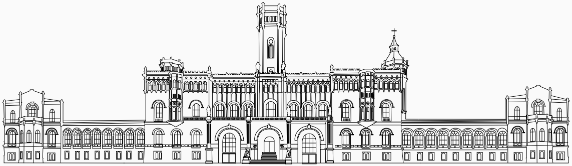
Homotopietypentheorie

Oberseminar Theoretische Informatik

Florian Chudigiewitsch

DATUM

Institut für Theoretische Informatik



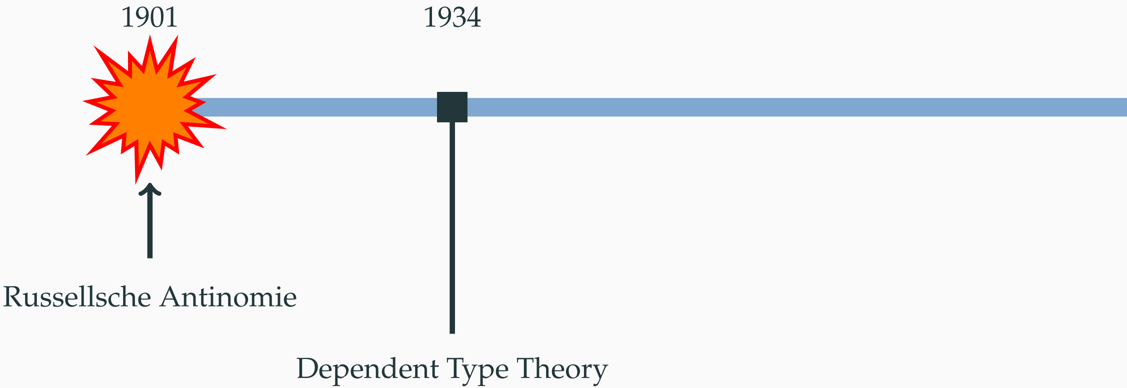
Themen

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

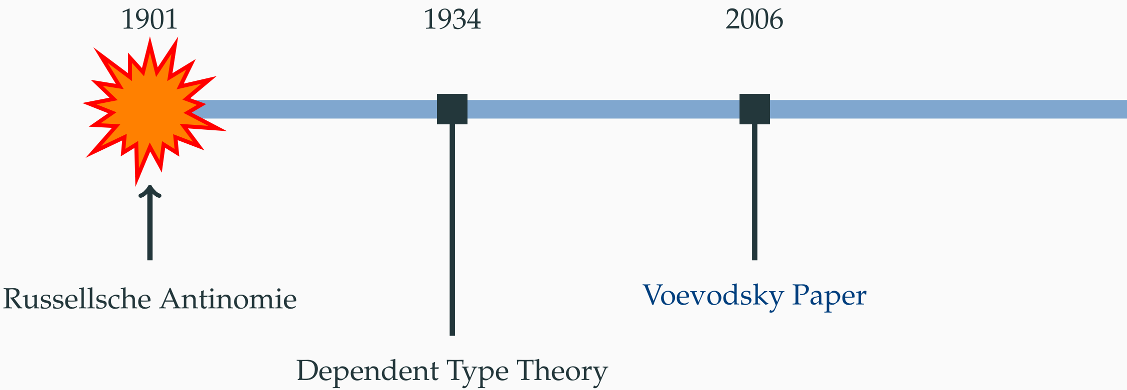
Geschichtliches

Geschichtliches

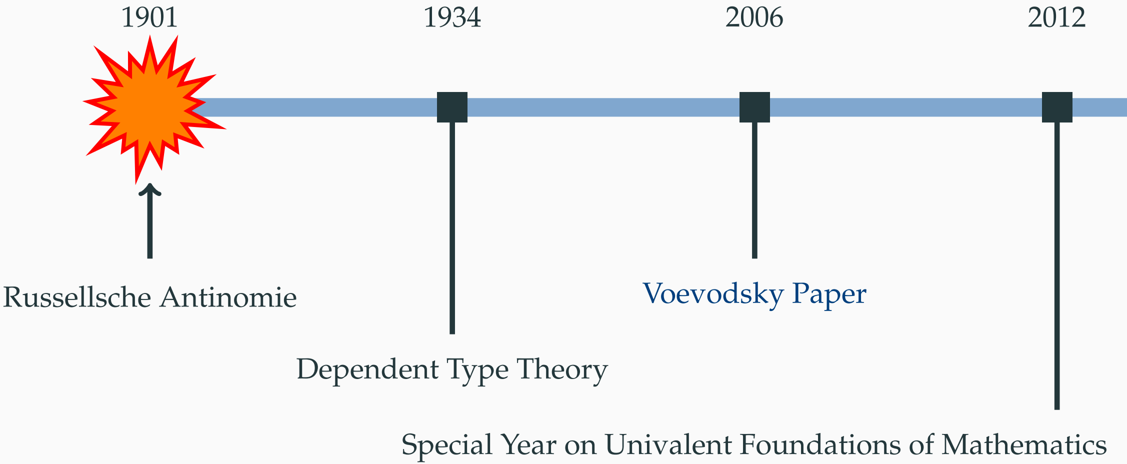
Geschichtliches



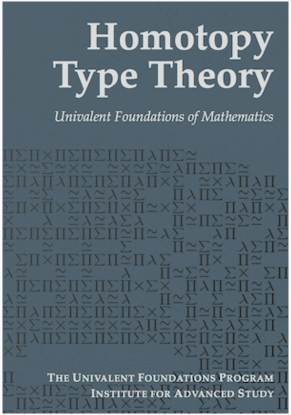
Geschichtliches



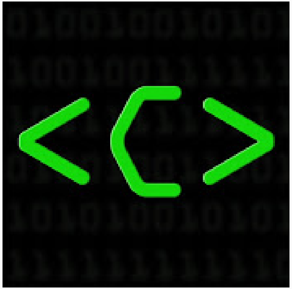
Geschichtliches



Quellen und Referenzen



https://homotopytypetheory.org/book/



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
- $\forall \varphi (\varphi \lor \neg \varphi)$
- Ermöglicht Widerspruchsbeweise
- Konstruktive Logik verzichtet auf LEM
- Dadurch wird durch einen Beweis immer ein „Witness“ erzeugt

Dependent Type Theory: Die vier „Grundaussagen“

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“	$n : \mathbb{N} \vdash \vec{0} : \mathbb{R}^n$

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

Universum: Typ, dessen Elemente Typen sind. Bilden Hierarchie

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

Dependent Type Theory: Die vier „Grundregelarten“ I

- Name: x-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 „Grundregelarten“ II

- Name: x-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 „Grundregelarten“ 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
- TODO

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. x-elimination liefert uns $\mathrm{pr}_1(x) : P$ und $\mathrm{pr}_2(x) : P \to Q$.
$\to$-elimination liefert $(\mathrm{pr}_2(x))(\mathrm{pr}_1(x)) : Q$.
Somit: $\text{modus-ponens} :\equiv \lambda x.(\mathrm{pr}_2(x))(\mathrm{pr}_1(x))$. ■

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 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.

Homotopietypentheorie

Path induction homotopisch interpretiert

- Typ A $\leftrightsquigarrow$ 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 ∞-Gruppoiden

„Indiscernibility of identicals“

Beantwortete offene Frage der Beweistheorie:

- „Indiscernibility of identicals“: Wenn zwei Beweise p und q beide A zeigen, kann man dann immer $p = q$ zeigen?
- Nein!

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:

$$\mathrm{Magma} :\equiv \sum_{A:\mathcal{U}} (A \to A \to A)$$

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 Sektionen
  Beispiel:

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

Zusammenziehbare Typen

Definition 2: (Zusammenziehbare 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 *zusammenziehbar* ist.

Typenäquivalenz

Definition 3: (Typenäquivalenz)

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

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

bewohnt ist.

Univalenzaxiom

Man kann leicht beweisen, dass

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

Definition 4: (Univalenzaxiom (Voevodsky))

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

Anwendungen & Aktuelle Forschungsfragen

Anwendungen

- Homotopietheorie
- Kategorientheorie
- Theorembeweiser
- Programmverifikation
- Funktionale Progammierung

Aktuelle Forschungsfragen

- Informelle Typentheorie
- Konstruktivität des Univalenzaxioms
- Komplexitätstheorie?

Quellenverweise i

Danke!

Fragen?