Universes for Inductive and Inductive-Recursive Definitions in Martin-Löf Type Theory

Peter Dybjer
Chalmers Tekniska Högskola

Stockholm-Uppsala Logic Seminar
3 September 2003

References

- A finite axiomatization of inductive-recursive definitions (with Anton Setzer). Pages 129 - 146 in Proceedings of TLCA 1999, LNCS 1581.

- Induction-recursion and initial algebras (with Anton Setzer), 2001. To appear in Annals of Pure and Applied Logic.

- Indexed induction-recursion (with Anton Setzer). Pages 93-113 in Proof Theory in Computer Science International Seminar, Dagstuhl Castle, Germany, October 7-12, 2001, LNCS 2183.

- Universes for generic programs and proofs in dependent type theory (with Marcin Benke and Patrik Jansson), 2003. Accepted for publication in Nordic Journal of Computing.

Inductive definitions – examples

- the rules for generating well-formed formulas of a logic
- the axioms and inference rules generating theorems of the logic
- the productions of a context-free grammar
- the computation rules for a programming language
- the typing rules for a programming language
- the natural numbers generated by 0 and successor

Recursive datatypes

- lists generated by Nil and Cons
- binary trees generated by $\mathrm{EmptyTree}$ and MkTree
- algebraic types in general: parameterized, many sorted term algebras
- infinitely branching trees; Brouwer ordinals; etc.
- inductive dependent types (vectors of a certain length, trees of a certain height, balanced trees, etc)
- inductive-recursive defintions (freshlists, etc)

Recursive types in functional languages

Note that recursive types in functional languages (ML, Haskell) include reflexive datatypes and nested datatypes which have more complex semantics.

Inductive definitions and foundations

Classically, inductive definitions are understood as least fixed points of monotone operators (the Knaster-Tarski theorem).

P. Aczel (An introduction to inductive definitions, Handbook of Mathematical Logic, 1976, pp 779 and 780.):

> An alternative approach is to take induction as a primitive notion, not needing justification in terms of other methods. ... It would be interesting to formulate a coherent conceptual framework that made induction the principal notion.

Inductive definitions and the notion of set in Martin-Löf type theory

Martin-Löf type theory is such a coherent conceptual framework. Sets are inductively defined: "to know a set is to know how its elements are *formed*" this is to know its introduction rules, i e, the rules for inductively generating its members.

Martin-Löf type theory and inductive definitions

- Basic set formers: $\Pi, \Sigma, +, \mathrm{I}, \mathrm{N}, \mathrm{N}_n, \mathrm{W}, \mathrm{U}_n$
- Adding new set formers with their rules when there is a need for them: lists, binary trees, the well-founded part of a relation, ...
- Exactly what is a good inductive definition? Schemata for inductive definitions, indexed inductive definitions, inductive-recursive definitions
- Universes for inductive definitions, indexed inductive definitions, inductive-recursive definitions

Plan

1. A simple case: a universe for one-sorted term algebras
2. Generic programming
3. A universe for generalized inductive definitions
4. Universes for indexed inductive definitions: general and restricted versions
5. Universes for inductive-recursive definitions; induction-recursions and algebras in slice categories
6. Embeddings and equivalences between various theories

One-sorted term algebras

A one-sorted signature is a finite list of natural numbers, representing the arities of the operations of the signature. Examples are

- the empty type with $\Sigma = [\,]$
- the natural numbers with $\Sigma = [0, 1]$,
- the Booleans with $\Sigma = [0, 0]$,
- lists of Booleans with $\Sigma = [0, 1, 1]$,
- binary trees without information in the nodes with $\Sigma = [0, 2]$

Signature = universe for one-sorted term algebras

We introduce the type of signatures

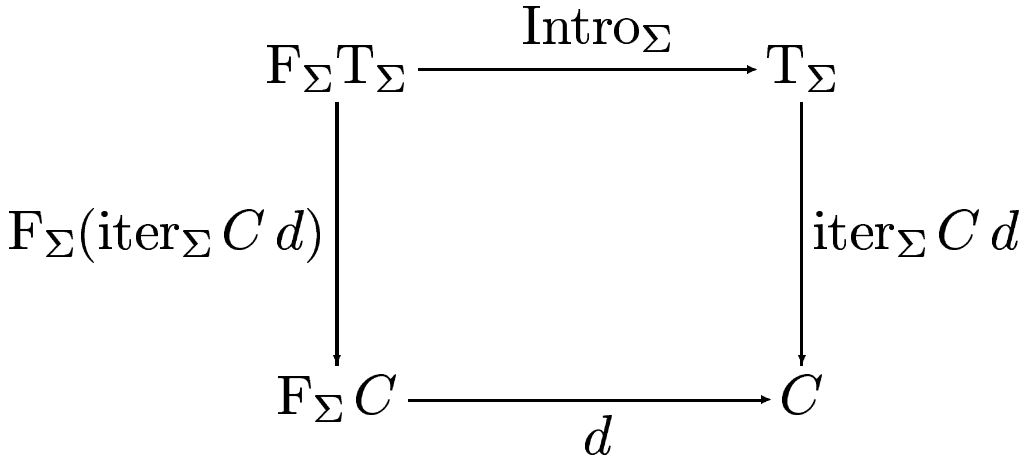
$$\mathrm{Sig} : \mathrm{Type}$$

and the decoding function

$$\mathrm{T} : \mathrm{Sig} \to \mathrm{Set}$$

which maps a signature Σ to (the carrier of) its term algebra $\mathrm{T}_\Sigma : \mathrm{Set}$.

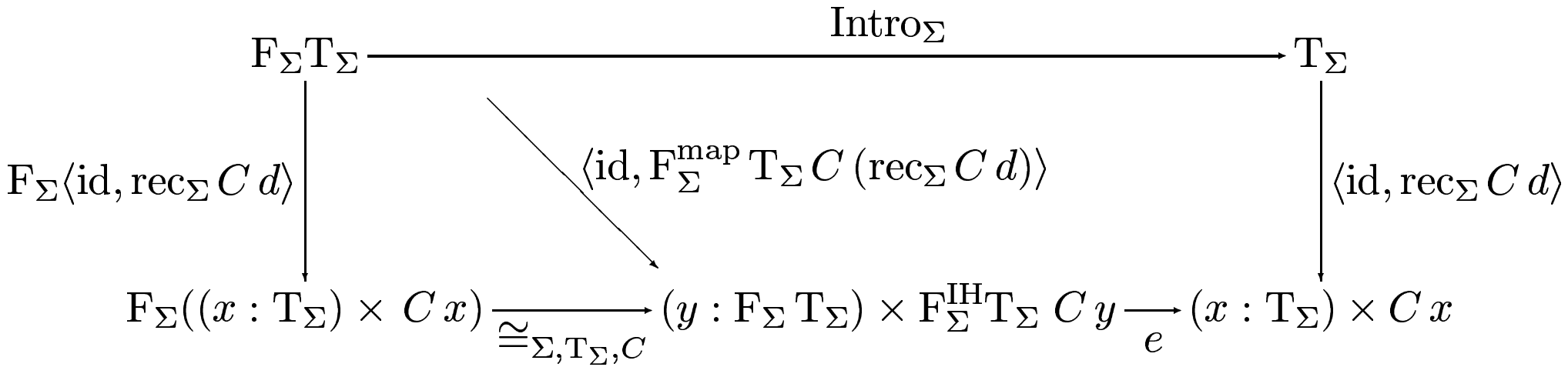
The initial Σ-algebra diagram



Pattern functor:

$$\mathrm{F}_{[n_1,\ldots,n_m]}\, X \quad = \quad X^{n_1} + \cdots + X^{n_m}$$

A diagram for T_Σ-elimination



$$e\,(y, z) \quad = \quad (\mathrm{Intro}_\Sigma\, y, d\, y\, z)$$

Notation

Dependent function types are written

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

Dependent product types are written

$$(x : A) \times B$$

Auxiliary constructions

$$\mathrm{F}_{\Sigma}^{\mathrm{IH}} : (X : \mathrm{Set}) \to (X \to \mathrm{Set}) \to \mathrm{F}_{\Sigma}\, X \to \mathrm{Set}$$

$$\mathrm{F}_{[n_1,\ldots,n_m]}^{\mathrm{IH}}\, X\, C\, (\mathrm{In}_i\, (x_1, \ldots, x_{n_i})) = C\, x_1 \times \cdots \times C\, x_{n_i}$$

and

$$\begin{aligned} \mathrm{F}_{\Sigma}^{\mathrm{map}} \quad : \quad & (X : \mathrm{Set}) \to (C : X \to \mathrm{Set}) \\ & \to ((x : X) \to C\, x) \to (y : \mathrm{F}_{\Sigma}\, X) \to \mathrm{F}_{\Sigma}^{\mathrm{IH}}\, X\, C\, y \end{aligned}$$

$$\mathrm{F}_{[n_1,\ldots,n_m]}^{\mathrm{map}}\, X\, C\, h\, (\mathrm{In}_i\, (x_1, \ldots, x_{n_i})) = (h\, x_1, \ldots, h\, x_{n_i})$$

Generic rules for T_Σ

$$
\begin{aligned}
\mathrm{T}_\Sigma &: \mathrm{Set} \\
\mathrm{Intro}_\Sigma &: \mathrm{F}_\Sigma \mathrm{T}_\Sigma \to \mathrm{T}_\Sigma \\
\mathrm{rec}_\Sigma &: (C : \mathrm{T}_\Sigma \to \mathrm{Set}) \to ((y : \mathrm{F}_\Sigma\, \mathrm{T}_\Sigma) \to \mathrm{F}_\Sigma^{\mathrm{IH}}\, C\, y \to C\, (\mathrm{Intro}_\Sigma\, y)) \\
&\quad \to (x : \mathrm{T}_\Sigma) \to C\, x
\end{aligned}
$$

Equality rule

$$\mathrm{rec}_\Sigma\, C\, d\, (\mathrm{Intro}_\Sigma\, y) = d\, y\, (\mathrm{F}_\Sigma^{\mathrm{map}}\, C\, (\mathrm{rec}_\Sigma\, C\, d)\, y)$$

Large elimination

We may add a large version of this elimination too, where C can be an arbitrary family of types, that is,

$$C[x]\ \mathrm{Type} \quad (x : \mathrm{T}_\Sigma)$$

not just a family of sets.

Martin-Löf type theory with one-sorted term algebras

Start with logical framework (including at least dependent function and product types, and $\mathbf{0}, \mathbf{1}, \mathbf{2}$). Add

arities formation, introduction, (and elimination and equality) rules for N

signatures formation, introduction (and elimination and equality) rules for Sig, i.e., for lists of natural numbers

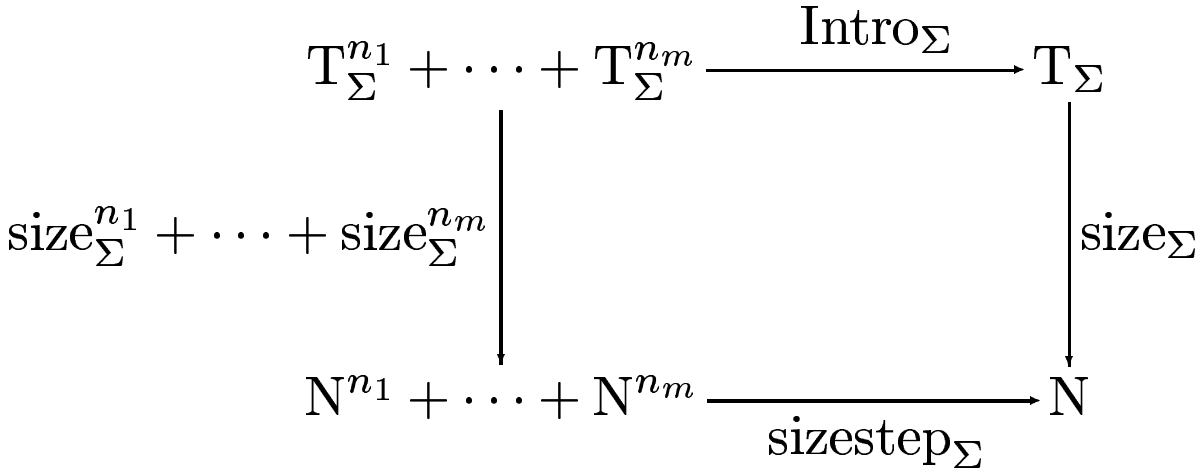
pattern functors defining rules for object and arrow parts of the functor F_Σ, and the auxiliary $\mathrm{F}_\Sigma^{\mathrm{IH}}$ and $\mathrm{F}_\Sigma^{\mathrm{map}}$

term algebras formation, introduction, elimination, and equality rules for T_Σ with constants Intro_Σ and rec_Σ.

Generic programming

A generic size function.

A special case of the initial algebra diagram. Let $\Sigma = [n_1, \ldots, n_m]$.



Generic recursion step

The recusion step is defined by induction on the signature:

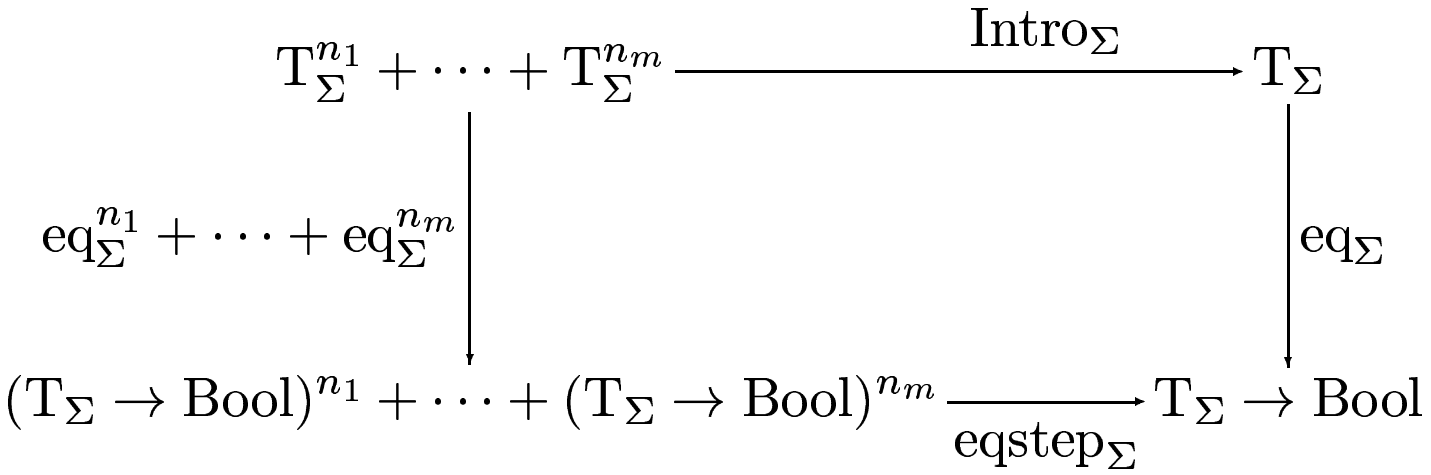
$$\begin{aligned} \mathrm{sizestep}_{n::\Sigma}\ (\mathrm{Inl}\, xs) &= 1 + \mathrm{sum}_n\, xs \\ \mathrm{sizestep}_{n::\Sigma}\ (\mathrm{Inr}\, y) &= \mathrm{sizestep}_\Sigma\ y \end{aligned}$$

where

$$\mathrm{sum} : (n : \mathrm{N}) \to \mathrm{N}^n \to \mathrm{N}$$

is a function summing the elements of a vector of natural numbers.

Generic equality



Generic recursion step

$$\mathrm{eqstep}_\Sigma\,(\mathrm{In}_i\,(p_1,\ldots,p_{n_i}))\,(\mathrm{Intro}\,(\mathrm{In}_i\,(y_1,\ldots,y_{n_i}))) = p_1\,y_1 \wedge \cdots \wedge p_{n_i}y_{n_i}$$

and for $i \neq j$

$$\mathrm{eqstep}_\Sigma\,(\mathrm{In}_i\,(p_1,\ldots,p_{n_i}))\,(\mathrm{Intro}\,(\mathrm{In}_j\,(y_1,\ldots,y_{n_j}))) = \texttt{False}$$

can be defined by induction on the signature.

There are also generic proofs of reflexivity and substitutivity of equality.

Generalizing the notion of a signature

iterated inductive definitions

$$\mathrm{Cons} : \mathrm{N} \to \mathrm{ListN} \to \mathrm{ListN}$$

the first argument is a side-condition.

generalized inductive definitions

$$\mathrm{Sup} : (\mathrm{N} \to \mathcal{O}) \to \mathcal{O}$$

constructors with dependent types

$$\mathrm{Sup} : (x : A) \to (B\,x \to \mathrm{W}) \to \mathrm{W}$$

Parameterized inductive definitions

are important for generic programming, but not for formalizing inductive definitions in Martin-Löf type theory, where parameterization is taken care of by the logical framework.

Signatures for generalized inductive definitions

$$
\begin{array}{lcl}
\epsilon & : & \mathrm{Sig} \\
\sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}) \to \mathrm{Sig} \\
\rho & : & \mathrm{Set} \to \mathrm{Sig} \to \mathrm{Sig}
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{F}_{\epsilon}\, X & = & \mathbf{1} \\
\mathrm{F}_{\sigma\, A\, \Sigma}\, X & = & (x : A) \times \mathrm{F}_{\Sigma\, x}\, X \\
\mathrm{F}_{\rho\, A\, \Sigma}\, X & = & (A \to X) \times \mathrm{F}_{\Sigma}\, X
\end{array}
$$

Indexed inductive definitions; two versions

general à la "Inductive Families" (PD) and "Calculus of Inductive Constructions" (Coquand and Paulin); based on a Curry-Howard interpretation of Martin-Löf's "Intuitionistic Theory of Iterated Inductive Definitions"

restricted à la Half and the original implementation of Agda (Coquand); Haskell-like syntax and simplified pattern matching with dependent types; also simpler set-theoretic (least fixed point of monotone operator) and category-theoretic (initial algebra) semantics.

Two indexed inductive definitions of Even

General:

$$\begin{aligned} \mathrm{C0} &: \mathrm{Even}\,0 \\ \mathrm{C1} &: (m : \mathrm{N}) \to \mathrm{Even}\,m \to \mathrm{Even}\,(\mathrm{Succ}\,(\mathrm{Succ}\,m)) \end{aligned}$$

Restricted:

$$\begin{aligned} \mathrm{C0} &: (n : \mathrm{N}) \to (n = 0) \to \mathrm{Even}\,n \\ \mathrm{C1} &: (n : \mathrm{N}) \to (m : \mathrm{N}) \to (n = \mathrm{Succ}\,(\mathrm{Succ}\,m)) \to \mathrm{Even}\,m \to \mathrm{Even}\,n \end{aligned}$$

... in Agda syntax:

$$\mathbf{data}\ \mathrm{Even}\,n = \mathrm{C0}\,(n = 0) \mid \mathrm{C1}\,(m : \mathrm{N})\,(n = \mathrm{Succ}\,(\mathrm{Succ}\,m))\,(\mathrm{Even}\,m)$$

Inductively defined equality

Only makes sense as general indexed inductive defintion:

$$\mathrm{Refl}_A : (a : A) \to \mathrm{I}\, A\, a\, a$$

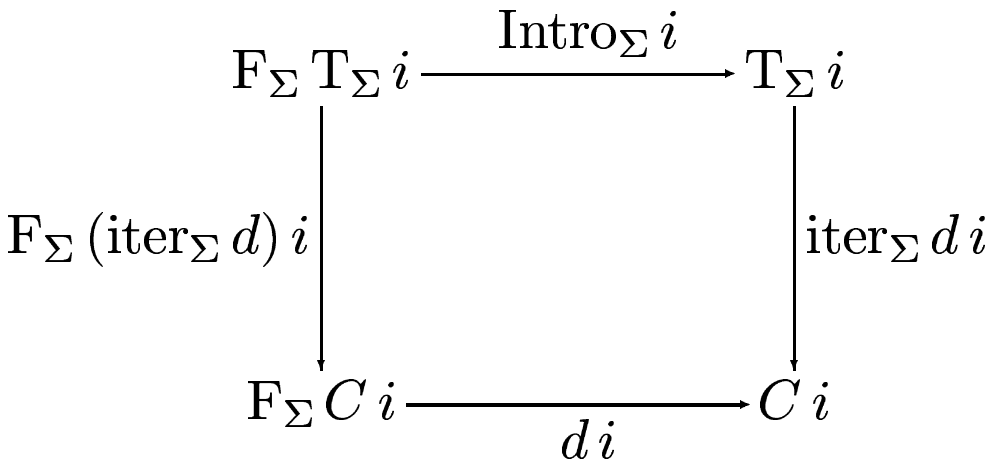
Restricted ...

$$\mathrm{Refl}_A : (a, b : A) \to (a =_A b) \to \mathrm{I}\, A\, a\, b$$

Agda's original approach is similar to Martin-Löf 1972: no general equality proposition; equality has to be defined for each set separately. Eg equality of natural numbers is defined by N-elimination.

The I-indexed initial algebra diagram

A diagram in the category of I-indexed families of sets:



A diagram for the elimination rule – as for the non-indexed case.

Signatures for restricted I-indexed inductive definitions

$$
\begin{array}{lcl}
\epsilon & : & \mathrm{Sig}\, I \\
\sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}\, I) \to \mathrm{Sig}\, I \\
\rho & : & (A : \mathrm{Set}) \to (A \to I) \to \mathrm{Sig}\, I \to \mathrm{Sig}\, I
\end{array}
$$

Pattern functor for restricted I-indexed inductive definitions

$$\mathrm{F}_\Sigma\, X\, i \quad = \quad \mathrm{G}_{\Sigma\, i}\, X$$

where

$$\begin{aligned}
\mathrm{G}_\epsilon\, X &= \mathbf{1} \\
\mathrm{G}_{\sigma\, A\, \Sigma}\, X &= (x : A) \times \mathrm{G}_{\Sigma\, x}\, X \\
\mathrm{G}_{\rho\, A\, \iota\, \Sigma}\, X &= ((x : A) \to X\, (\iota\, x)) \times \mathrm{G}_\Sigma\, X
\end{aligned}$$

Generic rules for restricted I-indexed inductive definitions

$$
\begin{aligned}
\mathrm{T}_\Sigma &: I \to \mathrm{Set} \\
\mathrm{Intro}_\Sigma &: (i : I) \to \mathrm{G}_{\Sigma\, i}\, \mathrm{T}_\Sigma \to \mathrm{T}_\Sigma\, i \\
\mathrm{rec}_\Sigma &: (C : (i : I) \to \mathrm{T}_\Sigma\, i \to \mathrm{Set}) \\
&\quad \to ((i : I) \to (y : \mathrm{G}_{\Sigma\, i}\, \mathrm{T}_\Sigma) \to \mathrm{G}^{\mathrm{IH}}_{\Sigma\, i}\, \mathrm{T}_\Sigma\, C\, y \to C\, (\iota_\Sigma\, \mathrm{T}_\Sigma\, y)\, (\mathrm{Intro}_\Sigma\, i\, y)) \\
&\quad \to (i : I) \to (x : \mathrm{T}_\Sigma\, i) \to C\, i\, x
\end{aligned}
$$

Equality rule

$$\mathrm{rec}_\Sigma\, C\, d\, i\, (\mathrm{Intro}_\Sigma\, i\, y) = d\, i\, y\, (\mathrm{G}^{\mathrm{map}}_{\Sigma\, i}\, \mathrm{T}_\Sigma\, C\, (\mathrm{rec}_\Sigma\, C\, d)\, y)$$

Example: signature for accessibility

$$\begin{aligned} \mathrm{Acc} &: I \to \mathrm{Set} \\ \mathrm{AccIntro} &: (i : I) \to ((j : I) \to (j < i) \to \mathrm{Acc}\, j) \to \mathrm{Acc}\, i \end{aligned}$$

Signature (arity). Note that we get an uncurried version of the premise.

$$\begin{aligned} \Sigma\, i &= \rho\, ((j : I) \times (j < i))\, \mathrm{fst}\, \epsilon \\ \mathrm{AccIntro} &: (i : I) \to ((k : (j : I) \times (j < i)) \to \mathrm{Acc}\, (\mathrm{fst}\, k)) \to \mathrm{Acc}\, i \end{aligned}$$

Example: signature for even numbers

Write $\sigma x : A.\, \Sigma$ for $\sigma\, A\, ((x)\Sigma)$.

$$
\begin{aligned}
\Sigma\, n &= \sigma i : \mathbf{2}.\, \alpha_i\, n \\
\alpha_0\, n &= \sigma p : (n = 0).\, \epsilon \\
\alpha_1\, n &= \sigma m : \mathrm{N}.\, \sigma p : (n = \mathrm{Succ}\, (\mathrm{Succ}\, m)).\, \rho\, \mathbf{1}\, ((x)m)\, \epsilon
\end{aligned}
$$

Generic rules for general I-indexed inductive definitions

$$
\begin{aligned}
\mathrm{T}_\Sigma &: I \to \mathrm{Set} \\
\mathrm{Intro}_\Sigma &: (y : \mathrm{G}_\Sigma\,\mathrm{T}_\Sigma) \to \mathrm{T}_\Sigma\,(\iota_\Sigma\,\mathrm{T}_\Sigma\,y) \\
\mathrm{rec}_\Sigma &: (C : (i : I) \to \mathrm{T}_\Sigma\,i \to \mathrm{Set}) \\
&\quad \to ((y : \mathrm{G}_\Sigma\,\mathrm{T}_\Sigma) \to \mathrm{G}_\Sigma^{\mathrm{IH}}\,\mathrm{T}_\Sigma\,C\,y \to C\,(\iota_\Sigma\,\mathrm{T}_\Sigma\,y)\,(\mathrm{Intro}_\Sigma\,y)) \\
&\quad \to (i : I) \to (x : \mathrm{T}_\Sigma\,i) \to C\,i\,x
\end{aligned}
$$

Equality rule

$$\mathrm{rec}_\Sigma\,C\,d\,(\iota_\Sigma\,\mathrm{T}_\Sigma\,y)\,(\mathrm{Intro}_\Sigma\,y) = d\,y\,(\mathrm{G}_\Sigma^{\mathrm{map}}\,\mathrm{T}_\Sigma\,C\,(\mathrm{rec}_\Sigma\,C\,d)\,y)$$

Signatures for general I-indexed inductive definitions

$$
\begin{array}{lcl}
\epsilon & : & I \to \mathrm{Sig}\, I \\
\sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}\, I) \to \mathrm{Sig}\, I \\
\rho & : & (A : \mathrm{Set}) \to (A \to I) \to \mathrm{Sig}\, I \to \mathrm{Sig}\, I
\end{array}
$$

$$
\iota_{\epsilon\, i} = i
$$

Example: signature for even numbers

Write $\sigma x : A.\, \Sigma$ for $\sigma\, A\, ((x)\Sigma)$.

$$
\begin{aligned}
\Sigma &= \sigma i : \mathbf{2}.\, \alpha_i \\
\alpha_0 &= \epsilon\, 0 \\
\alpha_1 &= \sigma m : \mathrm{N}.\, \rho\, \mathbf{1}\, ((x)m)\, (\epsilon\, (\mathrm{Succ}\, (\mathrm{Succ}\, m)))
\end{aligned}
$$

Inductive-recursive definitions

Usually, we *first* define a set T_Σ, *then* we define a function

$$h = \mathrm{rec}_\Sigma\, d : \mathrm{T}_\Sigma \to C$$

In an inductive-recursive definition the function h may appear in the introduction rules for T_Σ.

How can this arise? As an example, we will see how inductive-recursive definitions arise naturally when analyzing the termination of functions defined by nested recursion (Bove and Capretta 2001).

Quicksort in Haskell

```
qSort :: [Nat] -> [Nat]

qSort [] = []
qSort (x : xs)
  = qSort (filter (< x) xs)
    ++ x : qSort (filter (>= x) xs)
```

(A more efficient version which partitions `xs` in one pass can be written.)

A termination predicate for quicksort

```
D :: [Nat] -> Set

C0 :: D []
C1 :: (x :: Nat) -> (xs :: [Nat])
      -> D (filter (< x) xs)
      -> D (filter (>= x) xs)
      -> D (x : xs)
```

A general indexed inductive definition!

Quicksort in Martin-Löf type theory

Quicksort can then be represented as a function of two arguments: a list and a proof that quicksort terminates for this list.

```
qSort :: (xs :: [Nat]) -> D xs -> [Nat]

qSort [] C0 = []
qSort (x : xs) (C1 x xs p q)
  = qSort (filter (< x) xs) p
    ++ x : qSort (filter (>= x) xs) q
```

Termination of quicksort

Quicksort terminates for all lists:

```
qSortTerm :: (xs :: [Nat]) -> D xs
```

Hence

```
\xs -> qSort xs (qSortTerm xs) :: [Nat] -> [Nat]
```

McCarthy's 91-function in Haskell

The Bove-Capretta method is applicable to nested recursion as well.

Haskell code for McCarthy's 91-function:

```
f91 :: Nat -> Nat

f91 n = if n > 100 then n - 10
                  else f91 (f91 (n + 11))
```

McCarthy's 91-function in Martin-Löf type theory

We get a restricted indexed inductive-recursive definition of the termination predicate and the structural recursive version of `f91`:

```
D   :: Nat -> Set
f91 :: (n :: Nat) -> D n -> Nat

C0 :: (n :: Nat) -> T (n > 100) -> D n
C1 :: (n :: Nat) -> T (n <= 100)
      -> (p :: D (n + 11)) -> D (f91 (n + 11) p) -> D n

f91 n (C0 n r)     = n - 10
f91 n (C1 n r p q) = f91 (f91 (n + 11) p) q
```

The first universe à la Tarski

First example of an inductive-recusive definition (Martin-Löf, 1984):

$$\begin{aligned} \mathrm{U} &: \mathrm{Set} \\ \mathrm{T} &: \mathrm{U} \to \mathrm{Set} \end{aligned}$$

A *simultaneous* inductive-recursive definition

$$\begin{aligned} \hat{\mathrm{N}} &: \mathrm{U} \\ \hat{\Pi} &: (a : \mathrm{U}) \to (b : \mathrm{T}\, a \to \mathrm{U}) \to \mathrm{U} \\ \mathrm{T}\, \hat{\mathrm{N}} &= \mathrm{N} \\ \mathrm{T}\, (\hat{\Pi}\, a\, b) &= \Pi\, (\mathrm{T}\, a)\, (\mathrm{T} \circ b) \end{aligned}$$

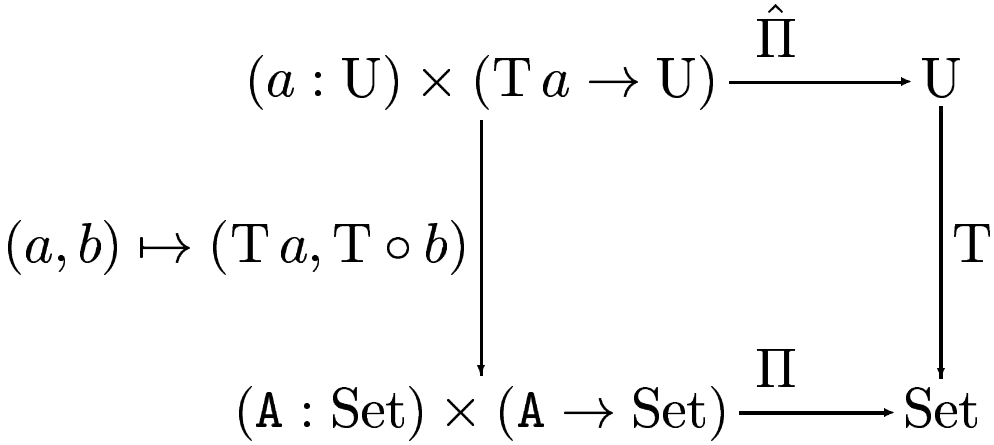
Constructive analogues of large cardinals

universe	inaccessible cardinal
superuniverse	hyperinaccessible cardinal
Mahlo universe	Mahlo cardinal

all example of inductive-recursive definitions and instances of our general formulation.

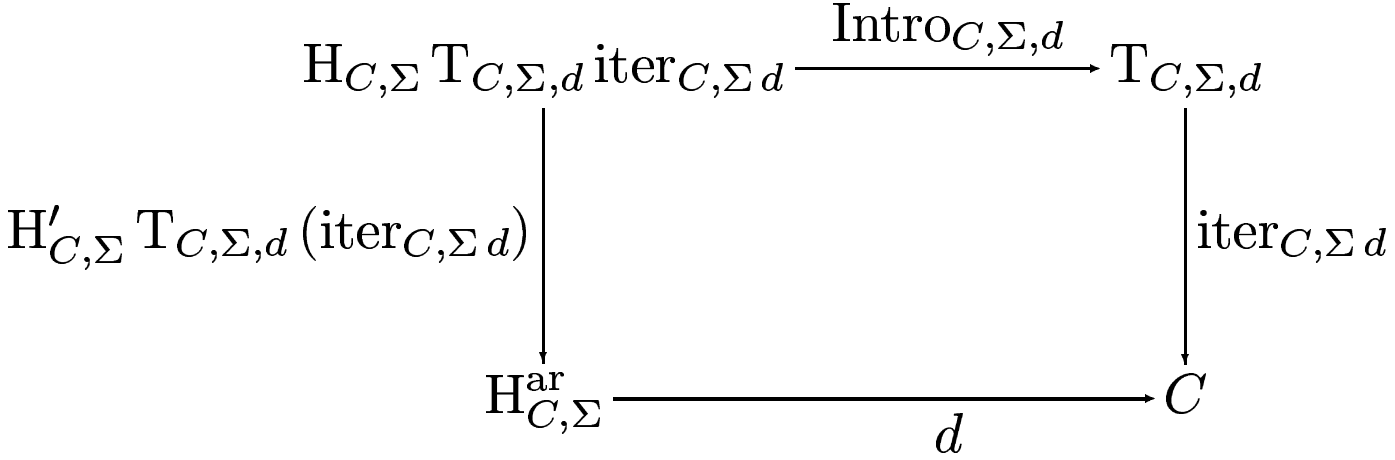
Palmgren's higher-order universes are given by an indexed inductive-recursive definition.

A diagram for U and T

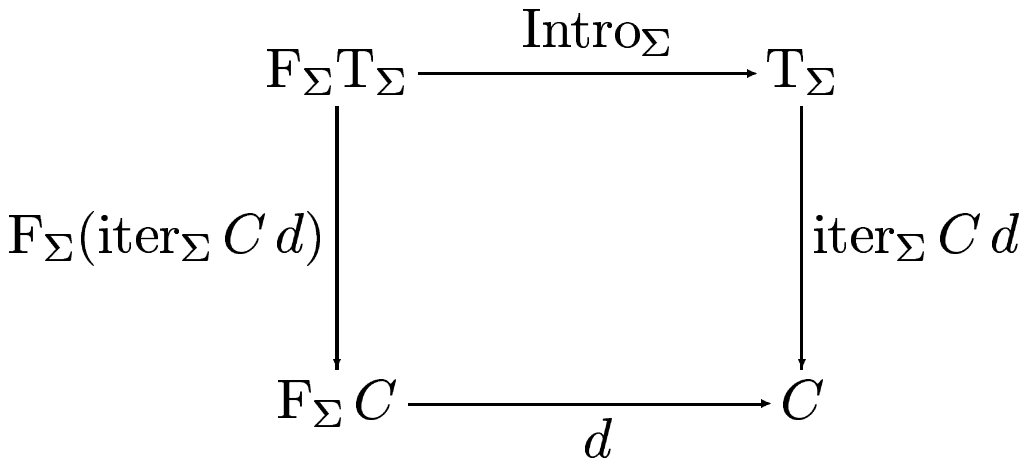


This is not an initial algebra diagram, due to the simultaneous inductive-recursive nature of U and T. But it is close to it!

Induction-recursion as a reflection principle



Inductive definitions are special cases of inductive-recursive ones, where $\mathrm{H}_{C,\Sigma}\,X\,f$ does not depend on f. In this case the above diagram degenerates to the usual initial algebra diagram.


Intro_Σ
$\mathrm{F}_\Sigma \mathrm{T}_\Sigma$
T_Σ
$\mathrm{F}_\Sigma(\mathrm{iter}_\Sigma\, C\, d)$
$\mathrm{iter}_\Sigma\, C\, d$
$\mathrm{F}_\Sigma\, C$
C
d

Signatures for inductive-recursive definitions

$$
\begin{array}{rcl}
\epsilon & : & \mathrm{Sig}_C \\
\sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}_C) \to \mathrm{Sig}_C \\
\rho & : & (A : \mathrm{Set}) \to ((A \to C) \to \mathrm{Sig}_C) \to \mathrm{Sig}_C
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{H}^{\mathrm{ar}}_{C,\epsilon} & = & \mathbf{1} \\
\mathrm{H}^{\mathrm{ar}}_{C,\sigma\, A\, \Sigma} & = & (x : A) \times \mathrm{H}^{\mathrm{ar}}_{C,\Sigma\, x} \\
\mathrm{H}^{\mathrm{ar}}_{C,\rho\, A\, \Sigma} & = & (f : A \to C) \times \mathrm{H}^{\mathrm{ar}}_{C,\Sigma\, f}
\end{array}
$$

$$
\begin{aligned}
\mathrm{H}_{\epsilon}\, X\, h &= \mathbf{1} \\
\mathrm{H}_{\sigma\, A\, \Sigma}\, X\, h &= (x : A) \times \mathrm{F}_{\Sigma\, x}\, X\, h \\
\mathrm{H}_{\rho\, A\, \Sigma}\, X\, h &= (f : A \to X) \times \mathrm{F}_{\Sigma\, (h \circ f)}
\end{aligned}
$$

Example: a signature (arity) for a universe closed under Π

$$\Sigma_{\Pi} = \rho\,\mathbf{1}\,((A)\rho\,(A\,0)\,((B)\epsilon)) : \mathrm{Sig}_{\mathrm{Set}}$$

$$\begin{aligned} \mathrm{H}^{\mathrm{ar}}_{\mathrm{Set},\Sigma_{\Pi}} &= (A : \mathbf{1} \to \mathrm{Set}) \times (((A\,0) \to \mathrm{Set}) \times \mathbf{1}) \\ \mathrm{H}_{\mathrm{Set},\Sigma_{\Pi},\Pi}\,U\,T &= (a : \mathbf{1} \to U) \times ((T\,(a\,0) \to U) \times \mathbf{1}) \end{aligned}$$

Generic rules for inductive-recursive definitions

Let C be a type, $\Sigma : \mathrm{Sig}_C$, and $d : \mathrm{H}^{\mathrm{ar}}_{C,\mathrm{Sig}} \to C$.

$$\begin{aligned} \mathrm{T}_{C,\Sigma,d} &: \mathrm{Set} \\ \mathrm{iter}_{C,\Sigma,d} &: \mathrm{T}_{C,\Sigma,d} \to C \\ \mathrm{Intro}_{C,\Sigma,d} &: \mathrm{H}_{C,\Sigma}\,\mathrm{T}_{C,\Sigma,d}\,\mathrm{iter}_{C,\Sigma\,d} \to \mathrm{T}_{C,\Sigma,d} \end{aligned}$$

See the diagram for the equality rule. There is also an elimination rule (à la universe elimination) with its own equality rule.

Set-theoretic semantics

The rules of Martin-Löf type theory (including inductive-recursive definitions) are valid under a "naive" interpretation of a constructive concept as the corresponding classical concept with the same name.

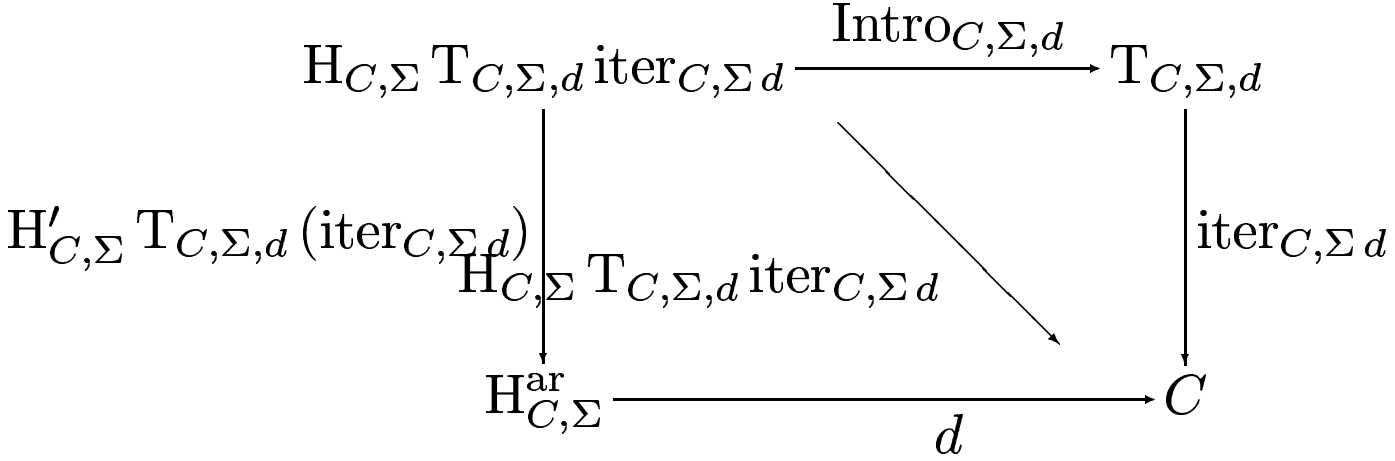
Set is interpreted as (inductively defined) set; element as element; equal elements as equal elements; function as function (graph); Π as Π; etc.

We get the semantics of (the recursive part of) an inductive-recursive definition by iterating a monotone operator to a fixed point. The only difficulty is to prove that such a fixed point exists. This can be done by using the axiom that Mahlo cardinals exist.

An inaccessible cardinal M is Mahlo if every normal function $f : \mathrm{M} \to \mathrm{M}$ has an inaccessible fixed point

Inductive-recursive definitions as initial algebras in slice categories

Draw a commuting triangle instead of a square.



This is a $\mathrm{H}_{C,\Sigma}$-algebra in the slice category $\mathbf{Type}/C$.

Reflection principle vs initial algebras

See P. Dybjer and A. Setzer "Induction-recursion and initial algebras", for details.

Theorem. Induction-recursion can be equivalently formalized as a *reflection principle* and as the existence of *initial algebras in slice categories*. The theories $\mathbf{IRD}^{\mathbf{refl}}_{\mathbf{ext}}$, $\mathbf{IRD}^{\mathbf{elim}}_{\mathbf{ext}}$, and $\mathbf{IRD}^{\mathbf{init}}_{\mathbf{ext}}$ can all be interpreted in each other.

Metatheorems

See P. Dybjer and A. Setzer "Indexed induction-recursion", 2002 (long version, submitted for publication) for details.

Theorem. *General* and *restricted* indexed inductive-recursive definitions are equivalent. The theories $\mathbf{IIRD}^{\mathbf{g}}_{\mathbf{ext}}$ and $\mathbf{IIRD}^{\mathbf{r}}_{\mathbf{ext}}$ can be interpreted in each other.

Theorem. *Indexed* inductive-recursive definitions can be interpreted as *non-indexed* inductive-recursive definitions. The theories $\mathbf{IIRD}^{\mathbf{r}}_{\mathbf{ext}}, \mathbf{IIRD}^{\mathbf{g}}_{\mathbf{ext}}$ and $\mathbf{IRD}_{\mathbf{ext}}$ can be interpreted in each other.

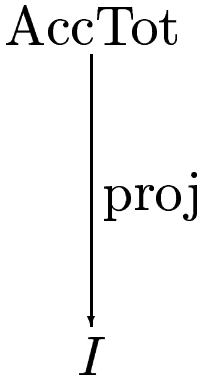
Theorem. *Small* indexed *inductive-recursive* definitions can be interpreted as indexed *inductive* definitions. The theory $\mathbf{IIRD}_{\mathbf{ext}}$ where C is a family of *sets* can be interpreted in the theory $\mathbf{IID}_{\mathbf{ext}}$.

Accessibility as a fibred set

Replace

$$\mathrm{Acc} : I \to \mathrm{Set}$$

by the fibred set



so that $\mathrm{Acc}\, i$ is represented by $(x : \mathrm{AccTot}) \times (\mathrm{proj}\, x =_I i)$.

Accessibility as an inductive-recursive definition

$$
\begin{aligned}
\mathrm{AccIntro} \quad &: \quad (i : I) \\
&\quad \to (p : (j : I) \to (j < i) \to \mathrm{AccTot}) \\
&\quad \to ((j : I) \to (j < i) \to (\mathrm{proj}\,(p\,j\,r) =_I j)) \\
&\quad \to \mathrm{AccTot} \\
\mathrm{proj}\,(\mathrm{AccIntro}\,i\,p\,q) \quad &= \quad i
\end{aligned}
$$

Cf indexed inductive definition:

$$
\mathrm{AccIntro} \quad : \quad (i : I) \to ((j : I) \to (j < i) \to \mathrm{Acc}\,j) \to \mathrm{Acc}\,i
$$