Extensions of Inductive Definitions: Indexed Inductive, Inductive-Recursive and Inductive-Inductive Definitions

Anton Setzer
Swansea University
Swanesa, UK

Parts on indexed induction/induction-recursion joint work with Peter Dybjer.
Parts on induction-induction joint work with Frederik Forsberg.
Parts on extended predicative Mahlo joint work with Reinhard Kahle.

September 8, 2010

Preliminaries

- Martin-Löf Type Theory $=$ λ-calculus extended by dependent types, inductive-recursive definitions.
- **Propositions as types**.
  $t : A$ could mean:
  - t is an element of the data type A,
  - t is a proof of proposition A.

Notations

- We use functional notation for application so $f\ a\ b$ is what in standard mathematics is denoted by $f(a)(b)$.
- Notations like $\_ :: \_$ for mixfix symbols
  - $s :: t$ instead of $\_ :: \_\ s\ t$.
- Set is the collection of small types,
  Type is the collection of large types.
  Example: Type of matrices depending on dimensions is

$$\begin{array}{lcl} \mathrm{Matrix} & : & \mathbb{N} \to \mathbb{N} \to \mathrm{Set} \\ \mathbb{N} \to \mathbb{N} \to \mathrm{Set} & : & \mathrm{Type} \end{array}$$

Notations

- $$(x : A) \to B$$

  is type of functions f mapping $x : A$ to $f\ x : B$ where B might dependent on A.
  Example: Matrix multiplication:

  $$\mathrm{matmult} : (n, m, k : \mathbb{N}) \to \mathrm{Matrix}\ n\ m \to \mathrm{Matrix}\ m\ k \to \mathrm{Matrix}\ n\ k$$

  Other people use $\Pi x : A.B$
  (subtle difference in Martin-Löf Type Theory).
- $\{x : A\} \to B$ for a hidden argument (usually omitted, can be inferred).

Notations

- $$(x : A) \times B$$
  is the dependent cross-product.
  Elements are $\langle x, y \rangle$ where $x : A$ and $y : B$ where B might dependent on A.
  Example: sorted lists
  $$\begin{aligned} \mathrm{SortedList} &= (l : \mathrm{List}) \times \mathrm{Sorted}\ l \\ \mathrm{SortedList} &: \mathrm{Set} \end{aligned}$$
  Other people use $\Sigma x : A.B$
  (subtle difference in Martin-Löf Type Theory).

Inductive Definitions

Inductive Definitions given essentially as algebraic data types.
Given as a set A together with constructors which are strictly positive in A.
Example using Agda notation.

$$\begin{array}{l} \mathrm{data}\ \mathrm{List}\mathbb{N} : \mathrm{Set}\ \ \mathrm{where} \\ \quad \begin{array}{lcl} [] & : & \mathrm{List}\mathbb{N} \\ \_ :: \_ & : & \mathbb{N} \to \mathrm{List}\mathbb{N} \to \mathrm{List}\mathbb{N} \end{array} \end{array}$$

Means that we have

$$\begin{array}{lcl} \mathrm{List}\mathbb{N} & : & \mathrm{Set} \\ [] & : & \mathrm{List}\mathbb{N} \\ \_ :: \_ & : & \mathbb{N} \to \mathrm{List}\mathbb{N} \to \mathrm{List}\mathbb{N} \end{array}$$

Induction Principle

$$
\begin{array}{l}
\mathrm{data}\ \mathrm{List}\mathbb{N} : \mathrm{Set}\ \mathrm{where} \\
\quad [] \quad : \quad \mathrm{List}\mathbb{N} \\
\quad \_ :: \_ \quad : \quad \mathbb{N} \to \mathrm{List}\mathbb{N} \to \mathrm{List}\mathbb{N}
\end{array}
$$

Additionally we have an induction principle expressing $\mathrm{List}\mathbb{N}$ is least set closed under these constructors:

$$
\begin{array}{l}
\mathrm{inductionList}\mathbb{N} : (A : \mathrm{List}\mathbb{N} \to \mathrm{Set}) \\
\qquad \to (\mathrm{step}_{[]} : A\ []) \\
\qquad \to (\mathrm{step}_{::} : (n : \mathbb{N}) \to (l : \mathrm{List}\mathbb{N}) \to A\ l \to A\ (n :: l)) \\
\qquad \to (l : \mathrm{List}\mathbb{N}) \\
\qquad \to A\ l \\
\mathrm{inductionList}\mathbb{N}\ A\ \mathrm{step}_{[]}\ \mathrm{step}_{::}\ [] \quad = \quad \mathrm{step}_{[]} \\
\mathrm{inductionList}\mathbb{N}\ A\ \mathrm{step}_{[]}\ \mathrm{step}_{::}\ (n :: l) \quad = \\
\qquad \mathrm{step}_{::}\ n\ l\ (\mathrm{inductionList}\mathbb{N}\ A\ \mathrm{step}_{[]}\ \mathrm{step}_{::}\ l)
\end{array}
$$

(We won't mention those induction principles in the future).

Models in General

- Our models will be PER modules, e.g.
  - We have a set of raw terms Term plus a reduction relation $\longrightarrow$ on it which is confluent.
  - Sets A are interpreted as partial equivalence relations on Term,

$$[\![A]\!] \subseteq \mathrm{Term} \times \mathrm{Term}$$

$$\langle r, s\rangle \in [\![A]\!] \text{ means that } r \text{ and } s \text{ are equal elements in } [\![A]\!]$$
$$\langle r, r\rangle \in [\![A]\!] \text{ means } r \text{ is an element of } [\![A]\!]$$

  - For simplicity we will usually omit dealing with the fact that sets have an equality on them and do as if

$$[\![A]\!] \subseteq \mathrm{Term}$$

  - $[\![\mathrm{Set}]\!] := \mathcal{P}(\mathrm{Term})(= \{X \mid X \subseteq \mathrm{Term}\})$, the set of interpretations of sets (with this simplication).

Model

$\mathrm{List}\mathbb{N}$ defined as the least fixed point of a monotone operator Γ on the cpo

- $[\![\, \mathrm{Set} \,]\!]$
- with ordering $X \leq Y :\Leftrightarrow X \subseteq Y$.

where

$$\begin{aligned} \Gamma &\in [\![\, \mathrm{Set} \,]\!] \to [\![\, \mathrm{Set} \,]\!] \\ \Gamma(X) &= \mathrm{Closure}_{\longrightarrow}(\{[]\} \cup \{n :: l \mid n \in [\![\, \mathbb{N} \,]\!] \wedge l \in X\}) \end{aligned}$$

When defining models as fixed point, we will for simplicity omit $\mathrm{Closure}_{\longrightarrow}$ (upward closure under $\longrightarrow$).

Inductive and Non-inductive Arguments

$$\begin{array}{lcl} \mathrm{List}\mathbb{N} & : & \mathrm{Set} \\ [] & : & \mathrm{List}\mathbb{N} \\ \_ :: \_ & : & \mathbb{N} \to \mathrm{List}\mathbb{N} \to \mathrm{List}\mathbb{N} \end{array}$$

- The first argument of $\_ :: \_$ is a **non-inductive argument**.
  - Refers to a set defined before one introduced $\mathrm{List}\mathbb{N}$.
- The second argument is an **inductive argument**.

Parametrised Inductive Definitions

The above type can be made generic in the set argument.

$$
\begin{array}{l}
\mathrm{data}\ \mathrm{List}\ (A : \mathit{Set}) : \mathrm{Set}\ \mathrm{where} \\
\quad [] \quad\quad : \ \mathrm{List}\ A \\
\quad \_ :: \_ \quad : \ A \rightarrow \mathrm{List}\ A \rightarrow \mathrm{List}\ A
\end{array}
$$

A is a uniform parameter:

- The result type of the constructor is $\mathrm{List}\ A$ for arbitrary A
  - Constructors

    $$\mathrm{C} : \mathrm{List}\ \mathbb{N}$$

    or

    $$\mathrm{C}' : (A : \mathrm{Set}) \rightarrow \mathrm{List}\ (A \times A)$$

    are not allowed
- The constructor refers to the same set $\mathrm{List}\ A$.

Infinitary Inductive Definitions

We can as well have constructors having **infinitary inductive arguments**.

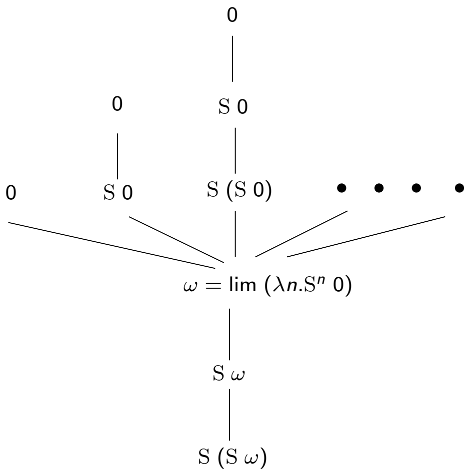
Example

$$\begin{array}{l}\mathrm{data}\ \mathrm{KleenesO} : \mathrm{Set}\ \mathrm{where} \\ \quad \begin{array}{lcl} 0 & : & \mathrm{KleenesO} \\ \mathrm{S} & : & \mathrm{KleenesO} \to \mathrm{KleenesO} \\ \mathsf{lim} & : & (\mathbb{N} \to \mathrm{KleenesO}) \to \mathrm{KleenesO} \end{array}\end{array}$$

Height of $\mathrm{KleenesO}$ is $\aleph_1^{\mathrm{rec}}$.
Iterations of this type allows to define finitely iterated inductive definitions or ordinals of height $\aleph_n^{\mathrm{rec}}$.
Corresponding operator is

$$\begin{aligned}\Gamma(X) = \{0\} \cup \{\mathrm{S}\ x \mid x \in X\} \\ \cup \{\mathsf{lim}\ f \mid \forall n \in [\![\mathbb{N}]\!]. f\ n \in X\}\end{aligned}$$

Kleene's O



Dependencies in Constructors

Because of dependent type theory, later arguments can depend on previous arguments:

- Only on **non-inductive** arguments.
- When introducing the new set, say A, we don't know what A is, so cannot define a type truly depending on $a : A$.

Example:

$$\begin{array}{l} \mathrm{data}\ \mathrm{W}\ (A : \mathrm{Set})\ (B : A \to \mathrm{Set}) : \mathrm{Set}\ \mathrm{where} \\ \quad \mathsf{sup} \quad : \quad (a : A) \to (B\ a \to \mathrm{W}\ A\ B) \to \mathrm{W}\ A\ B \end{array}$$

W-type

$$\mathrm{data}\ \mathrm{W}\ (A : \mathrm{Set})\ (B : A \to \mathrm{Set}) : \mathrm{Set}\ \mathrm{where}$$
$$\mathsf{sup} \quad : \quad (a : A) \to (B\ a \to \mathrm{W}\ A\ B) \to \mathrm{W}\ A\ B$$

We can see

$$\begin{aligned} \mathrm{KleenesO} &\approx \mathrm{W} x : \{0,1,2\}.B\ x \\ \text{where } B\ 0 &= \emptyset \\ B\ 1 &= \{*\} \\ B\ 2 &= \mathbb{N} \end{aligned}$$

Third number class (of height $\aleph_3^{\mathrm{rec}}$) can be defined as

$$\begin{aligned} \mathrm{KleenesO}_2 &:= \mathrm{W} x : \{0,1,2,3\}.B\ x \\ &\text{where } B\ 0, B\ 1, B\ 2 \text{ as before} \\ &B\ 3 = \mathrm{KleenesO} \end{aligned}$$

Iterating $\mathrm{KleenesO}$ infinitely often

We can form as well if can define the nth iteration of $\mathrm{KleenesO}$ (by having large elimination on $\mathbb{N}$ or a universe closed under W-type)

$$\mathrm{KleenesO} : \mathbb{N} \to \mathrm{Set}$$

the set

$$\mathrm{KleenesO}_\omega = \mathrm{W} n : \mathbb{N}.\mathrm{KleenesO}_n$$

of height

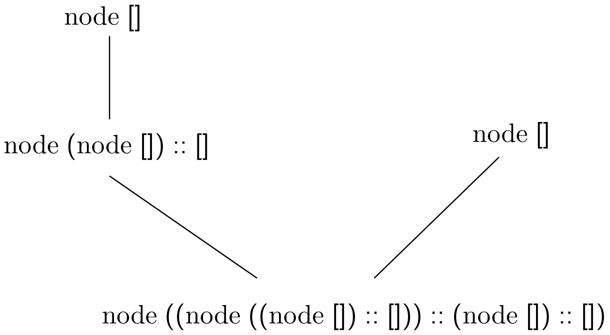
$$\aleph_\omega^{\mathrm{rec}} = \sup_{n \in \omega} \aleph_n^{\mathrm{rec}}$$

Simultaneous Inductive Definitions

Simultaneous inductive definitions allow to define several sets inductively simultaneously:
Finitely branching trees:

$$
\begin{array}{l}
\text{mutual} \\
\quad \text{data Fintree} : \text{Set where} \\
\qquad \text{node} : \text{FintreeList} \to \text{Fintree} \\
\\
\quad \text{data FintreeList} : \text{Set where} \\
\qquad [] \quad : \quad \text{FintreeList} \\
\qquad \_ :: \_ \quad : \quad \text{Fintree} \to \text{FintreeList} \to \text{FintreeList}
\end{array}
$$

Example of FinTree



Model

Consider the cpo

- $[\![\, \mathrm{Set} \,]\!]^2 (= [\![\, \mathrm{Set} \,]\!] \times [\![\, \mathrm{Set} \,]\!])$
- $\langle X, Y \rangle \leq \langle X', Y' \rangle :\Leftrightarrow X \subseteq X' \wedge Y \subseteq Y'$.

Let $\Gamma \in [\![\, \mathrm{Set} \,]\!]^2 \to [\![\, \mathrm{Set} \,]\!]^2$ monotone s.t.

$$\begin{aligned}\Gamma(\langle X, Y \rangle) = \langle & \{\mathrm{node}\, x \mid x \in Y\}, \\ & \{[]\} \cup \{x :: y \mid x \in X \wedge y \in Y\} \rangle\end{aligned}$$

We then define

$$\langle [\![\, \mathrm{Fintree} \,]\!], [\![\, \mathrm{FintreeList} \,]\!] \rangle$$

as the last fixed point of Γ.

Indexed Inductive Definitions

Generalised Inductive Definitions introduce sets $A : I \to \mathrm{Set}$ for some index set I simultaneously.

Example:

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathrm{data\ Vector} : \mathbb{N} \to \mathrm{Set\ \ where}} \\
\quad [] & : & \mathrm{Vector}\ 0 \\
\quad \_ :: \_ & : & \{n : \mathbb{N}\} \to \mathbb{N} \to \mathrm{Vector}\ n \to \mathrm{Vector}\ (n+1)
\end{array}
$$

E.g.

$$(3 :: 2 :: []) : \mathrm{Vector}\ 2$$

Generalised Indexed Inductive Definitions

$$
\begin{array}{l}
\mathrm{data\ Vector} : \mathbb{N} \rightarrow \mathrm{Set\ where} \\
\quad [] \quad : \quad \mathrm{Vector}\ 0 \\
\quad \_ :: \_ \quad : \quad \{n : \mathbb{N}\} \rightarrow \mathbb{N} \rightarrow \mathrm{Vector}\ n \rightarrow \mathrm{Vector}\ (n+1)
\end{array}
$$

This is a generalised indexed inductive definition:

- index of the result type of a constructor arbitrary,
- constructor can refer to elements for this set for arbitrary other indices.

Restricted Indexed Inductive Definitions

An example of a restricted indexed inductive definition is (assuming $A : \mathrm{Set}$, $\_ < \_ : A \to A \to \mathrm{Set}$)

$$\begin{array}{l} \mathrm{data\ Acc} : A \to \mathrm{Set\ where} \\ \quad \mathrm{acc} \quad : \quad (a : A) \to ((b : A) \to (b < a) \to \mathrm{Acc}\ b) \to \mathrm{Acc}\ a \end{array}$$

The constructor (there could be several) of a restricted indexed inductive definition of a set $A : I \to \mathrm{Set}$ has the form

$$\mathrm{C} : (i : I) \to \cdots \to (f : (b : B) \to A\,(g\ i)) \to \cdots \to A\ i$$

- result type is A applied to a variable,
- that variable is the first argument of the constructor,
- the constructor can refer to $A\ i'$ for arbitrary i'.

Reason for Restricted Indexed Inductive Definitions

Restricted indexed inductive definitions allow definition by case distinction: Assume

$$\begin{array}{l} \mathrm{data}\ A : I \to \mathrm{Set\ where} \\ \quad \mathrm{C}_0 \quad : \quad (i : I) \to A\ i \\ \quad \mathrm{C}_1 \quad : \quad (i : I) \to (b : B) \to A\ t \to A\ i \end{array}$$

We can define for $t : I$

$$\begin{array}{l} f : A\ t' \to C \\ f\ x = \mathrm{case}\ x\ \mathrm{of}\{(\mathrm{C}_0\ t)) \quad \longrightarrow \quad \cdots \\ \qquad\qquad\qquad\quad (\mathrm{C}_1\ t\ b\ a) \quad \longrightarrow \quad \cdots \end{array}$$

Possible since for $A\ t'$ can be introduced by all constructors.

Reason for Restricted Indexed Inductive Definitions

In case of generalised indexed inductive definitions this is not possible.
Consider

$$\begin{array}{l} \mathrm{data}\ A : (\mathbb{N} \to \mathbb{N}) \to \mathrm{Set\ where} \\ \quad \mathrm{C}_0 \quad : \quad A\,(\lambda x.x) \\ \quad \mathrm{C}_1 \quad : \quad (b : B) \to A\ t' \to A\,(\lambda x.\mathrm{S}\ x) \end{array}$$

- We cannot define

$$\begin{array}{l} f : (g : \mathbb{N} \to \mathbb{N}) \to A\ g \to \cdots \\ f\ \ g\ \ x = \mathrm{case}\ x\ \mathrm{of} \cdots \end{array}$$

  because we don't know whether $g = \lambda x.x$ or not.
- However, we can define f by pattern matching ("." in front of an argument means that this argument is enforced by matching of another argument with a pattern):

$$\begin{array}{l} f : (g : \mathbb{N} \to \mathbb{N}) \to A\ g \to \cdots \\ f\ .(\lambda x.x)\ \mathrm{C}_0 = \cdots \\ f\ .(\lambda x.\mathrm{S}\ x)\ (\mathrm{C}\ b\ a) = \cdots \end{array}$$

Pattern Matching

- So we could deal with pattern matching for a restricted indexed inductive definition $H : I \to \mathrm{Set}$ if the function was of type

$$f : (i : I) \to H\ i \to \cdots$$

- We can as well deal with the situation where I is an inductive definition, the result types of the constructors start with certain constructors and the index of an argument starts with one constructor, e.g.

$$\begin{array}{l} \mathrm{data}\ H : \mathbb{N} \to \mathrm{Set}\ \mathrm{where} \\ \quad \begin{array}{lcl} \mathrm{C}_0 & : & H\ 0 \\ \mathrm{C}_1 & : & (n : \mathbb{N}) \to H\ (n+4) \to H\ (\mathrm{S}\ (\mathrm{S}\ n)) \end{array} \end{array}$$

$$\begin{array}{l} f : H\ (\mathrm{S}\ (\mathrm{S}\ (\mathrm{S}\ 0))) \to \mathbb{N} \\ f\ (\mathrm{C}_1\ .(\mathrm{S}\ 0))\ x = \cdots \end{array}$$

Reduction of Generalised to Restricted Indexed Inductive Definitions

- Assuming an equality on arbitrary sets, generalised indexed inductive definitions can be reduced to restricted ones:

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathrm{data}\ A : (\mathbb{N} \to \mathbb{N}) \to \mathrm{Set\ where}} \\
\quad \mathrm{C}_0 & : & A\,(\lambda x.x) \\
\quad \mathrm{C}_1 & : & (b : B) \to A\ t' \to A\,(\lambda x.\mathrm{S}\ x)
\end{array}
$$

can be replaced by

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathrm{data}\ A' : (\mathbb{N} \to \mathbb{N}) \to \mathrm{Set\ where}} \\
\quad \mathrm{C}_0 & : & (f : \mathbb{N} \to \mathbb{N}) \to f == \lambda x.x \to A\ f \\
\quad \mathrm{C}_1 & : & (f : \mathbb{N} \to \mathbb{N}) \to (b : B) \to A\ t' \to (f == \lambda x.\mathrm{S}\ x) \to A\ f
\end{array}
$$

Reduction of Generalised to Restricted Indexed Inductive Definitions

- However the equality itself needs to be defined and can be defined by a generalised indexed inductive definition:

$$\begin{array}{l} \text{data } \_ == \_ \ A : \text{Set}(a : A) : A \to \text{Set where} \\ \quad \text{refl} : a == a \end{array}$$

Model

Consider a generalised indexed inductive definition $A : I \to \mathrm{Set}$.
Model uses the cpo

- $[\![\mathrm{Set}]\!]^I \ (= I \to [\![\mathrm{Set}]\!])$
- $f \leq g :\Leftrightarrow \forall i \in I. f\ i \subseteq f\ j$.

and for instance in case of

$$\begin{array}{l} \mathrm{data}\ A : (\mathbb{N} \to \mathbb{N}) \to \mathrm{Set}\ \mathrm{where} \\ \qquad \mathrm{C}_0 \quad : \quad A\,(\lambda x.x) \\ \qquad \mathrm{C}_1 \quad : \quad (b : B) \to A\ t' \to A\,(\lambda x.\mathrm{S}\ x) \end{array}$$

$$\begin{array}{l} \Gamma \in ((\mathbb{N} \to \mathbb{N}) \to \mathrm{Set}) \to ((\mathbb{N} \to \mathbb{N}) \to \mathrm{Set}) \\ \Gamma\ X\ f = \{\mathrm{C}_0 \mid f = \lambda x.x\} \cup \{\mathrm{C}_1\ b\ a \mid f = \lambda x.\mathrm{S}\ x \wedge b \in [\![B]\!] \wedge a \in X\ t'\} \end{array}$$

Note that this contains the reduction to restricted indexed inductive definitions.

Universes

A universe closed under $\mathbb{N}$ and Π is defined as follows:

$$
\begin{array}{l}
\mathrm{mutual} \\
\quad \mathrm{data}\ \mathrm{U} : \mathrm{Set}\ \mathrm{where} \\
\qquad \widehat{\mathbb{N}} \quad : \quad \mathrm{U} \\
\qquad \widehat{\Pi} \quad : \quad (a : \mathrm{U}) \to (b : \mathrm{T}\ a \to \mathrm{U}) \to \mathrm{U} \\
\\
\quad \mathrm{T} : \mathrm{U} \to \mathrm{Set} \\
\quad \mathrm{T}\ \widehat{\mathbb{N}} \qquad\quad = \quad \mathbb{N} \\
\quad \mathrm{T}\ (\widehat{\Pi}\ a\ b) \quad = \quad (x : \mathrm{T}\ a) \to \mathrm{T}\ (b\ x)
\end{array}
$$

Observations

- U, T are defined simultaneously.
- Intuition is that whenever a new element $u : \mathrm{U}$ is introduced, then $\mathrm{T}\ u$ is defined recursively.
- So when we make via an inductive argument use of $a : \mathrm{U}$, we know what $\mathrm{T}\ x$ is.
- Later arguments can now depend on inductive arguments via T.
  - E.g. in
    $$\widehat{\Pi} : (a : \mathrm{U}) \to (b : \mathrm{T}\ a \to \mathrm{U}) \to \mathrm{U}$$
    the type of the second argument b depends on $\mathrm{T}\ a$.

Generalisation

- Definition of $\mathrm{U} : \mathrm{Set}$ and $\mathrm{T} : \mathrm{U} \to D$ for an arbitrary type D.
- Precise formulation introduces:
  - A data type OP_D of inductive-recursive definitions.
  - For $\gamma : \mathrm{OP}_D$ operations

$$\begin{array}{lcl} \mathrm{Arg}^{\mathrm{U}}_{\gamma} & : & (U : \mathrm{Set}) \to (T : U \to D) \to \mathrm{Set} \\ \mathrm{Arg}^{\mathrm{T}}_{\gamma} & : & (U : \mathrm{Set}) \to (T : U \to D) \to \mathrm{Arg}^{\mathrm{U}}_{\gamma}\ U\ T \to D \end{array}$$

  - $(\mathrm{Arg}^{\mathrm{U}}_{\gamma}, \mathrm{Arg}^{\mathrm{T}}_{\gamma})$ form an endofunctor

$$\mathrm{Arg}_{\gamma} : \mathrm{Fam}\ D \to \mathrm{Fam}\ D$$

    where

$$\mathrm{Fam}\ D = (X : \mathrm{Set}) \times (X \to D)$$

  - So OP_D is a universe of functors on $\mathrm{Fam}\ D$.
  - $\langle \mathrm{U}_{\gamma}, \mathrm{T}_{\gamma} \rangle$ is the least fixed point of Arg_{γ}.

Model

The cpo is

- $\mathrm{Fam}(D) = (X \in [\![\mathrm{Set}]\!]) \times (X \to [\![D]\!])$
- $\langle X, Y\rangle \leq \langle X', Y'\rangle :\Leftrightarrow X \subseteq X' \wedge Y' \restriction X = Y$.

The operator is

$$
\begin{array}{l}
\Gamma \in \mathrm{Fam}(D) \to \mathrm{Fam}(D) \\
\Gamma\ \langle U, T\rangle = \langle \{\widehat{\mathbb{N}}\} \\
\qquad\qquad\qquad \cup \{\widehat{\Pi}\ a\ b \mid a \in U \wedge b \in T(a)\ [\![\to]\!]\ U\}, \\
\qquad\qquad\qquad \begin{array}{lcl} \widehat{\mathbb{N}} & \mapsto & [\![\mathbb{N}]\!] \\ \widehat{\Pi}\ a\ b & \mapsto & (x : T\ a)\ [\![\to]\!]\ T\ (b\ x)\rangle \end{array}
\end{array}
$$

Inductive-Inductive Definitions

- Universes can be used to develop models of type theory.
- Inductive-inductive definitions were used for formulating the calculus of type theory and the derivable sets and their elements inside type theory.
- A consistency proof is obtained by
  - formulating the calculus as an inductive-inductive definition,
  - formulating a model as an inductive-recursive definitions,
  - showing that $a : \bot$ is not derivable since in the model $[\![\bot]\!] = \emptyset$

Inductive-Inductive Definitions

$$
\begin{array}{lcl}
\mathrm{DerivableContex} & : & \mathrm{Set} \\
\mathrm{DerivableSet} & : & \mathrm{DerivableContex} \to \mathrm{Set} \\
\mathrm{DerivableTerm} & : & (\Gamma : \mathrm{DerivableContex}) \to \mathrm{DerivableSet}\ \Gamma \to \mathrm{Set}
\end{array}
$$

- Here $\mathrm{DerivableContex}$, $\mathrm{DerivableSet}$ and $\mathrm{DerivableTerm}$ are defined simultaneously inductively.
- $\mathrm{DerivableSet}\ \Gamma$ is not fixed once Γ is introduced, but might grow.

Formulating Type Theory inside Type Theory

The rules for deriving Contexts are

$$\emptyset : \mathrm{Context} \qquad \frac{\Gamma : \mathrm{Context} \qquad \Gamma \Rightarrow A : \mathrm{Set}}{(\Gamma :: A) : \mathrm{Context}}$$

This is mapped to rules:

$$\begin{array}{lll} \emptyset & : & \mathrm{DerivableContex} \\ \_ :: \_ & : & (\Gamma : \mathrm{DerivableContex}) \to (A : \mathrm{DerivableSet}\ \Gamma) \to \mathrm{DerivableContex} \end{array}$$

Formulating Closure under Π

The rule for closure under $(x : A) \to B$ is:

$$\frac{\Gamma \Rightarrow A : \mathrm{Set} \qquad \Gamma, x : A \Rightarrow B : \mathrm{Set}}{\Gamma \Rightarrow (x : A) \to B : \mathrm{Set}}$$

This is mapped to the following rules:

$$\begin{array}{lll} \Pi & : & (\Gamma : \mathrm{DerivableContex}) \\ & & \to (A : \mathrm{DerivableSet}\ \Gamma) \\ & & \to (B : \mathrm{DerivableSet}\ (A :: \Gamma)) \\ & & \to \mathrm{DerivableSet}\ \Gamma \end{array}$$

Observations

- $\mathrm{DerivableContex} : \mathrm{Set}$ and $\mathrm{DerivableSet} : \mathrm{DerivableContex} \to \mathrm{Set}$ are defined **simultaneously inductively**.
- $\_ :: \_$ constructs an element of $\mathrm{DerivableContex}$ using an element of $\mathrm{DerivableSet}\ \Gamma$.
- Π constructs an element of $\mathrm{DerivableSet}\ \Gamma$ by referring to $\mathrm{DerivableSet}\ (\Gamma :: A)$ where $(\Gamma :: A)$ is a constructed element of $\mathrm{DerivableContex}$.
- So the definition of these two sets cannot be separated.
- More details in talk by Frederik Forsberg.

Model

A first approximation of the model uses the cpo

- $\mathrm{Fam}([\![\,\mathrm{Set}\,]\!]) = (X \in [\![\,\mathrm{Set}\,]\!]) \times (X \to [\![\,\mathrm{Set}\,]\!])$,
- $\langle X, Y \rangle \leq \langle X', Y' \rangle :\Leftrightarrow X \subseteq X' \wedge \forall x \in X . Y(x) \subseteq Y'(x)$.

However, it is easier to replace

$$X \to [\![\,\mathrm{Set}\,]\!]$$

by the fibration

$$(Y : [\![\,\mathrm{Set}\,]\!]) \times (Y \to X)$$

Model

So we have the cpo

- $\mathrm{Fam}'(\llbracket \mathrm{Set} \rrbracket) = (X \in \llbracket \mathrm{Set} \rrbracket) \times (Y \in \llbracket \mathrm{Set} \rrbracket) \times (Y \to X)$,
- $\langle X, Y, f \rangle \leq \langle X', Y', f' \rangle :\Leftrightarrow X \subseteq X' \wedge Y \subseteq Y' \wedge f' \restriction Y = f$.

$$
\begin{aligned}
&\Gamma(\langle \mathit{DerivableContext}, \mathit{DerivableSet}, \mathit{DerivableSetIndex} \rangle) \\
&= \langle \mathit{DerivableContext} \cup \{[]\} \\
&\quad \cup \{A :: \Gamma \mid \Gamma \in \mathit{DerivableContext} \\
&\qquad\qquad \wedge A \in \mathit{DerivableSet} \\
&\qquad\qquad \wedge \mathit{DerivableSetIndex}(A) = \Gamma\}, \\
&\quad \{\Pi\ \Gamma\ A\ B \mid \Gamma \in \mathit{DerivableContext} \\
&\qquad\qquad \wedge A \in \mathit{DerivableSet} \\
&\qquad\qquad \wedge \mathit{DerivableSetIndex}\ A = \Gamma \\
&\qquad\qquad \wedge B \in \mathit{DerivableSet} \\
&\qquad\qquad \wedge \mathit{DerivableSetIndex}\ B = A :: \Gamma\}, \\
&\quad \Pi\ \Gamma\ A\ B \mapsto \Gamma \rangle
\end{aligned}
$$

The Mahlo universe

- Mahlo universe = universe V s.t. for every function $f : \mathrm{Fam}(\mathrm{V}) \to \mathrm{Fam}(\mathrm{V})$ we have
  - $\mathrm{U}_f : \mathrm{V}$ s.t.
  - U_f is a universe with embedding $\widehat{\mathrm{T}}_f : \mathrm{U}_f \to \mathrm{V}$,
  - and there exist constructors $\widehat{\mathrm{f}}$ s.t.

$$
\begin{CD}
\mathrm{Fam}(\mathrm{V}) @>{f}>> \mathrm{Fam}(\mathrm{V}) \\
@A{\mathrm{Fam}(\widehat{\mathrm{T}}_f)}AA @AA{\mathrm{Fam}(\widehat{\mathrm{T}}_f)}A \\
\mathrm{Fam}(\mathrm{U}_f) @>>{\widehat{\mathrm{f}}}> \mathrm{Fam}(\mathrm{U}_f)
\end{CD}
$$

The Mahlo Universe

- The strength of the Mahlo universe is conjectured to be slightly stronger than arbitrary inductive-recursive definitions.
- Note that V has a constructor which is negative in V.

The Mahlo Universe

- We uncurry $f : \mathrm{Fam}(\mathrm{V}) \to \mathrm{Fam}(\mathrm{V})$ and obtain

$$\begin{array}{lll} f & : & (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{V} \\ g & : & (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{S}\ (f\ a\ b) \end{array}$$

- Correspondingly, $\widehat{\mathrm{f}}$ is replaced by two constructors $\widehat{\mathrm{f}}, \widehat{\mathrm{g}}$ of type

$$\begin{array}{lll} \widehat{\mathrm{f}} & : & (a : \mathrm{U}_{f,g}) \to (b : \mathrm{S}\ (\widehat{\mathrm{T}}_{f,g}\ a) \to \mathrm{U}_{f,g}) \to \mathrm{U}_{f,g} \\ \widehat{\mathrm{g}} & : & (a : \mathrm{U}_{f,g}) \to (b : \mathrm{S}\ (\widehat{\mathrm{T}}_{f,g}\ a) \to \mathrm{U}_{f,g}) \\ & & \to \mathrm{T}\ (f\ (\widehat{\mathrm{T}}_{f,g}\ a)\ (\widehat{\mathrm{T}}_{f,g} \circ b)) \\ & & \to \mathrm{U}_{f,g} \end{array}$$

Rules for the Mahlo Universe

mutual
  data V : Set where
    $\widehat{\mathbb{N}} \;:\; \cdots$
    $\widehat{\Pi} \;:\; \cdots$
    $\widehat{\mathrm{U}} \;:\; (f : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{V})$
      $\to (g : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{S}\ (f\ a\ b) \to \mathrm{V})$
      $\to \mathrm{V}$

  $\mathrm{S} : \mathrm{V} \to \mathrm{Set}$
  $\mathrm{S}\ \widehat{\mathbb{N}} = \cdots$
  $\mathrm{S}\ (\widehat{\Pi}\ a\ b) = \cdots$
  $\mathrm{S}\ (\widehat{\mathrm{U}}_{f,g}) = \mathrm{U}_{f,g}$

Rules for the Mahlo Universe (Cont.)

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathrm{data}\ \mathrm{U}(f : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{V})} \\
\multicolumn{3}{l}{\qquad (g : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{S}\ (f\ a\ b))} \\
\multicolumn{3}{l}{\qquad : \mathrm{Set}\ \mathrm{where}} \\
\widehat{\mathbb{N}}_0 & : & \cdots \\
\widehat{\Pi}_0 & : & \cdots \\
\widehat{\mathrm{f}} & : & (a : \mathrm{U}_{f,g}) \to (b : \mathrm{S}\ (\widehat{\mathrm{T}}_{f,g}\ a) \to \mathrm{U}_{f,g}) \to \mathrm{U}_{f,g} \\
\widehat{\mathrm{g}} & : & (a : \mathrm{U}_{f,g}) \to (b : \mathrm{S}\ (\widehat{\mathrm{T}}_{f,g}\ a) \to \mathrm{U}_{f,g}) \\
 & & \to \mathrm{T}\ (f\ (\widehat{\mathrm{T}}_{f,g}\ a)\ (\widehat{\mathrm{T}}_{f,g} \circ b)) \\
 & & \to \mathrm{U}_{f,g}
\end{array}
$$

Rules for the Mahlo Universe (Cont.)

$$
\begin{aligned}
&\widehat{\mathrm{T}} : (f : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{V}) \\
&\quad \to (g : (a : \mathrm{V}) \to (b : \mathrm{S}\ a \to \mathrm{V}) \to \mathrm{S}\ (f\ a\ b)) \\
&\quad \to \mathrm{U}_{f,g} \\
&\quad \to \mathrm{V} \\
&\widehat{\mathrm{T}}_{f,g}\ \widehat{\mathbb{N}}_0 = \cdots \\
&\widehat{\mathrm{T}}_{f,g}\ (\widehat{\Pi}_0\ a\ b) = \cdots \\
&\widehat{\mathrm{T}}_{f,g}\ (\widehat{\mathrm{f}}\ a\ b) = f\ (\widehat{\mathrm{T}}_{f,g}\ a)\ (\widehat{\mathrm{T}}_{f,g} \circ b) \\
&\widehat{\mathrm{T}}_{f,g}\ (\widehat{\mathrm{g}}\ a\ b\ c) = g\ (\widehat{\mathrm{T}}_{f,g}\ a)\ (\widehat{\mathrm{T}}_{f,g} \circ b)\ c
\end{aligned}
$$

Problem of Mahlo Universe

- Mahlo universe in type theory has some impredicative character, since we define V by referring to the collection of all functions

$$f : \mathrm{Fam}(\mathrm{V}) \to \mathrm{Fam}(\mathrm{V})$$

- No elimination rule allowed for the Mahlo universe.

In explicit mathematics a predicative construction of the Mahlo universe is possible.

Explicit Mathematics

- Explicit mathematics alternative framework for constructive mathematics.
- Based on untyped λ-calculus.
- In explicit mathematics access to the collection of arbitrary terms possible.
- Simplification: In explicit mathematics we can encode $\mathrm{Fam}(\mathrm{Set})$ into Fam, and therefore consider Mahlo universes M closed under $f : \mathrm{M} \to \mathrm{M}$ rather than $f : \mathrm{Fam}(\mathrm{M}) \to \mathrm{Fam}(\mathrm{M})$.

Idea

- Instead of adding to our Mahlo universe M a code $\widehat{\mathrm{U}}_f$ for total $f : \mathrm{M} \to \mathrm{M}$, add to M a code $\widehat{\mathrm{U}}_f$ whenever we can form $\widehat{\mathrm{U}}_f$ inside of M.
- We will consider here universes not only closed under f but containing as well an element a.
- We use here the name $\mathrm{sub}\ a\ f$ instead of $\widehat{\mathrm{U}}_f$.

pre(a, f, v)

For all a, f (no restriction) and every set-indexed family of sets v axiomatize that $\mathrm{pre}(a, f, v)$,

- is a set such that all elements are sets,
- closed under universe constructions **relative to v**, e.g.:

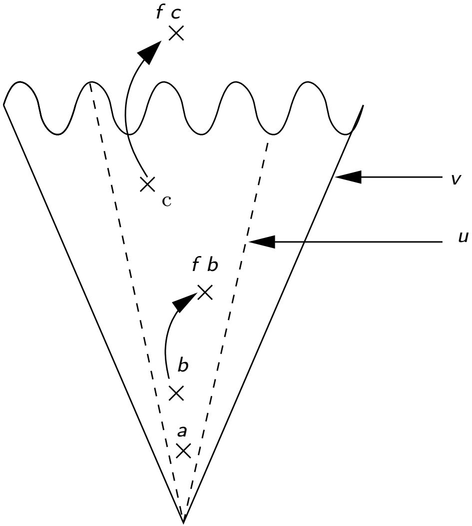
$$(x \in \mathrm{pre}(a, f, v) \land y \in (x \to \mathrm{pre}(a, f, v)) \land \boldsymbol{\Sigma(x, y) \in v}) \\ \to \Sigma(x, y) \in \mathrm{pre}(a, f, v)$$

- closed under a, f relative to v, i.e.:

$$(\mathbf{a \in v} \to a \in \mathrm{pre}(a, f, v)) \\ (x \in \mathrm{pre}(a, f, v) \land \mathbf{f\, x \in v} \\ \to f\, x \in \mathrm{pre}(a, f, v)$$

- $\mathrm{pre}(a, f, v)$ is the least such set (expressed as an induction principle).

pre(a, f, v)



Indep(a,f,u,v)

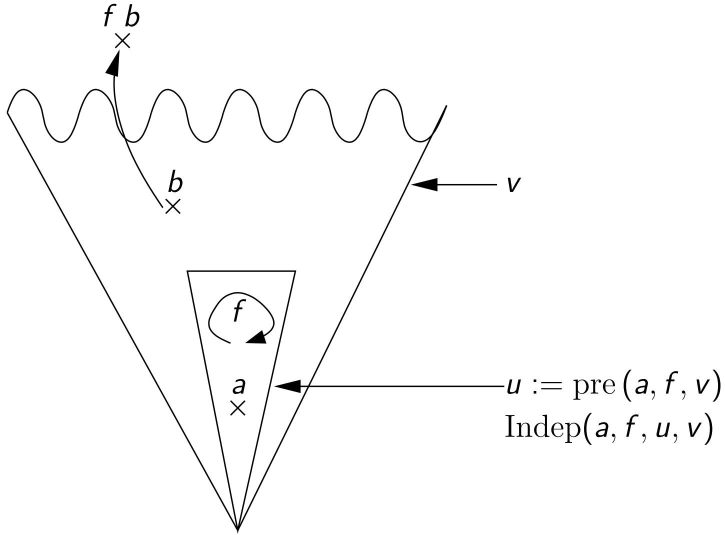
We introduce a predicate $\mathrm{Indep}(a, f, u, v)$ expressing that u is independent of v relative to a, f:
If $u = \mathrm{pre}(a, f, v)$, then the premise for adding an element to it is already fulfilled:

$$\begin{aligned}
&\mathrm{Indep}(a, f, u, v) :\Leftrightarrow \\
&\quad (\forall x \in u.\forall y \in (a \to u).\Sigma(x, y) \in v) \\
&\quad \wedge \cdots \text{ (other universe operators)} \cdots \\
&\quad \wedge a \in v \\
&\quad \wedge (\forall x \in u \to f\, x \in v)
\end{aligned}$$

We have

- $\mathrm{pre}(a, f, v)$ monotone in v.
- $\mathrm{Indep}(a, f, \mathrm{pre}(a, f, v), v) \wedge v \subseteq v'$
  $\to \mathrm{Indep}(a, f, \mathrm{pre}(a, f, v), v') \wedge \mathrm{pre}(a, f, v) =_{\mathrm{ext}} \mathrm{pre}(a, f, v')$.

Indep(a, f, pre(a, f, v), u)



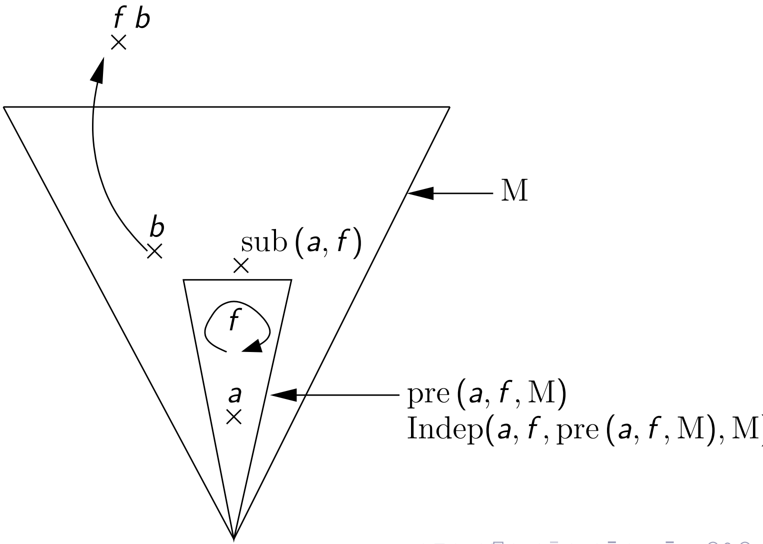
M

Axiomatize:

- M is a set such that all elements are sets,
- closed under universe operations,
- s.t.:
  If $\mathrm{Indep}(a, f, \mathrm{pre}(a, f, \mathrm{M}), \mathrm{M})$ then
  - $\mathrm{sub}(a, f)$ is a set,
  - $\mathrm{sub}(a, f) =_{\mathrm{ext}} \mathrm{pre}(a, f, \mathrm{M})$,
  - $\mathrm{sub}(a, f) \in \mathrm{M}$,
- M is the least such set.

(**Elimination rules for M!**).

M



Interpretation of the Direct Variant of the Mahlo Universe

Assume

$$f : \mathrm{M} \to \mathrm{M}$$

We have

- $\mathrm{pre}(a, f, \mathrm{M}) \subseteq \mathrm{M}$. (Trivial ind. over $\mathrm{pre}(a, f, \mathrm{M})$).
- M is a universe.

Therefore $\mathrm{Indep}(a, f, \mathrm{pre}(a, f, \mathrm{M}), \mathrm{M})$.
$\mathrm{sub}(a, f) \in \mathrm{M}$.