Inductive-Recursive Types

A Finite Axiomatization of Inductive-Recursive Definitions

Bas van der Linden and Remco van Os

Radboud University

5 December 2024

Internalization

Before the break:	Inductive-recursive types via an *external* schema
Now:	*Internal* schema

Internalization

Before the break: Inductive-recursive types via an *external* schema
Now: *Internal* schema

- What does that mean?
- No external concepts
- Everything is Type Theory

Internalization

Before the break:	Inductive-recursive types via an *external* schema
Now:	*Internal* schema

- What does that mean?
- No external concepts
- Everything is Type Theory
- Allows us to prove over the axiomatization
- Makes the meta theory easier (e.g. building models)

Recap: Why Induction-Recursion?

Say we want to inductively define a universe closed under $\mathbb{N}$ and Σ.

Recap: Why Induction-Recursion?

Say we want to inductively define a universe closed under $\mathbb{N}$ and Σ.

```
Inductive  U : Type :=
           | N̂ : U
           | Σ̂ : (x:U) → (x → U) → U
```

Recap: Why Induction-Recursion?

Say we want to inductively define a universe closed under $\mathbb{N}$ and Σ.

```
Inductive U : Type :=
          | N̂ : U
          | Σ̂ : (x:U) → (x→U) → U
```

Recap: Why Induction-Recursion?

So actually:

```
Inductive U : Type :=
   | N̂:U
   | Σ̂:(x:U) → (T x→U)  → U
T : U → Type :=
   | T  N̂ = ℕ
   | T ( Σ̂ a b) = Σ (x:T a) (T (b x))
```

Recap: Why Induction-Recursion?

So actually:

```
Inductive U : Type :=
   | N̂:U
   | Σ̂:(x:U) → (T x→U)  → U
T : U → Type :=
   | T  N̂ = ℕ
   | T ( Σ̂ a b) = Σ (x:T a) (T (b x)) = Σ (T a) (T b)
```

Notation

- We write $(x : A) \to \ \ldots$ for $\Pi x : A \ldots$

set, type, stype

We have:

- $set : type$
- $A : set \implies A : type$
- $A : set \implies A : stype$
- set is not of $stype$

set, type, stype

We have:

- *set* : *type*
- $A : set \implies A : type$
- $A : set \implies A : stype$
- *set* is not of *stype*

And:

- $\diamond : \mathbb{1}$
- $tt, ff : \mathbb{B}$

Inductive defintions

Generalized inductive definitions

A type can be inductively defined by a finite number of constructors:

$$intro_i : \Phi_i(U) \to U$$

Where the functor Φ_i is recursively defined as follows ($A : stype$):

- No premises: $\Phi_i(U) = \mathbb{1}$
- Non-inductive premise: $\Phi_i(U) = A \times \Psi(U)$
- Inductive premise: $\Phi_i(U) = (A \to U) \times \Psi(U)$

Given such $(\Phi_1; \ldots; \Phi_n)$, they define an inductive type

Inductive Types as Initial Algebra's

For each constructor and any D, d_i we have the following diagram

$$\begin{array}{ccc}
\Phi_i(U) & \xrightarrow{intro_i} & U \\
{\scriptstyle \Phi_i(T)} \downarrow & & \downarrow {\scriptstyle \exists! T} \\
\Phi_i(D) & \xrightarrow{d_i} & D
\end{array}$$

Moving towards Inductive-Recursive Types

We need to add more:

- Dependent types
- Single functor
- Induction-Recursion

Dependent Types

Consider the Σ type: It has one constructor

$$p : (x : A) \to (y : B\ x) \to \ \Sigma\ A\ B$$

Dependent Types

Consider the Σ type: It has one constructor

$$p : \overbrace{(x : A)}^{\text{nonind}} \to \overbrace{(y : B\ x)}^{\text{nonind}} \to \ \Sigma\ A\ B$$

Dependent Types

Consider the Σ type: It has one constructor

$$p : \overbrace{(x : A)}^{\text{nonind}} \to \overbrace{(y : B\ x)}^{\text{nonind}} \to \ \Sigma\ A\ B$$

But recall the rule for non-inductive premises

Non-inductive premise

For $A : \mathit{stype}$, Ψ a strictly positive functor: $\Phi_i(U) = A \times \Psi(U)$

Dependent Types

Consider the Σ type: It has one constructor

$$p : \overbrace{(x : A)}^{\text{nonind}} \to \overbrace{(y : B\ x)}^{\text{nonind}} \to \ \Sigma\ A\ B$$

But recall the rule for non-inductive premises

Non-inductive premise

For $A : \mathit{stype}$, Ψ a strictly positive functor: $\Phi_i(U) = A \times \Psi(U)$

We need to modify this to:

Non-inductive premise

For $A : \mathit{stype}$, Ψ_x a strictly positive functor *depending on* x:
$\Phi_i(U) = (\Pi x : A) \times \Psi_x(U)$

Single Functor

Replace the sequence of functors by a single functor

$$(\Phi_1; \ldots; \Phi_n) \Rightarrow \Phi(U) := (x : N_n) \times \Psi_x(U)$$

N_n is the finite set with n elements.
Can be constructed with the booleans $\mathbb{B}$ and the empty set N_0

Induction-Recursion

$$\hat{\Sigma} : (\mathtt{x} : \mathtt{U}) \to (\mathtt{T}\ \mathtt{x} \to \mathtt{U}) \to \mathtt{U}$$

Induction-Recursion

$$\hat{\Sigma} : \underbrace{(\mathtt{x} : \mathtt{U})}_{\text{ind}} \rightarrow \underbrace{(\mathtt{T}\ \ \mathtt{x} \rightarrow \mathtt{U})}_{\text{ind}} \rightarrow \mathtt{U}$$

Induction-Recursion

$$\hat{\Sigma} : \underbrace{(\mathtt{x} : \mathtt{U})}_{\text{ind}} \to \underbrace{(\mathtt{T}\ \mathtt{x} \to \mathtt{U})}_{\text{ind}} \to \mathtt{U}$$

The second premise depends on the first premise and T

Inductive premise

For A : *stype*, Ψ a strictly positive functor:
$\Phi(U) = (A \to U) \times \Psi(U)$

Induction-Recursion

$$\hat{\Sigma} : \underbrace{(\mathtt{x} : \mathtt{U})}_{\text{ind}} \to \underbrace{(\mathtt{T}\ \mathtt{x} \to \mathtt{U})}_{\text{ind}} \to \mathtt{U}$$

The second premise depends on the first premise and T

Inductive premise

For $A : \textit{stype}$, Ψ a strictly positive functor:
$\Phi(U) = (A \to U) \times \Psi(U)$

We need to modify this to:

Inductive premise

For $A : \textit{stype}$, Ψ_g a strictly positive functor
depending on $g : A \to D$:
$\Phi_g(U, T) = (f : A \to U) \times \Psi_{T \circ f}(U)$

Induction-Recursion

Generalized inductive-recursive definitions

A type can be inductively defined by a single functor Φ:

$$\Phi(U, T) := (x : N_n) \times \Psi_x(U, T)$$

Where the functor Φ_i is recursively defined as follows ($A : stype$):

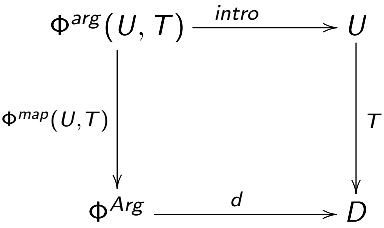
- No premises: $\Phi(U, T) = \mathbb{1}$
- Non-inductive premise: $\Phi(U, T) = (\Pi x : A) \times \Psi_x(U)$
- Inductive premise: $\Phi(U, T) = (f : A \to U) \times \Psi_{T \circ f}(U)$

Reflection Principle

What about the constructors *intro*?

Reflection Principle

What about the constructors *intro*?



Axiomatization

Type SP_D

SP_D represents the strictly positive functors.

$$\frac{D : \mathrm{type}}{\mathrm{SP}_D : \mathrm{type}} \qquad \frac{\phi : \mathrm{SP}_D \quad U : \mathrm{set} \quad T : U \to D}{\mathrm{arg}_\phi : \mathrm{stype}}$$

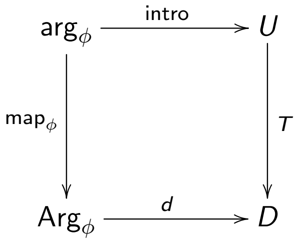
$$\frac{D : \mathrm{type} \quad \phi : \mathrm{SP}_D}{\mathrm{Arg}_\phi : \mathrm{type}} \qquad \frac{\phi : \mathrm{SP}_D \quad U : \mathrm{set} \quad T : U \to D}{\mathrm{map}_\phi : \mathrm{arg}_\phi \to \mathrm{Arg}_\phi}$$

Type SP_D

SP_D represents the strictly positive functors.

$$\frac{D : \text{type}}{\mathrm{SP}_D : \text{type}} \qquad \frac{\phi : \mathrm{SP}_D \quad U : \text{set} \quad T : U \to D}{\mathrm{arg}_\phi : \text{stype}}$$

$$\frac{D : \text{type} \quad \phi : \mathrm{SP}_D}{\mathrm{Arg}_\phi : \text{type}} \qquad \frac{\phi : \mathrm{SP}_D \quad U : \text{set} \quad T : U \to D}{\mathrm{map}_\phi : \mathrm{arg}_\phi \to \mathrm{Arg}_\phi}$$



Constructors

$$\mathsf{nil} : \mathsf{SP}_D$$

$$\frac{A : \mathsf{set} \qquad \theta : A \to \mathsf{SP}_D}{\mathsf{nonind}(A, \theta) : \mathsf{SP}_D}$$

$$\frac{A : \mathsf{set} \qquad \theta : (A \to D) \to \mathsf{SP}_D}{\mathsf{ind}(A, \theta) : \mathsf{SP}_D}$$

Definitions

$$
\begin{aligned}
\mathrm{Arg}_{\mathrm{nil}} &= \mathbb{1} \\
\mathrm{Arg}_{\mathrm{nonind}(A,\theta)} &= (x : A) \times \mathrm{Arg}_{\theta(x)} \\
\mathrm{Arg}_{\mathrm{ind}(A,\theta)} &= (f : A \to D) \times \mathrm{Arg}_{\theta(f)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{arg}_{\mathrm{nil}} &= \mathbb{1} \\
\mathrm{arg}_{\mathrm{nonind}(A,\theta)} &= (x : A) \times \mathrm{arg}_{\theta(x)} \\
\mathrm{arg}_{\mathrm{ind}(A,\theta)} &= (f : A \to U) \times \mathrm{arg}_{\theta(T \circ f)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{map}_{\mathrm{nil}}(\diamond) &= \diamond \\
\mathrm{map}_{\mathrm{nonind}(A,\theta)}(\langle a, \gamma \rangle) &= \langle a, \mathrm{map}_{\theta(a)}(\gamma) \rangle \\
\mathrm{map}_{\mathrm{ind}(A,\theta)}(\langle f, \gamma \rangle) &= \langle T \circ f, \mathrm{map}_{\theta(T \circ f)}(\gamma) \rangle
\end{aligned}
$$

Typing rules

Common premises: D : type, $\phi : \mathrm{SP}_D$, $d : \mathrm{Arg}_\phi \to D$

Formation rules U_ϕ : set

$$T_\phi : U_\phi \to D$$

Introduction rule:

$$\frac{a : \mathrm{arg}_\phi}{\mathrm{intro}_\phi(a) : U_\phi}$$

Equality rule:

$$\frac{a : \mathrm{arg}_\phi}{T_\phi(\mathrm{intro}_\phi(a)) = d(map_\phi(a))}$$

Natural numbers

- We set $D = \mathbb{1}$ and $d = \lambda x : \mathrm{Arg}_{\phi}\,.\diamond$

Natural numbers

- We set $D = \mathbb{1}$ and $d = \lambda x : \mathrm{Arg}_\phi . \diamond$
- Let $\phi = \mathrm{nonind}(\mathbb{B}, \lambda x.\mathrm{if}\ x\ \mathrm{then}\ \ \mathrm{nil}\ \ \mathrm{else}\ \ \mathrm{ind}(\mathbb{1}, \lambda y.\,\mathrm{nil}))$
  Then we set $\mathbb{N} := U_\phi$

Natural numbers

- We set $D = \mathbb{1}$ and $d = \lambda x : \mathrm{Arg}_\phi\,.\diamond$
- Let $\phi = \mathrm{nonind}(\mathbb{B}, \lambda x.\mathbf{if}\ x\ \mathbf{then}\ \ \mathrm{nil}\ \ \mathbf{else}\ \ \mathrm{ind}(\mathbb{1}, \lambda y.\,\mathrm{nil}))$
  Then we set $\mathbb{N} := U_\phi$ with
  - $O := \mathrm{intro}(\langle tt, \diamond\rangle) : \mathbb{N}$ and
  - $S := \lambda n.\,\mathrm{intro}(\langle ff, \langle \lambda y.n, \diamond\rangle\rangle : \mathbb{N} \to \mathbb{N}$

Structural recursion

We will now consider structural recursion of U into a type E.
We'll write $E[t]$ for substitution of some fixed variable in E by t, and E instead of $\lambda x.E[x]$. Global assumption $x : U_\phi \Rightarrow E[x]$: type

Structural recursion

We will now consider structural recursion of U into a type E.
We'll write $E[t]$ for substitution of some fixed variable in E by t, and E instead of $\lambda x.E[x]$. Global assumption $x : U_\phi \Rightarrow E[x]$: type

$$\begin{array}{ccc}
\mathrm{arg}_\phi & \xrightarrow{\quad \mathrm{intro} \quad} & U \\
\Big\downarrow \langle \mathrm{id}, \mathrm{mapIH}_\phi(R(e)) \rangle & & \Big\downarrow \langle \mathrm{id}, R(e) \rangle \\
(\gamma : \mathrm{arg}_\phi) \times \mathrm{IH}_\phi(\gamma) & \xrightarrow{\langle \mathrm{intro} \circ \pi_0, e \rangle} & (x : U) \times E[x]
\end{array}$$

More operations

For this, we introduce the operations IH and mapIH.

$$\frac{U : \mathrm{set} \qquad T : (x : U) \to D \qquad \gamma : \mathrm{arg}_\phi}{\mathrm{IH}_\phi(\gamma)}$$

$$\frac{U : \mathrm{set} \qquad T : (x : U) \to D \qquad R : (x : U) \to E[x]}{\mathrm{mapIH}_\phi(R) : (x : \mathrm{arg}_\phi) \to \mathrm{IH}_\phi(x)}$$

More definitions

$$\begin{aligned}
\mathsf{IH}_{\mathsf{nil}}(\diamond) &= \mathbb{1} \\
\mathsf{IH}_{\mathsf{nonind}(A,\theta)}(\langle a, \gamma\rangle) &= \mathsf{IH}_{\theta(a)}(\gamma) \\
\mathsf{IH}_{\mathsf{ind}(A,\theta)}(\langle f, \gamma\rangle) &= ((y : A) \to E[f(y)]) \times (\mathsf{IH}_{\theta(T\circ f)}(\gamma))
\end{aligned}$$

$$\begin{aligned}
\mathsf{mapIH}_{\mathsf{nil}}(R, \diamond) &= \diamond \\
\mathsf{mapIH}_{\mathsf{nonind}(A,\theta)}(R, \langle a, \gamma\rangle) &= \mathsf{mapIH}_{\theta(a)}(\gamma) \\
\mathsf{mapIH}_{\mathsf{ind}(A,\theta)}(R, \langle f, \gamma\rangle) &= \langle R \circ f, \mathsf{mapIH}_{\theta(T\circ f)}(R, \gamma)\rangle
\end{aligned}$$

Elimination rule

Elimination rule: $$\frac{e : (\gamma : \mathrm{arg}_\phi) \to (\mathrm{IH}_\phi(\gamma)) \to (E[\mathrm{intro}_\phi(\gamma)])}{\mathrm{R}_\phi(e) : (a : U_\phi) \to E[a]}$$

Equality rule: $$R_\phi(e, \mathrm{intro}_\phi(\gamma)) = e(\gamma, \mathrm{mapIH}_\phi(R_\phi(e), \gamma))$$

Universe closed under Σ and ℕ

$$
\begin{aligned}
D &: \text{set} \\
\phi := \text{nonind}(\mathbb{B}, \lambda x.&\text{if } x \\
&\quad \text{then nil} \\
&\quad \text{else ind}(\mathbb{1}, \lambda f.\,\text{ind}(f(\diamond), \lambda y.\,\text{nil})))
\end{aligned}
$$

Universe closed under Σ and $\mathbb{N}$

$$
\begin{aligned}
D &: \text{set} \\
\phi &:= \text{nonind}(\mathbb{B}, \lambda x.\text{if } x \\
&\qquad\qquad\qquad \text{then nil} \\
&\qquad\qquad\qquad\ \text{else ind}(\mathbb{1}, \lambda f.\,\text{ind}(f(\diamond), \lambda y.\,\text{nil})))
\end{aligned}
$$

- $\text{Arg}_\phi = (b : \mathbb{B}) \times E(x)$
  - $E(tt) = \mathbb{1}$
  - $E(ff) = (x : \mathbb{1} \to \text{set}) \times (f : x(\diamond) \to \text{set}) \times \mathbb{1}$

$$
\begin{aligned}
\text{Arg}_{\text{nil}} &= \mathbb{1} \\
\text{Arg}_{\text{nonind}(A,\theta)} &= (x : A) \times \text{Arg}_{\theta(x)} \\
\text{Arg}_{\text{ind}(A,\theta)} &= (f : A \to D) \times \text{Arg}_{\theta(f)}
\end{aligned}
$$

Universe closed under Σ and $\mathbb{N}$ (cont.)

$\mathrm{Arg}_\phi = (x : \mathbb{B}) \times E(x)$

- $E(tt) = \mathbb{1}$
- $E(ff) = (x : \mathbb{1} \to \mathrm{set}) \times (f : x(\diamond) \to \mathrm{set}) \times \mathbb{1}$

- Then we define $d : \mathrm{Arg}_\phi \to \mathrm{set}$ such that
  - $d(\langle tt, \diamond \rangle) = \mathbb{N}$
  - $d(\langle ff, \langle A, \langle B, \diamond \rangle \rangle \rangle) = \Sigma(A(\diamond), \lambda y.B(y))$

Universe closed under Σ and $\mathbb{N}$ (cont.)

$\mathrm{Arg}_\phi = (x : \mathbb{B}) \times E(x)$

- $E(tt) = \mathbb{1}$
- $E(\mathit{ff}) = (x : \mathbb{1} \to \mathrm{set}) \times (f : x(\diamond) \to \mathrm{set}) \times \mathbb{1}$

- Then we define $d : \mathrm{Arg}_\phi \to \mathrm{set}$ such that
  - $d(\langle tt, \diamond \rangle) = \mathbb{N}$
  - $d(\langle \mathit{ff}, \langle A, \langle B, \diamond \rangle\rangle\rangle) = \Sigma(A(\diamond), \lambda y.B(y))$
- We can then set $T' := T_\phi$, $U' := U_\phi$ and our effective constructors:
  - $\hat{\mathbb{N}} := \mathrm{intro}(\langle tt, \diamond \rangle) : U'$
  - $\hat{\Sigma} := \lambda ab.\, \mathrm{intro}(\langle \mathit{ff}, \langle \lambda x.a, \langle b, \diamond \rangle\rangle\rangle)$
    $: (a : U', b : T'(a) \to U') \to U'$

Universe closed under Σ and $\mathbb{N}$ (cont.)

$\mathrm{Arg}_\phi = (x : \mathbb{B}) \times E(x)$

- $E(tt) = \mathbb{1}$
- $E(ff) = (x : \mathbb{1} \to \mathrm{set}) \times (f : x(\diamond) \to \mathrm{set}) \times \mathbb{1}$

- Then we define $d : \mathrm{Arg}_\phi \to \mathrm{set}$ such that
  - $d(\langle tt, \diamond \rangle) = \mathbb{N}$
  - $d(\langle ff, \langle A, \langle B, \diamond \rangle\rangle\rangle) = \Sigma(A(\diamond), \lambda y.B(y))$
- We can then set $T' := T_\phi$, $U' := U_\phi$ and our effective constructors:
  - $\hat{\mathbb{N}} := \mathrm{intro}(\langle tt, \diamond \rangle) : U'$
  - $\hat{\Sigma} := \lambda ab.\,\mathrm{intro}(\langle ff, \langle \lambda x.a, \langle b, \diamond \rangle\rangle\rangle)$
    $: (a : U', b : T'(a) \to U') \to U'$
- $T'(\hat{\mathbb{N}}) = \mathbb{N}$ and $T'(\hat{\Sigma}(a, b)) = \Sigma(T'(a), T' \circ b)$