Programming Research Laboratory

FLOLAC '16

Martin-Löf type theory

Theory of natural numbers

12 July 2016

柯向上

(日本) 国立情報学研究所

hsiang-shang@nii.ac.jp

Natural numbers

- Formation:

$$\frac{}{\Gamma \vdash \mathbb{N} : \mathcal{U}}\;(\mathbb{N}\mathsf{F})$$

- Introduction:

$$\frac{}{\Gamma \vdash \mathtt{zero} : \mathbb{N}}\;(\mathbb{N}\mathsf{IZ}) \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathtt{suc}\; n : \mathbb{N}}\;(\mathbb{N}\mathsf{IS})$$

- Elimination:

$$\frac{\begin{array}{ll} \Gamma, x : \mathbb{N} & \vdash P : \mathcal{U} \\ \Gamma & \vdash z : P[\mathtt{zero}/x] \\ \Gamma, y : \mathbb{N}, h : P[y/x] & \vdash s : P[\mathtt{suc}\; y/x] \\ \Gamma & \vdash n : \mathbb{N} \end{array}}{\Gamma \vdash \mathtt{ind}\,(x.\, P)\; z\; (y.\, h.\, s)\; n : P[n/x]}\;(\mathbb{N}\mathsf{E})$$

Logically this expresses the *induction principle* on natural numbers; computationally this provides *primitive recursion*.

Natural numbers — computation rules

- Computation:

$$\frac{\begin{array}{ll} \Gamma, x : \mathbb{N} & \vdash P : \mathcal{U} \\ \Gamma & \vdash z : P[\mathtt{zero}/x] \\ \Gamma, y : \mathbb{N}, h : P[y/x] & \vdash s : P[\mathtt{suc}\ y/x] \end{array}}{\Gamma \vdash \mathtt{ind}\,(x.\,P)\,z\,(y.\,h.\,s)\,\mathtt{zero} = z \in P[\mathtt{zero}/x]}\ (\mathbb{N}\mathrm{CZ})$$

$$\frac{\begin{array}{ll} \Gamma, x : \mathbb{N} & \vdash P : \mathcal{U} \\ \Gamma & \vdash z : P[\mathtt{zero}/x] \\ \Gamma, y : \mathbb{N}, h : P[y/x] & \vdash s : P[\mathtt{suc}\ y/x] \\ \Gamma & \vdash n : \mathbb{N} \end{array}}{\begin{array}{l} \Gamma \vdash \mathtt{ind}\,(x.\,P)\,z\,(y.\,h.\,s)\,(\mathtt{suc}\ n) = \\ \qquad s[n, \mathtt{ind}\,(x.\,P)\,z\,(y.\,h.\,s)\,n/y, h] \in P[\mathtt{suc}\ n/x] \end{array}}\ (\mathbb{N}\mathrm{CS})$$

An induction combinator

We can define an induction combinator in terms of the more primitive '`ind`' eliminator:

$$\begin{aligned}
\mathit{induction} &: \Pi(P : \mathbb{N} \to \mathcal{U}) \\
&\quad P\,\texttt{zero} \to (\Pi(y : \mathbb{N})\ P\,y \to P\,(\texttt{suc}\ y)) \to \\
&\quad \Pi(n : \mathbb{N})\ P\,n \\
\mathit{induction} &= \lambda P.\,\lambda z.\,\lambda f.\,\lambda n.\ \texttt{ind}\,(x.\ P\,x)\ z\,(y.\ h.\ f\,y\,h)\ n
\end{aligned}$$

When we construct the theory of natural numbers, we will work "within type theory", focusing on constructing programs of closed types and leaving their typing derivations to informal reasoning or even the machine.

Exercise. Define predecessor, addition, and multiplication with *induction*.

Identity types

Identity types are also called *propositional equality*, especially when being contrasted with the meta-level judgemental equality.

- Formation:

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash \mathtt{Id}\ A\ a\ b : \mathcal{U}}\ (\mathtt{IdF})$$

- Introduction:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathtt{refl} : \mathtt{Id}\ A\ a\ a}\ (\mathtt{IdI})$$

Exercise. Assume $\Gamma \vdash a = b \in A$ and derive $\Gamma \vdash \mathtt{refl} : \mathtt{Id}\ A\ a\ b$.

Identity types — elimination and computation rules

- Elimination:

$$\frac{\begin{array}{ll} \Gamma, x : A, y : A, e : \mathtt{Id}\, A\, x\, y & \vdash P : \mathcal{U} \\ \Gamma, z : A & \vdash p : P[z, z, \mathtt{refl}/x, y, e] \\ \Gamma & \vdash a : A \\ \Gamma & \vdash b : A \\ \Gamma & \vdash q : \mathtt{Id}\, A\, a\, b \end{array}}{\Gamma \vdash \mathtt{J}\,(x.\, y.\, e.\, P)\,(z.\, p)\, a\, b\, q : P[a, b, q/x, y, e]}\ (\mathtt{IdE})$$

- Computation:

$$\frac{\begin{array}{ll} \Gamma, x : A, y : A, e : \mathtt{Id}\, A\, x\, y & \vdash P : \mathcal{U} \\ \Gamma, z : A & \vdash p : P[z, z, \mathtt{refl}/x, y, e] \\ \Gamma & \vdash a : A \end{array}}{\begin{array}{l} \Gamma \vdash \mathtt{J}\,(x.\, y.\, e.\, P)\,(z.\, p)\, a\, a\, \mathtt{refl} = p[a/z] \\ \qquad\qquad \in P[a, a, \mathtt{refl}/x, y, e] \end{array}}\ (\mathtt{IdC})$$

A substitution/transportation combinator

In this lecture we do not need the full power of $\mathtt{J}$, but only the following transportation combinator for substituting equals for equals in types:

$$\begin{aligned}&\mathit{transport} : \\ &\quad \Pi(P : A \to \mathcal{U})\ \Pi(x : A)\ \Pi(y : A)\ \mathtt{Id}\,A\,x\,y \to P\,x \to P\,y\end{aligned}$$

Exercise. Define *transport* in terms of $\mathtt{J}$.

Exercise. Using *transport*, prove that all functions respect identity types; that is, construct

$$\begin{aligned}\mathit{ap} : \Pi(A : \mathcal{U})\ \Pi(B : \mathcal{U})\ \Pi(f : A \to B)& \\ \Pi(x : A)\ \Pi(y : A)\ \mathtt{Id}\,A\,x\,y \to \mathtt{Id}\,B\,(f\,x)\,(f\,y)&\end{aligned}$$

$\texttt{Id}$ is an equivalence relation

$\texttt{Id}$ is obviously reflexive:

$$\lambda A.\, \lambda x.\, \texttt{refl} : \Pi(A : \mathcal{U})\ \Pi(x : A)\ \texttt{Id}\, A\, x\, x$$

Exercise. Prove that $\texttt{Id}$ is symmetric and transitive, i.e., construct

$$\mathit{sym} : \Pi(A : \mathcal{U})\ \Pi(x : A)\ \Pi(y : A)\ \texttt{Id}\, A\, x\, y \to \texttt{Id}\, A\, y\, x$$

and

$$\begin{aligned} \mathit{trans} : \Pi(A : \mathcal{U})\ \Pi(x : A)\ \Pi(y : A)\ \Pi(z : A) \\ \texttt{Id}\, A\, x\, y \to \texttt{Id}\, A\, y\, z \to \texttt{Id}\, A\, x\, z \end{aligned}$$

Peano axioms

Peano axioms specify an equational theory of natural number arithmetic (addition and multiplication); all of them are provable in type theory.

- Zero is a natural number. If n is a natural number, so is the successor of n.
  — The introduction rules.
- Equality on natural numbers is an equivalence relation.
  — We use `Id`, which has been proved to be an equivalence relation.
- The successor operation is an injective function, i.e.,

  $$\Pi(m : \mathbb{N})\ \Pi(n : \mathbb{N})\ \mathtt{Id}\ \mathbb{N}\ m\ n \leftrightarrow \mathtt{Id}\ \mathbb{N}\ (\mathtt{suc}\ m)\ (\mathtt{suc}\ n)$$

- The successor operation never yields zero, i.e.,

  $$\Pi(n : \mathbb{N})\ \neg\,\mathtt{Id}\ \mathbb{N}\ (\mathtt{suc}\ n)\ \mathtt{zero}$$

Peano axioms

- Addition satisfies

$$\Pi(n : \mathbb{N})\ \mathtt{Id}\ \mathbb{N}\ (\mathtt{zero} + n)\ n$$

  and

$$\Pi(m : \mathbb{N})\ \Pi(n : \mathbb{N})\ \mathtt{Id}\ \mathbb{N}\ ((\mathtt{suc}\ m) + n)\ (\mathtt{suc}\ (m + n))$$

- Multiplication satisfies

$$\Pi(n : \mathbb{N})\ \mathtt{Id}\ \mathbb{N}\ (\mathtt{zero} \times n)\ \mathtt{zero}$$

  and

$$\Pi(m : \mathbb{N})\ \Pi(n : \mathbb{N})\ \mathtt{Id}\ \mathbb{N}\ ((\mathtt{suc}\ m) \times n)\ (n + m \times n)$$

- The induction principle holds for natural numbers.
  — The *induction* combinator.

The traditional approach

- Formalise a deduction system for first-order logic (propositional logic extended with first-order quantification).
- Collect the Peano axioms into a set $\mathbf{PA}$.
- Start deriving theorems from $\mathbf{PA}$, like

$$\mathbf{PA} \vdash \forall x.\ \forall y.\ x + y = y + x$$

This approach leads to, for example, an overblown proof of $1 + 1 = 2$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{PA} \vdash \textit{transitivity}}}{\mathbf{PA} \vdash \forall \mathtt{y}.\ \forall \mathtt{z}.\ \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{y} \land \mathtt{y} = \mathtt{z} \to \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{z}}\,(\forall\mathrm{E})}{\mathbf{PA} \vdash \forall \mathtt{z}.\ \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (zero} + \mathtt{suc\ zero)} \land \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{z} \to \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{z}}\,(\forall\mathrm{E})}{\mathbf{PA} \vdash \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (zero} + \mathtt{suc\ zero)} \land \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{suc\ (suc\ zero)} \to \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (suc\ zero)}}\,(\forall\mathrm{E}) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{PA} \vdash \textit{additionS}}}{\mathbf{PA} \vdash \forall \mathtt{y}.\ \mathtt{suc\ zero} + \mathtt{y} = \mathtt{suc\ (zero} + \mathtt{y)}}\,(\forall\mathrm{E})}{\mathbf{PA} \vdash \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (zero} + \mathtt{suc\ zero)}}\,(\forall\mathrm{E}) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{PA} \vdash \textit{congruence}}}{\mathbf{PA} \vdash \forall \mathtt{y}.\ \mathtt{zero} + \mathtt{suc\ zero} = \mathtt{y} \to \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{suc\ y}}\,(\forall\mathrm{E})}{\mathbf{PA} \vdash \mathtt{zero} + \mathtt{suc\ zero} = \mathtt{suc\ zero} \to \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{suc\ (suc\ zero)}}\,(\forall\mathrm{E}) \quad \dfrac{\dfrac{}{\mathbf{PA} \vdash \textit{additionZ}}}{\mathbf{PA} \vdash \mathtt{zero} + \mathtt{suc\ zero} = \mathtt{suc\ zero}}\,(\forall\mathrm{E})}{\mathbf{PA} \vdash \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{suc\ (suc\ zero)}}\,(\to\mathrm{E})}{\mathbf{PA} \vdash \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (zero} + \mathtt{suc\ zero)} \land \mathtt{suc\ (zero} + \mathtt{suc\ zero)} = \mathtt{suc\ (suc\ zero)}}\,(\land\mathrm{I})}{\mathbf{PA} \vdash \mathtt{suc\ zero} + \mathtt{suc\ zero} = \mathtt{suc\ (suc\ zero)}}\,(\to\mathrm{E})$$

Computational foundation

In type theory, Peano "axioms" are merely consequences, and do not play an important role in actual theorem proving.

We now have a more natural foundation of mathematics based on the unifying idea of typed computation.

We do not "prove" $1 + 1 = 2$, but just "check" it mechanically:

$$\texttt{refl} : \texttt{Id}\,\mathbb{N}\,(\texttt{suc}\,\texttt{zero} + \texttt{suc}\,\texttt{zero})\,(\texttt{suc}\,(\texttt{suc}\,\texttt{zero}))$$