Logical Verification
Guest Lecture: Inductive Types

Niels van der Weide

Inductive Types

- Inductive types are a key feature in proof assistants, and important for building data types
- They generalize algebraic data types: natural numbers, lists, trees
- They are also useful in mathematics: transitive closure, free monoid, free group, being even/odd

Example: natural numbers

```
inductive ℕ : Type where
  | Z : ℕ
  | S : ℕ → ℕ
```

Recursive Functions

We use pattern matching and recursion to define functions on inductive types.

```
def fact : ℕ → ℕ
| Z ⇒ Z
| S n ⇒ (n+1) · fact n
```

Example: lists

```
inductive list (α : Type) : Type where
  | nil : list α
  | cons : α → list α → list α
```

Example: vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

Features of Inductive Types

We desire the following of inductive types

- We specify them by giving parameters, indices, and constructors
- One can do pattern matching on inductive types
- There should be many examples: natural numbers, lists, vectors, ...

This Lecture

Goal of this lecture

- A **scheme** for inductive types
- **Examples** of inductive types
- The **elimination rule** for inductive types
- Recent work on inductive types

The material of this lecture is based on "Inductive families" by Peter Dybjer[1]

[1] https://dl.acm.org/doi/10.1007/BF01211308

Outline

Outline

Schemes for Inductive Types

Inductive types are specified by the following ingredients:

- **Parameters**: arguments of the inductive type that are **uniform** over the constructors

Schemes for Inductive Types

Inductive types are specified by the following ingredients:

- **Parameters**: arguments of the inductive type that are **uniform** over the constructors
- **Indices**: arguments of the inductive type that can **vary** over the constructors

Schemes for Inductive Types

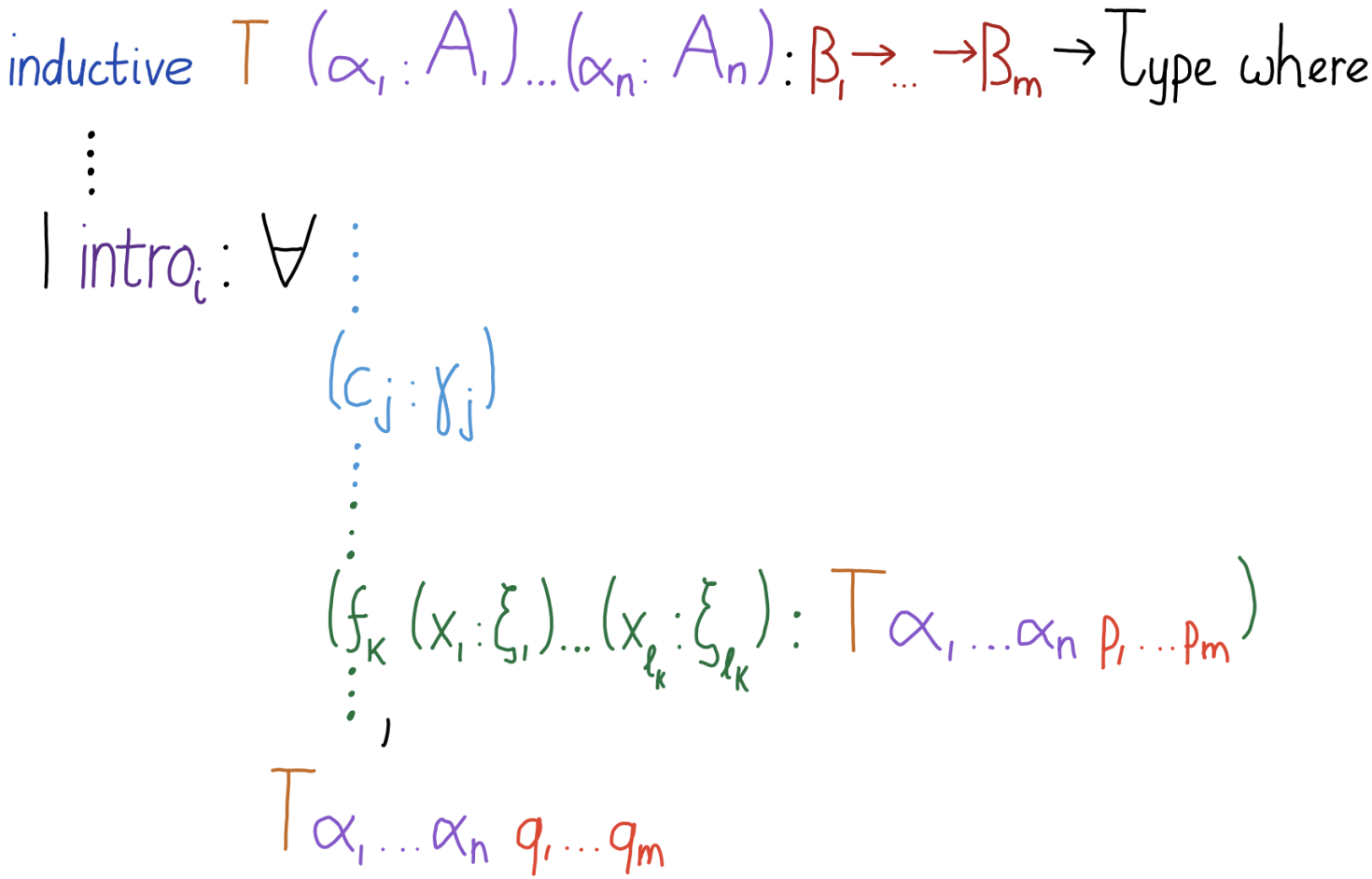
Inductive types are specified by the following ingredients:

- **Parameters**: arguments of the inductive type that are **uniform** over the constructors
- **Indices**: arguments of the inductive type that can **vary** over the constructors
- **Constructors**: determined by their recursive and non-recursive premises

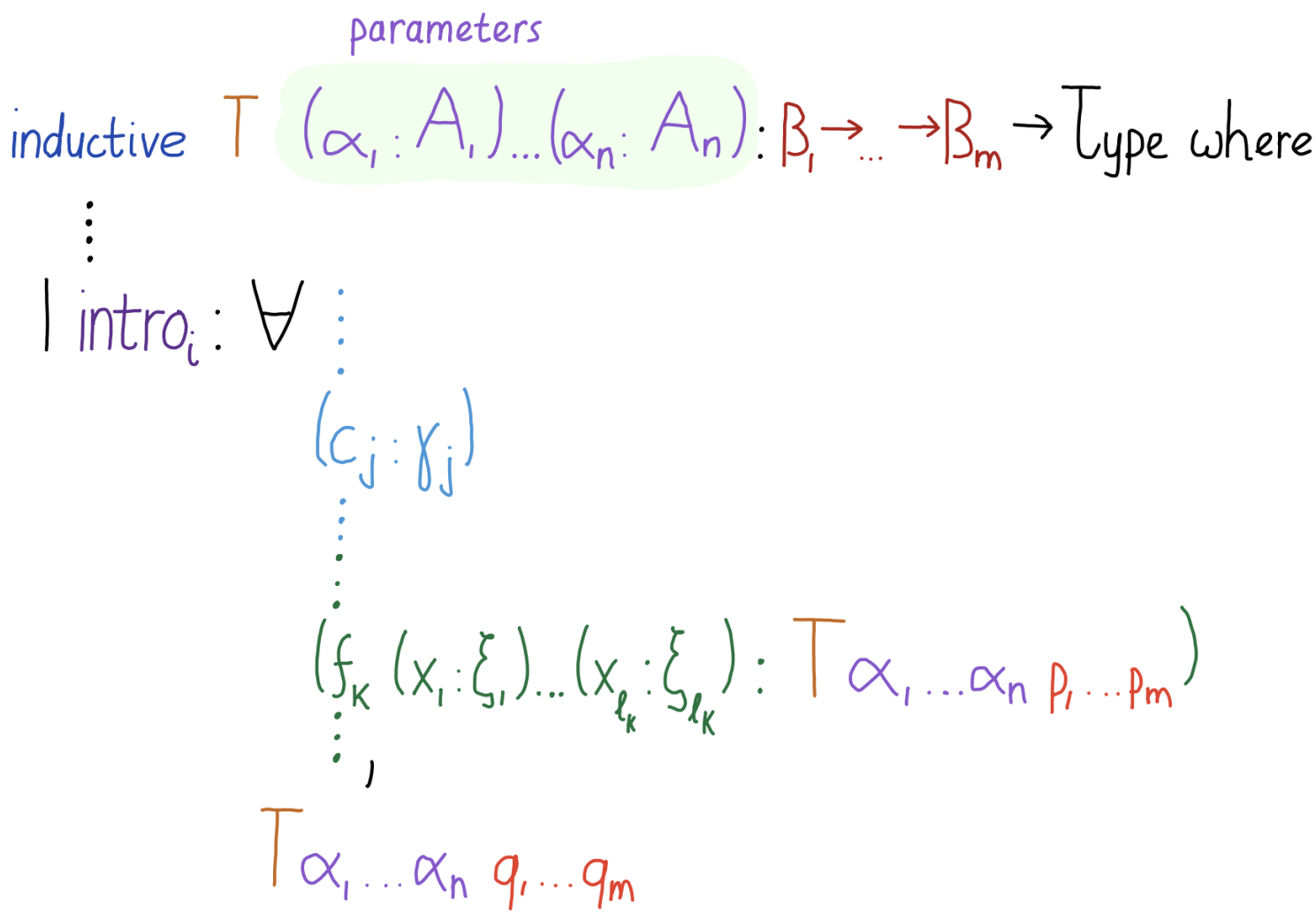
Recall: vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

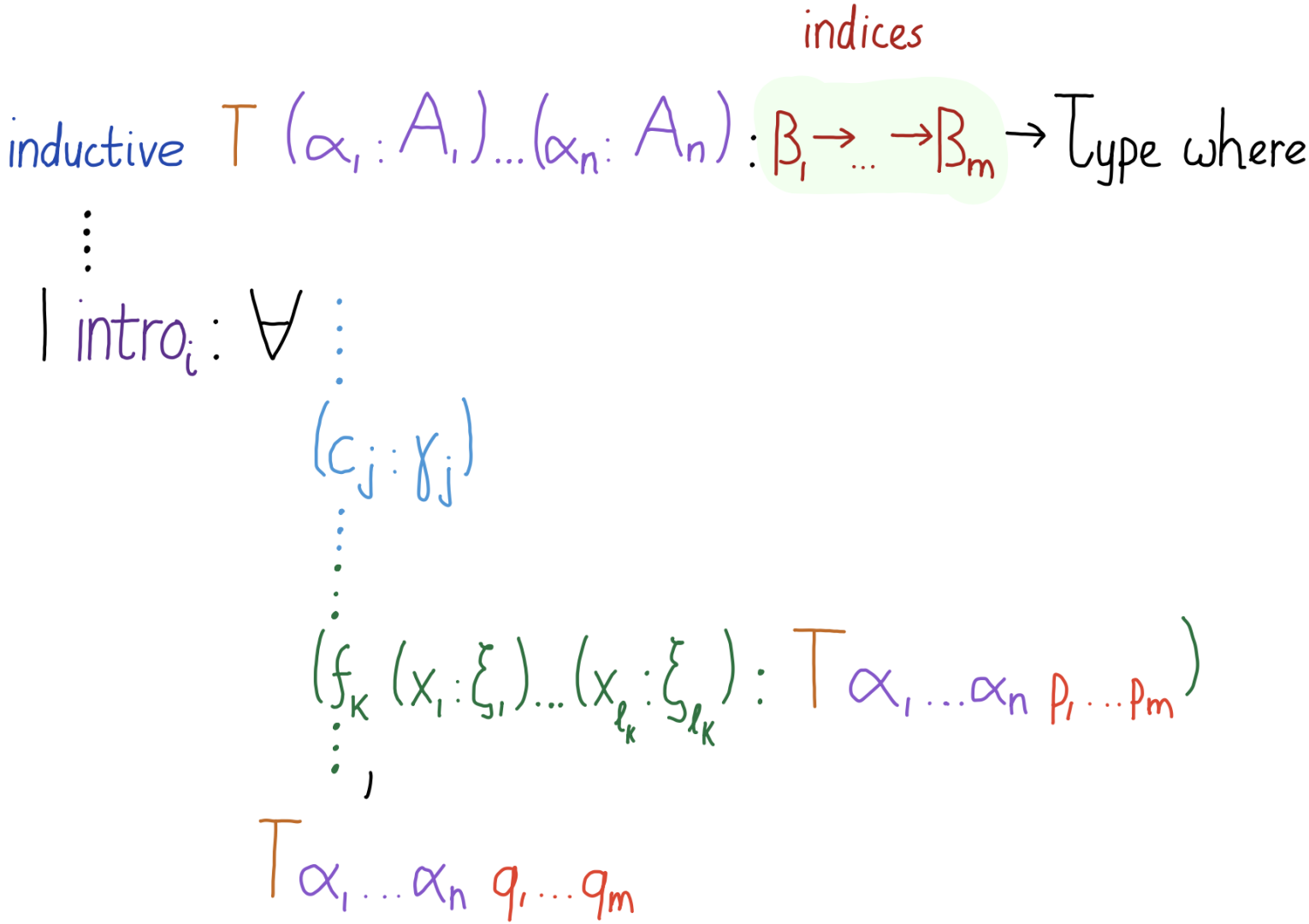
Schemes for Inductive Types



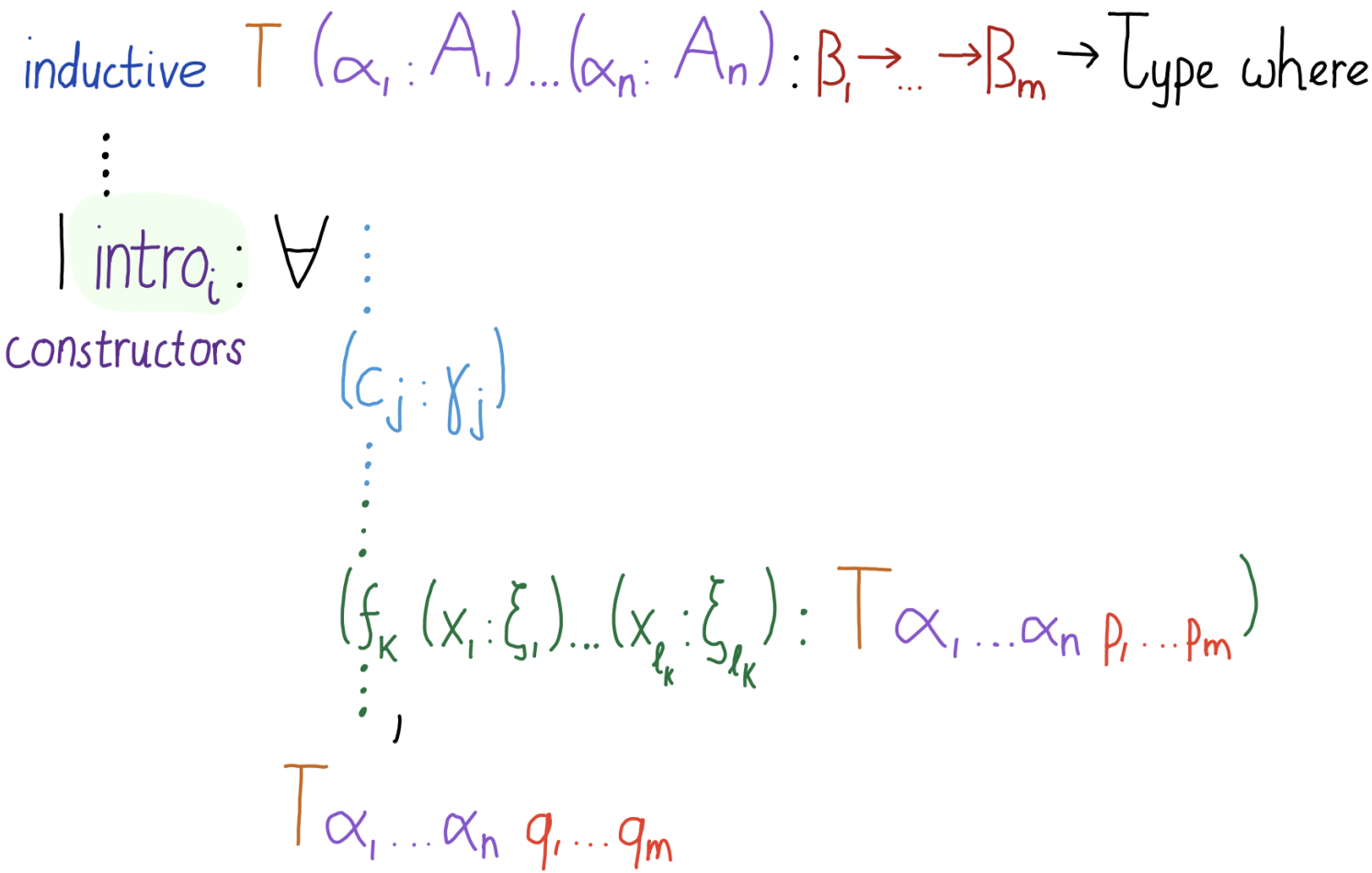
Schemes for Inductive Types



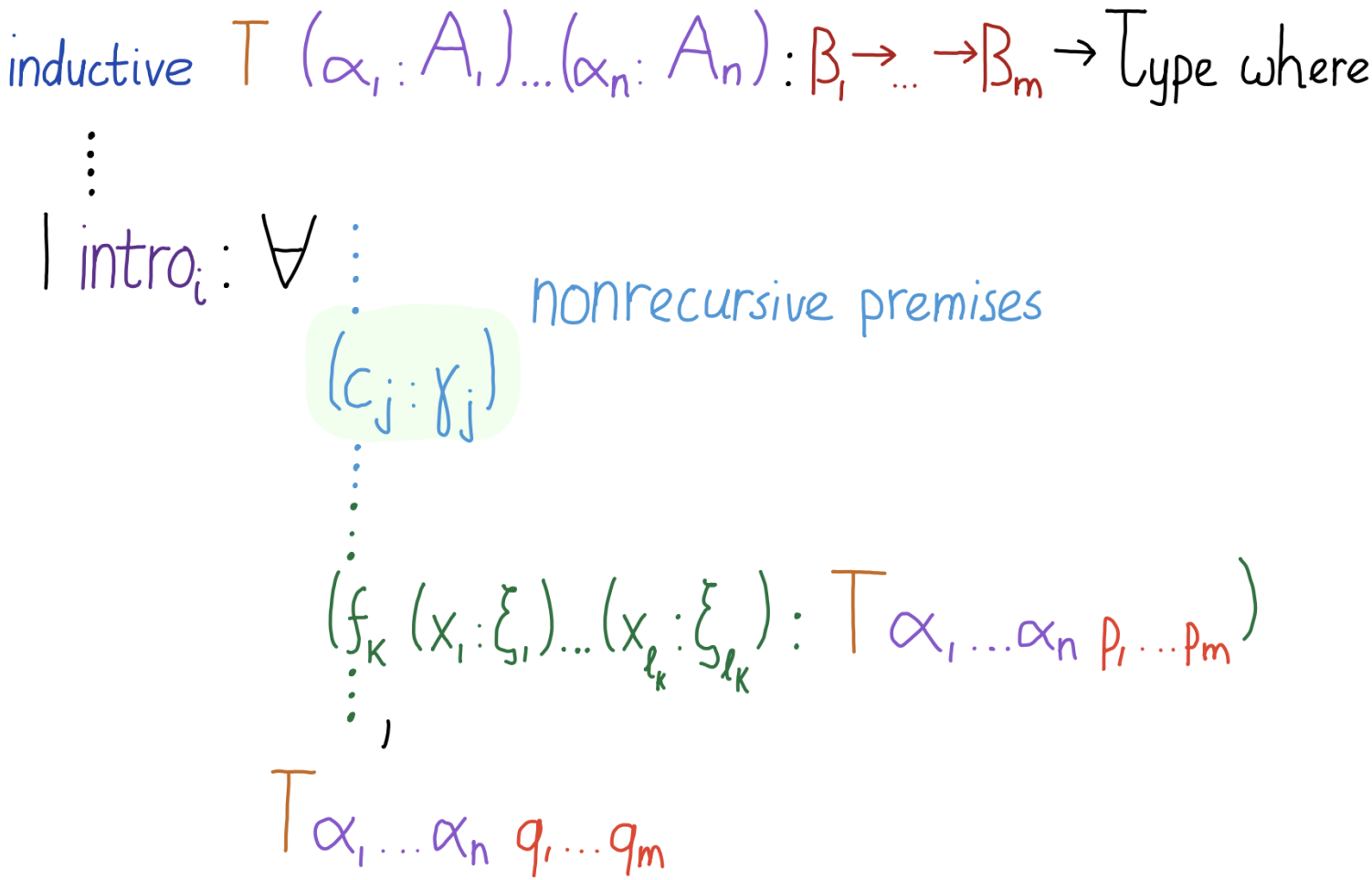
Schemes for Inductive Types



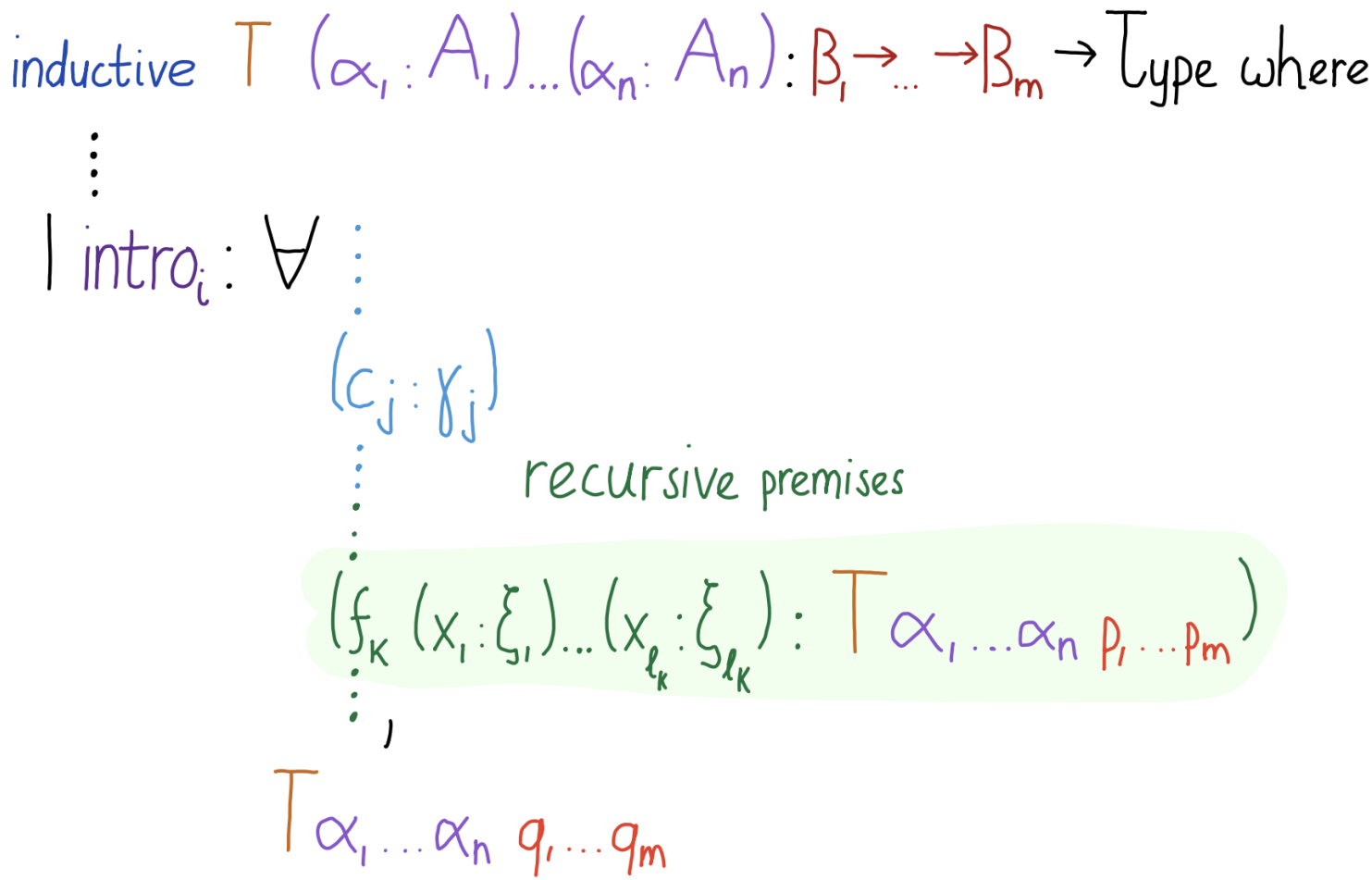
Schemes for Inductive Types



Schemes for Inductive Types



Schemes for Inductive Types

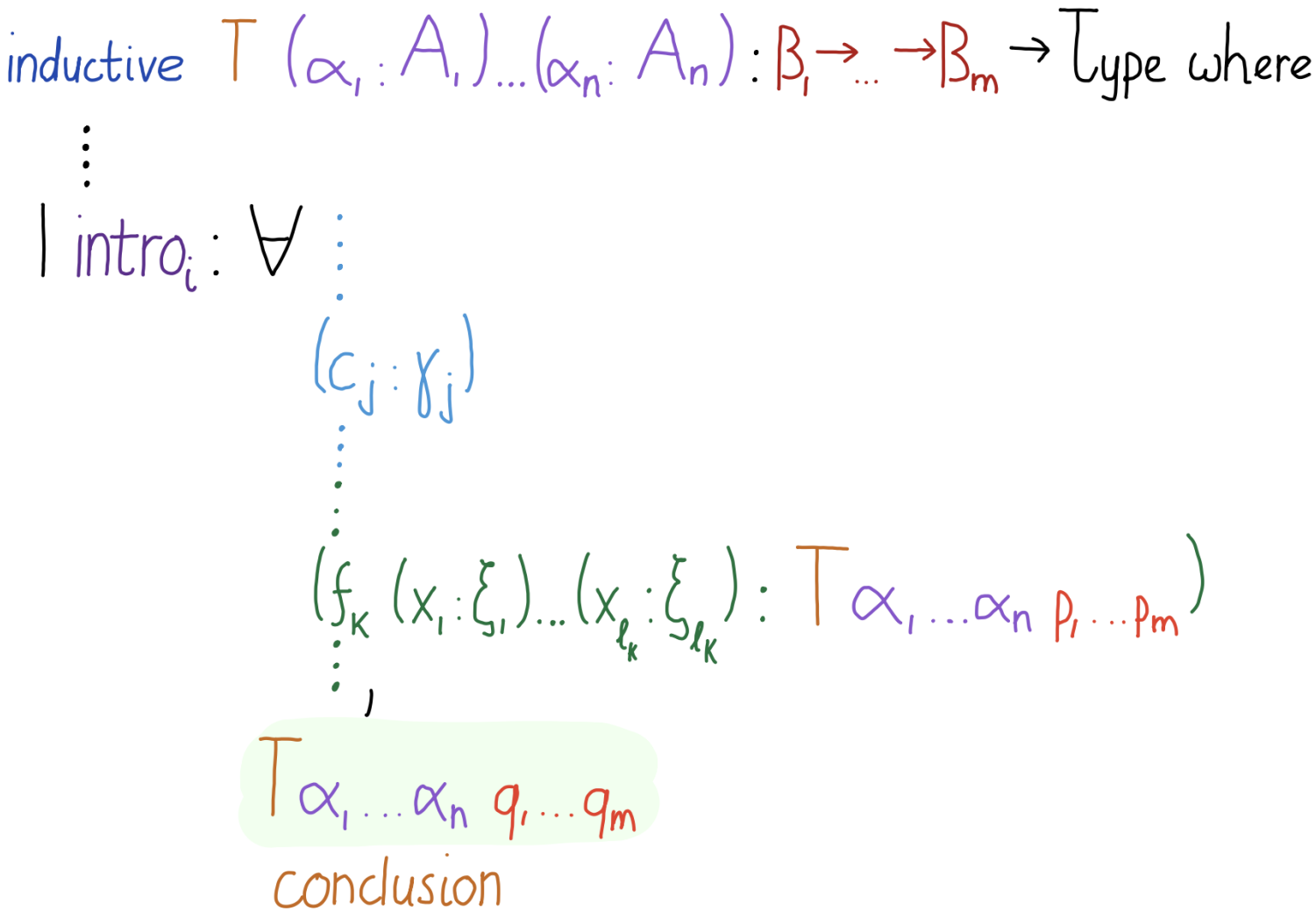


Schemes for Inductive Types

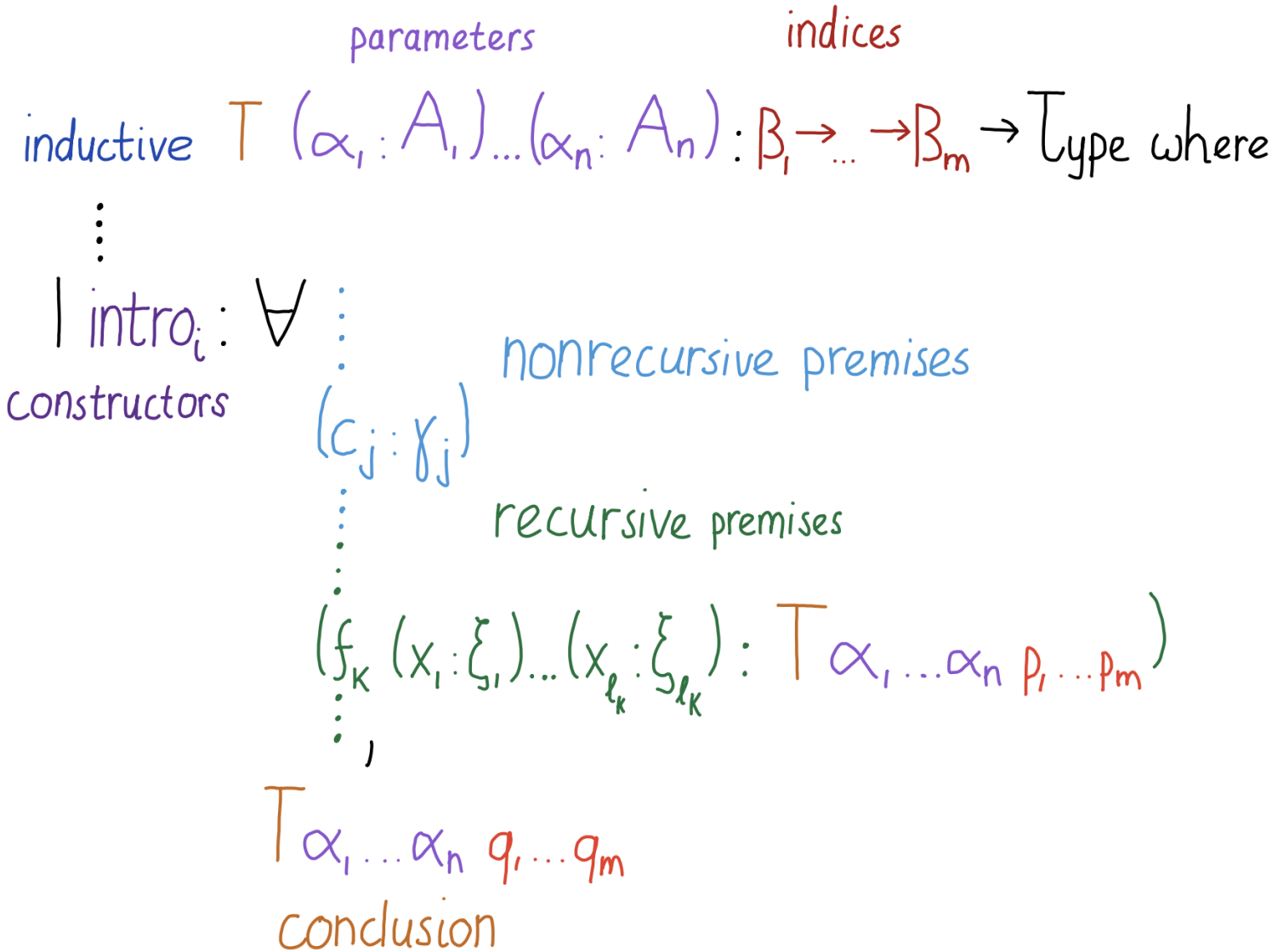
Constructors are of the shape

$$\ldots \to \gamma_j \to \ldots$$
$$\to \ldots \to (\xi_1 \to \ldots \to \xi_l \to T\ \alpha_1\ \ldots\ \alpha_n\ p_1\ \ldots\ p_m) \to \ldots$$
$$\to T\ \alpha_1\ \ldots\ \alpha_n\ q_1\ \ldots\ q_m$$

Schemes for Inductive Types

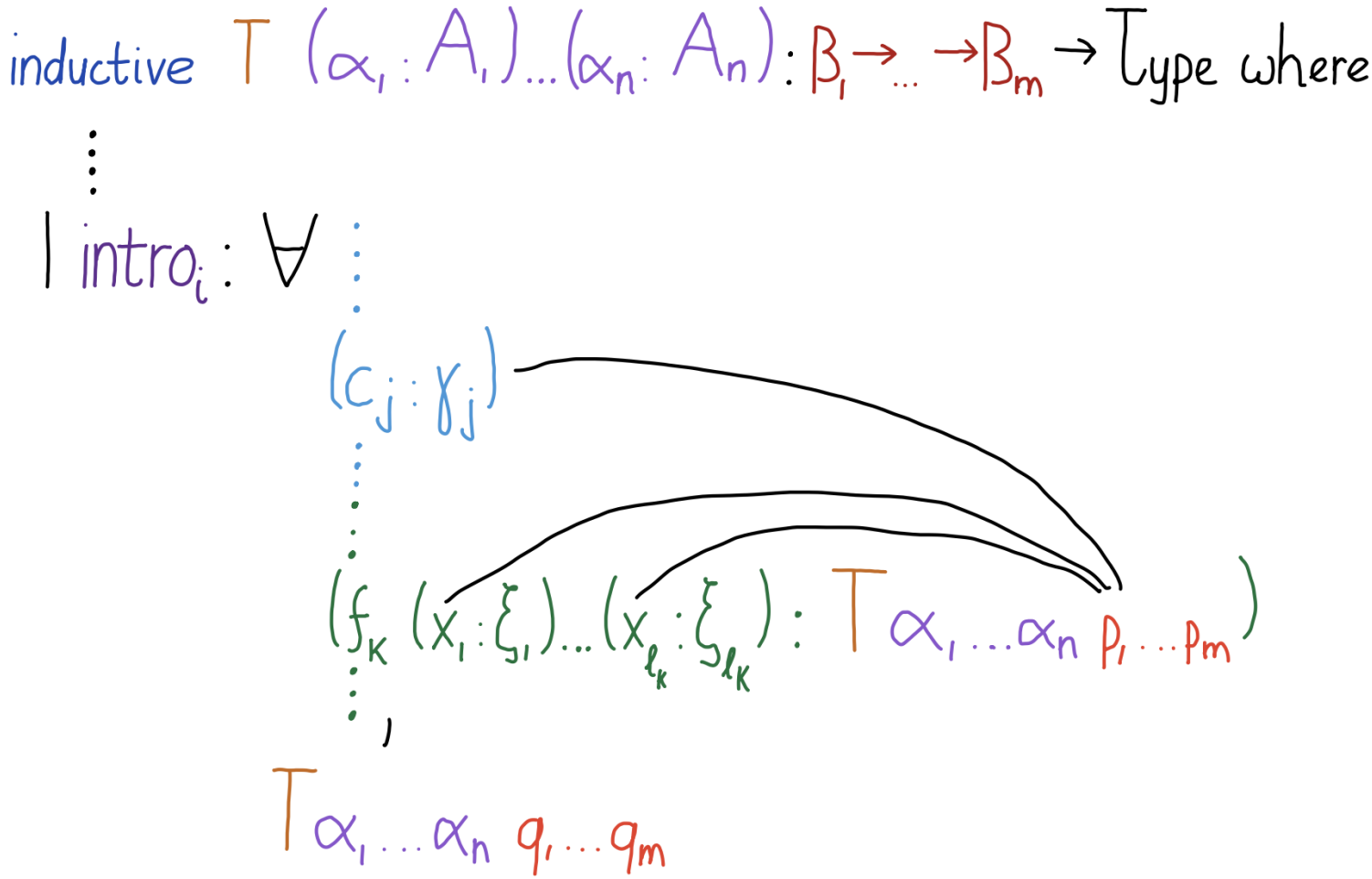


Schemes for Inductive Types



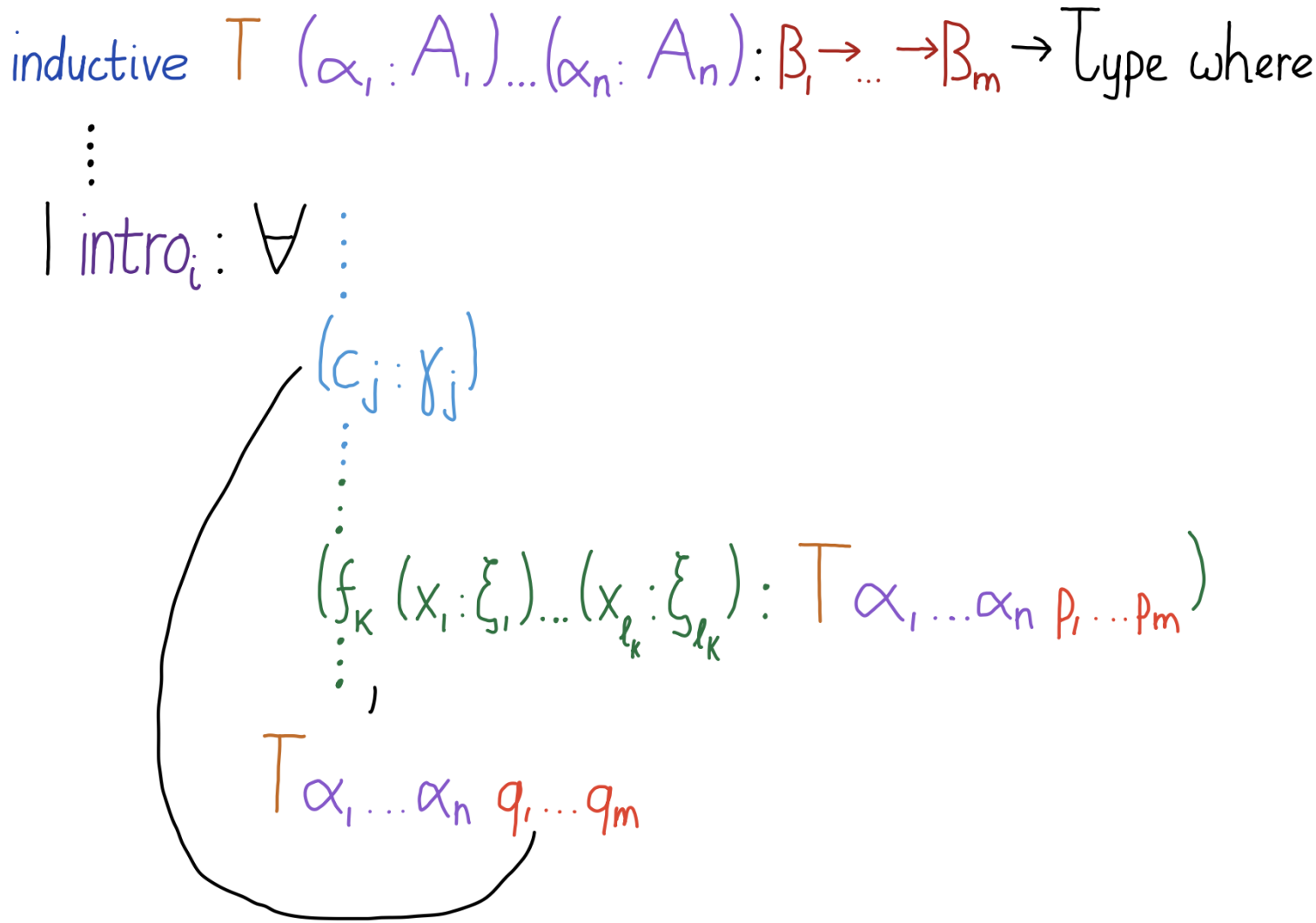
Schemes for Inductive Types

The indices are terms using previously introduced variables



Schemes for Inductive Types

The indices are terms using previously introduced variables



The Natural Numbers

```
inductive ℕ : Type where
  | Z : ℕ
  | S : ℕ → ℕ
```

The Natural Numbers

no parameters or indices

```
inductive ℕ : Type where
  | Z : ℕ
  | S : ℕ → ℕ
```

The Natural Numbers

inductive $\mathbb{N}$: Type where

| $Z : \mathbb{N}$ no premises

| $S : \mathbb{N} \to \mathbb{N}$

The Natural Numbers

```
inductive ℕ : Type where
  | Z : ℕ
  | S : ℕ → ℕ
```

↳ only a recursive premise

Lists

```
inductive list (α : Type) : Type where
  | nil : list α
  | cons : α → list α → list α
```

Lists

one parameter, no indices

```
inductive list (α : Type) : Type where
  | nil : list α
  | cons : α → list α → list α
```

Lists

```
inductive list (α : Type) : Type where
  | nil : list α
  | cons : α → list α → list α
```

one nonrecursive
and one recursive premise

Vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

Vectors

one parameter and index

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

Vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

two nonrecursive premises

Vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

one recursive premise

Vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

Binary Trees

```
inductive tree (α : Type) : Type where
  | leaf : tree α
  | node : α → tree α → tree α → tree α
```

Binary Trees

one parameter, no indices

```
inductive tree (α : Type) : Type where
  | leaf : tree α
  | node : α → tree α → tree α → tree α
```

Binary Trees

```
inductive tree (α : Type) : Type where
  | leaf : tree α
  | node : α → tree α → tree α → tree α
```

one nonrecursive and two recursive premises

Countably Branching Trees

```
inductive tree∞ (α : Type) : Type where
  | leaf : tree∞ α
  | node : (ℕ → tree∞ α) → tree∞ α
```

Countably Branching Trees

One parameter

```
inductive tree∞ (α : Type) : Type where
  | leaf : tree∞ α
  | node : (ℕ → tree∞ α) → tree∞ α
```

Countably Branching Trees

```
inductive tree∞ (α : Type) : Type where
  | leaf : tree∞ α
  | node : (ℕ → tree∞ α) → tree∞ α
```

one recursive premise

And there are many more examples

- Height indexed trees
- AVL trees
- Equality (the least reflexive relation)
- The transitive closure of a relation
- Wellfounded trees (aka W-types)

Outline

Rules for Data Types

Data types are specified by four kinds of rules

- **Formation rules**: how to **form** that type
- **Introduction rules**: how to construct **inhabitants** of that type
- **Elimination rules**: how to construct **functions** from that type
- **Computation rules**: computational behavior of the elimination rule

Rules for Data Types

Data types are specified by four kinds of rules

- **Formation rules**: how to **form** that type
- **Introduction rules**: how to construct **inhabitants** of that type
- **Elimination rules**: how to construct **functions** from that type
- **Computation rules**: computational behavior of the elimination rule

Since inductive types are specified by their formation and introduction rules, the challenge lies in deriving the elimination and computation rules.

Formation Rule

$$\frac{\alpha_1 : A_1 \quad \ldots \quad \alpha_n : A_n \qquad b_1 : \beta_m \ldots b_m : \beta_m}{T\ \alpha_1 \ldots \alpha_m \ \ b_1 \ldots b_m : \mathrm{Type}}$$

Introduction Rules

$$\frac{\ldots\ (c_j : \gamma_j)\ldots\ldots\ (f_K : \forall (x_1 : \xi_1)\ldots(x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \cdots p_m\ \ldots}{\text{intro}_i\ \ldots\ c_j \ldots \ldots f_K \ldots : T\ \alpha_1 \ldots \alpha_n\ q_1 \ldots q_m}$$

Elimination Rules for Inductive Types

- The elimination rule says how to make functions from some inductive type
- They generalize recursive functions and proofs by induction

Elimination Rules for Inductive Types

- The elimination rule says how to make functions from some inductive type
- They generalize recursive functions and proofs by induction
- However, we specify inductive types via their formation and introduction rules
- So: we need to derive the elimination rule from their description

Elimination Rules for Inductive Types

- The elimination rule says how to make functions from some inductive type
- They generalize recursive functions and proofs by induction
- However, we specify inductive types via their formation and introduction rules
- So: we need to derive the elimination rule from their description
- Main idea: functions are specified by what they do on each constructor

Note: the elimination rule can be derived systematically

Example: Elimination for Natural Numbers

$$\overline{\mathit{elim}} : \forall (n : \mathbb{N}),\ P\,n$$

$$\overline{\mathit{elim}}\ Z = \delta_Z$$

$$\overline{\mathit{elim}}\ (S\,n) = \delta_S\ n\ (\overline{\mathit{elim}}\ n)$$

Example: Elimination for Natural Numbers

$$
\begin{aligned}
\mathrm{elim}: \forall\ & (P : \mathbb{N} \to \mathrm{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \\
& (x : \mathbb{N}), \\
& P\ x
\end{aligned}
$$

Example: Elimination for Vectors

$$\overline{elim} : \forall\, (\alpha : Type)\,(n : \mathbb{N})\,(x : vec\ \alpha\ n),\ P\ n\ x$$

$$\overline{elim}\ 0\ \ nil = \delta_{nil}$$

$$\overline{elim}\ (n+1)\ (cons\ n\ x\ xs) = \delta_{cons}\ n\ x\ xs\ (\overline{elim}\ xs)$$

Example: Elimination for Vectors

$$
\begin{aligned}
&\mathsf{elim}: \forall\, (\alpha : \mathsf{Type}) \\
&\qquad (P : \forall (n : \mathbb{N}),\ \mathsf{vec}\ \alpha\ n \to \mathsf{Type}) \\
&\qquad (\delta_{\mathsf{nil}} : P\ 0\ \mathsf{nil}) \\
&\qquad (\delta_{\mathsf{cons}} : \forall\, (n : \mathbb{N})(x : \alpha)(xs : \mathsf{vec}\ \alpha\ n)(H : P\ n\ xs), \\
&\qquad\qquad P\ (n+1)\ (\mathsf{cons}\ n\ x\ xs)) \\
&\qquad (n : \mathbb{N}) \\
&\qquad (x : \mathsf{vec}\ \alpha\ n), \\
&\quad P\ n\ x
\end{aligned}
$$

Derivation of the Elimination Rule

$$\mathit{elim}: \forall\,(\alpha_1 : A_1)\ldots(\alpha_n : A_n)$$

Derivation of the Elimination Rule

$$\mathit{elim}: \forall\,(\alpha_1 : A_1)\ldots(\alpha_n : A_n)$$
$$(P : \forall\,(b_1 : \beta_1)\ldots(b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})$$

Derivation of the Elimination Rule

$$elim : \forall (\alpha_1 : A_1) \ldots (\alpha_n : A_n)$$

$$(P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to Type)$$

$$\vdots$$

$$(\delta_i :$$

$$)$$

Derivation of the Elimination Rule

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\
& (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type}) \\
& \vdots \\
& (\delta_i : \forall \ldots (c_j : \gamma_j) \ldots \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad)
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
\mathit{elim}: \forall\, & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\
& (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type}) \\
& \vdots \\
& (\delta_i : \forall \ldots (c_j : \gamma_j) \ldots \\
& \qquad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad)
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\
& (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type}) \\
& \vdots \\
& (\delta_i : \forall \ldots (c_j : \gamma_j) \ldots \\
& \qquad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
& \qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K})) \\
& \qquad\qquad).
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
&\mathit{elim}: \forall\,(\alpha_1 : A_1)\ldots(\alpha_n : A_n)\\
&\quad (P : \forall (b_1 : \beta_1)\ldots(b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\\
&\quad \vdots\\
&\quad (\delta_i : \forall\ \ldots (c_j : \gamma_j)\ldots\\
&\qquad \ldots\ (f_K : \forall (x_1 : \xi_1)\ldots(x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n)\ldots\\
&\qquad \ldots\ (H_K : \forall (x_1 : \xi_1)\ldots(x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K}))\,.\\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots\ \ldots (f_K\ x_1 \ldots x_{l_K})\ldots))\\
&\quad \vdots
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
&\mathit{elim}: \forall\,(\alpha_1 : A_1)\ldots(\alpha_n : A_n)\\
&\quad (P : \forall (b_1 : \beta_1)\ldots(b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\\
&\quad \vdots\\
&\quad (\delta_i : \forall\ \ldots (c_j : \gamma_j)\ldots\\
&\qquad \ldots\ (f_K : \forall (x_1 : \xi_1)\ldots(x_{\ell_K} : \xi_{\ell_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n)\ldots\\
&\qquad \ldots\ (H_K : \forall (x_1 : \xi_1)\ldots(x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K}))\ldots\\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{\ell_K})\ldots))\\
&\quad \vdots\\
&\quad (b_1 : \beta_1)\ldots(b_m : \beta_m)
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
&\mathit{elim}: \forall\,(\alpha_1 : A_1)\ldots(\alpha_n : A_n)\\
&\quad (P : \forall (b_1 : \beta_1)\ldots(b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\\
&\quad \vdots\\
&\quad (\delta_i : \forall\ \ldots (c_j : \gamma_j) \ldots\\
&\qquad \ldots\ (f_K : \forall (x_1 : \xi_1)\ldots(x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots\\
&\qquad \ldots (H_K : \forall (x_1 : \xi_1)\ldots(x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K}))\,.\\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots))\\
&\quad \vdots\\
&\quad (b_1 : \beta_1)\ldots(b_m : \beta_m)\\
&\quad (x : T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m)\,.
\end{aligned}
$$

Derivation of the Elimination Rule

$$
\begin{aligned}
&\mathit{elim}: \forall\, (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\
&\quad (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type}) \\
&\quad \vdots \\
&\quad (\delta_i : \forall \ldots (c_j : \gamma_j) \ldots \\
&\qquad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
&\qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})). \\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots)) \\
&\quad \vdots \\
&\quad (b_1 : \beta_1) \ldots (b_m : \beta_m) \\
&\quad (x : T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m), \\
&\quad P\ b_1 \ldots b_m\ x
\end{aligned}
$$

Computation Rules

Write $\overline{\text{elim}}$ for $\text{elim}\ \alpha_1 \ldots \alpha_n\ P \ldots\ \delta_i \ldots\ p_1 \ldots p_m$

$\overline{\text{elim}}\ (\text{intro}_i \ \ldots c_j \ldots \ \ldots (f_K\ x_1 \ldots x_{\ell_K}) \ldots)$

$= \delta_i \ \ldots c_j \ldots \ \ldots (f_K\ x_1 \ldots x_{\ell_K}) \ldots$

Computation Rules

Write $\overline{\text{elim}}$ for $\text{elim}\ \alpha_1 \ldots \alpha_n\ P \ldots\ \delta_i \ldots\ p_1 \ldots p_m$

$\overline{\text{elim}}\ (\text{intro}_i \ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_r}) \ldots)$

$= \delta_i \ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots \ldots (\lambda y_1 \ldots y_{l_K}.\ \overline{\text{elim}} \ldots (f_K\ x_1 \ldots x_{l_r}) \ldots)$

Recall: The Natural Numbers

```
inductive ℕ : Type where
  | Z : ℕ
  | S : ℕ → ℕ
```

Elimination for the Natural Numbers

$$\begin{aligned}\mathit{elim}: \forall\ & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\ & (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\end{aligned}$$

Elimination for the Natural Numbers

$$\mathsf{elim} : \forall\ (P : \mathbb{N} \to \mathsf{Type})$$

Elimination for the Natural Numbers

$$
\begin{aligned}
&\mathit{elim}\colon \forall\ (P : \mathbb{N} \to \mathit{Type}) \\
&\qquad (\delta_z : \forall\ \ldots (c_j : \gamma_j) \ldots \\
&\qquad\qquad \ldots\ (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
&\qquad\qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K})) \ldots \\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 \ldots x_\ell) \ldots))
\end{aligned}
$$

Elimination for the Natural Numbers

$$
\begin{aligned}
&\mathit{elim}\colon \forall\ (P : \mathbb{N} \to \mathit{Type}) \\
&\qquad (\delta_z : \forall \\
&\qquad\qquad \ldots\ (f_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
&\qquad\qquad \ldots\ (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots \\
&\qquad\qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 .\ x_l) \ldots))
\end{aligned}
$$

Elimination for the Natural Numbers

$$\begin{aligned} elim : \forall\ & (P : \mathbb{N} \to Type) \\ & (\delta_z : P\ q_1 \ldots q_m\ (intro_i\ \ldots c_j \ldots \ldots (f, x_1 .\ x_l) \ldots)) \end{aligned}$$

Elimination for the Natural Numbers

$$\mathsf{elim} : \forall\ (P : \mathbb{N} \to \mathsf{Type})$$
$$(\delta_z : P\ Z)$$

Elimination for the Natural Numbers

$$
\begin{aligned}
\mathit{elim}\colon \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall \ldots (c_j : \gamma_j) \ldots \\
& \quad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
& \quad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K})) \ldots \\
& \quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_\ell\ x_1 \ldots x_\ell) \ldots))
\end{aligned}
$$

Elimination for the Natural Numbers

$$
\begin{aligned}
\mathit{elim}\colon \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall \\
& \quad \ldots\ (f_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
& \quad \ldots\ (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots \\
& \quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 .\ x_l) \ldots))
\end{aligned}
$$

Elimination for the Natural Numbers

$$
\begin{aligned}
\mathit{elim}\colon \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall \\
& \qquad (f : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \\
& \qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots \\
& \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 \ldots x_l) \ldots))
\end{aligned}
$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N})
\end{aligned}$$

$$\begin{aligned}
& \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P_{p_1 \ldots p_m}\ (f_K\ x_1 \ldots x_{l_K})) \ldots \\
& P_{q_1 \ldots q_m}\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f, x_1 \ldots x_l) \ldots))
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ \mathbb{Z}) \\
& (\delta_s : \forall\ (n : \mathbb{N}) \\
& \qquad (H : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P_{p_1 \ldots p_m}\ (f_K\ x_1 \ldots x_{\ell_K})) \\
& \qquad P_{q_1 \ldots q_m}\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 \ldots x_\ell) \ldots))
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N}) \\
& \qquad (H : P_{p_1 \ldots p_m}\ (f_k\ x_1 \ldots x_{l_k})) \\
& \qquad P_{q_1 \ldots q_m}\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f,\ x_1 \ldots x_l) \ldots))
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N}) \\
& \qquad (H : P\ (f_K\ x_1 \ldots x_{l_K})) \\
& \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i \ldots c_j \ldots \ldots (f_i\ x_1 \ldots x_{l_i}) \ldots))
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\ & (\delta_z : P\ Z) \\ & (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n), \\ & \qquad \cdot \\ & \qquad \cdot \\ & \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f, x_1 \ldots x_l) \ldots))\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\ & (\delta_z : P\ Z) \\ & (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n), \\ & \\ & \\ & P \qquad (\mathit{intro}_i \ \ldots c_j \ldots \ldots (f, x_1 \ldots x_\ell) \ldots))\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\ & (\delta_z : P\ Z) \\ & (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\end{aligned}$$

$$P \qquad (\mathit{intro}_i \qquad \ldots (f, x_1 .. x_k) \ldots))$$

Elimination for the Natural Numbers

$$\begin{aligned} \mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\ & (\delta_z : P\ Z) \\ & (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \end{aligned}$$

Elimination for the Natural Numbers

$$
\begin{aligned}
\mathrm{elim}: \forall\ & (P : \mathbb{N} \to \mathrm{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \\
& (b_1 : \beta_1) \ldots (b_m : \beta_m) \\
& (x : T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m), \\
& P\ b_1 \ldots b_m\ x
\end{aligned}
$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathit{elim}: \forall\ & (P : \mathbb{N} \to \mathit{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \\
& \\
& (x : T\ \alpha_1 \ldots \alpha_n) \\
& \\
P \qquad & x
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}
&\mathrm{elim}: \forall\ (P : \mathbb{N} \to \mathrm{Type}) \\
&\qquad (\delta_z : P\ Z) \\
&\qquad (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \\
&\qquad \\
&\qquad (x : T \qquad\qquad),\ \\
&P \qquad\quad x
\end{aligned}$$

Elimination for the Natural Numbers

$$\begin{aligned}
\mathrm{elim}: \forall\ & (P : \mathbb{N} \to \mathrm{Type}) \\
& (\delta_z : P\ Z) \\
& (\delta_s : \forall\ (n : \mathbb{N})\ (H : P\ n),\ P\ (S\ n)) \\
& (x : \mathbb{N}), \\
& P\ x
\end{aligned}$$

Elimination for the Natural Numbers

$$\overline{\mathsf{elim}}\ \mathsf{Z} = \delta_{\mathsf{Z}}$$

$$\overline{\mathsf{elim}}\ (\mathsf{S}\ n) = \delta_{\mathsf{s}}\ n\ (\overline{\mathsf{elim}}\ n)$$

Recall: Lists

```
inductive list (α : Type) : Type where
  | nil : list α
  | cons : α → list α → list α
```

Elimination for Lists

$$\begin{aligned}\mathit{elim}: \forall\ & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\ & (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\end{aligned}$$

Elimination for Lists

$$\begin{aligned} \mathsf{elim}\colon \forall\, (\alpha : \mathsf{Type}) \\ (P : \mathsf{list}\ \alpha \to \mathsf{Type}) \end{aligned}$$

Elimination for Lists

$$\begin{aligned}\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\ & (P : \mathit{list}\ \alpha \to \mathit{Type}) \\ & (\delta_{nil} : P\ nil)\end{aligned}$$

Elimination for Lists

$$
\begin{aligned}
\mathsf{elim}\colon \forall\ & (\alpha : \mathsf{Type}) \\
& (P : \mathsf{list}\ \alpha \to \mathsf{Type}) \\
& (\delta_{\mathsf{nil}} : P\ \mathsf{nil}) \\
& (\delta_{\mathsf{cons}} : \forall\ \ldots (c_j : \gamma_j) \ldots \\
& \qquad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\,\alpha_1 \ldots \alpha_n\, p_1 \ldots p_n) \ldots \\
& \qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K})) \ldots, \\
& \quad P\ q_1 \ldots q_m\ (\mathsf{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{\ell_K}) \ldots))
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
&\mathit{elim}: \forall\,(\alpha : \mathit{Type})\\
&\qquad (P : \mathit{list}\ \alpha \to \mathit{Type})\\
&\qquad (\delta_{nil} : P\ \mathit{nil})\\
&\qquad (\delta_{cons} : \forall\,(x : \alpha)\\
&\qquad\qquad \ldots\ (f_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots\\
&\qquad\qquad \ldots\ (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots,\\
&\qquad\qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots))
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \mathit{list}\ \alpha \to \mathit{Type}) \\
& (\delta_{nil} : P\ nil) \\
& (\delta_{cons} : \forall\ (x : \alpha) \\
& \qquad (f : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \\
& \qquad (H : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ P\ p_1 \ldots p_m\ (f\ x_1 \ldots x_\ell)) \\
& \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i \ldots c_j \ldots \ldots (f\ x_1 \ldots x_\ell) \ldots))),
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \mathit{list}\ \alpha \to \mathit{Type}) \\
& (\delta_{nil} : P\ \mathit{nil}) \\
& (\delta_{cons} : \forall\ (x : \alpha) \\
& \qquad (f : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ \mathit{list}\ \alpha) \\
& \qquad (H : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ P\ (f\ x_1 \ldots x_\ell)), \\
& \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f\ x_1 \ldots x_\ell) \ldots))
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
&\mathit{elim}: \forall\ (\alpha : \mathit{Type}) \\
&\qquad (P : \mathit{list}\ \alpha \to \mathit{Type}) \\
&\qquad (\delta_{nil} : P\ \mathit{nil}) \\
&\qquad (\delta_{cons} : \forall\ (x : \alpha)(xs : \mathit{list}\ \alpha)\ (H : P\ xs), \\
&\qquad\qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f\ x_1 \ldots x_l) \ldots))
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \mathit{list}\ \alpha \to \mathit{Type}) \\
& (\delta_{nil} : P\ \mathit{nil}) \\
& (\delta_{cons} : \forall\ (x : \alpha)(xs : \mathit{list}\ \alpha)\ (H : P\ xs), \\
& \qquad P\ (\mathit{cons}\ x\ xs))
\end{aligned}
$$

Elimination for Lists

$$
\begin{aligned}
&\mathit{elim}: \forall\ (\alpha : \mathit{Type}) \\
&\qquad (P : \mathit{list}\ \alpha \to \mathit{Type}) \\
&\qquad (\delta_{nil} : P\ \mathit{nil}) \\
&\qquad (\delta_{cons} : \forall\ (x : \alpha)(xs : \mathit{list}\ \alpha)\ (H : P\ xs), \\
&\qquad\qquad P\ (\mathit{cons}\ x\ xs)) \\
&\qquad (b_1 : \beta_1) \ldots (b_m : \beta_m) \\
&\qquad (x : T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m), \\
&\quad P\ b_1 \ldots b_m\ x
\end{aligned}
$$

Elimination for Lists

```
elim: ∀ (α : Type)
        (P : list α → Type)
        (δ_nil : P nil)
        (δ_cons : ∀ (x : α) (xs : list α) (H : P xs),
                    P (cons x xs))
        (x : list α),
      P x
```

Elimination for Lists

$$\overline{elim}\ nil = \delta_{nil}$$

$$\overline{elim}\ (cons\ x\ xs) = \delta_{cons}\ x\ xs\ (\overline{elim}\ xs)$$

Recall: Vectors

```
inductive vec (α : Type) : ℕ → Type where
  | nil : vec α 0
  | cons : ∀ (n : ℕ), α → vec α n → vec α (n+1)
```

Elimination for Vectors

$$\begin{aligned}\mathit{elim}: \forall\ & (\alpha_1 : A_1) \ldots (\alpha_n : A_n) \\ & (P : \forall (b_1 : \beta_1) \ldots (b_m : \beta_m),\ T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m \to \mathit{Type})\end{aligned}$$

Elimination for Vectors

$$\begin{aligned}\mathit{elim}: \forall\, (\alpha : \mathit{Type})\\ (P : \forall (n : \mathbb{N}),\ \mathit{vec}\ \alpha\ n \to \mathit{Type})\end{aligned}$$

Elimination for Vectors

$$\begin{aligned}\mathsf{elim}: \forall\, &(\alpha : \mathsf{Type}) \\ &(P : \forall (n : \mathbb{N}).\ \mathsf{vec}\ \alpha\ n \to \mathsf{Type}) \\ &(\delta_{\mathsf{nil}} : P\ 0\ \mathsf{nil})\end{aligned}$$

Elimination for Vectors

$$
\begin{aligned}
\mathit{elim}: \forall\, & (\alpha : \mathit{Type}) \\
& (P : \forall (n : \mathbb{N}),\ \mathit{vec}\ \alpha\ n \to \mathit{Type}) \\
& (\delta_{nil} : P\ 0\ \mathit{nil}) \\
& (\delta_{cons} : \forall \ldots (c_j : \gamma_j) \ldots \\
& \qquad \ldots (f_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ T\ \alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
& \qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots, \\
& \quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots))
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
\mathit{elim}\colon\ & \forall\,(\alpha : \mathit{Type}) \\
& (P\colon \forall (n:\mathbb{N}).\ \mathit{vec}\ \alpha\ n \to \mathit{Type}) \\
& (\delta_{nil} : P\ 0\ \mathit{nil}) \\
& (\delta_{cons}\colon \forall\,(n:\mathbb{N})(x:\alpha) \\
& \qquad \ldots\ (f_K : \forall (x_1:\xi_1)\ldots(x_{l_K}:\xi_{l_K}),\ T\,\alpha_1\ldots\alpha_n\ p_1\ldots p_n)\ldots, \\
& \qquad \ldots\ (H_K : \forall (x_1:\xi_1)\ldots(x_{l_K}:\xi_{l_K}),\ P\ p_1\ldots p_m\ (f_K\ x_1\ldots x_{l_K}))\ldots, \\
& \qquad P\ q_1\ldots q_m\ (\mathit{intro}_i\ \ldots c_j\ldots\ \ldots(f_K\ x_1\ldots x_{l_K})\ldots))
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \forall (n : \mathbb{N}),\ \mathit{vec}\ \alpha\ n \to \mathit{Type}) \\
& (\delta_{nil} : P\ 0\ \mathit{nil}) \\
& (\delta_{cons} : \forall\ (n : \mathbb{N})(x : \alpha) \\
& \qquad (xs : \mathit{vec}\ \alpha\ n) \\
& \qquad \ldots (H_K : \forall (x_1 : \xi_1) \ldots (x_{l_K} : \xi_{l_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{l_K})) \ldots, \\
& \qquad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots))
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \forall (n : \mathbb{N}),\ \mathit{vec}\ \alpha\ n \to \mathit{Type}) \\
& (\delta_{nil} : P\ 0\ \mathit{nil}) \\
& (\delta_{cons} : \forall\ (n : \mathbb{N})(x : \alpha) \\
& \qquad (xs : \mathit{vec}\ \alpha\ n) \\
& \qquad (H : P\ n\ xs), \\
& \quad P\ q_1 \ldots q_m\ (\mathit{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{l_K}) \ldots))
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
\mathsf{elim}: \forall\, & (\alpha : \mathsf{Type}) \\
& (P : \forall (n : \mathbb{N}),\ \mathsf{vec}\ \alpha\ n \to \mathsf{Type}) \\
& (\delta_{\mathsf{nil}} : P\ 0\ \mathsf{nil}) \\
& (\delta_{\mathsf{cons}} : \forall\, (n : \mathbb{N})(x : \alpha)(xs : \mathsf{vec}\ \alpha\ n)(H : P\ n\ xs), \\
& \qquad P\ (n+1)\ (\mathsf{cons}\ n\ x\ xs))
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
\mathit{elim}: \forall\ & (\alpha : \mathit{Type}) \\
& (P : \forall (n : \mathbb{N}),\ \mathit{vec}\ \alpha\ n \to \mathit{Type}) \\
& (\delta_{nil} : P\ 0\ nil) \\
& (\delta_{cons} : \forall\ (n : \mathbb{N})(x : \alpha)(xs : \mathit{vec}\ \alpha\ n)(H : P\ n\ xs), \\
& \qquad P\ (n+1)\ (cons\ n\ x\ xs)) \\
& (b_1 : \beta_1) \ldots (b_m : \beta_m) \\
& (x : T\ \alpha_1 \ldots \alpha_n\ b_1 \ldots b_m), \\
& P\ b_1 \ldots b_m\ x
\end{aligned}
$$

Elimination for Vectors

$$
\begin{aligned}
&\mathsf{elim}: \forall\, (\alpha : \mathsf{Type}) \\
&\qquad (P : \forall (n:\mathbb{N}),\ \mathsf{vec}\ \alpha\ n \to \mathsf{Type}) \\
&\qquad (\delta_{nil} : P\ 0\ \mathsf{nil}) \\
&\qquad (\delta_{cons} : \forall\, (n:\mathbb{N})(x:\alpha)(xs : \mathsf{vec}\ \alpha\ n)(H : P\ n\ xs), \\
&\qquad\qquad P\ (n+1)\ (\mathsf{cons}\ n\ x\ xs)) \\
&\qquad (n:\mathbb{N}) \\
&\qquad (x : \mathsf{vec}\ \alpha\ n), \\
&\quad P\ n\ x
\end{aligned}
$$

Elimination for Vectors

$$\overline{elim}\ nil = \delta_{nil}$$

$$\overline{elim}\ (cons\ n\ x\ xs) = \delta_{cons}\ n\ x\ xs\ (\overline{elim}\ xs)$$

Recall: Binary Trees

```
inductive tree (α : Type) : Type where
  | leaf : tree α
  | node : α → tree α → tree α → tree α
```

Elimination for Binary Trees

$$\begin{aligned}\mathit{elim}: \forall\, &(\alpha : \mathit{Type}) \\ &(P : \mathit{tree}_\infty\ \alpha \to \mathit{Type}) \\ &(\delta_{\mathit{leaf}} : P\ \mathit{leaf})\end{aligned}$$

Elimination for Binary Trees

$$
\begin{aligned}
&\mathsf{elim}: \forall\, (\alpha : \mathsf{Type}) \\
&\quad (P : \mathsf{tree}_\infty\ \alpha \to \mathsf{Type}) \\
&\quad (\delta_{\mathsf{leaf}} : P\ \mathsf{leaf}) \\
&\quad (\delta_i : \forall\ \ldots (c_j : \gamma_j) \ldots \\
&\qquad \ldots\ (f_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \ldots \\
&\qquad \ldots\ (H_K : \forall (x_1 : \xi_1) \ldots (x_{\ell_K} : \xi_{\ell_K}),\ P\ p_1 \ldots p_m\ (f_K\ x_1 \ldots x_{\ell_K})) \ldots, \\
&\qquad\quad P\ q_1 \ldots q_m\ (\mathsf{intro}_i\ \ldots c_j \ldots \ldots (f_K\ x_1 \ldots x_{\ell_K}) \ldots)) \\
&\quad \vdots \\
&\quad (b_1 : \beta_1) \ldots (b_m : \beta_m) \\
&\quad (x : T\,\alpha_1 \ldots \alpha_n\ b_1 \ldots b_m), \\
&P\ b_1 \ldots b_m\ x
\end{aligned}
$$

Elimination for Binary Trees

$$
\begin{aligned}
&\mathsf{elim}: \forall\, (\alpha : \mathsf{Type}) \\
&\qquad (P : \mathsf{tree}_\infty\ \alpha \to \mathsf{Type}) \\
&\qquad (\delta_{\mathsf{leaf}} : P\ \mathsf{leaf}) \\
&\qquad (\delta_{\mathsf{node}} : \forall\, (f : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ T\,\alpha_1 \ldots \alpha_n\ p_1 \ldots p_n) \\
&\qquad\qquad (H : \forall (x_1 : \xi_1) \ldots (x_\ell : \xi_\ell),\ P\ p_1 \ldots p_m\ (f\ x_1 \ldots x_\ell)), \\
&\qquad\qquad P\,(\mathsf{node}\,(f\ x_1 \ldots x_{\ell_1}))) \\
&\qquad\quad \vdots \\
&\qquad (x : \mathsf{tree}_\infty\ \alpha), \\
&\quad P\ x
\end{aligned}
$$

Elimination for Binary Trees

$$
\begin{aligned}
&\mathsf{elim}\colon \forall\,(\alpha : \mathsf{Type})\\
&\qquad (P\colon \mathsf{tree}_\infty\ \alpha \to \mathsf{Type})\\
&\qquad (\delta_{\mathsf{leaf}}\colon P\ \mathsf{leaf})\\
&\qquad (\delta_{\mathsf{node}}\colon \forall\,(f : \forall\,(n:\mathbb{N}),\ \mathsf{tree}_\infty\ \alpha)\\
&\qquad\qquad (H : \forall\,(n:\mathbb{N}),\ P\,(f\,n))\,,\\
&\qquad\qquad P\,(\mathsf{node}\ f))\\
&\qquad\quad \vdots\\
&\qquad (x\colon \mathsf{tree}_\infty\ \alpha)\,.\\
&\quad P\ x
\end{aligned}
$$

Elimination for Binary Trees

$$
\begin{aligned}
&\mathrm{elim}: \forall\, (\alpha : \mathrm{Type}) \\
&\qquad (P : \mathrm{tree}_\infty\ \alpha \to \mathrm{Type}) \\
&\qquad (\delta_{\mathrm{leaf}} : P\ \mathrm{leaf}) \\
&\qquad (\delta_{\mathrm{node}} : \forall\, (f : \mathbb{N} \to \mathrm{tree}_\infty\ \alpha)\ (H : \forall (n:\mathbb{N}),\ P\ (f\ n)), \\
&\qquad\qquad P\ (\mathrm{node}\ f)) \\
&\qquad (x : \mathrm{tree}_\infty\ \alpha), \\
&\quad P\ x
\end{aligned}
$$

Elimination for Binary Trees

$$\overline{elim}\ leaf = \delta_{leaf}$$

$$\overline{elim}\ (node\ f) = \delta_{node}\ f\ (\lambda n.\ \overline{elim}\ (f\ n))$$

Outline

Extensions of Inductive Types

Throughout the times, various extended and more general schemes of inductive types have been developed. Among those are

- **Mutual inductive types**: define multiple inductive types simultaneously
- **Inductive-recursive types**: define an inductive type T and a recursive function $f : T \to A$ simultaneously
- **Higher inductive types**: define a type by specifying constructors and equalities

Mutual Inductive Types

```
inductive isEven : ℕ → Type where
  | evenZ : isEven 0
  | oddS : ∀ (n:ℕ), isOdd n → isEven (n+1)
with isOdd : ℕ → Type where
  | evenS : ∀ (n:ℕ), isEven n → isOdd (n+1)
```

Mutual Inductive Types

Mutual inductive types can be reduced to inductive types.

```
inductive parity : ℕ → bool → Type where
| evenZ : parity 0 true
| oddS : ∀ (n:ℕ), parity n false → parity (n+1) true
| evenS : ∀ (n:ℕ), parity n true → parity (n+1) false
```

Inductive-Recursive Types

- Inductive-recursive types are often used to inductively define universe types
- In general, they can**not** be reduced to inductive types
- They increase the proof theoretic strength of the system

Inductive-Recursive Types

$$
\begin{aligned}
&\mathsf{inductive}\ \mathsf{dList} : \mathsf{Type}\ \mathsf{where}\\
&\quad \mid \mathsf{nil} : \mathsf{dList}\\
&\quad \mid \mathsf{cons} : \forall\, (n : \mathbb{N})\, (xs : \mathsf{dList}),\ \mathsf{fresh}\ x\ xs \to \mathsf{dList}\\
&\mathsf{with}\ \mathsf{fresh} : \mathsf{dList} \to \mathbb{N} \to \mathsf{Type}\\
&\qquad \mathsf{fresh}\ \ \mathsf{nil} \qquad\qquad\qquad\ \ m = \bot\\
&\qquad \mathsf{fresh}\ \ (\mathsf{cons}\ n\ xs\ p)\ \ m = n \neq m \wedge \mathsf{fresh}\ xs\ m
\end{aligned}
$$

References for Inductive-Recursive Types

- "A general formulation of simultaneous inductive-recursive definitions in type theory" by Peter Dybjer
- "A finite axiomatization of inductive-recursive definitions" by Peter Dybjer and Anton Setzer
- "The extended predicative Mahlo universe in Martin-Löf type theory" by Peter Dybjer and Anton Setzer
- "Small induction recursion" by Peter Hancock, Conor McBride, Neil Ghani, Lorenzo Malatesta, and Thorsten Altenkirch

Extensions: Higher Inductive Types

- Higher inductive types (HIT) combine inductive types with quotients
- Specifically, HITs are specified by constructors and equations
- Higher inductive types are especially popular in **homotopy type theory**

If you are interested, there is a MasterMath course on Homotopy Type Theory

Higher Inductive Types

```
inductive FinSet : Type where
  | ∅ : FinSet
  | add : ℕ → FinSet → FinSet
  | idem : ∀ (n:ℕ) (xs: FinSet),
             add n (add n xs) = add n xs
  | swap : ∀ (n m :ℕ) (xs: FinSet),
             add n (add m xs) = add m (add n xs)
```

Higher Inductive Types

```
inductive FinSet : Type where
  | ∅ : FinSet
  | add : ℕ → FinSet → FinSet
  | idem : ∀ (n:ℕ) (xs: FinSet),
          add n (add n xs) = add n xs
  | swap : ∀ (n m :ℕ) (xs: FinSet),
          add n (add m xs) = add m (add n xs)
  | trunc : ∀ (xs ys : FinSet) (p q : xs = ys),
          p = q
```

References for HITs

- "Homotopy type theory: Univalent foundations of mathematics"
- "Higher Inductive Types as Homotopy-Initial Algebras" by Kristina Sojakova
- "Finitary Higher Inductive Types in the Groupoid Model" by Hugo Moeneclaey and Peter Dybjer
- "Constructing Higher Inductive Types" by me
- "Impredicative encodings of (higher) inductive types" by Steve Awodey, Jonas Frey, Sam Speight

More Extensions of Inductive Types

And there is more...

- **Indexed inductive-recursive types**: allow indices in inductive-recursive types
- **Inductive-inductive types**: define an inductive type T together with an inductive predicate P on T
- **Higher inductive-recursive types**: combine higher inductive types and inductive-recursive types

Outline

Conclusion

- We saw a scheme for inductive types
- Inductive types are specified by parameters, indices, and their constructors
- Constructors are specified by their recursive and non-recursive premises
- From this description, one can derive an elimination rule