Bachelorarbeit

Induktive Typen in der univalenten Typentheorie

angefertigt von Rebecca Salome Fiebiger

30. Juni 2018

für den Studiengang

Mathematik

an der

Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

Betreuer: Dr. Iosif Petrakis (LMU München)

Inhaltsverzeichnis

0 Einleitung

In der heutigen Zeit spielt die Technik eine immer wichtigere Rolle im Alltag. Immer mehr Prozesse werden digitalisiert und programmierten Maschinen, Computern, überlassen.

Die Grundlagen der Mathematik, mit der die meisten Menschen am ehesten vertraut sind, ist die Zermelo-Fraenkel-Mengenlehre mit Auswahlaxiom (kurz: ZFC). Doch so gut sich andere Dinge in eine Programmiersprache übersetzen lassen, so schwer tun wr uns leider immer noch darin, die Axiome der Mengenlehre in ein Computerprogramm zu packen.

Daher hat sich im letzten Jahrhundert ein neuer Zweig der Mathematik aufgetan, indem berühmte Mathematiker wie Per Martin-Löf, Bertrand Russel nund Vladimir Voevodksy begannen, ein neues System zu entwickeln: die sogenannte *Typentheorie*.

Heutzutage wird immer doch daran geforscht und gearbeitet, Aussagen der weitaus bekannteren Mathematik, der Mengenlehre, in die Typentheorie zu übersetzen, was es uns schließlich erleichtern soll, die Mathematik in eine Programmiersprache zu übersetzen.

Um einen kleinen Einblick in die Grundlagen der Typentheorie zu geben, werden wir uns im ersten Abschnitt dieser Arbeit den fundamentalen induktiven Typen widmen, die in der Typentheorie existieren. Wir werden die nötigen Methoden zeigen, wie man mit ihnen arbeiten kann und Aussagen für den jeweiligen Typ beweisen kann. Dabei wollen wir speziell auf das Prinzip der Pfadinduktion für Identitätstypen eingehen.

Im Zweiten Abschnitt dieser Arbeit werden wir uns mit der univalenten Typentheorie beschäftigen, die gerade einmal zwei Axiome benötigt, und auch die Eindeutigkeit induktiver Typen erläutern. Aber wir werden auch zwei Beispiele aufzeigen, welche Gesetze der Mengenlehre in der Typentheorie nicht so einfach gelten. So können wir den Satz vom ausgeschlossenen Dritten hier nicht anwenden und ebenso keinen Widerspruchsbeweis führen.

Am Ende wollen wir noch einen sehr speziellen induktiven Typ vorstellen, den W-Typ, der uns dann auch zu einem Ausblick auf die ganz allgemeine Definition jeglicher induktiver Typen führt.

Nachdem Voevodsky in [7] und Awodey und Warren in [1] mithilfe von Hofmann und Streicher in [2] die intensionale Typentheorie (ITT) von Martin-Löf in [3], [4] und [5] neu erforscht und in der univalenten Typentheorie mithilfe von Homotopie neu interpretiert haben, ist es mit der Zeit immer besser möglich, die Mathematik und die Informatik zu verbinden.

Die univalente Typentheorie (UTT), die wir in dieser Arbeit betrachten werden, ist somit eine Erweiterung der ITT, da die UTT das Univalenzaxiom beinhaltet, das wir in Kapitel 2.1.3 anführen werden. Es sagt aus, dass eine Äquivalenz zwischen zwei Typen auch gleich den Beweis für ihre Gleichheit liefert. Umgekehrt aber hat Martin-Löf durch die induktiven Definitionen auch eine Möglichkeit gegeben, aus dem Beweis der Gleichheit die Äquivalenz zu folgern, da es reicht, dass ein Induktionsprinzip das andere impliziert, wie wir auch noch an einem Beispiel in Kapitel 2.2 sehen werden.

DANKSAGUNG

An dieser Stelle danke ich Herrn Dr. Iosif Petrakis von der LMU München, dass er sich die Zeit genommen hat, mich bei der Erstellung dieser Arbeit zu unterstützen und anzuleiten, und mir die Möglichkeit gegeben hat, mich im Rahmen meiner Bachelorarbeit mit diesem faszinierenden Thema der Mathematik zu beschäftigen.

1 Fundamentale induktive Typen der Homotopietypentheorie

Beginnen wir damit, einen kleinen Einblick in die Homotopietypentheorie zu geben, einer der grundlegenden Sprachen in der Mathematik, wie sie auch in Kapitel 1 von [6] zu finden ist, aus das sich der Inhalt des ersten Kapitels dieser Arbeit bezieht.

Eine vermutlich weitaus bekanntere Sprache ist die Mengentheorie, die vor allem auf die Logik erster Ordnung und Axiome aufbaut, die uns sagen, wie wir mit Mengen arbeiten können. In der Typentheorie allerdings ist die eigentliche Idee, nicht mit Mengen und Propositionen zu arbeiten, sondern mit *Typen*. Wollen wir also eine Aussage beweisen, fassen wir diese Aussage als Typ auf und konstruieren uns ein Objekt diesen Typs, was wir daher als Beweis der Aussage auffassen.

Innerhalb eines Beweises machen wir immer wieder Annahmen, wie zum Beispiel $a : A$ (siehe Kapitel 1.1). Diese zusammen nennen wir den *Kontext*, den Raum, in dem wir arbeiten. Eine geordnete Liste von Typen, in der spätere Annahmen (Typen) auf frühere aufbauen können.

Für all diese Dinge gibt es in der Typentheorie einige Regeln, auf die das System aufbaut, mit denen wir arbeiten können und die sich immer ohne Beweis anwenden lassen. Alle Beweise lassen sich auf diese Regeln zurückführen. Generell verwenden wir in der Typentheorie also kaum Axiome, sondern nur Urteile und Regeln, die alle Informationen über das Verhalten von Mengen und Typen geben. Es gibt also Regeln, die uns sagen, wie wir neue Typen und Objekte dieses Typs konstruieren können, und welche, die uns sagen, wie wir mit ihnen arbeiten können.

Für induktiv definierte Typen gibt es sogenannte Rekursionsprinzipien und Induktionsprinzipien, wobei die Rekursion ein Spezialfall der Induktion ist, nämlich der Fall, dass der Ausdruck, der Typ, in dem wir ein Objekt konstruieren wollen, konstant ist.

Im Allgemeinen hilft uns folgende Tabelle uns etwas unter diesen Typen vorzustellen und gibt uns ein Schema, mit dem wir Aussagen in Typen übersetzen können:

falsch	$\mathbf{0}$
wahr	$\mathbf{1}$
$A \wedge B$	$A \times B$
$A \vee B$	$A + B$
$A \Rightarrow B$	$A \to B$
nicht A	$A \to \mathbf{0}$
für alle x aus A gilt P(x)	$\Pi_{(x:A)} P(x)$
es gibt x aus A, sodass P(x) gilt	$\Sigma_{(x:A)} P(x)$

Abbildung 1

Diese Tabelle werden wir auch vor allem in Kapitel 1.3 verwenden, um Aussagen in Typen zu übersetzen oder uns etwas unter diesen Typen vorstellen zu können.

1.1 Urteile

Urteile sind die Schreibweisen für die fundamentalsten Konstruktionen in der Typentheorie. Bevor wir mit einem Typ arbeiten können, müssen wir erst einmal die Bausteine kennenlernen und wissen, was es heißt, dass ein Objekt einen bestimmten Typ hat und wie wir dies notieren.

Definition 1: $a : A$
Der Ausdruck $a : A$ zeigt an, dass ein Objekt a vom Typ A ist. Informeller können wir auch sagen, *'a ist ein Element von A'*, oder *'A hat einen Beweis'*. In der Typentheorie können wir keine Objekte ohne Typ konstruieren. Haben wir also ein Objekt a mit Typ A konstruiert, können wir auch nicht mehr widerlegen, dass a diesen Typ hat.

Definition 2: $a \equiv b$
Für $a, b : A$ nennen wir dies *Gleichheit nach Definition*, wobei es sich hier nicht um einen Typ handelt, weswegen wir auch einen Ausdruck dieser Form nicht widerlegen können. Gleichheit nach Definition kann dementsprechend auch nicht im *Kontext* eines Beweises auftauchen, da dieser aus Typen besteht. Wollen wir etwas derartiges trotzdem in den Kontext aufnehmen, müssen wir hierfür Substitution verwenden, indem wir einen Ausdruck mit $x : A$ in die Annahmen aufnehmen und dieses x durch ein spezifischeres Element $a : A$ substituieren.

Gleichheit nach Definition gibt es natürlich ebenso für Typen, geschrieben $A \equiv B$.

In jeglichen Ausdrücken binden die beiden Zeichen $\equiv$ und $:$ immer am schwächsten und werden dementsprechend als letztes geklammert.

1.2 Induktive Typen

Nun wollen wir zu den grundlegenden Typen kommen, mit denen wir in der Typentheorie arbeiten können. Das sind vor allem jene Typen aus der Tabelle (vgl. Seite 5), die wir als Bausteine und Übersetzungen verwenden, um Aussagen in Typen zu übersetzen. Noch dazu werden wir uns den Typ der natürlichen Zahlen anschauen.

Zu all diesen Typen gibt es Konstruktoren, die uns sagen, wie wir Objekte diesen Typs bilden können, und Prinzipien, um Aussagen über diesen Typ beweisen zu können.

Als letztes werden wir auf ein wichtiges Prinzip der Typentheorie eingehen, der Pfadinduktion, deren Anwendung wir in Kapitel 1.3 dann näher betrachten werden.

Jedes Objekt, das wir in der Typentheorie verwenden oder konstruieren, muss wie wir in Kapitel 1.1 bereits gesagt haben, einen Typ haben. Dies gilt auch für einen beliebigen Typ A, was uns zu folgender Definition führt.

Definition 3: Universum und Familie
Ein Universum $\mathcal{U}$ ist ein Typ, deren Elemente wiederum Typen sind. Wollen wir also einen Typ A konstruieren, schreiben wir $A : \mathcal{U}$, um anzuzeigen, welchen Typ der Typ A hat, statt nur zu sagen, dass A ein Typ ist.

Eine Familie von Typen $B : A \to \mathcal{U}$ ist eine Sammlung von Typen abhängig vom Typ A. Eine Familie konstanter Typen ist also ein Typ $B : \mathcal{U}$.

Es gibt verschieden große Universen, so liegt der Typ $P \to \mathcal{U}$ in einem größeren Universum $\mathcal{U}'$ als $\mathcal{U}$, also $P \to \mathcal{U} : \mathcal{U}'$. Im weiteren ist aber nur wichtig zu wissen, dass wir uns in der Typentheorie immer in solchen Universen $\mathcal{U}$ bewegen. Auf die verschiedenen Größen werden wir in dieser Arbeit nicht genauer eingehen.

Für einige der kommenden Typen werden wir jeweils zwei Prinzipien erläutern. Einmal das Rekursionsprinzip, das wir nutzen, um eine Aussage zu beweisen, die einem konstanten Typ entspricht. Und einmal das Induktionsprinzip, das wir nutzen, wenn wir eine Aussage, die einer Familie von Typen entspricht, beweisen wollen. Beide Prinzipien bauen darauf auf, dass es reicht, die Aussage für die jeweiligen Konstruktoren des Typs zu zeigen, um sie für alle möglichen Objekte des Typs bewiesen zu haben.

1.2.1 Funktionentypen

Funktionen, die von einem Typ A auf einen Typ B abbilden, haben den Typ $A \to B$. Eine Funktion mit Namen $f : A \to B$ ist ein Ausdruck der Form

$$f(x) \equiv \phi$$

wobei ϕ ein Term ist, der $x : A$ als Variable enthalten kann.

Wollen wir eine Funktion konstruieren, ohne ihr einen Namen zu geben, nutzen wir dafür das λ-Kalkül. Damit sieht eine Funktion dann wie folgt aus:

$$\lambda(x : A).\phi : A \to B \text{ oder } \lambda x.\phi : A \to B$$

So ist $(\lambda x.\phi)(a) \equiv \phi'$, wobei ϕ' aus ϕ gebildet wird, indem alle x in ϕ durch dieses a ersetzt werden.

Eine Funktion $f : A \to B \to C$ ist eine Funktion namens f, die von zwei Variablen $x : A$, $y : B$ abhängt, also ein Ausdruck der Form

$$f(x, y) \equiv \phi$$

mit ϕ als Term, in dem $x : A$ und $y : B$ als Variablen vorkommen können.

Warnung: Nutzen wir hierfür ebenfalls das λ-Kalkül, müssen wir beim Ersetzen von Variablen darauf achten, dass sich dadurch der Ausdruck nicht verändert. Zum Beispiel ist $(\lambda x.\lambda y.y + x)(y) \neq \lambda y.y + y$, da y nicht als freie Variable vorkommt, sondern durch das λ gebunden wurde. Stattdessen gilt:

$$(\lambda x.\lambda y.y + x)(y) = \lambda z.z + y.$$

Wir nennen die gebundene Variable also jetzt z statt y, damit y eine freie Variable wird und eingesetzt werden kann, ohne dass der Ausdruck seine Bedeutung verändert.

1.2.2 Abhängige Funktionentypen

Funktionen, deren Zielmengen von Termen der Startmengen abhängen, nennen wir abhängige Funktionen. Haben wir also einen Typ $A : \mathcal{U}$ und eine Familie von Typen $B : A \to \mathcal{U}$, dann ist

$$\Pi_{(x:A)} B(x) : \mathcal{U}$$

eine abhängige Funktion. Diesen Typ können wir gemäß der Tabelle am Anfang des Kapitels als Aussage auch in *'Für alle x aus A gilt B(x)'* übersetzen. Für $f : \Pi_{(x:A)} B(x)$, definiert durch $f(x) \equiv \phi$ mit $x : A$ und $\phi : B(x)$, gilt also $f(a) : B(a)$. Ist B eine Familie konstanter Typen, dann gilt

$$\Pi_{(x:A)} B \equiv A \to B$$

Der Typ abhängiger Funktionen mit konstanter Zielmenge entspricht also dem bereits bekannten Funktionentyp.

1.2.3 Produkttypen

Der Produkttyp $A \times B : \mathcal{U}$ für $A, B : \mathcal{U}$ ist mit dem aus der Mengentheorie bekannten kartesischen Produkt zu vergleichen und hat ebenfalls Paare (a, b) mit $a : A$ und $b : B$ als Objekte.

Eine Funktion $f : A \times B \to C$ mit $(a, b) : A \times B$ kann auch mithilfe einer weiteren Funktion $g : A \to B \to C$ geschrieben werden, die keine Produkttypen mehr beinhaltet, indem wir diese definieren durch

$$f((a, b)) \equiv g(a)(b)$$

Dies nutzen wir auch um abhängige Funktionen über Produkttypen zu konstruieren, denn für $P : A \times B \to \mathcal{U}$ definieren wir die abhängige Funktion $f : \Pi_{(x:A\times B)}P(x)$ gerne über eine Funktion $g : \Pi_{(x:A)}\Pi_{(y:B)}P((x, y))$ mit $f((x, y)) \equiv g(x)(y)$.

Folgende Prinzipien brauchen wir, um Aussagen über Objekte des Produkttyps $A \times B$ zu beweisen:

Rekursionsprinzip:

$$\mathrm{rec}_{A\times B} : \Pi_{(P:\mathcal{U})}(A \to B \to P) \to A \times B \to P$$

mit der Gleichung

$$\mathrm{rec}_{A\times B}(P, g, (a, b)) \equiv g(a)(b)$$

Induktionsprinzip:

$$\mathrm{ind}_{A\times B} : \Pi_{(P:A\times B\to\mathcal{U})}(\Pi_{(x:A)}\Pi_{(y:B)}P((x, y))) \to \Pi_{(x:A\times B)}P(x)$$

mit der Gleichung

$$\mathrm{ind}_{A\times B}(P, g, (a, b)) \equiv g(a)(b)$$

Für den Produkttyp gibt es zwei wichtige Funktionen, mithilfe derer auch gezeigt werden kann, dass jedes Objekt des Produkttyps $A \times B$ tatsächlich ein Paar (a, b) mit $a : A$ und $b : B$ ist. Das sind die Projektionsfunktionen, die wir definieren als

$$pr_1 : A \times B \to A \text{ mit } pr_1((a, b)) \equiv a$$
$$pr_2 : A \times B \to B \text{ mit } pr_2((a, b)) \equiv b$$

Diese Funktionen geben uns also die erste oder zweite Komponente des Paares zurück, sodass wir alle Objekte $x : A \times B$ auch schreiben können als

$$x \equiv (pr_1(x), pr_2(x))$$

1.2.4 Abhängige Paartypen

Bei Objekten von abhängigen Paartypen $\Sigma_{(x:A)}B(x)$ mit einem Typ $A : \mathcal{U}$ und einer Familie von Typen $B : A \to \mathcal{U}$ handelt es sich um Paare, deren zweite Komponente von der ersten abhängt. Somit ist ein Objekt diesen Typs ein Paar $(a, b) : \Sigma_{(x:A)}B(x)$ mit $a : A$ und $b : B(a)$. Einen Ausdruck dieser Form können wir auch gemäß der Tabelle am Anfang dieses Kapitels in *'Es gibt x aus A, sodass B(x) gilt'* übersetzen. Wenn B konstant ist, gilt hier:

$$\Sigma_{(x:A)}B(x) = A \times B,$$

da $b : B(a) \equiv B$ gilt und somit $(a, b) : A \times B$ gilt mit $a : A$ und $b : B$.

Folgende sind die nötigen Prinzipien, um mit abhängigen Paartypen arbeiten zu können:

Rekursionsprinzip:

$$\mathrm{rec}_{\Sigma_{(x:A)}B(x)} : \Pi_{(P:\mathcal{U})}(\Pi_{(x:A)}B(x) \to P) \to \Sigma_{(x:A)}B(x) \to P$$

mit der Gleichung

$$\mathrm{rec}_{A\times B}(P, g, (a, b)) \equiv g(a)(b)$$

Induktionsprinzip:

$$\mathrm{ind}_{\Sigma_{(x:A)}B(x)} : \Pi_{(P:(\Sigma_{(x:A)}B(x))\to\mathcal{U})}(\Pi_{(a:A)}\Pi_{(b:B(a))}P((a, b))) \to \Pi_{(p:\Sigma_{(x:A)}B(x))}P(p)$$

mit der Gleichung

$$\mathrm{ind}_{\Sigma_{(x:A)}B(x)}(P, g, (a, b)) \equiv g(a)(b)$$

1.2.5 Koprodukttypen

Für $A, B : \mathcal{U}$ können wir durch $A + B : \mathcal{U}$ den Koprodukttyp definieren. Dies entspricht in der Mengentheorie sozusagen der Vereinigung zweier Mengen. Um Objekte diesen Typs zu konstruieren, gibt es einmal $inl(a) : A + B$ und $inr(b) : A + B$, wobei $inl(a)$ einem $a : A$ und $inr(b)$ einem $b : B$ entspricht. Also zeigt $inl(x)$ an, dass $x : A$ gilt, und $inr(y)$ zeigt an, dass $y : B$ gilt.

Eine unabhängige Funktion $f : A + B \to C$ lässt sich auch durch zwei weitere Funktionen $g_0 : A \to C$ und $g_1 : B \to C$ definieren, indem wir f über die Gleichungen

$$f(inl(a)) \equiv g_0(a)$$
$$f(inr(b)) \equiv g_1(b)$$

definieren.

Für den Koprodukttyp können wir folgende Prinzipien nutzen:

Rekursionsprinzip:

$$\mathrm{rec}_{A+B} : \Pi_{(P:\mathcal{U})}(A \to P) \to (B \to P) \to A + B \to P$$

mit den Gleichungen

$$\mathrm{rec}_{A+B}(P, g_0, g_1, inl(a)) \equiv g_0(a)$$
$$\mathrm{rec}_{A+B}(P, g_0, g_1, inr(b)) \equiv g_1(b)$$

Induktionsprinzip:

$$\mathrm{ind}_{A+B} : \Pi_{(P:A+B\to\mathcal{U})}(\Pi_{(a:A)}P(inl(a))) \to (\Pi_{(b:B)}P(inr(b)))) \to \Pi_{(x:A+B)}P(x)$$

1.2.6 Der leere Typ

Der leere Typ $\mathbf{0} : \mathcal{U}$ enthält tatsächlich keine Objekte, was dazu führt, dass wir Funktionen $f : \mathbf{0} \to C$ mit $C : \mathcal{U}$ immer ohne Definition verwenden können, da es keine Objekte in $\mathbf{0}$ gibt, auf denen wir f definieren müssten.

Für den leeren Typ gibt es folgende Prinzipien:

Rekursionsprinzip: $\mathrm{rec}_{\mathbf{0}} : \Pi_{(P:\mathcal{U})}\mathbf{0} \to P$

Induktionsprinzip: $\mathrm{ind}_{\mathbf{0}} : \Pi_{(P:\mathbf{0}\to\mathcal{U})}\Pi_{(z:\mathbf{0})}P(z)$

Der leere Typ hilft uns die Negation eines Typs zu definieren:

$$\neg A \equiv A \to \mathbf{0}.$$

Damit können wir nun auch wie folgt beweisen, dass $\mathbf{0}$ kein Objekt enthält:

Beweis. Um zu zeigen, dass ein Typ A kein Objekt enthält, reicht es offensichtlich zu zeigen, dass $\neg A$ ein Objekt enthält. Für $A \equiv \mathbf{0}$ müssen wir also zeigen, dass $\neg\mathbf{0}$ ein Element enthält. Dafür gilt nun

$$\neg\mathbf{0} \equiv \mathbf{0} \to \mathbf{0},$$

und wir wissen, dass die Identitätsfunktion $\mathrm{id}_{\mathbf{0}}$ diesen Typs ist. Damit gibt es ein Objekt des Typs $\neg\mathbf{0}$ und somit kein Objekt des Typs $\mathbf{0}$, woraus wir folgern können, dass der leere Typ $\mathbf{0}$ keine Elemente enthält.

□

1.2.7 Der Einheitstyp

Der Typ $\mathbf{1} : \mathcal{U}$ besitzt sozusagen nur ein Element $\star : \mathbf{1}$.

Dies bedeutet, dass wir zeigen können, dass jedes Element $x : \mathbf{1}$ gleich $\star$ ist. Darauf werden wir in Kapitel 1.2.11 noch einmal genauer eingehen, wenn wir auch definieren, was es bedeutet, dass zwei Terme $x, y : A$ für einen Typ $A : \mathcal{U}$ gleich sind.

Definieren wir also eine Funktion auf dieser Menge, könnten wir sie genauso gut ohne diesen Typ definieren, da $\mathbf{1}$ nur eine Element hat, auf dem wir die Funktion definieren müssen. Wir können die Funktion ebenso direkt als konstante Funktion definieren, ohne $\star$ darauf abbilden zu lassen.

Für den Einheitstyp gibt es folgende Prinzipien:

Rekursionsprinzip:

$$\mathrm{rec}_{\mathbf{1}} : \Pi_{(P:\mathcal{U})} P \to \mathbf{1} \to P$$

mit der Gleichung

$$\mathrm{rec}_{\mathbf{1}}(P, c, \star) \equiv c \text{ für } c : P$$

Induktionsprinzip:

$$\mathrm{ind}_{\mathbf{1}} : \Pi_{(P:\mathbf{1}\to\mathcal{U})} P(\star) \to \Pi_{(x:\mathbf{1})} P(x)$$

mit der Gleichung

$$\mathrm{ind}_{\mathbf{1}}(P, c, \star) \equiv c \text{ für } c : P$$

1.2.8 Der Boolean-Typ

Der Name Boolean erinnert die meisten sofort an Wahrheitswerte, an die Werte 'wahr' und 'falsch'. An dieser Stelle möchte ich anmerken, dass der Typ $\mathbf{2} : \mathcal{U}$ in der Typentheorie nicht die Wahrheitswerte repräsentiert, auch wenn wir den Objekten trotzdem bei der Übersetzung dieselbe Bedeutung geben, wenn wir Typen, die den Boolean-Typ beinhalten, in Aussagen übersetzen. So enthält $\mathbf{2}$ genau 2 Objekte: $0_{\mathbf{2}}, 1_{\mathbf{2}} : \mathbf{2}$.

Funktionen $f : \mathbf{2} \to C$ mit $C : \mathcal{U}$ müssen also auf diesen beiden Werten definiert werden mit $f(0_{\mathbf{2}}) \equiv c_0$ und $f(1_{\mathbf{2}}) \equiv c_1$ für geeignete $c_0, c_1 : C$.

Die dazugehörigen Prinzipien lauten wie folgt:

Rekursionsprinzip:

$$\mathrm{rec}_{\mathbf{2}} : \Pi_{(P:\mathcal{U})} P \to P \to \mathbf{2} \to P$$

mit den Gleichungen

$$\mathrm{rec}_{\mathbf{2}}(P, c_0, c_1, 0_{\mathbf{2}}) \equiv c_0 \text{ für } c_0 : P$$
$$\mathrm{rec}_{\mathbf{2}}(P, c_0, c_1, 1_{\mathbf{2}}) \equiv c_1 \text{ für } c_1 : P$$

Induktionsprinzip:

$$\mathrm{ind}_{\mathbf{2}} : \Pi_{(P:\mathbf{2}\to\mathcal{U})} P(0_{\mathbf{2}}) \to P(1_{\mathbf{2}}) \to \Pi_{(x:\mathbf{2})} P(x)$$

mit den Gleichungen

$$\mathrm{ind}_{\mathbf{2}}(P, c_0, c_1, 0_{\mathbf{2}}) \equiv c_0 \text{ für } c_0 : P$$
$$\mathrm{ind}_{\mathbf{2}}(P, c_0, c_1, 1_{\mathbf{2}}) \equiv c_1 \text{ für } c_1 : P$$

Mit diesem Induktionsprinzip und den bisher bereits bekannten Typen können wir nun auch beweisen, dass $0_{\mathbf{2}}$ und $1_{\mathbf{2}}$ tatsächlich die einzigen Objekte in dem Typ $\mathbf{2}$ sind.

Beweis. Wollen wir diese Aussage in einen Typ umformen, erhalten wir also den Typ

$$P \equiv \Pi_{(x:\mathbf{2})}(x =_{\mathbf{2}} 0_{\mathbf{2}}) + (x =_{\mathbf{2}} 1_{\mathbf{2}})$$

Seien also

$$c_0 : P(1_{\mathbf{2}}) \equiv (0_{\mathbf{2}} =_{\mathbf{2}} 0_{\mathbf{2}}) + (0_{\mathbf{2}} =_{\mathbf{2}} 1_{\mathbf{2}})$$
$$c_1 : P(0_{\mathbf{2}}) \equiv (1_{\mathbf{2}} =_{\mathbf{2}} 0_{\mathbf{2}}) + (1_{\mathbf{2}} =_{\mathbf{2}} 1_{\mathbf{2}}).$$

Diese Objekte existieren offensichtlich und damit folgt

$$\mathrm{ind}_{\mathbf{2}}(P, c_0, c_1, 0_{\mathbf{2}}) \equiv c_0$$
$$\mathrm{ind}_{\mathbf{2}}(P, c_0, c_1, 1_{\mathbf{2}}) \equiv c_1.$$

Hiermit haben wir bewiesen, dass $0_{\mathbf{2}}$ und $1_{\mathbf{2}}$ die einzigen Objekte im Typ $\mathbf{2}$ sind.

□

1.2.9 Die natürlichen Zahlen

Die natürlichen Zahlen $\mathbb{N}$ sind vor allem in der Dezimalschreibweise 0, 1, 2, 3, ... bekannt. Konstruieren können wir sie durch das Element $0 : \mathbb{N}$ und eine Abbildung $succ : \mathbb{N} \to \mathbb{N}$, definiert durch: $succ(0) \equiv 1$, $succ(1) \equiv 2$, ...

Damit lassen sich unabhängige Funktionen $f : \mathbb{N} \to C$ einfach definieren durch:

$$\begin{aligned} f(0) &\equiv c_0 \\ f(succ(n)) &\equiv c_s(n, f(n)) \end{aligned}$$

für $c_0 : P$ und eine Funktion $c_s : \mathbb{N} \to P \to P$. Diese Vorgehensweise zum Definieren einer Funktion auf $\mathbb{N}$ nennen wir auch primitive Rekursion.

Folgende Prinzipien gibt es für den Typ der natürlichen Zahlen, um Aussagen zu beweisen:

Rekursionsprinzip:

$$\mathrm{rec}_{\mathbb{N}} : \Pi_{(P:\mathcal{U})} P \to (\mathbb{N} \to P \to P) \to \mathbb{N} \to P$$

mit den Gleichungen

$$\begin{aligned} \mathrm{rec}_{\mathbb{N}}(P, c_0, c_s, 0) &\equiv c_0 \\ \mathrm{rec}_{\mathbb{N}}(P, c_0, c_s, succ(n)) &\equiv c_s(n, \mathrm{rec}_{\mathbb{N}}(P, c_0, c_s, n)) \end{aligned}$$

für $c_0 : P$ und $c_s : \mathbb{N} \to P \to P$.

Induktionsprinzip:

$$\mathrm{ind}_{\mathbb{N}} : \Pi_{(P:\mathbb{N}\to\mathcal{U})} P(0) \to (\Pi_{(n:\mathbb{N})} P(n) \to P(succ(n))) \to \Pi_{(n:\mathbb{N})} P(n)$$

mit den Gleichungen

$$\begin{aligned} \mathrm{ind}_{\mathbb{N}}(P, c_0, c_s, 0) &\equiv c_0 \\ \mathrm{ind}_{\mathbb{N}}(P, c_0, c_s, succ(n)) &\equiv c_s(n, \mathrm{ind}_{\mathbb{N}}(P, c_0, c_s, n)) \end{aligned}$$

für $c_0 : P$ und $c_s : \Pi_{(n:\mathbb{N})} P(n) \to P(succ(n))$.

1.2.10 Identitätstypen und Pfadinduktion

Jetzt haben wir bereits einige wichtige Typen kennengelernt, mit denen wir in der Typentheorie arbeiten. Jetzt kann es aber natürlich genauso sein, dass wir einfach mit irgendeinem Typ $A : \mathcal{U}$ arbeiten wollen, nicht nur mit einem der bisher explizit definierten Typen. Zum Beispiel haben wir schon gesehen, dass sich Aussagen in Typen übersetzen lassen, und wenn sich diese Aussagen nicht auf einen bestimmten Typ beziehen, brauchen wir ein neues Prinzip, um auch dann diese Aussagen beweisen zu können, um zu zeigen, dass ein Element diesen Typs existiert. Dafür gibt es das Prinzip der Pfadinduktion, wofür wir zuerst einmal noch eine weitere Definition benötigen.

Definition 4: $a =_A b$
Bei diesem Ausdruck handelt es sich für $a, b : A$ und $A : \mathcal{U}$ um den Ausdruck, den wir *propositionale Gleichheit* nennen. Wir sagen also a und b sind propositional gleich, wenn es ein Objekt $p : a =_A b$ existiert. So ein Objekt nennen wir auch Pfad mit Namen p, der a als Startpunkt und b als Endpunkt besitzt. Propositionale Gleichheit ist, anders als Gleichheit nach Definition, auch ein Typ.

Wie bereits bei der Definition des Einheitstyps **1** gesagt, soll folgende Skizze uns nun veranschaulichen, was propositionale Gleichheit ist und was es für einen Typ wie den Einheitstyp **1** bedeutet:

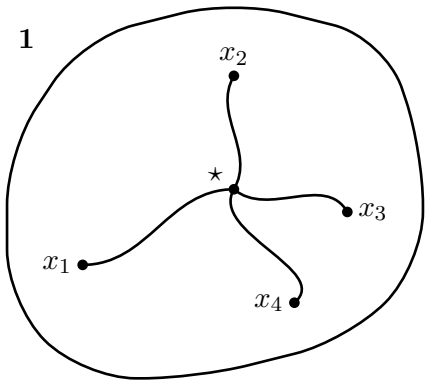

Abbildung 2

Jedes Element $x : \mathbf{1}$ ist also durch einen Pfad mit $\star : \mathbf{1}$ verbunden, es gilt also $\Pi_{(x:\mathbf{1})} x =_{\mathbf{1}} \star$, was wir so interpretieren, dass der Einheitstyp sozusagen nur $\star$ als Element enthält.

Ein spezielles Objekt des Typs der propositionalen Gleichheit, das wir beim Prinzip der Pfadinduktion brauchen, ist das Reflexivitätselement $\mathrm{refl}_x : x =_A x$ für $x : A$, also der konstante Pfad von x nach x.

Propositionale Gleichheit hilft uns auch die Ungleichheit zweier Objekte x und y zu definieren, indem gilt:

$$(x \neq_A y) \equiv \neg(x =_A y).$$

Zwei Objekte sind also nicht propositional gleich, wenn es keinen Pfad vom einen zum anderen gibt. Allerdings hilft uns dies in der Typentheorie nicht zu zeigen, dass zwei Objekte gleich sind. Denn Gleichheit zweier Objekte kann nicht dadurch gezeigt werden, dass sie nicht ungleich sind.

Diese propositionale Gleichheit bringt uns zu folgender Aussage:

Sei $P : A \to \mathcal{U}$ eine Familie von Typen. Dann gibt es eine abhängige Funktion

$$f : \Pi_{(x,y:A)}\Pi_{(p:x=_A y)}P(x) \to P(y)$$

mit

$$f(x, x, \mathrm{refl}_x) \equiv \mathrm{id}_{P(x)},$$

das heißt, die Gleichheit von Objekten im Typ A wird von Familien von Typen über den Typ A anerkannt.

Sei also zum Beispiel P eine Eigenschaft, die x besitzt, und seien x und y propositional gleich, dann muss also auch y diese Eigenschaft P besitzen. Aussagen wie diese werden wir in Kapitel 1.3 noch genauer betrachten.

Das nächste Induktionsprinzip, das wir nun anführen, nennt sich Pfadinduktion. Dieses baut genau auf diesen Typ der propositionalen Gleichheit auf, dem sogenannten *Identitätstyp*. Es sagt uns, wenn wir eine Aussage für einen Pfad $\mathrm{refl}_x : x =_A x$ beweisen können, dann können wir von diesem Pfad auch auf alle anderen Pfade $p : x =_A y$ schließen und somit die Aussage auch für alle $x, y : A$ und $p : x =_A y$ als bewiesen ansehen, da wir den Pfad $\mathrm{refl}_x : x =_A x$ immer zu einem Pfad $p : x =_A y$ verändern können.

Formal bringt uns dies zu dem folgenden Induktionsprinzip für Identitätstypen: Sei $D : \Pi_{(x,y:A)}(x =_A y) \to \mathcal{U}$ eine Familie von Typen und

$$d : \Pi_{(x:A)}D(x, x, \mathrm{refl}_x)$$

eine abhängige Funktion. Dann gibt es auch eine abhängige Funktion

$$f : \Pi_{(x,y:A)}\Pi_{(p:x=_A y)}D(x, y, p)$$

mit

$$f(x, x, \mathrm{refl}_x) \equiv d(x).$$

Wollen wir dies nun mit der uns bisher bereits bekannten Schreibweise für Induktionsprinzipien formulieren, bekommen wir folgendes Prinzip:

Pfadinduktion:

$$\mathrm{ind}_{=_A} : \Pi_{(D:\Pi_{(x,y:A)}(x=_A y)\to\mathcal{U})}(\Pi_{(x:A)}D(x, x, \mathrm{refl}_x)) \to \Pi_{(x,y:A)}\Pi_{(p:x=_A y)}D(x, y, p)$$

mit der Gleichung

$$\mathrm{ind}_{=_A}(D, d, x, x, \mathrm{refl}_x) \equiv d(x)$$

Haben wir also eine Aussage über Pfade, die wir beweisen wollen, nutzen wir hierfür das Prinzip der Pfadinduktion. Mit diesem reicht es also zu zeigen, dass die Aussage für den konstanten Pfad $\mathrm{refl}_x : x =_A x$ für $x : A$ gilt, um auf alle anderen Pfade dieser Art zwischen zwei Objekten schließen zu können.

1.3 Aussagen über Pfade

Wir haben jetzt schon alle wichtigen Typen mit samt ihren Rekursions- und Induktionsprinzipien bereits kennen gelernt und können somit Aussagen in die entsprechenden Typen übersetzen und beweisen. Am wichtigsten für uns ist dabei das Prinzip der Pfadinduktion, auf das wir als nächstes genauer eingehen wollen, in dem wir es an einigen Aussagen anwenden.

Sätze und Beweise dieses Kapitels sind großteils auch in Kapitel 2.1 bis Kapitel 2.3 von [6] zu finden.

Der konstante Pfad mit demselben Anfangs- und Endpunkt ist nennt sich das Reflexivitätselement $\mathrm{refl}_x : x = x$. Doch auch andere Eigenschaften wie Symmetrie und Transitivität besitzen solche Pfade, die wir mit folgenden zwei Lemmas einführen wollen.

Lemma 1.3.1: Symmetrie der Pfade
Das Inverse eines Pfades $p : x =_A y$ ist eine Funktion

$$p^{-1} : (x = y) \to (y = x),$$

also eine Abbildung $p \to p^{-1}$, die für jeden Typ $A : \mathcal{U}$ und alle $x, y : A$ existiert und für die für alle $x : A$ gilt: $\mathrm{refl}_x^{-1} = \mathrm{refl}_x$.

Beweis. Die Existenz des Inversen, also dieser Funktion $p \to p^{-1}$, wollen wir nun beweisen. Dafür wollen wir die Aussage zuerst einmal in einen Typ übersetzen, bevor wir mithilfe der Pfadinduktion zeigen, dass ein Objekt dieses Typs existiert.

Mithilfe der Tabelle am Anfang des Kapitels können wir die Aussage in folgenden Typ übersetzen:

$$\Pi_{(A:\mathcal{U})}\Pi_{(x,y:A)}(x = y) \to (y = x)$$

Sei also $A : \mathcal{U}$ beliebig, aber fest und sei $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen, die wir definieren durch

$$D(x, y, p) \equiv (y = x).$$

Nach dem Prinzip der Pfadinduktion brauchen wir jetzt also nur zeigen, dass für dieses beliebige $A : \mathcal{U}$ ein Objekt des Typs $D(x, x, \mathrm{refl}_x)$ für $x : A$ existiert. Nach Definition unseres Typs gilt

$$D(x, x, \mathrm{refl}_x) \equiv (x = x)$$

$x = x$ ist bekanntlich der Typ unseres Refexivitätselements refl_x, also gibt es ein Objekt mit dem gewünschten Typ und somit können wir uns die abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_x : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

eindeutig konstruieren, wodurch nach dem Prinzip der Pfadinduktion nun also ein Objekt

$$\mathrm{ind}_{=A}(D, d, x, y, p) : (y = x)$$

für alle $p : x = y$ existiert.

Also gibt es ein Objekt des angegebenen Typs und somit gilt die Aussage als bewiesen, das Inverse existiert und liefert uns ebenso $\mathrm{refl}_x^{-1} \equiv \mathrm{refl}_x$.

□

Lemma 1.3.2: Transitivität der Pfade
Die Verknüpfung, auch Komposition, zweier Pfade $p : x = y$ und $q : y = z$, also eine Abbildung $p \to q \to p \cdot q$ der Form

$$(x = y) \to (y = z) \to (x = z)$$

existiert für alle Typen $A : \mathcal{U}$ und $x, y, z : A$ und für alle $x : A$ gilt dann

$$\mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x.$$

Folgende Skizze soll die Aussage dieses Lemma verdeutlichen:

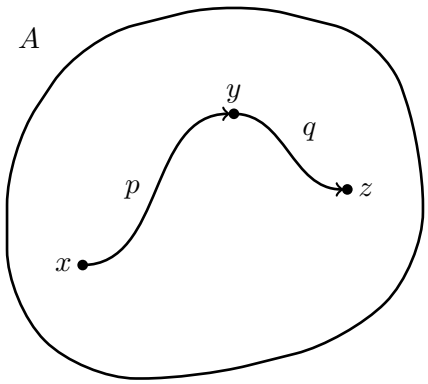

Abbildung 3

Wir gehen also zuerst entlang des Pfades von x nach y und dann von dort aus weiter den nächsten Pfad entlang nach z. Insgesamt sind wir also einen Pfad von x nach z entlang gegangen.

Kommen wir nun zum Beweis der Existenz der Funktion, die uns die Komposition zweier Pfade beschreibt.

Beweis. Diesmal haben wir zwei Pfade p und q, für die wir die Pfadinduktion anwenden wollen, denn letztlich wollen wir ein konkretes Objekt konstruieren. Würden wir nur über p oder q induzieren, hätten wir den jeweils anderen Pfad immer noch stellvertretend für alle Pfade zwischen den beiden Variablen innerhalb unseres Objekts, sodass wir nicht nur ein einzelnes, genau definiertes Objekt unseres Typs erhalten, sondern einen weiteren Typ, der vom Typ des gesuchten Typs ist.

Beginnen wir also mit der Induktion über den ersten Pfad p. Sei dafür jetzt $D_1 : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen. Übersetzen wir dafür die Aussage in einen Typ, erhalten wir für D_1 die Definition

$$D_1(x, y, p) \equiv \Pi_{(z:A)}\Pi_{(q:y=z)}(x = z).$$

Damit folgt

$$D_1(x, x, \mathrm{refl}_x) \equiv \Pi_{(z:A)}\Pi_{(q:x=z)}(x = z).$$

An dieser Stelle müssen wir eine weitere Induktion über q einbauen, indem wir die Familie von Typen $D_2 : \Pi_{(x,z:A)}(x = z) \to \mathcal{U}$ definieren durch

$$D_2(x, z, q) \equiv (x = z)$$

Damit gilt

$$D_2(x, x, \mathrm{refl}_x) \equiv (x = x)$$

und dieser Typ beinhaltet wieder das Reflexivitätselement refl_x.

Somit können wir uns mit zweimaliger Pfadinduktion ein Objekt des gesuchten Typs konkret definieren durch

$$d \equiv \lambda x.\mathrm{refl}_x : \Pi_{(x:A)} D_2(x, x, \mathrm{refl}_x)$$

und die Existenz des Typs

$$\Pi_{(x,y,z:A)}(x = y) \to (y = z) \to (x = z)$$

der Komposition zweier Pfade p und q als bewiesen ansehen.

□

Um mit dem Inversen und der Komposition von Pfaden auch arbeiten zu können, wollen wir noch ein paar Aussagen darüber beweisen.

Lemma 1.3.3: *Für $A : \mathcal{U}$ und $x, y, z, w : A$ mit $p : x = y$, $q : y = z$ und $r : z = w$ gilt:*

a) $p \cdot \mathrm{refl}_y = p$ *und* $\mathrm{refl}_x \cdot p = p$

b) $p \cdot p^{-1} = \mathrm{refl}_x$ *und* $p^{-1} \cdot p = \mathrm{refl}_y$

c) $p = (p^{-1})^{-1}$

d) $(p \cdot q) \cdot r = p \cdot (q \cdot r)$

Beweis. Alle vier Aussagen können wir uns anhand ähnlichen Skizzen wie Abbildung 3 (vgl. Seite 18) ziemlich gut vorstellen. Kommen wir nun also zum formalen Beweis dieser Aussagen, wofür wir wieder Pfadinduktion verwenden.

a) Beginnen wir wieder damit, dass $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ für ein $A : \mathcal{U}$ eine Familie von Typen sei, definiert durch

$$D(x, y, p) \equiv (p \cdot \mathrm{refl}_y = p).$$

Dann gilt hier $D(x, x, \mathrm{refl}_x) \equiv (\mathrm{refl}_x \cdot \mathrm{refl}_x = \mathrm{refl}_x)$.

Wir wissen bereits, dass $\mathrm{refl}_x \cdot \mathrm{refl}_x = \mathrm{refl}_x$ gilt, womit folgt, dass

$$D(x, x, \mathrm{refl}_x) \equiv (\mathrm{refl}_x = \mathrm{refl}_x).$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_x} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x).$$

Damit liefert uns das Prinzip der Pfadinduktion nun für alle $p : x = y$ ein Objekt

$$\mathrm{ind}_{=A}(D, d, x, y, p) : (p \cdot \mathrm{refl}_y = p)$$

Also haben wir ein Objekt im gegebenen Typ gefunden und somit ist die Aussage bewiesen.

Die zweite Gleichung $\mathrm{refl}_x \cdot p = p$ kann man analog zeigen, weswegen wir den Beweis dafür hier auslassen werden.

b) Sei $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ wieder eine Familie von Typen für ein $A : \mathcal{U}$ definiert durch

$$D(x, y, p) \equiv (p \cdot p^{-1} = \mathrm{refl}_x)$$

Dann gilt $D(x, x, \mathrm{refl}_x) \equiv (\mathrm{refl}_x \cdot \mathrm{refl}_x^{-1} = \mathrm{refl}_x)$.

Wir wissen bereits, dass $\mathrm{refl}_x^{-1} \equiv \mathrm{refl}_x$ und $\mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x$. Da also $\mathrm{refl}_x \cdot \mathrm{refl}_x^{-1} \equiv \mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x$, folgt:

$$D(x, x, \mathrm{refl}_x) \equiv (\mathrm{refl}_x = \mathrm{refl}_x)$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_x} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

und somit mithilfe der Pfadinduktion auch für alle $p : x =_A y$ ein Objekt

$$\mathrm{ind}_{=A}(D, d, x, y, p) : (p \cdot p^{-1} = \mathrm{refl}_x),$$

womit die Aussage bewiesen wäre.

Die zweite Gleichung $p^{-1} \cdot p = \mathrm{refl}_y$ kann wieder analog gezeigt werden.

c) Sei die Familie von Typen $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ für ein $A : \mathcal{U}$ definiert durch

$$D(x, y, p) \equiv (p = (p^{-1})^{-1})$$

Mit dem Wissen, dass $\mathrm{refl}_x^{-1} \equiv \mathrm{refl}_x$ folgt hier

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv (\mathrm{refl}_x = (\mathrm{refl}_x^{-1})^{-1}) \\ &\equiv (\mathrm{refl}_x = (\mathrm{refl}_x)^{-1}) \\ &\equiv (\mathrm{refl}_x = \mathrm{refl}_x) \end{aligned}$$

Somit finden wir wieder eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_x} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

und nach dem Prinzip der Pfadinduktion für alle $p : x =_A y$ ein Objekt

$$\mathrm{ind}_{=A}(D, d, x, y, p) : (p = (p^{-1})^{-1})$$

womit wir ein Objekt im gesuchten Typ gefunden und somit die Aussage bewiesen haben.

d) Hier haben wir wieder mehrere Pfade über die wir induzieren müssen, um ein konkretes Element konstruieren zu können. Sei $A : \mathcal{U}$ ein beliebiger aber fester Typ und sei zuerst $D_1 : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen, die definiert ist durch

$$D_1(x, y, p) \equiv \Pi_{(z,w:A)}\Pi_{(q:y=z)}\Pi_{(r:z=w)}((p \cdot q) \cdot r = p \cdot (q \cdot r))$$

Dann gilt

$$D_1(x, x, \mathrm{refl}_x) \equiv \Pi_{(z,w:A)}\Pi_{(q:x=z)}\Pi_{(r:z=w)}((\mathrm{refl}_x \cdot q) \cdot r = \mathrm{refl}_x \cdot (q \cdot r)).$$

Sei als nächstes $D_2 : \Pi_{(x,z:A)}(x = z) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D_2(x, z, q) \equiv \Pi_{(w:A)}\Pi_{(r:z=w)}((\mathrm{refl}_x \cdot q) \cdot r = \mathrm{refl}_x \cdot (q \cdot r))$$

Dann gilt

$$D_2(x, x, \mathrm{refl}_x) \equiv \Pi_{(w:A)}\Pi_{(r:x=w)}((\mathrm{refl}_x \cdot \mathrm{refl}_x) \cdot r = \mathrm{refl}_x \cdot (\mathrm{refl}_x \cdot r))$$

Und sei als letztes $D_3 : \Pi_{(x,w:A)}(x = w) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D_3(x, w, r) \equiv ((\mathrm{refl}_x \cdot \mathrm{refl}_x) \cdot r = \mathrm{refl}_x \cdot (\mathrm{refl}_x \cdot r))$$

Da wir wissen, dass $\mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x$ gilt, folgt nun

$$\begin{aligned} D_3(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl}_x \cdot \mathrm{refl}_x) \cdot \mathrm{refl}_x = \mathrm{refl}_x \cdot (\mathrm{refl}_x \cdot \mathrm{refl}_x)) \\ &\equiv (\mathrm{refl}_x \cdot \mathrm{refl}_x = \mathrm{refl}_x \cdot \mathrm{refl}_x) \\ &\equiv (\mathrm{refl}_x = \mathrm{refl}_x) \end{aligned}$$

Also gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_x} : \Pi_{(x:A)} D_3(x, x, \mathrm{refl}_x)$$

Dreimalige Pfadinduktion liefert uns also den Beweis der Assoziativität bei der Komposition von Pfaden.

□

Da wir nun schon ein wenig mit Pfaden arbeiten können, wollen wir uns als nächstes anschauen, wie sich Funktionen auf diese auswirken. Denn wenn es in A einen Pfad von x nach y gibt, müssen wohl auch $f(x) : B$ und $f(y) : B$ für eine Funktion $f : A \to B$ einen Zusammenhang haben.

Lemma 1.3.4: *Für eine Funktion $f : A \to B$ und $x, y : A$ gibt es eine Funktion*

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

mit $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$.

Nachfolgende Skizze soll verdeutlichen, was diese Funktion aussagt:

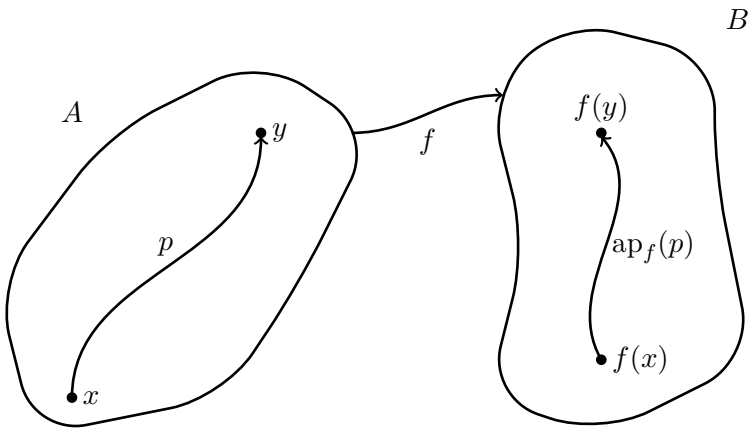

Abbildung 4

Gibt es also einen Pfad p von x nach y, sind x und y also propositional gleich, dann soll es auch einen Pfad geben, der $f(x)$ und $f(y)$ verbindet, also auch diese sollen propositional gleich sein.

Beweis. Sei für $A : \mathcal{U}$ eine Familie von Typen $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ definiert durch

$$D(x, y, p) \equiv (f(x) = f(y))$$

Dann gilt also $D(x, x, \mathrm{refl}_x) \equiv (f(x) = f(x))$ und somit gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{f(x)} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

Pfadinduktion liefert uns also ein Objekt des gesuchten Typs und gibt uns damit die Existenz der Funktion $\mathrm{ap}_f : (x =_A y) \to (f(x) =_B f(y))$, bei der für alle $x : A$ gilt: $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$

□

Um auch mit dieser Funktion arbeiten zu können, gibt es ein paar hilfreiche Regeln, die wir im nächsten Lemma anführen.

Lemma 1.3.5: *Seien $f : A \to B$ und $g : C \to E$ Funktionen und $p : x =_A y$ und $q : y =_A z$ zwei Pfade in A mit $x, y, z : A$. Dann gilt:*

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

b) $\mathrm{ap}_f(p^{-1}) = \mathrm{ap}_f(p)^{-1}$

c) $\mathrm{ap}_g(\mathrm{ap}_f(p)) = \mathrm{ap}_{g \circ f}(p)$

d) $\mathrm{ap}_{\mathrm{id}_A}(p) = p$

Beweis.

a) Nachdem die Aussage zwei Pfade p und q beinhaltet, müssen wir wieder über beide Pfade induzieren. Dafür beginnen wir mit einer Familie von Typen $D_1 : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ für $A : \mathcal{U}$ und $f : A \to B$ definiert durch

$$D_1(x, y, p) \equiv \Pi_{(z:A)}\Pi_{(q:y=z)}(\mathrm{ap}_f(p \cdot q) = \mathrm{ap}_f(p) \cdot \mathrm{ap}_f(q))$$

Damit folgt

$$D_1(x, x, \mathrm{refl}_x) \equiv \Pi_{(z:A)}\Pi_{(q:x=z)}(\mathrm{ap}_f(\mathrm{refl}_x \cdot q) = \mathrm{ap}_f(\mathrm{refl}_x) \cdot \mathrm{ap}_f(q)).$$

Sei also weiter $D_2 : \Pi_{(x,z:A)}(x = z) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D_2(x, z, q) \equiv (\mathrm{ap}_f(\mathrm{refl}_x \cdot q) = \mathrm{ap}_f(\mathrm{refl}_x) \cdot \mathrm{ap}_f(q))$$

Wir wissen bereits, dass $\mathrm{refl}_x \cdot \mathrm{refl}_x = \mathrm{refl}_x$ und $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$ gilt, und deswegen folgt

$$\begin{aligned} D_2(x, x, \mathrm{refl}_x) &\equiv (\mathrm{ap}_f(\mathrm{refl}_x \cdot \mathrm{refl}_x) = \mathrm{ap}_f(\mathrm{refl}_x) \cdot \mathrm{ap}_f(\mathrm{refl}_x)) \\ &\equiv (\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)} \cdot \mathrm{refl}_{f(x)}) \\ &\equiv (\mathrm{refl}_{f(x)} = \mathrm{refl}_{f(x)}) \end{aligned}$$

Somit gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_{f(x)}} : \Pi_{(x:A)} D_2(x, x, \mathrm{refl}_x)$$

Mit zweimaliger Pfadinduktion finden wir also ein konkretes Objekt des zuerst definierten Typs und somit ist die Aussage bewiesen.

b) Sei hierfür $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen mit $A : \mathcal{U}$ und $f : A \to B$ definiert durch

$$D(x, y, p) \equiv (\mathrm{ap}_f(p^{-1}) = \mathrm{ap}_f(p)^{-1})$$

Mit dem Wissen, dass $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$ und $\mathrm{refl}_x^{-1} = \mathrm{refl}_x$ folgt dann

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv (\mathrm{ap}_f(\mathrm{refl}_x^{-1}) = \mathrm{ap}_f(\mathrm{refl}_x)^{-1}) \\ &\equiv (\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}^{-1}) \\ &\equiv (\mathrm{refl}_{f(x)} = \mathrm{refl}_{f(x)}) \end{aligned}$$

Also gibt es also eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_{f(x)}} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

und somit liefert das Prinzip der Pfadinduktion den Rest des Beweises.

c) Sei $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen für $A : \mathcal{U}$ und $f : A \to B$ und $g : C \to E$ definiert durch

$$D(x, y, p) \equiv (\mathrm{ap}_g(\mathrm{ap}_f(p)) = \mathrm{ap}_{g \circ f}(p))$$

Da wir wissen, dass $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$ gilt, folgt hierfür

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv (\mathrm{ap}_g(\mathrm{ap}_f(\mathrm{refl}_x)) = \mathrm{ap}_{g \circ f}(\mathrm{refl}_x)) \\ &\equiv (\mathrm{ap}(\mathrm{refl}_{f(x)}) = \mathrm{refl}_{g \circ f(x)}) \\ &\equiv (\mathrm{refl}_{g(f(x))} = \mathrm{refl}_{g \circ f(x)}) \\ &\equiv (\mathrm{refl}_{g \circ f(x)} = \mathrm{refl}_{g \circ f(x)}) \end{aligned}$$

Also gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_{g \circ f(x)}} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x),$$

womit das Prinzip der Pfadinduktion uns nun die Aussage als bewiesen ansehen lässt.

d) Sei hier $\mathrm{id}_A : A \to A$ für $A : \mathcal{U}$ gegeben und eine Familie von Typen $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ definiert durch

$$D(x, y, p) \equiv (\mathrm{ap}_{\mathrm{id}_A}(p) = p)$$

Wegen $\mathrm{id}_A(x) = x$ und $\mathrm{ap}_f(\mathrm{refl}_x) = \mathrm{refl}_{f(x)}$ folgt für den Fall $f \equiv \mathrm{id}_A$

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv (\mathrm{ap}_{\mathrm{id}_A}(\mathrm{refl}_x) = \mathrm{refl}_x) \\ &\equiv (\mathrm{refl}_{\mathrm{id}_A(x)} = \mathrm{refl}_x) \\ &\equiv (\mathrm{refl}_x = \mathrm{refl}_x) \end{aligned}$$

Damit gibt es also eine Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_x} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x).$$

Somit gilt die Aussage nach dem Prinzip der Pfadinduktion als bewiesen.

□

Nun wissen wir also bereits für Funktionen, dass diese Rücksicht auf propositionale Gleichheit nehmen. Gibt es einen Pfad p von x nach y, so gibt es auch einen Pfad von $f(x)$ nach $f(y)$, den uns die Funktion ap_f liefert. Doch wir wollen uns ebenfalls noch ansehen, wie es mit Familien von Typen aussieht, wie $P(x)$ und $P(y)$ zueinander stehen, wenn es einen Pfad p von x nach y gibt.

Lemma 1.3.6: Transport
Für eine Familie von Typen $P : A \to \mathcal{U}$ mit einem Typ $A : \mathcal{U}$ und $p : x =_A y$ gibt es eine Funktion

$$p_* : P(x) \to P(y).$$

Beweis. Lasst uns jetzt also zeigen, dass tatsächlich solch eine Funktion existiert. Dafür nutzen wir wieder das Prinzip der Pfadinduktion, beginnend mit einer Familie von Typen $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ für $A : \mathcal{U}$ definiert durch

$$D(x, y, p) \equiv P(x) \to P(y)$$

Dann gilt natürlich

$$D(x, x, \mathrm{refl}_x) \equiv P(x) \to P(x).$$

Und somit gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{id}_{P(x)} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

und mit Pfadinduktion können wir uns nun also für alle $p : x = y$ die Funktion p_* wie folgt definieren:

$$p_* \equiv \mathrm{ind}_{=A}(D, d, x, y, p) : P(x) \to P(y).$$

Also existiert eine derartige Funktion.

□

Diese Aussage bedeutet also, dass eine Familie von Typen P über einen Typ A Gleichheit in A anerkennt. Wenn x und y in A propositional gleich sind, dann sind auch $P(x)$ und $P(y)$ gleich, was bedeutet: Gibt es in A einen Pfad von x nach y, dann gibt es auch einen Pfad zwischen $P(x)$ und $P(y)$, wie folgende Abbildung verdeutlicht:

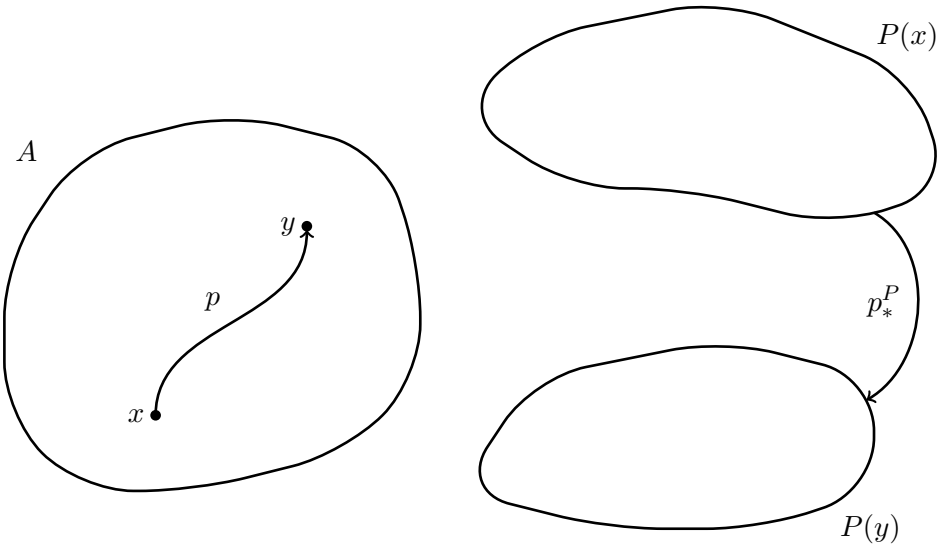

Abbildung 5

Die Notation p_*^P nutzen wir anstelle von p_*, um die Familie von Typen P, mit der wir gerade arbeiten, ebenfalls anzugeben.

Um damit auch arbeiten zu können, brauchen wir aber noch eine Aussage über den Zusammenhang zwischen diesen beiden Funktionen.

Lemma 1.3.7: *Sei $f : \Pi_{(x:A)}P(x)$ eine abhängige Funktion über einen Typ $A : \mathcal{U}$ und eine Familie von Typen $P : A \to \mathcal{U}$, dann gibt es eine ebenfalls abhängige Funktion*

$$\mathrm{apd}_f : \Pi_{(p:x=y)}(p_*^P(f(x)) =_{P(y)} f(y)).$$

Dieses Lemma sagt uns also, wenn es in A einen Pfad p zwischen x und y gibt, dann gibt es auch in $P(y)$ einen Pfad apd_f zwischen $p_*^P(f(x))$ und $f(y)$. Die Funktion p_*^P transportiert den Punkt $f(x) : P(x)$ also in den Typ $P(y)$, sodass dann $p_*^P(f(x)) : P(y)$ gilt, damit wir innerhalb des Typs $P(y)$ einen Pfad nach $f(y) : P(y)$ finden können.

Diese Abbildung soll die gerade getroffene Aussage verdeutlichen.

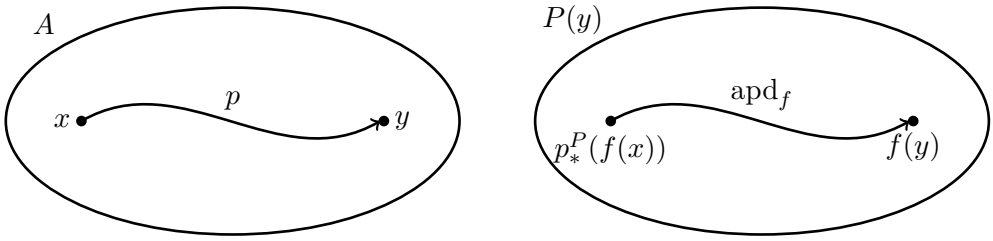

Abbildung 6

Beweis. Kommen wir nun also dazu, die Aussage auch formal zu beweisen. Dafür sei $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D(x, y, p) \equiv (p_*^P(f(x)) = f(y))$$

Wie vorher bereits erklärt, gilt $(\mathrm{refl}_x)_*^P(f(x)) \equiv f(x)$ und somit folgt

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl})_*^P(f(x)) = f(x)) \\ &\equiv (f(x) = f(x)) \end{aligned}$$

Also gibt es eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{f(x)} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x)$$

Für alle $p : x = y$ mit $x, y : A$ liefert uns das Prinzip der Pfadinduktion nun

$$\mathrm{apd}_f : p_*^P(f(x)) = f(y) \ .$$

□

Nicht alle Familien von Typen hängen von einem anderen Typ ab und wie bereits bei der Definition von Familien von Typen gesehen gibt es auch Familien konstanter Typen, also eine Familie von Typen $P : A \to \mathcal{U}$ mit $P(x) \equiv B$ für $A, B : \mathcal{U}$ und $x : A$. Auch hierfür können wir eine Aussage über diesen gerade kennen gelernten Pfad apd_f machen.

Lemma 1.3.8: *Sei $P : A \to \mathcal{U}$ mit $P(x) \equiv B$ für $x : A$ und ein festes $B : \mathcal{U}$ und sei $b : B$. Dann gibt es für $p : x = y$ mit $x, y : A$ den Pfad*

$$\mathrm{transportconst}_p^B(b) : p_*^P(b) = b.$$

Beweis. Sei also $b : B$ beliebig, aber fest für $B : \mathcal{U}$. Dann definieren wir die Familie von Typen $D_b : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ durch

$$D_b(x, y, p) \equiv (p_*^P(b) = b)$$

Wir wissen bereits, dass $(\mathrm{refl}_x)^P_*(b) \equiv \mathrm{id}_B(b) \equiv b$ gilt, und somit folgt

$$\begin{aligned} D_b(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl}_x)^P_*(b) = b) \\ &\equiv (b = b) \end{aligned}$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_b : \Pi_{(x:A)} D_b(x, x, \mathrm{refl}_x)$$

womit Pfadinduktion den Rest des Beweises und somit auch die Existenz von $\mathrm{transportconst}^B_p(b) : p^P_*(b) = b$ für alle $p : x = y$ und $x, y : A$ liefert.

□

Damit können wir nun auch einen Zusammenhang zwischen ap_f und apd_f herstellen, wie folgendes Lemma zeigt:

Lemma 1.3.9: *Sei $f : A \to B$ und $p : x =_A y$, dann gilt:*

$$\mathrm{apd}_f(p) = \mathrm{transportconst}^B_p(f(x)) \cdot \mathrm{ap}_f(p).$$

Beweis. Dafür beginnen wir wieder mit $D : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$, einer Familie von Typen, definiert durch

$$D(x, y, p) \equiv (\mathrm{apd}_f(p) = \mathrm{transportconst}^B_p(f(x)) \cdot \mathrm{ap}_f(p))$$

Mit dem Wissen, dass sowohl $\mathrm{transportconst}^B_{\mathrm{refl}_x}(f(x)) \equiv \mathrm{refl}_{f(x)}$ und ebenfalls $\mathrm{ap}_f(\mathrm{refl}_x) \equiv \mathrm{refl}_{f(x)}$ als auch $\mathrm{apd}_f(\mathrm{refl}_x) \equiv \mathrm{refl}_{f(x)}$ und $\mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x$ gilt, folgt hier

$$\begin{aligned} D(x, x, \mathrm{refl}_x) &\equiv (\mathrm{apd}_f(\mathrm{refl}_x) = \mathrm{transportconst}^B_{\mathrm{refl}_x}(f(x)) \cdot \mathrm{ap}_f(\mathrm{refl}_x)) \\ &\equiv (\mathrm{refl}_{f(x)} = \mathrm{refl}_{f(x)} \cdot \mathrm{refl}_{f(x)}) \\ &\equiv (\mathrm{refl}_{f(x)} = \mathrm{refl}_{f(x)}) \end{aligned}$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{\mathrm{refl}_{f(x)}} : \Pi_{(x:A)} D(x, x, \mathrm{refl}_x),$$

womit das Prinzip der Pfadinduktion nun den Beweis vervollständigt.

□

Für diese Funktionen gibt es noch ein paar weitere hilfreiche Aussagen, die wir jetzt am Ende dieses Kapitels beweisen wollen.

Lemma 1.3.10:

a) *Für* $P : A \to \mathcal{U}$, $p : x =_A y$, $q : y =_A z$ *und* $u : P(x)$ *gilt:*

$$q_*(p_*(u)) = (p \cdot q)_*(u)$$

b) *Sei* $f : A \to B$ *eine Funktion,* $P : B \to \mathcal{U}$ *eine Familie von Typen,* $u : P(f(x))$ *und* $p : x =_A y$*. Dann gilt*

$$p_*^{P \circ f}(u) = (\mathrm{ap}_f(p))_*^P(u)$$

c) *Seien* $P, Q : A \to \mathcal{U}$ *Familien von Typen über einen Typ* $A : \mathcal{U}$ *und* $f : \Pi_{(x:A)} P(x) \to Q(x)$ *eine abhängige Funktion. Seien* $u : P(x)$ *und* $p : x =_A y$*, dann gilt*

$$p_*^Q(f_x(u)) = f_y(p_*^P(u))$$

Beweis.

a) Hier gibt es wieder zwei Pfade p und q, über die wir induzieren müssen, um ein konkretes Objekt angeben zu können. Beginnen wir also mit einer Familie von Typen $D_1^u : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ definiert durch

$$D_1^u(x, y, p) \equiv \Pi_{(z:A)} \Pi_{(q:y=z)}(q_*(p_*(u)) = (p \cdot q)_*(u)).$$

Dann gilt

$$D_1^u(x, x, \mathrm{refl}_x) \equiv \Pi_{(z:A)} \Pi_{(q:x=z)}(q_*((\mathrm{refl}_x)_*(u)) = (\mathrm{refl}_x \cdot q)_*(u))$$

Sei weiter $D_2^u : \Pi_{(x,z:A)}(x = z) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D_2^u(x, z, q) \equiv (q_*((\mathrm{refl}_x)_*(u)) = (\mathrm{refl}_x \cdot q)_*(u))$$

Mit dem Wissen, dass $\mathrm{refl}_x \cdot \mathrm{refl}_x \equiv \mathrm{refl}_x$ und $(\mathrm{refl}_x)_*(u) \equiv u$ für $u : P(x)$ folgt nun

$$\begin{aligned} D_2^u(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl}_x)_*((\mathrm{refl}_x)_*(u)) = (\mathrm{refl}_x \cdot \mathrm{refl}_x)_*(u)) \\ &\equiv ((\mathrm{refl}_x)_*(u) = (\mathrm{refl}_x)_*(u)) \\ &\equiv (u = u) \end{aligned}$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_u : \Pi_{(x:A)} D_2^u(x, x, \mathrm{refl}_x),$$

womit die Aussage nach dem Prinzip der Pfadinduktion als bewiesen gilt.

b) Sei $f : A \to B$ eine Funktion und $D_u : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen mit

$$D_u(x, y, p) \equiv (p_*^{P \circ f}(u) = (\mathrm{ap}_f(p))_*^P(u))$$

Dann folgt mit $(\mathrm{refl}_x)_*^P(a) \equiv a$ für $a : P(x)$ und $\mathrm{ap}_f(\mathrm{refl}_x) \equiv \mathrm{refl}_{f(x)}$ und da $u : P(f(x))$

$$\begin{aligned} D_u(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl}_x)_*^{P \circ f}(u) = (\mathrm{ap}_f(\mathrm{refl}_x))_*^P(u)) \\ &\equiv (u = (\mathrm{refl}_{f(x)})_*^P(u)) \\ &\equiv (u = u) \end{aligned}$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_u : \Pi_{(x:A)} D_u(x, x, \mathrm{refl}_x)$$

und Pfadinduktion gibt uns damit die gewünschte Gleichheit für alle $p : x = y$ mit $x, y : A$.

c) Sei $D_u : \Pi_{(x,y:A)}(x = y) \to \mathcal{U}$ eine Familie von Typen definiert durch

$$D_u(x, y, p) \equiv (p_*^Q(f_x(u)) = f_y(p_*^P(u)))$$

Da $f_x(u) : Q(x)$ für $u : P(x)$ für ein $x : A$ und $(\mathrm{refl}_x)_*(a) \equiv a$ für $a : P(x)$, folgt also

$$\begin{aligned} D_u(x, x, \mathrm{refl}_x) &\equiv ((\mathrm{refl}_x)_*^Q(f_x(u)) = f_x((\mathrm{refl}_x)_*^P(u))) \\ &\equiv (f_x(u) = f_x(u)) \end{aligned}$$

Also finden wir eine abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{f_x(u)} : \Pi_{(x:A)} D_u(x, x, \mathrm{refl}_x),$$

sodass das Prinzip der Pfadinduktion den Beweis vervollständigt.

□

2 Äquivalenz und Eindeutigkeit in der univalenten Typentheorie

Im ersten Kapitel haben wir nun mehrere grundlegende Typen kennengelernt und dabei bereits auch einige spezielle induktive Typen.

Im zweiten Kapitel dieser Arbeit wollen wir uns dieses Prinzip induktiv definierter Typen noch einmal genauer ansehen. Für einen induktiv definierten Typ haben wir also endlich viele Konstruktoren, durch die wir alle Elemente des entsprechenden Typs beschreiben können, so besteht zum Beispiel der Typ der natürlichen Zahlen nur aus zwei Konstruktoren, aus der 0 und der Funktion, die eine natürliche Zahl auf ihren Nachfolger abbildet. Jede natürliche Zahl ist also entweder die 0 oder der Nachfolger einer anderen natürlichen Zahl. Außerdem haben wir zu jedem dieser Typen das passende Induktionsprinzip kennen gelernt.

Zuerst wollen wir aber noch zeigen, dass diese Induktionsprinzipien auch eindeutig sind, dass induktiv definierte Typen generell eindeutig sind. Damit werden wir uns im nächsten Teil dieser Arbeit beschäftigen.

Sätze und Beweise dieses Kapitels sind auch in den Kapiteln 2.4, 2.9., 2.10, den Kapiteln 3.1 bis 3.3 und den Kapiteln 5.1 und 5.2 von [6] zu finden.

2.1 Axiome der Typentheorie

Zu zeigen, dass ein Induktionsprinzip oder ein induktiver Typ eindeutig ist, ist damit verknüpft, zu zeigen, dass zwei gleich aussehende induktive Typen auch tatsächlich gleich sind. Um aber überhaupt eine Aussage darüber machen zu können, ob zwei Typen äquivalent zueinander sind, müssen wir erst einmal wissen, was dies bedeutet.

2.1.1 Äquivalenz von Typen

Bevor wir aber zu den nötigen Axiomen der Typentheorie kommen, die uns letztlich erst Äquivalenz von Typen liefern können, müssen wir uns davor anschauen, wie eben gesagt, was es heißt, dass zwei Typen gleich sind, dass sie also äquivalent zueinander sind. Beginnen wir dafür mit der Äquivalenz von Funktionen. Dies benötigt jedoch ein wenig Vorarbeit.

Unsere Intuition sagt uns, dass zwei Funktionen f, g gleich sind, wenn sie an jedem Punkt gleich sind. Übersetzen wir dies mithilfe unserer Tabelle am Anfang des ersten Kapitels (vgl. Seite 5), erhalten wir den Typ der folgenden Definition:

Definition 5: $f \sim g$
Seien $f, g : \Pi_{(x:A)} P(x)$ zwei abhängige Funktionen über die Familie von Typen $P : A \to \mathcal{U}$. Dann nennen wir die abhängige Funktion

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

eine Homotopie von f nach g.

Für den Fall, dass wir eine unabhängige Funktion $f : A \to B$ haben, können wir dennoch mit dieser und allen folgenden Definitionen und Aussagen arbeiten, indem wir die Familie von Typen $P : A \to \mathcal{U}$ einfach als konstante Familie ansehen, dass P also unabhängig von $x : A$ ist. Wenn wir dies aber genauer betrachten, beinhaltet dieser Typ nur Pfade, gibt uns also nur propositionale Gleichheit zwischen den Funktionswerten. Dieser Typ gibt uns somit tatsächlich noch nicht die Gleichheit beider Funktionen.

Definition 6: (g, α, β)
Das Tripel (g, α, β), wobei $g : B \to A$ eine Funktion ist und α und β Homotopien sind vom Typ

$$\alpha : f \circ g \sim \mathrm{id}_B \qquad \textit{und} \qquad \beta : g \circ f \sim \mathrm{id}_A,$$

nennen wir das Quasi-Inverse einer Funktion $f : A \to B$, wobei wir den Typ dieser Quasi-Inversen definieren als $\mathrm{qinv}(f)$, sodass gilt: $(g, \alpha, \beta) : \mathrm{qinv}(f)$.

Um nun auch Äquivalenz von Typen untersuchen zu können, brauchen wir folgende, daraus resultierende Definition:

Definition 7: (Voevodsky) $A \simeq_{\mathcal{U}} B$
Der Typ, der definiert ist durch

$$(A \simeq B) \equiv \Sigma_{(f:A\to B)} \Big(\Big(\Sigma_{(g:B\to A)}(f \circ g \sim \mathrm{id}_B)\Big) \times \Big(\Sigma_{(h:B\to A)}(h \circ f \sim \mathrm{id}_A)\Big)\Big),$$

ist der Typ einer Äquivalenz von A nach B.

Gibt es also ein Objekt im Typ $A \simeq B$ sprechen wir von Äquivalenz zwischen den Typen $A : \mathcal{U}$ und $B : \mathcal{U}$. Mithilfe dieser Definition würde uns unsere Intuition sagen, dass für zwei abhängige Funktionen $f, g : \Pi_{(x:A)}P(x)$ gilt:

$$(f = g) \simeq_{\mathcal{U}} \Big(\Pi_{(x:A)}(f(x) =_{P(x)} g(x))\Big)$$

Funktionen wären also gleich, wenn sie an allen Punkten gleich sind. Doch wenn wir dies nun genauer betrachten, bedeutet das, dass ein Pfad im Raum der Funktionen, also ein Pfad von f nach g, dasselbe ist wie eine Homotopie, wie wir sie in Definition 2.1.1 kennen gelernt haben, also Pfade zwischen $f(x)$ und $g(x)$ für alle $x : A$.

Tatsächlich können wir mithilfe von Pfadinduktion zeigen, dass es eine Funktion gibt, die vom Typ $f = g$ in den Typ $\Pi_{(x:A)}(f(x) =_{P(x)} g(x))$ abbildet.

Beweis. Sei die Familie von Typen $D : \Pi_{(f,g:\Pi_{(x:A)})P(x)}(f = g) \to \mathcal{U}$ definiert durch

$$D(f, g, p) \equiv \Pi_{(x:A)} f(x) =_{P(x)} g(x)$$

Dann gilt

$$D(f, f, \mathrm{refl}_f) \equiv \Pi_{(x:A)} f(x) =_{P(x)} f(x)$$

Somit können wir uns die abhängige Funktion

$$d \equiv \lambda x.\mathrm{refl}_{f(x)} : \Pi_{(f:\Pi_{(x:A)}P(x))} D(f, f, \mathrm{refl}_f)$$

definieren, sodass das Prinzip der Pfadinduktion uns die Existenz der Funktion

$$\mathrm{happly} : \Pi_{(f,g:\Pi_{(x:A)}P(x))}(f = g) \to \Pi_{(x:A)} f(x) =_{P(x)} g(x)$$

mit $\mathrm{happly}(\mathrm{refl}_f) \equiv \lambda x.\mathrm{refl}_{f(x)}$ liefert.

□

2.1.2 Das Extensionalitätsaxiom

Wir haben bereits gezeigt, dass es eine Funktion

$$\text{happly} : \Pi_{(f,g:\Pi_{(x:A)}P(x))}(f = g) \to \Pi_{(x:A)}f(x) =_{P(x)} g(x)$$

gibt. Zu zeigen, dass es auch eine Funktion

$$\text{funext} : \Big(\Pi_{(x:A)}(f(x) = g(x))\Big) \to (f = g)$$

gibt, insbesondere eine Funktion, die quasi-invers ist zu der Funktion, deren Existenz wir bereits gezeigt haben, ist mit Pfadinduktion nicht mehr möglich. Um also wirklich davon ausgehen zu können, dass Funktionen in beide Richtungen existieren, brauchen wir folgendes Axiom.

Extensionalitätsaxiom:
Die Funktion

$$\text{happly} : (f = g) \to \Pi_{(x:A)}(f(x) =_{P(x)} g(x))$$

ist für jeglichen Typ $A : \mathcal{U}$ und jegliche abhängige Funktionen f, g über jegliche Familie von Typen $P : A \to \mathcal{U}$ eine Äquivalenz.

Um erst einmal auf etwas zurückzugreifen, was wir bereits im ersten Kapitel kennen gelernt haben, wollen wir noch einmal auf die Funktion p_* zurückkommen, deren Existenz wir erstmals in Lemma 1.3.6 gezeigt haben. Diese Funktion lieferte uns im Groben eine Möglichkeit zu sagen, wenn x zum Beispiel eine Eigenschaft besitzt und x und y propositional gleich sind, dass dann auch y diese Eigenschaft besitzen muss. Wir hatten also gezeigt, dass gilt: $p_*(u) : P(y)$ mit $u : P(x)$.

Lemma 2.1.2.1: *Für $A, B : X \to \mathcal{U}$, $p : x =_X y$, $f : \Pi_{(x:A)}A(x) \to B(x)$ und $g : A(y) \to B(y)$ gilt:*

$$(p_*(f) = g) \simeq_{\mathcal{U}} \Pi_{(a:A(x))}(p_*(f(a)) = g(p_*(a)))$$

Um an dieser Stelle davon ausgehen zu können, dass diese beiden Typen äquivalent sind, brauchen wir also das Extensionalitätsaxiom, um tatsächlich die Existenz von quasi-inversen Funktionen in beide Richtungen annehmen zu können.

Anschaulich können wir die Aussage des Lemmas mithilfe folgender Skizze ein wenig besser nachvollziehen.

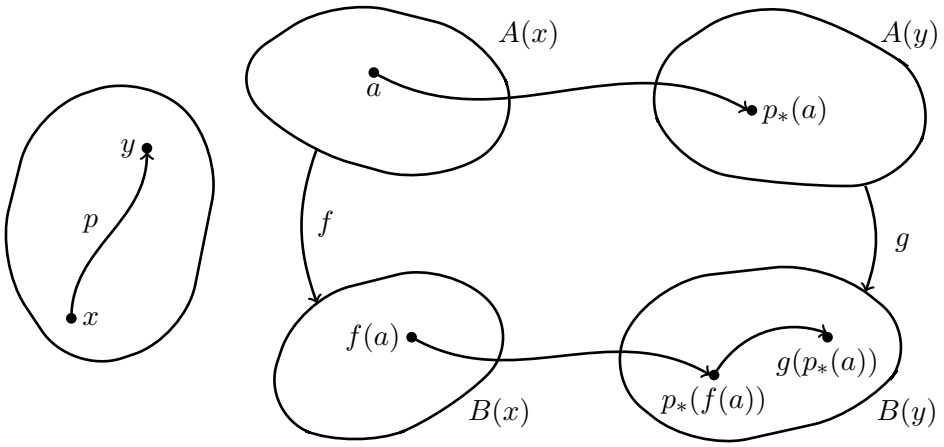

Abbildung 7

2.1.3 Das Univalenzaxiom

Als nächstes wollen wir noch einmal auf die Äquivalenz von Typen zurückkommen. Für Typen $A, B : \mathcal{U}$ können wir uns bekanntlich den Typ $A =_{\mathcal{U}} B$ definieren, der im Grunde genommen aussagen soll, dass A und B innerhalb eines Universums gleich sind, dass also $A \simeq B$ gelten soll. Wir können tatsächlich auch wieder zeigen, dass es eine Funktion

$$\text{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq_{\mathcal{U}} B).$$

für Typen $A, B : \mathcal{U}$ gibt. Doch auch hier können wir die Existenz einer quasiinversen Funktion mithilfe dessen, was uns in der Typentheorie zur Verfügung steht, nicht garantieren. Daher gibt es folgendes Axiom.

Univalenzaxiom:
Für $A, B : \mathcal{U}$ ist die Funktion idtoeqv *eine Äquivalenz.*

Das Axiom gibt uns so die Existenz einer Funktion

$$\text{ua} : (A \simeq_{\mathcal{U}} B) \to (A =_{\mathcal{U}} B),$$

und garantiert damit

$$(A =_{\mathcal{U}} B) \simeq_{\mathcal{U}} (A \simeq_{\mathcal{U}} B).$$

Es ähnelt also dem Extensionalitätsaxiom mit dem Unterschied, dass hier Typen und keine Funktionen betrachtet werden.

2.2 Eindeutigkeit induktiver Typen

Da wir also bereits mehr darüber wissen, wann Funktionen und Typen gleich sind und wir das eine mit dem anderen identifizieren können, haben wir die nötigen Axiome kennen gelernt, die uns überhaupt erst die Äquivalenz von Typen und Funktionen geben. Wir können nun also Aussagen über die Eindeutigkeit von Funktionen und Typen machen, doch zu keinem Zeitpunkt haben wir etwas über die Eindeutigkeit von Induktionsprinzipien oder der induktiv definierten Typen gesagt. Dies wollen wir jetzt nachholen. Dafür behaupten wir, dass ein Induktionsprinzip sogar die eigene Eindeutigkeit zeigen kann, was wir zuerst am Beispiel der natürlichen Zahlen (siehe Kapitel 1.2.9) auch zeigen wollen.

Lemma 2.2.1: *Seien $f, g : \Pi_{(x:\mathbb{N})} P(x)$ zwei abhängige Funktionen und*

$$e_0 : P(0) \qquad e_1 : \Pi_{(n:\mathbb{N})} P(n) \to P(\mathrm{succ}(n))$$

sodass gilt:

$$f(0) = e_0 \qquad \text{und} \qquad g(0) = e_0$$

und

$$\Pi_{(n:\mathbb{N})} f(\mathrm{succ}(n)) = e_1(n, f(n))$$
$$\Pi_{(n:\mathbb{N})} g(\mathrm{succ}(n)) = e_1(n, g(n))$$

Dann sind f und g gleich.

Beweis. Wir zeigen wir dies mithilfe des Induktionsprinzip der natürlichen Zahlen. Dafür beginnen wir mit der Familie von Typen $D : \mathbb{N} \to \mathcal{U}$ definiert durch

$$D(x) \equiv (f(x) = g(x)).$$

Für das Induktionsprinzip für natürliche Zahlen haben wir also schon Objekte des Typs $P(0)$ und $\Pi_{(n:N)} P(n) \to P(succ(n))$. Das Prinzip sagt uns außerdem, dass wir diese Gleichung zuerst einmal für das Element 0 zeigen müssen. Dafür folgt natürlich:

$$f(0) = e_0 = g(0)$$

Nun nehmen wir an, dass es ein $n : \mathbb{N}$ gibt, sodass gilt: $f(n) = g(n)$. Mithilfe dieser Annahme folgt dann:

$$f(\mathrm{succ}(n)) = e_1(n, f(n)) = e_1(n, g(n)) = g(\mathrm{succ}(n))$$

Damit sagt uns nun das Induktionsprinzip für natürliche Zahlen, dass gilt:

$$\Pi_{(x:\mathbb{N})}(f(x) = g(x))$$

Also haben wir jetzt gezeigt, dass es für jedes $x : \mathbb{N}$ einen Pfad zwischen $f(x)$ und $g(x)$ gibt. Um daraus folgern zu können, dass wirklich $f = g$ gilt, brauchen wir hier wieder das Extensionalitätsaxiom, das uns die Funktion

$$\text{funext} : \left(\Pi_{(x:\mathbb{N})}(f(x) = g(x))\right) \to (f = g)$$

liefert, mit der folgt, dass f und g gleich sind.

□

Und damit haben wir schließlich gezeigt, dass das Induktionsprinzip für natürliche Zahlen auch eindeutig ist. Mithilfe von Beweisen gleicher Art lässt sich dies natürlich auch für alle weiteren Induktionsprinzipien zeigen.

Aber eigentlich wollen wir nicht nur wissen, dass das Induktionsprinzip für einen induktiv definierten Typ eindeutig ist, sondern auch, dass induktive Typen selbst eindeutig sind. Haben wir also zwei Typen, die durch gleiche Konstruktoren definiert sind, wollen wir wissen, dass diese beiden Typen gleich sind und wir den einen immer durch den jeweils anderen ersetzen können, wenn wir dies benötigen. Dies wollen wir wieder am Beispiel der natürlichen Zahlen veranschaulichen und erklären.

Nehmen wir also an, wir hätten zwei induktiv definierte Typen, $\mathbb{N}$ und $\mathbb{N}'$, wobei $\mathbb{N}$ dem Typ der natürlichen Zahlen aus Kapitel 1.2.9 entspricht und $\mathbb{N}'$ als Konstruktoren die $0'$ und eine Abbildung $\text{succ}' : \mathbb{N}' \to \mathbb{N}'$ besitzt. Dann werden auch die Rekursion- und Induktionsprinzipien für $\mathbb{N}'$ genauso aussehen wie die bereits bekannten Prinzipien für $\mathbb{N}$. Definieren wir uns dafür die beiden Funktionen $f : \mathbb{N} \to \mathbb{N}'$ und $g : \mathbb{N}' \to \mathbb{N}$ durch

$$\begin{aligned} f(0) &\equiv 0' & \qquad g(0') &\equiv 0 \\ f(\text{succ}(x)) &\equiv \text{succ}'(f(x)) & \qquad g(\text{succ}'(x)) &\equiv \text{succ}(g(x)) \end{aligned}$$

Dann ist offensichtlich, dass diese beiden Funktionen quasi-invers zueinander sind, da gilt:

$$\Pi_{(n':\mathbb{N}')}(f(g(n')) =_{\mathbb{N}'} n') \qquad \Pi_{(n:\mathbb{N})}(g(f(n)) =_{\mathbb{N}} n)$$

Beweis. Um zu zeigen, dass $\Pi_{(n':\mathbb{N}')}(f(g(n')) =_{\mathbb{N}'} n')$ gilt nutzen wir also das Induktionsprinzip über $\mathbb{N}'$. Dafür zeigen wir zuerst die propositionale Gleichheit für $0' : \mathbb{N}$, wofür gilt:

$$f(g(0')) \equiv f(0) \equiv 0'$$

Gelte nun $f(g(n')) = n'$ für ein $n' : \mathbb{N}'$. Dann folgt

$$f(g(\text{succ}'(n'))) \equiv f(\text{succ}(g(n'))) \equiv \text{succ}'(f(g(n'))) \equiv \text{succ}'(n')$$

Also gilt propositionale Gleichheit nach dem Induktionsprinzip für alle $n' : \mathbb{N}'$.

Um zu zeigen, dass $\Pi_{(n:\mathbb{N})}(g(f(n)) =_{\mathbb{N}} n)$ gilt, nutzen wir Induktion über $\mathbb{N}$. Dafür beginnen wir mit

$$g(f(0)) \equiv g(0') = 0$$

und nehmen nun an, dass $g(f(n)) = (n)$ gilt für ein $n : \mathbb{N}$. Dann folgt

$$g(f(\mathrm{succ}(n))) \equiv g(\mathrm{succ}'(f(n))) \equiv \mathrm{succ}(g(f(n))) \equiv \mathrm{succ}(n)$$

wodurch auch die zweite Aussage bewiesen ist.

□

Also sind $f \circ g \sim \mathrm{id}_{\mathbb{N}'}$ und $g \circ f \sim \mathrm{id}_{\mathbb{N}}$ Homotopien. Mit der in Kapitel 2.1 kennen gelernten Definition der Äquivalenzen von einem in einen anderen Typ, wissen wir folglich, dass der Typ $\mathbb{N} \simeq \mathbb{N}'$ existiert. Das Univalenz-Axiom liefert uns wiederum eine Funktion

$$\mathrm{ua} : (\mathbb{N} \simeq_{\mathcal{U}} \mathbb{N}') \to (\mathbb{N} =_{\mathcal{U}} N),$$

die uns letzten Endes bestätigt, dass die beiden Typen $\mathbb{N}$ und $\mathbb{N}'$ gleich sind und somit immer nach Belieben ausgetauscht werden können.

Beispiel 2.2.2: Definieren wir uns eine Funktion $\mathrm{double} : \mathbb{N} \to \mathbb{N}$ durch

$$\begin{aligned} \mathrm{double}(0) &\equiv 0 \\ \mathrm{double}(\mathrm{succ}(n)) &\equiv \mathrm{succ}(\mathrm{succ}(\mathrm{double}(n))), \end{aligned}$$

Dann bringen die oben definierten Funktionen $f : \mathbb{N} \to \mathbb{N}'$ und $g : \mathbb{N}' \to \mathbb{N}$ zusammen mit der Funktion double' : $\mathbb{N}' \to \mathbb{N}'$ definiert durch

$$\begin{aligned} \mathrm{double}'(0') &\equiv 0' \\ \mathrm{double}'(\mathrm{succ}'(n')) &\equiv (f \circ \mathrm{double} \circ g)(n') \end{aligned}$$

das nachfolgende Diagramm zum kommutieren.

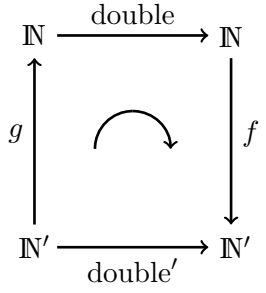

Abbildung 8

Betrachten wir noch einmal, was wir jetzt tatsächlich gemacht haben: Wir haben uns einen weiteren Typ induktiv definiert, mit dem wir auch genauso arbeiten können und der genauso aussieht, wie ein induktiv definierter Typ, den wir bereits kannten. Doch mithilfe von zwei quasi-inversen Funktionen haben wir dann gezeigt, dass beide Typen doch gleich sind, was eben genau bedeutet, dass dieser induktiv definierte Typ eindeutig ist. Da wir dies für jeglichen induktiv definierten Typ tun könnten, liefert uns dies somit die Eindeutigkeit induktiver Typen.

Jetzt wollen wir noch anhand eines Beispiels zeigen, dass es auch reicht, das ein Induktionsprinzip das andere impliziert, damit beide Typen gleich sind.

Dafür betrachten wir den Typ $\mathrm{List}(\mathbf{1})$, dessen Objekte einelementige Listen über den in Kapitel 1.2.7 kennengelernten Einheitstyp $\mathbf{1}$ sind. Diesen definieren wir uns über ein Objekt

$$\mathrm{nil} : \mathrm{List}(\mathbf{1})$$

und eine Funktion

$$\mathrm{cons} : \mathbf{1} \to \mathrm{List}(\mathbf{1}) \to \mathrm{List}(\mathbf{1})$$

als Konstruktoren. Dann würde uns das daraus entstehende Induktionsprinzip sagen, dass wir für eine Familie von Typen $P : \mathrm{List}(\mathbf{1}) \to \mathcal{U}$ die zwei Objekte $e_{\mathrm{nil}} : P(\mathrm{nil})$ und $e_{\mathrm{cons}} : \Pi_{(l:\mathrm{List}(\mathbf{1}))} P(l) \to P(\mathrm{cons}(\star, l))$ finden müssen, wobei wir bereits benutzt haben, dass jedes Objekt im Einheitstyp gleich $\star : \mathbf{1}$ ist. Das bedeutet wiederum, dass es eine abhängige Funktion $f : \Pi_{(l:\mathrm{List}(\mathbf{1}))} P(l)$ gibt mit

$$\begin{aligned} f(\mathrm{nil}) &\equiv e_{\mathrm{nil}} \\ f(\mathrm{cons}(\star, l)) &\equiv e_{\mathrm{cons}}(\star, l, f(l)) \end{aligned}$$

Definieren wir uns jetzt $0' \equiv \mathrm{nil}$ mit $e_0 : P(0')$ für eine Familie von Typen $P : \mathrm{List}(\mathbf{1}) \to \mathcal{U}$ und eine Funktion $\mathrm{succ}' : \mathrm{List}(\mathbf{1}) \to \mathrm{List}(\mathbf{1})$ durch $\mathrm{succ}'(l) \equiv \mathrm{cons}(\star, l)$ mit $e_s : \Pi_{(l:\mathrm{List}(\mathbf{1}))} P(l) \to P(\mathrm{succ}'(l))$, dann können wir ebenfalls setzen:

$$\begin{aligned} e_{\mathrm{nil}} &\equiv e_0 \\ e_{\mathrm{cons}}(\star, l, x) &\equiv e_s(l, x) \end{aligned}$$

Dann können wir damit folgende Gleichungen aufstellen:

$$\begin{aligned} f(0') \equiv f(\mathrm{nil}) \equiv e_{\mathrm{nil}} \equiv e_0 \\ f(\mathrm{succ}'(l)) \equiv f(\mathrm{cons}(\star, l)) \equiv e_{\mathrm{cons}}(\star, l, f(l)) \equiv e_s(l, f(l)) \end{aligned}$$

Daran sehen wir, dass das Induktionsprinzip für den Typ $\mathrm{List}(\mathbf{1})$ das Induktionsprinzip für den Typ $\mathbb{N}'$ impliziert, wobei wir bereits wissen, dass $\mathbb{N}' \simeq_{\mathcal{U}} \mathbb{N}$ gilt. Daraus können wir dann folgern, dass es eine Funktion $f : \mathrm{List}(\mathbf{1}) \to \mathbb{N}$ gibt, definiert durch:

$$f(\mathrm{nil}) \equiv 0$$
$$f(\mathrm{cons}(\star, l)) \equiv \mathrm{succ}(l)$$

und mit den in Kapitel 2.1 kennen gelernten Axiomen und Definitionen ebenfalls eine Funktion $g : \mathbb{N} \to \mathrm{List}(\mathbf{1})$ definiert durch

$$g(0) \equiv \mathrm{nil}$$
$$g(\mathrm{succ}(l)) \equiv \mathrm{cons}(\star, l)$$

Diese Funktionen veranschaulichen wir an folgendem Diagramm:

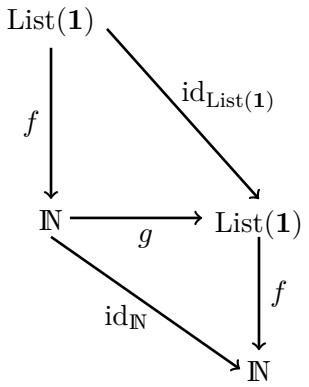

Abbildung 9

Somit sind f und g offensichtlich quasi-invers zueinander, wobei wir $\mathrm{id}_{\mathbb{N}}$ und $\mathrm{id}_{\mathrm{List}(\mathbf{1})}$ bezüglich propositionaler Gleichheit betrachten. Also folgt mit den aus Kapitel 2.1 bekannten Axiomen und Definitionen, dass $\mathrm{List}(\mathbf{1}) \simeq_{\mathcal{U}} \mathbb{N}$ gilt.

2.3 Mengen und univalente Propositionen

Bereits ganz am Anfang dieser Arbeit haben wir gesagt, dass wir den Ausdruck $x : A$ für einen Typ $A : \mathcal{U}$ auch als *"x ist ein Element von A"* lesen können. Das verleitet einen vielleicht dazu, einen Typ A als Menge anzusehen, was es jedoch nicht immer ist. Doch auch in der Typentheorie gibt es Typen, die ebenfalls Mengen, auch 0-Typen genannt, sind. Doch bevor wir zu Mengen kommen, wollen wir noch auf univalente Propositionen eingehen. [1]

Definition 8: univalente Proposition
Gilt für alle $x, y : A$ für einen Typ $A : \mathcal{U}$

$$x = y$$

dann nennen wir A eine univalente Proposition.
Diese Aussage in einen Typ übersetzt lautet:

$$\text{isProp}(A) \equiv \Pi_{(x,y:A)}(x = y)$$

Ein Beispiel für eine univalente Proposition bietet der Einheitstyp $\mathbf{1}$. Rufen wir uns noch einmal Abbildung 2 (vgl. Seite 15) in Erinnerung, so wissen wir, dass es für alle $x : \mathbf{1}$ einen Pfad $p_x : x = \star$ gibt. Definieren wir uns nun $p \equiv p_x \cdot p_y^{-1} : x = y$, können wir so immer einen Pfad $p : x = y$ für alle $x, y : \mathbf{1}$ konstruieren, womit also $\text{isProp}(\mathbf{1})$ nicht leer ist, sodass folgt, dass $\mathbf{1}$ eine univalente Proposition ist.

Lemma 2.3.1: *Gibt es $f : P \to Q$ und $g : Q \to P$ für zwei univalente Propositionen $P, Q : \mathcal{U}$, dann gilt auch $P \simeq_{\mathcal{U}} Q$.*

Beweis. Nachdem P, Q univalente Propositionen sind, also je zwei Objekte $x, y : P$ bzw. $x, y : Q$ gleich sind, gilt hier also $\Pi_{(x:Q)} f(g(x)) = x$ und $\Pi_{(y:P)} g(f(y)) = y$. Somit sind f und g quasi-invers zueinander und es folgt mit den in Kapitel 2.1 kennen gelernten Definition und Axiomen, dass also $P \simeq_{\mathcal{U}} Q$ gilt.

□

Mithilfe dieses Lemmas können wir nun noch einen Schritt weiter gehen und folgende Aussage behaupten:

Lemma 2.3.2: *Es gilt $P \simeq_{\mathcal{U}} \mathbf{1}$, wenn P eine univalente Proposition ist und $x_0 : P$, also P nicht leer ist.*

Beweis. Sei $x_0 : P$ und seien $f : P \to \mathbf{1}$ und $g : \mathbf{1} \to P$ definiert durch

$$f(x) \equiv \star \qquad g(x) \equiv x_0$$

[1] Soweit wir wissen, ist der Begriff ünivalente Proposition"M.Escardó zu verdanken. Im HoTT-Buch wird stattdessen der Begriff "mere proposition"verwendet.

Da P eine univalente Proposition ist und somit alle Elemente $x : P$ propositional gleich $x_0 : P$ sind, reicht es zu zeigen, dass f und g für $\star : \mathbf{1}$, $x_0 : P$ quasi-inverse sind, da $\mathbf{1}$ ebenfalls eine univalente Proposition ist, wie oben bereits erwähnt. Hierfür gilt:

$$g(f(x_0) \equiv g(\star) \equiv x_0$$
$$f(g(\star)) \equiv f(x_0) \equiv \star$$

Also sind f und g tatsächlich quasi-inverse zueinander und mit dem letzten Lemma folgt nun bereits $P \simeq_{\mathcal{U}} \mathbf{1}$.

□

Somit wissen wir jetzt, dass für alle univalenten Propositionen P gilt

$$P \simeq_{\mathcal{U}} \mathbf{1} \text{ oder } P \simeq_{\mathcal{U}} \mathbf{0}.$$

Schließlich wollen wir aber noch auf Mengen in der Typentheorie eingehen.

Definition 9: Menge oder 0-Typ
Wenn für alle $x, y : A$ und alle $p, q : x =_A y$ gilt

$$p = q$$

dann nennen wir den Typ $A : \mathcal{U}$ auch Menge.

Diese Aussage mithilfe der Tabelle am Anfang des ersten Kapitels (vgl. Seite 5) in einen Typ übersetzt lautet dann:

$$\text{isSet}(A) \equiv \Pi_{(x,y:A)}\Pi_{(p,q:x=y)}p = q$$

Ein einfaches Beispiel für eine Menge ist wieder der Einheitstyp $\mathbf{1}$. Wir wissen, dass alle Elemente $x, y : \mathbf{1}$ propositional gleich sind. Wenn also alle Elemente in $\mathbf{1}$ gleich sind, vor allem gleich $\star$, dann müssen dementsprechend auch alle Elemente $p, q : x = y$ gleich sind.

Einen Zusammenhang zwischen univalenten Propositionen und Mengen gibt uns folgendes Lemma:

Lemma 2.3.3: *Ist P eine univalente Proposition, dann ist P auch eine Menge.*

Beweis. Für eine univalente Proposition P sei $f : \text{isProp}(P)$. Dann gilt also für $x, y : P$

$$f(x, y) : x = y$$

Sei nun $x : P$ fest und definieren wir uns $g(y) \equiv f(x, y)$, dann gilt für $y, z : P$ und $p : y = z$

$$\text{apd}_g(p) : p_*(g(y)) = g(z)$$

Stellen wir uns dies anhand einer Skizze wieder vor, erhalten wie folgende Gleichung

$$g(y) \cdot p = g(z)$$

und somit auch

$$p = g(y)^{-1} \cdot g(z)$$

Also gilt für beliebige $x, y : P$ und $p, q : x = y$

$$p = g(x)^{-1} \cdot g(y) = q$$

□

Mit diesem Lemma können wir ein weiteres Beispiel für eine Menge in der Typentheorie anführen. Denn auch die natürlichen Zahlen $\mathbb{N}$ sind eine Menge. Der Typ $x =_{\mathbb{N}} y$ ist eine univalente Proposition und somit gilt $(x =_{\mathbb{N}} y) \simeq_{\mathcal{U}} \mathbf{1}$ oder $(x =_{\mathbb{N}} y) \simeq_{\mathcal{U}} \mathbf{0}$ und sowohl in $\mathbf{1}$ als auch in $\mathbf{0}$ sind alle Objekte gleich. Daher gilt also $p = q$ für jegliche Pfade $p, q : x =_{\mathbb{N}} y$. Und somit ist $\mathbb{N}$ auch eine Menge.

Ausgehend davon können wir nun auch eine Aussage über die Typen isProp und isSet machen, wofür wir aber noch eine weitere Definition benötigen.

Definition 10: 1-Typ
Wenn für alle $x, y : P$ und $p, q : x = y$ und $r, s : p = q$ gilt $r = s$, dann nennen wir p einen 1-Typ.

Lemma 2.3.4: *P ist ein 1-Typ, wenn P eine Menge ist, also wenn P ein 0-Typ ist.*

Beweis. Sei P also eine Menge, was bedeutet, dass es $f : \mathrm{isSet}(P)$ gibt, sodass für $x, y : P$ und $p, q : x = y$ gilt:

$$f(x, y, p, q) : p = q$$

Sei nun $g : \Pi_{(q:x=y)}(p = q)$ für feste x, y, p definiert durch

$$g(q) \equiv f(x, y, p, q)$$

Sei noch $r : q = q'$, dann gilt

$$\mathrm{apd}_g(r) : r_*(g(p)) = g(q')$$

und somit, wenn man es anschaulich betrachtet, gilt auch

$$g(p) \cdot r = g(q')$$

und somit folgt auch $g(p) \cdot s = g(q)$ für $s : p = q$, also gilt natürlich $r = s$.

□

Kommen wir zu den Typen isProp und isSet zurück.

Lemma 2.3.5: *Die Typen* $\mathrm{isProp}(P)$ *und* $\mathrm{isSet}(P)$ *für einen Typ* $A : \mathcal{U}$ *sind univalente Propositionen.*

Beweis. Seien $f, g : \mathrm{isProp}(P)$. Nach Kapitel 2.1 reicht es zu zeigen, dass $f(x) = g(x)$ für alle $x, y : P$ gilt, um zu wissen, dass $f = g$ gilt. Wir wissen, dass P eine univalente Proposition ist und somit auch eine Menge. Somit gilt alle Pfade $p, q : x = y$ für $x, y : P$, dass diese gleich sind. Da $p \equiv f(x, y)$ und $q \equiv g(x, y)$ genau Pfade in P sind, gilt also auch hier

$$f(x, y) = g(x, y)$$

für alle $x, y : P$, womit nun also folgt, dass $f = g$ gilt.
Da P eben eine Menge ist, gilt für $x, y : P$ und $p, q : x = y$

$$p = q$$

Für $f, g : \mathrm{isSet}(P)$ gilt somit außerdem: $f(x, y, p, q) : p = q$ und $g(x, y, p, q) : p = q$. Da wir gezeigt haben, dass für eine Menge P gilt, dass sie auch ein 1-Typ ist, wissen wir, dass gilt:

$$f(x, y, p, q) = g(x, y, p, q)$$

womit nun Kapitel 2.1 wieder $f = g$ liefert, was bedeutet, dass $\mathrm{isSet}(P)$ ebenfalls eine univalente Proposition ist.

□

Am Anfang des ersten Kapitels haben wir eine Tabelle angegeben, wie wir Aussagen in Typen übersetzen können. Ebenfalls haben wir schon einmal erwähnt, dass in der Typentheorie Widerspruchsbeweise nicht möglich sind. Das Prinzip eines Widerspruchsbeweises kann im Allgemeinen durch den Typ $\neg(\neg P) \to P$ dargestellt werden. Dass wir eben eine Widerspruchsbeweis in der Typentheorie nicht führen können, wollen wir nun zeigen.

Theorem 2.3.6: (Coquand) *Nicht für alle* $P : \mathcal{U}$ *gibt es folgenden Typ* $\neg(\neg P) \to P$.

Beweis. Wie wir bereits wissen, gilt $\neg P \equiv P \to \mathbf{0}$. Um nun also zu zeigen, dass der Typ $\neg\neg P \to P$ nicht für alle $P : \mathcal{U}$ existiert, reicht es also, ein Objekt $f : \Pi_{(P:\mathcal{U})}(\neg\neg P \to P)$ zu nehmen und damit dann ein Objekt des Typs $\mathbf{0}$ zu konstruieren.

Dafür nehmen wir zuerst ein $e : \mathbf{2} \simeq_{\mathcal{U}} \mathbf{2}$, das definiert ist durch

$$e(1_{\mathbf{2}}) \equiv 0_{\mathbf{1}} \qquad \text{und} \qquad e(0_{\mathbf{2}}) \equiv 1_{\mathbf{2}}$$

Sei als nächstes $p \equiv \mathrm{ua}(e)$, also $p : \mathbf{2} = \mathbf{2}$ der dazugehörige Pfad, den uns das Univalenzaxiom liefert.

Dann gilt:

$$f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$$

und für $Q \equiv P \to (\neg\neg P \to P)$ gilt dann

$$\mathrm{apd}_f(p) : p_*^Q(p, f(\mathbf{2})) = f(\mathbf{2})$$

Für ein $u : \neg\neg\mathbf{2}$ folgt damit dann

$$\mathrm{happly}(\mathrm{apd}_f(p), u) : p_*^Q(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u)$$

Doch damit gilt wiederum

$$p_*^Q(p, f(\mathbf{2}))(u) = p_*^{P\to P}(p, f(\mathbf{2}))(p_*^{P\to\neg\neg P}(p^{-1}, u))$$

Aufgrund des Extensionalitätsaxiom gilt aber, dass zwei Objekte $u, v : \neg\neg\mathbf{2}$ immer gleich sind, da $u(x) : \mathbf{0}$ gilt für ein $x : \neg(2)$. Denn somit können wir auch immer annehmen, dass $u(x) = v(x)$ gilt. Daher folgt also

$$p_*^{P\to\neg\neg P}(p^{-1}, u) = u$$

womit sich folgenden Gleichheit ergibt

$$p_*^{P\to P}(p.f(\mathbf{2})) = f(\mathbf{2})(u)$$

mit $p \equiv \mathrm{ua}(e)$ folgt wiederum

$$e(f(\mathbf{2})(u)) = f(\mathbf{2})(u).$$

Mit dem Wissen, dass $0_\mathbf{2} \neq 1_\mathbf{2}$ gilt, womit wir zusammen mit der Definition von e zeigen können, dass der Typ

$$\Pi_{(x:\mathbf{2})}\neg(e(x) = x)$$

existiert, können wir uns nun also ein Objekt des Typ $\mathbf{0}$ konstruieren.

Damit haben wir nun angenommen, dass es für jedes $P : \mathcal{U}$ den Typ $\neg\neg P \to P$ gibt und damit ein Objekt des Typs $\mathbf{0}$ konstuiert, was uns die Aussage liefert, dass dieser Typ $\neg\neg P \to P$ nicht für jedes $P : \mathcal{U}$ existiert.

□

Mithilfe dessen können wir nun auch zeigen, dass es spezielle andere Aussagen gibt, die zum Beispiel in der Mengentheorie gelten, die wir aber in Typentheorie nicht einfach für jeden Typ annehmen dürfen, was wir hier noch zum Abschluss dieses Kapitels zeigen wollen.

Theorem 2.3.7: *Nicht für alle $P : \mathcal{U}$ gibt es den Typ $P + (\neg P)$.*

Beweis. Nehmen wir zuerst einmal an, der Typ $\Pi_{(P:\mathcal{U})}(P + (\neg P))$ würde existieren, also es gibt ein $f : \Pi_{(P:\mathcal{U})}(P + (\neg P))$. Dann gilt

$$f(P) : P + (\neg P)$$

und somit muss, wie in Kapitel 1.2.5 bereits kennen gelernt, gelten

$$f(P) \equiv inl(a) \qquad \text{oder} \qquad f(P) \equiv inr(b)$$

für ein $a : P$ und ein $b : \neg P$. So gilt im ersten Fall also $f(P) : P$ und im zweiten Fall $f(P) : \mathbf{0}$, da $\neg P \equiv P \to \mathbf{0}$ gilt.

Angenommen wir haben nun ein $u : \neg\neg P$, dann gilt hierfür

$$u(b) : \mathbf{0}.$$

Für den zweiten Fall können wir als annehmen, was auch immer wir wollen, da $\mathbf{0}$ leer ist, und somit können wir auch annehmen, dass P gilt. Somit haben wir über $u : \neg\neg P$ ein Element aus P konstruiert. Damit hätten wir also gezeigt, dass $\neg\neg P \to P$ für diese beliebige $P : \mathcal{U}$ existiert, was aber nach Theorem 2.3.6 nicht immer so ist. Und somit haben wir unser Ziel erreicht und gezeigt, dass $(P + (\neg P))$ nicht für alle $P : \mathcal{U}$ existiert.

□

3 Generelle Syntax induktiver Typen

Jetzt haben wir bereits einen Blick in die Typentheorie geworfen und einige wichtige Typen kennengelernt und ebenfalls Techniken, mit diesen Typen zu arbeiten. Angefangen vom leeren Typ bis hin zum Prinzip der Pfadinduktion. Doch natürlich sind dies nicht die einzigen existierenden induktiv definierten Typen, denn die Typentheorie gibt uns die Möglichkeit, auch selbst weitere neue Typen induktiv zu konstruieren und zu definieren.

Darauf wollen wir im letzten Abschnitt dieser Arbeit noch einen kleinen Ausblick geben. Wir werden hier noch die Fragen beantworten, wie generell induktiv definierte Typen aufgebaut sind, worauf wir also bei der Definition neuer Typen achten müssen, und wie wir auf die dazu gehörigen Prinzipien kommen.

Doch bevor wir dazu kommen, werden wir noch einen ganz speziellen induktiven Typ betrachten, der uns schließlich auch zur allgemeinen Antwort der vorigen Fragen führt.

Folgende Ideen und Inhalte sind auch in Kapitel 5.3 und Kapitel 5.6 von [6] zu finden.

3.1 W-Typen von Martin-Löf

Der eben schon erwähnte Spezialfall induktiver Typen nennt sich W-Typ. Genauer gesagt werden wir in diesem Kapitel Die W-Typen von Martin Löf genauer betrachten. W-Typen sind eine Verallgemeinerung induktiver Typen wie zum Beispiel $\mathbb{N}$ oder List($\mathbf{1}$).

Einen W-Typ konstruieren wir über einen Typ $A : \mathcal{U}$ und eine Familie von Typen $P : A \to \mathcal{U}$. Mit diesen ergibt sich der W-Typ

$$W_{(x:A)}P(x).$$

So einen W-Typ können wir somit auch als Baum betrachten, wobei die Objekte $a : A$ die Knoten darstellen, von denen jeweils $P(a)$-viele Äste abgehen, die wiederum durch eine Funktion $f : P(a) \to W_{(a:A)}P(a)$ beschriftet werden, indem der b-te von a abgehende Ast für $b : P(a)$ die Beschriftung $f(b)$ erhält.

Insgesamt ergibt sich somit für W-Typen folgender Konstruktor:

$$\text{sup} : \Pi_{(a:A)}(P(a) \to W_{(x:A)}P(x)) \to W_{(x:A)}P(x)$$

Um dies zu veranschaulichen wollen wir nun die natürlichen Zahlen noch einmal neu definieren, diesmal als W-Typ. Unser neuer Typ ist dann definiert durch

$$\mathbf{N}^W \equiv W_{(x:\mathbf{2})}P(x)$$

wobei wir die Familie von Typen $P : A \to \mathcal{U}$ definieren als

$$P(x) \equiv \text{rec}_{\mathbf{2}}(\mathcal{U}, \mathbf{0}, \mathbf{1}, x)$$

So gilt also $P(0_{\mathbf{2}}) \equiv \mathbf{0}$, wenn wir $0_{\mathbf{2}}$ als Konstruktor für das konstante Objekt 0 auffassen, und $P(1_{\mathbf{2}}) \equiv \mathbf{1}$, wenn wir $1_{\mathbf{2}}$ als Konstruktor für den Nachfolger für ein gegebenes Objekt $n : \mathbf{N}^W$ auffassen. Zeichnen wir den dazu gehörigen Baum erhalten wir folgende Skizze:

Abbildung 10

Damit können wir uns auch die ersten Objekte dieser Definition der natürlichen Zahlen definieren durch

$$\begin{aligned} 0^W &\equiv \text{sup}(0_{\mathbf{2}}, \lambda x.\text{rec}_{\mathbf{0}}(\mathbf{N}^W, x)) \\ 1^W &\equiv \text{sup}(1_{\mathbf{2}}, \lambda x.0^W) \\ \text{succ}^W &\equiv \lambda n.\, \text{sup}(1_{\mathbf{2}}, \lambda x.n) \end{aligned}$$

Schauen wir uns dies genauer an, sagt uns diese Definition, dass das Objekt 0^W als Konstruktor $0_{\mathbf{2}}$ hat und dann über Rekursion über den leeren Typ definiert ist, wobei dies eine Funktion ist, die auf dem leeren Typ definiert ist. In Kapitel 1.2.6 haben wir aber bereits gesehen, dass es in $\mathbf{0}$ keine Objekte gibt, auf denen wir diese Funktion definieren müssten. Das Objekt 1^W ist der Nachfolger einer natürlichen Zahl, nämlich der Nachfolger der 0^W, womit wir für sie nun $1_{\mathbf{2}}$ als Konstruktor haben und 0^W als Vorgänger. Dasselbe Prinzip nutzen wir ebenso für die Nachfolgerabbildung, deren Konstruktor eben $1_{\mathbf{2}}$ ist und die den Nachfolger einer natürlichen Zahl $n : \mathbf{N}^W$ gibt.

Ausgehend davon können wir uns jetzt auch das Rekursions- und Induktionsprinzip für W-Typen erschließen:

Rekursionsprinzip:

$$\mathrm{rec}_{W_{(x:A)}B(x)} : \Pi_{(P:\mathcal{U})} P \to W_{(x:A)}B(x) \to P$$

mit der Gleichung

$$\mathrm{rec}_{W_{(x:A)}B(x)}(P, c, \sup(a, f)) \equiv c(a, f, \lambda b.\mathrm{rec}_{W_{(x:A)}B(x)}(P, c, f(b)))$$

wobei $a : A$, $f : B(a) \to W_{(x:A)}B(x)$ und $c : A \to (B(a) \to W_{(x:A)}B(x)) \to P$ gilt.

Induktionsprinzip:

$$\mathrm{ind}_{W_{(x:A)}B(x)} :$$
$$\Pi_{(P:W_{(x:A)}B(x)\to\mathcal{U})}\Pi_{(a:A)}\Pi_{(f:B(a)\to W_{(x:A)}B(x))}\Pi_{(b:B)}$$
$$P(f(b)) \to P(\sup(a, f))$$

Um zu zeigen, dass eine Aussage für alle Objekte des W-Typs, gilt, nehmen wir an, dass sie für alle $f(b)$ mit $b : B(a)$ gilt, wobei $f : B(a) \to W_{x:A}B(x)$, und zeigen damit dann, dass sie auch für $\sup(a, f)$ gilt, was den Beweis vervollständigt.

Nun wollen wir noch zeigen, dass auch das Induktionsprinzip für W-Typen eindeutig ist.

Lemma 3.1.1: *Für zwei abhängige Funktionen* $g, h : \Pi_{(w:W_{(x:A)}B(x))}P(x)$ *mit*

$$\Pi_{(a:A)}\Pi_{(f:B(a)\to W_{(x:A)}P(x))}g(\sup(a, f)) = c(a, f, \lambda b.g(f(b)))$$
$$\Pi_{(a:A)}\Pi_{(f:B(a)\to W_{(x:A)}P(x))}h(\sup(a, f)) = c(a, f, \lambda b.h(f(b)))$$

mit $c : \Pi_{(a:A)}\Pi_{(f:B(a)\to W_{(x:A)}P(x))}\left(\Pi_{(b:B(a))}P(f(b))\right) \to P(\sup(a, f))$ *gilt, dass* g *und* h *gleich sind.*

Beweis. Um zu zeigen, dass $g = h$ gilt, nutzen wir das Induktionsprinzip für W-Typen und definieren uns die Familie von Typen $P : W_{(x:A)}B(x) \to \mathcal{U}$ durch:

$$P(x) \equiv (f(x) = g(x))$$

Nehmen wir nun also an, dass $P(f(b))$ für ein $b : B(a)$ mit $a : A$ gilt, also dass gilt:

$$g(f(b)) = h(f(b))$$

Dann müssen wir nach dem Induktionsprinzip für W-Typen zeigen, dass auch $g(\text{sup}(a, f)) = h(\text{sup}(a, f))$ gilt. Dafür folgt aber sofort mit unserer Annahme:

$$g(\text{sup}(a, f)) = c(a, f, \lambda b.g(f(b))) = c(a, f, \lambda b.h(f(b))) = h(\text{sup}(a, f))$$

Damit ist die Aussage bewiesen, dass g und h gleich sind.

□

3.2 Syntax der Konstruktoren

Grundsätzlich besteht die Definition eines induktiven Typs W immer aus einer endlichen Liste von Konstruktoren. Diese sind Funktionentypen, sowohl abhängige wie auch unabhängige Funktionen sind möglich, die beliebig viele Argumente haben können.

Wollen wir nun einen Typ W induktiv definieren, so wird durch die Funktion

$$f : \mathbf{1} \to W$$

definiert durch

$$f(\star) \equiv w \text{ mit } w : W$$

$f(\star) \equiv w$ ein konstantes Objekt w des Typs W konstruiert. Wollen wir dem Konstruktor mehrere Argumente geben, sieht dieser wie folgt aus:

$$g : A \to W$$

für ein $A : \mathcal{U}$, wobei A ebenfalls wieder ein Funktionentyp sein kann. Ein wichtiger Aspekt hierbei ist, dass Konstruktoren immer strikt positiv sein müssen. Bildet ein Konstruktor also von einem Typ A in den Typ W ab und ist A ebenfalls ein Funktionentyp, also $A \simeq_{\mathcal{U}} (B \to C)$, dann darf der eben definierte Typ W niemals Definitionsmenge einer solchen Funktion sein, was heißt, dass $B \simeq_{\mathcal{U}} W$ nicht gelten darf.

Einfacher gesagt: Ein Konstruktor f ist entweder vom Typ $W \to W$, $A \to W$ oder $A \to W \to W$, wobei A wieder ein Funktionentyp sein kann, bei dem W nicht auf der linken Seite eines Pfeils stehen darf.

Beispiel 3.2.1: Sei W ein induktiv definierter Typ.

a) Ein Konstruktor $f : (W \to A) \to W$ mit einem Typ $A : \mathcal{U}$ ist nicht zulässig.

b) Im Gegensatz dazu ist ein Konstruktor $g : (B \to W) \to W$ mit einem Typ $B : \mathcal{U}$ zulässig.

Konstruktoren mit unendlich vielen Argumenten sind ebenfalls erlaubt, also zum Beispiel ein Konstruktor $f : (\mathbb{N} \to W) \to W$. Wir können diesen Konstruktor auch als Funktion auffassen, die auf unendlichen Folgen von Objekten des Typs W definiert ist.

Ein Konstruktor kann natürlich ebenfalls eine abhängige Funktion

$$f : (\Pi_{(x:A)} P(x)) \to W$$

mit $P : A \to \mathcal{U}$ und $A : \mathcal{U}$ sein. Hierfür gelten dieselben Regeln wie für unabhängige Funktionentypen, was wir daraus folgern können, dass eine abhängige Funktion $f : (\Pi_{(x:A)} P(x)) \to W$ für eine konstante Familie von Typen $P : \mathcal{U}$ einer unabhängigen Funktion $f : A \to P \to W$ entspricht. Auch hier darf also nicht $A \simeq_{\mathcal{U}} W$ gelten.

3.3 Syntax der Prinzipien

Haben wir uns schließlich einen Typ W induktiv durch die Konstruktoren $w_1, ..., w_n$ mit $n : \mathbb{N}$ definiert, wobei für diese gilt:

$$w_i : W_i \to W$$

mit $W_i : \mathcal{U}$, müssen wir dafür auch noch die passenden Rekursions- und Induktionsprinzipien definieren, um mit dem Typ W arbeiten zu können.

Das Rekursionsprinzip liefert uns eine Möglichkeit einen konstanten Typ $P : \mathcal{U}$ für den Typ W zu zeigen.

Dies bringt uns für den Typ W zu folgendem Prinzip:

Rekursionsprinzip:

$$\text{rec}_W : (W_1 \to W) \to (W_1 \to P) \to ... \to (W_n \to W) \to (W_n \to P) \to P$$

mit der Gleichung

$$f(w) \equiv d(w_1, f \circ w_1, ..., w_n, f \circ w_n)$$

wobei $f : W \to P$ und $w : W$ gilt und d den Beweis für P darstellt.

Das Induktionsprinzip dagegen nutzen wir, wenn wir eine Familie von Typen $P : W \to \mathcal{U}$ gegeben haben. Hier müssen wir also $P(w)$ für jedes Objekt $w : W$ zeigen, wobei uns genau das Prinzip der Induktion sagt, dass es genügt, $P(w_i(x))$ für $x : W_i$ für alle Konstruktoren $w_i : W_i \to W$ zu zeigen.

Daraus entsteht folgendes Prinzip:

Induktionsprinzip:

$$\begin{aligned} d : \Pi_{(w_1:W_1 \to W)} \left(\Pi_{(x_1:W_1)} P(w_1(x_1)) \right) \to ... \\ \to \Pi_{(w_n:W_n \to W)} \left(\Pi_{(x_n:W_n)} P(w_n(x_n)) \right) \\ \to \Pi_{(w:W)} P(w) \end{aligned}$$

mit der Gleichung

$$f(w) \equiv d(w_1, f \circ w_1, ..., w_n, f \circ w_n)$$

wobei $f : W \to P$.

Innerhalb der Induktion nehmen wir immer wieder $f(x_i)$ für $x_i : W_i$ an, um mithilfe dessen dann $f(w)$ mit $w = w_i(x_i)$ zu zeigen.

Mit all diesen Regeln sind wir jetzt am Ende dieser Arbeit in der Lage, mit den grundlegenden Typen der Typentheorie und Martin Löf's W-Typen zu arbeiten, und ebenso neue Typen induktiv zu konstruieren und passend dazu alle nötigen Prinzipien zu definieren, um mit diesem neuen Typ genauso arbeiten zu können wie mit den bereits bekannten Typen der Typentheorie.

So hilft uns die Typentheorie mittlerweile gut funktionierende Beweisassistenten als Computerprogramme zu schreiben. Mit diesem kann letzten Endes auch die Kunst, einen Beweis richtig zu führen, von einem Computer überprüft werden, was bisher größtenteils noch allein durch menschlichen Verstand geschieht.

Literaturverzeichnis

[1] S. Awodey, M. A. Warren. Homotopy theoretic models of identity types, Mathematical Proceedings of the Cambridge Philosopical Society, 2009, 146:45-55.

[2] M. Hofmann, T. Streicher: The groupoid interpretation of tyoe theory, G. Sambin, J. M. Smith (Eds.) *Twenty-fice years of constructive type theory* (Venice, 1995), volume 36 of Oxford Logic Guides, Oxford University Press, 1998, 83-111.

[3] P. Martin-Löf: An intuitionistic theory of types: predicative part, in H. E. Rose and J. C. Sherpherdson (Eds.) *Logic Colloquium'73*, pp.73-118, Horth-Holland, 1975.

[4] P. Martin-Löf: *Intuitionistic type theory: Notes by Giovanni Sambin on a series of lectures given in Padua, June 1980*, Napoli: Bibliopolis, 1984.

[5] P. Martin-Löf: An intuitionistic theory of types, in G. Sambin, J. M. Smith (Eds.) *Twenty-five years of constructive type theory* (Venice, 1995), volume 36 of Oxford Logic Guides, Oxford University Press, 1998, 127-172.

[6] The Univalent Foundations Program: *Homotopy Type Theory: The Univalent Foundations of Mathematics*, Institute for Advanced Study, Princeton, 2013.

[7] V.Voevodsky. A very short note on the homotopy λ-calculus, in http://www.math.ias.edu/ vladimir/Site3/ Univalent_Foundations_files /Hlambda_short_current.pdf, 2006.

[8] Typentheorie, in https://de.wikipedia.org/wiki/Typentheorie.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit eigenständig und ohne fremde Hilfe angefertigt habe. Ich versichere, dass ich keine anderen als die angegebenen Quellen verwendet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahren gewesen ist.

Ort, Datum	Unterschrift