UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE CIENCIAS MATEMÁTICAS

Teoría Homotópica de Tipos

Tesis para optar el Grado Académico de Magíster en Matemática

AUTOR

Fernando Rafael Chu Rivera

ASESOR

Jorge Alberto Coripaco Huarcaya

Lima - Perú
Diciembre 2022

Teoría Homotópica de Tipos

Fernando Rafael Chu Rivera

Tesis presentada a consideración del Cuerpo Docente de la Facultad de Ciencias Matemáticas, de la Universidad Nacional Mayor de San Marcos, como parte de los requisitos para obtener el Grado Académico de Magíster en Matemática.

Aprobada por:

.....
Dr. –

.....
Dr. –

.....
Dr. –

Lima - Perú
Diciembre - 2022

TEORÍA HOMOTÓPICA DE TIPOS

Fernando Rafael Chu Rivera

Diciembre - 2022

Asesor : Dr. Jorge Alberto Coripaco Huarcaya
Grado obtenido : Magíster en Matemática

. . . . . .

RESUMEN

Presentamos la Teoría Homotópica de Tipos, que uniformiza los conceptos de proposiciones y de conjuntos en uno solo más general, el de "tipos". Desarrollamos las nociones principales de esta teoría, y observamos que esta tiene una profunda estructura, que puede ser vista desde tres perspectivas distintas, la categórica, la lógica y la homotópica. Formalizamos algunos conceptos clásicos de topología algebraica, como contractibilidad, retracciones, secciones y equivalencias homotópica. Finalmente, culminamos con una demostración de que $\pi_1(\mathbb{S}^1) = \mathbb{Z}$, a modo de aplicación. Adicionalmente, comprobamos todos nuestros resultados a través de un asistente de pruebas, Agda.

Palabras clave: Teoría de Tipos, Topología Algebraica, Equivalencias Homotópicas, Teoría de Categorías, Grupo Fundamental.

HOMOTOPY TYPE THEORY

Fernando Rafael Chu Rivera

December - 2022

Advisor : Dr. Jorge Alberto Coripaco Huarcaya
Degree obtained : Master in Mathematics

. . . . . .

Abstract

We present Homotopy Type Theory, which combines the concepts of propositions and sets in a more general one, that of "types". We develop the main notions of this theory, and observe that it has a very deep structure, which can be seen through three different lenses, categorical, logical, and homotopical. We formalize some classic concepts from algebraic topology, like contractibility, retractions, sections, and homotopic equivalences. Finally, we finish with a proof that $\pi_1(\mathbb{S}^1) = \mathbb{Z}$, as an application. Additionally, we check all our results through the use of a proof assistant, Agda.

Keywords: Type Theory, Algebraic Topology, Homotopic Equivalences, Category Theory, Fundamental Group.

Dedicado a todo ser o proceso involucrado en la generación del conocimiento humano, y a todos aquellos que participaron en mi desarrollo como persona.

Agradecimientos

Agradezco a mi familia, por su constante apoyo, y por la formación que me han dado como persona. Agradezco a mis amigos que me han acompañado a lo largo de mi vida, y me han motivado a perseverar ante las adversidades.

Agradezco a los profesores que me han enseñado diversos cursos durante mis estudios de maestría, aprecio todo el conocimiento transmitido en estas interesantes materias. Agradezco en particular, a mi asesor Dr. Jorge Coripaco, por aceptar explorar conmigo el tema nuevo que es Teoría Homotópica de Tipos.

Finalmente, quiero agradecer a los organizadores y participantes de la escuela *Homotopy Type Theory Electronic Seminar Talks Summer School 2022*, así como a toda la comunidad de *Homotopy Type Theory*, por su amabilidad y su voluntad de compartir su conocimiento.

Índice general

Introducción

En la actualidad, la mayoría de las matemáticas está basada en teoría de conjuntos. Así, se pueden construir a los reales como cortes de Dedekind de los racionales; los cuales, a su vez, son clases de equivalencias de los enteros, que se construyen a partir de los naturales, y estos finalmente pueden ser definidos en términos puramente de conjuntos [8].

Sin embargo, este procedimiento se da en realidad en dos niveles distintos. En el primero, el puramente lógico, los objetos de análisis son las proposiciones, y las reglas de la lógica indican qué inferencias y deducciones son correctas, al asumir ciertas hipótesis como verdaderas. En el segundo, los objetos de estudio son los conjuntos, y son los axiomas de la teoría de conjuntos los que permiten definir qué operación o relación está bien definida. Nótese que estos dos niveles están completamente separados: la lógica sola no puede razonar sobre la teoría de conjuntos (o cualquier otra teoría), mientras que la teoría de conjuntos sola no puede razonar sobre proposiciones o inferencias lógicas.

En contraste, la Teoría Homotópica de Tipos unifica estos dos constructos, las proposiciones y los conjuntos, y solo usa una noción central, la de "tipos". Así, podemos expresar diversas proposiciones, por ejemplo, que existe el tipo de los reales ($\cdot \vdash \mathbb{R} : \mathcal{U}_i$), o que π es un elemento de este tipo ($\cdot \vdash \pi : \mathbb{R}$). También tendremos que las proposiciones mismas pueden ser representadas por un tipo, con expresiones como $\cdot \vdash p : 3^2 = 9$ dando a entender que p es una demostración de que $3^2 = 9$.

Además de tener una mayor elegancia teórica, existen múltiples ventajas de esta nueva teoría, mencionaremos solo algunas de ellas. Primero, permite formalizar algunos conceptos que intuitivamente deberían de existir y, sin embargo, son imposibles de formalizar en teoría de conjuntos. Por ejemplo, la función identidad universal, aplicable a cualquier conjunto, no está bien definida puesto que su "conjunto" de dominio y de llegada es la colección de todos los conjuntos, y este no es un conjunto, sino una clase propia.

Segundo, y más importante aún, es qué las proposiciones y las demostraciones se vuelven también objetos de estudio en el mismo nivel que otras estructuras matemáticas, como los grupos o los espacios vectoriales, por lo que se pueden analizar y manipular como es común en otras áreas de estudio. Esto será vital para la interpretación homotópica que detallaremos en el Capítulo 3, en donde una demostración de que $a = b$ se interpretará como un camino en cierto espacio topológico.

Finalmente, esta teoría permite una mayor facilidad para formalizar las de-

finiciones y la demostración de proposiciones en computadora. Hemos realizado este trabajo para todos los resultados aquí presentados, y se puede ver el código en la siguiente página web https://shiranaiyo.github.io/MastersThesis/.

Dada que la teoría es substancialmente distinta a no visto comunmente, la introduciremos lentamente. En el Capítulo 1 presentaremos algunos conceptos de Teoría de Categoría que utilizaremos a lo largo de este documento. En el Capítulo 2 presentaremos la Teoría de Tipos Dependientes, la teoría base que utilizaremos para desarrollar la Teoría Homotópica de Tipos. En el Capítulo 3 veremos la relación entre los tipos dependientes y la homotopía. En el Capítulo 4 exploraremos algunos conceptos y resultados de topología algebraica, formalizados en esta nueva teoría. En el Capítulo 5 resumimos lo desarrollado, y realizamos unos comentarios adicionales.

Capítulo 1

Preliminares categóricos

En este capítulo mencionaremos algunas definiciones clave de teoría de categorías. Seguiremos las convenciones de Riehl [20].

Definición 1.1. Una **categoría** C consiste de

- una colección $\mathrm{Ob}(\mathcal{C})$ de objetos $X, Y, Z, \ldots$, y
- una colección $\mathrm{Ar}(\mathcal{C})$ de morfismos $f, g, h, \ldots$,

tales que:

- Cada morfismo tiene dos objetos asociados, su **domino** y su **codominio**. Escribiremos $f : X \to Y$ cuando f sean un morfismo con dominio X y codominio Y.
- Cada objeto X tiene un morfismo identidad $\mathsf{id}_X : X \to X$.
- Para cada par de morfismos $f : X \to Y$ y $g : Y \to Z$, existe un morfismo $g \circ f$, llamado la composición de g y f.

Adicionalmente, se requiere que se cumplan las siguientes condiciones:

- Para todo $f : X \to Y$, se tiene que

$$\mathsf{id}_Y \circ f = f \circ \mathsf{id}_X = f.$$

- Para todos $f : X_1 \to X_2$, $g : X_2 \to X_3$, $h : X_3 \to X_4$, se tiene que

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

Nótese que utilizamos el concepto de *colección*, no de *conjunto*; pues necesitamos una mayor generalidad para las categorías que veremos a lo largo de la presente tesis.

Ejemplo 1.1. La categoría Set tiene como objetos conjuntos, y como morfismos funciones entre conjuntes. El morfismo identidad es la función identidad, y la composición de morfismos es la composición de funciones usual. Las dos condiciones adicionales se cumplen inmediatamente.

Al igual que en el ejemplo previo, en la mayoría de casos la satisfacción de las dos últimas condiciones es inmediata. Asimismo, el morfismo identidad y la composición de morfismos es usualmente la función identidad y la composición de funciones usual, respectivamente. Por estos motivos, omitiremos mención de estos detalles en los ejemplos siguiente, salvo la construcción sea diferente.

Ejemplo 1.2. La categoría Vec_K tiene como objetos espacios vectoriales sobre el cuerpo K, y como morfismos transformaciones K-lineales entre espacios vectoriales. Nuevamente, el morfismo identidad es la función identidad, y la composición es la usual.

Ejemplo 1.3. La categoría Top_* tiene como objetos pares (X, x), donde X es un espacio topológico y x es un elemento de X, el cual es llamado el elemento base. Los morfismos son funciones continuas que preservan elementos base.

Ejemplo 1.4. La categoría $\mathbf{0}$ no tiene ningún objeto ni ningún morfismo. La categoría $\mathbf{1}$ tiene un solo elemento, $\star$, y solo un morfismo $\mathsf{id}_\star$.

Ejemplo 1.5. Sea C una categoría, podemos definir una nueva categoría C^{op}, la categoría **opuesta** o **dual**. Esta está definida de la siguiente manera: los objetos son los mismos que en C, pero los morfismos tienen el dominio y codominio intercambiado. Es decir, asignamos a cada morfismo $f : X \to Y$ un morfismo $f^{\mathrm{op}} : Y \to X$.

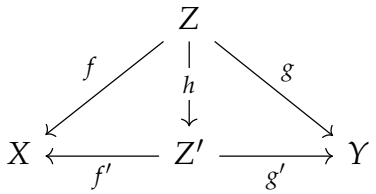
Finalmente, la composición entre dos morfismos $f^{\mathrm{op}} : Y \to X$ y $g^{\mathrm{op}} : Z \to Y$ la definimos por

$$f^{\mathrm{op}} \circ g^{\mathrm{op}} := (g \circ f)^{\mathrm{op}}.$$

Ejemplo 1.6. Sea C una categoría y sean X y Y objetos de C. A partir de estos datos, podemos formar una nueva categoría $\mathsf{C}_{X,Y}$.

Sus objetos son triples (Z, f, g), donde Z es un objeto de C, y $f : Z \to X$ y $g : Z \to Y$ son morfismos. Un morfismo $\alpha : (Z, f, g) \to (Z', f', g')$ es un morfismo $h : Z \to Z'$ tal que $f \circ h = f'$ y $g \circ h = g'$.

En otras palabras, se requiere que $h : Z \to Z'$ sea tal que el siguiente diagrama conmute.



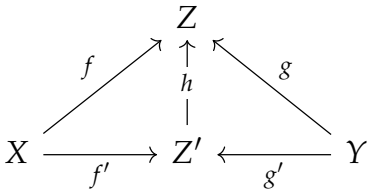
La composición de morfismos es la usual composición de morfismos.

Ejemplo 1.7. Veamos la construcción dual a la previa, es decir $\mathsf{C}^{\mathrm{op}}_{X,Y}$.

Sea C una categoría y sean X y Y objetos de C. A partir de estos datos, podemos formar una nueva categoría $\mathsf{C}^{X,Y}$.

Sus objetos son triples (Z, f, g), donde Z es un objeto de C, y $f : X \to Z$ y $g : Y \to Z$ son morfismos. Un morfismo $\alpha : (Z', f', g') \to (Z, f, g)$ es un morfismo $h : Z' \to Z$ tal que $h \circ f' = f$ y $h \circ g' = g$.

En otras palabras, se requiere que $h : Z' \to Z$ sea tal que el siguiente diagrama conmute.



La composición de morfismos es la usual composición de morfismos. Nótese que el diagrama resultante es el mismo que el del ejemplo anterior, solo que con todas las flechas volteadas.

Procederemos con algunas definiciones comunes involucrando objetos y morfismos en una categoría.

Definición 1.2. Sea C una categoría, entonces:

- Un objeto X es **inicial** si para todo objeto Y existe un único morfismo $f : X \to Y$.
- Un objeto X es **terminal** si para todo objeto Y existe un único morfismo $f : Y \to X$.
- Un morfismo $f : X \to Y$ es un **isomorfismo** si existe un morfismo $f^{-1} : Y \to X$ tal que $f \circ f^{-1} = \mathsf{id}_Y$ y $f^{-1} \circ f = \mathsf{id}_X$. Cuando esto se de, decimos que f^{-1} es una **inversa** de f, y que X y Y son **isomórficos**, lo cual denotamos por $X \simeq Y$.

Nótese que ser un objeto terminal en C es lo mismo que ser un objeto inicial en C. Veamos algunos ejemplos.

Ejemplo 1.8. En Set, el conjunto vacío $\emptyset$ es inicial, puesto que dado un conjunto X, la única función $! : \emptyset \to X$ es la trivial. Por otro lado, cualquier conjunto T con un solo elemento es terminal, pues la única función $X \to T$ es la que lleva todos los elementos de X a el único elemento de T. Finalmente, una función $f : X \to Y$ es un isomorfismo si y solo si es una biyección, siendo el morfismo inverso la función inversa usual.

Ejemplo 1.9. En Vec_K, el espacio vectorial $\mathbf{0}$ de dimensión 0 es ambos inicial y terminal, pues dado un espacio vectorial V, la única función $! : \mathbf{0} \to V$ es la que lleva el 0 al 0, y la única función $V \to \mathbf{0}$ es la que lleva todos los vectores de V a 0. Finalmente, un morfismo es un isomorfismo exactamente cuando es un isomorfismo de espacios vectoriales.

Nótese que existen múltiples objetos terminales en Set, pero todos estos son isomórficos entre sí. Esta no es una casualidad.

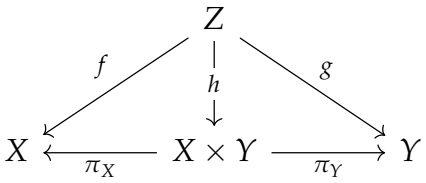
Teorema 1.0.1. *Si X y Y son objetos terminales en una categoría C, entonces $X \simeq Y$. Lo mismo se cumple si X y Y son ambos iniciales.*

Demostración. Por ser Y terminal, existe una única función $f : X \to Y$. Por ser X terminal, existe una única función $g : Y \to X$. Pero ahora vemos que tenemos dos funciones $X \to X$, las cuales son $(g \circ f)$ y id_X. Por unicidad, concluimos que $g \circ f = \mathsf{id}_X$. De manera similar, concluimos que $f \circ g = \mathsf{id}_Y$. Entonces, $X \simeq Y$.

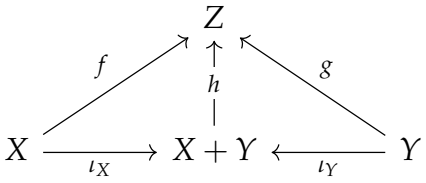
La demostración cuando X y Y son iniciales es análoga. □

Continuemos con unos ejemplos de las categorías introducidas en los Ejemplos 1.6 y 1.7.

Ejemplo 1.10. Sean X y Y conjuntos. En la categoría $\mathsf{Set}_{X,Y}$, el producto usual $X \times Y$ junto con las proyecciones es un objeto terminal. En efecto, dados Z, $f : Z \to X$ y $g : Z \to Y$ la única forma de definir una función $h : Z \to (X \times Y)$ tal que el diagrama conmute es por $h(z) = (f(z), g(z))$.



Ejemplo 1.11. Sean X y Y conjuntos. En la categoría $\mathsf{Set}^{X,Y}$ el coproducto usual $X + Y$ junto con las inserciones naturales es un objeto inicial. En efecto, dados Z, $f : X \to Z$ y $g : Y \to Z$ la única forma de definir una función $h : (X + Y) \to Z$ tal que el diagrama conmute es por $h(x) = f(x)$ y $h(y) = f(y)$.



Similares resultados aplican para otras categorías, por lo que formularemos las siguientes definiciones.

Definición 1.3. Sea C una categoría y sean X y Y objetos de C. Si Z es un objeto terminal de $\mathsf{C}_{X,Y}$, diremos que Z es el **producto** de X y Y, y lo denotaremos por $X \to Y$. Si Z es un objeto inicial de $\mathsf{C}^{X,Y}$, diremos que Z es el **coproducto** de X y Y, y lo denotaremos por $X + Y$.

Los fundadores de la teoría de categorías mencionaron que el concepto de categoría se creo para poder definir lo que es un functor, y los functores se definieron para poder definir lo que es una transformación natural [15]. Veamos estos conceptos ahora.

Definición 1.4. Sean C y D categorías. Un functor consiste de

- un objeto $F(X)$ en D para cada objeto X en C, y
- un morfismo $F(f) : F(X) \to F(Y)$ en D para cada morfismo $f : X \to Y$ en C.

tales que se cumplen las siguientes dos condiciones

- Para todo par de morfismos $f : X \to Y$ y $g : Y \to Z$ en C, se tiene que $F(g \circ f) = Fg \circ Ff$.
- Para todo objeto X en C, se tiene que $F(\mathsf{id}_X) = \mathsf{id}_{F(X)}$.

Ejemplo 1.12. El functor identidad $\mathsf{id}_\mathsf{C} : \mathsf{C} \to \mathsf{C}$ lleva cada objeto a sí mismo, y cada morfismo así mismo. Las dos condiciones adicionales se cumplen inmediatamente, y omitiremos mención de estas en los futuros ejemplos.

Ejemplo 1.13. El functor conjunto poder $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ lleva objetos a su conjunto poder, y funciones $f : X \to Y$ a una función $\mathcal{P}(f) : \mathcal{P}(X) \to \mathcal{P}(Y)$, la cual está definida por

$$\mathcal{P}(f)(S) := \{f(s) \mid s \in S\}$$

En general, cuando tengamos un functor $F : \mathsf{C} \to \mathsf{C}$ que va de una categoría C hacia si misma, diremos que es un **endofunctor**.

Ejemplo 1.14. El functor espacio dual $(-)^* : \mathsf{Vec}_K \to \mathsf{Vec}_K^{\mathrm{op}}$ lleva cada espacio vectorial V a su espacio dual V^*, y lleva una transformación lineal $f : V \to W$ a una función $f^* : W^* \to V^*$ definida por

$$f^*(\alpha)(v) := \alpha(f(v)).$$

En casos como este, cuando tengamos $F : \mathsf{C} \to \mathsf{D}^{\mathrm{op}}$, diremos que F es un functor **contravariante**; caso contrario, diremos que es **covariante**.

Ejemplo 1.15. Dados dos functores $F : \mathsf{C} \to \mathsf{D}$ y $G : \mathsf{D} \to \mathsf{E}$, podemos definir un nuevo functor $G \circ F : \mathsf{C} \to \mathsf{E}$, definido por

$$(G \circ F)(c) = G(F(c)) \text{ y } (G \circ F)(f) = G(F(f)),$$

en objetos y morfismos, respectivamente.

Para los siguientes ejemplos, necesitaremos una definición adicional.

Definición 1.5. Sea C una categoría. Diremos que C es una **categoría localmente pequeña** si para todo par de objetos X, Y de C la colección de morfismos con dominio X y codominio Y es un conjunto, y lo denotaremos por $\mathrm{Hom}_\mathsf{C}(X, Y)$. Escribiremos también $\mathrm{Hom}(X, Y)$ cuando la categoría esté sobrentendida.

Ejemplo 1.16. Para todo objeto Z en una categoría C localmente pequeña, tenemos un functor contravariante

$$\mathrm{Hom}_\mathsf{C}(-, Z) : \mathsf{C} \to \mathsf{Set}^{\mathrm{op}},$$

el cual lleva objetos X al conjunto $\mathrm{Hom}_\mathsf{C}(X, Z)$, y morfismos $f : X \to Y$ a una función de conjuntos

$$\mathrm{Hom}_\mathsf{C}(f, Z) : \mathrm{Hom}_\mathsf{C}(Y, Z) \to \mathrm{Hom}_\mathsf{C}(X, Z)$$

definida por

$$\mathrm{Hom}_{\mathsf{C}}(f, Z)(\alpha) = \alpha \circ f$$

Análogamente, tenemos un functor covariante

$$\mathrm{Hom}_{\mathsf{C}}(Z, -) : \mathsf{C} \to \mathsf{Set},$$

el cual lleva objetos X a el conjunto $\mathrm{Hom}_{\mathsf{C}}(Z, X)$, y morfismos $f : X \to Y$ a una función de conjuntos

$$\mathrm{Hom}_{\mathsf{C}}(Z, f) : \mathrm{Hom}_{\mathsf{C}}(Z, X) \to \mathrm{Hom}_{\mathsf{C}}(Z, Y)$$

definida por

$$\mathrm{Hom}_{\mathsf{C}}(Z, f)(\alpha) = f \circ \alpha$$

Ejemplo 1.17. La colección de todas las categorías localmente pequeñas es una categoría Cat, siendo los objetos las categorías, y los morfismos los functores entre las categorías.

Sean C y D categorías localmente pequeñas, se puede verificar que la categoría que tiene como objetos pares (X, Y) con $X \in \mathsf{C}$ y $Y \in \mathsf{D}$, y como morfismos $h : (X, Y) \to (X', Y')$ pares (f, g) de morfismos $f : X \to X'$ y $g : Y \to Y'$ es el producto de C y D en Cat [15].

Definición 1.6. Dadas dos categorías C y D, y dos functores $F, G : \mathsf{C} \to \mathsf{D}$, una **transformación natural** $\tau : F \to G$ consiste de

- un morfismo $\tau_c : F(X) \to G(X)$ para cada $X : C$, llamado el **componente** de τ en X

tal que para todo morfismo $f : X \to Y$ en C, el siguiente diagrama conmute

$$\begin{array}{ccc} F(X) & \xrightarrow{\tau_X} & G(X) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(Y) & \xrightarrow[\tau_Y]{} & G(Y) \end{array}$$

Ejemplo 1.18. Tenemos una transformación $\tau : \mathsf{id}_{\mathsf{Set}} \to \mathcal{P}$, definiendo cada componente por $\tau_X(x) = \{x\}$. Necesitamos comprobar que para toda función entre conjuntos $f : X \to Y$ el siguiente diagrama conmuta

$$\begin{array}{ccc} X & \xrightarrow{\tau_X} & \mathcal{P}(X) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathcal{P}(f)} \\ Y & \xrightarrow[\tau_Y]{} & \mathcal{P}(Y) \end{array}$$

Pero ahora tenemos que para todo $x \in X$

$$\begin{aligned} \mathcal{P}(f)(\tau_X(x)) &= \mathcal{P}(f)(\{x\}) \\ &= \{f(x)\} \\ &= \tau_Y(f(x)) \end{aligned}$$

Ejemplo 1.19. Tenemos una transformación natural $\tau : \mathsf{id}_{\mathsf{Vec}_K} \to (-)^{**}$, donde el functor $(-)^{**}$ es el functor espacio dual aplicado dos veces. Esta transformación está definida en cada componente por $\tau_V(v)(\alpha) = \alpha(v)$. Sea $\alpha : V \to W$, vemos que el siguiente diagrama conmuta

$$\begin{array}{ccc} V & \xrightarrow{\tau_V} & V^{**} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f^{**}} \\ W & \xrightarrow[\tau_W]{} & W^{**} \end{array}$$

pues para todo $v \in V$ y $\beta \in W^*$ tenemos

$$\begin{aligned} f^{**}(\tau_V(v))(\beta) &= \tau_V(v)(f^*(\beta)) \\ &= f^*(\beta)(v) \\ &= \beta(f(v)) \\ &= \tau_W(f(v))(\beta) \end{aligned}$$

Lo interesante del ejemplo previo es que, restringiéndonos al caso de espacios vectoriales de dimensión finita, cada componente es un isomorfismo.

Definición 1.7. Dadas dos categorías C y D, y dos functores $F, G : \mathsf{C} \to \mathsf{D}$. Diremos que una transformación natural $\tau : F \to G$ es un **isomorfismo natural** si cada componente τ_c es un isomorfismo, y escribiremos $F \cong G$.

Notamos que las transformaciones naturales se pueden componer en el siguiente sentido, si $\tau : F \to G$ y $\eta : G \to H$ son transformaciones naturales, podemos definir $\eta \circ \tau : F \to H$ por $(\eta \circ \tau)_c = \eta_c \circ \tau_c$. Para ver que esta definición satisface el requerimiento de conmutatividad, observamos que el siguiente diagrama conmuta

$$\begin{array}{ccccc} F(x) & \xrightarrow{\tau_x} & G(x) & \xrightarrow{\eta_x} & H(x) \\ {\scriptstyle F(f)}\downarrow & & \vert\,{\scriptstyle G(f)}\,\downarrow & & \downarrow{\scriptstyle H(f)} \\ F(y) & \xrightarrow[\tau_x]{} & G(y) & \xrightarrow[\eta_y]{} & H(y) \end{array}$$

pues cada uno de los cuadrados individuales conmuta.

Esto hace que la colección de functores $\mathrm{Hom}_{\mathsf{Cat}}(X, Y)$ sea una categoría, siendo los objetos los functores, y los morfismos las transformaciones naturales entre ellos.

De esta manera, vemos que un isomorfismo en esta categoría es un isomorfismo natural τ, pues podemos definir una transformación natural inversa $\tau^{-1} : G \to F$ dada por $(\tau^{-1})_d = (\tau_d)^{-1}$, y esta satisface que $\tau \circ \tau^{-1} = \mathsf{id}$ y $\tau^{-1} \circ \tau = \mathsf{id}$.

Sin embargo, este propiedad rara vez se da en aplicaciones, una definición similar, y más conveniente es la siguiente.

Definición 1.8. Una **equivalencia de categorías** consiste de dos functores $F : C \to D$ y $G : D \to C$ tales que existen isomorfismos naturales $F \circ G \cong \mathsf{id}_{\mathsf{D}}$ y $G \circ F \cong \mathsf{id}_{\mathsf{C}}$. Cuando se de esto, escribiremos $\mathsf{C} \simeq \mathsf{D}$.

Ejemplo 1.20. Un ejemplo clásico de equivalencia es la adjunción Tensor-Hom de espacios vectoriales, o de una manera más general, de R-módulos:

$$\mathrm{Hom}(M \times N, P) \simeq \mathrm{Hom}(M, \mathrm{Hom}(N, P)),$$

para todo trío de R-módulos M, N, P. La demostración de este resultado se puede encontrar en [3].

Más general aún es la siguiente equivalencia entre las categorías

$$\mathrm{Hom}_{\mathsf{Cat}}(X \times Y, Z) \simeq \mathrm{Hom}_{\mathsf{Cat}}(X, \mathrm{Hom}_{\mathsf{Cat}}(Y, Z)),$$

la cual se puede encontrar en [15].

Por último, presentamos las siguientes definiciones.

Definición 1.9. Una categoría C es una **subcategoría** de una categoría D si todos los objetos de C son objetos de D, y todos los morfismos de C son morfismos de D.

Decimos que C es una **subcategoría completa** si $\mathrm{Hom}_{\mathsf{C}}(X, Y) = \mathrm{Hom}_{\mathsf{D}}(X, Y)$, para todo par de objetos X, Y en C.

Ejemplo 1.21. La categoría $\mathsf{Set}_{\mathsf{Fin}}$, que tiene como objetos conjuntos finitos y como morfismos funciones entre estos conjuntos, es una subcategoría completa de Set.

Ejemplo 1.22. Dada una colección de objetos C de una categoría D, la subcategoría completa generada por C es aquella que tiene como objetos los objetos de C, y tiene como morfismos de X, Y en C, la colección $\mathrm{Hom}_{\mathsf{D}}(X, Y)$. La identidad y la composición son las de D.

Capítulo 2

Teoría de Tipos Dependientes

En este capítulo daremos una descripción formal y rigurosa de la Teoría de Tipos Dependientes (DTT, en adelante, por sus siglas en inglés), haciendo una comparación a la Matemática Clásica (MC, en adelante) cuando sea conveniente.

2.1. Expresiones, términos y contextos

Dado que las proposiciones y las demostraciones son objetos mismos en DTT, es necesario tener mayor precisión respecto a los significados de ciertos términos.

Definición 2.1.1. Una **signatura** Sg es una colección de símbolos tipográficos. Una **expresión** Exp_{Sg} es una secuencia finita de símbolos provenientes de la signatura Sg.

Ejemplo 2.1.2. La MC utiliza como signatura la colección que contiene a los caracteres alfanuméricos, los conectores lógicos y a los operadores; es decir,

$$Sg = \{\text{`1'}, \text{`('}, \text{`}n\text{'}, \text{`+'}, \text{`}\Sigma\text{'}, \text{`}x_i\text{'}, \dots\}.$$

Una expresión típica de MC es $\langle \text{`1'}, \text{`+'}, \text{`1'}, \text{`='}, \text{`2'} \rangle_{Sg}$.

Notación 2.1.3. Asumiremos siempre que esta misma signatura está implícita en todas las próximas expresiones que escribamos. Además, las expresiones se escribirán sin comillas, sin corchetes y sin hacer referencia a esta signatura.

Justificación. Puesto que la signatura será la misma para el resto de este documento y dado que existe una sola forma de interpretar una expresión simplificada como una secuencia de caracteres, no habrá riesgo de ambigüedad. □

La justificación de esta notación es simple, pero se ha dado para enfatizar que *alguna* justificación es necesaria. En las próximas notaciones, omitiremos las justificaciones triviales y solo justificaremos las más complicadas de realizar.

Para nuestras próximas definiciones, necesitaremos el siguiente concepto.

Definición 2.1.4. Una **metavariable** es una variable que puede tomar como valor cualquier expresión, a excepción de aquellas que contienen alguno de los símbolos '≡', ':', '⊢' o 'ctx'.

La razón por la que omitimos estos tres símbolos de la definición es porque estos son *caracteres reservados*, los cuales tienen un significado particular y serán introducidos en la próxima sección.

Definición 2.1.5. Sean a y A metavariables, dada una expresión de la forma $((a) : (A))$, diremos que A es un **tipo** y que a es un **término** o **elemento** del tipo A.

Ejemplo 2.1.6. En la expresión $((2 + n) : (\mathbb{N}))$, $\mathbb{N}$ es un tipo y $(2 + n)$ es un término de tipo A.

Notación 2.1.7. En la mayoría de casos, omitiremos los paréntesis, entendiendo que el símbolo ':' tiene menor precedencia que otros símbolos por introducir, a excepción de los símbolos ',' (en contextos), y '$\vdash$' y 'ctx' (en juicios).

Para el resto de este documento utiliaremos metavariables sin mencionar que lo son, dejando la tarea de discernirlas al lector.

Existen ciertas semejanzas entre $a : A$ en DTT y $a \in A$ en MC, pero también hay algunas diferencias importantes. Primero, a no es un elemento que existe independientemente de A; es decir, un término siempre debe estar acompañado del tipo al que corresponde. Relacionado a esto, un término pertenece únicamente a un tipo (con una excepción, ver Sección 2.3), mientras que en teoría de conjuntos un elemento puede pertenecer a varios conjuntos.

Definición 2.1.8. Un **contexto** es una lista finita de expresiones de la forma $a : A$. Es decir, un contexto es una expresión de la forma

$$\langle x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n \rangle.$$

Los x_i son llamados **variables**.

Ejemplo 2.1.9. La expresión $\langle n{:}\mathbb{N}, v{:}\mathbb{R}^n, M{:}\mathbb{R}^{n\times n}, Mv{:}\mathbb{R}^n \rangle$ es un contexto.

Notación 2.1.10. Omitiremos siempre los corchetes, entendiendo que los símbolos ',' en la lista tienen menor precedencia que otros símbolos por introducir, a excepción de los símbolos '$\vdash$' y 'ctx' en juicios. Además, denotamos el contexto vacío con el símbolo '$\cdot$'.

Nótese que las expresiones y los contextos pueden no estar bien formadas; es decir, pueden ser una secuencia de símbolos sin significado alguno, como $\int 0/0 : \sin \mathbb{Q}$. Los juicios evitan este problema.

2.2. Juicios y reglas de inferencia

Definición 2.2.1. Sea Γ es un contexto, un **juicio** es una expresión de una de las tres siguientes formas:

$$(\Gamma)\ \text{ctx} \qquad (\Gamma) \vdash (a : A) \qquad (\Gamma) \vdash (a \equiv a' : A)$$

Notación 2.2.2. Omitiremos siempre los paréntesis, entendiendo que los símbolos '$\vdash$' y 'ctx' tienen menor precedencia que todos los otros símbolos. En el caso de los últimos dos tipos de juicios, omitiremos la mención del contexto Γ y el símbolo $\vdash$ cuando el contexto no sea relevante o este implícito.

La noción de juicios en DTT toma un rol similar al de proposiciones en MC. El significado de estos juicios es detallado en el siguiente cuadro.

Juicio	Interpretación
Γ ctx	Γ es un contexto bien formado; es decir, una lista de suposiciones bien formadas.
$\Gamma \vdash a : A$	El contexto Γ implica que a es un elemento del tipo A.
$\Gamma \vdash a \equiv a' : A$	El contexto Γ implica que a y a' son objetos iguales por definición del tipo A.

Cuadro 2.1: Juicios en DTT y su significado.

Notamos que el tercer tipo de juicios se refiere solo a igualdades por definición, en la Sección 2.10 introduciremos otra noción de igualdad. Por otro lado, los primeros dos juicios formalizan la noción de que una expresión tenga "sentido".

Definición 2.2.3. Una expresión de la forma a se dice **bien tipada** si es que existe un contexto Γ y un tipo A tal que Γ ctx y $\Gamma \vdash a : A$.

Similarmente, una expresión de la forma $b : B$ se dice **bien tipada** si es que existe un contexto Γ tal que Γ ctx y $\Gamma \vdash a : A$.

De esta manera, veremos que la expresión previa, $\int 0/0 : \sin \mathbb{Q}$, no está bien tipada. Para llegar a esta conclusión, debemos entender el proceso a través del cual llegamos a estos juicios: la aplicación de reglas de inferencia.

Definición 2.2.4. Una **regla de inferencia** es de la forma

$$\frac{\mathcal{H}_1 \qquad \cdots \qquad \mathcal{H}_k}{\mathcal{C}}\ \text{NOMBRE}$$

donde $\mathcal{H}_1, \ldots, \mathcal{H}_k$ y $\mathcal{C}$ son expresiones. Las expresiones $\mathcal{H}_1, \ldots, \mathcal{H}_k$ son llamadas **hipótesis**, mientras que $\mathcal{C}$ es llamada la **conclusión**.

Escribimos a la derecha el nombre de la regla, para ser referenciada posteriormente. Cabe notar que una regla puede tener restricciones adicionales que deben ser corroboradas antes de poder ser aplicada. Si la lista de hipótesis es muy larga, las apilaremos unas sobre otras (ver Reglas 2.7.7). Las reglas toman un rol similar al de la deducción lógica en MC.

Definición 2.2.5. Una **derivación** de un juicio es un árbol invertido con el juicio por derivar en la raíz del árbol, donde el paso de un nodo a otro nodo está justificado por una regla de inferencia.

Ejemplo 2.2.6. Con las reglas que presentaremos posteriormente, el siguiente árbol es una derivación de $\cdot \vdash \mathbf{0} + \mathbf{1} : \mathcal{U}_0$.

$$\dfrac{\dfrac{\dfrac{}{\cdot \text{ ctx}}\,\text{ctx-EMP}}{\cdot \vdash \mathbf{0} : \mathcal{U}_0}\,\text{0-FORM} \qquad \dfrac{\dfrac{}{\cdot \text{ ctx}}\,\text{ctx-EMP}}{\cdot \vdash \mathbf{1} : \mathcal{U}_0}\,\text{1-FORM}}{\cdot \vdash \mathbf{0} + \mathbf{1} : \mathcal{U}_0}\,\text{+-FORM}$$

Con los conceptos previos ya definidos, comenzaremos a introducir las reglas de inferencias de DTT.

2.3. Universos

Como mencionamos previamente, todo término es un elemento de un tipo. Para ser manipulados efectivamente, estos, a su vez, son elementos de otro tipo.

Definición 2.3.1. El tipo de un tipo es llamado un **universo**.

A fin de evitar la paradoja de Russell[1], un universo no es un elemento de sí mismo. Al contrario, existe una infinita jerarquía de universos, la cual se ve formalizada en las siguientes reglas.

Reglas 2.3.2. (Reglas básicas de universos)

$$\dfrac{\Gamma \text{ ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\,\mathcal{U}_i\text{-INTRO} \qquad \dfrac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\,\mathcal{U}_i\text{-CUMUL}$$

La primera regla indica que si Γ es un contexto bien formado, entonces Γ implica que el universo $\mathcal{U}_i$ es un elemento del universo $\mathcal{U}_{i+1}$. La segunda regla indica que si Γ implica que A es un elemento del universo $\mathcal{U}_i$, entonces Γ implica que A también es un elemento del universo $\mathcal{U}_{i+1}$.

Dado que son las primeras reglas introducidas, hemos brindado una interpretación de ellas. Para las próximas reglas no realizaremos este tipo de comentarios, salvo para aclarar alguna posible confusión. La lista completa de reglas se encuentra en el Apéndice A.

Notación 2.3.3. Omitiremos los subíndices de los universos en la mayoría de escenarios, por lo que interpretaremos expresiones sin sentido como $\mathcal{U} : \mathcal{U}$ agregando índices adecuados, obteniendo $\mathcal{U}_i : \mathcal{U}_{i+1}$. Esta práctica puede traer inconsistencias si no es manejada con precisión, pero la usaremos igualmente para reducir la carga notacional.

Nótese que los subíndices no son elementos del tipo de los naturales, sino son parte del símbolo '$\mathcal{U}_i$'; es decir, consideramos a '$\mathcal{U}_i$' como un solo caracter. Por este motivo, expresiones como $n : \mathbb{N} \vdash A : \mathcal{U}_n$ no están bien formadas.

Con los universos ya definidos, introducimos las reglas respecto a la formación de contextos.

[1] La paradoja de Russell [21] es la siguiente: sea X el conjunto de todos los conjuntos que no se contienen a sí mismos. Si X se contiene a sí mismo, no debería de contenerse a sí mismo por definición. Si X no se contiene a sí mismo, sí debería de hacerlo por definición. Entonces ambos casos llevan a una contradicción.

Reglas 2.3.4. (Reglas básicas de contextos y variables)

$$\frac{}{\cdot \text{ ctx}}\ \text{ctx-EMP} \qquad \frac{x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \text{ctx}}\ \text{ctx-EXT}$$

$$\frac{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \text{ctx}}{x_1{:}A_1, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \text{Vble}$$

donde la regla ctx-EXT tiene la condición adicional de que la variable x_n debe ser distinta a las demás variables $x_1, \ldots, x_{n-1}$, y la regla Vble requiere que $1 \leq i \leq n$.

Nótese que la regla ctx-EMP tiene 0 hipótesis, por lo que siempre es posible aplicarla. A continuación, detallamos el comportamiento de igualdades por definición.

Reglas 2.3.5. (Reglas básicas de igualdades por definición)

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

Las tres primeras reglas de $\equiv$ indican que esta es una relación de equivalencia, mientras las otras formalizan el buen comportamiento de juicios respecto a tipos iguales por definición.

En las siguientes secciones introduciremos reglas para la formación, introducción y eliminación de algunos constructos. Para cada una de estas reglas, existe una regla correspondiente indicando que estas reglas preservan la igualdad por definición. Como es común es la presentación de reglas de DTT, estas serán omitidas.

2.4. El tipo de funciones

El concepto de una funciones es fundamental en las diferentes áreas de investigación de la Matemática Clásica, en esta sección presentaremos la versión análoga a esta noción en Teoría de Tipos Dependientes.

Definición 2.4.1. Dados dos tipos A y B, el tipo $A \to B$ es llamado el **tipo de funciones** de A a B. Un elemento $f : A \to B$ es llamado una **función**. En este caso, decimos que A es el **dominio** de f y B es el **codominio** de f.

Intuitivamente, para construir una función $f : A \to B$, es suficiente que, dado $x : A$, podamos generar una expresión $b : B$ que esté bien definida, donde b puede contener la variable x dentro de ella. Esto sugiere las siguientes reglas.

Reglas 2.4.2. (Reglas de formación de funciones)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A \to B : \mathcal{U}_i}\ \to\text{-FORM} \qquad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda(x : A).\, b : A \to B}\ \to\text{-INTRO}$$

La expresión $\lambda(x:A).b$ puede entenderse como la función $f : A \to B$ definida por $f(x) = b$, solo que se ha omitido el nombre f. Por este motivo, a veces en la literatura estas funciones se llaman funciones anónimas. En este documento nosotros las llamaremos funciones lambda, el nombre original que les dio Alonzo Church [6].

En la expresión $\lambda(x:A).b$, se dice que x es una **variable ligada** en b. Esto es similar a cómo las expresiones ''$\forall x$' o ''$\int - \mathrm{dx}$' ligan la variable x dentro de estas. Si una variable no es ligada, se dice que es una **variable libre**.

Si f es una función de A en B, entonces podemos aplicarla a un elemento $a : A$ para conseguir un elemento de $b : B$. Intuitivamente, esto se da reemplazando todos las apariciones de x por a. Este proceso se ve formalizado a través de las siguientes reglas.

Reglas 2.4.3. (Reglas de aplicación de funciones)

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B} \to\text{-ELIM}$$

$$\frac{\Gamma, x : A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).b)(a) \equiv b[a/x] : B} \to\text{-COMP}$$

donde la expresión $b[a/x]$ indica que reemplazaremos todas las apariciones libres de x con a.

Nótese que la regla $\to$-COMP implica que la imagen de una función es única, mientras que en MC una función se define como una relación que cumple esta propiedad.

Ejemplo 2.4.4. Para este ejemplo, asumiremos la existencia del tipo de los naturales $\mathbb{N}$, el cual será introducido posteriormente.

Formaremos la función identidad en $\mathbb{N}$ y la aplicaremos en 0. Sea Γ igual a $\mathbb{N}{:}\mathcal{U}$, entonces

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{\Gamma, n{:}\mathbb{N}\ \mathrm{ctx}}\ \text{ctx-Ext}}{\Gamma, n{:}\mathbb{N} \vdash n : \mathbb{N}}\ \text{Vble}}{\Gamma \vdash \lambda(n:\mathbb{N}).n : \mathbb{N} \to \mathbb{N}}\ \to\text{-INTRO}$$

Esto muestra que la función existe y tiene el tipo adecuado. Sea Γ igual a $\mathbb{N}{:}\mathcal{U}, 0{:}\mathbb{N}$, veremos que la función se comporta adecuadamente

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{\Gamma, n{:}\mathbb{N}\ \mathrm{ctx}}\ \text{ctx-Ext}}{\Gamma, n{:}\mathbb{N} \vdash n : \mathbb{N}}\ \text{Vble} \qquad \dfrac{\Gamma}{\Gamma \vdash 0 : \mathbb{N}}\ \text{Vble}}{\Gamma \vdash (\lambda(n:\mathbb{N}).n)(0) \equiv n[0/n] : \mathbb{N}}\ \to\text{-COMP}}{\Gamma \vdash (\lambda(n:\mathbb{N}).n)(0) \equiv 0 : \mathbb{N}}$$

Definición 2.4.5. Sea A un tipo, la función identidad $\mathsf{id}_A : A \to A$ está definida por

$$\mathsf{id}_A :\equiv \lambda(x:A).x$$

Notación 2.4.6. Usaremos el símbolo $:\equiv$ para dar definiciones. Una definición debe considerarse solo como una abreviación. El símbolo $:\equiv$ no pertenece a DTT per se, sino solo es un mecanismo para reducir la notación en la práctica matemática.

Como mencionamos en la introducción, ya esta función simple representa una mejora respecto a MC. Dado el tipo de todos los conjuntos Set, la función $\mathsf{id}_{\mathsf{Set}}$ es la función identidad en todos los conjuntos, un concepto imposible de formalizar en MC.

Notación 2.4.7. Omitiremos a veces el tipo de una expresión y su contexto asociado, cuando estos no sean relevantes o sean posibles de inferir fácilmente. De esta forma, escribiríamos el juicio derivado en el ejemplo previo como $\mathsf{id}_{\mathbb{N}}(0) \equiv 0$.

Para introducir funciones de varias variables, podríamos introducir el tipo de productos $A \times B$ y definir $f : (A \times B) \to C$. Una alternativa equivalente, pero más conveniente, es introducir el uso de funciones "currificadas" (*curried functions*), nombradas así en honor a Haskell Curry [7].

La currificación de una función $f : (A \times B) \to C$ es $f' : A \to (B \to C)$, de tal forma que si $a : A$ y $b : B$, entonces $f'(a)(b) : C$.

Ejemplo 2.4.8. Formaremos una función $A \to (B \to A)$, la cual ignora el valor de la segunda variable, y solo devuelve el valor de la primera variable. Sea $\Gamma :\equiv A{:}\mathcal{U}, B{:}\mathcal{U}$, tenemos

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma}{\Gamma, x{:}A \text{ ctx}}\,\text{ctx-Ext}}{\Gamma, x{:}A, y{:}B \text{ ctx}}\,\text{ctx-Ext}}{\Gamma, x{:}A, y{:}B \vdash x{:}A}\,\text{Vble}}{\Gamma, x{:}A \vdash \lambda(y:B).\, x : B \to A}\,\to\text{-INTRO}}{\Gamma \vdash \lambda(x:A).\,(\lambda(y:B).\, x) : A \to (B \to A)}\,\to\text{-INTRO}$$

Poniendo $\Gamma' :\equiv A{:}\mathcal{U}, B{:}\mathcal{U}, a{:}A, b{:}B$, el buen comportamiento es mostrado por

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'}{\Gamma', x{:}A \text{ ctx}}\,\text{ctx-Ext}}{\Gamma', x{:}A, y{:}B \text{ ctx}}\,\text{ctx-Ext}}{\Gamma', x{:}A, y{:}B \vdash a : A}\,\text{Vble}}{\Gamma', x{:}A \vdash \lambda(y:B).\, a : B \to A}\,\to\text{-INTRO} \qquad \dfrac{\Gamma'}{\Gamma' \vdash a : A}\,\text{Vble}}{\dfrac{\Gamma' \vdash \lambda(x:A).\,(\lambda(y:B).\, x)(a) \equiv (\lambda(y:B).\, x)[a/x] : B \to A}{\Gamma' \vdash \lambda(x:A).\,(\lambda(y:B).\, x)(a) \equiv \lambda(y:B).\, a : B \to A}}\,\to\text{-COMP}$$

y

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'}{\Gamma', y{:}B \text{ ctx}}\,\text{ctx-Ext}}{\Gamma', y{:}B \vdash a : A}\,\text{Vble} \qquad \dfrac{\Gamma'}{\Gamma' \vdash b : B}\,\text{Vble}}{\Gamma' \vdash (\lambda(y:B).\, a)(b) \equiv a[b/y] : A}\,\to\text{-COMP}}{\Gamma' \vdash (\lambda(y:B).\, a)(b) \equiv a : A}$$

Juntando estas dos derivaciones, obtenemos que

$$\lambda(x:A).\,(\lambda(y:B).\,x)(a)(b) \equiv (\lambda(y:B).\,a)(b) \equiv a$$

Notación 2.4.9. Cuando puedan ser inferidas o no sean relevantes, omitiremos también el tipo de las variables dentro de una función lambda. Por ejemplo, escribiríamos $\lambda(x:A).\,\lambda(y:B).\,\Phi$ como $\lambda x.\,\lambda y.\,\Phi$.

Si tenemos $f : A_1 \to \cdots \to (A_{n-1} \to A_n)$, escribiremos $f(x_1)(x_2)\cdots(x_n)$ como $f(x_1, x_2, \ldots, x_n)$. Esto no traerá ambigüedad, puesto que el tipo de f indicará si es una función currificada o una función cuyo dominio es un producto de tipos.

Una operación común entre funciones es la composición, la siguiente derivación muestra que esta operación efectivamente existe.

Ejemplo 2.4.10. Sea $\Gamma :\equiv A{:}\mathcal{U}, B{:}\mathcal{U}, C{:}\mathcal{U}$, primero veremos que podemos expandir este contexto adecuadamente; es decir $\Gamma \vdash \Gamma, g{:}B \to C, f{:}A \to B, x{:}A$ ctx.

$$\dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \vdash B : \mathcal{U}_i \text{ ctx}}\,\text{Vble} \qquad \dfrac{\Gamma \text{ ctx}}{\Gamma \vdash C : \mathcal{U}_i \text{ ctx}}\,\text{Vble}}{\Gamma \vdash B \to C{:}\mathcal{U}_i}\,{\to}\text{-FORM}}{\Gamma, (g{:}B \to C) \text{ ctx}}\,\text{ctx-Ext}$$

Además,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, (g{:}B \to C) \text{ ctx}}{\Gamma, (g{:}B \to C) \vdash A : \mathcal{U}_i}\,\text{Vble} \qquad \dfrac{\Gamma, (g{:}B \to C) \text{ ctx}}{\Gamma, (g{:}B \to C) \vdash B : \mathcal{U}_i}\,\text{Vble}}{\Gamma, (g{:}B \to C) \vdash A \to B{:}\mathcal{U}_i}\,{\to}\text{-FORM}}{\Gamma, (g{:}B \to C), (f{:}A \to B) \text{ ctx}}\,\text{ctx-Ext}}{\Gamma, (g{:}B \to C), (f{:}A \to B) \vdash x : A}\,\text{Vble}}{\Gamma, (g{:}B \to C), (f{:}A \to B), (x : A) \text{ ctx}}\,\text{ctx-Ext}$$

Ahora, tenemos que

$$\dfrac{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \text{ ctx}}{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \vdash f : A \to B}\,\text{Vble}$$

$$\dfrac{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \text{ ctx}}{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \vdash x : A}\,\text{Vble}$$

Juntando estas dos derivaciones, obtenemos

$$\dfrac{\Gamma, \ldots, (x{:}A) \vdash f : A \to B \qquad \Gamma, \ldots, (x{:}A) \vdash x : A}{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \vdash f(x) : B}\,{\to}\text{-ELIM}$$

Similarmente, tenemos que

$$\dfrac{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \text{ ctx}}{\Gamma, (g{:}B \to C), (f{:}A \to B), (x{:}A) \vdash g : B \to C}\,\text{Vble}$$

Finalmente, llegamos a

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma,\ldots,(x{:}A)\vdash g : B\to C \qquad \Gamma,\ldots,(x{:}A)\vdash f(x) : B}{\Gamma,(g{:}B\to C),(f{:}A\to B),(x{:}A)\vdash g(f(x)) : C}\to\text{-ELIM}}{\Gamma,(g{:}B\to C),(f{:}A\to B)\vdash \lambda(x:A).\, g(f(x)) : A\to C}\to\text{-INTRO}}{\Gamma,(g{:}B\to C)\vdash \lambda f.\,(\lambda(x:A).\, g(f(x))) : (A\to B)\to(A\to C)}\to\text{-INTRO}}{\Gamma\vdash \lambda g.\,\lambda f.\,(\lambda(x:A).\, g(f(x))) : (B\to C)\to(A\to B)\to(A\to C)}\to\text{-INTRO}$$

Esto muestra que la función

$$\begin{aligned}&\mathsf{comp}_{A,B,C} : (B\to C)\to(A\to B)\to(A\to C)\\ &\mathsf{comp}_{A,B,C} :\equiv \lambda(g:B\to C).\,\lambda(f:A\to B).\,(\lambda x{:}A.\, g(f(x)))\end{aligned}$$

existe. La prueba de que esta función se comporta como esperaríamos, es decir que $\mathsf{comp}_{A,B,C}(g,f,x)\equiv g(f(x))$, es semejante a la del Ejemplo 2.4.8, y por lo tanto la omitiremos.

También se puede verificar el hecho conocido de que la composición es asociativa; es decir, que se tiene

$$\mathsf{comp}_{A,C,D}(h,\mathsf{comp}_{A,B,C}(g,f))\equiv \mathsf{comp}_{A,B,D}(\mathsf{comp}_{B,C,D}(h,g),f)$$

Con esta operación de composición de funciones entonces obtenemos una categoría.

Definición 2.4.11. La categoría Type tiene como objetos tipos, y como morfismos funciones entre tipos. El morfismo identidad asociado a un objeto A está dado por la función identidad id_A.

La última regla de funciones indica que si formamos una nueva función lambda, que recibe x y devuelve $f(x)$ entonces esta es la misma función que la f original.

Reglas 2.4.12. (Principio de unicidad para funciones)

$$\dfrac{\Gamma\vdash f : A\to B}{\Gamma\vdash f\equiv(\lambda(x:A).\, f(x)) : A\to B}\to\text{-UNIQ}$$

Nótese que esta regla no dice que si tenemos dos funciones $f,g : A\to B$ tales que $f(x)\equiv g(x)$ para todo x, entonces $f\equiv g$. Esta propiedad, apropiadamente modificada, será introducida en la Sección 3.7.

2.5. El tipo de funciones dependientes

En MC, una práctica común es utilizar sucesiones infinitas de elementos x_i de un conjunto X. Formalmente, esto corresponde a una función $f : \mathbb{N}\to X$. Similarmente, a veces es necesario indexar no solo elementos, sino conjuntos con cierta estructura por otros conjuntos.

Por ejemplo, a cada punto p en una variedad M, le corresponde su espacio tangente T_pM. Se entiende que esta última construcción está asociada a cierta función $g : M \to \bigcup_{p\in M}\{T_pM\}$. En este último caso, es común decir que $\{T_pM\}_{p\in M}$ es una familia de conjuntos indexada por $p \in M$. Presentamos el constructo análogo en DTT.

Definición 2.5.1. Si tenemos que $\Gamma, x : A \vdash B : \mathcal{U}$, entonces diremos que B es una **familia de tipos** indexada por $x : A$.

Recordamos que B no es el carácter 'B', sino una metavariable, por lo que esta puede contener la variable x dentro de sí. Además, notamos que el requerimiento que $\Gamma, x : A \vdash B : \mathcal{U}$ es equivalente a que exista un $f : A \to \mathcal{U}$, por lo que en este caso también diremos que B es una familia de tipos.

Ejemplo 2.5.2. Cada punto x en un espacio topológico X tiene asociado un grupo, su grupo fundamental. Es decir, $\Gamma, x : X \vdash \pi_1(x, X) : \mathcal{U}$.

Ejemplo 2.5.3. Sea $B : \mathcal{U}$ un tipo, para todo $a : A$, se tiene $\Gamma, a : A \vdash B : \mathcal{U}$. En este caso, decimos que B es una **familia constante**.

Nótese que en el caso de familias constantes, una función $f : A \to B$ asigna a cada $a : A$ un elemento $f(a) : B$. Esto se puede generalizar para familias arbitrarias.

Ejemplo 2.5.4. Sea X un espacio topológico, para cada $x : X$ podemos escoger un elemento de $\pi_1(x, X) : \mathcal{U}$, el elemento neutro de ese grupo $0_{\pi_1(x,X)}$.

En este último ejemplo tenemos una regla de correspondencia, por lo que se esperaría que podemos formar una función f con el tipo $X \to \pi_1(x, X)$. Sin embargo, este tipo previo no está bien tipado, el x que aparece en el codominio no está definido. El problema es que el codominio depende del $x : X$ escogido en el dominio, para estos casos necesitaremos el tipo de **funciones dependientes** de A en B, denotado por $\prod_{(x:A)} B$.

Reglas 2.5.5. (Reglas de formación de funciones dependientes)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-FORM}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda(x : A).\, b : \prod_{(x:A)} B}\ \Pi\text{-INTRO}$$

donde la expresión $\prod_{(x:A)} B$ liga a x hasta el final de esta.

Con estas reglas, se puede definir el tipo $\prod_{(x:X)} \pi_1(x, X)$, y una de las funciones pertenecientes a este tipo sería $\lambda(x : X).\, 0_{\pi_1(x,X)}$.

Igual que en el caso de funciones no dependientes, tenemos reglas que nos indican el proceso de aplicación de estas funciones, así como un principio de unicidad.

Reglas 2.5.6. (Reglas de aplicación de funciones)

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-ELIM}$$

$$\frac{\Gamma, x : A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-COMP}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda(x:A).f(x)) : \prod_{(x:A)} B}\ \Pi\text{-UNIQ}$$

Notamos que estas reglas son generalizaciones directas de las reglas de funciones introducidas en la sección anterior. En efecto, tomaremos estas como las reglas oficiales y definiremos el tipo $A \to B$ como $\prod_{(x:A)} B$; puesto que en este caso B es una familia constante, y se tiene que $B[a/x]$ es B, con lo que se obtienen las reglas originales.

Notación 2.5.7. Los variables ligadas por Π tienen menor precedencia que otros operadores dentro de un tipo, por lo que $\prod_{(x:A)} B \to C$ se entiende como $\prod_{(x:A)}(B \to C)$, por ejemplo. Además, para acentuar el énfasis de la (posible) dependencia de B sobre x, escribiremos $\prod_{(x:A)} B(x)$, entendiendo por esto como simplemente $\prod_{(x:A)} B$. Finalmente, mencionamos que existe otra notación alternativa para $\prod_{(x:A)} B(x)$, como

$$\prod_{x:A} B(x) \quad \text{y} \quad \Pi(x:A), B(x).$$

El uso del símbolo Π sugiere que este tipo puede ser interpretado también como el producto cartesiano de lo B_i, como lo muestra el próximo ejemplo. La siguiente discusión es informal, y utilizaremos algunos términos de teoría de conjuntos para ayudar la exposición. Sin embargo, estas nociones no están definidas para DTT en este punto.

Ejemplo 2.5.8. Sea A un tipo con tres elementos a_1, a_2, a_3 y sean X, Y, Z tres tipos con m, n y p elementos respectivamente. Podemos asignar al elemento a_1 el tipo X, al elemento a_2 el tipo Y y al elemento a_3 el tipo Z. Esto nos da una familia de tipos $B : A \to \mathcal{U}$.

Una función $f : \prod_{(x:A)} B$ asigna a cada $x : A$ un elemento de $B(x)$.

$x : A$	$B(x)$
a_1	$B(a_1) \equiv X \equiv \{x_1, \ldots, x_m\}$
a_2	$B(a_2) \equiv Y \equiv \{y_1, \ldots, y_n\}$
a_3	$B(a_3) \equiv Z \equiv \{z_1, \ldots, z_p\}$

Definamos la función $f_{i,j,k} : \prod_{(x:A)} B(x)$ por $f(a_1) = x_i, f(b_1) = y_j, f(a_3) = z_k$ donde $1 \leq i \leq m$, $i \leq j \leq n$, $1 \leq k \leq p$. De esta forma, vemos que los elementos de $\prod_{(x:A)} B(x)$ corresponden exactamente a triples (x, y, z) con $x : X, y : Y$ y $z : Z$.

Así, el número de funciones en $\prod_{(x:A)} B(x)$, es el número de elecciones posibles para cada a_i, es decir

$$\left|\prod_{x:A} B(x)\right| = \prod_{x:A} |B(x)| = |X| \cdot |Y| \cdot |Z|$$

El hecho de que $|A \to B| = |B|^{|A|}$ es un caso particular de esto.

Una pequeña limitación de la función $\mathsf{id}_A : A \to A$ es que está definida solo para el tipo A. Formalmente, tendríamos que crear una nueva función id_T para cada tipo T. Las funciones dependientes evitan la repetición de este proceso.

Ejemplo 2.5.9. Existe una función id que asigna a cada $A : \mathcal{U}$, su función identidad $\mathsf{id}_A : A \to A$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A : \mathcal{U}\ \text{ctx}}{A : \mathcal{U}, x : A\ \text{ctx}}\ \text{ctx-Ext}}{A : \mathcal{U}, x : A \vdash x : A}\ \text{Vble}}{A : \mathcal{U} \vdash \lambda(x:A).\, x : A \to A}\ \to\text{-INTRO}}{\cdot \vdash \lambda(A:\mathcal{U}).\, \lambda(x:A).\, x : \prod_{(A:\mathcal{U})} A \to A}\ \Pi\text{-INTRO}$$

La derivación de su buen comportamiento es similar al Ejemplo 2.4.8, y será omitida.

Dados estos últimos ejemplos, vemos que dada una función lambda bien tipada, es un proceso simple (pero tedioso) generar la derivación de su existencia y buen comportamiento. Por este motivo, en las secciones siguientes comenzaremos a escribir solamente la función lambda, omitiendo las derivaciones asociadas.

Notación 2.5.10. Es común en MC definir funciones a partir de reglas de correspondencia; es decir, definiendo f por su comportamiento en un x arbitrario. Utilizaremos esta misma práctica, entendiendo estas definiciones como una abreviación de funciones lambda. Por ejemplo, la definición de la función

$$\mathsf{id} : \prod_{A:\mathcal{U}} A \to A$$
$$\mathsf{id} :\equiv \lambda(A:\mathcal{U}).\, \lambda(x:A).\, x$$

Puede ser reescrita simplemente como

$$\mathsf{id} : \prod_{A:\mathcal{U}} A \to A$$
$$\mathsf{id}(A, x) :\equiv x$$

Finalmente, para funciones currificadas de varios parámetros, se asociaran aplicaciones repetidas de '$\to$' por la derecha, por lo que $A \to B \to C$ se entiende como $A \to (B \to C)$, por ejemplo. Todos los demás constructos por introducir asociarán hacia la izquierda, de tal forma que $A \times B \times C$ es $(A \times B) \times C$.

Ejemplo 2.5.11. Existe una función swap que invierte el orden de las variables de una función de dos parámetros.

$$\mathsf{swap} : \prod_{(A:\mathcal{U})} \prod_{(B:\mathcal{U})} \prod_{C:\mathcal{U}} (A \to B \to C) \to (B \to A \to C)$$

Y esta está definida por

$$\mathsf{swap}(A, B, C, f) :\equiv \lambda(b : B).\, \lambda(a : A).\, f(a, b)$$

Notamos que de acuerdo a la notación anterior, esta también pudo haber sido definido como:

$$\mathsf{swap}(A, B, C, f, b, a) :\equiv f(a, b)$$

Como se esperaría, podemos generalizar la definición de composición de funciones del Ejemplo 2.4.10.

Definición 2.5.12. Podemos definir la función de composición de funciones

$$\mathsf{comp}^* :\equiv \prod_{(A:\mathcal{U})} \prod_{(B:\mathcal{U})} \prod_{(C:\mathcal{U})} \left(\prod_{y:B} C(y) \right) \to \prod_{(f:A\to B)} \prod_{(x:X)} C(f(x))$$

dada por

$$\mathsf{comp}^*(A, B, C, g, f, x) :\equiv g(f(x))$$

Esta función generaliza a la función del Ejemplo 2.4.10, en el sentido de que

$$\mathsf{comp}_{A,B,C}(g, f) \equiv \mathsf{comp}^*(A, B, C, g, f)$$

Como es usual, escribiremos $g \circ f$ en vez de $\mathsf{comp}^*(A, B, C, g, f)$, dejando los tipos A, B y C implícitos.

2.6. El tipo de pares dependientes

Así como el tipo de funciones $A \to B$ es un caso particular del tipo de funciones dependientes $\prod_{(x:A)} B(x)$ el tipo de pares $A \times B$ es un caso particular del tipo de pares dependientes $\sum_{(x:A)} B$ donde B puede depender de $x : A$.

Intuitivamente, dados dos elementos $a : A$ y $b : B(a)$ es posible generar el par (a, b). Las siguientes reglas formalizan esto[2].

Reglas 2.6.1. (Reglas de formación de pares dependientes)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : A \to \mathcal{U}_i}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}_i}\ \Sigma\text{-FORM}$$

$$\frac{\Gamma \vdash B : A \to \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B(a)}{\Gamma \vdash (a, b) : \sum_{(x:A)} B}\ \Sigma\text{-INTRO}$$

[2]El uso del tipo de funciones en estas reglas podría hacer parecer que estas tienen un rol más fundamental que el de pares dependientes; sin embargo, es posible formalizar estas reglas sin hacer mención a las funciones, ver [25, Apéndice A.2.]

donde la expresión $\sum_{(x:A)} B$ liga a x hasta el final de esta.

Notación 2.6.2. En caso B sea una familia constante, escribiremos también $A \times B$ en vez de $\sum_{(x:A)} B$. Los variables ligadas por Σ tienen menor precedencia que otros operadores dentro de un tipo, por lo que $\sum_{(x:A)} B \to C$ se entiende como $\sum_{(x:A)}(B \to C)$, por ejemplo.

En el caso del producto '$\times$', este tiene mayor precedencia que '$\to$', por lo que $A \times B \to C$ se interpreta como $(A \times B) \to C$. Lo mismo aplica para el coproducto '$+$' definido en la próxima sección.

Además, para acentuar el énfasis de la (posible) dependencia de B sobre x, escribiremos $\sum_{(x:A)} B(x)$, entendiendo por esto como simplemente $\sum_{(x:A)} B$. Finalmente, mencionamos que existe otra notación alternativa para $\sum_{(x:A)} B(x)$, como

$$\sum_{x:A} B(x) \quad \text{y} \quad \Sigma(x:A), B(x).$$

Similarmente al caso de funciones dependientes, el uso del símbolo Σ sugiere que este tipo puede ser interpretado también como una suma (disjunta) de los B_i, como lo muestra el próximo ejemplo informal.

Ejemplo 2.6.3. Sea $A = \{a_1, a_2, a_3\}$ y sean X, Y, Z, B como en el Ejemplo 2.5.8. El tipo de pares dependientes $\sum_{(x:A)} B$ contiene pares (a, b) con $a : A$ y $b : B(a)$. Por ejemplo, podemos definir un par $(a_1, x_i) : \sum_{(x:A)} B$.

De esta forma, los elementos de $\sum_{(x:A)} B$ corresponden exactamente a elementos de la unión disjunta de los elementos de X, Y y Z. En efecto, tenemos

$$\left|\sum_{(x:A)} B\right| = \Sigma_{x:A}|B(x)| = |X| + |Y| + |Z|$$

Las reglas previas solo muestran cómo formar pares, para que estos sean útiles es necesario ver cómo se pueden usar; es decir, cómo formar funciones cuyo dominio sea un tipo de pares dependientes.

Reglas 2.6.4. (Reglas de eliminación de pares dependientes.)

$$\frac{\Gamma \vdash C : (\sum_{(x:A)} B) \to \mathcal{U}_i \qquad \Gamma \vdash g : \prod_{(a:A)} \prod_{(b:B(a))} C((a,b)) \qquad \Gamma \vdash p : \sum_{(x:A)} B}{\Gamma \vdash \mathsf{ind}^{C,g,p}_{\sum_{(x:A)} B} : C(p)} \;\Sigma\text{-ELIM}$$

$$\frac{\Gamma \vdash C : (\sum_{(x:A)} B) \to \mathcal{U}_i \qquad \Gamma \vdash g : \prod_{(a:A)} \prod_{(b:B(a))} C(a,b) \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash \mathsf{ind}^{C,g,(a,b)}_{\sum_{(x:A)} B} \equiv g(a)(b) : C((a,b))} \;\Sigma\text{-COMP}$$

Aquí, $\mathsf{ind}^{C,g,p}_{\sum_{(x:A)} B}$ es un *primitivo*, un elemento que afirmamos existe, y es un elemento de $C(p)$. En otras palabras, para cada C, g y p que satisfagan los requisitos de la regla, Σ-ELIM indica que exite este primitivo, el cual depende de estas tres metavariables. Similarmente para la regla Σ-COMP.

Estas dos reglas nos dan el siguiente principio de inducción para pares dependientes.

Teorema 2.6.5. *Sea A un tipo, $B : A \to \mathcal{U}$ una familia de tipos sobre A, entonces existe una función*

$$\mathsf{ind}_{\sum_{(x:A)} B(x)} : \prod_{(C:(\sum_{(x:A)} B(x))\to\mathcal{U})} \left(\prod_{(a:A)}\prod_{(b:B(a))} C((a,b))\right) \to \prod_{(p:\sum_{(x:A)} B(x))} C(p)$$

tal que

$$\mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g, (a,b)) \equiv g(a)(b) : C((a,b))$$

Demostración. Definimos la función por

$$\mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g, p) :\equiv \mathsf{ind}^{C,g,p}_{\sum_{(x:A)} B} : C(p),$$

el cual existe por la regla Σ-ELIM, mientras que la regla Σ-COMP nos indica que se comporta apropiadamente en pares. □

El principio de inducción para pares dependientes captura la adjunción Hom (Ejemplo 1.20):

$$\mathrm{Hom}(A \times B, C) \cong \mathrm{Hom}(A, \mathrm{Hom}(B, C))$$

En efecto, es esta equivalencia la que aprovechamos para nuestra definición de funciones currificadas.

Veamos como podemos utilizar este principio para mostrar la existencia de las funciones de proyección en cada uno de las coordenadas.

Teorema 2.6.6. *Sea A un tipo y B una familia de tipos sobre A. Entonces existen funciones pr_1 y pr_2 tales que $\mathsf{pr}_1(a,b) = a$ y $\mathsf{pr}_2(a,b) = b$.*

Demostración. Sean

$$C :\equiv \lambda p.\, A : \sum_{x:A} B(x) \to \mathcal{U}$$
$$g :\equiv \lambda x.\, \lambda y.\, x : \prod_{(a:A)} \prod_{(b:B(a))} A$$

Notando que $C(a,b) \equiv A$, y aplicando el principio de inducción obtenemos

$$\mathsf{pr}_1 :\equiv \mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g) : \prod_{p:\sum_{(x:A)} B(x)} A$$
$$\mathsf{pr}_1(a,b) :\equiv \mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g, (a,b)) \equiv g(a)(b) \equiv a$$

De manera similar, tomando

$$C' :\equiv \lambda p.\, B(\mathsf{pr}_1(p)) : \sum_{x:A} B(x) \to \mathcal{U}$$
$$g :\equiv \lambda x.\, \lambda y.\, y : \prod_{(a:A)} \prod_{(b:B(a))} B(a)$$

y notando que $C(a,b) \equiv B(\mathsf{pr}_1((a,b))) \equiv B(a)$, aplicando el principio de inducción obtenemos

$$\mathsf{pr}_2 :\equiv \mathsf{ind}_{\sum_{(x:A)} B(x)}(C', g') : \prod_{p:\sum_{(x:A)} B(x)} B(\mathsf{pr}_1(p))$$
$$\mathsf{pr}_2 :\equiv \mathsf{ind}_{\sum_{(x:A)} B(x)}(C', g', (a,b)) \equiv g'(a,b) \equiv b$$

Por lo que ambas proyecciones existen. □

Aplicar directamente el principio de inducción requiere cierto nivel de cuidado, en MC clásica definiríamos la función pr_2 simplemente como $\mathsf{pr}_2((a,b)) :\equiv b$. Veremos que esta práctica también se puede realizar en DTT.

Notación 2.6.7. (Búsqueda de patrones para pares dependientes)
Sea A un tipo, $B : A \to \mathcal{U}$ una familia de tipos sobre A y $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}$. Podemos definir una función $f : \prod_{(p:\sum_{(x:A)} B(x))} C(p)$ por

$$f((a,b)) :\equiv \Phi$$

donde $\Phi : C((a,b))$ es una expresión que puede contener a o b.

Justificación. Dados $a : A$, $b : B(a)$ y $\Phi : C((a,b))$, podemos definir

$$g :\equiv \lambda(a:A).\lambda(b:B(a)).\Phi : \prod_{(a:A)} \prod_{(b:B(a))} C((a,b))$$

Entonces,

$$f :\equiv \mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g) : \prod_{p:\sum_{(x:A)} B(x)} C(p)$$

es tal que $f((a,b)) \equiv \Phi$. □

Esto justifica el hecho "obvio" de que basta definir una función en un par arbitrario para que la función esté bien definida en todo el tipo de pares dependientes.

Notación 2.6.8. Omitiremos a veces los paréntesis de un par cuando estos estén dentro de una aplicación de una función o dentro de otro par. Por ejemplo, escribiremos $C(a,b)$ en vez de $C((a,b))$, y (a,b,c) en vez de $(a,(b,c))$ o $((a,b),c)$.

2.7. 0, 1, 2 y el tipo del coproducto

El tipo 0

El tipo $\mathbf{0} : \mathcal{U}$ representa el tipo vacío, por lo que si obtuviésemos un elemento de él, esto sería un absurdo, y podríamos concluir cualquier cosa. Las siguientes reglas capturan esta intuición.

Reglas 2.7.1. (Reglas del tipo **0**.)

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_0}\ \mathbf{0}\text{-FORM} \qquad \frac{\Gamma \vdash C : \mathbf{0} \to \mathcal{U}_0 \qquad \Gamma \vdash z : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}^{C,z} : C(z)}\ \mathbf{0}\text{-ELIM}$$

Aquí, al igual que para el caso de los pares dependientes, $\mathsf{ind}_{\mathbf{0}}^{C,z}$ es un primitivo, el cual afirmamos que existe siempre que se satisfagan las condiciones necesarias. Esto mismo aplica para los elementos $\mathsf{ind}_{\mathbf{1}}^{C,c,a}$, inl_{A+B}^{a} y inr_{A+B}^{b} introducidos posteriormente en esta sección.

El principio de inducción del **0** implica que, dado un $z : \mathbf{0}$, podemos generar un elemento de un tipo que (posiblemente) depende de z.

Teorema 2.7.2. *Existe una función*

$$\mathsf{ind}_{\mathbf{0}} : \prod_{(C:\mathbf{0}\to\mathcal{U})} \prod_{(z:\mathbf{0})} C(z)$$

Demostración. Definimos la función por

$$\mathsf{ind}(C, z) :\equiv \mathsf{ind}_{\mathbf{0}}^{C,z}.$$

□

Nótese que, en particular, el principio de inducción del tipo **0** nos da una función $\mathsf{rec}_{\mathbf{0}}(C) : \mathbf{0} \to C$, para todo tipo C. En efecto, podemos definir

$$\mathsf{rec}_{\mathbf{0}}(C, x) :\equiv \mathsf{ind}_{\mathbf{0}}(\lambda x.\, C, x)$$

Esto nos da indicios de que el tipo **0** es un objeto inicial de la categoría Type, y este es efectivamente el caso; pero la demostración de la unicidad de la función $\mathsf{rec}_{\mathbf{0}}(C)$ tendrá que esperar (ver Proposición 3.7.2).

El tipo 1

De manera análoga, el tipo $\mathbf{1} : \mathcal{U}$ representa el tipo con un solo elemento $\star$, por lo que para generar una función con dominio es suficiente definirla en $\star$. Las siguientes reglas dicen precisamente esto.

Reglas 2.7.3. (Reglas del tipo **1**.)

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\ \mathbf{1}\text{-FORM} \qquad \frac{\Gamma \text{ ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-INTRO}$$

$$\frac{\Gamma \vdash C : \mathbf{1} \to \mathcal{U}_i \qquad \Gamma \vdash c : C(\star) \qquad \Gamma \vdash a : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}^{C,c,a} : C(a)}\ \mathbf{1}\text{-ELIM}$$

$$\frac{\Gamma \vdash C : \mathbf{1} \to \mathcal{U}_i \qquad \Gamma \vdash c : C(\star)}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}^{C,c,\star} \equiv c : C(\star)}\ \mathbf{1}\text{-COMP}$$

El principio de inducción para **1** es el siguiente.

Teorema 2.7.4. *Existe una función*

$$\mathsf{ind}_{\mathbf{1}} : \prod_{C:\mathbf{1}\to\mathcal{U}} C(\star) \to \prod_{x:\mathbf{1}} C(x)$$

tal que

$$\mathsf{ind}_{\mathbf{1}}(C, c, \star) :\equiv c.$$

Demostración. La función definida por

$$\mathsf{ind}_{\mathbf{1}}(C, c, a) :\equiv \mathsf{ind}_{\mathbf{1}}^{C,c,a}$$

cumple los requisitos □

El principio de inducción del **1** nos permite usar la siguiente notación.

Notación 2.7.5. (Búsqueda de patrones para **1**)
Sea $C : \mathbf{1} \to \mathcal{U}$ una familia de tipos sobre **1**. Podemos definir una función $f : \prod_{(x:\mathbf{1})} C(x)$ por

$$f(\star) :\equiv \Phi$$

donde $\Phi : C(\star)$.

Justificación. Dado $\Phi : C(\star)$, podemos definir

$$f :\equiv \mathsf{ind}_{\mathbf{1}}(C, \Phi) : \prod_{x:\mathbf{1}} C(x)$$

y este cumple que $f(\star) \equiv \Phi$. □

Categóricamente, el tipo **1** es un objeto terminal de Type, pues dado un tipo C arbitrario, podemos definir una función $!\mathbf{1}_C : C \to \mathbf{1}$ por

$$!\mathbf{1}_C :\equiv \lambda(x : C).\star$$

Verificaremos que esta función es única en la Proposición 4.2.3.

El tipo 2 y el tipo del coproducto

Podríamos ahora esperar poder definir **2**, el tipo con dos elementos, como la unión disjunta de **1** consigo mismo, a través de una un producto $\mathbf{2} :\equiv X \times \mathbf{1}$, donde X tiene dos elementos. Esto es posible, pero no tenemos ningún X que satisfaga esta propiedad.

Una alternativa es usando la noción del coproducto. Dados dos tipos A y B, su coproducto es denotado por $A + B$. Este puede ser entendido como una unión disjunta de los dos tipos.

Reglas 2.7.6. (Reglas de formación del coproducto.)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A + B : \mathcal{U}_i} \text{ +-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}_{A+B}(a) : A + B} \text{ +-INTRO}_1$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{inr}_{A+B}(b) : A + B} \text{ +-INTRO}_2$$

De esta forma, inl_{A+B} y inr_{A+B} actúan como las inyecciones naturales hacia el coproducto. Así, podemos definir $\mathbf{2} :\equiv \mathbf{1} + \mathbf{1}$, y este posee dos elementos $\mathsf{inl}(\star)$ e $\mathsf{inr}(\star)$.

Sin embargo, para poder utilizar un coproducto, debemos saber cómo definir funciones cuyo dominio sean estos.

Reglas 2.7.7. (Reglas de eliminación del coproducto.)

$$\frac{\begin{array}{c}\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \\ \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash e : A + B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,e} : C(e)} \text{ +-ELIM}$$

$$\frac{\begin{array}{c}\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \\ \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,\mathsf{inl}(a)} \equiv c(a) : C(\mathsf{inl}(a))} \text{ +-COMP}_1$$

$$\frac{\begin{array}{c}\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \\ \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash b : B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,\mathsf{inr}(b)} \equiv d(b) : C(\mathsf{inr}(b))} \text{ +-COMP}_2$$

Estas reglas justifican el hecho de que basta definir una función en A y en B para que esté definida en $A + B$. Esto es el principio de inducción para el coproducto.

Teorema 2.7.8. *Sean A y B tipos. Entonces existe una función*

$$\mathsf{ind}_{A+B} : \prod_{(C:(A+B)\to\mathcal{U})} \left(\prod_{(x:A)} C(\mathsf{inl}(x))\right) \to \left(\prod_{(y:B)} C(\mathsf{inr}(y))\right) \to \prod_{(e:A+B)} C(e)$$

tal que

$$\mathsf{ind}_{A+B}(C, c, d, \mathsf{inl}(a)) \equiv c(a) : C(\mathsf{inl}(a))$$
$$\mathsf{ind}_{A+B}(C, c, d, \mathsf{inr}(b)) \equiv d(b) : C(\mathsf{inl}(b))$$

Demostración. La función definida por

$$\mathsf{ind}_{A+B}(C, c, d, e) :\equiv \mathsf{ind}_{A+B}^{C,c,d,e}$$

satisface los requisitos. □

Como en los tipos previos, podemos simplificar considerablemente la notación realizando una búsqueda de patrones.

Notación 2.7.9. (Búsqueda de patrones para el coproducto)
Sean A, B tipos, y $C : A + B \to \mathcal{U}$ una familia de tipos sobre $A + B$. Podemos definir una función $f : \prod_{(e:A+B)} C(e)$ por

$$f(\mathsf{inl}(a)) :\equiv \Phi_0$$
$$f(\mathsf{inr}(b)) :\equiv \Phi_1$$

donde $\Phi_0 : C(\mathsf{inl}(a))$ y $\Phi_1 : C(\mathsf{inr}(b))$ son expresiones que pueden tener a 'a' o a 'b' respectivamente.

Justificación. Dados $\Phi_0 : C(\mathsf{inl}(a))$ y $\Phi_1 : C(\mathsf{inr}(b))$, podemos definir

$$f :\equiv \mathsf{ind}_{A+B}(C, \lambda(a : A).\, \Phi_0, \lambda(b : B).\, \Phi_1) : \prod_{e:A+B} C(e)$$

y esta satisface $f(\mathsf{inl}(a)) :\equiv \Phi_0$ y $f(\mathsf{inr}(b)) :\equiv \Phi_1$. □

Analizaremos con más detalle el tipo $\mathbf{2} :\equiv \mathbf{1} + \mathbf{1}$. Escribiendo $0_{\mathbf{2}} :\equiv \mathsf{inl}(\star)$ y $1_{\mathbf{2}} :\equiv \mathsf{inr}(\star)$, y por el principio de inducción para el coproducto, vemos que para definir una función $\prod_{(x:\mathbf{2})} C(x)$ es suficiente definir dos funciones $f : \mathbf{1} \to C(0_{\mathbf{2}})$ y $g : \mathbf{1} \to C(1_{\mathbf{2}})$. Pero por el principio de inducción de $\mathbf{1}$, esto es lo mismo que seleccionar dos elementos, uno de $C(0_{\mathbf{2}})$ y otro de $C(1_{\mathbf{2}})$. De esta forma, obtenemos el principio de inducción para $\mathbf{2}$.

Teorema 2.7.10. *Sea $C : \mathbf{2} \to \mathcal{U}$ una familia de tipos sobre $\mathbf{2}$. Entonces existe una función*

$$\mathsf{ind}_{\mathbf{2}} : \prod_{(C:\mathbf{2}\to\mathcal{U})} C(0_{\mathbf{2}}) \to C(1_{\mathbf{2}}) \to \prod_{(x:\mathbf{2})} C(x)$$

tal que

$$\mathsf{ind}_{\mathbf{2}}(C, c_0, c_1, 0_{\mathbf{2}}) \equiv c_0 : C(0_{\mathbf{2}})$$
$$\mathsf{ind}_{\mathbf{2}}(C, c_0, c_1, 1_{\mathbf{2}}) \equiv c_1 : C(1_{\mathbf{2}})$$

Sea $C : \mathbf{2} \to \mathcal{U}$ definido por $C(0_{\mathbf{2}}) :\equiv A$ y $C(1_{\mathbf{2}}) :\equiv B$, es intuitivo que $\sum_{(x:2)} C(x)$ sea equivalente, en cierto sentido, a $A + B$. Este es efectivamente el caso (ver Capítulo 2), y pudimos también haber introducido primero el tipo $\mathbf{2}$ y definir $A + B$ a partir de este. Cuál es introducido primero no causa ninguna diferencia en los resultados.

2.8. El tipo de los naturales

Hemos introducido varios principios de inducción para diferentes tipos. Veamos que para el tipo de los naturales, su principio de inducción en DTT coincide con el principio usal de MC. Intuitivamente, el tipo de los naturales consiste del conjunto $\{0, 1, 2, \dots\}$, pero nótese que, a excepción del 0, todo elemento m puede ser escrito como $n+1$ para algún n. Esto sugiere las siguientes reglas.

Reglas 2.8.1. (Reglas de formación de $\mathbb{N}$.)

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-FORM}$$

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1 \qquad \frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathsf{succ} : \mathbb{N} \to \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

Ahora, para formar funciones que tengan como dominio el tipo $\mathbb{N}$, necesitamos introducir las reglas de eliminación.

Reglas 2.8.2. (Reglas de eliminación de $\mathbb{N}$.)

$$\frac{\begin{array}{c}\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \\ \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n)) \qquad \Gamma \vdash n : \mathbb{N}\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,n} : C(n)}\ \mathbb{N}\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \\ \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,0} \equiv c_0 : C(n)}\ \mathbb{N}\text{-COMP}_1$$

$$\frac{\begin{array}{c}\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \\ \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n)) \qquad \Gamma \vdash n : \mathbb{N}\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,\mathsf{succ}(n)} \equiv c_s(n, \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,n}) : C(\mathsf{succ}(n))}\ \mathbb{N}\text{-COMP}_2$$

Aquí, nuevamente, $\mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,\mathsf{succ}(n)}$ es un primitivo que pertenece a $C(n)$. Con estas reglas, obtenemos el principio de inducción usual.

Teorema 2.8.3. *Existe una función*

$$\mathsf{ind}_{\mathbb{N}} : \prod_{(C:\mathbb{N}\to\mathcal{U})} C(0) \to \left(\prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))\right) \to \prod_{(n:\mathbb{N})} C(n)$$

tal que

$$\begin{aligned}\mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, 0) &:\equiv c_0, \\ \mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, \mathsf{succ}(n)) &:\equiv c_s(n, \mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, n)).\end{aligned}$$

Demostración. La función definida por

$$\mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, n) :\equiv \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,n}$$

cumple los requisitos. □

Notación 2.8.4. (Búsqueda de patrones para los naturales)
Podemos definir una función con dominio $\mathbb{N}$ a una familia de tipos $C : \mathbb{N} \to \mathcal{U}$ definiendo su comportamiento en 0 y $\mathsf{succ}(n)$, es decir

$$f(0) :\equiv \Phi_0 : C(0)$$
$$f(\mathsf{succ}(n)) :\equiv \Phi_1 : C(\mathsf{succ}(n))$$

donde n y $f(n)$ pueden aparecer en Φ_1.

Justificación. f puede ser definida como

$$f :\equiv \mathsf{ind}_{\mathbb{N}}(C, \Phi_0, \lambda n.\, \lambda r.\, \Phi_1^*)$$

donde Φ_1^*, es Φ con todas las ocurrencias de $f(n)$ reemplazadas por r. □

Definimos los símbolos usuales para los números naturales como aplicaciones repetidas del constructo succ. Es decir $1 :\equiv \mathsf{succ}(0)$, $2 :\equiv \mathsf{succ}(1)$, $3 :\equiv \mathsf{succ}(2), \ldots$.

En el caso en el que la familia $C : \mathbb{N} \to \mathcal{U}$ es constante, el principio de inducción se convierte en el principio de recursión.

Teorema 2.8.5. *Existe una función*

$$\mathsf{rec}_{\mathbb{N}} : \prod_{C:\mathcal{U}} \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to C)$$

Demostración. Podemos definir $\mathsf{rec}_{\mathbb{N}}$ por

$$\mathsf{rec}_{\mathbb{N}}(C, c, f, 0) :\equiv c : C$$
$$\mathsf{rec}_{\mathbb{N}}(C, c, f, \mathsf{succ}(n)) :\equiv f(\mathsf{rec}_{\mathbb{N}}(C, c, f, n)) : C$$

□

Es más, para todos los tipos existe un principio de recursión, el cual es simplemente el principio de inducción aplicado a una familia constante. Veamos dos aplicaciones de esto.

Ejemplo 2.8.6. Existe una función $\mathsf{double} : \mathbb{N} \to \mathbb{N}$ tal que duplica el valor su input.

$$\mathsf{double} : \mathbb{N} \to \mathbb{N}$$
$$\mathsf{double}(0) :\equiv 0$$
$$\mathsf{double}(\mathsf{succ}(n)) :\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{double}(n)))$$

Por ejemplo,

$$\begin{aligned}\mathsf{double}(2) &\equiv \mathsf{double}(\mathsf{succ}(\mathsf{succ}(0)))\\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{double}(\mathsf{succ}(0))))\\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{double}(0)))))\\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(0))))\\ &\equiv 4\end{aligned}$$

La función suma también es definida por recursión

Ejemplo 2.8.7.

$$\begin{aligned}\mathsf{add} &: \mathbb{N} \to \mathbb{N} \to \mathbb{N}\\ \mathsf{add}(0) &:\equiv \mathsf{id}_{\mathbb{N}}\\ \mathsf{add}(\mathsf{succ}(n)) &:\equiv \lambda m.\, \mathsf{succ}(\mathsf{add}(n)(m))\end{aligned}$$

A forma de recordatorio que la notación previa es solo una simplificación, mostramos la definición usando solo el principio de inducción:

$$\mathsf{add} :\equiv \mathsf{ind}_{\mathbb{N}}(\lambda n.\, (\mathbb{N} \to \mathbb{N}), \mathsf{id}_{\mathbb{N}}, \lambda n.\, \lambda g.\, \lambda m.\, \mathsf{succ}(g(m)))$$

Como es común, utilizaremos $a + b$ para referirnos a $\mathsf{add}(a)(b)$.

2.9. Proposiciones como tipos

En la introducción habíamos mencionado que las proposiciones eran representadas como tipos en DTT. Comenzaremos analizando cada uno de los constructos lógicos principales en MC.

- Para mostrar que se cumple $P \wedge Q$, es suficiente mostrar que se cumplen ambos P y Q. Podemos entonces considerar una prueba de $P \wedge Q$ como un par de pruebas de P y Q.
- Para mostrar que se cumple $P \vee Q$, es suficiente mostrar que se cumple alguno de P o Q. Podemos entonces considerar una prueba de $P \vee Q$ como la unión disjunta de pruebas de P y Q.
- Para mostrar que se cumple $P \implies Q$, es suficiente mostrar que dado P, se puede mostrar que se cumple Q. Podemos entonces considerar una prueba de $P \implies Q$ como una función que toma una prueba de P y devuelve una prueba de Q.
- Para mostrar que se cumple $\neg P$, es suficiente mostrar que si es que tenemos P, podemos llegar a una contradicción. Podemos entonces considerar una prueba de $\neg P$ como una función que toma una prueba de P y devuelve una prueba de una contradicción.
- Para mostrar que se cumple $\forall x \in A, P(x)$, es suficiente mostrar que, dado un x arbitrario en A, se puede mostrar que este satisface $P(x)$. Podemos entonces considerar una prueba de $\forall x \in A, P(x)$ como una función que toma cualquier elemento $x \in A$ y devuelve una prueba de $P(x)$.

- Para mostrar que se cumple $\exists x \in A, P(x)$, es suficiente mostrar que existe un $a \in A$ tal que $P(a)$. Podemos entonces considerar una prueba de $\exists x \in A, P(x)$ como un par, el cual consiste de un elemento $a \in A$ y una prueba de $P(a)$.

La interpretación de estos constructos de esta manera es llamada la **interpretación BHK**, debido a los nombres de sus principales proponentes L. E. J. Brouwer, Arend Heyting, y Andrey Kolmogorov. Notemos su similitud con las reglas de introducción para los tipos que ya hemos introducido.

Por ejemplo, la regla de introducción para el tipo del producto indica que para construir un elemento de $P \times Q$ es suficiente mostrar dos elementos $a : P$ y $b : Q$, y el nuevo elemento construido sería el par (a, b) de estos. Así, los tipos se interpretan como proposiciones y los elementos de estos tipos se interpretan como pruebas o evidencia de estas proposiciones. Esta correspondencia entre lógica y tipos es llamada el **isomorfismo de Curry-Howard**.

Matemática Clásica	Teoría de Tipos dependientes
$A \wedge B$	$A \times B$
$A \vee B$	$A + B$
$A \implies B$	$A \to B$
$A \iff B$	$(A \to B) \times (B \to A)$
$\neg A$	$A \to \mathbf{0}$
$\forall x \in A, P(x)$	$\prod_{(x:A)} P(x)$
$\exists x \in A, P(x)$	$\sum_{(x:A)} P(x)$

Cuadro 2.2: Isomorfismo de Curry-Howard.

La correspondencia es relativamente simple por lo que no la explicaremos más a detalle, solo el caso de $\neg A$ merece una mayor atención. En DTT el $\mathbf{0}$ toma el rol de un absurdo, puesto que su mismo principio de inducción indica que es posible formar una función con cualquier codominio dado un elemento de $\mathbf{0}$. Por este motivo, interpretamos $A \to \mathbf{0}$ como $\neg A$.

Definición 2.9.1. Diremos que dos tipos A y B son lógicamente equivalentes cuando se tiene $A \iff B$; es decir, tenemos dos funciones $f : A \to B$ y $g : B \to A$, como indicado en el Cuadro 2.2.

Veamos algunos ejemplos de lógica proposicional y lógica predicativa.

Ejemplo 2.9.2. Una de las leyes de Morgan indica que para todo par de proposiciones A y B tenemos

$$\neg A \vee \neg B \implies \neg(A \wedge B).$$

El equivalente en DTT es la existencia de una función

$$f : \prod_{(A:\mathcal{U})} \prod_{(B:\mathcal{U})} (A \to \mathbf{0}) + (B \to \mathbf{0}) \to (A \times B \to \mathbf{0})$$

En efecto, combinando la búsqueda de patrones en el coproducto y en el producto, podemos definir f como

$$f(A, B, \mathsf{inl}(f_1), (a, b)) :\equiv f_1(a)$$
$$f(A, B, \mathsf{inr}(f_2), (a, b)) :\equiv f_2(b)$$

Notación 2.9.3. A veces omitiremos la mención explícita de los Π tipos involucrados, cuando estos toman el rol lógico de "para todo", entendiendo que el término que estamos construyendo es una función dependiente con un dominio apropiado.

Ejemplo 2.9.4. Consideremos la siguiente implicación usada comúnmente

$$\exists x \in A, \neg P(x) \implies \neg(\forall x \in A, P(x))$$

Esta es representada por la existencia de una función

$$f : \textstyle\sum_{(x:A)}(P(x) \to \mathbf{0}) \to (\prod_{(x:A)} P(x)) \to \mathbf{0}$$

Esta puede ser definida por

$$f((a, g), h) :\equiv g(h(a))$$

Ejemplo 2.9.5. El axioma de elección en MC indica que dada una colección de conjuntos no vacíos X, es posible crear una función $f : X \to \bigcup X$ que selecciona un elemento de cada conjunto. Es decir,

$$\left(\forall x \in X, \exists y \in \bigcup x, (y \in x)\right) \implies \left(\exists f : X \to \bigcup X, \forall x \in X, (f(x) \in x)\right)$$

En DTT, esto podría ser entendido[3] como un caso particular de

$$\mathsf{ac} : \left(\textstyle\prod_{(x:A)}\sum_{(y:B)} R(x, y)\right) \to \left(\sum_{(f:A\to B)}\prod_{(x:A)} R(x, f(x))\right)$$

Podemos definir esta función como

$$\mathsf{ac}(h) :\equiv \left(\lambda x.\, \mathsf{pr}_1(h(x)), \lambda x.\, \mathsf{pr}_2(h(x))\right)$$

Esta función está bien definida puesto que

$$\lambda x.\, \mathsf{pr}_1(g(x)) : A \to B,$$
$$\lambda x.\, \mathsf{pr}_2(g(x)) : \textstyle\prod_{(x:A)} R(x, \mathsf{pr}_1(g(x))).$$

como es requerido por el tipo del codomino de ac.

Definición 2.9.6. Dado un tipo A, si existe un elemento de él tal que $a : A$, diremos que A es un tipo **habitado**.

Así, se dice una proposición P es **verdadera** cuando es un tipo habitado. El ejemplo previo muestra que el axioma de elección es verdadero en esta teoría.

[3]La verdadera formalización del axioma de elección como visto de manera clásica, es ligeramente distinta a la presentación actual, ver [25, Sección 3.8].

2.10. El tipo de identidades

De acuerdo a la interpretación de proposiciones como tipos, debería haber un tipo que refleje la propiedad de que dos elementos sean iguales. Es decir, si tenemos $x, y : A$, tenemos un tipo de identidades entre x y y, el cual denotamos por $x =_A y$.

Reglas 2.10.1. (Reglas de formación del tipo de identidades.)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i} \text{ =-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a} \text{ =-INTRO}$$

Intuitivamente, el tipo $x =_A y$ representa la noción de igualdad, y este está habitado justamente cuando x es igual a y. En particular, $x =_A x$ siempre está habitado, puesto que por =-INTRO tenemos una función

$$\mathsf{refl} : \prod_{x:A} x =_A x,$$

definida por $\mathsf{refl}(x) :\equiv \mathsf{refl}_x$. Además, notemos que es imposible formar el tipo de igualdades para dos elementos en dos tipos distintos.

La introducción de este tipo permite enunciar y probar teoremas que involucren la igualdad.

Ejemplo 2.10.2. Por definición, para todo $n : \mathbb{N}$ tenemos $0 + n :\equiv n$, por lo que podemos definir

$$\lambda n.\, \mathsf{refl}_n : \prod_{n:\mathbb{N}} (0 + n =_{\mathbb{N}} n)$$

Por otro lado, la demostración de $\prod_{(n:\mathbb{N})}(n + 0 =_{\mathbb{N}} n)$ requiere de otros resultados, como el principio de inducción para el tipo de identidades.

De manera análoga a los casos de la suma dependiente y el coproducto, dado que la forma canónica de introducir un elemento del tipo de identidades es por refl_x, basta definir una función dependiente en estos elementos para que esté definida en todo $x =_A y$. Este razonamiento nos da:

Reglas 2.10.3. (Reglas de eliminación del tipo de identidades.)

$$\frac{\begin{array}{c}\Gamma \vdash C : \prod_{(x,y:A)}(x =_A y) \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(z:A)} C(z, z, \mathsf{refl}_z) \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : a =_A b\end{array}}{\Gamma \vdash \mathsf{ind}_{=_A}^{C,c,a,b,p} : C(a, b, p)} \text{ =-ELIM}$$

$$\frac{\begin{array}{c}\Gamma \vdash C : \prod_{(x,y:A)}(x =_A y) \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(z:A)} C(z, z, \mathsf{refl}_z) \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{=_A}^{C,c,a,a,\mathsf{refl}_a} : C(a, a, \mathsf{refl}_a)} \text{ =-COMP}$$

Como en los tipos previos, $\mathsf{ind}_{=_A}^{C,c,a,a,p}$ es un primitivo que existe bajo ciertas condiciones, y estas reglas nos dan un principio de introducción y una búsqueda de patrones.

Teorema 2.10.4. *Para todo tipo $A : \mathcal{U}$, existe una función*

$$\mathsf{ind}_{=_A} : \prod_{(C:\prod_{(x,y:A)}(x=_A y)\to\mathcal{U})} \left(\prod_{(x:A)} C(x,x,\mathsf{refl}_x)\right) \to \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x,y,p)$$

tal que

$$\mathsf{ind}_{=_A}(C,c,x,x,\mathsf{refl}_x) :\equiv c(x).$$

Demostración. Podemos definir $\mathsf{ind}_{=_A}$ por

$$\mathsf{ind}_{=_A}(C,c,x,x,p) :\equiv \mathsf{ind}_{=_A}^{C,c,a,a,p}.$$

□

Notación 2.10.5. (Búsqueda de patrones para el tipo de identidades)
Podemos definir una función $f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x,y,p)$ definiendo su comportamiento en refl_x, es decir

$$f(\mathsf{refl}_x) :\equiv \Phi : C(x,x,\mathsf{refl}_x)$$

Justificación. La función f puede ser definida como

$$f :\equiv \mathsf{ind}_{=_A}(C, \lambda x.\, \Phi)$$

□

Así, para definir una función $f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x,y,p)$ basta asumir que x es y, y definir el comportamiento de la función en los caminos refl_x. Utilizaremos esto múltiples veces en los siguientes capítulos.

El tipo de identidades es quizás el que mayor riqueza trae a esta teoría, y el siguiente capítulo estará dedicado principalmente a este.

Notación 2.10.6. Escribiremos $x = y$ en vez de $x =_A y$ cuando no haya riesgo de confusión.

Para diferenciar el tipo $x =_A y$ del juicio $x \equiv y$, a veces se llama al primero de estos como una **igualdad proposicional**.

2.11. Grupos

Veamos como los tipos introducidos nos permiten formalizar en DTT el concepto de un grupo. Recordemos que en MC un grupo es par $(G, *)$, tal que G es un conjunto y $*$ es una operación binaria en G tal que

- $*$ es asociativa,

- existe un elemento neutro e, tal que para todo $g \in G$ se tiene que $e * g = g * e = g$, y
- para todo $g \in G$ existe un $g^{-1} \in G$ tal que $g * g^{-1} = g^{-1} * g = e$.

Esta estructura puede ser reflejada directamente en la siguiente definición

Definición 2.11.1. Dado un tipo A, diremos que este tiene una **estructura de grupo** si es que el tipo

$$\begin{aligned}\mathsf{GroupStr}(A) :\equiv \sum_{m:A\to A\to A} \prod_{x,y,z:A} (m(x,m(y,z)) = m(m(x,y),z)) \\ \times \sum_{e:A} (e*g=g)\times(g*e=g) \\ \times \sum_{i:A\to A} (g*i(g)=e)\times(i(g)*g=e)\end{aligned}$$

está habitado.

Cada línea de la definición previa corresponde a uno de los puntos de la definición clásica. Con el concepto de estructura ya definido, podemos definir lo que es un grupo.

Definición 2.11.2. Un **grupo** es un tipo junto con una estructura de grupo. Es decir, un elemento del tipo

$$\mathsf{Group} :\equiv \sum_{A:\mathcal{U}} \mathsf{GroupStr}(A)$$

Nótese que, a diferencia de MC, sí tenemos un tipo de todos los grupos, el cual es Group.

Es claro que una formalización similar aplica para la gran mayoría de teorías algebraicas, como las de los anillos, módulos, categorías, etc.

2.12. Metateoría de DTT

Hemos introducido las reglas de DTT que usaremos en lo que sigue del presente documento. En la presente sección discutiremos sobre las reglas mismas, y sobre las propiedades de la teoría que estas generan; es decir, la *metateoría*.

En primer lugar, parecería que existe cierta inconsistencia en algunas reglas. Por ejemplo, consideremos la siguiente regla ya introducida

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}$$

Se tiene $\Gamma \vdash a : A$ en la hipótesis; sin embargo, no estamos afirmando que A sea un tipo, por lo que podría ser posible que $a : A$ sea una expresión sin sentido.

Esta inquietud, aunque válida, no está justificada, pues tenemos el siguiente resultado.

Teorema 2.12.1. *Si se tiene* $\Gamma \vdash a : A$ *para algún* Γ*, también se tiene que* $\Gamma \vdash A : \mathcal{U}_i$.

Nótese que a diferencia de los resultados y ejemplos previos, este resultado es un resultado sobre la teoría misma. Los teoremas de incompletitud de Gödel [11] están en este mismo nivel.

Demostración. La técnica a través la cual se muestran este tipo de teoremas en DTT es a través de *inducción estructural*. Recordemos que una derivación de un juicio es en realidad un árbol invertido con el juicio por derivar en la raíz del árbol. Por lo tanto, podemos utilizar inducción en el árbol, de acuerdo a la última regla usada para llegar a $\Gamma \vdash a : A$.

Por ejemplo, si la última regla usada fue $\mathbb{N}$-INTRO$_1$, tenemos que $\Gamma \vdash 0 : \mathbb{N}$, y por $\mathbb{N}$-FORM sabemos que $\mathbb{N}$ es un tipo. Un razonamiento análogo por casos para cada regla relevante concluye la demostración. □

Otros dos resultados importantes, que ya hemos utilizado libremente en los ejemplos son:

Teorema 2.12.2. *(Substitución) Si se tiene* $\Gamma \vdash a : A$ *y* $\Gamma, x : A, \Delta \vdash b : B$*, entonces es posible derivar* $\Gamma, \Delta[a/x] \vdash b[a/x] : B[a/x]$.

Teorema 2.12.3. *(Debilitación) Si se tiene* $\Gamma \vdash A : \mathcal{U}_i$ *y* $\Gamma, \Delta \vdash b : B$*, entonces es posible derivar* $\Gamma, x : A, \Delta \vdash b : B$.

La demostración de estos teoremas para teorías similares se puede encontrar en [19], por ejemplo.

El último metateorema que presentaremos sobre DTT es que esta teoría es consistente, es decir no se puede derivar una contradicción a partir de las reglas ya introducidas.

Teorema 2.12.4. *(Consistencia de DTT) No es posible derivar una contradicción en DTT, es decir* $\cdot \vdash x : \mathbf{0}$ *para algún* x*; asumiendo que la teoría usual de MC también es consistente.*

Omitimos también la demostración de este resultado, que se puede encontrar en [17].

2.13. Comentarios adicionales

La complejidad de DTT

La introducción de las reglas y la derivación de juicios podría hacer parecer que DTT es una teoría mucho más complicada que la lógica clásica; sin embargo, este no es el caso. La lógica clásica en realidad también está formalizada de una forma similar, por ejemplo se tiene la siguiente regla para la introducción de la conjunción:

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi} \wedge\text{-INTRO}$$

Así existen reglas para cada constructo lógico ($\forall, \exists, \wedge, \vee, \neg, \implies$), y resultados análogos a los de la sección previa [5], los cuales son asumidos implicitamente por los matemáticos, razón por lo que omitimos las para DTT.

La diferencia es entonces que las reglas de la lógica clásica son usualmente introducidas de manera informal, mientras que hemos decidido presentar las reglas de DTT formalmente en este trabajo. Diversas presentaciones de DTT introducen los constructos de forma informal, como [18] y [10].

Constructivismo

La lógica subyacente en DTT es una lógica **constructiva**, también llamada **lógica intuicionista**. Por ejemplo, en esta no es posible demostrar $P \vee \neg P$, la ley del medio excluido. Es posible añadir esta ley como una regla adicional[4], pero esto no será necesario en el desarrollo actual.

Ante cierta inspección, es fácil notar que esta ley es en cierto sentido problemática. Esta permite afirmar la existencia de objetos que no han sido construidos explícitamente, o simplificar problemas de una forma no obvia. Por ejemplo, consideremos la siguiente proposición:

Proposición 2.13.1. *Existe un número natural n, tal que $n = 0$ si y solo si la hipótesis de Riemann es verdadera.*

Demostración. Si la hipótesis de Riemann es verdadera, entonces n es 0. Si la hipótesis de Riemann es falsa, entonces podemos poner $n = 1$. □

Este número n refleja la veracidad de la hipótesis de Riemann de una forma no obvia. Por otro lado, si la prueba fuese constructiva, habríamos generado un n específico, y podríamos verificar fácilmente si este es o no diferente de 0.

No asumir la ley del medio excluido significa que todos los elementos de los cuales tenemos información has sido construidos paso a paso, por lo que además de su existencia, podemos indagar sobre otras propiedades relevantes que pueden tener.

Otro principio relacionado es el de reducción al absurdo, $\neg\neg P \implies P$. Este es en realidad equivalente a la ley del medio excluido, y permite probar la existencia de elementos sin saber cuáles son. Por ejemplo:

Proposición 2.13.2. *En la expansión decimal de π, existen dos dígitos que se repiten infinitas veces.*

Demostración. Si no fuese así, se tendría que un solo dígito se repite infinitas veces, o que ningún dígito se repite infinitas veces. En ambos casos, concluimos que π es un racional, lo cual es un absurdo. □

[4]Esto es posible en DTT, pero una vez introducido el axioma de univalencia esto se vuelve inconsistente. En cambio, se puede introducir un axioma similar, que solo involucra proposiciones simples (ver Sección 4.2.10).

Sin embargo, en esta prueba no hemos hallado cuáles son estos dos dígitos; es decir, la prueba no es constructiva.

Esto no quiere decir que no podamos utilizar nunca el razonamiento de $P \vee \neg P$, solo que para utilizarlo es necesario primeramente demostrar que esto se cumple para el tipo P en cuestión. Realizaremos esto para las igualdades en los naturales en la Sección 3.9.

Asistentes de pruebas

Incluso grandes matemáticos como Leibniz, Gauss, Andrew Wiles, entre otros, han cometido graves errores en sus demostraciones, y ellos no fueron los primeros ni serán los últimos. A fin de evitar esto, existen los llamados **asistentes de pruebas**, programas que verifican que una demostración es correcta. La mayoría de estos usan DTT en vez de teoría de conjuntos, consideremos el siguiente ejemplo tomado de [1] para ver por qué:

Proposición 2.13.3. *Sean U y V espacios vectoriales y $f : U \to V$ una función lineal. Entonces $f(2x + y) = 2f(x) + f(y)$.*

La proposición es fácilmente entendida por un lector que conozca estos conceptos, y pareciese que se ha presentado con precisión los variables necesarias, pero veamos la gran cantidad de información omitida:

- Se ha entendido que existe un campo K subyacente a U y a V.
- Se ha entendido que f es en realidad una función entre los conjuntos subyacentes $f : |U| \to |V|$, recordemos que un espacio vectorial U es un triple $(|U|, \cdot, +)$.
- Se ha entendido que x y y son elementos arbitrarios de $|U|$.
- Se ha entendido que el '$+$' en la izquierda de la ecuación es la suma asociada a U, mientras que el '$+$' en la derecha es el asociado a V.
- Finalmente, se ha entendido que 2 es $1 + 1$, donde 1 es el neutro de la multiplicación del campo K.

Esta gran cantidad de información no puede ser inferida correctamente por computadoras que formalicen la teoría de conjuntos. Uno de los principales problemas es que la estructura que tiene cierto constructo, como los espacios vectoriales, no es reflejada de una manera única en teoría de conjuntos. Por poner otro ejemplo, dada la construcción de los reales como cortes de Dedekind, no solo tiene sentido la proposición '$0_{\mathbb{Q}} \in 1_{\mathbb{R}}$'; peor aún, es verdadera.

En contraste, en DTT, los elementos de un tipo pertenecen a un único tipo (a excepción de los tipos mismos), y la estructura adicional es parte esencial y única de una definición (ver 2.11.2, por ejemplo). Estos hechos permiten la inferencia del significado de variables no completamente especificadas, en la gran mayoría de los casos. Esto hace que DTT sea una teoría más apropiada para la formalización de las matemáticas, a través de programas de computadoras.

Ya existe un gran esfuerzo a nivel global de traducir la matemática clásica al lenguaje de DTT, usado por los asistentes de pruebas, y así permitir la

verificación de estos resultados por computadora. Entre estos, mencionamos las formalizaciones [16] y [24], proyectos con más de 250 contribuidores en conjunto.

Gran parte de resultados que consideramos básicos, y que se enseñan a nivel de pregrado ya están formalizados. Sin embargo, también se han verificado algunos resultados más modernos y complicados. El teorema de Feit-Thompson [9], que indica que todo grupo de orden finito e impar es soluble, fue verificado usando el programa Coq en el 2013 [12].

Otro ejemplo es la verificación de un resultado avanzado de geometría algebraica de Peter Scholze, ganador del premio Fields 2018, en colaboración con Dustin Clausen. El proceso de formalización comenzó como un reto de Scholze a la comunidad de asistentes de pruebas [23]:

> Considero que este teorema es de una gran importancia fundacional, por lo que estar 99.9 % seguro no es suficiente… Pasé mucho del 2019 obsesionado con la prueba de este teorema, casi volviéndome loco sobre esta. Al final logramos escribir el argumento en un artículo, pero creo que nadie más se ha atrevido a ver los detalles de este, así que todavía tengo algunas pequeñas dudas.

Solo 6 meses después, con la ayuda de varios matemáticos y científicos de la computación, Scholze escribe que su reto era prácticamente un éxito: aunque todavía no se había demostrado el teorema, todos los lemas que causaban cierta duda ya estaban formalizados [22]. A través de este ejercicio, él comenta, no solo encontró múltiples errores (que afortunadamente fueron posibles de solucionar), también profundizó el entendimiento de su propia prueba.

Esta es la propuesta de los proponentes de los asistentes de pruebas: introducir a las computadoras como una herramienta más en el arsenal que posee un matemático, y obtener una mejor comprensión del tema de estudio, así como la posibilidad de no volver a equivocarnos nunca más[5].

El presente trabajo advoca esta posición, por lo que todos los resultados aquí descritos has sido formalizados en un asistente de pruebas, llamado Agda. El código se encuentra disponible en línea, a través de la siguiente página web interactiva https://shiranaiyo.github.io/MastersThesis/.

Enfatizamos que la posibilidad de la formalización en un asistente de pruebas es solo una ventaja más de DTT. Como mencionamos, la ventaja principal es el tratamiento uniforme de proposiciones y tipos, la cual será aprovechada en los siguientes capítulos.

[5]Podría preguntarse uno, ¿qué garantiza que el programa no cometa un error? La respuesta es: matemática. Se puede razonar de una forma similar a la de inducción estructural, y verificar que cada paso del programa es correcto, y por lo tanto, el algoritmo entero.

Capítulo 3

La Interpretación Homotópica

3.1. Homotopía y caminos

Para facilitar la siguiente discusión, utilizaremos la notación y nomenclatura de MC. Veremos que los tipos se comportan como espacios topológicos, y, por lo tanto, también se pueden interpretar como ∞-grupoides. Para esto, recordaremos unas nociones previas.

Dado un espacio topológico X, y dos puntos $x, y \in X$, podemos generar el conjunto de caminos de x a y. Sin embargo, este conjunto es muy fino, generalmente no es importante el camino exacto tomado p, solo su clase de homotopía $[p]$.

Esto es, la clase de equivalencia bajo la relación $p \sim q$ si y solo si existe una homotopía entre p y q rel ∂I; es decir, que existe una función continua $H : I \times I \to X$ tal que

$$H(0,t) = x, \quad H(1,t) = y, \quad H(s,0) = p(s), \quad H(s,1) = q(s),$$

donde $I = [0,1]$.

De esta manera, el conjunto relevante es:

$$\mathrm{Hom}_X(x,y) = \{[p] \mid p : I \to X,\ p(0) = x,\ p(1) = y\}$$

Como es conocido, podemos generar una operación de concatenación y de camino inverso que respeta las clases de homotopía.

$$\blacksquare : \mathrm{Hom}_X(x,y) \times \mathrm{Hom}_X(y,z) \to \mathrm{Hom}_X(x,z)$$
$$(-)^{-1} : \mathrm{Hom}_X(x,y) \to \mathrm{Hom}_X(y,x)$$

donde la concatenación es asociativa, tiene como neutro al camino constante refl_x, y es preservada adecuadamente por funciones continuas. Así, estas operaciones le dan una estructura de grupo a $\mathrm{Hom}_X(x,x)$, el cual es llamado el grupo fundamental de X, y es denotado por $\pi_1(X,x)$.

Por otro lado, el espacio X puede ser entendido también como una categoría que tiene como objetos los puntos en X, y como morfismos caminos de x, y (de ahí nuestra notación $\mathrm{Hom}_X(x,y)$). La concatenación de caminos es entonces la composición de morfismos, y el camino constante refl_x es el morfismo identidad. Como categoría, X tiene una propiedad adicional, cada morfismo es invertible.

Definición 3.1.1. Un **grupoide** es una categoría en donde todo morfismo es invertible.

Hemos descrito el grupoide fundamental $\Pi_1(X)$ de X, y observamos que el grupo fundamental usual $\pi_1(X, x)$ aparece como la subcategoría completa generada por $\{x\}$.

Pero este este solo es el nivel 1 de una jerarquía infinita de homotopías y de estructuras algebraicas. Una homotopía H usual entre dos caminos p y q puede ser entendida también como una superficie tal que sus bordes coinciden exactamente con p y q; es decir, H es un 2-camino entre 1-caminos. Un 3-camino sería una homotopía entre dos homotopías H y G que mantiene los bordes del 2-camino que generan.

Generalizando, dados dos n-caminos P y Q con el mismo borde, definimos por recursión que un $(n+1)$-camino entre P y Q, es un mapa $H : I^{n+1} \to X$ tal que $\partial\mathrm{Im}(H) = P \cup Q$.

Se puede verificar que las clases de homotopía de estos n-caminos poseen una operación de concatenación y una operación que genera caminos inversos, y además satisfacen algunas reglas adicionales, llamadas **reglas de coherencia**, las cuales los hacen n-grupoides para todo $n \in \mathbb{Z}^+$. Esto muestra que los espacios topológicos tienen estructura de ∞-grupoides[1].

Denotamos la estructura de n-grupoide de X por $\Pi_n(X)$. Y se tiene que, como en el caso del grupo fundamental, $\pi_n(X, x)$ es la subcategoría completa de $\Pi_n(X)$ generada por x.

Regresando a DTT, podemos entender una prueba p de que $x = y$ como un camino p de x a y. Las reglas del tipo de identidad nos permiten comparar si dos pruebas (caminos) $p, q : x =_X y$ son iguales entre sí; es decir si existe un $r : p =_{(x=_A y)} q$. Este es justamente un 2-camino, y de manera análoga podemos iterar infinitamente y observar una estructura de ∞-grupoides [14] [4].

En las siguientes secciones, verificaremos explícitamente la estructura de 1-grupoide de los tipos y caracterizaremos (hasta homotopía) los caminos en algunos de los tipos introducidos la sección previa.

Veremos que varias definiciones y resultados pueden interpretarse de tres formas distintas, de una forma lógica, de una forma homotópica, y de una forma categórica. Esto se debe a la correspondencia Curry-Howard, al hecho de que los tipos corresponden a tipos de homotopía de espacios topológicos y al hecho de que también son ∞-grupoides, respectivamente.

3.2. Los tipos son 1-grupoides

Desagregamos todas las condiciones que se deben cumplir para que los tipos sean 1-grupoides como descrito en la sección previa:

(ı) Existe la composición de morfismos $\blacksquare$.

[1] Dado que las reglas de coherencia solo se cumplen hasta homotopía, la estructura algebraica generada es de ∞-grupoides débiles; cada vez que hablemos de grupoides nos referiremos a grupoides débiles.

(II) Para cada objeto x, existe un morfismo identidad c_x tal que $c_x \cdot f = f \cdot c_y = f$ para todo $f : x \to y$.
(III) La composición de morfismos es asociativa.
(IV) Existe un morfismo f^{-1} inverso para cada morfismo $f : x \to y$, tal que $f^{-1} \cdot f = c_y$ y $f \cdot f^{-1} = c_x$.

Las primeras tres condiciones vienen de los axiomas de una categoría, mientras que el último es la condición de un grupoide. Procederemos a demostrar estas condiciones en orden.

Lema 3.2.1. *Para todo tipo A y todo $x, y, z : A$ existe la función de concatenación de caminos*

$$\_ \cdot \_ : (x = y) \to (y = z) \to (x = z),$$

tal que $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$.

Nótese que visto lógicamente, este lema enuncia la propiedad de transitividad de la igualdad. Probaremos este lema dos veces; la primera, usando el principio de inducción del tipo de identidades, y la segunda, usando búsqueda de patrones, a fin de comparar estos dos métodos.

Demostración por inducción. Definiremos una función $f : \prod_{(x,y:A)}(x = y) \to \prod_{(z:A)}(y = z) \to (x = z)$. Para aplicar inducción, sea $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ la familia de tipos definida por

$$D(x, y, p) :\equiv \prod_{(z:A)} \prod_{(q:y=z)} (x = z).$$

Nótese que $D(x, x, \mathsf{refl}_x) \equiv \prod_{(z:A)} \prod_{(q:x=z)}(x = z)$. Para obtener un elemento de este tipo, volveremos a aplicar inducción, sea $E : \prod_{(x,z:A)} \prod_{(q:x=z)} \to \mathcal{U}$ la familia de tipos definida por $E(x, z, q) :\equiv (x = z)$. Podemos definir

$$e(x) :\equiv \mathsf{refl}_x : E(x, x, \mathsf{refl}_x)$$

Por lo tanto, obtenemos una función

$$d : \prod_{(x,z:A)} \prod_{(q:x=z)} E(x, z, q)$$
$$d :\equiv \mathsf{ind}_{=_A}(E, e)$$

Con esta función, podemos definir

$$f : \prod_{(x,y:A)} \prod_{(p:x=y)} D(x, y, p)$$
$$f :\equiv \mathsf{ind}_{=_A}(D, d)$$

Finalmente, podemos definir

$$p \cdot q :\equiv f(x, y, p, z, q)$$

Además, se tiene que

$$\begin{aligned}\mathsf{refl}_x \cdot \mathsf{refl}_x &\equiv f(x, x, \mathsf{refl}_x, x, \mathsf{refl}_x)\\ &\equiv \mathsf{ind}_{=_A}(D, d, x, x, \mathsf{refl}_x, x, \mathsf{refl}_x)\\ &\equiv d(x, x, \mathsf{refl}_x)\\ &\equiv \mathsf{ind}_{=_A}(E, e, x, x, \mathsf{refl}_x)\\ &\equiv e(x)\\ &\equiv \mathsf{refl}_x\end{aligned}$$

□

Demostración por búsqueda de patrones. Buscamos una función

$$- \cdot - : (x = y) \to (y = z) \to (x = z)$$

Podemos asumir que y es x y que el primer camino es refl_x, por lo que reducimos la tarea a encontrar una función $\mathsf{refl}_x \cdot - : (x = z) \to (x = z)$.

Nuevamente, podemos asumir que z es x y el segundo camino también es refl_x, con lo que solo es necesario encontrar un elemento de $(x = x)$. Tomando $\mathsf{refl}_x : x = x$, podemos definir $\cdot$ por $\mathsf{refl}_x \cdot \mathsf{refl}_x :\equiv \mathsf{refl}_x$. □

Es claro que la segunda prueba, aunque omite ciertos detalles, es mucho más legible que la primera, mientras que conserva suficiente información para ser entendible. En efecto, los asistentes de prueba pueden inferir correctamente todos los datos omitidos. Por estos motivos, este es el estilo que tomaremos en las siguientes demostraciones.

Lema 3.2.2. *Para todo tipo A, $x, y : A$ y $p : x = y$, se tiene que $\mathsf{refl}_x \cdot p = p$ y $p \cdot \mathsf{refl}_y = p$.*

Demostración. Podemos asumir que p es refl_x, por lo que ambas ecuaciones se reducen a $\mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x$, lo cual se da por definición de $\cdot$. □

Este lema muestra que refl_x toma el rol del camino constante, y genera dos 2-caminos $\mathsf{refl\text{-}left} : \mathsf{refl}_x \cdot p =_{(x =_A y)} p$ y $\mathsf{refl\text{-}right} : p \cdot \mathsf{refl}_y =_{(x =_A y)} p$. No nombraremos a todos los n-caminos que definiremos explícitamente.

Lema 3.2.3. *Para todo tipo A, $x, y, z, w : A$, $p : x = y$, $q : y = z$ y $r : z = w$ se tiene $(p \cdot q) \cdot r = p \cdot (q \cdot r)$.*

Demostración. A través de repetidas aplicaciones de búsqueda de patrones, podemos asumir que p, q y r son refl_x, por lo que la ecuación se reduce a $(\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot \mathsf{refl}_x = \mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot \mathsf{refl}_x)$, lo cual se da por definición de $\cdot$. □

Dado este último lema, no utilizaremos paréntesis cuando haya una concatenación de varios caminos, como es común en álgebra.

Lema 3.2.4. *Para todo tipo A, $x, y : A$, y $p : x = y$, existe un $p^{-1} : y = x$, y este satisface que $p^{-1} \cdot p = \mathsf{refl}_y$ y $p \cdot p^{-1} = \mathsf{refl}_x$.*

Demostración. Asumiendo que p es refl_x, ponemos $\mathsf{refl}_x^{-1} :\equiv \mathsf{refl}_x$. En este caso, las dos ecuaciones se reducen a $\mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x$, lo cual se da por definición de $\cdot$. □

Nótese que el camino inverso representa el hecho lógico de que las igualdades son simétricas. Juntando estos lemas, obtenemos:

Teorema 3.2.5. *Los tipos son 1-grupoides, siendo los elementos los objetos, y los caminos $p : x = y$ los morfismos.*

Otros dos resultados clásicos y útiles, que se derivan de las reglas de 1-grupoides, son:

Lema 3.2.6. *Para todo tipo A, $x,y,z : A$, y $p : x = y$, $q : y = z$, se tiene que $(p^{-1})^{-1} = p$ y $(p \cdot q)^{-1} = q^{-1} \cdot p^{-1}$.*

Demostración. Asumiendo que p y q son refl_x, ambos lados de ambas expresiones se reducen a refl_x. □

Resaltamos que, a diferencia de en topología, la operación de concatenación y de inversas ha sido definida uniformemente para todos los caminos. Así, inmediatamente todos que los resultados previos, y los de las próximas secciones, aplican para cualquier n-camino.

3.3. Funciones y functores

En topología algebraica, es sabido que las funciones continuas entre espacios topológicos inducen un homomorfismo de grupos entre los grupos fundamentales. Pero más es cierto, las funciones continuas inducen un homomorfismo de ∞-grupoides.

Este es el caso también para las funciones no dependientes en DTT; verificaremos este hecho para la estructura de 1-grupoides. Comenzamos con las funciones no dependientes.

Lema 3.3.1. *Sea $f : A \to B$ una función, para todo $x,y : A$ existe una función*

$$\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y)).$$

tal que para todo $x : A$, se tiene que $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$.

Demostración. Asumiendo que el camino del dominio es refl_x, necesitamos encontrar un camino $f(x) =_B f(x)$, pero tenemos que $\mathsf{refl}_{f(x)}$ es un elemento de este tipo. □

Notación 3.3.2. En algunos casos, escribiremos $f(p)$ o fp en vez de $\mathsf{ap}_f(p)$, como es común en teoría de categorías.

Lógicamente, ap_f refleja el hecho obvio de que la igualdad es preservada bajo aplicación de funciones. Topológicamente, vemos que las funciones llevan caminos a caminos, lo que nos indica cierta noción de continuidad de todas las funciones. En efecto, las funciones entre tipos corresponden a funciones continuas entre tipos de homotopías de espacios topológicos. Categóricamente, como sugerido por la Notación 3.3.2, las funciones corresponden a functores entre grupoides.

Lema 3.3.3. *Sea $f : A \to B$ una función, y $p : x =_A y$ y $q : y =_A z$ dos caminos en A, entonces*

$$\mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q)$$

Demostración. Asumiendo que p y q son refl_x, ambos lados se reducen a $\mathsf{refl}_{f(x)}$, por definición de $\cdot$ y ap_f. □

Con el Lema 3.3.1 en mano, podemos aplicar un razonamiento algebraico sobre los caminos:

Lema 3.3.4. *Sea $f : A \to B$ una función, y $p : x =_A y$ un camino, entonces se tiene que $(f(p))^{-1} = f(p^{-1})$*

Demostración. El Lema 3.2.4 nos da un camino $q_1 : p^{-1} \cdot p = \mathsf{refl}_y$. Entonces, definiendo

$$g : (y = y) \to (f(y) = f(y))$$
$$g(r) :\equiv \mathsf{ap}_f(r),$$

obtenemos un camino

$$g(q_1) : f(p^{-1} \cdot p) = \mathsf{refl}_{f(y)}$$

Por otro lado, el Lema 3.3.3 nos da un camino $q_2 : f(p^{-1} \cdot p) = f(p^{-1}) \cdot f(p)$.

Juntando estos resultados, obtenemos que

$$q_2^{-1} \cdot g(q_1) : f(p^{-1}) \cdot f(p) = \mathsf{refl}_{f(y)}$$

Definiendo

$$h_1 : (f(y) = f(y)) \to (f(y) = f(x))$$
$$h_1(r) :\equiv r \cdot (f(p))^{-1},$$

obtenemos

$$h_1(q_2^{-1} \cdot g(q_1)) : \Big(f(p^{-1}) \cdot f(p)\Big) \cdot (f(p))^{-1} = \mathsf{refl}_{f(y)} \cdot (f(p))^{-1}$$

Por el Lema 3.2.2, tenemos caminos

$$q_3 : f(p^{-1}) \cdot \mathsf{refl}_{f(x)} = f(p^{-1})$$
$$q_4 : \mathsf{refl}_{f(y)} \cdot (f(p))^{-1} = (f(p))^{-1}$$

Por el Lema 3.2.4, tenemos el camino

$$q_5 : f(p) \cdot (f(p))^{-1} = \mathsf{refl}_{f(x)}$$

Poniendo

$$h_2 : (f(x) = f(x)) \to (f(y) = f(x))$$
$$h_2(r) :\equiv f(p^{-1}) \cdot r,$$

vemos que

$$h_2(q_5) : f(p^{-1}) \cdot \Big(f(p) \cdot (f(p))^{-1}\Big) = f(p^{-1}) \cdot \mathsf{refl}_{f(x)}$$

Por el Lema 3.2.3, existe un camino

$$q_6 : \Big(f(p^{-1}) \cdot f(p)\Big) \cdot (f(p))^{-1} = f(p^{-1}) \cdot \Big(f(p) \cdot (f(p))^{-1}\Big)$$

Juntando estos resultados previos, obtenemos:

$$\begin{aligned}
(f(p))^{-1} &= \mathsf{refl}_{f(y)} \cdot (f(p))^{-1} && \text{Por } q_4^{-1} \\
&= \Big(f(p^{-1}) \cdot f(p)\Big) \cdot (f(p))^{-1} && \text{Por } (h_1(q_2^{-1} \cdot g(q_1)))^{-1} \\
&= f(p^{-1}) \cdot \Big(f(p) \cdot (f(p))^{-1}\Big) && \text{Por } q_6 \\
&= f(p^{-1}) \cdot \mathsf{refl}_{f(x)} && \text{Por } h_2(q_5) \\
&= f(p^{-1}) && \text{Por } q_3
\end{aligned}$$

Nótese que esta cadena de igualdades corresponde en realidad a la siguiente concatenación de caminos:

$$(q_4)^{-1} \cdot (h_1(q_2^{-1} \cdot g(q_1)))^{-1} \cdot q_6 \cdot h_2(q_5) \cdot q_3 : (f(p))^{-1} = f(p^{-1})$$

□

Utilizaremos la práctica común de escribir la cadena de igualdades, dejando implícito el camino específico que se está usando, solo haciendo referencia al resultado previo del cual este se deriva, si es necesario.

Hemos construido un 2-camino específico en la demostración del lema anterior, pero nótese que este no es único. Pudimos, por ejemplo, utilizar inducción sobre p. Uno esperaría que estos dos 2-caminos sean iguales; es decir, que existe un 3-camino que los relaciona. Este es efectivamente el caso, como se puede ver por inducción, y es una de las consecuencias de las reglas de coherencia de un 2-grupoide.

No obstante, trabajar con las reglas de coherencia de n-caminos para $n \geq 2$ se vuelve rápidamente muy complicado. Veremos luego que en varios casos es posible reducir una pregunta sobre n-caminos, a una de 1-caminos, por lo que nuestro énfasis es en estos.

Otros resultados útiles e inmediatos son los siguientes:

Lema 3.3.5. *Sean $f : A \to B$ y $g : B \to C$ funciones, y $p : x =_A y$ un camino, entonces se tiene:*

(I) $\mathsf{ap}_{(g \circ f)}(p) = (\mathsf{ap}_g \circ \mathsf{ap}_f)(p)$
(II) $\mathsf{ap}_{\mathsf{id}}(p) = p$
(III) Sean $q, r : y =_A z$, con $p \cdot q = p \cdot r$, entonces $q = r$
(IV) Sean $q : x =_A y$ y $r : y =_A z$, con $p \cdot r = q \cdot r$, entonces $p = q$

Demostración. Para (I) y (II), asumiendo que p es refl_x, ambos lados se reducen a refl_x. Para (III) y (IV), estos son consecuencia del Lema 3.2.2, cuando asumimos que p y r son refl_y, respectivamente. □

3.4. Funciones dependientes y fibraciones

Sea $f : \prod_{(x:A)} B(x)$ una función dependiente, dado un camino $p : x = y$, se podría esperar que se tenga también un camino en $f(x) = f(y)$. Sin embargo, inspección de este último término muestra que este no tiene sentido: $f(x)$ está en $B(x)$, mientras que $f(y)$ está en $B(y)$, y caminos entre tipos diferentes no está definido. Lo que necesitamos es una forma de relacionar estos dos tipos, la función transporte nos brinda esta posibilidad.

Lema 3.4.1 (Transporte)**.** *Sea B una familia de tipos sobre A, y sea $p : x =_A y$ un camino, entonces existe una función*

$$\mathsf{tr}^B(p, -) : B(x) \to B(y)$$

Demostración. Podemos asumir que p es refl_x, en cuyo caso necesitamos una función $B(x) \to B(x)$, tomamos a la función identidad $\mathsf{id}_{B(x)}$ para esto. □

Entendiendo B como una propiedad que un elemento puede tener, el lema previo nos dice que, si x y y son iguales, entonces $B(x)$ implica $B(y)$. Aplicando el transporte en el camino inverso, obtenemos que $B(x)$ y $B(y)$ son lógicamente equivalentes. Así, podemos "transportar" propiedades B que tenga un elemento x a un elemento y que sea igual a este.

Categóricamente, vemos que tenemos otra categoría cuyos objetos son de la forma $B(x)$ para algún $x : A$, y los morfismos son funciones $f : B(x) \to B(y)$. La función transporte entonces lleva morfismos de A (visto como una categoría) a morfismos en esta nueva categoría presentada. Como uno esperaría, el transporte es un functor (contravariante):

Lema 3.4.2. *Sea B una familia de tipos sobre A, y sean $p : x =_A y$ y $q : y =_A z$ caminos, entonces*

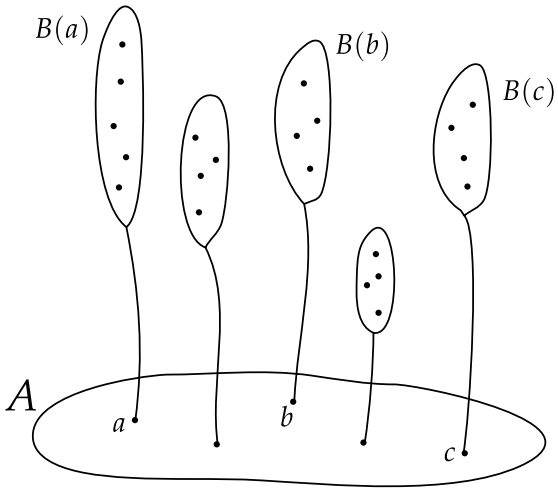
$$\mathsf{tr}^B(p \cdot q, -) = \mathsf{tr}^B(q, -) \circ \mathsf{tr}^B(p, -)$$

Demostración. Podemos asumir que p y q son refl_x, en cuyo caso la ecuación se reduce a $\mathsf{id}_B = \mathsf{id}_B \circ \mathsf{id}_B$, y podemos tomar $\mathsf{refl}_{\mathsf{id}_B}$. □

Aunque el punto de vista categórico nos brinda mucha información, es aún más útil considere la perspectiva homotópica la cual presentamos a continuación.

Dada una familia de tipos $B : A \to \mathcal{U}$, la idea es considerar el tipo $B(x)$ como la fibra sobre x, como se ve en la Figura 3.1. En efecto, en la primera proyección $\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$, aquellos elementos que son mapeados a a son justamente de la forma (a, y), por lo que podemos identificarlos con los elementos $y : B(x)$. Es más, se puede demostrar que este concepto de fibra es equivalente al concepto usual de preimagen (ver Proposición 3.5.10).

Figura 3.1: Tipos de familias como fibras



Con esta imagen en mente, dado un camino $p : x = y$, vemos que la función transporte induce una función entre las fibras $B(a)$ y $B(y)$, tal que la concatenación de caminos respeta la composición de las funciones inducidas. Así, un camino puede ser "levantado" a una función entre fibras.

Otra noción aún más importante de levantamiento, es la de levantamiento de caminos. Como ya mencionamos, no puede haber un camino entre elementos de $u : B(x)$ y $v : B(y)$ pues son tipos distintos, pero sí puede haber un camino entre los elementos $(x, u), (y, v) : \sum_{(x:A)} B(x)$.

Definición 3.4.3. Sea $B : A \to \mathcal{U}$ una familia de tipos. Diremos que un camino $q : (x, u) =_{\sum_{(x:A)} B(x)} (y, v)$ está sobre $p : x = y$, si $\mathsf{pr}_1(q) = p$. Utilizaremos la notación $q : u =^B_p v$ para estos casos.

Con esta definición, podemos enunciar la siguiente propiedad.

Lema 3.4.4. *(Propiedad de levantamiento de caminos) Sea $B : A \to \mathcal{U}$ una familia de tipos y sea $u : B(x)$ para algún $x : A$. Entonces, para todo $p : x = y$ tenemos un camino*

$$\mathsf{lift}(u, p) : (x, u) = (y, \mathsf{tr}^B(p, u))$$

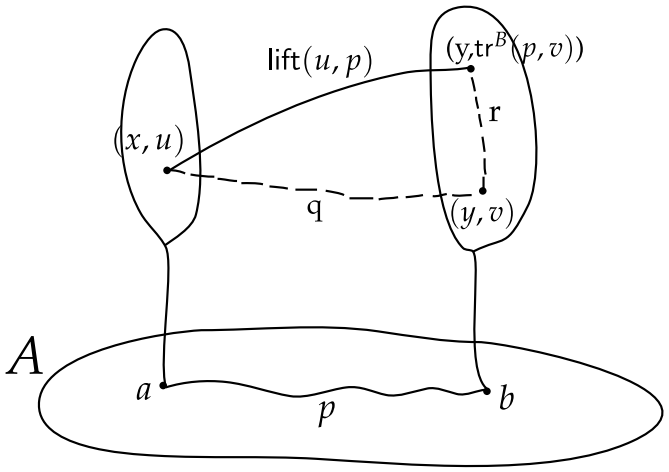
tal que $\mathsf{lift}(u, p)$ está sobre p.

Demostración. Podemos asumir que p es refl_x, en cuyo caso necesitamos una igualdad $(x,u) = (y, \mathsf{tr}^B(\mathsf{refl}_x, u))$, pero el lado derecho de esta igualdad es (x,u), por lo que podemos tomar $\mathsf{refl}_{(x,u)}$.

Para ver que $\mathsf{pr}_1(\mathsf{lift}(u,p)) = p$, supongamos nuevamente que p es refl_x, entonces nos queda probar $\mathsf{ap}_{\mathsf{pr}_1}\left(\mathsf{refl}_{(x,u)}\right) = \mathsf{refl}_x$. Pero esto es cierto por definición de ap. □

Aunque esta noción de caminos es muy natural y permite expresar y probar varias propiedades, no es muy fácil manejar. Notando que existe un camino canónico $\mathsf{lift}(u,p) : (x,u) = (y, \mathsf{tr}^B(p,u))$, entonces vemos que todo camino $q : (x,u) = (y,v)$ sobre $p : x = y$ factoriza por $\mathsf{lift}(x,u)$ a un camino $r : (y, \mathsf{tr}^B(p,u)) = (y,v)$ sobre el camino constante en y, ver Figura 3.2. Análogamente, todo camino $r : (y, \mathsf{tr}^B(p,u)) = (y,v)$ sobre refl_y puede ser expandido a un camino $q : (x,u) = (y,v)$ sobre p, pre-concatenando con $\mathsf{lift}(x,u)$.

Figura 3.2: Caminos sobre caminos



Mostraremos luego que esta correspondencia es en realidad una equivalencia (Proposición 3.5.11); es decir, tenemos que

$$\left((x,u) =_{\sum_{(x:A)} B(x)} (y,v)\right) \simeq \left(\mathsf{tr}^B(p,u) =_{B(y)} v\right)$$

El tipo de la derecha es mucho más manejable por lo que será el que utilizaremos para las aplicaciones.

Recordemos que una **fibración** $p : E \to X$ es un mapa que puede levantar cualquier homotopía en X hacia una en E. Con el lema previo, vemos entonces la relevancia del tipo $\sum_{(x:A)} B(x)$ en la perspectiva homotópica. El hecho de que haya una proyección $\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$, y que además tenemos esta propiedad de levantamiento de caminos, nos da indicios de que pr_1 es una fibración, con espacio total $\sum_{(x:A)} B(x)$. Este es efectivamente el caso, y es demostrado en la Proposición 4.1.1.

Dado esto, podemos entender a las funciones dependientes $f : \prod_{(x:A)} B(x)$ como **secciones**, pues toman un elemento de cada fibra $B(x)$. Como se esperaría,

dada una sección $f : \prod_{(x:A)} B(x)$, es posible levantar caminos $p : x = y$ a caminos $p' : f(x) =^B_p f(y)$.

Lema 3.4.5. *Sea $f : \prod_{(x:A)} B(x)$ una función dependiente, entonces existe una función*

$$\mathsf{apd}_f : \prod_{(p:x=y)} \left(\mathsf{tr}^B(p, f(x)) =_{B(y)} f(y)\right)$$

Demostración. Podemos asumir que p es refl_x, en cuyo caso necesitamos una igualdad $\mathsf{tr}^B(\mathsf{refl}_x, f(x)) =_{B(x)} f(x)$, pero el lado izquierdo de esta igualdad es $f(x)$, por lo que podemos tomar $\mathsf{refl}_{f(x)}$. □

Finalmente, retomando la perspectiva categórica, vemos que el transporte tiene una propiedad asociatividad con las funciones (vistas como functores):

Lema 3.4.6. *Sea $B : A \to \mathcal{U}$ una familia de tipos sobre el tipo A, y sea $f : A \to A$ una función. Para todos $x, y : A$ y $p : x = y$ se tiene*

$$\mathsf{tr}^B(f(p), -) = \mathsf{tr}^{B \circ f}(p, -)$$

Lema 3.4.7. *Sea $B : A \to \mathcal{U}$ una familia de tipos sobre el tipo A. Para todos $x, y : A$ y $p : x = y$ se tiene*

$$\mathsf{tr}^{\mathsf{id}}(B(p), -) = \mathsf{tr}^B(p, -)$$

Demostración. Por inducción en p, ambos lados de ambas ecuaciones se reducen a la función identidad. □

3.5. Equivalencias homotópicas

En MC, dos funciones son homotópicas cuando existe una deformación continua de sus imágenes. Dicho de otra forma, existe una función continua que une uniformemente las dos imágenes de cada punto en el dominio a través de un camino. Podemos generalizar esto para funciones dependientes:

Definición 3.5.1. Sean $f, g : \prod_{(x:A)} P(x)$ dos secciones. Una **homotopía** entre estas es una funcioń H en el tipo

$$(f \sim g) :\equiv \prod_{x:A} (f(x) = g(x))$$

Diremos que f y g son homotópicas cuando tengamos $f \sim g$.

Es claro que la propiedad de ser homotópicas es una relación de equivalencia, pues esto sigue del hecho de que las igualdades son una relación de equivalencia.

Desde la perspectiva lógica, una homotopía dice que dos funciones son iguales en cada punto del dominio. Categóricamente, una homotopía es una transformación natural.

Lema 3.5.2. *Sea $H : f \sim g$ una homotopía entre las funciones $f, g : A \to B$, y sea $p : x =_A y$. Entonces tenemos*

$$H(x) \cdot g(p) = f(p) \cdot H(y).$$

Diagramáticamente,

$$\begin{array}{ccc} f(x) & \xrightarrow{f(p)} & f(y) \\ {\scriptstyle H(x)}\downarrow & & \downarrow{\scriptstyle H(y)} \\ g(x) & \xrightarrow[g(p)]{} & g(y) \end{array}$$

Demostración. Por inducción, podemos suponer que p es refl_x. Entonces, es suficiente demostrar

$$H(x) \cdot \mathsf{refl}_{g(x)} = \mathsf{refl}_{f(x)} \cdot H(x).$$

Pero tenemos que ambos lados son iguales a $H(x)$. □

En topología clásica, cuando para una función $f : A \to B$ existe otra función $g : B \to A$ tal que $f \circ g \sim \mathsf{id}_B$ y $g \circ f \sim \mathsf{id}_A$, decimos que A y B tienen el mismo tipo de homotopía, y f es llamada una equivalencia homotópica. En DTT, esto correspondería a afirmar que el siguiente tipo está habitado.

$$\mathsf{qinv}(f) :\equiv \sum_{g:B\to A} \big((f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A)\big)$$

Categóricamente, y recordando que las homotopías son transformaciones naturales, este tipo indica la existencia de una equivalencia de categorías entre A y B. Sin embargo, este tipo no corresponde adecuadamente al concepto equivalencias homotópicas, ni al de equivalencias categóricas, sino al de quasi-inversas.

Definición 3.5.3. Sea $f : A \to B$ una función, una **quasi-inversa** de f es un triple (g, α, β) perteneciente al tipo $\mathsf{qinv}(f)$. Abusaremos la notación, y también llamaremos quasi-inversa a la función g del triple.

Ejemplo 3.5.4. La función identidad $\mathsf{id}_A : A \to A$ es su propia quasi-inversa, pues las dos homotopías requeridas se dan por reflexividad.

Ejemplo 3.5.5. Para todo $p : x =_A y$ y $P : A \to \mathcal{U}$, la función

$$\mathsf{tr}^P(p, -) : P(x) \to P(y)$$

tiene como quasi-inversa $\mathsf{tr}^P(p^{-1}, -)$, por el Lema 3.4.2.

La razón por la cual el tipo anterior no corresponde a equivalencias homotópicas se debe a que dos elementos $g, h : \mathsf{qinv}(f)$ podrían no ser iguales, lo cual traería problemas adelante. Así, lo que necesitamos es una noción lógicamente equivalente a $\mathsf{qinv}(f)$, pero que todos los elementos en este tipo sean iguales entre sí. Existen múltiples nociones que satisfacen este requisito, utilizaremos la de mapas bi-invertibles, al ser la más sencilla.

Definición 3.5.6. Decimos que una función $f : A \to B$ es **bi-invertible** si el siguiente tipo está habitado:

$$\mathsf{isequiv}(f) :\equiv \Big(\sum_{g:B\to A} (f \circ g \sim \mathsf{id}_B)\Big) \times \Big(\sum_{h:B\to A} (h \circ f \sim \mathsf{id}_A)\Big).$$

Proposición 3.5.7. *Para todo $f : A \to B$, los tipos $\mathsf{qinv}(f)$ y $\mathsf{isequiv}(f)$ son lógicamente equivalentes.*

Demostración. Es fácil dar una función $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$, pues dado un triple $(g, \alpha, \beta) : \mathsf{qinv}(f)$, tenemos que $(g, \alpha, g, \beta) : \mathsf{isequiv}(f)$. Por otro lado, dado (g, α, h, β), definamos γ como la homotopía

$$g \overset{\beta}{\sim} h \circ f \circ g \overset{\alpha}{\sim} h,$$

es decir, $\gamma(x) :\equiv \beta(g(x))^{-1} \cdot h(\alpha(x))$. Ahora definamos $\beta' : g \circ f \sim \mathsf{id}_A$ por $\beta'(x) :\equiv \gamma(f(x)) \cdot \beta(x)$. Entonces $(g, \alpha, \beta') : \mathsf{qinv}(f)$. □

La otra condición, que todos los elementos de $\mathsf{isequiv}(f)$ sean iguales entre sí es un resultado técnico, el cual se asumiremos como dado (ver [25, Sección 4.3]).

Con el concepto de bi-invertibilidad en mano, podemos definir qué es una equivalencia entre dos tipos.

Definición 3.5.8. Una **equivalencia** entre dos tipos A y B es una función $f : A \to B$ junto con un $p : \mathsf{isequiv}(f)$; es decir, una f junto con una prueba de que esta es bi-invertible. Definiremos el tipo de equivalencias entre A y B por

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{isequiv}(f)$$

Abusaremos el lenguaje, y diremos que f es una equivalencia cuando existe un $p : \mathsf{isequiv}(f)$. Recíprocamente, si tenemos $g : (A \simeq B)$, escribiremos $g(x)$, en vez de $\mathsf{pr}_1(g(x))$.

La noción de equivalencia previamente descrita es la de una equivalencia entre la estructura de tipos. Puesto que los tipos tienen una estructura natural de tipos de homotopía de espacios topológicos, y también la de ∞-grupoides, estas dos estructuras se preservan automáticamente. Nótese que esta implica inmediatamente una equivalencia lógica entre los dos tipos.

Asimismo, como la noción de bi-invertibilidad es lógicamente equivalente a la de tener una quasi-inversa, podemos usar este segundo concepto a la hora de realizar pruebas, como en el siguiente lema.

Lema 3.5.9. *La equivalencia entre tipos es una relación de equivalencia.*

Demostración. Hemos visto que la función identidad es su propia quasi-inversa, por lo que $\simeq$ es reflexiva.

Sea $f : A \to B$ una equivalencia, esta entonces debe tener una quasi-inversa $f^{-1} : B \to A$. Pero vemos que f es una quasi-inversa de f^{-1}, por lo que f^{-1} es una equivalencia $B \to A$.

Finalmente, dados $f : A \simeq B$ y $g : B \simeq C$ con quasi-inversas f^{-1} y g^{-1}; tenemos que para todo $x : A$

$$f^{-1}g^{-1}gfx = f^{-1}fx = x$$

mientras que para todo $y : C$

$$gff^{-1}g^{-1}y = gg^{-1}y = y$$

Por lo tanto, $f^{-1} \circ g^{-1}$ es una quasi-inversa de $g \circ f$, y concluimos $A \simeq C$. □

Veamos las demostraciones de algunas equivalencias que habíamos mencionado previamente.

Proposición 3.5.10. *Sea $B : A \to \mathcal{U}$ una familia de tipos. Entonces, para todo $x : A$ tenemos*

$$B(x) \simeq \sum_{z:\sum_{(a:A)} B(a)} \mathsf{pr}_1(z) = x$$

Demostración. Fijamos un x y definimos

$$f : B(x) \to \sum_{z:\sum_{(a:A)} B(a)} \mathsf{pr}_1(z) = x$$
$$f(y) :\equiv ((x, y), \mathsf{refl}_x)$$

En sentido contrario, definimos

$$g : \sum_{z:\sum_{(a:A)} B(a)} \mathsf{pr}_1(z) = x \to B(x)$$
$$g((x, y), \mathsf{refl}_x) :\equiv y$$

Por cómo fueron definidas, es inmediato que son quasi-inversas. □

Proposición 3.5.11. *Sea $B : A \to \mathcal{U}$ una familia de tipos. Entonces, para todo $x, y : A$, $p : x = y$, $u : B(x)$ y $v : B(y)$ tenemos*

$$\left(\sum_{q:(x,u)=(y,v)} \mathsf{pr}_1(q) = p\right) \simeq \left(\mathsf{tr}^B(p, u) =_{B(x)} v\right)$$

Demostración. Primero definimos

$$f : \prod_{p:x=y} \left(\sum_{q:(x,u)=(y,v)} \mathsf{pr}_1(q) = p\right) \to \left(\mathsf{tr}^B(p, u) =_{B(x)} v\right)$$
$$f(p, \mathsf{refl}_{(x,u)}, \mathsf{refl}_p) = \mathsf{refl}_u$$

En sentido contrario, definimos

$$g : \prod_{p:x=y} \left(\mathsf{tr}^B(p, u) =_{B(x)} v\right) \to \left(\sum_{q:(x,u)=(y,v)} \mathsf{pr}_1(q) = p\right)$$
$$g(p, \mathsf{refl}_{\mathsf{tr}^B(p,u)}) = (\mathsf{lift}(u, p), q)$$

donde $q : \mathsf{pr}_1(\mathsf{lift}(u, p)) = p$ es el camino dado por el Lema 3.4.4. Ahora, realizando inducción en p, vemos que $g(p) \circ f(p) \sim \mathsf{id}$ y $f(p) \circ g(p) \sim \mathsf{id}$, por lo que son quasi-inversas. □

Con estos conceptos ya definidos, procedemos a caracterizar los caminos en algunos de los tipos introducidos previamente.

3.6. Caminos entre pares dependientes

Dado un camino $p : (a, b) =_{A\times B} (a', b')$, podemos proyectar el camino sobre A y sobre B obteniendo dos caminos, $\mathsf{pr}_1(p) : a =_A a'$ y $\mathsf{pr}_2(p) : b =_B b'$. Intuitivamente, el camino p está únicamente determinado (hasta homotopía) por estos dos caminos.

Para el caso de pares dependientes, se esperaría un resultado similar. Es decir, que caminos $p : (a, b) =_{\sum_{(x:A)} B(x)} (a', b')$ están determinados por un par de caminos $\mathsf{pr}_1(p) : a =_A a'$ y $b =^B_p b'$. Este es efectivamente el caso.

Teorema 3.6.1. *Sea $P : A \to \mathcal{U}$ una familia de tipos sobre A, y sean $w, w' : \sum_{(x:A)} P(x)$. Entonces existe una equivalencia*

$$(w = w') \simeq \sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} \mathsf{pr}_2(w) =^P_p \mathsf{pr}_2(w').$$

Demostración. Podemos definir una función

$$f : \prod_{w,w':\sum_{(x:A)} P(x)} (w = w') \to \sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} \mathsf{tr}^P(p, \mathsf{pr}_2(w)) = \mathsf{pr}_2(w')$$

por inducción de caminos:

$$f(w, w, \mathsf{refl}_w) :\equiv (\mathsf{refl}_{\mathsf{pr}_1(w)}, \mathsf{refl}_{\mathsf{pr}_2(w)}).$$

En la dirección contraria, podemos definir

$$g : \prod_{w,w':\sum_{(x:A)} P(x)} \Big(\sum_{p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w')} \mathsf{tr}^P(p, \mathsf{pr}_2(w)) = \mathsf{pr}_2(w') \Big) \to (w = w')$$

realizando inducción en w y en w', por lo que es suficiente mostrar

$$\Big(\sum_{p:w_1=w'_1} \mathsf{tr}^P(p, w_2) = w'_2 \Big) \to ((w_1, w_2) = (w'_1, w'_2)).$$

Realizando inducción nuevamente en ambos caminos del dominio, vemos que podemos tomar $\mathsf{refl}_{(w_1,w_2)}$ como el caso base.

Ahora, solo falta demostrar que $f \circ g \sim \mathsf{id}$ y $g \circ f \sim \mathsf{id}$, pero esto es inmediato por las definiciones, luego de aplicar inducción. □

La función g definida arriba es particularmente útil, y la llamaremos $\mathsf{pair}^=$, usualmente usándola omitiendo los puntos base w, w'.

Como un corolario, tenemos que los pares dependientes están caracterizados por sus dos coordenadas.

Corolario 3.6.2. *Para* $z : \sum_{(x:A)} P(x)$*, tenemos* $z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$.

Demostración. Tenemos $\mathsf{refl}_{\mathsf{pr}_1(z)} : \mathsf{pr}_1(z) = \mathsf{pr}_1(\mathsf{pr}_1(z), \mathsf{pr}_2(z))$, por lo que por el teorema anterior es suficiente mostrar un camino

$$\mathsf{tr}^P((\mathsf{refl}_{\mathsf{pr}_1(z)}), \mathsf{pr}_2(z)) = \mathsf{pr}_2(\mathsf{pr}_1(z), \mathsf{pr}_2(z))$$

Pero ambos lados son iguales a $\mathsf{pr}_2(z)$. □

Este mismo argumento aplica para pares no dependientes, por lo que obtenemos el resultado mencionado al inicio de la sección.

Corolario 3.6.3. *Sean* A *y* B *dos tipos, y sean* $w, w' : A \times B$*. Entonces existe una equivalencia*

$$(w = w') \simeq \left(\mathsf{pr}_1(w) = \mathsf{pr}_1(w')\right) \times \left(\mathsf{pr}_2(w) = \mathsf{pr}_2(w')\right).$$

3.7. Caminos entre funciones dependientes

En MC, cuando dos funciones $f, g : A \to B$ son iguales en cada punto del dominio, estas son iguales. Entonces, uno esperaría tener

$$(f = g) \simeq \left(\prod_{x:A} (f(x) =_{B(x)} g(x))\right)$$

Por un lado, podemos definir

$$\mathsf{happly} : (f = g) \to \prod_{x:A} (f(x) =_{B(x)} g(x))$$
$$\mathsf{happly}(p) :\equiv \lambda(x : A).\, \mathsf{ap}_{\lambda h.\, h(x)} p$$

Para definir la quasi-inversa, se puede usar univalencia y una serie de argumentos sofisticados (ver [25, Sección 4.9]). En el Teorema 4.3.3, veremos una demostración de esto para el caso de funciones no dependientes. En todo caso, es común introducir el siguiente (redundante) axioma:

Axioma 3.7.1 (Extensionalidad de funciones)**.** *La función* happly *es una equivalencia.*

Este axioma entonces, postula la existencia de un elemento φ en el tipo $\mathsf{isequiv}(\mathsf{happly})$. Todos los axiomas en DTT son de esta forma; es decir, indican la existencia de un elemento en algún tipo. En particular, el axioma previo implica la existencia de una quasi-inversa de happly

$$\mathsf{funext} : \left(\prod_{x:A} (f(x) = g(x))\right) \to (f = g),$$

junto con homotopías $\mathsf{happly} \circ \mathsf{funext} \sim \mathsf{id}$ y $\mathsf{funext} \circ \mathsf{happly} \sim \mathsf{id}$.

Topológicamente, estamos identificando las funciones homotópicas entre sí. Esto sugeriría que podemos identificar espacios homotópicamente equivalentes, lo cual realizamos en la siguiente sección.

Con la función funext en mano, podemos mostrar que el tipo $\mathbf{0}$ es un objeto inicial en Type.

Proposición 3.7.2. *Para todo tipo C y función $f : \mathbf{0} \to C$, tenemos que $f = \mathsf{rec}_{\mathbf{0}}(C)$.*

Demostración. Para todo $x : \mathbf{0}$ podemos inmediatamente concluir que $f(x) = \mathsf{rec}_{\mathbf{0}}(C)(x)$, por lo que $f = \mathsf{rec}_{\mathbf{0}}(C)$. □

Veamos con interactúan las el transporte en familias de tipos de funciones.

Lema 3.7.3. *Sea X un tipo y $A, B : X \to \mathcal{U}$ dos familias de tipos. Para todos $x_1, x_2 : X$, $p : x_1 = x_2$, $f : A(x_1) \to B(x_1)$ y $y : A(x_2)$ se tiene que*

$$\mathsf{tr}^{\lambda(x:X).\, A(x)\to B(x)}(p, f) = \mathsf{tr}^{B}(p, f(\mathsf{tr}^{A}(p^{-1}, x)))$$

Demostración. Por inducción, podemos asumir que p es refl_{x_1}, en cuyo caso ambos lados de la igualdad se reducen a x. □

Lema 3.7.4. *Sea X un tipo, y sean $A : X \to \mathcal{U}$ y $B : \prod_{(x:X)} A(x) \to \mathcal{U}$ dos familias de tipos. Para todos $x_1, x_2 : X$, $p : x_1 = x_2$, dos funciones $f : \prod_{(y:A(x_1))} B(x_1, y)$ y $g : \prod_{(y:A(x_2))} B(x_2, y)$.*

Si para todos $a_1 : A(x_1)$ y $a_2 : A(x_2)$ y $q : a_1 =^{A}_{p} a_2$ se tiene que

$$f(a_1) =^{\lambda(w:\sum_{(x:A)} A(x)).\, B(\mathsf{pr}_1(w), \mathsf{pr}_2(w))}_{\mathsf{pair}^{=}(p,q)} g(a_2)$$

entonces

$$f =^{\lambda(x:X).\, \prod_{(a:A(x))} B(x,a)}_{p} g$$

Demostración. Por extensionalidad de funciones, es suficiente mostrar que las funciones son iguales cuando son aplicadas en un elemento arbitrario $y : A(x_2)$. Por inducción, podemos asumir que p es refl_{x_1}, en cuyo caso, aplicando la hipótesis para $q = \mathsf{refl}_y$, obtenemos que $f(y) = g(y)$. □

3.8. Caminos entre tipos

Una de las prácticas comunes de MC es identificar objetos isomórficos, pues estos comparten todas las propiedades relevantes del área de estudio. Así, hablamos *del* grupo cíclico de orden 3, no de *un* grupo cíclico de orden 3.

Por un lado, vemos que tipos identificables son equivalentes entre sí:

Lema 3.8.1. *Para tipos $A, B : \mathcal{U}$, existe una función*

$$\mathsf{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B)$$

Demostración. Primero notamos que la función $\mathsf{id}_{\mathcal{U}}$ es una familia de tipos que tiene como base el universo entero. Ahora, afirmamos que para todo $p : A = B$, $\mathsf{tr}^{\mathsf{id}_{\mathcal{U}}}(p, -) : A \to B$ es una equivalencia. Aplicando inducción sobre p esta función se reduce a id_A, y sabemos que la identidad es una equivalencia, con lo que concluimos el resultado. □

Sin embargo, no es posible, en general, generar una igualdad entre dos tipos a partir de una equivalencia entre ellos. Para esto, es necesario otro axioma.

Axioma 3.8.2 (Univalencia)**.** *Para todos* $A, B : \mathcal{U}$, $\mathsf{idtoeqv}$ *es una equivalencia.*

Tenemos entonces $(A = B) \simeq (A \simeq B)$, así como una quasi-inversa de $\mathsf{idtoeqv}$:

$$\mathsf{ua} : (X \simeq Y) \to (X = Y).$$

Este es el primer axioma que nos separa de Dependent Type Theory. DTT usualmente asume también el axioma K, que indica que todos los caminos con mismos extremos son homotópicos entre sí. Aunque esto no tiene sentido topológicamente, sí es intuitivo desde una perspectiva lógica, pues un camino es una prueba de que $x = y$. Así, K postula que todas las pruebas son iguales entre sí.

A fin de no colapsar la estructura que hay entre los caminos, no hemos introducido este axioma; adicionalmente, si se asume este junto con el axioma de Univalencia, la teoría se vuelve inconsistente [25, Lema 6.4.1.].

Entonces, hemos departido definitivamente de la teoría original, y comenzamos propiamente una nueva teoría llamada *Homotopy Type Theory* (HoTT), la cual traducimos como Teoría Homotópica de Tipos. Si bien todos los resultados previos se cumplen en DTT, varios de estos se cumplen trivialmente, al ser todos los caminos iguales entre sí, asumiendo K.

En particular, el axioma de univalencia formaliza adecuadamente la práctica mencionada de identificar objetos isomórficos. Si A y B son tipos equivalentes, podemos generar un camino $p : A = B$; por lo que cualquier propiedad que tenga A, también la tiene B, pues puede ser transportada a lo largo de p.

3.9. Caminos entre naturales

A diferencia de los otros tipos, caracterizar a los caminos entre naturales requiere una técnica más sofisticada, la cual se llama el método **encode-decode**. Se llama así pues encodificamos el tipo $m = n$ en otro tipo más manejable, lo que nos facilita realizar pruebas respecto al tipo $m = n$. Esta técnica se puede utilizar para caracterizar el coproducto, un resultado que no necesitaremos, y también para calcular el grupo fundamental del círculo (ver Sección 4.4).

Retornando a los naturales, encodificaremos caminos en los naturales por el tipo

$$\mathsf{code} : \mathbb{N} \to \mathbb{N} \to \mathcal{U}$$

el cual está definido por

$$\begin{aligned} \mathsf{code}(0,0) &:\equiv \mathbf{1} \\ \mathsf{code}(\mathsf{succ}(m),0) &:\equiv \mathbf{0} \\ \mathsf{code}(0,\mathsf{succ}(n)) &:\equiv \mathbf{0} \\ \mathsf{code}(\mathsf{succ}(m),\mathsf{succ}(n)) &:\equiv \mathsf{code}(m,n). \end{aligned}$$

Nótese que $\mathsf{code}(m,n)$ corresponde al algoritmo que resta 1 a m y a n hasta que al menos uno de los dos sea igual a 0.

Teorema 3.9.1. *Para todo $m, n : \mathbb{N}$, se tiene que $(m = n) \simeq \mathsf{code}(m, n)$.*

Demostración. Por un lado, para definir

$$\mathsf{encode} : \prod_{m,n:\mathbb{N}} (m = n) \to \mathsf{code}(m, n),$$

notamos que podemos definir una función $r : \prod_{(n:\mathbb{N})} \mathsf{code}(n, n)$ por inducción

$$\begin{aligned} r(0) &:\equiv \star \\ r(\mathsf{succ}(n)) &:\equiv r(n) \end{aligned}$$

De tal forma que podemos definir

$$\mathsf{encode}(m, n, p) :\equiv \mathsf{tr}^{\mathsf{code}(m,-)}(p, r(m))$$

En sentido contrario, podemos definir

$$\mathsf{decode} : \prod_{m,n:\mathbb{N}} \mathsf{code}(m, n) \to (m = n)$$

realizando inducción en ambos m y n:

$$\begin{aligned} \mathsf{decode}(0, 0, c) &:\equiv \mathsf{refl}_0 \\ \mathsf{decode}(\mathsf{succ}(m), 0, c) &:\equiv \mathsf{ind}_{\mathbf{0}}(\mathsf{succ}(m) = 0, c) \\ \mathsf{decode}(0, \mathsf{succ}(n), c) &:\equiv \mathsf{ind}_{\mathbf{0}}(0 = \mathsf{succ}(n), c) \\ \mathsf{decode}(\mathsf{succ}(m), \mathsf{succ}(n), c) &:\equiv \mathsf{succ}(\mathsf{decode}(m, n)) \end{aligned}$$

Ahora, solo falta comprobar que estas funciones son quasi-inversas. Veamos primero que

$$\mathsf{decode}(m, n) \circ \mathsf{encode}(m, n) \sim \mathsf{id}.$$

Dado $p : m = n$, podemos realizar inducción y asumir que n es m; es decir, es suficiente mostrar

$$\mathsf{decode}(m, m, \mathsf{encode}(m, m, \mathsf{refl}_m)) = \mathsf{refl}_m.$$

Pero por definición, $\mathsf{encode}(n, n, \mathsf{refl}_n) \equiv r(n)$, así que solo es suficiente demostrar que $\mathsf{decode}(n, n, r(n)) = \mathsf{refl}_n$, realizamos esto por inducción. Para el caso base, vemos que $\mathsf{decode}(0, 0, r(0))$ es igual a refl_0 por definición; para el caso de un $\mathsf{succ}(n)$ necesitamos mostrar que

$$\mathsf{succ}(\mathsf{decode}(n, n, r(n))) = \mathsf{refl}_{\mathsf{succ}(n)}$$

Pero por la hipótesis de inducción, $\mathsf{decode}(n, n, r(n)) = \mathsf{refl}_n$, por lo que obtenemos el resultado deseado.

Solo falta demostrar la otra homotopía,

$$\mathsf{encode}(m, n) \circ \mathsf{decode}(m, n) \sim \mathsf{id}.$$

Dado $c : \mathsf{code}(m, n)$, procederemos por inducción en ambos m y n. Cuando ambos son 0, $\mathsf{code}(0, 0) \equiv \mathbf{1}$, por lo que podemos asumir que $c \equiv \star$, y ambos lados se reducen $\star$. Si uno es un sucesor y el otro no, $\mathsf{code}(m, n)$ es $\mathbf{0}$, por lo que podemos derivar cualquier resultado a partir de $c : \mathbf{0}$. Finalmente, si los dos son sucesores, tenemos

$$
\begin{aligned}
&\mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{decode}(\mathsf{succ}(m), \mathsf{succ}(n), c)) \\
&\quad \equiv \mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{succ}(\mathsf{decode}(m, n, c))) && \text{(Def. de } \mathsf{decode}\text{)} \\
&\quad \equiv \mathsf{tr}^{\mathsf{code}(\mathsf{succ}(m), -)}(\mathsf{succ}(\mathsf{decode}(m, n, c)), r(\mathsf{succ}(m))) && \text{(Def. de } \mathsf{encode}\text{)} \\
&\quad = \mathsf{tr}^{\mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(-))}(\mathsf{decode}(m, n, c), r(\mathsf{succ}(m))) && \text{(Lema 3.4.6)} \\
&\quad \equiv \mathsf{tr}^{\mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(-))}(\mathsf{decode}(m, n, c), r((m))) && \text{(Def. de } r\text{)} \\
&\quad \equiv \mathsf{tr}^{\mathsf{code}(m, -)}(\mathsf{decode}(m, n, c), r(m)) && \text{(Def. de } \mathsf{code}\text{)} \\
&\quad \equiv \mathsf{encode}(m, n, \mathsf{decode}(m, n, c)) && \text{(Def. de } \mathsf{encode}\text{)} \\
&\quad = c && \text{(Hip. inductiva.)}
\end{aligned}
$$

□

Como una aplicación de esta caracterización, veamos que la función $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ es inyectiva.

Corolario 3.9.2. *Sean $m, n : \mathbb{N}$ tal que $\mathsf{succ}(m) = \mathsf{succ}(n)$, entonces, $m = n$.*

Demostración. Dado $p : \mathsf{succ}(m) = \mathsf{succ}(n)$, tenemos

$$\mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), p) : \mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(n)) \equiv \mathsf{code}(m, n),$$

por lo que

$$\mathsf{decode}(m, n, \mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), p)) : m = n.$$

□

Otra consecuencia es que la igualdad entre naturales es **decidible**; es decir, tenemos el siguiente resultado.

Corolario 3.9.3. *Sean $m, n : \mathbb{N}$, entonces se tiene que $(m = n) \vee \neg(m = n)$.*

Demostración. En otras palabras, la proposición nos indica que tenemos un elemento del tipo

$$\prod_{m,n:\mathbb{N}} (m = n) + ((m = n) \to \mathbf{0}).$$

Lo generaremos por inducción en m y n. Cuando ambos son 0, podemos tomar $\mathsf{inl}(\mathsf{refl}_0)$. Si el primero es de la forma $\mathsf{succ}(m)$ mientras que el segundo es 0, si tuviésemos $p : \mathsf{succ}(m) = 0$ entonces aplicando encode obtenemos un elemento de $\neg(m = n)$. Análogamente cuando el primero es 0 y el segundo es $\mathsf{succ}(n)$.

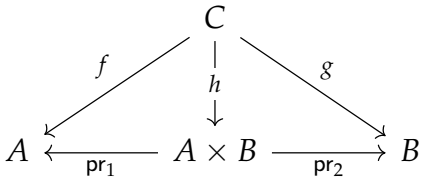
Finalmente, cuando tenemos $\mathsf{succ}(m)$ y $\mathsf{succ}(n)$, la hipótesis de inducción nos dice que tenemos un elemento de $(m = n) + \neg(m = n)$. Por inducción en el coproducto, si tenemos $m = n$, podemos concluir $\mathsf{succ}(m) = \mathsf{succ}(n)$, mientras que si tenemos $\neg(m = n)$, podemos generar una contradicción cuando $\mathsf{succ}(m) = \mathsf{succ}(n)$, por el corolario previo. □

3.10. Propiedades Universales

Con los conceptos ya introducidos, podemos ver que los tipos introducidos tienen las propiedades universales esperdads.

Teorema 3.10.1. *El tipo $A \times B$ es el producto de A y B en la categoría* Type.

Demostración. Dados un tipo $C : \mathcal{U}$, y un par de morfismos $f : C \to A$ y $g : C \to B$, buscamos encontrar una única función $h : C \to A \times B$ tal que el diagrama conmute.



Podemos definir h por $h(c) :\equiv (f(c), g(c))$. Por otro lado, si hubiese otra función $h' : C \to A \times B$ que hace que el diagrama conmute, para obtener $h' = h$ basta mostrar que $h'(c) = h(c)$ para todo $c : C$, por extensionalidad de funciones.

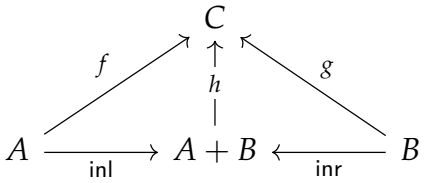
Ahora, por Corolario 3.6.3, es suficiente mostrar que

$$\mathsf{pr}_1(h'(c)) = \mathsf{pr}_1(h(c)) \quad \text{y} \quad \mathsf{pr}_2(h'(c)) = \mathsf{pr}_2(h(c)),$$

pero esto se da por definición de h y por la condición de que h' hace que el diagrama conmute. □

Teorema 3.10.2. *El tipo $A + B$ es el coproducto de A y B en la categoría* Type.

Demostración. Dados un tipo $C : \mathcal{U}$, y un par de morfismos $f : A \to C$ y $g : B \to C$, buscamos encontrar una única función $h : A + B \to C$ tal que el diagrama conmute.



Podemos definir h por $h(\mathsf{inl}(a)) :\equiv f(a)$ y $h(\mathsf{inr}(b)) :\equiv g(b)$. Por otro lado, si hubiese otra función $h' : A + B \to C$ que hace que el diagrama conmute, para obtener $h' = h$ basta mostrar que $h'(x) = h(x)$ para todo $x : A + B$, por extensionalidad de funciones.

Ahora, por inducción en $A + B$, basta que

$$h'(\mathsf{inl}(a)) = h(\mathsf{inl}(a)) \quad \text{y} \quad h'(\mathsf{inr}(b)) = h(\mathsf{inr}(b)),$$

pero esto se da por definición de h y por la condición de que h' hace que el diagrama conmute. □

El tipo de los naturales también satisface una propiedad universal; es decir, es el objeto inicial la siguiente categoría.

Definición 3.10.3. La categoría $\mathsf{Type}_{\mathbb{N}}$ tiene como objetos triples

$$(C, c, f) : \sum_{C:\mathcal{U}} C \times (C \to C),$$

y como morfismos $\alpha : (A, a, f) \to (B, b, g)$ funciones $h : A \to B$ tales que $h(a) = b$ y $h \circ f = g \circ h$. Diagramáticamente:

$$\begin{array}{ccc} A & \xrightarrow{f} & A \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h} \\ B & \xrightarrow[g]{} & B \end{array} \qquad\qquad \begin{array}{c} a \\ {\scriptstyle h}\downarrow \\ b \end{array}$$

Teorema 3.10.4. *El triple* $(\mathbb{N}, 0, \mathsf{succ})$ *es inicial en la categoría* $\mathsf{Type}_{\mathbb{N}}$.

Demostración. Dado un triple (C, c, f) en $\mathsf{Type}_{\mathbb{N}}$ podemos definir una función $h : \mathbb{N} \to C$ por recursión; es decir, $h :\equiv \mathsf{rec}_{\mathbb{N}}(C, c, f)$. Por definición de h, se cumplen las dos propiedades requeridas:

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\mathsf{succ}} & \mathbb{N} \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[f]{} & C \end{array} \qquad\qquad \begin{array}{c} 0 \\ {\scriptstyle h}\downarrow \\ c \end{array}$$

Ahora, dado otro $h' : \mathbb{N} \to C$ tal que $h'(0) = c$ y $h' \circ \mathsf{succ} = f \circ h'$, mostraremos que $h' = h$. Por extensionalidad de funciones basta mostrar que $h'(n) = n$ para todo $n : \mathbb{N}$. Realizaremos esto por inducción en los naturales.

Para el caso base, tenemos

$$h'(0) = c \equiv h(0),$$

mientras que para el paso inductivo tenemos

$$\begin{aligned} h'(\mathsf{succ}(n)) &= f(h'(n)) && \text{Por definición de } h' \\ &= f(h(n)) && \text{Por el supuesto inductivo} \\ &\equiv h(\mathsf{succ}(n)) && \text{Por definición de } h \end{aligned}$$

□

La razón de que $\mathbb{N}$ sea inicial en esta categoría específica se debe a los **constructores** de $\mathbb{N}$; es decir, las formas en las que se pueden generar elementos de $\mathbb{N}$. Los naturales tienen dos constructores, el elemento 0 y la función $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$; de ahí que sea inicial en la categoría que tiene como objetos aquellos con la misma estructura interna.

De este resultado se sigue que pudo haberse definido a los naturales como aquel tipo que está libremente generado por los dos siguientes constructores:

- un elemento $0 : \mathbb{N}$
- una función $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$.

Los tipos generados de esta manera se llaman **tipos inductivos** y veremos una generalización de estos en la Sección 4.3.

3.11. Metateoría de HoTT

Puestos que hemos introducido el axioma de univalencia, los resultados previos descritos para la metateoría de DTT no necesariamente aplican en HoTT. Sin embargo, existen modelos de HoTT, como el desarrollado en [13].

A grandes rasgos, un modelo es una estructura matemática que satisface todas las reglas y axiomas de una teoría matemática. En el caso del modelo previo, este es un modelo simplicial; los objetos son representados por ciertas clases de conjuntos simpliciales, y las funciones son, a grosso modo, representadas por (clases de equivalencia) de funciones continuas.

Resaltamos dos consecuencias de esto. Primero, tenemos un análogo del Teorema 2.12.4.

Teorema 3.11.1. *(Consistencia de HoTT) No es posible derivar una contradicción en HoTT, es decir* $\cdot \vdash x : \mathbf{0}$ *para algún* x*; asumiendo que la teoría usual de MC también es consistente.*

Por otro lado, este resultado nos garantiza que podemos pensar en los tipos como clases de homotopía de espacios topológicos, y, como ya habíamos sugerido, las funciones realmente son continuas.

Capítulo 4

Teoría Homotópica de Tipos

En este capítulo, veremos algunas aplicaciones de HoTT en la topología algebraica. Analizaremos a mayor profundidad la estructura homotópica de los tipos, y veremos algunos resultados clásicos involucrando conceptos como el de contractibilidad, CW complejos, y el grupo fundamental.

4.1. Topología y tipos

Puesto que los tipos son tipos de homotopía de espacios topológicos, se puede formalizar varias propiedades clásicas que estos cumplen. Para comenzar, ahora podemos ver la demostración completa de la proposición mencionada previamente de que las familias de tipos son fibraciones.

Proposición 4.1.1. *Sea $P : B \to \mathcal{U}$ una familia de tipos sobre B, y $f, g : X \to B$ dos funciones tales que existe una homotopía h entre ellas. Si existe un levantamiento de f, $\tilde{f} : \prod_{(x:X)} P(f(x))$; entonces existe una homotopía $\tilde{h}$ entre $\tilde{f}$ y un levantamiento de g, tal que $\tilde{h}$ levanta a h,*

Demostración. Podemos definir una homotopía

$$\tilde{h} : \prod_{x:X} (f(x), \tilde{f}(x)) =_{\sum_{(x:X)} P(x)} \left(g(x), \mathsf{tr}^P(h(x), \tilde{f}(x))\right)$$

por $\tilde{h}(x) = \mathsf{pair}^{=}(h(x), \mathsf{refl})$. Para ver que esta levanta a H, necesitamos comprobar que

$$\prod_{x:X} \mathsf{pr}_1(\tilde{h}(x)) = h(x)$$

Sin embargo, por inducción, es fácil notar que $\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{pair}^{=}(p, q)) = p$, para todo p, q. Aplicando esto en la ecuación anterior, obtenemos el resultado deseado. □

Notamos que, con los conceptos ya introducidos, es fácil formalizara varios conceptos topológicos. Por ejemplo, podemos definir los conceptos de retracciones y secciones.

Definición 4.1.2. Una **retracción** es una función $r : A \to B$ tal que existe una función $s : B \to A$, su **sección**, y una homotopía $\epsilon : \prod_{(y:B)}(r(s(y)) = y)$. Cuando esto se de, decimos que B es un **retracto** de A.

Veamos cómo podemos hacer un análisis similar a otros conceptos.

4.2. n-tipos

En topología clásica, un espacio topológico es llamado contractible cuando la función identidad es homotópica a la imagen de un solo punto. Esto sugiere:

Definición 4.2.1. Un tipo A es **contractible** cuando posee un elemento $a : A$, llamado el **centro de contracción**, tal que $a = x$, para todo $x : A$. Es decir, definimos la propiedad de ser contractible como:

$$\mathsf{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x).$$

Ejemplo 4.2.2. El tipo $\mathbf{1}$ es contractible, pues para mostrar que $\prod_{(x:\mathbf{1})}(\star = x)$, podemos asumir que x es $\star$, y podemos tomar el camino constante.

Con este resultado, podemos demostrar que el tipo $\mathbf{1}$ es un objeto terminal.

Proposición 4.2.3. *Para todo tipo C y función $f : C \to \mathbf{1}$, se tiene que $f = !\mathbf{1}_C$.*

Demostración. Para todo $x : C$, se tiene $f(x) = \star$, por el resultado previo. Por extensionalidad de funciones, entonces $f = !\mathbf{1}_C$. □

Desde una perspectiva lógica, la contractibilidad indica que todos los elementos del tipo son iguales a un elemento en específico. Topológicamente, tenemos una función continua que lleva cualquier punto $x : A$ hacia a. Esto sugiere que el tipo A es equivalente al tipo $\mathbf{1}$.

Proposición 4.2.4. *Un tipo A es contractible sí y solo sí este es equivalente al tipo $\mathbf{1}$.*

Demostración. Sea $(c, p) : \mathsf{isContr}(A)$, podemos definir $f : A \to \mathbf{1}$ por $f(x) = \star$. En sentido contrario, tenemos $g : \mathbf{1} \to A$ dado por $g(y) = c$. Ahora, por un lado, realizando inducción en $\mathbf{1}$, vemos que para todo $y : \mathbf{1}$, $f(g(y)) = \star$. Por otro lado, para todo $x : A$, tenemos $g(f(x)) = c = x$, donde la segunda igualdad se da por $p(x)$. Juntando estos resultamos, obtenemos que $A \simeq \mathbf{1}$.

Para mostrar la recíproca, supongamos que $A \simeq \mathbf{1}$, y sean $f : A \to \mathbf{1}$ y $g : \mathbf{1} \to A$ las quasi-inversas. Por el ejemplo previo, tenemos que $\star = f(x)$. Así, se tiene:

$$g(\star) = g(f(x)) = x,$$

con lo que $g(\star)$ es el centro de contracción. □

Veamos que este, al igual que en MC, esta propiedad se preserva bajo varias operaciones, como productos y retracciones.

Proposición 4.2.5. *Sea $B : A \to \mathcal{U}$ una familia de tipos. Si A es contractible y para todo $x : A$, $B(x)$ es contractible, entonces $\sum_{(x:A)} B(x)$ también lo es.*

Demostración. Sea $a_0 : A$ el centro de contracción de A, y sea $b_0 : B(a_0)$ el centro de contración de $B(a_0)$. Mostraremos que (a_0, b_0) es un centro de contracción. Dado $(a, b) : \sum_{(x:A)} B(x)$, necesitamos mostrar camino $p : a_0 = a$ tal que $\mathsf{tr}^B(p, b_0) = b$. Por un lado, este p existe por contractibilidad de A. Por otro lado, tenemos:

$$\begin{aligned} \mathsf{tr}^B(p, b_0) &= \mathsf{tr}^B(p, \mathsf{tr}^B(p^{-1}, b)) && \text{Por contractibilidad de } B(a_0) \\ &= \mathsf{tr}^P(p^{-1} \cdot p, b) && \text{Por el Lema 3.4.2} \\ &= \mathsf{tr}^P(\mathsf{refl}_a, b) \\ &\equiv b \end{aligned}$$

□

Corolario 4.2.6. *La contractibilidad se preserva bajo productos.*

Demostración. Aplicación directa de la proposición anterior en el tipo $\lambda(x : A).\, B$. □

Proposición 4.2.7. *Sea B contractible y sea $r : B \to A$ una retracción. Entonces A es contractible.*

Demostración. Sea $b : B$ el centro de contracción de B, y sea $s : A \to B$ la sección correspondiente de r. Mostraremos que $r(b_0)$ es un centro de contracción de A, en efecto, tenemos:

$$\begin{aligned} r(b_0) &= r(s(a)) && \text{Por contractibilidad de } B \\ &= a && \text{Por ser } s \text{ la sección de } r \end{aligned}$$

□

Consideremos el tipo

$$\prod_{x,y:A} \mathsf{isContr}(x =_A y) \equiv \prod_{(x,y:A)} \sum_{(p:x=y)} \prod_{(q:x=y)} (p = q) \tag{4.1}$$

Este tipo indica que, para todo par de elementos de A, el tipo de caminos entre ellos es contractible. En particular, para todo $x, y : A$, tenemos un camino $p : x = y$ entre estos. Para ver que esta condición es también suficiente, necesitaremos un lema previo.

Lema 4.2.8. *Sea A un tipo, $a, x_1, y_1 : A$ y $p : x_1 = x_2, q : a = x_1$ caminos. Entonces*

$$\mathsf{tr}^{\lambda x \to a = x}(p, q) = q \cdot p$$

Demostración. Aplicando inducción a ambos caminos, ambos lados de la igualdad se reducen a refl_a. □

Nótese que este lema corresponde a la acción functorial (covariante) del functor $\mathrm{Hom}(a, -)$ mencionado en el Ejemplo 1.16. Existe un resultado análogo para $\mathsf{tr}^{\lambda x \to x=a}$, correspondiente a $\mathrm{Hom}(-, a)$, el cual no necesitaremos.

Proposición 4.2.9. *Sea A un tipo. El tipo* (4.1) *está habitado sí y solo sí para todo $x, y : A$, tenemos que $x = y$.*

Demostración. Ya mostramos un lado de la implicación, veamos el caso de la recíproca. Por hipótesis, tenemos una función $f : \prod_{(x,y:A)} x = y$. Consideremos la función $g : \prod_{(z:A)} x = z$ definida por $g(z) = f(x, z)$.

Mostraremos que $(g(x))^{-1} \cdot g(y)$ es el centro de contracción. En efecto, dado $p : x = y$ tenemos:

$$\begin{aligned}
(g(x))^{-1} \cdot g(y) &= (g(x))^{-1} \cdot \mathsf{tr}^{\lambda z \to x=z}(p, g(x)) && \text{Por el Lema 3.4.5} \\
&= (g(x))^{-1} \cdot (g(x) \cdot p) && \text{Por el Lema 4.2.8} \\
&= p
\end{aligned}$$

□

Ya habíamos encontrado esta propiedad previamente, en nuestra discusión del axioma K en la sección 3.8, pues este axioma indica justamente que todos los elementos del tipo $x =_A y$ son iguales entre sí. Desde una perspectiva lógica, las proposiciones están caracterizadas por la propiedad de tener un valor de verdad, ser verdaderas o falsas. No es relevante qué elemento en particular tenemos de una proposición, por lo que podemos considerar a todas como iguales. Esto sugiere que el tipo (4.1) previo corresponde a la propiedad de ser una proposición.

Definición 4.2.10. Decimos que un tipo P es una **proposición simple** si es que el siguiente tipo está habitado.

$$\mathsf{isProp}(P) :\equiv \prod_{x,y:A} (x = y)$$

Ejemplo 4.2.11. Todo tipo contractible es una proposición simple. En efecto, sea $(c, p) : \mathsf{isContr}(A)$, entonces para todo $x, y : A$ tenemos $x = c = y$. En particular, el tipo $\mathbf{1}$ es una proposición simple.

Ejemplo 4.2.12. El tipo $\mathbf{0}$ es una proposición simple. Si tenemos $x, y : \mathbf{0}$ tenemos inmediatamente una contradicción, por lo que podemos derivar que el resultado deseado.

Ejemplo 4.2.13. Para toda $f : A \to B$, $\mathsf{isequiv}(f)$ es una proposición simple; este es justo uno de los requisitos que habíamos pedido de una noción correcta de equivalencia homotópica.

Ejemplo 4.2.14. Para todo $m, n : \mathbb{N}$, el tipo $m = n$ es una proposición simple. Puesto que $(m = n) \simeq (\mathsf{code}(m, n))$, es suficiente demostrar que $\mathsf{code}(m, n)$ es una proposición simple. Procedemos por inducción en m y n. Cuando ambos son 0, entonces $\mathsf{code}(0, 0) \equiv \mathbf{1}$, y ya hemos visto que esta es una proposición simple.

Cuando uno es un sucesor y el otro no, $\mathsf{code}(m,n) \equiv \mathbf{0}$, y sabemos que este tipo también es una proposición simple. Finalmente, cuando tenemos $\mathsf{succ}(m)$ y $\mathsf{succ}(n)$, entonces $\mathsf{code}(\mathsf{succ}(m),\mathsf{succ}(n)) \equiv \mathsf{code}(m,n)$, por lo que podemos aplicar la hipótesis de inducción.

Podemos volver a iterar, y preguntarnos qué propiedad satisfacen aquellos tipos cuyos caminos son todos proposiciones simples; veremos que esta es justamente la propiedad de ser un conjunto.

Definición 4.2.15. Un tipo A es un **conjunto** si el siguiente tipo está habitado

$$\mathsf{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q)$$

Topológicamente, esta propiedad nos indica que todo par de caminos con los mismos extremos son homotópicos entre sí. Es decir, tienen el mismo tipo de homotopía que un espacio con la topología discreta. Así, lo único relevante en un conjunto son sus elementos diferentes, lo que captura la noción usual de MC.

Ejemplo 4.2.16. El tipo $\mathbf{0}$ es un conjunto, pues de $x, y : \mathbf{0}$ se deriva un absurdo.

Ejemplo 4.2.17. El tipo $\mathbb{N}$ es un conjunto, pues por el Ejemplo 4.2.14, para todo $m, n : \mathbb{N}$, $m = n$ es una proposición simple.

Nótese que con este tipo, podemos hacer definiciones como el tipo de todos los conjuntos:

$$\mathsf{Set}_{\mathcal{U}} :\equiv \sum_{A:\mathcal{U}} \mathsf{isSet}(A)$$

Es claro que podemos siguer iterando este tipo de preguntas al infinito. Podemos indigar respecto a todas estas propiedades, definiendo la siguiente familia de tipos.

Definición 4.2.18. Definimos ser un n-tipo, a través de recursión en los naturales de la siguiente forma

$$\begin{aligned} \mathsf{is}\text{-}(n-2)\text{-}\mathsf{type} &: \mathcal{U} \to \mathcal{U} \\ \mathsf{is}\text{-}(0-2)\text{-}\mathsf{type}(A) &:\equiv \mathsf{isContr}(A) \\ \mathsf{is}\text{-}(\mathsf{succ}(n)-2)\text{-}\mathsf{type}(A) &:\equiv \prod_{x,y:A} \mathsf{is}\text{-}(n-2)\text{-}\mathsf{type}(A) \end{aligned}$$

Así, los (-2)-tipos son los tipos contractibles, los (-1)-tipos las proposiciones simples, los 0-tipos los conjuntos, etc. La razón de empezar desde el -2 es porque ahora tenemos la siguiente descripción más sugerente: para $n : \mathbb{N}$, el tipo A es un n-tipo si solo tiene m-caminos no-triviales para $m \leq n$. Esta es una noción comúnmente estudiada en topología algebraica, llamada ahí el n-tipo de homotopía.

Dada esta nueva interpretación, nuestra intuición geométrica sugiere una serie de resultados. Mostraremos aquí algunos.

Proposición 4.2.19. *La jerarquía de n-tipos es cumulativa; es decir, si A es un n-tipo, también es un* $\mathsf{succ}(n)$*-tipo.*

Demostración. Procedemos por inducción en n. Para el caso base, sea A un tipo contractible, necesitamos mostrar que los caminos en A son contractibles. Sea $(c, p) : \mathsf{isContr}(A)$, y sean x, y en A. Primero, tenemos que $p(x)^{-1} \cdot (p(y)) : x = y$. Queremos mostrar que este es el centro de contracción. Dado otro camino $q : x = y$, por inducción, es suficiente mostrar que $p(x) \cdot (p(x))^{-1} = \mathsf{refl}_x$, lo que ya hemos visto previamente.

Para el caso general, si A es un $\mathsf{succ}(n)$-tipo, tenemos que $x = y$ es un n-tipo por definición, por lo que por la hipótesis de inducción concluimos que $x = y$ también es un $\mathsf{succ}(n)$-tipo □

Proposición 4.2.20. *Los n-tipos se preservan bajo retracciones. Es decir, para todo* $n \geq -2$*, si* $r : B \to A$ *es una retracción y B es un n-tipo, entonces A también es un n-tipo.*

Demostración. Realizaremos la prueba por inducción en n. El caso base, lo muestra la Proposición 4.2.7. Entonces, supongamos que B es un $\mathsf{succ}(n)$-tipo, y sean $s : A \to B$ la sección de r, y $\epsilon : r \circ s \sim \mathsf{id}$ la homotopía correspondiente.

Dados $a_1, a_2 : A$ arbitrarios, necesitamos mostrar que $a_1 = a_2$ es un n-tipo, pero por la hipótesis de inducción, es suficiente mostrar que tipo es un retracto de $s(a_1) = s(a_2)$. Realizaremos esto a continuación.

Para la sección, tomamos

$$\mathsf{ap}_s : (a_1 = a_2) \to (s(a_1) = s(a_2)),$$

mientras que la retracción $t : (s(a_1) = s(a_2)) \to (a_1 = a_2)$ la definimos por

$$t(q) :\equiv \epsilon_{a_1}^{-1} \cdot r(q) \cdot \epsilon_{a_2}.$$

Para concluir la prueba, necesitamos $t \circ \mathsf{ap}_{\sim}\mathsf{id}$; es decir, que para todo $p : a_1 = a_2$ se tenga que

$$\epsilon_{a_1}^{-1} \cdot r(s(p)) \cdot \epsilon_{a_2} = p,$$

pero esto es una consecuencia del Lema 3.5.2. □

Como una consecuencia de esta proposición, podemos obtener el siguiente resultado sin usar univalencia.

Corolario 4.2.21. *Los n-tipos se preservan bajo equivalencias.*

Demostración. Una la existencia de una equivalencia $A \simeq B$ implica que A es una retracción de B, por lo que si B es un n-tipo, entonces A también lo es. □

Finalmente, presentamos la generalización de la Proposición 4.2.5.

Proposición 4.2.22. *Los n-tipos se preservan bajo sumas dependientes. Es decir, sea para todo* $n \geq -2$*, si A es un n-tipo y la familia de tipos* $B : A \to \mathcal{U}$ *satisface que* $B(a)$ *es un n-tipo para todo* $a : A$*, entonces* $\sum_{(x:A)} B(x)$ *es un n-tipo.*

Demostración. Realizaremos la prueba por inducción en n. El caso base, lo muestra la Proposición 4.2.5. Entonces, supongamos que A es un $\mathsf{succ}(n)$-tipo y $B(a)$ es un $\mathsf{succ}(n)$-tipo para todo $a : A$. Dados $(a_1, b_1), (a_2, b_2) : \sum_{(x:A)} B(x)$ arbitrarios, necesitamos mostrar que $(a_1, b_1) = (a_2, b_2)$ es un n-tipo. Pero por la caracterización de caminos dependientes, tenemos:

$$((a_1, b_1) = (a_2, b_2)) \simeq \sum_{p:a_1=a_2} (\mathsf{tr}^B(p, b_1) = b_2)$$

Pero el tipo de la derecha es un n-tipo por la hipótesis de inducción, por lo que, por el Corolario 4.2.21, el tipo de la izquierda también lo es. □

4.3. Tipos Inductivos Superiores

Como ya mencionamos en la Sección 3.10, los tipos que hemos introducidos hasta ahora, pueden ser considerados como aquellos libremente generados por sus *constructores*, aquellas constantes y funciones que nos permiten formar elementos del tipo. Así, por ejemplo, los naturales están libremente generados por la existencia de un elemento $0 : \mathbb{N}$ y la función $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$.

Podemos expandir este procedimiento para que los constructores incluyan no solo elementos del tipo, sino también caminos entre los elementos. Aquellos tipos que incluyen constructores de caminos son llamados **Tipos Inductivos Superiores**, o **HITs**, por su nombre en inglés, *Higher Inductive Types*. Veamos uno de estos.

Definición 4.3.1. El **intervalo**, denotado I, es el tipo generado por

- un punto $0_I : I$,
- un punto $1_I : I$, y
- un camino $\mathsf{seg} : 0_I = 1_I$.

Este tipo representa un camino abstracto; es decir, el tipo de homotopía de un intervalo cerrado en los reales. Nótese que la definición logra esto sin hacer ninguna mención a toda la maquinaria usual de MC (distancia, bolas, topología, clases de equivalencia, etc).

El principio de recursión de los tipos indica que estos pueden ser mapeados hacia tipos que comparten su estructura interna. En efecto, existe un algoritmo que permite deducir el principio de recursión (y el de inducción), dado los constructores de un tipo. No abordaremos este tema, y daremos estos principios para los tipos que usaremos.

En el caso del tipo del intervalo, el principio de recursión dice que dado un tipo B junto con

- un punto $b_0 : B$,
- un punto $b_1 : B$, y
- un camino $s : b_0 = b_1$,

existe una única función $f : I \to B$ tal que $f(0_I) \equiv b_0, f(1_I) \equiv b_1$, y $f(\mathsf{seg}) = s$. Nótece que las primeras dos igualdades son por definición, mientras que la tercera de estas es una igualdad proposicional.

El principio de inducción de un tipo, como ya hemos mencionado, es el principio de recursión generalizado para familias dependientes. Entonces, para el caso dice que dado una familia de tipos $P : I \to \mathcal{U}$, junto con

- un punto $b_0 : P(0_I)$,
- un punto $b_1 : P(1_I)$, y
- un camino levantado $s : P(b_0) =^P_{\mathsf{seg}} P(b_1)$,

existe una única función $f : \prod_{(x:I)} P(x)$ tal que $f(0_I) \equiv b_0, f(1_I) \equiv b_1$, y $\mathsf{apd}_f(\mathsf{seg}) = s$. Con este principio, podemos demostrar que, como se esperaría, este tipo es contractible.

Proposición 4.3.2. *El tipo I es contractible.*

Demostración. Mostraremos que 0_I es un centro de contracción. Así, necesitamos una función $\prod_{(x:I)}(0_I = x)$. Por el principio de inducción, podemos definir f por:

$$f(0_I) :\equiv \mathsf{refl}_{0_I} : 0_I = 0_I,$$
$$f(1_I) :\equiv \mathsf{seg} \ : 0_I = 1_I.$$

junto con el hecho de que existe un camino $p : \mathsf{refl}_{0_I} =^{\lambda x.\, 0_I = x}_{\mathsf{seg}} \mathsf{seg}$. Pero por el Lema 4.2.8, esto es equivalente a un camino $\mathsf{refl}_{0_I} \cdot \mathsf{seg} = \mathsf{seg}$, el cual tenemos por el Lema 3.2.2. □

Por este resultado, junto con la Proposición 4.2.4, el tipo I es equivalente al tipo $\mathbf{1}$. A pesar de esto, este tipo igual es interesante, pues nos permite realizar otros tipos de argumentos.

Teorema 4.3.3. *Sean $f, g : A \to B$ funciones tales que para todo $x : A$ se tiene que $f(x) = g(x)$. Entonces, $f = g$.*

Demostración. Por recursión, podemos definir una función $\alpha : A \to (I \to B)$ por

$$\alpha(x)(0_I) :\equiv f(x) : B,$$
$$\alpha(x)(1_I) :\equiv g(x) : B.$$

notando que por suposición, tenemos un camino $p : f(x) = g(x)$. Ahora, podemos invertir el orden de las variables, generando una función

$$\beta :\equiv \lambda(x : I).\, \lambda(y : A).\, \alpha(y)(x) : I \to (A \to B).$$

Finalmente, vemos que $\beta(seg) : \beta(0_I) = \beta(1_I)$, pero $\beta(0_I) \equiv f$ y $\beta(1_I) \equiv g$. □

Otro HIT, quizás más interesante, es el círculo, el cual estudiaremos a más detalle en la próxima sección.

Definición 4.3.4. El **círculo**, denotado $\mathbb{S}^1$, es el tipo generado por

- un punto $\mathsf{base} : \mathbb{S}^1$, y
- un camino $\mathsf{loop} : \mathsf{base} = \mathsf{base}$.

Su principio de recursión indica que dados

- un punto $b : B$, y
- un camino $\ell : b = b$,

existe una única función $f : \mathbb{S}^1 \to B$ tal que $f(\mathsf{base}) \equiv b$ y $f(\mathsf{loop}) = \ell$. El principio de inducción dice que dada una familia de tipos $P : \mathbb{S}^1 \to \mathcal{U}$, junto con

- un punto $b : B(\mathsf{base})$, y
- un camino $\ell : b =^B_{\mathsf{loop}} b$,

existe una única función $f : \prod_{(x:\mathbb{S}^1)} B(x)$ tal que $f(\mathsf{base}) \equiv b$ y $\mathsf{apd}_f(\mathsf{loop}) = \ell$.

Similarmente, usando esta misma técnica, podemos definir la esfera.

Definición 4.3.5. La **esfera** $\mathbb{S}^2$ es el tipo generado por

- un punto $\mathsf{base} : \mathbb{S}^2$, y
- un 2-camino $\mathsf{surf} : \mathsf{refl}_{\mathsf{base}} = \mathsf{refl}_{\mathsf{base}}$,

Este tipo de definiciones es análogo a la construcción de algunos espacios como CW complejos. Vamos un último ejemplo.

Definición 4.3.6. Podemos definir al **toro** T^2 como el tipo generado por

- un punto $b : T^2$,
- un camino $p : b = b$,
- otro camino $q : b = b$, y
- un 2-camino camino $t : p \cdot q = q \cdot p$,

Esta definición corresponde al toro como un cuadrado con los lados opuestos identificados. Nótese la mayor elegancia y simplicidad de esta construcción, que evita apelar a nociones técnicas como lo es la topología cociente.

Los HITs no solo sirven para crear espacios topológicos (hasta homotopía), sino también permite la implementación de algunas operaciones sobre tipos, veamos un ejemplo.

Definición 4.3.7. Dado un tipo A, podemos definir su **0-truncación** $\|A\|_0$ como el tipo generado por

- una función $|-|_0 : A \to \|A\|_0$, y
- para todo par de puntos $x, y : \|A\|_0$ y par de caminos $p, q : x = y$, una igualdad $p = q$.

Su principio de recursión indica que dados

- una función $g : A \to B$, y
- para todo par de puntos $x, y : B$ y par de caminos $p, q : x = y$, una igualdad $p = q$,

existe una única función $f : \|A\|_0 \to B$ tal que $f(|x|_0) \equiv g(x)$, para todo $x : A$, y ap_f lleva el camino de $p = q$ en $\|A\|_0$ en el camino especificado $f(p) = f(q)$ en B.

Este tipo efectivamente trivializa los 1-caminos, pues todos se vuelven homotópicos entre sí. Por otro lado, esto es lo único que hace, como uno esperaría, deja a los conjuntos intactos.

Proposición 4.3.8. *Sea A un conjunto, entonces $\|A\|_0 = A$.*

Demostración. Por univalencia, es suficiente demostrar que $\|A\|_0 \simeq A$. Por un lado, tenemos una función $f : \|A\|_0 \to A$ definida por recursión, con $f(|x|_0) \equiv x$. Por otro lado, tomamos $|-|_0 : A \to \|A\|_0$ como la quasi-inversa.

Por definición de f, vemos que $f \circ |-|_0 \sim \mathsf{id}$. Para ver que $|-|_0 \circ f \sim \mathsf{id}$, por unicidad de la función proveniente de recursión, solo necesitamos comprobar que

$$(|-|_0 \circ f \circ |-|_0)(x) = |x|_0,$$

pero nuevamente esto se da por definición de f. □

Existen n-truncaciones para $n \geq -1$, las cuales "truncan" A hacia un n-tipo, pero no abordaremos estas construcciones.

Finalmente, los HITs permiten la definición de tipos de una manera más concisa. Por ejemplo, el tipo de los enteros $\mathbb{Z}$ generalmente se define como un cociente de $\mathbb{N} \times \mathbb{N}$; sin embargo, la siguiente descripción [2] es más útil (y más elegante).

Definición 4.3.9. El tipo de los **enteros** $\mathbb{Z}$ es el tipo generado por

- un elemento $0_\mathbb{Z} : \mathbb{Z}$
- una equivalencia $\mathsf{succ}_\mathbb{Z} : \mathbb{Z} \simeq \mathbb{Z}$,

junto con el hecho de que $\mathbb{Z}$ es un conjunto.

Aquí, la equivalencia $\mathsf{succ}_\mathbb{Z}$ corresponde al hecho de que la función sucesor posee una quasi-inversa, la función predecesor.

El principio de recursión de $\mathbb{Z}$ indica que dados

- un punto $b : B$, y
- una equivalencia $s : B \simeq B$,

existe una única función $\mathbb{Z} \to B$ tal que $f(0_\mathbb{Z}) \equiv b$ y para todo $x : \mathbb{Z}$, $f(\mathsf{succ}_\mathbb{Z}(x)) = s(f(x))$.

El principio de inducción, lo omitimos, pues no lo necesitaremos.

4.4. El grupo fundamental del círculo

El grupo fundamental $\pi_1(X)$ permite generar un objeto algebraico, a partir de las operaciones de concatenación de los 1-caminos en X. Sin embargo, esto se puede generalizar para n-caminos, para $n \geq 1$.

Definición 4.4.1. El espacio de n-caminos cerrados en un tipo A con base $a : A$, es el tipo

$$\Omega^0(A, a) :\equiv (A, a)$$
$$\Omega^{\mathsf{succ}(n)}(A, a) :\equiv \Omega^n\left((a =_A a), \mathsf{refl}_a\right)$$

Definición 4.4.2. Sea $n \geq 1$ y sea A un tipo con $a : A$, definimos el **n-ésimo grupo de homotopía** de A centrado en a como el tipo

$$\pi_n(A, a) :\equiv \left\|\mathsf{pr}_1\left(\Omega^n(A, a)\right)\right\|_0$$

El objetivo de esta sección es demostrar que $\pi_1(\mathbb{S}^1, \mathsf{base}) = \mathbb{Z}$. De hecho, demostraremos algo aún más fuerte, que $(\mathsf{base} = \mathsf{base}) \simeq \mathbb{Z}$. Comencemos con el regreso de las quasi-inversas.

Lema 4.4.3. *Existe una función* $\mathsf{loop}\hat{} : \mathbb{Z} \to (\mathsf{base} = \mathsf{base})$.

Demostración. Por el principio de recursión de los enteros, necesitamos un elemento de $\mathsf{base} = \mathsf{base}$, el cual escogemos que sea $\mathsf{refl}_{\mathsf{base}}$, junto con una equivalencia $s : (\mathsf{base} = \mathsf{base}) \simeq (\mathsf{base} = \mathsf{base})$. Definiremos esta equivalencia como la asociada al siguiente par de quasi-inversas, $f(p) :\equiv p \cdot \mathsf{loop}$ y $g(p) :\equiv p \cdot \mathsf{loop}^{-1}$.

$$f(g(p)) \equiv p \cdot \mathsf{loop}^{-1} \cdot \mathsf{loop} = p$$
$$g(f(p)) \equiv p \cdot \mathsf{loop} \cdot \mathsf{loop}^{-1} = p$$

□

Esta función lleva al $0_{\mathbb{Z}}$ al camino constante, lleva a los enteros positivos n a n concatenaciones de loop, y enteros negativos $-n$ a N concatenaciones de loop^{-1}, de igual manera que en la demostración usual de MC.

Para generar una función $(\mathsf{base} = \mathsf{base}) \to \mathbb{Z}$ utilizaremos el método encode-decode, caracterizando el tipo de caminos $(\mathsf{base} = x)$, para $x : \mathbb{S}^1$.

Definición 4.4.4. Definimos la función $\mathsf{Cover} : \mathbb{S}^1 \to \mathcal{U}$ por recursión en el círculo:

$$\mathsf{Cover}(\mathsf{base}) :\equiv \mathbb{Z}$$
$$\mathsf{Cover}(\mathsf{loop}) := \mathsf{ua}(\mathsf{succ}_{\mathbb{Z}})$$

Así, Cover representa el espacio de cubrimiento usual del círculo. Podemos realizar cálculos con este tipo, por ejemplo, que transportando un natural a lo largo de loop es lo mismo que sumarle uno.

Lema 4.4.5. *Para todo* $x : \mathbb{Z}$*, se tiene que*

$$\mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}, x) = \mathsf{succ}_{\mathbb{Z}}(x)$$

Demostración. Tenemos que

$$\begin{aligned}
\mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}, x) &\equiv \mathsf{tr}^{\mathsf{id}\circ\mathsf{Cover}}(\mathsf{loop}, x) && \text{(Def. de } \mathsf{id}\text{)}\\
&= \mathsf{tr}^{\mathsf{id}}(\mathsf{Cover}(\mathsf{loop}), x) && \text{(Lema 3.4.7)}\\
&= \mathsf{tr}^{\mathsf{id}}(\mathsf{ua}(\mathsf{succ}_{\mathbb{Z}}), x) && \text{(Def. de } \mathsf{Cover}\text{)}\\
&\equiv \mathsf{pr}_1(\mathsf{idtoeqv}(\mathsf{ua}(\mathsf{succ}_{\mathbb{Z}}))) && \text{(Def. de } \mathsf{idtoeqv}\text{)}\\
&= \mathsf{succ}_{\mathbb{Z}} && \text{(Def. de } \mathsf{ua}\text{)}
\end{aligned}$$

□

Procederemos a demostrar ahora la equivalencia $(\mathsf{base} = x) \simeq \mathsf{Cover}(x)$, para todo $x : \mathbb{S}^1$. Tenemos que generar las dos quasi-inversas y mostrar que sus composiciones son homotópicas a la identidad.

Lema 4.4.6. *Tenemos una función*

$$\mathsf{encode} : \prod_{x:\mathbb{S}^1} (\mathsf{base} = x) \to \mathsf{Cover}(x)$$

Demostración. Definimos encode por $\mathsf{encode}(x, p) :\equiv \mathsf{tr}^{\mathsf{Cover}}(p, 0_{\mathbb{Z}})$. □

Intuitivamente, encode toma un camino $\mathsf{base} = x$ y lo lleva al tipo asociado transportando $0_{\mathbb{Z}}$.

Lema 4.4.7. *Tenemos una función*

$$\mathsf{decode} : \prod_{x:\mathbb{S}^1} \mathsf{Cover}(x) \to (\mathsf{base} = x)$$

Demostración. Definimos decode usando el principio de inducción sobre la familia de tipos

$$C :\equiv \lambda(x : \mathbb{S}^1).\ (\mathsf{Cover}(x) \to (\mathsf{base} = x))$$

Para esto, primero necesitamos un elemento de $C(\mathsf{base})$, tomaremos a $\mathsf{loop}^{\wedge}$ como este. Para completar la inducción necesitamos un camino $\mathsf{loop}^{\wedge} =^{C}_{\mathsf{loop}} \mathsf{loop}^{\wedge}$. Pero observamos que

$$\begin{aligned}
&\mathsf{tr}^{(\lambda x.\,(\mathsf{Cover}(x)\to(\mathsf{base}=x)))}(\mathsf{loop}, \mathsf{loop}^{\wedge})\\
&\quad= \mathsf{tr}^{\mathsf{base}=-}(\mathsf{loop}, -) \circ \mathsf{loop}^{\wedge} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}^{-1}, -) && \text{(Lema 3.7.3)}\\
&\quad= (- \centerdot \mathsf{loop}) \circ \mathsf{loop}^{\wedge} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}^{-1}, -) && \text{(Lema 4.2.8)}\\
&\quad= \mathsf{loop}^{\wedge} \circ \mathsf{succ}_{\mathbb{Z}} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}^{-1}, -) && \text{(Def. de } \mathsf{loop}^{\wedge}\text{)}\\
&\quad= \mathsf{loop}^{\wedge} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}, -) \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}^{-1}, -) && \text{(Lema 4.4.5)}\\
&\quad= \mathsf{loop}^{\wedge} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}^{-1} \centerdot \mathsf{loop}, -) && \text{(Lema 3.4.2)}\\
&\quad= \mathsf{loop}^{\wedge} \circ \mathsf{tr}^{\mathsf{Cover}}(\mathsf{refl}_{\mathsf{base}}, -) && \text{(Lema 3.2.4)}\\
&\quad= \mathsf{loop}^{\wedge} && \text{(Def. de } \mathsf{tr}\text{)}
\end{aligned}$$

□

Intuitivamente, decode toma un elemento de $\mathsf{Cover}(x)$ y lo lleva al tipo asociado inducido por $\mathsf{loop}\hat{}$.

Ahora, la primera de las dos homotopía requeridas es inmediata.

Lema 4.4.8. *Para todo* $x : \mathbb{S}^1$ *y* $p : \mathsf{base} = x$*, tenemos que*

$$\mathsf{decode}(x, \mathsf{encode}(x, p)) = p$$

Demostración. Por inducción, podemos asumir que p es $\mathsf{refl}_{\mathsf{base}}$, pero entonces:

$$\begin{aligned}\mathsf{decode}(\mathsf{base}, \mathsf{encode}(\mathsf{base}, \mathsf{refl}_{\mathsf{base}})) &\equiv \mathsf{decode}(\mathsf{base}, \mathsf{tr}^{\mathsf{Cover}}(\mathsf{refl}_{\mathsf{base}}, 0_{\mathbb{Z}})) \\ &\equiv \mathsf{decode}(\mathsf{base}, 0_{\mathbb{Z}}) \\ &\equiv \mathsf{loop}\hat{}(0_{\mathbb{Z}}) \\ &\equiv \mathsf{refl}_{\mathsf{base}}\end{aligned}$$

□

La otra homotopía es más complicada, y necesitaremos uno lemas previos.

Lema 4.4.9. *Sea* $f : \mathbb{Z} \to \mathbb{Z}$ *una función tal que* $f(0_{\mathbb{Z}}) = 0_{\mathbb{Z}}$ *y tal que*

$$f \circ \mathsf{succ}_{\mathbb{Z}} \sim \mathsf{succ}_{\mathbb{Z}} \circ f,$$

entonces $f \sim \mathsf{id}$.

Demostración. Nótese que la función identidad cumple que

$$\mathsf{id} \circ \mathsf{succ}_{\mathbb{Z}} \sim \mathsf{succ}_{\mathbb{Z}} \circ \mathsf{id},$$

por lo que, por la unicidad de la función preveniente de recursión, $f \sim \mathsf{id}$. □

Este resultado indica que la identidad es la única función que conmuta con la función sucesor.

Lema 4.4.10. *Para todo* $x : \mathbb{S}^1$*,* $p : x = \mathsf{base}$ *y* $y : \mathsf{Cover}(x)$ *se tiene que*

$$\mathsf{tr}^{\mathsf{Cover}}(p \centerdot \mathsf{loop}, y) = \mathsf{succ}_{\mathbb{Z}}(\mathsf{tr}^{\mathsf{Cover}}(p, y))$$

Demostración. Por inducción, podemos asumir que p es $\mathsf{refl}_{\mathsf{base}}$. Entonces, se tiene que

$$\mathsf{tr}^{\mathsf{Cover}}(\mathsf{refl}_{\mathsf{base}} \centerdot \mathsf{loop}, y) = \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}, y) = \mathsf{succ}_{\mathbb{Z}}(y).$$

□

Este resultado indica cuando transportamos Cover a lo largo de un camino $p \centerdot \mathsf{loop}$, esto es lo mismo que haber transportado a lo largo de p, y luego sumarle 1.

Lema 4.4.11. *Para todo* $x : \mathbb{Z}$*, se tiene que*

$$\mathsf{encode}(\mathsf{base}, \mathsf{loop}\hat{}(x)) = x$$

Demostración. Por el Lema 4.4.9, es suficiente mostrar que la función

$$h = \lambda(x : \mathbb{Z}).\, \mathsf{encode}(\mathsf{base}, \mathsf{loop}\hat{}(x))$$

satisface que evaluada en $0_\mathbb{Z}$ es $0_\mathbb{Z}$, y que conmuta con $\mathsf{succ}_\mathbb{Z}$. Por un lado, vemos que

$$h(0_\mathbb{Z}) \equiv \mathsf{encode}(\mathsf{base}, \mathsf{refl}_{\mathsf{base}}) \equiv 0_\mathbb{Z}$$

Por otro lado, tenemos que

$$\begin{aligned}
h(\mathsf{succ}_\mathbb{Z}(x)) &\equiv \mathsf{encode}(\mathsf{base}, \mathsf{loop}\hat{}(\mathsf{succ}_\mathbb{Z}(x))) && \text{(Def. } h\text{)}\\
&\equiv \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}\hat{}(\mathsf{succ}_\mathbb{Z}(x)), 0_\mathbb{Z}) && \text{(Def. encode)}\\
&= \mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}\hat{}(x) \cdot \mathsf{loop}, 0_\mathbb{Z}) && \text{(Def. loop}\hat{}\text{)}\\
&= \mathsf{succ}_\mathbb{Z}(\mathsf{tr}^{\mathsf{Cover}}(\mathsf{loop}\hat{}(x), 0_\mathbb{Z})) && \text{(Lema 4.4.10)}\\
&\equiv \mathsf{succ}_\mathbb{Z}(\mathsf{encode}(\mathsf{base}, \mathsf{loop}\hat{}(x))) && \text{(Def. encode)}\\
&\equiv \mathsf{succ}_\mathbb{Z}(h(x)) && \text{(Def. } h\text{)}
\end{aligned}$$

□

Este resultado indica que transportar un el entero asociado a $x : \mathbb{Z}$ concatenaciones del camino loop, es x mismo.

Con todos estos lemas, procedemos a mostrar la última homotopía.

Lema 4.4.12. *Para todo $x : \mathbb{S}^1$ y $p : \mathsf{Cover}(x)$, se tiene que*

$$\mathsf{encode}(x, \mathsf{decode}(x, p)) = p$$

Demostración. Sea

$$C = \lambda(x : \mathbb{S}^1).\, \prod_{p:\mathsf{Cover}x} \mathsf{encode}(x, \mathsf{decode}(x, p)) = p.$$

El lema es equivalente a mostrar una función $\lambda(x : \mathbb{S}^1).\, C(x)$. Probaremos la existencia de esta función usando el principio de inducción del círculo. Primero debemos dar un camino

$$q : \mathsf{encode}(\mathsf{base}, \mathsf{loop}\hat{}(x)) = x,$$

el cual tenemos por el lema previo. Ahora, necesitamos un camino $q =^{C}_{\mathsf{loop}} q$. Pero por el Lema 3.7.4, es suficiente que para todo camino $\beta : a_1 =^{\mathsf{Cover}}_{\mathsf{loop}} a_2$ se tenga que

$$q(a_1) =^{\lambda((x,y) : \sum_{(x:\mathbb{S}^1)} \mathsf{Cover}(x)).\, \mathsf{encode}(x, \mathsf{decode}(x,y)) = y}_{\mathsf{pair}^{=}(\mathsf{loop}, \alpha)} q(a_2)$$

Pero puesto que $\mathbb{Z}$ es un conjunto, ambos caminos son iguales inmediatamente. □

Con todos estos resultados, obtenemos una caracterización del tipo de caminos $\mathsf{base} = x$.

Proposición 4.4.13. *Para todo* $x : \mathbb{S}^1$*, tenemos que* $\mathsf{base} = x \simeq \mathsf{Cover}(x)$.

Aplicando esto en $x :\equiv \mathsf{base}$, por univalencia obtenemos el siguiente corolario.

Corolario 4.4.14. *El tipo* $\mathsf{base} = \mathsf{base}$ *es igual a el tipo de los enteros* $\mathbb{Z}$.

Finalmente, podemos calcular el grupo fundamental del círculo.

Corolario 4.4.15. *El grupo fundamental del círculo es* $\mathbb{Z}$.

Demostración. Tenemos que $\|\mathbb{Z}\|_0 = \mathbb{Z}$ por la proposición 4.3.8. Entonces, como $\mathsf{base} = \mathsf{base}$ es $\mathbb{Z}$, obtenemos $\pi_1(\mathbb{S}^1) = \mathbb{Z}$. □

Capítulo 5

Conclusiones

Hemos introducido la Teoría Homotópica de Tipos, en donde el concepto principal es de tipos y elementos de tipos. Hemos visto que esta teoría puede interpretarse desde tres perspectivas distintas, la lógica, la categórica y la homotópica.

Desde la perspectiva lógica, los tipos representan proposiciones y los elementos representan las pruebas de estas. Desde la perspectiva categórica, los tipos son ∞-grupoides, siendo los objetos los elementos del tipo, y los morfismos las igualdades entre ellos. Desde la perspectiva homotópica, los tipos son tipos de homotopía de espacios topológicos, y las funciones entre tipos corresponden a funciones continuas.

Hemos visto que esta teoría permite formalizar de una forma más abstracta conceptos comunes de la matemática como lo son los grupos, caminos, homotopía, entre otros.

En particular, vemos que podemos definir CW complejos de una forma más elegante, utilizando Tipos Inductivos Superiores, y podemos manejarlos de una manera similar a la usual, lo que nos ha permitido demostrar que $\pi_1(\mathbb{S}^1) = \mathbb{Z}$, a modo de ejemplo.

HoTT todavía es una rama nueva en desarrollo, y existen múltiples grupos que están traduciendo la matemática clásica a este nuevo lenguaje. Creemos que en el futuro HoTT, o una variante similar a esta, será la teoría principal sobre la cual se desarrollará las matemáticas, y la teoría de conjuntos ya no será la herramienta principal para formalizar conceptos.

Finalizamos exhortando al lector a indagar más sobre HoTT, y a contribuir en algunas de las organizaciones que promueven su adopción, como lo son *Agda-Unimath* y *1Lab*.

Apéndice A

Lista de reglas

Reglas A.1. (Reglas básicas de universos)

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}_i\text{-INTRO} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}_i\text{-CUMUL}$$

Reglas A.2. (Reglas básicas de contextos y variables)

$$\frac{}{\cdot \text{ ctx}}\ \text{ctx-EMP} \qquad \frac{x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \ldots, x_n{:}A_n) \text{ ctx}}\ \text{ctx-EXT}$$

$$\frac{(x_1{:}A_1, \ldots, x_n{:}A_n) \text{ ctx}}{x_1{:}A_1, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \text{Vble}$$

donde la regla ctx-EXT tiene la condición adicional de que x_n debe ser distinta a las demás variables $x_1, \ldots, x_{n-1}$, y la regla Vble requiere que $1 \leq i \leq n$. La regla ctx-EMP tiene 0 hipótesis, por lo que siempre es posible aplicarla.

Reglas A.3. (Reglas básicas de igualdades por definición)

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

Reglas A.4. (Reglas de funciones dependientes)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x : A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-FORM}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda(x : A).\, b : \prod_{(x:A)} B}\ \Pi\text{-INTRO}$$

donde la expresión $\prod_{(x:A)} B$ liga a x hasta el final de esta.

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-ELIM}$$

$$\frac{\Gamma, x : A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-COMP}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda(x:A).\, f(x)) : \prod_{(x:A)} B}\ \Pi\text{-UNIQ}$$

Reglas A.5. (Reglas de pares dependientes)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : A \to \mathcal{U}_i}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}_i}\ \Sigma\text{-FORM}$$

$$\frac{\Gamma \vdash B : A \to \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B(a)}{\Gamma \vdash (a, b) : \sum_{(x:A)} B}\ \Sigma\text{-INTRO}$$

donde la expresión $\sum_{(x:A)} B$ liga a x hasta el final de esta.

$$\frac{\begin{array}{c}\Gamma \vdash C : (\sum_{(x:A)} B) \to \mathcal{U}_i \qquad \Gamma \vdash g : \prod_{(a:A)} \prod_{(b:B(a))} C((a,b)) \\ \Gamma \vdash p : \sum_{(x:A)} B\end{array}}{\Gamma \vdash \mathsf{ind}^{C,g,p}_{\sum_{(x:A)} B} : C(p)}\ \Sigma\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma \vdash C : (\sum_{(x:A)} B) \to \mathcal{U}_i \qquad \Gamma \vdash g : \prod_{(a:A)} \prod_{(b:B(a))} C(a,b) \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}^{C,g,(a,b)}_{\sum_{(x:A)} B} \equiv g(a)(b) : C((a,b))}\ \Sigma\text{-COMP}$$

Reglas A.6. (Reglas del tipo $\mathbf{0}$.)

$$\frac{\Gamma\ \mathrm{ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_0}\ \mathbf{0}\text{-FORM} \qquad \frac{\Gamma \vdash C : \mathbf{0} \to \mathcal{U}_0 \qquad \Gamma \vdash z : \mathbf{0}}{\Gamma \vdash \mathsf{ind}^{C,z}_{\mathbf{0}} : C(z)}\ \mathbf{0}\text{-ELIM}$$

Reglas A.7. (Reglas del tipo $\mathbf{1}$.)

$$\frac{\Gamma\ \mathrm{ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\ \mathbf{1}\text{-FORM} \qquad \frac{\Gamma\ \mathrm{ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-INTRO}$$

$$\frac{\Gamma \vdash C : \mathbf{1} \to \mathcal{U}_i \qquad \Gamma \vdash c : C(\star) \qquad \Gamma \vdash a : \mathbf{1}}{\Gamma \vdash \mathsf{ind}^{C,c,a}_{\mathbf{1}} : C(a)}\ \mathbf{1}\text{-ELIM}$$

$$\frac{\Gamma \vdash C : \mathbf{1} \to \mathcal{U}_i \qquad \Gamma \vdash c : C(\star)}{\Gamma \vdash \mathsf{ind}^{C,c,\star}_{\mathbf{1}} \equiv c : C(\star)}\ \mathbf{1}\text{-COMP}$$

Reglas A.8. (Reglas del coproducto.)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A + B : \mathcal{U}_i}\ +\text{-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}(a) : A + B}\ +\text{-INTRO}_1$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{inr}(b) : A + B}\ +\text{-INTRO}_2$$

$$\frac{\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \qquad \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash e : A + B}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,e} : C(e)} \text{ +-ELIM}$$

$$\frac{\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \qquad \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,\mathsf{inl}(a)} \equiv c(a) : C(\mathsf{inl}(a))} \text{ +-COMP}_1$$

$$\frac{\Gamma \vdash C : A + B \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(x:A)} C(\mathsf{inl}(x)) \qquad \Gamma \vdash d : \prod_{(y:B)} C(\mathsf{inr}(y)) \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{ind}_{A+B}^{C,c,d,\mathsf{inr}(b)} \equiv d(b) : C(\mathsf{inr}(b))} \text{ +-COMP}_2$$

Reglas A.9. (Reglas del tipo $\mathbb{N}$.)

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i} \text{ N-FORM}$$

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash 0 : \mathbb{N}} \text{ N-INTRO}_1 \qquad \frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathsf{succ} : \mathbb{N} \to \mathbb{N}} \text{ N-INTRO}_2$$

$$\frac{\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \qquad \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n)) \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,n} : C(n)} \text{ N-ELIM}$$

$$\frac{\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \qquad \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,0} \equiv c_0 : C(n)} \text{ N-COMP}_1$$

$$\frac{\Gamma \vdash C : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash c_0 : C(0) \qquad \Gamma \vdash c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n)) \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,\mathsf{succ}(n)} \equiv c_s(n, \mathsf{ind}_{\mathbb{N}}^{C,c_0,c_s,n}) : C(\mathsf{succ}(n))} \text{ N-COMP}_2$$

Reglas A.10. (Reglas del tipo de identidades.)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i} \text{ =-FORM}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a} \text{ =-INTRO}$$

$$\frac{\Gamma \vdash C : \prod_{(x,y:A)} (x =_A y) \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(z:A)} C(z, z, \mathsf{refl}_z) \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : a =_A b}{\Gamma \vdash \mathsf{ind}_{=_A}^{C,c,a,b,p} : C(a, b, p)} \text{ =-ELIM}$$

$$\frac{\Gamma \vdash C : \prod_{(x,y:A)} (x =_A y) \to \mathcal{U}_i \qquad \Gamma \vdash c : \prod_{(z:A)} C(z, z, \mathsf{refl}_z) \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{ind}_{=_A}^{C,c,a,a,\mathsf{refl}_a} : C(a, a, \mathsf{refl}_a)} \text{ =-COMP}$$

Bibliografía

[1] Andrej Bauer (https://mathoverflow.net/users/1176/andrej-bauer). *What makes dependent type theory more suitable than set theory for proof assistants?* MathOverflow. URL: https://mathoverflow.net/q/376973.

[2] Thorsten Altenkirch y Luis Scoccola. «The Integers as a Higher Inductive Type». en. En: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. Saarbrücken Germany: ACM, jul. de 2020, págs. 67-73. ISBN: 978-1-4503-7104-9. DOI: 10.1145/3373718.3394760. URL: https://dl.acm.org/doi/10.1145/3373718.3394760 (visitado 05-09-2022).

[3] Paolo Aluffi. *Algebra: Chapter 0*. eng. OCLC: 1273675991. S.l.: American Mathematical Society, 2009. ISBN: 978-1-4704-6571-1.

[4] Benno van den Berg y Richard Garner. «Types are weak -groupoids». en. En: *Proceedings of the London Mathematical Society* 102.2 (feb. de 2011), págs. 370-394. ISSN: 00246115. DOI: 10.1112/plms/pdq026. URL: http://doi.wiley.com/10.1112/plms/pdq026 (visitado 05-07-2022).

[5] Ian Chiswell y Wilfrid Hodges. *Mathematical logic*. Oxford texts in logic 3. OCLC: ocm77012114. London ; New York: Oxford University Press, 2007. ISBN: 978-0-19-921562-1 978-0-19-857100-1.

[6] Alonzo Church. «A Set of Postulates for the Foundation of Logic». En: *The Annals of Mathematics* 33.2 (abr. de 1932), pág. 346. ISSN: 0003486X. DOI: 10.2307/1968337. URL: https://www.jstor.org/stable/1968337?origin=crossref (visitado 26-09-2022).

[7] Haskell B. Curry. «Some Philosophical Aspects of Combinatory Logic». en. En: *Studies in Logic and the Foundations of Mathematics*. Vol. 101. Elsevier, 1980, págs. 85-101. ISBN: 978-0-444-85345-5. DOI: 10.1016/S0049-237X(08)71254-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0049237X08712540 (visitado 26-09-2022).

[8] Herbert B. Enderton. *Elements of set theory*. New York: Academic Press, 1977. ISBN: 978-0-12-238440-0.

[9] Walter Feit y John G. Thompson. «Chapter I, from Solvability of groups of odd order, Pacific J. Math, vol. 13, no. 3 (1963)». en. En: *Pacific J. Math.* 13.4 (1963), págs. 775-787. URL: http://dml.mathdoc.fr/item/1103053943.

[10] Daniel P. Friedman y David Thrane Christiansen. *The little typer*. Cambridge, Massuchesetts: The MIT Press, 2018. ISBN: 978-0-262-53643-1.

[11] Kurt Gödel. «Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I». de. En: *Monatshefte für Mathematik und Physik* 38-38.1 (dic. de 1931), págs. 173-198. ISSN: 0026-9255, 1436-5081. DOI: 10.1007/BF01700692. URL: http://link.springer.com/10.1007/BF01700692 (visitado 09-07-2022).

[12] Georges Gonthier y col. «A Machine-Checked Proof of the Odd Order Theorem». En: *Interactive Theorem Proving*. Ed. por David Hutchison y col. Vol. 7998. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, págs. 163-179. ISBN: 978-3-642-39633-5 978-3-642-39634-2. DOI: 10.1007/978-3-642-39634-2_14. URL: http://link.springer.com/10.1007/978-3-642-39634-2_14 (visitado 09-07-2022).

[13] Krzysztof Kapulkin y Peter LeFanu Lumsdaine. «The simplicial model of Univalent Foundations (after Voevodsky)». en. En: *Journal of the European Mathematical Society* 23.6 (mar. de 2021), págs. 2071-2126. ISSN: 1435-9855. DOI: 10.4171/jems/1050. URL: https://ems.press/journals/jems/articles/274693 (visitado 25-06-2022).

[14] Peter LeFanu Lumsdaine. «Weak -Categories from Intensional Type Theory». En: *Typed Lambda Calculi and Applications*. Ed. por Pierre-Louis Curien. Vol. 5608. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, págs. 172-187. ISBN: 978-3-642-02272-2 978-3-642-02273-9. DOI: 10.1007/978-3-642-02273-9_14. URL: http://link.springer.com/10.1007/978-3-642-02273-9_14 (visitado 05-07-2022).

[15] Saunders Mac Lane. *Categories for the working mathematician*. eng. 2nd. ed., Softcover version of original hardcover edition 1998. Graduate texts in mathematics 5. New York, NY: Springer, 2010. ISBN: 978-1-4419-3123-8.

[16] Assia Mahboubi y Enrico Tassi. *Mathematical Components*. en. Version Number: 1.0.1. Zenodo, ene. de 2021. DOI: 10.5281/ZENODO.4457887. URL: https://zenodo.org/record/4457887 (visitado 09-07-2022).

[17] Per Martin-Löf. *Intuitionistic type theory: notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*. eng. Studies in proof theory 1. Napoli: Bibliopolis, 1984. ISBN: 978-88-7088-105-9.

[18] Bengt Nordström, Kent Petersson y Jan M. Smith. *Programming in Martin-Löf's type theory: an introduction*. International series of monographs on computer science 7. Oxford : New York: Clarendon Press ; Oxford University Press, 1990. ISBN: 978-0-19-853814-1.

[19] Benjamin C. Pierce. *Types and programming languages*. Cambridge, Mass: MIT Press, 2002. ISBN: 978-0-262-16209-8.

[20] Emily Riehl. *Category theory in context*. Aurora: Dover modern math originals. OCLC: ocn946461077. Mineola, New York: Dover Publications, 2016. ISBN: 978-0-486-80903-8.

[21] Bertrand Russell. *The Principles of Mathematics*. en. 1.ª ed. Routledge, feb. de 2020. ISBN: 978-0-203-82258-6. DOI: 10.4324/9780203822586. URL: https://www.taylorfrancis.com/books/9781136765742 (visitado 26-09-2022).

[22] Peter Scholze. *Half a year of the Liquid Tensor Experiment: Amazing developments*. Jun. de 2021. URL: https://xenaproject.wordpress.com/2021/06/05/half-a-year-of-the-liquid-tensor-experiment-amazing-developments/.

[23] Peter Scholze. *Liquid Tensor Experiment*. Dic. de 2020. URL: https://xenaproject.wordpress.com/2020/12/05/liquid-tensor-experiment/.

[24] The mathlib Community. «The lean mathematical library». en. En: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. New Orleans LA USA: ACM, ene. de 2020, págs. 367-381. ISBN: 978-1-4503-7097-4. DOI: 10.1145/3372885.3373824. URL: https://dl.acm.org/doi/10.1145/3372885.3373824 (visitado 09-07-2022).

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