Inductive Types in System F

We have seen:

$$Nat = \Pi X.\ (X \to X) \to X \to X$$

$$List_A = \Pi X.\ (A \to X \to X) \to X \to X$$

Other Example:

Trees with elements of A in the leaves

$Tree_A$ has the introduction rules

$$\frac{a : A}{\text{leaf}\ a : Tree_A}$$

$$\frac{t_1 : Tree_A \qquad t_2 : Tree_A}{\text{node}\ t_1\ t_2 : Tree_A}$$

To determine the system F encoding, look at the iterator:

$\mathrm{Tree}_A \to X \to \mathrm{Tree}_A \to X \to X$

for recursion

$g : X \to X \to X \qquad h : A \to X$

$\mathrm{itTree}\ g\ h : \mathrm{Tree}_A \to X$

with the reduction rules:

$\mathrm{itTree}\ g\ h\ (\mathrm{leaf}\ a) \rightsquigarrow h\ a$

$\mathrm{itTree}\ g\ h\ (\mathrm{node}\ t_1\ t_2) \rightsquigarrow g\ (\mathrm{itTree}\ g\ h\ t_1)\ (\mathrm{itTree}\ g\ h\ t_2)$

Use this iteration rule as inspiration for system F definition

$\mathrm{Tree}_A = \Pi X.\ (X \to X \to X) \to (A \to X) \to X$

In this way iteration is just application:

$\mathrm{itTree}\ g\ h = \lambda t.\ t\ X\ g\ h$

The constructors are just application of the components

(one step of iteration)

$$\text{leaf } a = \Lambda X.\, \lambda g.\, \lambda h.\, h\, a$$

$$\text{node } t_1\, t_2 = \Lambda X.\, \lambda g.\, \lambda h.\, g\, (t_1\, X\, g\, h)\, (t_2\, X\, g\, h)$$

Exercise:

Verify the reduction rules

Generalization to all inductive types

How do we represent μF in system F?

for any strictly positive functor F

Again: get inspiration from the elimination rule

Introduction:

$$\frac{t : F\,\mu F}{\text{in}\ t : \mu F}$$

Elimination:

$$\frac{f : FX \to X}{\text{cata}\ f : \mu F \to X}$$

Reduction:

$$\text{cata}\ f\ (\text{in}\ t) \rightsquigarrow f\ (F\ (\text{cata}\ f)\ t)$$

Use the assumption of the elimination rule to construct the system F encoding

$$\mu F = \Pi X.\ (FX \to X) \to X$$

The constructor is simply instantiation with μF itself:

$$\text{in}\ t = \Lambda X.\ \lambda f.\ f\ \mu F\ t$$

The elimination rule is just application:

$$\text{cata}\ f = \lambda t.\ t\ X\ f$$

Exercise:

Verify the reduction rule

Intuitive idea:

We represent μF as the product of all F-algebras

An F-algebra is a pair $\langle X, f \rangle$ where

X is a type

f is a function

$FX \to X$

An "element" of the algebra is an $x : X$

μF is defined as the product of all Falgebras:

$$\mu F = \prod_{\langle X, f \rangle} X$$

$\langle X, f \rangle$ F-algebra

an element of μF is a function that chooses an element from every Falgebra.

Can we also do coinductive types?

NOT in system F as it is

We would need sum types (the dual of product types)

$$\nu F = \sum_{\langle X, f \rangle} X$$

$\langle X, f \rangle$ F-coalgebra

So νF is the sum/union of all coalgebras

An F-coalgebra is a pair

$\langle X, f \rangle$ where

X is a type

f is a function

$X \to FX$

An "element" of the coalgebra is an $x : X$

A pointed coalgebra is a triple $\langle X, f, x \rangle$

$$\nu F = \sum_{\langle X, f \rangle \text{ F-coalgebra}} X$$

is the set of all pointed coalgebras

Exercise:

Can you define ana and out and verify the reduction rule ?