Representation of Partial Recursive Functions by Inductive-Recursive and by Inductive Definitions

Anton Setzer

Swansea University

Principle of Ind.-Rec. Defs.

- Developed by P. Dybjer.
- **Prime example: Universes**
  - Inductively define

$$\mathrm{U} : \mathrm{Set}$$

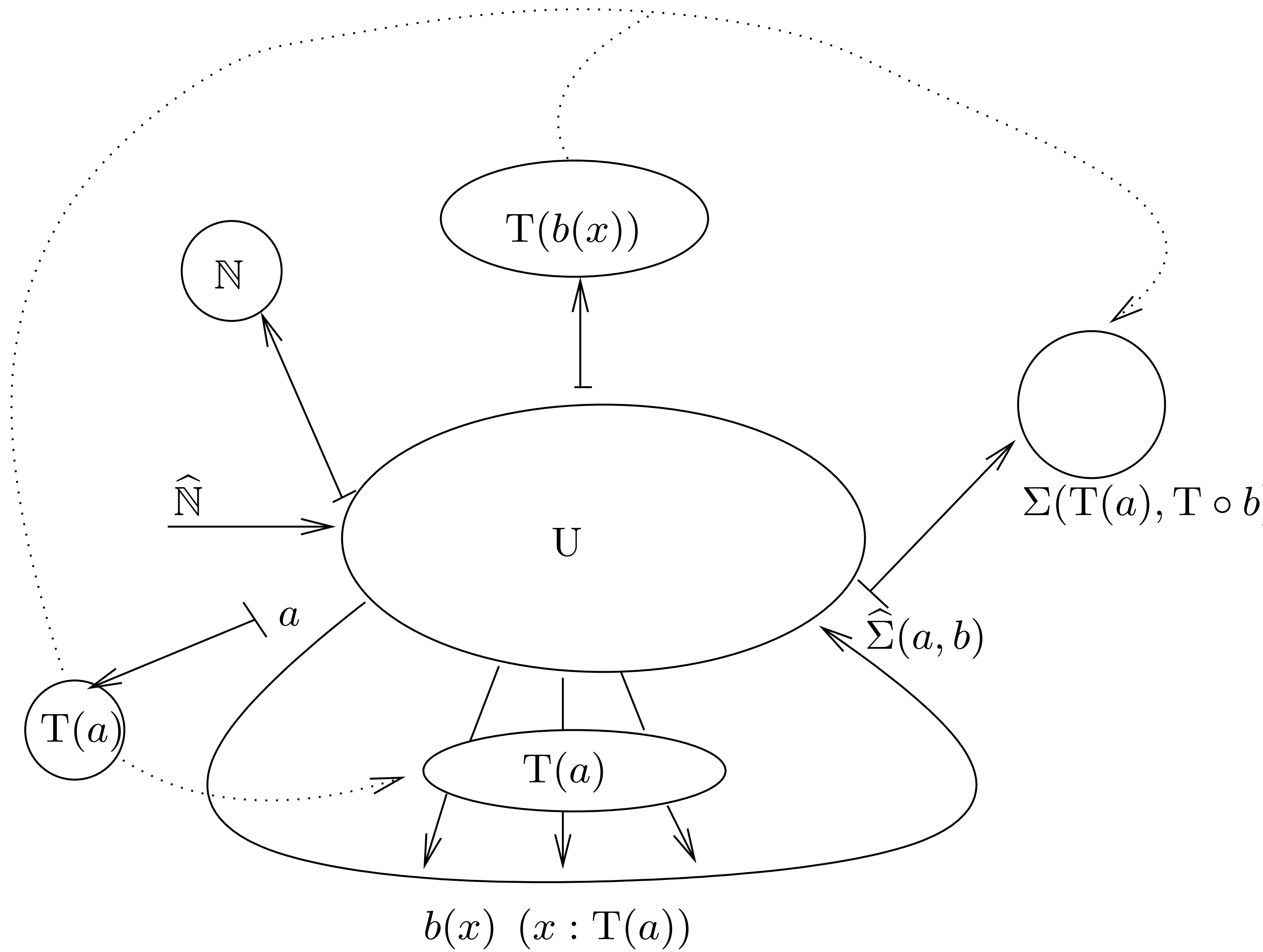
  - while simultaneously recursively defining

$$\mathrm{T}(u) : \mathrm{Set} \qquad (u : \mathrm{U})$$

  So $\mathrm{T} : \mathrm{U} \to \mathrm{Set}$.
- Generalization:
  - $\mathrm{T} : \mathrm{U} \to D$ for some arbitary type D.
  - Indexed ind.-rec. definitions:

$$\mathrm{U} : I \to \mathrm{Set} \qquad \mathrm{T} : (i : I, \mathrm{U}(i)) \to D[i]$$

Example



$\mathrm{T}(b(x))$
$\mathbb{N}$
$\widehat{\mathbb{N}}$
U
$\Sigma(\mathrm{T}(a), \mathrm{T} \circ b)$
a
$\widehat{\Sigma}(a, b)$
$\mathrm{T}(a)$
$\mathrm{T}(a)$
$b(x) \ \ (x : \mathrm{T}(a))$

Bove/Capretta Appr. to Par.-Rec. Fu

- **Example**

$$f : \mathbb{N} \rightharpoonup \mathbb{N}$$
$$f(0) = 0 \qquad f(n+1) = f(f(n))$$

- Represented by the following indexed ind.-rec. def.

$$\begin{array}{rcl} \mathrm{f}(\cdot)\downarrow & : & \mathbb{N} \to \mathrm{Set} \\ \mathrm{eval} & : & (n : \mathbb{N}, \mathrm{f}(n)\downarrow) \to \mathbb{N} \\ \\ \mathrm{f}(0)\downarrow & = & \mathrm{data\ true} \\ \mathrm{eval}(0, \mathrm{true}) & = & 0 \\ \\ \mathrm{f}(n+1)\downarrow & = & \mathrm{data\ C}\ (p : \mathrm{f}(n)\downarrow, q : \mathrm{f}(\mathrm{eval}(n,p))\downarrow) \\ \mathrm{eval}(n+1, \mathrm{C}(p,q)) & = & \mathrm{eval}(\mathrm{eval}(n,p), q) \end{array}$$

Standard Appr. to Part.-Rec. Func

$f(0) = 0 \qquad\qquad f(n+1) = f(f(n))$

- Standard approach to representing a part.-rec. funct.:
  - Define by an **ordinary indexed inductive definition**

$$\mathrm{Graph}_{\mathrm{f}} : \mathbb{N} \to \mathbb{N} \to \mathrm{Set}$$

  - In the example we have:

$$\begin{array}{lll} \mathrm{C}_0 & : & \mathrm{Graph}_{\mathrm{f}}(0,0) \\ \mathrm{C}_{\mathrm{S}} & : & (n : \mathbb{N}, m : \mathbb{N}, p : \mathrm{Graph}_{\mathrm{f}}(n,m), \\ & & \qquad\qquad k : \mathbb{N}, q : \mathrm{Graph}_{\mathrm{f}}(m,k)) \\ & & \to \mathrm{Graph}_{\mathrm{f}}(n+1,k) \end{array}$$

Standard Appr. to Part.-Rec. Func

$$f(0) = 0 \qquad f(n+1) = f(f(n))$$

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

$$\mathrm{C_0} : \mathrm{Graph_f}(0,0)$$

$$\mathrm{C_S} : (n : \mathbb{N}, m : \mathbb{N}, p : \mathrm{Graph_f}(n,m), \\ k : \mathbb{N}, q : \mathrm{Graph_f}(m,k)) \to \mathrm{Graph_f}(n+1,k)$$

- We can define $\mathrm{f}(\cdot)\downarrow$, eval as follows:

$$\begin{array}{rcl} \mathrm{f}(\cdot)\downarrow & : & \mathbb{N} \to \mathrm{Set} \\ \mathrm{f}(n)\downarrow & := & (m : \mathbb{N}) \times \mathrm{Graph_f}(n,m) \\ \mathrm{eval} & : & (n : \mathbb{N}, \mathrm{f}(n)\downarrow) \to \mathbb{N} \\ \mathrm{eval}(n, \langle m, p \rangle) & = & m \end{array}$$

Generalisation

- Assume a **small indexed ind.-rec. def.**

$$\begin{aligned} \mathrm{U} &: I \longrightarrow \mathrm{Set} \\ \mathrm{T} &: (i : I, \mathrm{U}(i)) \longrightarrow D(i) \end{aligned}$$

  where

$$D : I \longrightarrow \mathrm{Set}$$

- This can be simulated by an **indexed ind. def.**

$$\mathrm{Graph}_{\mathrm{T}} : (i : I, D(i)) \longrightarrow \mathrm{Set}$$

  Jump to conclusion.

Generalisation

$\mathrm{Graph}_{\mathrm{T}} : (i : I, D(i)) \to \mathrm{Set}$

- Now we can define

$$\begin{array}{rcl} \mathrm{U} & : & I \to \mathrm{Set} \\ \mathrm{U}(i) & := & (d : D(i)) \times \mathrm{Graph}_{\mathrm{T}}(i, d) \\ \mathrm{T} & : & (i : I, \mathrm{U}(i)) \to D(i) \\ \mathrm{T}(i, \langle d, p \rangle) & := & d \end{array}$$

- Simple case: U non-indexed, so $\mathrm{U} : \mathrm{Set}$, $\mathrm{T} : \mathrm{U} \to D$.
- Then we have

$$\mathrm{Graph}_{\mathrm{T}} : D \to \mathrm{Set}$$

Jump to conclusion.

Example

- Assume a single inductive argument (plus other constructors):

$$\begin{aligned} \mathrm{C} &: \mathrm{U} \to \mathrm{U} \\ \mathrm{T}(\mathrm{C}(u)) &= g(\mathrm{T}(u)) \end{aligned}$$

- Replace this by

$$\begin{aligned} \mathrm{Graph}_{\mathrm{T}} &: D \to \mathrm{Set} \\ \mathrm{C}' &: (d' : D, p : \mathrm{Graph}_{\mathrm{T}}(d')) \to \mathrm{Graph}_{\mathrm{T}}(g(d')) \end{aligned}$$

Conclusion

- Reduction of **small** indexed ind.-rec. definitions to indexed inductive definition.
- Maybe reason why not many real world examples of ind.-rec. definitions have been found.
- Need to explore whether using small ind.-rec. definitions or ind. definitions is easier.

- **Propaganda:**
  - Talk about object-oriented programming in dependent type theory on **Thursday at 11:45 in TFP**
  - Talk about functional concets in C++ on **Thursday at 15:15 in TFP** (presented by U. Berger).