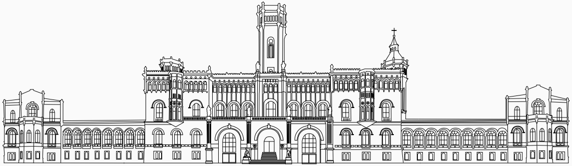
Leibniz Universität Hannover

Homotopietypentheorie

Oberseminar Theoretische Informatik

Florian Chudigiewitsch

Institut für Theoretische Informatik



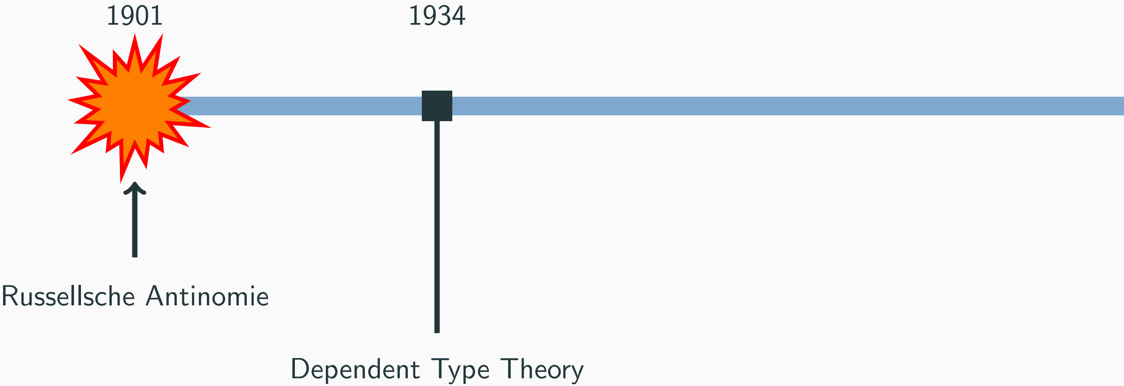
Themen

- Geschichtliches
- Grundlagen der Typentheorie
- Homotopietypentheorie
- Anwendungen & Aktuelle Forschungsfragen

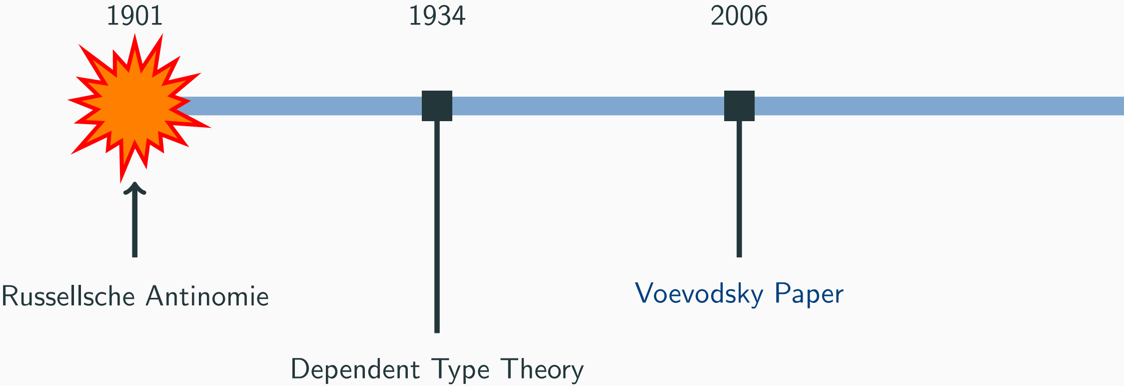
Geschichtliches

Geschichtliches

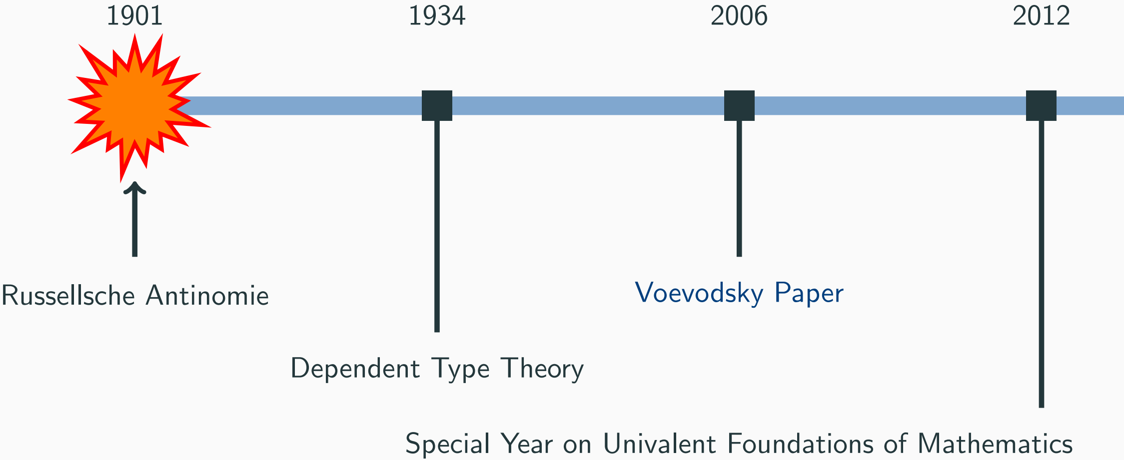
Geschichtliches



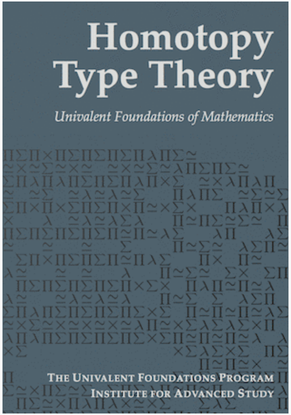
Geschichtliches



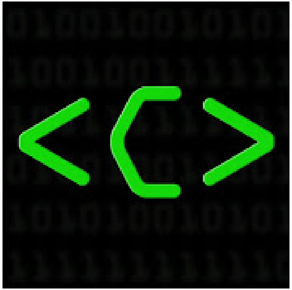
Geschichtliches



Quellen und Referenzen



https://homotopytypetheory.org/book/ [Uni13]



Computerphile (Youtube)

- Type Theory
- Propositions as Types
- Voevodsky
- Homotopy Type Theory

Emily Riehl

- Video
- Slides

Grundlagen der Typentheorie

Konstruktivität

- Law of excluded middle (LEM) wichtiges Axiom in der klassischen Mathematik
  - $\vdash \varphi \vee \neg\varphi$
  - Ermöglicht Widerspruchsbeweise
- Konstruktive Logik verzichtet auf LEM
- Dadurch wird durch einen Beweis immer ein „Witness“ erzeugt
- Keine wirkliche Einschränkung, da man LEM jederzeit als Annahme hinzunehmen kann
- Hilft „mentaler Hygiene“: (Wo) brauche ich LEM wirklich?

Dependent Type Theory: Die vier „Grundaussagen“

Γ ist der *Kontext*, der die Typen aller vorkommenen Variablen deklariert.

Die vier Grundformen („judgements“) der „wohlgeformten Formeln“ der Dependent Type Theory sind:

Formel	Interpretation	Beispiel
$\Gamma \vdash A$ type	„A ist ein Typ“	$\mathbb{N}$ type
$\Gamma \vdash a : A$	„a ist ein Term vom Typ A “	$1 : \mathbb{N}$
$\Gamma, x : A \vdash B(x)$ type	„$B(x)$ ist eine Typfamilie über A “	$n : \mathbb{N} \vdash \mathbb{R}^n$ type
$\Gamma, x : A \vdash b(x) : B(x)$	„$b(x)$ ist eine Termfamilie“	$n : \mathbb{N} \vdash \vec{0} : \mathbb{R}^n$

Universum: Typ, dessen Elemente Typen sind. Bilden Hierarchie

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$$

Dependent Type Theory: Die vier „Grundregelschemata“ I

- Name: $\times$-formation rules
- Beschreibung: Haben wir Typen A und B gegeben, gibt es einen Produkttypen $A \times B$.
- Formal:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \times B \text{ type}}$$

Dependent Type Theory: Die vier „Grundregelschemata“ II

- Name: $\times$-introduction rules
- Beschreibung: Haben wir Terme $a : A$ und $b : B$ gegeben, gibt es einen Term $(a, b) : A \times B$.
- Formal:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$$

Dependent Type Theory: Die vier „Grundregelschemata“ III

- Name: $\times$-elimination rules
- Beschreibung: Haben wir einen Term $p : A \times B$ gegeben, gibt es Terme $\mathrm{pr}_1(p) : A$ und $\mathrm{pr}_2(p) : B$.
- Formal:

$$\frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathrm{pr}_1(p) : A} \qquad \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathrm{pr}_2(p) : B}$$

Weiterhin: *Judgemental equality (α-conversion), computation rules (β-reduction):* verbinden introduction und elimination rules, optionales *uniqueness principle (η-expansion)*.

Funktionstypen

- $\to$-formation: Haben wir Typen A und B gegeben, gibt es einen Typen $A \to B$.
- $\to$-introduction: Haben wir im Kontext eines Terms $x : A$ einen Term $b(x) : B$ gegeben, gibt es einen Term $\lambda x.b(x) : A \to B$. Formal:

$$\frac{\Gamma, x : A \vdash b(x) : B}{\Gamma \vdash \lambda x.b(x) : A \to B}$$

- $\to$-elimination: Haben wir Terme $f : A \to B$ und $a : A$ gegeben, gibt es einen Term $f(a) : B$.
- Zwei computation rules.

Propositions as Types

- Aussagen werden durch Typen repräsentiert
- Man beweist sie indem man einen Term vom entsprechenden Typ erzeugt
- $$\frac{\text{Beweise}}{\text{Aussagen}} = \frac{\text{Programme}}{\text{Typen}}$$
- Klassisch: Prop $\leftrightsquigarrow$ Bool, „Wahrheit“
- Andere Möglichkeit: Prop $\leftrightsquigarrow$ Types, „Zeugnis“

Propositions as Types: Beispiel

Beispiel 1

Aussage: Für beliebige Typen P und Q gibt es den Term

$$\text{modus-ponens} : P \times (P \to Q) \to Q.$$

Beweis: Mit $\to$-introduction aus Term $x : P \times (P \to Q)$ einen Term aus Q generieren. $\times$-elimination liefert uns $\text{pr}_1(x) : P$ und $\text{pr}_2(x) : P \to Q$.
$\to$-elimination liefert $(\text{pr}_2(x))(\text{pr}_1(x)) : Q$.
Somit: $\text{modus-ponens} :\equiv \lambda x.(\text{pr}_2(x))(\text{pr}_1(x))$. ∎

Curry-Howard-Isomorphismus: Ein Beweis korrespondiert zu einem Computerprogramm, welches einen Term vom Typ der Aussage zurückgibt.

Gleichheit als Identitätstyp

Mathematische Gleichheit wird über Identitätstypen ausgedrückt.

- $=$-formation: Haben wir einen Typ A und zwei Terme $x, y : A$ gegeben, gibt es einen Typ $x =_A y$.
- $=$-introduction: Haben wir einen Term $x : A$ gegeben, so gibt es einen Term $\mathrm{refl}_x : x =_A x$.

Elimination rule für $x =_A y$ via path induction:

Haben wir eine Typfamilie $\Gamma, x, y : A, p : x =_A y \vdash B(x, y, p)$ gegeben und wollen einen Term von $B(x, y, p)$ konstruieren, reicht es anzunehmen, dass y x und p refl_x ist.

Gleichheit ist eine Äquivalenzrelation

Lemma 2 Für beliebige $x, y : A$ gilt $(x =_A y) \to (y =_A x)$.

Beweis.
Hausaufgabe. ☺ ■

Lemma 3 Für beliebige $x, y, z : A$ gilt $(x =_A y) \to ((y =_A z) \to (x =_A z))$.

Beweis.
Hausaufgabe. ☺ ■

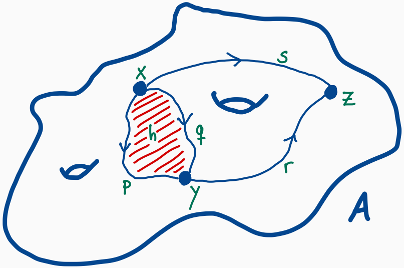
Homotopietypentheorie

Der „Stein von Rosette“ für HoTT

Typen	Logik	Mengen	Homotopie
A	Aussage	Menge	Raum
$a : A$	Beweis	Element	Punkt
$B(x)$	Prädikat	Mengenfamilie	Faserung
$b(x) : B(x)$	Bedingter Beweis	Elementfamilie	Schnitt
$0, 1$	$\bot, \top$	$\emptyset, \{\emptyset\}$	$\emptyset, \star$
$A + B$	$A \vee B$	Disjunkte Vereinigung	Coprodukt
$A \times B$	$A \wedge B$	Menge von Paaren	Produktraum
$A \to B$	$A \Rightarrow B$	Menge von Funktionen	Funktionsraum
$\sum_{(x:A)} B(x)$	$\exists_{x:A} B(x)$	Disjunkte Summe	Totalraum
$\prod_{(x:A)} B(x)$	$\forall_{x:A} B(x)$	Produkt	Raum der Schnitte
Id_A	Gleichheit $(=)$	$\{(x, x) \mid x \in A\}$	Pfadraum A^I

Path induction „homotopisch“ interpretiert

- Typ $A \leftrightsquigarrow$ (Topologischer) Raum A
- Term $a : A \leftrightsquigarrow$ Punkt a in A
- Term $p : x =_A y \leftrightsquigarrow$ Pfad p von x nach y in A
- Term $p : p =_{x=_A y} a \leftrightsquigarrow$ Homotopie h von p nach q in A

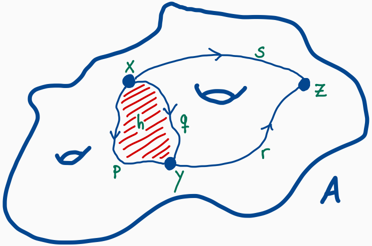


- Symmetrie und Transitivität wird als Umkehrung und Komposition von Pfaden, Homotopien, höheren Homotopien... interpretiert.
- van den Berg/Garner und Lumsdaine: Typen haben die Struktur eines schwachen ∞-Gruppoiden
- Unterschied: Homotopietheorie analytisch, Homotopietypentheorie synthetisch

Extensional vs. Intensional und UIP

Frage:

- „Uniqueness of identity proofs“ (UIP): Wenn zwei Beweise p und q beide A zeigen, gilt dann immer $p = q$?
- Extensional: Ja, Intensional (hier): Nein
- Homotopie-Äquivalenzklassen von Schleifen an einem Punkt x_0 bilden die *fundamentale Gruppe*.

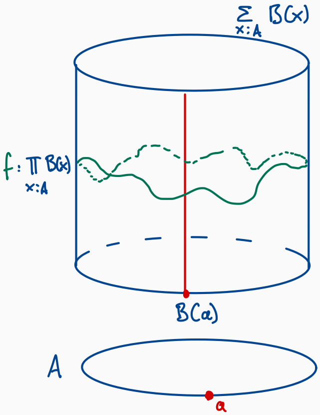


Weitere Typen – mit homotopischer Interpretation

- Typfamilie $x : A \vdash B(x)$ type $\leftrightsquigarrow$ Faserung über A
- Dependent sum Typ $\sum_{x:A} B(x)$ $\leftrightsquigarrow$ Totalraum einer Faserung

  Beispiel:

$$\text{Gruppoid} :\equiv \sum_{A:\mathcal{U}} (A \to A \to A)$$

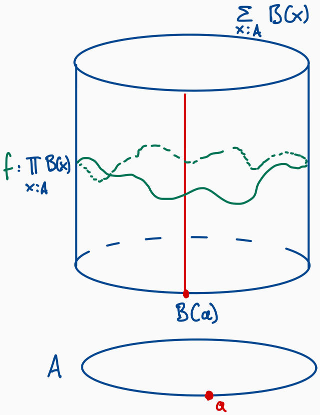


Weitere Typen – mit homotopischer Interpretation

- Typfamilie $x : A \vdash B(x)$ type $\leftrightsquigarrow$ Faserung über A
- Dependent function Typ $\prod_{x:A} B(x) \leftrightsquigarrow$ Raum der Schnitte

  Beispiel:

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



Kontrahierbare Typen

Definition 4: (Kontrahierbare Typen)

Es gibt einen eindeutigen Term vom Typ A gdw.

$$\sum_{a:A} \prod_{x:A} a =_A x$$

bewohnt ist, bzw. wenn der Raum A *kontrahierbar* ist.

Typenäquivalenz

Definition 5: (Typenäquivalenz)

Zwei Typen A und B sind *äquivalent*, wenn der Typ

$$A \simeq B :\equiv \sum_{h:A\to B} \left(\sum_{f:B\to A} \prod_{a:A} f(h(a)) =_A a \right) \times \left(\sum_{g:B\to A} \prod_{b:B} h(g(b)) =_B b \right)$$

bewohnt ist.

Univalenzaxiom

Man kann leicht beweisen, dass

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

Definition 6: (Univalenzaxiom (Voevodsky))

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

Was gibt es noch?

- HoTT und der λ-Kalkül
- Higher inductive types
  Typkonstruktoren können auch (höhere) Pfade erzeugen. Intervalltyp I:
  - Punkt $0_I : I$
  - Punkt $1_I : I$
  - Pfad $\mathsf{seg} : 0_I =_I 1_I$

  Hausaufgabe: Kreis als higher inductive type definieren
- Beweistheorie in HoTT

Anwendungen & Aktuelle Forschungsfragen

Anwendungen

- Homotopietheorie
- Kategorientheorie
- Theorembeweiser
- Programmverifikation
- Funktionale Progammierung

Aktuelle Forschungsfragen

- Informelle Typentheorie
- Formalisierung der klassischen Mathematik in HoTT
- Konstruktivität des Univalenzaxioms (Cubical Type Theory)
- HoTT auf diskreten Räumen (wie $\mathbb{N}$) führt zu vielen „unnötigen“ Identitätstermen
  - Möglichkeit, diese zu kollabieren
- HoTT und Topoi
  - Intuitionistische Higher Order Logic ist die interne Sprache von 1-Topoi, HoTT könnte die von $(\infty, 1)$-Topoi sein

Aktuelle Forschungsfragen: Komplexitätstheorie?

Wenn HoTT Grundlage der Mathematik sein kann, muss man mit ihr auch Komplexitätstheorie betreiben können!

- Klassische Komplexitätstheorie sehr „mengenzentriert"
- HoTT eng verbunden mit funktionalen Programmiersprachen

Motivierende Fragen

Betrachte fiktive Sprache $\mathcal{L}$: Funktional, total, mit Dependent Types.

- Ist die Korrektheit meines $\mathcal{L}$-Programmes effizient verifizierbar?
- Kann ich verifizieren, dass mein $\mathcal{L}$-Programm effizient ist?
- Ist ein Problem effizient lösbar und wie sieht der Algorithmus aus, der das Problem effizient löst?
- Kann ich die Daten meiner Datenbank effizient und korrekt migrieren?

Mögliche Ausgangspunkte

- Mathematische Strukturen sind „First Class Citizens“ in HoTT
  - Kodierung egal
- Eher rekursionstheoretische Ansätze erforderlich [Con95]
  - Maschinenunabhängige Komplexitätstheorie
  - Implizite Komplexitätstheorie [Lag12]
    - Wird schon für lineare Typentheorien angewendet

Aktuelle Forschungsfragen: Komplexitätstheorie?

Begriff der
„Linearität“ in HoTT*

*Danke an Prof. Thorsten Altenkirch [McB; Atk18]

Aktuelle Forschungsfragen: Komplexitätstheorie?

Begriff der „Linearität“ in HoTT*

Definition von Komplexitätsmaßen

*Danke an Prof. Thorsten Altenkirch [McB; Atk18]

Aktuelle Forschungsfragen: Komplexitätstheorie?

Begriff der „Linearität“ in HoTT*

Definition von Komplexitätsmaßen

Charakterisierung von sinnvollen Komplexitätsklassen und Hierarchien

*Danke an Prof. Thorsten Altenkirch [McB; Atk18]

Aktuelle Forschungsfragen: Komplexitätstheorie?

Begriff der „Linearität“ in HoTT*

Definition von Komplexitätsmaßen

Charakterisierung von sinnvollen Komplexitätsklassen und Hierarchien

Charakterisierung von effizienter Berechenbarkeit [BC]

*Danke an Prof. Thorsten Altenkirch [McB; Atk18]

Danke!

Fragen?

Quellenverweise i

Robert Atkey. *Syntax and Semantics of Quantitative Type Theory*. 2018. DOI: https://doi.org/10.1145/3209108.3209189.

Stephen Bellatoni und Stephen Cook. *A New Recursion-Theoretic Characterization of the Polytime Functions*. URL: https://www.cs.toronto.edu/~sacook/homepage/ptime.pdf.

R.L. Constable. "Expressing computational complexity in constructive type theory". In: *Leivant D. (eds) Logic and Computational Complexity* (1995). DOI: https://doi.org/10.1007/3-540-60178-3_82.

Quellenverweise ii

U. Dal Lago. "A Short Introduction to Implicit Computational Complexity". In: *Bezhanishvili N., Goranko V. (eds) Lectures on Logic and Computation* (2012). DOI: https://doi.org/10.1007/978-3-642-31485-8_3.

Conor McBride. *I Got Plenty o' Nuttin'*. URL: https://personal.cis.strath.ac.uk/conor.mcbride/PlentyO-CR.pdf.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.