How Many Numbers Can a Lambda-Term Contain?

Paweł Parys

University of Warsaw

Goal: characterize all higher-order functions operating on natural numbers

definable in simply-typed λ-calculus

(for any reasonable representation of natural numbers)

e.g. $[n] = \lambda f.\lambda x.\ \underbrace{f\ (f\ (f\ \ldots\ (f}_{n}\ x)\ldots))$

Goal: characterize all higher-order functions operating on natural numbers

definable in simply-typed λ-calculus

Consider the function:

$$g(f) = n_1+f(n_2+f(n_3+f(\ldots+f(n_k)\ldots)))$$

(where $n_1, n_2, \ldots, n_k$ are some constants)

If we want to know precisely the result of g for each f, we need to remember all the numbers $n_1, n_2, \ldots, n_k$ (arbitrarily many numbers).

Goal: characterize all higher-order functions operating on natural numbers

definable in simply-typed λ-calculus

Consider the function:

$$g(f) = n_1+f(n_2+f(n_3+f(\ldots+f(n_k)\ldots)))$$

(where $n_1, n_2, \ldots, n_k$ are some constants)

If we want to know precisely the result of g for each f, we need to remember all the numbers $n_1, n_2, \ldots, n_k$ (arbitrarily many numbers).

But if we allow approximation of the result, up to some error...

… our function is equivalent to:

$$g'(f) = n_1+f(m) \quad \text{where } m = n_2+n_3+\ldots+n_k$$

(assuming that all $n_1, n_2, \ldots, n_k$ are positive)

For example, if $f(x)=2x$, then $g'(f)\leq g(f)\leq g'(f)\cdot 2^{g'(f)}$.
In fact, for each fixed f we can give a similar relationship between g'(f) and g(f) (not depending on the values used in g and g').

Main result

For each type there exist only finitely many "shapes" of functions of that type, and for each shape we need to remember a vector of natural numbers (constants) of a fixed length.

E.g. for type $(\mathbb{N}\to\mathbb{N})\to\mathbb{N}$ one of possible shapes is $g'(f) = n_1+f(m)$, containing two constants n_1, m.

Main result

For each type there exist only finitely many “shapes” of functions of that type, and for each shape we need to remember a vector of natural numbers (constants) of a fixed length.

E.g. for type $(\mathbb{N}\to\mathbb{N})\to\mathbb{N}$ one of possible shapes is $g'(f) = n_1+f(m)$, containing two constants n_1, m.

Another possible shape is $g''(f)=\underbrace{f(f(f(\ldots(f(}_{n}0))\ldots)))$, containing one constant n.

Here, the constant is not written explicitly.
Thus, to each function we just assign a shape (from a finite set), and a vector of natural numbers (of a fixed length).

Main result

For each type there exist only finitely many “shapes” of functions of that type, and for each shape we need to remember a vector of natural numbers (constants) of a fixed length.

E.g. for type $(\mathbb{N}\to\mathbb{N})\to\mathbb{N}$ one of possible shapes is $g'(f) = n_1+f(m)$, containing two constants n_1, m.

Compositionality:

- the shape of application F(G) is determined by shapes of F and G
- the vector for F(G) is obtained by applying a linear function applied to the vectors for F and G; the linear function depends only on the shapes of F and G

Approximation:

- for terms of type $\mathbb{N}$ the number x in the vector approximates the number y represented by the term: $x \leq H(y)$ and $y \leq H(x)$ (for a fixed function H)

“Counterexample”

Consider the function:

$$f(x) = \begin{cases} n_1 & \text{if } x=0 \\ n_2 & \text{if } x=1 \\ \ldots & \\ n_k & \text{if } x \geq k \end{cases}$$

This function cannot be represented in simply-typed λ-calculus: it may contain arbitrarily many independent numbers.

“Counterexample”

Consider the function:

$$f(x) = \begin{cases} n_1 & \text{if } x=0 \\ n_2 & \text{if } x=1 \\ \dots & \\ n_k & \text{if } x \geq k \end{cases}$$

This function cannot be represented in simply-typed λ-calculus: it may contain arbitrarily many independent numbers.

Already this function cannot be represented (it cannot be computed while knowing only approximation of x):

$$f(x) = \begin{cases} n & \text{if } x<k \\ m & \text{if } x \geq k \end{cases}$$

Thank you!

For each type there exist only finitely many “shapes” of functions of that type, and for each shape we need to remember a vector of natural numbers (constants) of a fixed length.

E.g. for type $(\mathbb{N}\to\mathbb{N})\to\mathbb{N}$ one of possible shapes is $g'(f) = n_1+f(m)$, containing two constants n_1, m.

Compositionality:

- the shape of application F(G) is determined by shapes of F and G
- the vector for F(G) is obtained by applying a linear function applied to the vectors for F and G; the linear function depends only on the shapes of F and G

Approximation:

- for terms of type $\mathbb{N}$ the number x in the vector approximates the number y represented by the term: $x\leq H(y)$ and $y\leq H(x)$ (for a fixed function H)