Recursion

Piotr Polesiuk

May 7, 2014

Motivation

Natural numbers in untyped ς-calculus

$$
\begin{array}{lll}
zero & \triangleq & \\
 & & \begin{array}{llll}
[& iszero & = & \mathbf{true} \\
, & pred & = & \varsigma(x)x \\
, & succ & = & \varsigma(x)(x.iszero := \mathbf{false}).pred := x \\
] & & &
\end{array}
\end{array}
$$

Motivation

Natural numbers in untyped ς-calculus

$$
\begin{array}{lll}
\mathit{zero} & \triangleq & \\
& & \begin{array}{llll}
[& \mathit{iszero} & = & \mathbf{true} \\
, & \mathit{pred} & = & \varsigma(x)x \\
, & \mathit{succ} & = & \varsigma(x)(x.\mathit{iszero} := \mathbf{false}).\mathit{pred} := x \\
]
\end{array}
\end{array}
$$

The type of *zero* should have form:

$$
\begin{array}{lll}
\mathit{Nat} & \triangleq & \\
& & \begin{array}{llll}
[& \mathit{iszero} & : & \mathit{Bool} \\
, & \mathit{pred} & : & ? \\
, & \mathit{succ} & : & ? \\
]
\end{array}
\end{array}
$$

Motivation

Natural numbers in untyped ς-calculus

$$
\begin{array}{lll}
zero & \triangleq & \\
& & \begin{array}{llll}
[& iszero & = & \textbf{true} \\
, & pred & = & \varsigma(x)x \\
, & succ & = & \varsigma(x)(x.iszero := \textbf{false}).pred := x \\
]
\end{array}
\end{array}
$$

The type of *zero* should have form:

$$
\begin{array}{lll}
Nat & \triangleq & \\
& & \begin{array}{llll}
[& iszero & : & Bool \\
, & pred & : & \color{red}{Nat} \\
, & succ & : & \color{red}{Nat} \\
]
\end{array}
\end{array}
$$

Intuition

- We extend grammar of types

$$A \quad ::= \quad \ldots \mid \mu(X)A \mid X$$

- A recursive type $\mu(X)A$ is the unique solution of the equation

$$X = A$$

Intuition

- We extend grammar of types

$$A \quad ::= \quad \ldots \mid \mu(X)A \mid X$$

- A recursive type $\mu(X)A$ is the unique solution of the equation

$$X = A$$

- There are two different approaches to the recursive types called equi-recursive and iso-recursive types.

Equi-recursive Types

- Types $\mu(X)A$ and $A\{\!|X \leftarrow \mu(X)A|\!\}$ are syntactically different, but they are treated as equal
- More convenient for programmer
- More complicated metatheory

Equi-recursive Types

- Types $\mu(X)A$ and $A\{\!|X \leftarrow \mu(X)A|\!\}$ are syntactically different, but they are treated as equal
- More convenient for programmer
- More complicated metatheory
- Today we will not talk about equi-recursive types

Iso-recursive Types

- Types $\mu(X)A$ and $A\{\!|X \leftarrow \mu(X)A|\!\}$ are treated as isomorphic
- This isomorphism is given by two constructs on the level of terms

$$\begin{array}{c} \mu(X)A \\ \mathit{fold} \uparrow \quad \downarrow \mathit{unfold} \\ A\{\!|X \leftarrow \mu(X)A|\!\} \end{array}$$

Rules

Δ_X

$$\text{(Env X)}\quad \frac{E \vdash \diamond \quad X \notin dom(E)}{E, X \vdash \diamond}$$

$$\text{(Type X)}\quad \frac{E', X, E'' \vdash \diamond}{E', X, E'' \vdash X}$$

Rules

Δ_X

(Env X)
$$\frac{E \vdash \diamond \quad X \notin dom(E)}{E, X \vdash \diamond}$$

(Type X)
$$\frac{E', X, E'' \vdash \diamond}{E', X, E'' \vdash X}$$

Δ_μ

(Type Rec)
$$\frac{E, X \vdash A}{E \vdash \mu(X)A}$$

(Val Fold) (where $A \equiv \mu(X)B$)
$$\frac{E \vdash b \ : \ B\{\!|X \leftarrow A|\!\}}{E \vdash fold(A, b) \ : \ A}$$

(Val Unfold) (where $A \equiv \mu(X)B$)
$$\frac{E \vdash a \ : \ A}{E \vdash unfold(a) \ : \ B\{\!|X \leftarrow A|\!\}}$$

Calculi Definitions

$$
\begin{aligned}
Ob_{1\mu} &\triangleq Ob_1 \cup \Delta_X \cup \Delta_\mu \\
F_{1\mu} &\triangleq F_1 \cup \Delta_X \cup \Delta_\mu \\
FOb_{1\mu} &\triangleq FOb_1 \cup \Delta_X \cup \Delta_\mu
\end{aligned}
$$

Equational Theory

$\Delta_{=\mu}$

(Eq Fold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash b \leftrightarrow b' : B\{\!|X \leftarrow A|\!\}}{E \vdash fold(A, b) \leftrightarrow fold(A, b') : A}$$

(Eq Unfold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash a \leftrightarrow a' : A}{E \vdash unfold(a) \leftrightarrow unfold(a') : B\{\!|X \leftarrow A|\!\}}$$

Equational Theory

$\Delta_{=\mu}$

(Eq Fold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash b \leftrightarrow b' : B\{\!|X \leftarrow A|\!\}}{E \vdash fold(A, b) \leftrightarrow fold(A, b') : A}$$

(Eq Unfold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash a \leftrightarrow a' : A}{E \vdash unfold(a) \leftrightarrow unfold(a') : B\{\!|X \leftarrow A|\!\}}$$

(Eval Fold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash a : A}{E \vdash fold(A, unfold(a)) \leftrightarrow a : A}$$

(Eval Unfold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash b : B\{\!|X \leftarrow A|\!\}}{E \vdash unfold(fold(A, b)) \leftrightarrow b : B\{\!|X \leftarrow A|\!\}}$$

Expressive Power

- $F_{1\mu}$ is powerful enough to express untyped λ-calculus

$$Untyped \triangleq \mu(X)X \to X$$

$$\begin{array}{rcl} |x| & = & x \\ |a(b)| & = & unfold(|a|)(|b|) \\ |\lambda(x)a| & = & fold(Untyped, \lambda(x : Untyped)|a|) \end{array}$$

- This translation can be combined with translation from $F_{1\mu}$ to $Ob_{1\mu}$

Fixed-Point Operator

In $F_{1\mu}$ we can also define fixed-point operator

- In equi-recursive type system

$$\mu(x : A)b \triangleq (\lambda(x : B)b\{\!|x \leftarrow x(x)|\!\})(\lambda(x : B)b\{\!|x \leftarrow x(x)|\!\})$$

where $B \triangleq \mu(X)X \rightarrow A$

Fixed-Point Operator

In $F_{1\mu}$ we can also define fixed-point operator

- In equi-recursive type system

$$\mu(x : A)b \;\triangleq\; (\lambda(x : B)b\{\!|x \leftarrow x(x)|\!\})(\lambda(x : B)b\{\!|x \leftarrow x(x)|\!\})$$

where $B \triangleq \mu(X)X \to A$

- In iso-recursive type system

$$\mu(x : A)b \;\triangleq\; f(fold(\mu(X)X \to A, f))$$

where

$$f \;\triangleq\; \lambda(x : \mu(X)X \to A)b\{\!|x \leftarrow unfold(x)(x)|\!\}$$

Subtyping

$\Delta_{<:X}$

(Env X<:)

$$\frac{E \vdash A \quad X \notin dom(E)}{E, X <: A \vdash \diamond}$$

(Type X<:)

$$\frac{E', X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X}$$

(Sub X)

$$\frac{E', X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X \;<:\; A}$$

Subtyping

$\Delta_{<:\mu}$

(Type Rec<:)

$$\frac{E, X <: Top \vdash A}{E \vdash \mu(X)A}$$

(Sub Rec)

$$\frac{E \vdash \mu(X)A \quad E \vdash \mu(Y)B \quad E, Y <: Top, X <: Y \vdash A \ <: \ B}{E \vdash \mu(X)A \ <: \ \mu(Y)B}$$

Subtyping

$\Delta_{<:\mu}$

(Type Rec<:)

$$\frac{E, X <: Top \vdash A}{E \vdash \mu(X)A}$$

(Sub Rec)

$$\frac{E \vdash \mu(X)A \quad E \vdash \mu(Y)B \quad E, Y <: Top, X <: Y \vdash A \ <: \ B}{E \vdash \mu(X)A \ <: \ \mu(Y)B}$$

(Val Fold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash b \ : \ B\{\!|X \leftarrow A|\!\}}{E \vdash fold(A, b) \ : \ A}$$

(Val Unfold) (where $A \equiv \mu(X)B$)

$$\frac{E \vdash a \ : \ A}{E \vdash unfold(a) \ : \ B\{\!|X \leftarrow A|\!\}}$$

Calculi Definitions

$$
\begin{aligned}
Ob_{1<:\mu} &\triangleq Ob_{1<:} \cup \Delta_{<:X} \cup \Delta_{<:\mu} \\
F_{1<:\mu} &\triangleq F_{1<:} \cup \Delta_{<:X} \cup \Delta_{<:\mu} \\
FOb_{1<:\mu} &\triangleq FOb_{1<:} \cup \Delta_{<:X} \cup \Delta_{<:\mu}
\end{aligned}
$$

Operational semantics

$$\text{(Red Fold)}\quad \frac{\vdash a \;\leadsto\; v}{\vdash \mathit{fold}(A, a) \;\leadsto\; \mathit{fold}(A, v)} \qquad \text{(Red Unfold)}\quad \frac{\vdash a \;\leadsto\; \mathit{fold}(A, v)}{\vdash \mathit{unfold}(a) \;\leadsto\; v}$$

Properties of $Ob_{1<:\mu}$

Theorem (Minimum type)

If $\varnothing \vdash a : A$ then there exists B such that $\varnothing \vdash a : B$ and, for any A' if $\varnothing \vdash a : A'$ then $\varnothing \vdash B <: A'$.

Theorem (Subject reduction)

Let c be a closed term and v be a result, and assume $\vdash c \rightsquigarrow v$. If $\varnothing \vdash c : C$ then $\varnothing \vdash v : C$.

The Shortcomings of First-Order Typing

- Movable points

$$
\begin{aligned}
P_1 &\triangleq \mu(X)[x : \mathit{Int}, \mathit{mv\_x} : \mathit{Int} \to X] \\
P_2 &\triangleq \mu(X)[x, y : \mathit{Int}, \mathit{mv\_x}, \mathit{mv\_y} : \mathit{Int} \to X]
\end{aligned}
$$

- P_2 is not a subtype of P_1.

The Shortcomings of First-Order Typing

- Movable points

$$\begin{aligned} P_1 &\triangleq \mu(X)[x : \mathit{Int}, \mathit{mv\_x} : \mathit{Int} \rightarrow X] \\ P_2 &\triangleq \mu(X)[x, y : \mathit{Int}, \mathit{mv\_x}, \mathit{mv\_y} : \mathit{Int} \rightarrow X] \end{aligned}$$

- P_2 is not a subtype of P_1.
- Inclusion $P_2 \;<:\; P_1$ is unsound!

The Shortcomings of First-Order Typing

$$
\begin{array}{rcl}
P_1 & \triangleq & \mu(X)[x : Int, mv\_x : Int \to X] \\
P_2 & \triangleq & \mu(X)[x, y : Int, mv\_x, mv\_y : Int \to X] \\
p_2 : P_2 & \triangleq & [x = \varsigma(s_2)s_2.mv\_x(1).y \\
 & & , y = 0 \\
 & & , mv\_x = \varsigma(s_2)\lambda(dx)s_2 \\
 & & , mv\_y = \varsigma(s_2)\lambda(dy)s_2 \\
 & &] \\
p_1 : P_1 & \triangleq & [x = 0,\ mv\_x = \varsigma(s_1)\lambda(dx)s_1] \\
p : P_1 & \triangleq & p_2 \\
q : P_1 & \triangleq & p.mv\_x := \lambda(dx)p_1
\end{array}
$$

$q.x = p_1.y$

Questions?

Bonus

On blackboard