Inductive-Inductive types in Homotopy Type Theory

HUGUNIN Jasper

東京工業大学

April 9, 2018

Outline

Outline

Background

- データの形を考えてプログラムを作ったり定理を証明する.
- 自然数と文字列を根本的に違うものとして扱う.
- 集合論と違って，$1 \in 2$ は論理式になっていない.
- 型を命題として考えることで，数学と論理に使える.

Judgments

We use Γ, Δ for contexts, A, B, C for types, x, y, z for variables.

- $\Gamma \vdash$ means Γ is a well-formed context.
- $\Gamma \vdash A$ means A is a well-formed type in context Γ.
- $\Gamma \vdash x : A$ means a is a well-formed term of type A in context Γ.
- $\Gamma \vdash A = B$ means A and B are equal types.
- $\Gamma \vdash x = y : A$ means x and y are equal terms.

Rules

Well-formed contexts: $\dfrac{}{\varepsilon \vdash}$ $\qquad \dfrac{\Gamma \vdash A}{\Gamma, x : A \vdash}$

Well-formed types:

Forall - dependent function type $\dfrac{\Gamma, x : A \vdash B}{\Gamma \vdash \forall (x : A).B}$

Exists - dependent pair type $\dfrac{\Gamma, x : A \vdash B}{\Gamma \vdash \exists (x : A).B}$

Natural numbers $\dfrac{}{\Gamma \vdash \mathbb{N}}$

Equality - Paths $\dfrac{\Gamma \vdash A \qquad \Gamma \vdash x : A \qquad \Gamma \vdash y : A}{\Gamma \vdash \mathrm{Path}\, A x y}$

Universe - collection of small sets $\dfrac{}{\Gamma \vdash U}$ $\qquad \dfrac{\Gamma \vdash A : U}{\Gamma \vdash A}$

Rules

Well-formed terms:

$$\forall\text{-intro}\quad \frac{\Gamma, x : A \vdash y : B}{\Gamma \vdash \lambda x.y : \forall(x : A).B} \qquad \forall\text{-elim}\quad \frac{\Gamma \vdash f : \forall(x : A).B \qquad \Gamma \vdash x : A}{\Gamma \vdash fx : B[x]}$$

$$\forall\text{-comp}\quad \frac{\Gamma, x : A \vdash y : B \qquad \Gamma \vdash x : A}{\Gamma \vdash (\lambda x.y)x = y[x] : B[x]}$$

$$\exists\text{-intro}\quad \frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : B[x]}{\Gamma \vdash (x, y) : \exists(x : A).B}$$

$$\exists\text{-elim}\quad \frac{\Gamma \vdash z : \exists(x : A).B}{\Gamma \vdash z.1 : A} \qquad \frac{\Gamma \vdash z : \exists(x : A).B}{\Gamma \vdash z.2 : B[z.1]}$$

Rules for $\mathbb{N}$

$\mathbb{N}$-intro $\dfrac{}{\Gamma \vdash 0 : \mathbb{N}} \qquad \dfrac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash Sn : \mathbb{N}}$

$\mathbb{N}$-elim $\dfrac{\Gamma \vdash n : \mathbb{N} \quad \Gamma, n : \mathbb{N} \vdash P \qquad \Gamma \vdash x : P[0] \quad \Gamma, n : \mathbb{N}, a : P[n] \vdash f : P[Sn]}{\Gamma \vdash \mathsf{rec}\ n\ P\ x\ f : P[n]}$

$\mathbb{N}$-comp $\dfrac{}{\Gamma \vdash \mathsf{rec}\ 0\ P\ x\ f = x : P[0]}$

$$\frac{}{\Gamma \vdash \mathsf{rec}\ Sn\ P\ x\ f = f\,[n][\mathsf{rec}\ n\ P\ x\ f\,] : P[Sn]}$$

Outline

Inductive types

- $\mathbb{N}$ is one example of an inductive type.
- In set theory these are constructed by taking the least fixed point of some operator.
- In type theory, we will postulate the constructors, and add an elimination rule that computes given how to handle each constructor.

Example of an inductive type

Definition (terms in a simple language)

- 'var i' is a term for all $i \in \mathbb{N}$.
- If A and B are terms, then 'sum A B' is a term.
- If A is a term, then 'not A' is a term.

The set of terms is the smallest set satisfying the above conditions.

In type theory, we formalize this as having terms:

- $\text{var} : \mathbb{N} \to \text{term}$
- $\text{sum} : \text{term} \to \text{term} \to \text{term}$
- $\text{not} : \text{term} \to \text{term}$

And a term that expresses the induction principle (and computes)
$\text{elim-term} : \forall(P : \text{term} \to U).(\forall n.P(\text{var } n)) \to (\forall AB.P\ A \to P\ B \to P(\text{sum } A\ B)) \to (\forall A.P\ A \to P(\text{not } A)) \to \forall(t : \text{term}).P\ t$

More inductive types

- こういう定義はいつも使われている.
- 自然数もそうだし，文字列もそうだし，形式的な言語や論理式
- 証明体系もこのように書ける.
- 付いてくる帰納法も長さとか複雑さとか，無理やり自然数にするより使いやすいと思う.
- しかし，今の型論では複雑すぎて一般には使われていないケースがあります.

Outline

Outline

Type theory in Type theory

さっき上げた型体系を思い出してみよう.

- そこに二つのジャッジメントとして，$\Gamma \vdash$ と $\Gamma \vdash A$ があった．二つ目を考える前に，$\Gamma \vdash$ が成り立たなければなりません．つまり，$\Gamma \vdash A$ は　$\Gamma \vdash$ に依存している.
- しかし，$\dfrac{\Gamma \vdash A}{\Gamma, x : A \vdash}$ というルールがあって，それは $\Gamma \vdash A$ を使って，$\Gamma \vdash$ を定義している.
- つまり，$\Gamma \vdash$ を定義してから $\Gamma \vdash A$ を定義することができなくて，同時に定義しなければいけない.
- 今のところ，こういう定義は複雑だと思われている.

Inductive Inductive types

- We consider the simultaneous inductive definition of types $A : U$ and $B : A \to U$, where the constructors of B can refer to the constructors of A, to be an Inductive Inductive definition.
- Our goal is to build Inductive Inductive types out of $\forall$, $\exists$, $\mathbb{N}$, and regular inductive types.
- We succeed in defining a specific set of Inductive Inductive types with weak elimination rules.
- The proofs here are formalized in the proof assistant Agda (using the cubical mode).
- The technique is expected to generalize to a much larger set of definitions.

Our running example

We fix one class of relatively simple Inductive Inductive types to work with. These types have the advantage of being simple enough to be easy to work with, while being general enough to eliminate unnecessary detail, and being complex enough that results are generalizable.
Given

$$AA, BA : U \quad AB, AC : AA \to U \quad BB, BC : BA \to U$$

$$ACi : \forall a.AC\ a \to AB\ a \quad BCi : \forall a.BC\ a \to BB\ a \quad Bi : \forall a.BB\ a$$

Consider A and B as inductively generated by the constructors

$$\begin{aligned}
\text{supA} :& \forall a(\text{chA} : \forall(b : AB\ a).A).\\
& \forall(\text{chB} : \forall(c : AC\ a).B(\text{chA}\ (ACi\ a\ c))).A\\
\text{supB} :& \forall a(\text{chA} : \forall(b : BB\ a).A).\\
& \forall(\text{chB} : \forall(c : BC\ a).B(\text{chA}\ (BCi\ a\ c))).B(\text{chA}(Bi\ a))
\end{aligned}$$

Claim

In cubical Agda (a computer proof assistant), we have defined such types A and B, with functions supA and supB, such that the following induction principle holds.

$$\forall (P : A \to U)(Q : \forall x.B\ x \to U).$$
$$\forall (IHA : \forall a\ \text{chA}\ \text{chB}.(\forall b.P(\text{chA}\ b)) \to (\forall c.Q(\text{chB}\ c)) \to$$
$$P(\text{supA}\ a\ chA\ chB)).$$
$$\forall (IHB : \forall a\ \text{chA}\ \text{chB}.(\forall b.P(\text{chA}\ b)) \to (\forall c.Q(\text{chB}\ c)) \to$$
$$Q(\text{supB}\ a\ chA\ chB)).$$
$$(\forall x.P\ x) \wedge (\forall x\ (y : B\ x).Qy)$$

And such that the resulting functions compute to IHA and IHB on supA and supB respectively. Therefore, we are justified in using such types.

Outline

Base Case

まず，B と A の関係を忘れて，型 A_0 と B_0 を以下で定義する.

$$\mathsf{supA}_0 : \forall a(\mathsf{chA} : \forall(b : AB\ a).A_0)(\mathsf{chB} : \forall(c : AC\ a).B_0).A_0$$
$$\mathsf{supB}_0 : \forall a(\mathsf{chA} : \forall(b : BB\ a).A_0)(\mathsf{chB} : \forall(c : BC\ a).B_0).B_0$$

この定義に問題はないが，無意味なものも入ってしまう．そして $\mathsf{dep}_0 : B_0 \to A_0$ を Bi で定義する.

Induction Motive

An algebra consists of four parts:

- A_0 に対する述語 algA
- $\exists(x : A_0).\mathsf{algA}\ x$ と $\mathsf{dep}_0^{-1}(x)$ に対する述語 algB
- algsupA : $\forall a\ \mathsf{chA}\ \mathsf{chB}.(\forall b.\mathsf{algA}(\mathsf{chA}\ b)) \to (\forall c.\mathsf{algB}(\mathsf{chB}\ c)) \to \mathsf{algA}(\mathsf{supA}_0 a\ \mathsf{chA}\ \mathsf{chB})$
- algsupB : $\forall a\ \mathsf{chA}\ \mathsf{chB}.(\forall b.\mathsf{algA}(\mathsf{chA}\ b)) \to (\forall c.\mathsf{algB}(\mathsf{chB}\ c)) \to \mathsf{algB}(\mathsf{supB}_0 a\ \mathsf{chA}\ \mathsf{chB})$

We can clearly make some algebra by taking algA and algB to be $\top$.

Induction Step

Assume we have an algebra T.
By induction on A_0 and B_0, we define:

- For $x : A_0$, a set $A'\ x$, and a function $\pi_A : A'\ x \to \mathrm{algA}\ x$.
- For $y : B_0$, a set $B'\ y$, and a function $\pi_B : B'\ y \to \exists(xg : \mathrm{algA}(\mathrm{dep}_0\ y)).\mathrm{algB}\ xg\ y$.
- When $x = \mathrm{supA}_0\, a\ \mathrm{chA}\ \mathrm{chB}$, take A' to be

$$\exists(\mathrm{chAgood} : \forall b.A'(chA\ b)).\forall c.$$
$$(chB\ c = chA(ACi\ a\ c)) \wedge$$
$$\exists(cg : B'(chB\ c)).\pi_B\ cg\ .1 = \pi_A\ (\mathrm{chAgood}\ (ACi\ a\ c))$$

with π_A taking $(\mathrm{chAgood}, \mathrm{chBgood})$ to
$\mathrm{algsupA}\ a\ chA\ chB\ (\pi_A \circ \mathrm{chAgood})\ (\pi_B \circ \mathrm{chBgood})$.

Induction Step

- Practically the same definition works for defining B' and π_B.
- Given these definitions, we define a new algebra with
- algA $x = A'\ x$
- algB $x\ y = \exists(yg : B'y).\pi_B\ yg\ .1 = \pi_A x$ (note that this is the same as was used in the definition of A')
- algsupA, algsupB as the straight-forward definitions.
- This essentially adds equality requirements between x and x' where we had $y : \text{algB}_T\ x'$ being used where $\text{algB}_T\ x$ was expected in T. However, this requirement is new data which does not necessarily agree.

Chain

- So we get a sequence of algebras, one for each $n \in \mathbb{N}$.
- Define our final type A as the limit of this chain:

$$A_1 \xleftarrow{\pi_{A_1}} A_2 \xleftarrow{\pi_{A_2}} A_3 \xleftarrow{\pi_{A_3}} \cdots$$

- This is dependent on $x : A_0$. Let's consider what happens when $x = \mathrm{supA}_0\ a\ chA\ chB$.
- At each level, we have chAgood showing that chA b is also good for all b, and since π_A passes straight through to chAgood, we know that each is equal to π_A of the next. Thus we have A (chA b) for all b.

Chain of B

- The case for chB is a bit more difficult.
- We do have $B'\ (chB\ c)$ for all c at each level.
- We also have a proof that $chB\ c = chA\ (ACi\ a\ c)$ at each level, but we can set π_B up so that it doesn't mess with that, so we have proofs that they are all equal to each-other.
- And we also have proofs that $\pi_A\ a = \pi_A\ b$ for some a and b at each level, but because taking π_A of everything in an A-chain is the identity, we manage to recover a full equality.
- Therefore, our defined A has exactly the information that we need.
- The same logic applies to B.

Conclusion

- というわけで，成功した．指定されたものに合うような型と関数を定義した．
- さらに，この証明はもっと一般できる模様で，任意の Inductive Inductive type を作ることはできると思われる．
- さっきの説明は結構雑打倒言う自覚はあるが，ちゃんと Agda を使って証明しましたので，間違ってはいないはず．
- しかし，問題点は残っている．

Problem 1: The eliminator has poor computational behavior

- elimA 等を構成子に適応すると，正しい答えが出てくることは証明できても，その証明は難しい.
- 普通の Inductive type では，その証明はただ反射性によるものなので，使い勝手が悪くなっている.
- これは私の推測だが，二つの条件を満たせれば，これは解決できると思っているが，まだまだ研究が必要だ.

Problem 2: 一意性性はまだ証明できない

- さっき作った帰納法では,$P : A \to U$ と $Q : \forall(x : A).B\,x \to U$ を使っていた.
- しかしこれをもっと一般化にできる：$A : \forall(x : A).B\,x \to P\,x \to U$ にしてしまえば，最後に定義したいのは $elimA : \forall x.P\,x$ と $elimB : \forall x(y : B\,x).Q\,x\,y\,(elimA\,x)$ を作るのが目的になる.
- こっちが使えば，一意性は証明できる.
- これも証明できると思っているが，これはまた難しそうだ.