Beyond Inductive Definitions – Induction-Recursion, Induction-Induction, Coalgebras

Anton Setzer

Swansea University, Swansea UK

1 March 2012

Foundations of Mathematics

- By Gödel's Incompleteness Theorem we know we cannot prove absolutely that any reasonably strong axiomatization of mathematics is consistent.
- Even though we have a verified proof of Fermat's last theorem, we cannot exclude that there is a counterexample.
  - This is of course highly unlikely.
- When giving a mathematical proof of a number theoretic theorem, all we know is that it is extremely highly likely that it is correct.
- Mathematics doesn't have an entirely rational (in the sense of logical) foundation.
- Any foundation of mathematics will necessarily rely on a philosophical argument.
  - Such arguments are never as robust as mathematical arguments.
  - Such arguments need to rely on intuitive insights.

Foundations based on Ordinal Notation Systems

- The best we can do is to formulate certain principles in which we can put our trust.
- Gentzen's analysis of Peano Arithmetic based its trust on the well-foundedness of an ordinal notation system up to ϵ_0.
- One can push this further up to $(\Pi^1_1 - \mathrm{CA})_0$ (AS: Ordinal systems).
- However, it becomes increasingly difficult to get a direct insight into the well-foundedness of ordinal notation systems.
- Proof theoretic strong ordinal notation systems are currently too complicated to get a direct insight into their well-foundedness.

Constructive Foundations

- An alternative is to formulate theories in which we can directly put our trust.
- If we prove in such theories the well-foundedness of strong ordinal notation system we can prove the consistency of strong mathematical theories.
- The best approach seem to be based on constructive approaches:
  - Elements of sets are programs or terms with reductions.
  - One constructs from below sets (or types) based on principles into which we have a certain insight.
  - Then one formulates principles for adding elements into those sets and constructing elements of other sets from elements of this set.
  - We need to get an intuitive argument why this combination of introduction and elimination principle always gives correct results (termination).

Picture

$$\begin{tikzcd}
\text{Mathematical Theory} \arrow[d, "\text{Ordinal Analysis}"] \\
\begin{array}{c}\text{Well-foundedness of} \\ \text{Ordinal Notation System}\end{array} \arrow[d, "\text{Well-foundedness Proof}"] \\
\text{Constructive Theory}
\end{tikzcd}$$

No Need to Change Mathematical Practice Completely

- It is not necessary to completely switch to constructive mathematics.
- Once the consistency of a mathematical theory is shown, one has a mathematically sound working environment.
- And one knows that at least numerical Π^0_1-statements, and with some deeper analysis Π^0_2-statements are correct (provable in the constructive theory).

Two Main Candidates for Constructive Theories

- Martin-Löf Type Theory.
  - Based on type theory with total functions.
  - Philosophically the best developed theory (meaning explanations).
  - According to Martin-Löf the most serious attempt to build a theory into which we can put our trust.
  - Or: a theory which formulates the reasons why we can put our trust into it.
- Feferman's theory of explicit mathematics.
  - Based on partial functions.
  - Everything can be applied to everything.
  - Formulated axiomatically with an attempt to get a short definition.
    - Makes it more easy to carry out metamathematical analysis.
    - However this destroys some of its philosophical clarity.
  - More suitable for giving foundations of principles for Mahlo and beyond. (AS and R. Kahle: Formulation of extended predicative Mahlo.)
- In the following we follow mainly Martin-Löf Type Theory.

Applications

- In order to formulate proof theoretically strong theories, new sets or data types need to be introduced.
- These data types can be used in general computing.
- Best example: The Mahlo universe.
    - Was introduced to define a very strong predicatively justified type theory.
    - The data type of inductive-recursive definitions was defined using the same principles as the Mahlo universe.
    - This data type was
        - applied to generic programming,
        - is related to generic extensions of Haskell.

Principles for Formulating Sets – Finite Sets

- Finite sets.
  - There are finitely many fixed elements which are elements of it.
  - If from each of these finitely many elements we can construct an element of another set, then we can construct an element from an arbitrary element of this set by case distinction.

Principles for Formulating Sets – $(x : A) \times B[x]$

- $B[a]$ stands for $B[x := a]$ for some variable x.
- Dependent sum type $(x : A) \times B[x]$.
  - If $a : A$ and $b : B[a]$ then can introduce $\langle a, b \rangle : (x : A) \times B[x]$.
  - If $c : (x : A) \times B[x]$, we obtain $\pi_0(c) : A$ and $\pi_1(c) : B[\pi_0(c)]$.
- Using BHK interpretation
  - $\exists x : A.\varphi(x) = (x : A) \times \varphi[x]$
  - $\varphi \wedge \psi = \varphi \times \psi$.

Principles for Formulating Sets – $(x : A) \to B[x]$

- Formulation of the dependent function type $(x : A) \to B[x]$.
  - Elements of $(x : A) \to B[x]$ are programs f which if applied to $a : A$ compute an element of $B[a]$.
  - If from $x : A$ we can construct an element $b[x] : B[x]$, then $\lambda x.b[x]$ is an element of $(x : A) \to B[x]$.
    Comutes from $a : A$ the element $b[a]$.
  - In fact this is a form of coalgebraic definition.
- Using BHK interpretation
  - $\forall x : A.\varphi(x) = (x : A) \to \varphi[x]$.
  - $\varphi \supset \psi = \varphi \to \psi$.

Principles for Formulating Sets – $\mathbb{N}$

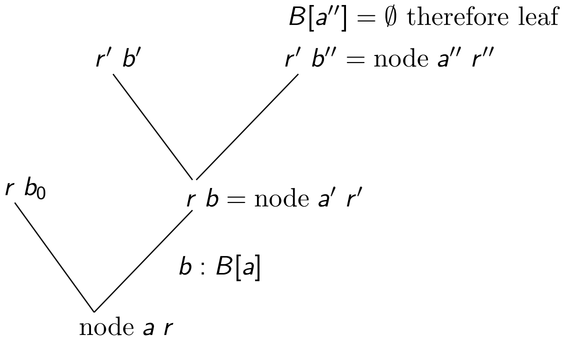
- Formulation of $\mathbb{N}$.
  - $0 : \mathbb{N}$ and if $n : \mathbb{N}$ then $\mathrm{S}\ n : \mathbb{N}$.
  - Assume we can form
    - an element of $A[0]$
    - from $n : \mathbb{N}$ and $a : A[n]$ an element of $A[\mathrm{S}\ n]$.

  Then we can form from $n : \mathbb{N}$ an element of $A[n]$.

Principles for Formulating Sets – W

- Assume $A : Set$ and $B[x] : \mathrm{Set}$ whenever $x : A$.
- We formulate $\mathrm{W}\ x : A.B[x]$:
  - Whenever $a : A$ and $r : B[a] \to \mathrm{W}\ x : A.B[x]$, then $\mathrm{node}\ a\ r : \mathrm{W}\ x : A.B[x]$.
  - Assume that from $a : A$, $r : B[a] \to \mathrm{W}\ x : A.B[x]$ and $s : (b : B[a]) \to C[r\ b]$ we can define $C[\mathrm{node}\ a\ r]$. Then we can construct $C[c]$ for every $c : \mathrm{W}\ x : A.B[x]$.

Picture


$B[a''] = \emptyset$ therefore leaf
$r'\ b'$
$r'\ b'' = \text{node}\ a''\ r''$
$r\ b_0$
$r\ b = \text{node}\ a'\ r'$
$b : B[a]$
$\text{node}\ a\ r$

Principles for Formulating Sets – U

- A universe U is a collection of sets.
- Formulated as a set U of codes for sets and a decoding function $\mathrm{T} : \mathrm{U} \to \mathrm{Set}$.
- We define $\mathrm{U} : \mathrm{Set}$ while recursively defining for every $u : \mathrm{U}$ a set $\mathrm{T}\ u$:
  - $\widehat{\mathbb{N}} : \mathrm{U}$
    $\mathrm{T}\ \widehat{\mathbb{N}} = \mathbb{N}$.
  - If $a : \mathrm{U}$ and $b : \mathrm{T}\ a \to \mathrm{U}$ then $\widehat{\Pi}\ a\ b : \mathrm{U}$.
    $\mathrm{T}\ (\widehat{\Pi}\ a\ b) = (x : \mathrm{T}\ a) \to \mathrm{T}\ (b\ x)$.
  - Furthermore assume $C[x] : \mathrm{Set}$ depending on $x : \mathrm{U}$. Assume
    - we can form $C[\widehat{\mathbb{N}}]$;
    - from $a : \mathrm{U}$, $C[a]$, $r : \mathrm{T}\ a \to \mathrm{U}$ and $(x : \mathrm{T}\ a) \to C[b\ x]$ we can form $C[\mathrm{sup}\ \ a\ r]$.

  Then we can form $C[r]$ from every $r : \mathrm{U}$.

Induction-Recursion

- U is the paradigm example of an inductive-recursive definition.
- In general let D be a type.
  We can define an $A : \mathrm{Set}$ inductively while recursively defining $B[a] : D$ for $a : A$.
- A can defined using constructors which are
  - strictly positive in A
  - can refer to $B[a]$ for any $a : A$ referred to
- Whenever we introduce an element $a : A$ we need to introduce $B[a]$.
- Induction-Recursion subsumes all standard extensions of Martin-Löf Type Theory at the time of formulating it.
- Strength of type theory with induction-recursion at least KPM.

Example Fresh Lists

- We define $\mathrm{Freshlist} : \mathrm{Set}$ inductively defining for $l : \mathrm{Freshlist}$ and $n : \mathbb{N}$ the Boolean $n \in l$ recursively:
  - $\mathrm{nil} : \mathrm{Freshlist}$ and $n \in \mathrm{nil} = \mathrm{false}$.
  - If $l : \mathrm{Freshlist}$, $n : \mathbb{N}$ and $n \in l = \mathrm{false}$, then $\mathrm{cons}\ n\ l : \mathrm{Freshlist}$.
    $m \in \mathrm{cons}\ n\ l = m \in l \vee_{\mathbb{B}} (m ==_{\mathbb{B}} n)$.

Induction-Induction

- Joint work with Fredrik Forsberg.
- We define A inductively while defining $B\ a$ for $a : A$ simultaneously inductively.
- Constructors of A and $B\ a$ need to be strictly positive in A and $B\ a'$ referred to.
- Complication: $B\ a$ can refer to $B\ c$ for elements c constructed using the constructors of A.

Example

- Syntax of type theory is defined inductive-inductively:
- We define $\mathrm{Context} : \mathrm{Set}$ and $\mathrm{Type}\ \Gamma$ for $\Gamma : \mathrm{Context}$ inductive-inductively:
  - $\emptyset : \mathrm{Context}$.
  - If $\Gamma : \mathrm{Context}$ and $A : \mathrm{Type}\ \Gamma$, then $(\Gamma, A) : \mathrm{Context}$.
  - If $\Gamma : \mathrm{Context}$ then $\widehat{\mathbb{N}} : \mathrm{Type}\ \Gamma$.
  - If $\Gamma : \mathrm{Context}$, $A : \mathrm{Type}\ \Gamma$, and $B : \mathrm{Type}\ (\Gamma, A)$, then $\widehat{\Pi}\ A\ B : \mathrm{Type}\ \Gamma$.
- Conways surreal numbers are an example of an extended form of induction-induction.

Coalgebras in Type Theory

- Whereas inductive data types are given by their introduction rules, coalgebras are given by their elimination rules.
- Elements of a coalgebras are everything which allows the elimination principle to be applied for.

Example: $\mathrm{co}\mathbb{N}$

- Elements n of $\mathrm{co}\mathbb{N}$ are programs such that we can define $\mathrm{case}\ n : (\{0\} + \mathrm{S}\ \mathbb{N})$.
- Assume $A : \mathrm{Set}$ and $f : A \to (\{0\} + \mathrm{S}\ A)$.
  Then $\mathrm{intro}\ A\ f : A \to \mathrm{co}\mathbb{N}$.
  If $f\ a = 0$ then $\mathrm{case}\ (\mathrm{intro}\ A\ f\ a) = 0$.
  If $f\ a = \mathrm{S}\ a'$ then $\mathrm{case}\ (\mathrm{intro}\ A\ f\ a) = \mathrm{S}\ (\mathrm{intro}\ A\ f\ a')$.
- Example: Let $A = \{0\}$, $f : A \to \{0\} + \mathrm{S}\ A$, $f\ a = \mathrm{S}\ a$.
  Let $b = \mathrm{intro}\ A\ f\ 0$.
  Then $\mathrm{case}\ b = \mathrm{S}\ b$.

Research Questions

- Can we extend the principle of Induction-Recursion to include the Mahlo principle, and combinations of the Mahlo principle and inductive definitions?
- Can obtain a predicatively justified type theory of strength $(\Pi^1_2 - \mathrm{CA})_0$ or beyond?