdot p$$

Ici, on voit donc la nécessité de supposer la gerbe $\mathcal{G}$ abélienne.

Dans la suite, on suppose donc $\mathcal{G}$ abélienne. Cela nous donne $(s_0, c) : T(x, x)$. Maintenant, on veut montrer que

$$\prod_{(s',c'):T(x,x)} (s_0, c) = (s', c')$$

Soit $(s', c') : T(x, x)$. Par c' appliqué à refl, on a pour tout $q : x = x$

$$\begin{aligned} s'(q) &= \mathrm{refl}^{-1} \cdot q \cdot \mathrm{refl} \\ &= q \\ &\equiv s_0(q) \end{aligned}$$

On a donc une preuve $\rho : s = s'$. Pour finir, il suffit de prouver que le transport de c au-dessus de ρ est égal c'. Or le type de c' est $\prod_{p:x=x} s' = f_p$, qui est équivalent au type $\prod_{p,q:x=x} s'(q) = f_p(q)$. Pour tout $q : x = x$, $s'(q)$ et $f_p(q)$ sont des éléments de $x = x$, et x est un élément de $\mathcal{G}$, qui par hypothèse est un groupoïde. Le type de c' est donc une proposition, et donc le transport de c au-desus de ρ est bien égal à c'. Au final, on a bien prouvé

Théorème 5.4. *Pour toute gerbe abélienne $\mathcal{G}$, pour tous $x, y : \mathcal{G}$, on a un isomorphisme de groupes*

$$s_{x,y} : (x = x) \simeq (y = y)$$

qui vérifie :

$$\prod_{p:x=y} \prod_{q:x=x} s_{x,y}(q) = p^{-1} \cdot q \cdot p \tag{1}$$

Remarque 5.5. Avec des raisonnements analogues, on peut assez aisément prouver que $s_{x,x}$ est l'identité, et que pour tous $x, y, z : \mathcal{G}$, $s_{y,z} \circ s_{x,y} = s_{x,z}$.

On peut finalement définir

Définition 5.6 (Lien d'une gerbe)**.** Soit $\mathcal{G}$ une gerbe abélienne, et A un groupe abélien. Un lien de $\mathcal{G}$ à A est la donnée

— d'une famille d'isomorphismes de groupes $e : \prod_{x:\mathcal{G}} (x = x) \simeq A$
— d'une condition de recollement : $\prod_{x,y:\mathcal{G}} e(x) = e(y) \circ s_{x,y}$

On note $\mathrm{link}(\mathcal{G}, A)$ le type des liens de $\mathcal{G}$ à A.

Remarque 5.7. Après avoir travaillé plusieurs jours avec cette définition, j'ai fini par me rendre compte qu'en THoT, l'hyptohèse de recollement est automatiquement garantie par l'abélianité de $\mathcal{G}$. J'ai néanmoins décidé de garder cette hypothèse dans la définition, pour rester au plus proche de l'intuition que m'en a donné Thierry Coquand.

Remarque 5.8. On peut prouver que le type des "liens d'une gerbe à un groupe" n'est pas propositionnel, mais est un ensemble, ce qui justifie le nom de "lien" et non pas de "être lié par". On dira néanmoins "gerbe liée par A" pour parler de la donnée d'une gerbe et de son lien à A.

Équipé de cette notion de lien, on peut définir le groupe fondamental d'une gerbe abélienne sans avoir à choisir un point de base.

Théorème 5.9. *Soit $\mathcal{G}$ une gerbe abélienne. Alors le type*

$$\sum_{A:\mathrm{Group}} \mathrm{link}(\mathcal{G}, A)$$

est contractile. Son centre de contraction définit un groupe abélien, noté $\pi(\mathcal{G})$, ainsi qu'un lien de $\mathcal{G}$ à $\pi(\mathcal{G})$.

Idée de preuve. Ce théorème repose encore sur un argument de caractérisation unique. Plutôt que de construire directement $\pi(\mathcal{G})$, on justifie que s'il existe, il est nécessairement unique. Dès lors, on se ramène à construire un élément d'une proposition, et on peut donc utiliser le fait que $\mathcal{G}$ est localement habitée pour se donner un point $x : \mathcal{G}$, et définir $\pi(\mathcal{G})$ comme étant $x = x$. □

Dans l'autre direction, étant donné un groupe abélien A, on peut construire une gerbe liée par A. La gerbe en question est simplement le groupoïde des A-torseurs. Par le théorème 4.2, on a un isomorphisme entre $T_A = T_A$ et A. Étant donné un A-torseur T, on peut composer l'isomorphisme précédent avec $s_{T_A,T}$ pour obtenir un isomorphisme entre $T = T$ et A. Cela nous donne bien un lien de la gerbe des A-torseurs au groupe A (que l'on appellera le *lien trivial*). On note $\mathcal{G}_A$ la gerbe des A-torseurs, et $\mathcal{L}_A$ son lien trivial.

6 Trivialisation de gerbes, $\mathcal{B}^2$

Avec les notions de gerbe et de lien, on peut définir le deuxième délacement d'un groupe abélien A.

Théorème 6.1. *Soit A un groupe abélien. Alors le type*

$$\sum_{\mathcal{G}:\mathrm{AbGerbe}} \mathrm{link}(\mathcal{G}, A)$$

pointé en $(\mathcal{G}_A, \mathcal{L}_A)$ est un $\mathcal{B}^2 A$.

Pour prouver ce théorème, on va d'abord montrer l'analogue du théorème 4.3 pour les gerbes. Pour rappel, on a vu qu'il y a une correspondance entre la donnée d'un point dans un torseur et d'une équivalence entre ce torseur et le torseur trivial. Pour les gerbes, on a de même :

Théorème 6.2. *Soit A un groupe abélien, $\mathcal{G}$ une gerbe et $\mathcal{L} : \mathrm{link}(\mathcal{G}, A)$. Alors on a une équivalence*

$$\mathrm{Triv} : \mathcal{G} \simeq ((\mathcal{G}, \mathcal{L}) = (\mathcal{G}_A, \mathcal{L}_A))$$

Recette de preuve. Soit $x : \mathcal{G}$. Étant donné $y : \mathcal{G}$, le type $x = y$ peut être muni d'une structure de A-torseur : le lien $\mathcal{L}$ associe à tout $a : A$ un chemin $p(a) : y = y$. Pour $q : x = y$, on pose donc

$$q \star a = q \cdot p(a)$$

On obtient donc une fonction $f_x : \mathcal{G} \to \mathcal{G}_A$. On peut prouver que cette fonction est une équivalence. L'axiome d'univalence nous donne donc une égalité $\rho_x : \mathcal{G} = \mathcal{G}_A$. On peut de plus prouver que cette égalité induit une égalité

$$\mathrm{transport}^{\mathrm{link}(-,A)}(\rho_x, \mathcal{L}) = \mathcal{L}_A$$

On a donc une première fonction

$$f : \mathcal{G} \to ((\mathcal{G}, \mathcal{L}) = (\mathcal{G}_A, \mathcal{L}_A))$$

Réciproquement, étant donné une égalité $p : ((\mathcal{G}, \mathcal{L}) = (\mathcal{G}_A, \mathcal{L}_A))$, on peut obtenir un élément de $\mathcal{G}$ donné par

$$g(p) :\equiv \mathrm{transport}^{\mathrm{fst}}(p^{-1}, T_A) : \mathcal{G}$$

Enfin, on peut montrer que f et g sont réciproques, d'où le résultat. □

Preuve du théorème. Le théorème 6.1 est alors juste un corollaire du théorème précédent. En effet, on a

$$\begin{aligned} \Omega^1(\mathcal{B}^2 A) &\equiv ((\mathcal{G}_A, \mathcal{L}_A) = (\mathcal{G}_A, \mathcal{L}_A)) \\ &= \mathcal{G}_A \qquad \text{(par univalence et le théorème 6.2)} \end{aligned}$$

Par 4.2, on a donc $\Omega^2(\mathcal{B}^2 A) = A$. Puisque $\Omega(\mathcal{B}^2 A) = \mathcal{G}_A$ et que $\mathcal{G}_A$ est un groupoïde, on peut déduire que $\mathcal{B}^2 A$ est un 2-groupoïde. Je ne détaille pas ici la preuve qu'il est bien 1-connexe. □

Remarque 6.3. Je n'ai pas pu directement implémenter la preuve précédente en Cubical Agda, ayant été confronté à des problèmes de niveaux d'univers. J'ai préféré omettre ces problèmes dans le corps de ce rapport car bien que je les ai souvent rencontrés lors de mon stage, ils m'ont toujours paru contournables, moyennant un certain effort technique.
Dans le cas précis de ce théorème, j'ai plutôt utilisé un argument adapté de [4], en montrant que $\Omega(\mathcal{B}^2 A)$ est égal à ce que les auteurs de [4] appellent le *centre concret* de $\mathcal{B}^1 A$, lui-même équivalent à $\mathcal{B}^1 A$ par abélianité de A.

Corollaire 6.3.1 (Unicité de $\mathcal{B}^1$)**.** *Le type pointé $\mathcal{B}^1 A$ est unique. Autrement dit, pour tout groupoïde connexe pointé (X, x) tel que le groupe $\Omega((X, x))$ est isomorphe à A, on a $\mathcal{B}^1 A = (X, x)$.*

Démonstration. X est un groupoïde connexe. Il est pointé, donc localement habité, c'est donc une gerbe. Puisque le groupe $x = x$ est isomorphe à A, il est abélien. Être abélien est une proposition, on en déduit par connexité que pour tout $y : X$, $y = y$ est un groupe abélien. X est donc une gerbe abélienne, et composer l'isomorphe $(x = x) \simeq A$ avec les fonctions $s_{x,y}$ fournit un lien de X à A.
Par le théorème 6.2, le point $x : X$ fournit une égalité $p : X = \mathcal{G}_A$. Dans la preuve du théorème, on voit que p transporte x vers le A-torseur $x = x$. Ce A-torseur contient le point refl_x, et est donc égal au A-torseur principal par le théorème 4.3.
On a donc bien $(X, x) = (\mathcal{G}_A, A-\text{torsors})$. □

7 Conclusion

J'espère avoir pu donner, dans les différentes preuves exposées, un aperçu de la remarquable efficacité de la théorie homotopique des types pour manipuler différentes constructions issues de la théorie des champs et faisceaux. L'approche étudiée lors de mon stage me semble généralisable pour les groupes de cohomologie suivants, en utilisant des n-gerbes, et en trouvant une bonne notion de n-lien. Bien qu'il y ait déjà une construction connue des espaces d'Eilenberg-MacLane en THoT, on peut espérer que les reconstruire en utilisant ces notions permette une meilleure compréhension de certains phénomènes.
Par exemple, si A est un anneau commutatif et X un espace, alors en notant $H^n(X, A)$ le n-ième groupe de cohomologie de X dans le groupe abélien issu de l'addition dans A, on peut associer à tout couple d'entiers (m, n) une fonction

$$- \smile - : H^m(X, A) \times H^n(X, A) \to H^{m+n}(X, A)$$

appelée cup-produit, qui permet de munir le groupe abélien

$$H^*(X, A) :\equiv \bigoplus_{n \in \mathbb{N}} H^n(X, A)$$

d'une structure d'anneau.
Cela peut se traduire par une fonction $B^m(A) \times B^n(A) \to B^{m+n}(A)$. Dans le cas particulier $m \equiv n \equiv 1$, cela s'interprète en disant qu'à tout couple de A-torseurs, on peut associer une A-gerbe. Implémenter cette construction en Agda était une direction possible de mon stage, qui n'a finalement pas aboutie, mais qui serait très intéressante à poursuivre ultérieurement.

A Détails d'implémentation

J'ai implémenté les différents résultats étudiés dans le langage Cubical Agda. Agda est un langage de programmation surcouche de Haskell, basé sur une théorie des types dépendants. Cubical Agda est une version d'Agda augmentée de certaines primitives permettant d'y faire de la *théorie cubique des types*. Cette théorie est une variante de la théorie homotopique des types où on définit axiomatiquement un type intervalle I avec un élément 0 et un élément 1, et où on interprète les fonctions $p : I \to A$ comme des égalités entre $p(0)$ et $p(1)$. On peut prouver que l'égalité définie ainsi est équivalente à l'égalité définie comme famille de type inductive. L'intérêt est, entre autres, de permettre de définir de manière assez générale des *types inductifs supérieurs* (tandis qu'en Agda-HoTT par exemple, chaque type inductif supérieur doit être déclaré axiomatiquement).

Le code est disponible à l'adresse `https://github.com/Edlyr/ELib`. À l'heure d'écriture, mon installation d'Agda et de Cubical Agda correspond aux commits suivants des bibliothèques respectives :

- Mon code : 7768bb4b41c24afcd8020977978d931659e30b5e.
- Bibliothèque Cubical Agda : `022d21280e313198f75ed5129f380195aa6244fa` du repository `https://github.com/agda/cubical`.
- Librairie standard Agda : `a79fa50aa5bf69f1804d292879a106bc9290bd06` du repository `https://github.com/agda/agda-stdlib`.
- Agda : `8255ee2a9e5eaa5966e7893005d3576c7970fb7f` du repository `https://github.com/agda/agda`.

Les parties exposées dans le corps du rapport se trouvent essentiellement dans les dossiers `Gerbe` et `Torsor`.

Remarque A.1. Pendant mon stage, j'ai régulièrement été confronté à des problèmes de temps de calcul. En effet, la vérification de typage utilisée par agda met parfois trop de temps à s'exécuter pour pouvoir être utilisée. Pour contourner ce problème, j'ai dû utiliser à différents endroits le mot-clef `abstract`, qui demande à agda de ne pas chercher à réduire un terme en forme normale. Ces problèmes ont parfois été handicapants, car il n'est pas toujours évident de comprendre quelle partie du code ralentit les vérifications faites par agda.

Remarque A.2. Un autre problème qui a parfois été handicapant est celui des niveaux d'univers. Dans le corps du rapport, j'ai globalement passé ces problèmes sous le tapis. Mais comme expliqué dans la remarque 6.3, il y a certains arguments que je n'ai pas réussi à implémenter à cause de ces problèmes.
Dans le cas de la remarque 6.3, le problème vient du fait que si A vit dans un univers $\mathcal{U}_i$, alors $\mathcal{G}_A$ vient dans l'univers $\mathcal{U}_{i+1}$. Pour pouvoir appliquer le théorème 4.3, il faut cependant que la gerbe $\mathcal{G}$ vive dans l'univers $\mathcal{U}_i$. Une égalité entre $\mathcal{G}$ et $\mathcal{G}_A$ n'a donc pas réellement de sens dans la théorie, car ces gerbes ne vivent pas dans le même univers.
Je pense que ce problème doit pouvoir être contourné sans changer la théorie, mais

qu'il serait plus propre de trouver des règles supplémentaires concernant les univers pour les rendre plus simples à manipuler en conjonction avec l'axiome d'univalence. On pourrait par exemple imaginer une règle d'inférence de la forme

$$\frac{A : \mathcal{U}_i \qquad B : \mathcal{U}_j \qquad e : A \simeq B}{B : \mathcal{U}_i}$$

mais j'ignore si la théorie resterait cohérente.

B Torseur non trivial, Gerbe non triviale

B.1 $\mathbb{Z}$-torseur non trivial

On note S^1 le type inductif supérieur engendré par les constructeurs base : S^1 et loop : base = base. On note Helix : $S^1 \to \mathcal{U}$ l'application définie par récursion sur S^1 par :

$$\begin{aligned} \text{Helix}(\text{base}) &:\equiv \mathbb{Z} \\ \text{ap}_{\text{Helix}}(\text{loop}) &:\equiv \text{ua}(\text{suc}) \end{aligned}$$

Le nom "hélice" vient d'une intuition géométrique : on a relié chaque point de $\mathbb{Z}$ à son successeur le long d'un cercle (voir figure 1). Pour tout $x : S^1$, Helix(x) est muni d'une structure de $\mathbb{Z}$-torseur. Cependant, Helix(x) n'est pas pointé : on a même

$$\neg \left(\prod_{x:S^1} \text{Helix}(x) \right)$$

où $\neg X$ signifie $X \to \bot$. Voici l'intuition géométrique : si on avait $e(x) : \text{Helix}(x)$ pour tout $x : S^1$, alors par trivialisation de torseur on aurait pour tout $x : X$, Helix$(x) = \mathbb{Z}$. En particulier, le type $\sum_{x:S^1} \text{Helix}(x)$ serait équivalent au type $S^1 \times \mathbb{Z}$. Or le type $\sum_{x:S^1} \text{Helix}(x)$ est justement l'hélice représentée figure 1, qui est contractible, tandis que $S^1 \times \mathbb{Z}$ ne l'est bien sûr pas. D'où la contradiction.

Une première manière d'interpréter ce résultat est de dire que lorsqu'on augmente la théorie homotopique des types d'un axiome $x : S^1$, alors on peut construire un $\mathbb{Z}$-torseur Helix(x) dont il est impossible de construire un élément. En particulier, il existe des modèles de la théorie des types où on peut exhiber des torseurs non triviaux.

On peut aussi interpréter cela en terme de cohomologie : Helix peut être vu comme une application $S^1 \to \mathcal{B}^1(\mathbb{Z})$, dont on peut prouver qu'elle est différente de l'application constante $x \mapsto \mathbb{Z}$. Cela prouve donc que le type $H^1(S^1, \mathbb{Z})$ est non trivial. En fait, il est connu en topologie algébrique que $H^1(S^1, \mathbb{Z})$ est isomorphe à $\mathbb{Z}$. Il pourrait être intéresant de prouver que Helix engendre le groupe $H^1(S^1, \mathbb{Z})$, mais ce n'est pas une piste que j'ai exploré lors de mon stage.

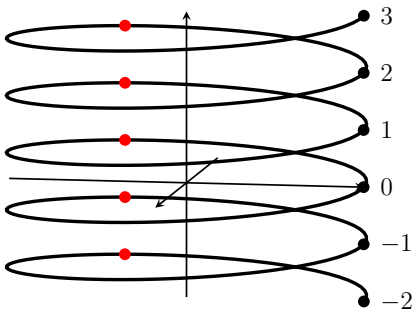

FIGURE 1 – Les points noirs représentent $\mathbb{Z}$. Les points rouges représentent $\text{Helix}(x)$ pour un $x : S^1$ quelconque. $\text{Helix}(x)$ a une structure de $\mathbb{Z}$-torseur, mais il n'y a pas de représentant canonique de 0.

L'implémentation du torseur non trivial en Agda a été faite par Victor lors de son stage en 2019.

B.2 $\mathbb{Z}$-gerbe non triviale

On peut exhiber une construction similaire pour les $\mathbb{Z}$-gerbes. En THoT, on peut définir les sphères de différentes dimensions comme des types inductifs. Ici, on va avoir besoin de S^1, S^2 et S^3. En topologie, il existe une *fibration* $S^1 \to S^3 \to S^2$ appelée la *fibration de Hopf*. Sans rentrer dans les détails, cela peut s'interpréter en THoT par l'existence d'une application

$$\text{Hopf} : S^2 \to \mathcal{U}$$

telle que pour tout $x : S^2$ on ait

$$\|\text{Hopf}(x) = S^1\|_0$$

et que

$$\left(\sum_{x:S^2} \text{Hopf}(x)\right) = S^3$$

(voir [6][8.5]). La première hypothèse permet de munir chaque $\text{Hopf}(x)$ d'une structure de $\mathbb{Z}$-gerbe. Par le même raisonnement que précédemment, si on avait un point dans chaque $\text{Hopf}(x)$, alors on aurait $\sum_{x:S^2} \text{Hopf}(x) = S^2 \times S^1$. Or on peut prouver que $S^2 \times S^1 \neq S^3$. On a donc construit une $\mathbb{Z}$-gerbe non triviale dans le contexte $x : S^2$. D'un point de vue cohomologique, cela veut dire que le groupe $H^2(S^2, \mathbb{Z})$ est non trivial, et en effet il est connu en cohomologie que ce groupe est isomorphe $\mathbb{Z}$. Je n'ai pas non plus cherché à montrer que la fibration de Hopf engendrait $H^2(S^2, \mathbb{Z})$ pendant mon stage.
Mon implémentation de la construction de cette gerbe non-triviale se trouve dans le fichier `Gerbe/NonTrivial.agda`.

Références

[1] Pierre DELIGNE. "Le symbole modéré". fr. In : *Publications Mathématiques de l'IHÉS* 73 (1991), p. 147-181. URL : http://www.numdam.org/item/PMIHES_1991__73__147_0.

[2] Jean GIRAUD. *Cohomologie non abélienne*. 1971. DOI : 10.1007/978-3-662-62103-5.

[3] Allen HATCHER. *Algebraic topology*. Cambridge : Cambridge Univ. Press, 2000. URL : https://cds.cern.ch/record/478079.

[4] Bjørn Ian DUNDAS et al. *Symmetry*. 2020. URL : https://github.com/UniMath/SymmetryBook.

[5] Daniel LICATA et Eric FINSTER. "Eilenberg-MacLane spaces in homotopy type theory". In : (juil. 2014). DOI : 10.1145/2603088.2603153.

[6] The UNIVALENT FOUNDATIONS PROGRAM. *Homotopy Type Theory : Univalent Foundations of Mathematics*. Institute for Advanced Study : https://homotopytypetheory.org/book, 2013.