Karlsruhe Institute of Technology

Formalisierung Nichtabelscher Topologie für Homotopietypentheorie

Jakob von Raumer | 23. Juni 2015

KARLSRUHER INSTITUT FÜR TECHNOLOGIE

(Abhängige) Typen

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie“)
- Typen modellieren Mengen *und* Aussagen, „propositions as types“
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A“
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ ($i \in \mathbb{N}$),
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \dots$.

(Abhängige) Typen

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie")
- Typen modellieren Mengen *und* Aussagen, „propositions as types"
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A"
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ ($i \in \mathbb{N}$),
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \ldots$.

(Abhängige) Typen

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie“)
- Typen modellieren Mengen *und* Aussagen, „propositions as types“
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A“
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ ($i \in \mathbb{N}$),
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \ldots$.

(Abhängige) Typen

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie“)
- Typen modellieren Mengen *und* Aussagen, „propositions as types“
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A“
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ $(i \in \mathbb{N})$,
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \ldots$.

(Abhängige) Typen

Karlsruhe Institute of Technology

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie")
- Typen modellieren Mengen *und* Aussagen, „propositions as types"
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A"
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ $(i \in \mathbb{N})$,
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \ldots$.

(Abhängige) Typen

- Erste Verwendung von Typentheorie als logisches Fundament der Mathematik: B. Russell, 1903
- Abhängige Typen: Im Wesentlichen von Per Martin-Löf („Intuitionistische Typentheorie“)
- Typen modellieren Mengen *und* Aussagen, „propositions as types“
- Notation: $x : A$ für „x ist eine Instanz / ein Objekt des Typs A“
- Typen sind Objekte in einem *Universum*: $A : \mathcal{U}_i$ ($i \in \mathbb{N}$),
- welche wieder Objekte in Universen sind: $\mathcal{U}_1 : \mathcal{U}_2 : \ldots$.

(Abhängige) Typen

- λ-Kalkül: Für $A : \mathcal{U}_i$, $B : \mathcal{U}_j$ existiert der Typ $A \to B : \mathcal{U}_{\max\{i,j\}}$ der *(nicht abhängigen) Funktionen* von A nach B.
- β-Reduktion: $(\lambda x.\phi)(a) \equiv \phi[a/x]$
- η-Konversion: $(\lambda x.f(x)) \equiv f$
- Spezialfall: $C : A \to \mathcal{U}$ heißt *Typfamilie* über A.
- Interpretation von Typfamilien: Mengenwertige Funktionen, Aussagen mit freier Variable

(Abhängige) Typen

- λ-Kalkül: Für $A : \mathcal{U}_i$, $B : \mathcal{U}_j$ existiert der Typ $A \to B : \mathcal{U}_{\max\{i,j\}}$ der *(nicht abhängigen) Funktionen* von A nach B.
- β-Reduktion: $(\lambda x.\phi)(a) \equiv \phi[a/x]$
- η-Konversion: $(\lambda x.f(x)) \equiv f$
- Spezialfall: $C : A \to \mathcal{U}$ heißt *Typfamilie* über A.
- Interpretation von Typfamilien: Mengenwertige Funktionen, Aussagen mit freier Variable

(Abhängige) Typen

- λ-Kalkül: Für $A : \mathcal{U}_i$, $B : \mathcal{U}_j$ existiert der Typ $A \to B : \mathcal{U}_{\max\{i,j\}}$ der *(nicht abhängigen) Funktionen* von A nach B.
- β-Reduktion: $(\lambda x.\phi)(a) \equiv \phi[a/x]$
- η-Konversion: $(\lambda x.f(x)) \equiv f$
- Spezialfall: $C : A \to \mathcal{U}$ heißt *Typfamilie* über A.
- Interpretation von Typfamilien: Mengenwertige Funktionen, Aussagen mit freier Variable

(Abhängige) Typen

- λ-Kalkül: Für $A : \mathcal{U}_i$, $B : \mathcal{U}_j$ existiert der Typ $A \to B : \mathcal{U}_{\max\{i,j\}}$ der *(nicht abhängigen) Funktionen* von A nach B.
- β-Reduktion: $(\lambda x.\phi)(a) \equiv \phi[a/x]$
- η-Konversion: $(\lambda x.f(x)) \equiv f$
- Spezialfall: $C : A \to \mathcal{U}$ heißt *Typfamilie* über A.
- Interpretation von Typfamilien: Mengenwertige Funktionen, Aussagen mit freier Variable

(Abhängige) Typen

- λ-Kalkül: Für $A : \mathcal{U}_i$, $B : \mathcal{U}_j$ existiert der Typ $A \to B : \mathcal{U}_{\max\{i,j\}}$ der *(nicht abhängigen) Funktionen* von A nach B.
- β-Reduktion: $(\lambda x.\phi)(a) \equiv \phi[a/x]$
- η-Konversion: $(\lambda x.f(x)) \equiv f$
- Spezialfall: $C : A \to \mathcal{U}$ heißt *Typfamilie* über A.
- Interpretation von Typfamilien: Mengenwertige Funktionen, Aussagen mit freier Variable

Abhängige Funktionen (Π-Typen)

- Bildung: Für $A : \mathcal{U}$, $B : A \to \mathcal{U}$ ist $\prod_{x:A} B(x) : \mathcal{U}$.
- Instanziierung: Ähnlich zu nicht-abhängigen Funktionen. Gebe zu jedem $x : A$ ein $f(x) : B(x)$.
- Eliminierung:

$$\Pi\text{-}\textsc{Elim}\ \frac{f : \prod_{a:A} B(a) \qquad x : A}{f(x) : B(x)}$$

- Interpretation: Auswahlfunktion, Allquantor

Abhängige Funktionen (Π-Typen)

- Bildung: Für $A : \mathcal{U}$, $B : A \to \mathcal{U}$ ist $\prod_{x:A} B(x) : \mathcal{U}$.
- Instanziierung: Ähnlich zu nicht-abhängigen Funktionen. Gebe zu jedem $x : A$ ein $f(x) : B(x)$.
- Eliminierung:

$$\Pi\text{-}\textsc{Elim}\ \frac{f : \prod_{a:A} B(a) \qquad x : A}{f(x) : B(x)}$$

- Interpretation: Auswahlfunktion, Allquantor

Abhängige Funktionen (Π-Typen)

- Bildung: Für $A : \mathcal{U}$, $B : A \to \mathcal{U}$ ist $\prod_{x:A} B(x) : \mathcal{U}$.
- Instanziierung: Ähnlich zu nicht-abhängigen Funktionen. Gebe zu jedem $x : A$ ein $f(x) : B(x)$.
- Eliminierung:

$$\Pi\text{-}\textsc{Elim}\ \frac{f : \prod_{a:A} B(a) \qquad x : A}{f(x) : B(x)}$$

- Interpretation: Auswahlfunktion, Allquantor

Abhängige Funktionen (Π-Typen)

Karlsruhe Institute of Technology

- Bildung: Für $A : \mathcal{U}$, $B : A \to \mathcal{U}$ ist $\prod_{x:A} B(x) : \mathcal{U}$.
- Instanziierung: Ähnlich zu nicht-abhängigen Funktionen. Gebe zu jedem $x : A$ ein $f(x) : B(x)$.
- Eliminierung:

$$\Pi\text{-}\textsc{Elim}\ \frac{f : \prod_{a:A} B(a) \qquad x : A}{f(x) : B(x)}$$

- Interpretation: Auswahlfunktion, Allquantor

Abhängige Paare (Σ-Typen)

- Bildung:

$$\Sigma\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{(\sum_{a:A} B(a)) : \mathcal{U}}$$

- Instanziierung:

$$\Sigma\text{-}\textsc{Intro}\ \frac{a : A \qquad b : B(a)}{(a, b) : \sum_{a:A} B(a)}$$

- Eliminierung: Projektionen?

$$\frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_1(q) : A} \qquad \frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_2(q) : B(\mathrm{pr}_1(q))}$$

Abhängige Paare (Σ-Typen)

- Bildung:

$$\Sigma\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{(\sum_{a:A} B(a)) : \mathcal{U}}$$

- Instanziierung:

$$\Sigma\text{-}\textsc{Intro}\ \frac{a : A \qquad b : B(a)}{(a, b) : \sum_{a:A} B(a)}$$

- Eliminierung: Projektionen?

$$\frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_1(q) : A} \qquad \frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_2(q) : B(\mathrm{pr}_1(q))}$$

Abhängige Paare (Σ-Typen)

- Bildung:

$$\Sigma\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{(\sum_{a:A} B(a)) : \mathcal{U}}$$

- Instanziierung:

$$\Sigma\text{-}\textsc{Intro}\ \frac{a : A \qquad b : B(a)}{(a, b) : \sum_{a:A} B(a)}$$

- Eliminierung: Projektionen?

$$\frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_1(q) : A} \qquad \frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_2(q) : B(\mathrm{pr}_1(q))}$$

Abhängige Paare (Σ-Typen)

- Bildung:

$$\Sigma\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{(\sum_{a:A} B(a)) : \mathcal{U}}$$

- Instanziierung:

$$\Sigma\text{-}\textsc{Intro}\ \frac{a : A \qquad b : B(a)}{(a, b) : \sum_{a:A} B(a)}$$

- Eliminierung: Projektionen?

$$\frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_1(q) : A} \qquad \frac{q : \sum_{a:A} B(a)}{\mathrm{pr}_2(q) : B(\mathrm{pr}_1(q))}$$

Abhängige Paare (Σ-Typen)

- Bildung:

$$\Sigma\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad B : A \to \mathcal{U}}{(\sum_{a:A} B(a)) : \mathcal{U}}$$

- Instanziierung:

$$\Sigma\text{-}\textsc{Intro}\ \frac{a : A \qquad b : B(a)}{(a,b) : \sum_{a:A} B(a)}$$

- Eliminierung: ~~Projektionen~~ Abhängige „Induktionsregel".

$$\Sigma\text{-}\textsc{Elim}\ \frac{\begin{array}{c} C : (\sum_{a:A} B(a)) \to \mathcal{U} \\ p : \prod_{a:A} \prod_{b:B(a)} C((a,b)) \qquad x : \sum_{a:A} B(a) \end{array}}{\mathrm{ind}_{(\sum_{a:A} B(a))}(C, p, x) : C(x)}$$

Abhängige Paare (Σ-Typen)

- Logische Interpretation: Konstruktiver Existenzquantor
- Iteriert für die Bildung von Records verwendet

Weitere induktive Datentypen

- Leerer Typ **0**, Einheitstyp **1**, Boolean **2**
- Koprodukt $A + B$
- Natürliche Zahlen $\mathbb{N}$, Listen, Vektoren
- Allgemeiner Kalkül induktiver Typen und Typfamilien

Weitere induktive Datentypen

- Leerer Typ **0**, Einheitstyp **1**, Boolean **2**
- Koprodukt $A + B$
- Natürliche Zahlen $\mathbb{N}$, Listen, Vektoren
- Allgemeiner Kalkül induktiver Typen und Typfamilien

Weitere induktive Datentypen

- Leerer Typ **0**, Einheitstyp **1**, Boolean **2**
- Koprodukt $A + B$
- Natürliche Zahlen $\mathbb{N}$, Listen, Vektoren
- Allgemeiner Kalkül induktiver Typen und Typfamilien

Weitere induktive Datentypen

- Leerer Typ **0**, Einheitstyp **1**, Boolean **2**
- Koprodukt $A + B$
- Natürliche Zahlen $\mathbb{N}$, Listen, Vektoren
- Allgemeiner Kalkül induktiver Typen und Typfamilien

Gleichheit

Karlsruhe Institute of Technology

- Bisher: Durch β-, η-Regeln, Eliminierungsregeln
- Bisher stets *entscheidbar*!
- „propositions as types" $\rightsquigarrow$ Gleichheit sollte ein Typ sein.
- Definiere für $A : \mathcal{U}$ die Typfamilie $\_ =_A \_ : A \to A \to \mathcal{U}$ als induktiv über dem Typ Konstruktor

$$\mathrm{refl} : \prod_{x:A} x =_A x.$$

Gleichheit

- Bisher: Durch β-, η-Regeln, Eliminierungsregeln
- Bisher stets *entscheidbar*!
- „propositions as types“ $\rightsquigarrow$ Gleichheit sollte ein Typ sein.
- Definiere für $A : \mathcal{U}$ die Typfamilie $\_ =_A \_ : A \to A \to \mathcal{U}$ als induktiv über dem Typ Konstruktor

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

Gleichheit

Ableitungsregeln für propositionelle Gleichheit:

$$=\text{-}\textsc{Form}\ \frac{A : \mathcal{U} \qquad a, b : A}{a =_A b : \mathcal{U}} \qquad =\text{-}\textsc{Intro}\ \frac{A : \mathcal{U} \qquad a : A}{\mathsf{refl}_a : a =_A a}$$

$$=\text{-}\textsc{Elim}\ \frac{\begin{array}{c} C : \prod_{a,b:A}(a =_A b) \to \mathcal{U} \\ c : \prod_{a:A} C(a, a, \mathsf{refl}_a) \qquad a, b : A \qquad p : a =_A b \end{array}}{\mathsf{ind}_{=_A}(C, c, a, b, p) : C(x, y, p)}$$

Einfache Folgerungen zur Gleichheit

- Symmetrie und Transitivität der Gleichheitsrelation
- „Transport" von Aussagen: Für eine Typfamilie $C : A \to \mathcal{U}$ und eine Gleichheit $p : x =_A y$ gibt es den *Transport* $p_*(\_) : C(x) \to C(y)$ entlang p.
- Funktionen sind funktoriell bzgl. Gleichheit: Eine Funktion $f : A \to B$ kann auf $p : x =_A y$ *angewendet* werden:

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

Einfache Folgerungen zur Gleichheit

- Symmetrie und Transitivität der Gleichheitsrelation
- „Transport" von Aussagen: Für eine Typfamilie $C : A \to \mathcal{U}$ und eine Gleichheit $p : x =_A y$ gibt es den *Transport* $p_*(\_) : C(x) \to C(y)$ entlang p.
- Funktionen sind funktoriell bzgl. Gleichheit: Eine Funktion $f : A \to B$ kann auf $p : x =_A y$ *angewendet* werden:

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

Einfache Folgerungen zur Gleichheit

- Symmetrie und Transitivität der Gleichheitsrelation
- „Transport" von Aussagen: Für eine Typfamilie $C : A \to \mathcal{U}$ und eine Gleichheit $p : x =_A y$ gibt es den *Transport* $p_*(\_) : C(x) \to C(y)$ entlang p.
- Funktionen sind funktoriell bzgl. Gleichheit: Eine Funktion $f : A \to B$ kann auf $p : x =_A y$ *angewendet* werden:

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

Einfache Folgerungen zur Gleichheit

- Symmetrie und Transitivität der Gleichheitsrelation
- „Transport“ von Aussagen: Für eine Typfamilie $C : A \to \mathcal{U}$ und eine Gleichheit $p : x =_A y$ gibt es den *Transport* $p_*(\_) : C(x) \to C(y)$ entlang p.
- Funktionen sind funktoriell bzgl. Gleichheit: Eine Funktion $f : A \to B$ kann auf $p : x =_A y$ *angewendet* werden:

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

Beweisirrelevanz

- Idee: Gleichheitsbeweise sollten selbst gleich sein: Für $x, y : A$ und $p, q : x =_A y$ sollte $\alpha : p =_{x =_A y} q$ geben.
- Nicht beweisbar
- In traditioneller Martin-Löf-Typentheorie als Axiom postuliert

Äquivalenz von Typen

Eine Funktion $f : A \to B$ heißt *Äquivalenz* zwischen A und B, falls es $g : B \to A$ gibt, sodass existieren:

- $\epsilon : \prod_{x:A} g(f(x)) =_A x$,
- $\eta : \prod_{y:B} f(g(y)) =_B y$ und
- $\tau : \prod_{x:A} \mathrm{ap}_f(\epsilon(x)) =_{f(g(f(x)))=f(x)} \eta(f(x))$.

Setze $A \simeq B :\equiv \sum_{f:A\to B} f$ ist eine Äquivalenz.

Das Univalenzaxiom

- Gleiche Typen sind äquivalent: Für $A, B : \mathcal{U}$ gibt es

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

- Die Gegenrichtung lässt sich *nicht* beweisen.

Axiom (Univalenzaxiom, Voevodsky (2012))

Für $A, B : \mathcal{U}$ ist die Funktion $\mathrm{idtoeqv}_{A,B}$ selbst eine Äquivalenz, insbesondere hat sie eine Inverse $\mathrm{ua}_{A,B} : (A \simeq B) \to (A =_{\mathcal{U}} B)$. Es gilt kurz: $(A \simeq B) \simeq (A =_{\mathcal{U}} B)$.

Das Univalenzaxiom

- Gleiche Typen sind äquivalent: Für $A, B : \mathcal{U}$ gibt es

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

- Die Gegenrichtung lässt sich *nicht* beweisen.

Axiom (Univalenzaxiom, Voevodsky (2012))

Für $A, B : \mathcal{U}$ ist die Funktion $\mathsf{idtoeqv}_{A,B}$ *selbst eine Äquivalenz, insbesondere hat sie eine Inverse* $\mathsf{ua}_{A,B} : (A \simeq B) \to (A =_{\mathcal{U}} B)$*. Es gilt kurz:* $(A \simeq B) \simeq (A =_{\mathcal{U}} B)$*.*

Das Univalenzaxiom

„Problem“: Univalenz ist unvereinbar mit Beweisirrelevanz!
$\rightsquigarrow$ ungleiche Gleichheitsbeweise

Auftritt Topologie

- Geometrie vergleicht Räume mithilfe von Abstandsmaßen, Winkeln, Flächen, ...
- Topologie ignoriert diese und betrachtet nur stetige Verformungen von Räumen
- Zentrale Struktur: Kategorie **Top** der Topologischen Räume
- Isomorphismen heißen Homöomorphismen

Homotopie

- Abbildungen $f, f' : X \to Y$ heißen *homotop*, falls es eine stetige Abb. $H : [0,1] \times X \to Y$ gibt mit

$$H(0,x) = f(x) \text{ und}$$
$$H(1,x) = f'(x) \text{ für alle } x \in X.$$

- $X, Y \in \mathbf{Top}$ heißen *homotopieäquivalent*, falls es $f : X \to Y$ und $g : Y \to X$ gibt sodass $f \circ g$ und $g \circ f$ homotop zur jeweiligen Identität sind.

Homotopie

Karlsruhe Institute of Technology

- Abbildungen $f, f' : X \to Y$ heißen *homotop*, falls es eine stetige Abb. $H : [0,1] \times X \to Y$ gibt mit

$$H(0,x) = f(x) \text{ und}$$
$$H(1,x) = f'(x) \text{ für alle } x \in X.$$

- $X, Y \in \mathbf{Top}$ heißen *homotopieäquivalent*, falls es $f : X \to Y$ und $g : Y \to X$ gibt sodass $f \circ g$ und $g \circ f$ homotop zur jeweiligen Identität sind.

Homotopie

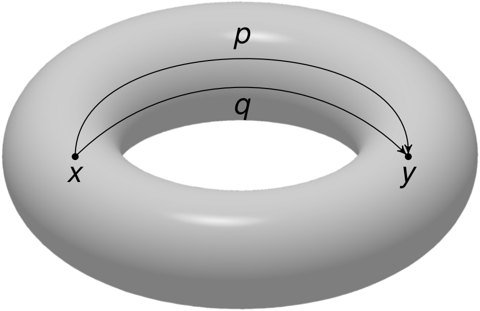
- Äquivalenzklassen von top. Räumen unter Homotopieäquivalenz heißen *Homotopietypen*.

Typen in beweisrelevanter, abh. Typentheorie entsprechen Homotopietypen!

Topologische Interpretation

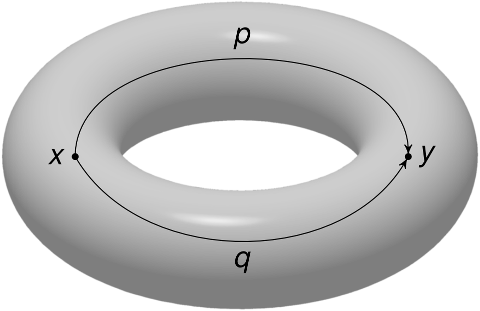
Typentheorie	Topologie
Typ $A : \mathcal{U}$	Raum / Homotopietyp
Funktion $f : A \to B$	Stetige Abbildung
Typfamilie $B : A \to \mathcal{U}$	Faserung
Abhängige Funktion $f : \prod_{a:A} B(a)$	Schnitt der Faserung B
Σ-Typ $\sum_{a:A} B(a)$	Totalraum der Faserung B
Gleichheit $x =_A y$	Raum der Pfade $x \rightsquigarrow y$

Gleichheitsyp



$$p =_{x=y} q$$

Gleichheitsyp



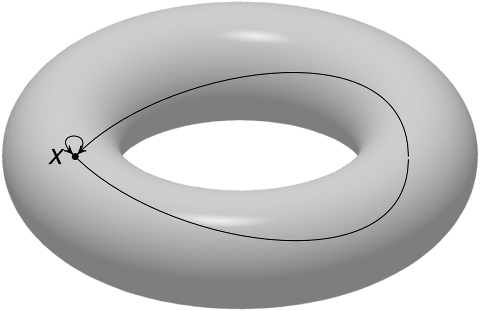
$$p \neq_{x=y} q$$

Homotopieeigenschaften in HoTT

Ein Typ $A : \mathcal{U}$ heißt...

- ...*zusammenziehbar*, falls $\mathrm{isContr}(A) :\equiv \sum_{x:A} \prod_{y:A} x =_A y$,
- ...eine *bloße Aussage*, falls $\mathrm{isProp}(A) :\equiv \prod_{x,y:A} x =_A y$,
- ...eine *Menge* oder
  ein *0-Typ*, falls $\prod_{x,y:A} \prod_{p,q:x=y} p = q$,
- ...ein *1-Typ*, falls $\prod_{x,y:A} \prod_{p,q:x=y} \prod_{\alpha,\beta:p=q} \alpha = \beta$,
- ...ein *2-Typ*, falls $\prod_{x,y:A} \prod_{p,q:x=y} \prod_{\alpha,\beta:p=q} \prod_{\gamma,\delta:\alpha=\beta} \gamma = \delta$,
- ...ein *n-Typ*, falls ...

Die Fundamentalgruppe



Die *Fundamentalgruppe* eines Raumes X in $x \in X$ ist definiert durch

$$\pi_1(X, x) = \{\text{Pfade } x \rightsquigarrow x\}/_{\text{Homotopien, die } x \text{ fix lassen}}$$

$$\pi_1(\text{Torus}, x) \cong \mathbb{Z} \times \mathbb{Z}$$

Fundamentalgruppe eines 1-Typs

Die Fundamentalgruppe eines Typs $A : \mathcal{U}$ in $x : A$,

$$\pi_1(A, x) :\equiv x =_A x,$$

ist für jedes $x : A$ trivial gdw. A eine Menge ist.
$\rightsquigarrow$ Internalisiere algebraische Strukturen und untersuche Typen anhand ihrer charakteristischen algebraischen Objekte!

Fundamentalgruppe eines 1-Typs

Die Fundamentalgruppe eines Typs $A : \mathcal{U}$ in $x : A$,

$$\pi_1(A, x) :\equiv x =_A x,$$

ist für jedes $x : A$ trivial gdw. A eine Menge ist.
$\rightsquigarrow$ Internalisiere algebraische Strukturen und untersuche Typen anhand ihrer charakteristischen algebraischen Objekte!

Höhere Homotopiegruppen

- Verallgemeinerung der Fundamentalgruppe: Die n-te Homotopiegruppe von X ist definiert durch

$$\pi_n(X, x) = \{\text{Stetige Abb. } \mathbb{S}^n \to X \text{ mit } N \mapsto x\}/\ldots$$

Fundamentale Gruppoide

- Zweite Verallgemeinerung der Fundamentalgruppe: Erlaube mehrere Basispunkte
- Für einen top. Raum X und $C \subseteq X$ ist

$$\pi_1(X, C) = \{\text{Pfade mit Endpkt. in } C\}/_{\text{Homotopien mit festen Endpunkten}}$$

- Algebraische Struktur dieser Konstruktion: Gruppoid auf der Menge C

Fundamentale Gruppoide

- Zweite Verallgemeinerung der Fundamentalgruppe: Erlaube mehrere Basispunkte
- Für einen top. Raum X und $C \subseteq X$ ist

  $$\pi_1(X, C) = \{\text{Pfade mit Endpkt. in } C\}/_{\text{Homotopien mit festen Endpunkten}}$$

- Algebraische Struktur dieser Konstruktion: Gruppoid auf der Menge C

Fundamentale Gruppoide

- Zweite Verallgemeinerung der Fundamentalgruppe: Erlaube mehrere Basispunkte
- Für einen top. Raum X und $C \subseteq X$ ist

$$\pi_1(X, C) = \{\text{Pfade mit Endpkt. in } C\}/_{\text{Homotopien mit festen Endpunkten}}$$

- Algebraische Struktur dieser Konstruktion: Gruppoid auf der Menge C

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen hom : $\prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen id : $\prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- idLeft : $\prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- idRight : $\ldots$,
- assoc : $\prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen $\mathrm{hom} : \prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen $\mathrm{id} : \prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- $\mathrm{idLeft} : \prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- $\mathrm{idRight} : \ldots$,
- $\mathrm{assoc} : \prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen hom : $\prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen id : $\prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- idLeft : $\prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- idRight : $\ldots$,
- assoc : $\prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen hom : $\prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen id : $\prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- idLeft : $\prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- idRight : $\ldots$,
- assoc : $\prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen $\mathrm{hom} : \prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen $\mathrm{id} : \prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- $\mathrm{idLeft} : \prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- $\mathrm{idRight} : \ldots$,
- $\mathrm{assoc} : \prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen $\mathrm{hom} : \prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen $\mathrm{id} : \prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- $\mathrm{idLeft} : \prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- $\mathrm{idRight} : \ldots$,
- $\mathrm{assoc} : \prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Kategorien und Gruppoide in HoTT

Eine Kategorie „besteht“ aus

- Einem Träger $A : \mathcal{U}$,
- Morphismen-Typen hom : $\prod_{a,b:A} \mathcal{U}$, ...
- ... welche Mengen sind: $\prod_{a,b:A} \mathrm{isSet}(\mathrm{hom}(a,b))$,
- einer Komposition
  $\circ : \prod_{a,b,c:A} \mathrm{hom}(b,c) \to \mathrm{hom}(a,b) \to \mathrm{hom}(a,c)$,
- Identitätsmorphismen id : $\prod_{a:A} \mathrm{hom}(a,a)$,

und den Eigenschaften

- idLeft : $\prod_{a,b:A} \prod_{f:\mathrm{hom}(a,b)} \mathrm{id}_b \circ f = f$,
- idRight : $\ldots$,
- assoc : $\prod_{a,b,c,d:A} \prod_{f,g,h:\ldots} h \circ (g \circ f) = (h \circ g) \circ f$.

Fundamentaler Gruppoid eines 1-Typs

Sei $X : \mathcal{U}$ ein 1-Typ, $C : \mathcal{U}$ eine Menge und $\iota : C \to X$.

- Setze $\mathrm{hom}(x, y) :\equiv (\iota(x) =_X \iota(y))$,
- Komposition ist die Transitivität der Gleichheit,
- Identitätsmorphismen sind $\mathrm{id}(x) :\equiv \mathrm{refl}_{\iota(x)}$.

Doppelgruppoide

Fragen:

- Wie können wir das Konzept von fundamentalen Gruppoiden auf die zweite Homotopiegruppe erweitern?
- Welche algebraischen Strukturen halten die Information?
  $\rightsquigarrow$ Doppelgruppoide und verschränkte Moduln sind zwei unterschiedliche algebraische Strukturen, die sich als äquivalent erwiesen haben.

Doppelkategorien

Eine *Doppelkategorie* ist eine Kategorie, bei der es neben Objekten und Morphismen auch 2-Zellen oder Quadrate gibt, welche von vier Morphismen begrenzt werden:

$$
\begin{tikzcd}
\partial^-(\partial_1^-(u)) = \partial^-(\partial_2^-(u)) \arrow[r, dash, "\partial_1^-(u)"] \arrow[d, dash, "\partial_2^-(u)"'] \arrow[dr, phantom, "u"] & \partial^+(\partial_1^-(u)) = \partial^-(\partial_2^+(u)) \arrow[d, dash, "\partial_2^+(u)"] \\
\partial^-(\partial_1^-(u)) = \partial^-(\partial_2^-(u)) \arrow[r, dash, "\partial_1^+(u)"'] & \partial^+(\partial_1^+(u)) = \partial^+(\partial_2^+(u))
\end{tikzcd}
$$

Als Typfamilie (D_0 ist der Träger, D_1 die Morphismen):

$$D_2 : \prod_{a,b,c,d:D_0} \prod_{f:D_1(a,b)} \prod_{g:D_1(c,d)} \prod_{h:D_1(a,c)} \prod_{i:D_1(b,d)} \mathcal{U}$$

Doppelkategorien

Eine *Doppelkategorie* ist eine Kategorie, bei der es neben Objekten und Morphismen auch 2-Zellen oder Quadrate gibt, welche von vier Morphismen begrenzt werden:

$$\begin{tikzcd} a \arrow[r, dash, "f"] \arrow[d, dash, "h"'] \arrow[dr, phantom, "u"] & b \arrow[d, dash, "i"] \\ c \arrow[r, dash, "g"'] & d \end{tikzcd}$$

Als Typfamilie (D_0 ist der Träger, D_1 die Morphismen):

$$D_2 : \prod_{a,b,c,d:D_0} \; \prod_{f:D_1(a,b)} \; \prod_{g:D_1(c,d)} \; \prod_{h:D_1(a,c)} \; \prod_{i:D_1(b,d)} \mathcal{U}$$

Doppelkategorien

Quadrate müssen horizontal und vertikal auch eine Kategorie bilden, d.h.

- Es gibt eine vertikale Komposition $\circ_1$ und eine horizontale Komposition $\circ_2$, ...
- ... sowie vertikale und horizontale „Identitätsquadrate“ id_1 und id_2.

$$\circ_1 : \prod_{\dots} D_2(g_1, g_2, h_2, i_2) \to D_2(f, g_1, h_1, i_1) \to D_2(f, g_2, h_2 \circ h_1, i_2 \circ i_1)$$

$$\begin{tikzcd}
\bullet \arrow[r, dash, "f"] \arrow[d, dash, "h_1"'] \arrow[dr, phantom, "u"] & \bullet \arrow[d, dash, "i_1"] \\
\bullet \arrow[r, dash, "g_1"] \arrow[d, dash, "h_2"'] \arrow[dr, phantom, "w"] & \bullet \arrow[d, dash, "i_2"] \\
\bullet \arrow[r, dash, "g_2"'] & \bullet
\end{tikzcd}$$

Doppelkategorien

Karlsruhe Institute of Technology

Quadrate müssen horizontal und vertikal auch eine Kategorie bilden, d.h.

- Es gibt eine vertikale Komposition $\circ_1$ und eine horizontale Komposition $\circ_2$, ...
- ... sowie vertikale und horizontale „Identitätsquadrate“ id_1 und id_2.

$$\mathrm{id}_1 : \prod_{a,b:D_0} \prod_{f:D_1(a,b)} D_2(f, f, \mathrm{id}_a, \mathrm{id}_b)$$

$$\begin{tikzcd}
\bullet \arrow[r, dash, "f"] \arrow[d, equal, "\mathrm{id}_a"'] \arrow[dr, phantom, "\mathrm{id}_1(f)"] & \bullet \arrow[d, equal, "\mathrm{id}_b"] \\
\bullet \arrow[r, dash, "f"'] & \bullet
\end{tikzcd}$$

Doppelkategorien

Quadrate müssen horizontal und vertikal auch eine Kategorie bilden, d.h.

- Identitätsquadrate müssen links- und rechtsneutral sein
- Assoziativität der Komposition von Quadraten

Problem: Für passende w, v, u ist

$$(w \circ_1 (v \circ_1 u)) : D_2(f, g, h_3 \circ (h_2 \circ h_1), i_3 \circ (i_2 \circ i_1)), \text{ aber}$$
$$((w \circ_1 v) \circ_1 u) : D_2(f, g, (h_3 \circ h_2) \circ h_1, (i_3 \circ i_2) \circ i_1).$$

$\rightsquigarrow$ Transportiere entlang des Assoziativitätszeugen im 1-Skelett:

$$\begin{aligned}\mathrm{assoc}_1(\ldots, w, v, u) : \mathrm{assoc}(i_3, i_2, i_1)_*(\mathrm{assoc}(h_3, h_2, h_1)_*(w \circ_1 (v \circ_1 u))\\ = ((w \circ_1 v) \circ_1 u)\end{aligned}$$

Doppelkategorien

Quadrate müssen horizontal und vertikal auch eine Kategorie bilden, d.h.

- Identitätsquadrate müssen links- und rechtsneutral sein
- Assoziativität der Komposition von Quadraten

Problem: Für passende w, v, u ist

$$(w \circ_1 (v \circ_1 u)) : D_2(f, g, h_3 \circ (h_2 \circ h_1), i_3 \circ (i_2 \circ i_1)), \text{ aber}$$
$$((w \circ_1 v) \circ_1 u) : D_2(f, g, (h_3 \circ h_2) \circ h_1, (i_3 \circ i_2) \circ i_1).$$

$\rightsquigarrow$ Transportiere entlang des Assoziativitätszeugen im 1-Skelett:

$$\begin{aligned}\mathrm{assoc}_1(\ldots, w, v, u) : \mathrm{assoc}(i_3, i_2, i_1)_*(\mathrm{assoc}(h_3, h_2, h_1)_*(w \circ_1 (v \circ_1 u)) \\ = ((w \circ_1 v) \circ_1 u)\end{aligned}$$

Doppelkategorien

Quadrate müssen horizontal und vertikal auch eine Kategorie bilden, d.h.

- Identitätsquadrate müssen links- und rechtsneutral sein
- Assoziativität der Komposition von Quadraten

Problem: Für passende w, v, u ist

$$(w \circ_1 (v \circ_1 u)) : D_2(f, g, h_3 \circ (h_2 \circ h_1), i_3 \circ (i_2 \circ i_1)), \text{ aber}$$
$$((w \circ_1 v) \circ_1 u) : D_2(f, g, (h_3 \circ h_2) \circ h_1, (i_3 \circ i_2) \circ i_1).$$

$\rightsquigarrow$ Transportiere entlang des Assoziativitätszeugen im 1-Skelett:

$$\begin{aligned}\mathrm{assoc}_1(\ldots, w, v, u) : \mathrm{assoc}(i_3, i_2, i_1)_*(\mathrm{assoc}(h_3, h_2, h_1)_*(w \circ_1 (v \circ_1 u)) \\ = ((w \circ_1 v) \circ_1 u)\end{aligned}$$

Doppelkategorien

Weitere Axiome, u.a.:

- Identitätsquadrate sollten im folgenden Sinne (und im transponierten Fall) distributiv sein:

$$\prod_{a,b,c:D_0} \; \prod_{f:D_1(a,b)} \; \prod_{g:D_1(b,c)} \mathrm{id}_1(g \circ f) = \mathrm{id}_1(g) \circ_2 \mathrm{id}_1(f)$$

$$\begin{tikzcd} \bullet \arrow[r, "f"] \arrow[d, equal, "\mathrm{id}_a"'] & \bullet \arrow[r, "g"] \arrow[d, equal, "\mathrm{id}_b"'] & \bullet \arrow[d, equal, "\mathrm{id}_c"] \\ \bullet \arrow[r, "f"'] & \bullet \arrow[r, "g"'] & \bullet \end{tikzcd}$$

Doppelkategorien

Weitere Axiome, u.a.:

- Identitätsquadrate sollten im folgenden Sinne (und im transponierten Fall) distributiv sein:

$$\prod_{a,b,c:D_0} \prod_{f:D_1(a,b)} \prod_{g:D_1(b,c)} \mathrm{id}_1(g \circ f) = \mathrm{id}_1(g) \circ_2 \mathrm{id}_1(f)$$

- Identitätsquadrate über Identitätsmorphismen sind eindeutig:

$$\prod_{a:D_0} \mathrm{id}_1(\mathrm{id}_a) =_{D_2(\mathrm{id}_a,\mathrm{id}_a,\mathrm{id}_a,\mathrm{id}_a)} \mathrm{id}_2(\mathrm{id}_a)$$

Thin Structures auf Doppelkategorien

Sei $D = (D_0, D_1, D_2)$ eine Doppelkategorie. Eine *Thin Structure* ist eine Teilmenge der Quadrate in D_2, sodass

- für jeden möglichen kommutierenden Rand genau ein Quadrat als *dünn* ausgezeichnet wird:

$$\text{thin} : \prod_{a,b,c,d:D_0} \prod_{f,g,h,i:\dots} (g \circ h = i \circ f) \to D_2(f, g, h, i),$$

- Identitätsquadrate stets dünn sind und
- die Komposition dünner Quadrate dünn ist.

Doppelgruppoide

Ein *Doppelgruppoid* ist eine Doppelkategorie mit Thin Structure, sodass alle drei Kategorien Gruppoide sind.

- Erhalte vertikal und horizontal „gespiegelte“ Quadrate

$$\mathrm{inv}_1 : \prod_{\dots} D_2(f, g, h, i) \to D_2(g, f, h^{-1}, i^{-1})$$

- Doppelgruppoide bilden eine Kategorie **DGpd**, Morphismen sind *Doppelfunktoren*.

Doppelgruppoide

Ein *Doppelgruppoid* ist eine Doppelkategorie mit Thin Structure, sodass alle drei Kategorien Gruppoide sind.

- Erhalte vertikal und horizontal „gespiegelte“ Quadrate

$$\mathrm{inv}_1 : \prod_{\dots} D_2(f, g, h, i) \to D_2(g, f, h^{-1}, i^{-1})$$

- Doppelgruppoide bilden eine Kategorie **DGpd**, Morphismen sind *Doppelfunktoren*.

Doppelgruppoide

Ein *Doppelgruppoid* ist eine Doppelkategorie mit Thin Structure, sodass alle drei Kategorien Gruppoide sind.

- Erhalte vertikal und horizontal „gespiegelte“ Quadrate

$$\mathrm{inv}_1 : \prod_{\ldots} D_2(f, g, h, i) \to D_2(g, f, h^{-1}, i^{-1})$$

- Doppelgruppoide bilden eine Kategorie **DGpd**, Morphismen sind *Doppelfunktoren*.

Fundamentaler Doppelgruppoid eines Tripels von Räumen

Seien $C \subseteq A \subseteq X$ top. Räume. Setze

$$
\begin{aligned}
R(X,A,C)_0 &:= C \\
R(X,A,C)_1 &:= \{\sigma : [0,1] \to A \mid \sigma \text{ stetig}, \sigma(0), \sigma(1) \in C\} \\
R(X,A,C)_2 &:= \Big\{\alpha : [0,1]^2 \to X \mid \alpha \text{ stetig}, \\
&\qquad \alpha(x,y) \in A, \text{ falls } x \text{ oder } y \in \{0,1\}, \\
&\qquad \alpha(x,y) \in C \text{ falls } x \text{ und } y \in \{0,1\}\Big\}.
\end{aligned}
$$

Teile Homotopien heraus, die Ecken fix lassen.

Fundamentaler Doppelgruppoid eines präsentierten Typs

Seien $X, A, C : \mathcal{U}$ mit isSet(C), is-1-type(A) und is-2-type(X).
Seien $\iota : C \to A$ und $\iota' : A \to X$. Definiere

$$G_0 :\equiv C,$$

$$G_1 :\equiv \prod_{a,b:C} \iota(a) =_A \iota(b) \text{ und}$$

$$G_2 :\equiv \prod_{a,b,c,d:C} \prod_{f:G_1(a,b)} \cdots \prod_{i:G_1(b,d)} \mathrm{ap}_{\iota'}(h) \cdot \mathrm{ap}_{\iota'}(g) = \mathrm{ap}_{\iota'}(f) \cdot \mathrm{ap}_{\iota'}(i)$$

Verschränkte Moduln

Sei P ein Gruppoid. Ein *verschränkter Modul* auf P ist eine Familie von Gruppen $(M_p)_{p \in P}$ mit Gruppenhomomorphismen $\mu_p : M_p \to \hom_P(p,p)$ und einer Gruppoidenoperation $\phi : \hom(p,q) \times M_p \to M_q$, sodass:

- $\mu_q(\phi(a,x)) = a \circ \mu_p(x) \circ a^{-1}$ für alle $a \in \hom(p,q)$, $x \in M_p$ und
- $\phi(\mu_p(c), x) = c \cdot x \cdot c^{-1}$ für alle $c, x : M_p$.

Verschränkte Moduln

Ein *Morphismus zwischen verschränkten Moduln* $(P, (M_p))$ und $(Q, (N_q))$ ist ein Funktor $F : P \to Q$ zusammen mit einer Familie von Gruppenhomomorphismen $\psi_p : M_p \to N_{F(p)}$, sodass für alle p das Diagramm

$$\begin{tikzcd}
M_p \arrow[r, "\psi_p"] \arrow[d, "\mu_p"'] & N_{F(p)} \arrow[d, "\mu_{F(p)}"] \\
\hom(p,p) \arrow[r, "F"'] & \hom(F(p), F(p))
\end{tikzcd}$$

kommutiert und für alle $a \in \hom(p, q)$ und $x \in M_p$ gilt:

$$\psi_q(\phi(a, x)) = \phi(F(a), \psi_p(x)) \qquad .$$

Die Äquivalenz

Verschränkte Moduln über Gruppoiden bilden mit den eben definierten Morphismen eine Kategorie **XMod**.
Es gilt:

Theorem (Ronald Brown)

Die Kategorien **XMod** *und* **DGpd** *sind äquivalent.*

Der Theorembeweiser Lean

- Projekt gestartet 2013 von Leonardo de Moura, Microsoft Research
- Emacs-Plugin: Soonho Kong, CMU
- Zwei Modi: Beweisirrelevanz oder HoTT
- Bibliotheken werden hauptsächlich an der CMU geschrieben

Was habe ich formalisiert?

Karlsruhe Institute of Technology

- Teile der HoTT-Basics
- Teile der Kategorientheorie-Bibliothek
- Doppelkategorien, Thin Structures, Doppelgruppoide und Verschränkte Moduln
- Instantiierung des Fundamentalen Gruppoids und Doppelgruppoids
- Große Teile des Äquivalenzbeweises

Statistik

Theorie	Zeilenanzahl	Kompilierungsdauer in s
`transport4`	49	0.333
`dbl_cat.`		
`basic`	224	5.272
`decl`	58	4.253
`dbl_gpd.`		
`basic`	199	6.375
`category_of`	41	1.314
`decl`	69	8.856
`functor`	582	47.997
`fundamental`	1270	21.091
`thin_structure.`		
`basic`	154	3.091
`decl`	33	0.766
`xmod.`		
`category_of`	25	0.327
`decl`	53	0.794
`morphism`	209	4.542

Statistik

Theorie	Zeilenanzahl	Kompilierungsdauer in s
`equivalence.`		
`equivalence`	371	*30.049
`gamma_functor`	51	6.109
`gamma`	164	6.054
`gamma_group`	229	5.973
`gamma_morphisms`	167	5.986
`gamma_mu_phi`	407	24.828
`lambda_functor`	27	4.086
`lambda`	569	16.148
`lambda_morphisms`	70	7.914

Doppelkategorien in Lean

```
structure worm_precat {D₀ : Type} (C  : precategory D₀)
  (D₂ : Π ⦃a b c d : D₀⦄
    (f : hom a b) (g : hom c d) (h : hom a c) (i : hom b d), Type) :=
  (comp₁ : proof Π ⦃a b c₁ d₁ c₂ d₂ : D₀⦄
    ⦃f₁ : hom a b⦄ ⦃g₁ : hom c₁ d₁⦄ ⦃h₁ : hom a c₁⦄ ⦃i₁ : hom b d₁⦄
    ⦃g₂ : hom c₂ d₂⦄ ⦃h₂ : hom c₁ c₂⦄ ⦃i₂ : hom d₁ d₂⦄,
    (D₂ g₁ g₂ h₂ i₂) → (D₂ f₁ g₁ h₁ i₁)
    → (@D₂ a b c₂ d₂ f₁ g₂ (h₂ ∘ h₁) (i₂ ∘ i₁)) qed)
  (ID₁ : proof Π ⦃a b : D₀⦄ (f : hom a b), D₂ f f (ID a) (ID b) qed)
  (assoc₁ : proof Π ⦃a b c₁ d₁ c₂ d₂ c₃ d₃ : D₀⦄
    ⦃f  : hom a b⦄   ⦃g₁ : hom c₁ d₁⦄ ⦃h₁ : hom a c₁⦄ ⦃i₁ : hom b d₁⦄
    ⦃g₂ : hom c₂ d₂⦄ ⦃h₂ : hom c₁ c₂⦄ ⦃i₂ : hom d₁ d₂⦄
    ⦃g₃ : hom c₃ d₃⦄ ⦃h₃ : hom c₂ c₃⦄ ⦃i₃ : hom d₂ d₃⦄
    (w : D₂ g₂ g₃ h₃ i₃) (v : D₂ g₁ g₂ h₂ i₂) (u : D₂ f g₁ h₁ i₁),
    (assoc i₃ i₂ i₁) ▹ ((assoc h₃ h₂ h₁) ▹
        (comp₁ w (comp₁ v u))) = (comp₁ (comp₁ w v) u) qed)
  (id_left₁ : proof Π ⦃a b c d : D₀⦄
    ⦃f : hom a b⦄ ⦃g : hom c d⦄ ⦃h : hom a c⦄ ⦃i : hom b d⦄
    (u : D₂ f g h i),
    (id_left i) ▹ ((id_left h) ▹ (comp₁ (ID₁ g) u)) = u qed)
  (id_right₁ : proof Π ⦃a b c d : D₀⦄
    ⦃f : hom a b⦄ ⦃g : hom c d⦄ ⦃h : hom a c⦄ ⦃i : hom b d⦄
    (u : D₂ f g h i),
    (id_right i) ▹ ((id_right h) ▹ (comp₁ u (ID₁ f))) = u qed)
  (homH' : proof Π ⦃a b c d : D₀⦄
    ⦃f : hom a b⦄ ⦃g : hom c d⦄ ⦃h : hom a c⦄ ⦃i : hom b d⦄,
    is_hset (D₂ f g h i) qed)
```

Doppelkategorien in Lean

```
structure dbl_precat {D₀ : Type} (C : precategory D₀)
  (D₂ : Π ⦃a b c d : D₀⦄
    (f : hom a b) (g : hom c d) (h : hom a c) (i : hom b d), Type)
  extends worm_precat C D₂,
    worm_precat C (λ ⦃a b c d : D₀⦄ f g h i, D₂ h i f g)
  renaming comp₁→comp₂ ID₁→ID₂ assoc₁→assoc₂
    id_left₁→id_left₂ id_right₁→id_right₂ homH'→homH'_dontuse :=
  (id_comp₁ : proof Π {a b c : D₀} (f : hom a b) (g : hom b c),
    ID₂ (g ∘ f) = comp₁ (ID₂ g) (ID₂ f) qed)
  (id_comp₂ : proof Π {a b c : D₀} (f : hom a b) (g : hom b c),
    ID₁ (g ∘ f) = comp₂ (ID₁ g) (ID₁ f) qed)
  (zero_unique : proof Π (a : D₀), ID₁ (ID a) = ID₂ (ID a) qed)
  (interchange : proof Π {a₀₀ a₀₁ a₀₂ a₁₀ a₁₁ a₁₂ a₂₀ a₂₁ a₂₂ : D₀}
    {f₀₀ : hom a₀₀ a₀₁} {f₀₁ : hom a₀₁ a₀₂} {f₁₀ : hom a₁₀ a₁₁}
    {f₁₁ : hom a₁₁ a₁₂} {f₂₀ : hom a₂₀ a₂₁} {f₂₁ : hom a₂₁ a₂₂}
    {g₀₀ : hom a₀₀ a₁₀} {g₀₁ : hom a₀₁ a₁₁} {g₀₂ : hom a₀₂ a₁₂}
    {g₁₀ : hom a₁₀ a₂₀} {g₁₁ : hom a₁₁ a₂₁} {g₁₂ : hom a₁₂ a₂₂}
    (x : D₂ f₁₁ f₂₁ g₁₁ g₁₂) (w : D₂ f₁₀ f₂₀ g₁₀ g₁₁)
    (v : D₂ f₀₁ f₁₁ g₀₁ g₀₂) (u : D₂ f₀₀ f₁₀ g₀₀ g₀₁),
    comp₁ (comp₂ x w) (comp₂ v u) = comp₂ (comp₁ x v) (comp₁ w u) qed)
```

Verschränkte Moduln in Lean

```
structure xmod {P₀ : Type} [P : groupoid P₀] (M : P₀ → Group) :=
  (P₀_hset : is_hset P₀)
  (μ : Π ⦃p : P₀⦄, M p → hom p p)
  (μ_respect_comp : Π ⦃p : P₀⦄ (b a : M p), μ (b * a) = μ b ∘ μ a)
  (μ_respect_id : Π (p : P₀), μ 1 = ID p)
  (φ : Π ⦃p q : P₀⦄, hom p q → M p → M q)
  (φ_respect_id : Π ⦃p : P₀⦄ (x : M p), φ (ID p) x = x)
  (φ_respect_P_comp : Π ⦃p q r : P₀⦄ (b : hom q r) (a : hom p q) (x : M p),
    φ (b ∘ a) x = φ b (φ a x))
  (φ_respect_M_comp : Π ⦃p q : P₀⦄ (a : hom p q) (y x : M p),
    φ a (y * x) = (φ a y) * (φ a x))
  (CM1 : Π ⦃p q : P₀⦄ (a : hom p q) (x : M p), μ (φ a x) = a ∘ (μ x) ∘ a⁻¹)
  (CM2 : Π ⦃p : P₀⦄ (c x : M p), φ (μ c) x = c * (x * c⁻¹ᵍ))
```

Verschränkte Moduln in Lean

Teil des Beweises, dass verschränkte Moduln eine Kategorie bilden:

```
definition xmod_morphism_id_left :
  xmod_morphism_comp (xmod_morphism_id Y) f = f :=
begin
  cases f,
  fapply xmod_morphism_congr,
      apply idp,
    apply idp,
  repeat (apply eq_of_homotopy ; intros),
  apply idp,
end

universe variables l₁ l₂ l₃
definition cat_xmod :
  precategory.{(max l₁ l₂ l₃)+1 (max l₁ l₂ l₃)} Xmod.{l₁ l₂ l₃} :=
begin
  fapply precategory.mk,
    intros [X, Y], apply (xmod_morphism X Y),
    intros [X, Y], apply xmod_morphism_hset,
    intros [X, Y, Z, g, f], apply (xmod_morphism_comp g f),
    intro X, apply xmod_morphism_id,
    intros [X, Y, Z, W, h, g, f], apply xmod_morphism_assoc,
    intros [X, Y, f], apply xmod_morphism_id_left,
  intros [X, Y, f], apply xmod_morphism_id_right,
end
```