Polymorphism is not set-theoretic

John C. Reynolds

Mart Rietdijk, s1147901 Martijn van de Wouw, s1052099

Radboud University

November 18, 2025

Presentation Overview

Aim of the paper

- Polymorphic typed lambda calculus ($\lambda 2$)
- Set theoretic model cannot exist

Syntax

Assume

- an infinite countable set T of *type variables*
- an infinite countable set V of *ordinary variables*

Definition of Ω

Ω is the least set of *type expressions* such that:

- if τ in T, then $\tau \in \Omega$
- if $\omega, \omega' \in \Omega$, then $\omega \to \omega' \in \Omega$
- if $\tau \in T$ and $\omega \in \Omega$, then $\Delta\tau.\omega \in \Omega$

Free type variables

FTV

The set of free type variables FTV is defined as

- $\mathrm{FTV}(\tau) = \{\tau\}$
- $\mathrm{FTV}(\omega \to \omega') = \mathrm{FTV}(\omega) \cup \mathrm{FTV}(\omega')$
- $\mathrm{FTV}(\Delta\tau.\omega) = \mathrm{FTV}(\omega) - \{\tau\}$

Example

Let $T = \{\ldots, X, Y, Z, \ldots\}$, then

$$\begin{aligned} \mathrm{FTV}(X \to (\Delta Z.Y)) &= \mathrm{FTV}(X) \cup \mathrm{FTV}(\Delta Z.Y) \\ &= \{X\} \cup \mathrm{FTV}(Y) - \{Z\} \\ &= \{X, Y\} \end{aligned}$$

Alpha conversion and substitution

Renaming bound variables

We can rename bound variables like we are used to. We will regard alpha-variants of an expression as equal.

Substitution

We will write $(\omega'/\tau \to \omega)$ to denote the type expression obtained from ω' by substituting ω for every free occurrence of τ

Example substitution

Let $\omega' = X \to Y$, $\omega = Z$, then

$$\begin{aligned}(\omega'/\tau \to \omega) &= (X \to Y/X \to Z) \\ &= Z \to Y\end{aligned}$$

Type assignment

Definition π

The function π basically assigns type expressions to ordinary variables. We call π a *type assignment*.
$\pi \in F \to \Omega$ where F is a finite subset of V

Example

Let $F = \{x, y, z\} \subset V$,
$T = \{\ldots, X, Y, Z, \ldots\}$
and $\Omega = \{\ldots, X \to X, Y \to X, \Delta Z.Z \to X, \ldots\}$,
then we could define $\pi = \{x \mapsto X \to X, y \mapsto X, z \mapsto \Delta Z.Z \to X\}$.
And we get:
$\pi(x) = X \to X$.

Set of type assignments

Defintion Ω^*

We define Ω^* as the set of all type assingments (π)
$\Omega^* = \{\pi | \pi \in F \to \Omega \text{ for some finite } F \subset V\}$

Ω^* basically defines all the possible contexts of ordinary variables and their types.

Syntax of ordinary expressions

To define the syntax of ordinary expressions (or lambda expressions), we define the familiy of sets:

$$\langle E_{\pi\omega} | \pi \in \Omega^*, \omega \in \Omega \rangle$$

If we take a set $E_{\pi\omega}$ from this family of sets, we get the set of ordinary expressions:

- whose free variables are in the domain of π
- which takes on the type ω under the assignment of π to its variables.

Example

Let $\pi = \{f \mapsto (X \to X) \to (X \to X)\}$, and $\omega = X \to X$.
Then we have $(f(\lambda y : X.y)) \in E_{\pi\omega}$.

Syntax of ordinary expressions

Variable

If $v \in dom(\pi)$, then $v \in E_{\pi,\pi v}$

Application

If $e_1 \in E_{\pi,\omega\to\omega'}$ and $e_2 \in E_{\pi\omega}$, then $e_1(e_2) \in E_{\pi,\omega'}$

Abstraction

If $e \in E_{[\pi|v:\omega],\omega'}$ then $\lambda v : \omega.e \in E_{\pi,\omega\to\omega'}$

Type β-reduction

If $e \in E_{\pi,\Delta\tau.\omega'}$ then $e[\omega] \in E_{\pi,(\omega'/\tau\to\omega)}$

Polymorphism

If $e \in E_{\pi-\tau,\omega}$ then $\Lambda\tau.e \in E_{\pi,\Delta\tau.\omega}$

Set assignment

For the set-theoretic semantics, we assume that every type expressions denotes a single set.

Definition set assignment

A set assignment S is a function from T to the class of sets

Example

Let $S = \{\ldots, X \mapsto \mathbb{N}, Y \mapsto \mathbb{B}, Z \mapsto \mathbb{Z}, \ldots\}$
Then we have:
$S(X) = \mathbb{N}$

Set assignment

Note that S does not map every type expression. It maps only the type variables. We need to extend the definition of S to include the other expressions.

Definition $S^{\#}$

A set assignment $S^{\#}$ is a function from Ω to the class of sets. Such that $S^{\#}(\omega)$ is the set denoted by ω under assignment S.
We require that $S(\tau) = S^{\#}(\tau)$ for all $\tau \in T$

Example $S^{\#}$

Let $S = \{\ldots, X \mapsto \mathbb{N}, Y \mapsto \mathbb{B}, Z \mapsto \mathbb{Z}, \ldots\}$
Then we have:
$S^{\#}(X) = \mathbb{N}$, $S^{\#}(Y) = \mathbb{B}$ and $S^{\#}(Z) = \mathbb{Z}$
$S^{\#}(X \to Y) = S^{\#}(X) \to S^{\#}(Y) = \mathbb{N} \to \mathbb{B}$, i.e. the set of all functions from the natural numbers to booleans.

Set assignment

Finally, we define the set of all possible environments defined by π

Definition $S^{\#*}$

We define $S^{\#*}$ to be the function from Ω^* to the set of class assignments which maps each $\pi \in \Omega^*$ into the cartesian product over $v \in dom(\pi)$ of the sets $S^{\#}(\pi(v))$

Example $S^{\#*}\pi$

Let $\pi = \{x \mapsto X, y \mapsto Y\}$, $S = \{\ldots, X \mapsto \mathbb{N}, Y \mapsto \mathbb{B}, \ldots\}$, then:
$S^{\#*}\pi = \{(0, \mathit{true}), (0, \mathit{false}), (1, \mathit{true}), \ldots\}$

Semantic rules: 1.

The definitions of S, $S^{\#}$ and $S^{\#*}$ as given before.

Semantic rules: 2.

Definition μ

For every $\pi \in \Omega^*$ and $\omega \in \Omega$ we have:

$$\mu_{\pi\omega} \in E_{\pi\omega} \to \Pi_{S \in SA}(S^{\#*}\pi \to S^{\#}\omega)$$

You can read this as $\mu_{\pi\omega}$ is the family of semantic functions i.e. how to interpret expressions in $E_{\pi\omega}$. We can extract the semantic meaning by passing an expression $e \in E_{\pi\omega}$, a set assignment, and some $\eta \in S^{\#*}\pi$ to $\mu_{\pi\omega}$

Example μ

Let $\pi = \eta = [\,]$, $\omega = X \to X$, $e = \lambda x : X.x$ and $S = \{\ldots, X \mapsto \mathbb{N}, \ldots\}$. Then we have: $\mu_{\pi\omega}[\![e]\!]S\eta =$ the identity function on the $\mathbb{N}$

Semantic rules: 3.

The 'meaning' of a type ω should only depend on the free variables (FTV(ω)). More precisely:

If $S\tau = S'\tau$ forall $\tau \in \text{FTV}(\omega)$ then $S^{\#}\omega = S'^{\#}\omega$

Semantic rules: 4.

If two set assignment functions S and S' assign the same sets to all of the free type variables in our 'known universe', then we expect semantic interpretation using both functions to be the same on the same input.

Assume:

- $e \in E_{\pi\omega}$
- $\eta \in S^{\#*}\pi$

Then:
If we have $S\tau = S'\tau$ for all $\tau \in \mathrm{FTV}(e) \cup \mathrm{FTV}(\omega) \cup \bigcup_{v \in dom(\pi)} \mathrm{FTV}(\pi v)$, then we have $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi\omega}[\![e]\!]S'\eta$.

Semantic rules: 5.

Extra unused information should not change the semantic meaning of the function.

Assume:

- $e \in E_{\pi\omega}$
- π' extends π
- $\eta' \in S^{\#*}\pi'$ extends $\eta \in S^{\#*}\pi$

Then:

$\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi'\omega}[\![e]\!]S\eta'$

Semantic rules: 6.

$$S^{\#}(\omega \to \omega') = S^{\#}(\omega) \to S^{\#}(\omega')$$

Semantic rules: 7.

The semantic meaning of an ordinary variable is its value.

If we have $v \in dom(\pi)$, then we have $\mu_{\pi,\pi v}[\![v]\!]S\eta = \eta v$

Semantic rules: 8.

The semantic meaning of a function application is the same as the semantic meaning of one function applied to the semantic meaning of the other function.

Assume:

- $e_1 \in E_{\pi,\omega\to\omega'}$
- $e_2 \in E_{\pi\omega}$

Then:

$\mu_{\pi\omega'}[\![e_1(e_2)]\!]S\eta = \mu_{\pi,\omega\to\omega'}[\![e_1]\!]S\eta(\mu_{\pi\omega}[\![e_2]\!]S\eta)$

Semantic rules: 9.

The semantic meaning of a substitution of an ordinary variable in a lambda expression, has the same semantic meaning as the body of the lambda expression with the ordinary variable substituted for the argument of the lambda expression.

If $e \in E_{[\pi|v:\omega],\omega'}$
Then $\mu_{\pi,\omega\to\omega'}[\![\lambda v : \omega.e]\!]S\eta = \lambda x \in S^{\#}\omega.\mu_{[\pi|v:\omega],\omega'}[\![e]\!]S[\eta|v : x]$

Semantic rules: 10.

Given 3 expressions e_1, e_2, e_3, if e_3 is the result of substituting e_2 in to e_1, then e_3 has the same semantic meaning of the substitution of $(\lambda v : \omega.e_1)(e_2)$.

Assume:

- $e_1 \in E_{[\pi|v:\omega],\omega'}$
- $e_2 \in E_{\pi\omega}$
- $e_3 = (e_1/v \to e_2) \in E_{\pi\omega'}$

Then:

$\mu_{\pi\omega'}[\![(\lambda v : \omega.e_1)(e_2)]\!]S\eta = \mu_{\pi\omega'}[\![e_3]\!]S\eta$

Semantic rules: 11.

If two type abstracted (i.e. we have $\Delta\tau.\omega$ as the type) expressions which have the same functions defining the semantic meaning, then we expect the meaning of the expressions to remain the same when applying beta reduction on the type.

If we have $\mu_{\pi,\Delta\tau.\omega'}[\![e_1]\!] = \mu_{\pi,\Delta\tau.\omega'}[\![e_2]\!]$ (note the absence of some S and some η).
Then we have $\mu_{\pi,(\omega'/\tau\to\omega)}[\![e_1[\omega]]\!] = \mu_{\pi,(\omega'/\tau\to\omega)}[\![e_2[\omega]]\!]$

Semantic rules: 12.

If we have an expression e where τ does not occur as a free type variable in the range of π, then the semantic meaning of e with an additional type abstraction on which we apply τ is the same as the 'inside' of that function (so we have $(\Lambda\tau.e)[\tau]$ which we expect to equal e).

If we have $e \in E_{\pi-\tau,\omega}$
Then we have $\mu_{\pi-\tau,\omega}[\![(\Lambda\tau.e)[\tau]]\!] = \mu_{\pi-\tau,\omega}[\![e]\!]$

Lemma 1

We are going to prove that there exists a type expression **B** for which the set given by $S^{\#}\ \mathbf{B}$ has more than one element independent of S.

Lemma

If the polymorphic typed lambda calculus has a set theoretic model, then there exists a type expression **B** *such that* $FTV(\mathbf{B}) = \emptyset$*, and* $B = S^{\#}\ \mathbf{B}$ *contains more than one element.*

Lemma 1 continued

Note

B is independent of S by semantic rule 3.

Assume

Let $\mathbf{B} = \Delta\mathbf{s}.\mathbf{s} \to (\mathbf{s} \to \mathbf{s})$ and S be an assignment mapping $\mathbf{s}$ to some set s where s has more than one element.
Then B contains:

- $\mu_{[],\mathbf{B}}[\![\Lambda\mathbf{s}.\lambda\mathbf{x} : \mathbf{s}.\lambda\mathbf{y} : \mathbf{s}.\mathbf{x}]\!]S[]$
- $\mu_{[],\mathbf{B}}[\![\Lambda\mathbf{s}.\lambda\mathbf{x} : \mathbf{s}.\lambda\mathbf{y} : \mathbf{s}.\mathbf{y}]\!]S[]$

From here on out, you can interpret $\mathbf{B}$ as the definition of the boolean type.

Lemma 1 continued

let $X = \Lambda \mathbf{s}.\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{x}$ and let $Y = \Lambda \mathbf{s}.\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{y}$ We are going to prove this by contradiction.

Proof.

Assume that B has only one element. In other words, we have:

$$\mu_{[],\mathbf{B}}[\![X]\!]S[\,] = \mu_{[],\mathbf{B}}[\![Y]\!]S[\,]$$

$$\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![X[\mathbf{s}]]\!]S[\,] = \mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![Y[\mathbf{s}]]\!]S[\,] \quad (11)$$

$$\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{x}]\!]S[\,] = \mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{y}]\!]S[\,] \quad (12)$$

Then for all $x, y \in s$

We get:

- $\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{x}]\!]S[\,]xy = x$ (by 9 and 7)
- $\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda \mathbf{x} : \mathbf{s}.\lambda \mathbf{y} : \mathbf{s}.\mathbf{y}]\!]S[\,]xy = y$ (by 9 and 7)

But this implies that $x = y$ and thus that s only contains one element which contradicts our definition of s. Thus we have shown that B contains more than one element. □

Lemma 2: Define Functor

Definition of functor T

- $T(X) = (X \to B) \to B$ where X is a semantic domain and B is the boolean class.
- If $\rho \in s \to s'$ then $T(\rho) = \lambda h \in (s \to B) \to B.\lambda g \in s' \to B.h(g \circ \rho)$

So if $\rho \in s \to s'$, then $T\rho \in Ts \to Ts'$

Lemma 2: Define Functor T

Example of a Functor:

Let $B = \{true, false\}$, $s = \{0, 1\}$, $s' = \{a, b, c\}$

let $\rho : s \to s'$ with $\rho(0) = a, \rho(1) = c$

let $g : s \to B$ with $g(0) = true, g(1) = false$

let $h : (s \to B) \to B$ with $h(g) = g(0)$

That is, h looks at a predicate $g : s \to B$ and returns its value at 0.

let $g' = s' \to B$ with $g'(a) = false, g'(b) = true, g'(c) = true$

Then $T\rho(h)g' = h(g' \circ \rho)$

first calculate $g' \circ \rho$ for domain of s:
$(g' \circ \rho)(0) = g'(\rho(0)) = g'(a) = false$
$(g' \circ \rho)(1) = g'(\rho(1)) = g'(c) = true$

so now $T\rho(h)g' = h(g' \circ \rho) = (g' \circ \rho)(0) = false$

Lemma 2: Define T-algebra

Definition T-algebra

A T-algebra is a pair $\langle s, H \rangle$ with s is a semantic domain and $H \in Ts \to s$

Example of T-algebra:

$B = \{true, false\}$, $s = \{0, 1\}$

let $h : (s \to B) \to B$ with $h(g) = g(0)$

That is, h looks at a predicate $g : s \to B$ and returns its value at 0.

then $H(h) = 0$

Lemma 2

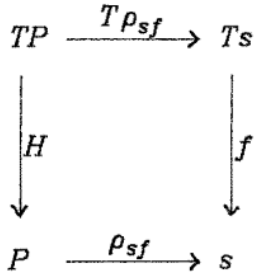


Figure: Morphism between 2 T-algebra's

we want to proof this property, or put otherwise $\rho_{sf} \circ H = f \circ T(\rho_{sf})$ with the following functions:

- $f \in Ts \to s$
- $H \in TP \to P$
- $\rho_{sf} \in P \to s$

Lemma 2: Defining H and P

We define the type expression:

- $\mathbf{W} = (((\mathbf{s} \to \mathbf{B}) \to \mathbf{B}) \to \mathbf{s}) \to \mathbf{s}$
- $\mathbf{P} = \Delta\mathbf{s}.\mathbf{W}$
- $P = S^{\#}\mathbf{P}$

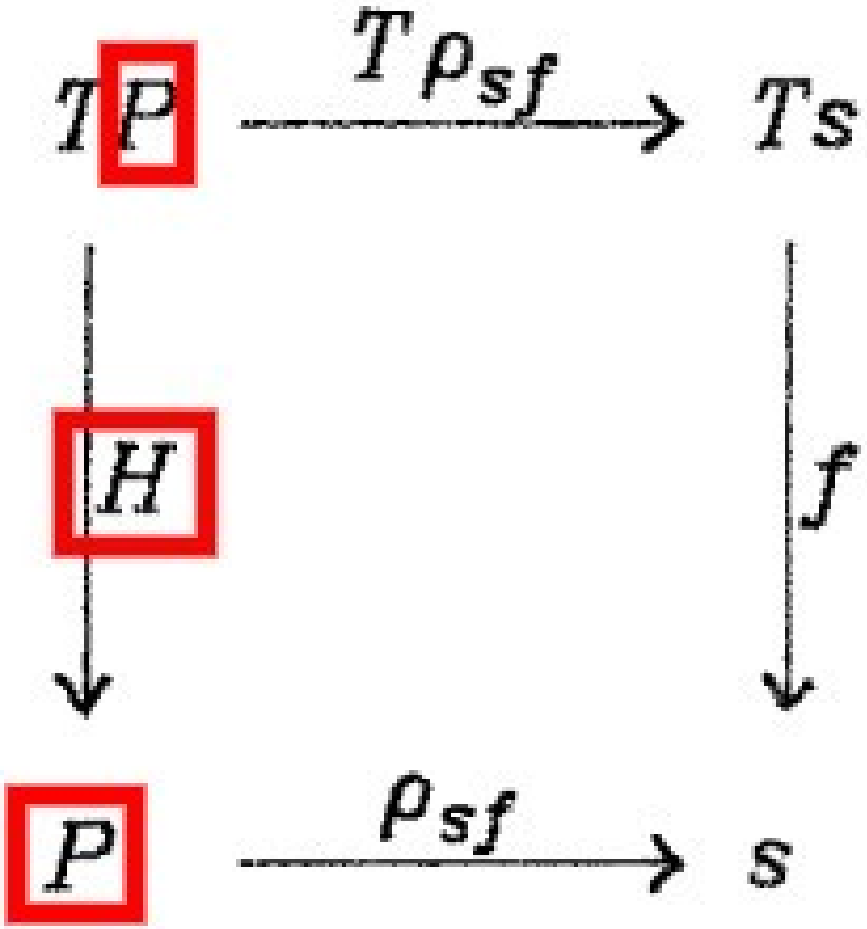
And we define the ordinary expressions:

- $\mathbf{M} = \lambda\mathbf{f} : ((\mathbf{s} \to \mathbf{B}) \to \mathbf{B}) \to \mathbf{s}.\mathbf{f}(\lambda\mathbf{g} : \mathbf{s} \to \mathbf{B}.\mathbf{h}(\lambda\mathbf{p} : \mathbf{P}.\mathbf{g}(\mathbf{p}[\mathbf{s}]\mathbf{f})))$
- $\mathbf{M} \in E_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}$
- $\mathbf{H} = \lambda\mathbf{h} : (\mathbf{P} \to \mathbf{B}) \to \mathbf{B}.\Lambda\mathbf{s}.\mathbf{M}$
- $\mathbf{H} \in E_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}$

it follows that:

- $\mu_{[],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S \in S^{\#}(\mathbf{P}) = P$
- $H = \mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S \in S^{\#}((\mathbf{P} \to \mathbf{B}) \to \mathbf{B}) \to \mathbf{P} = TP \to P$

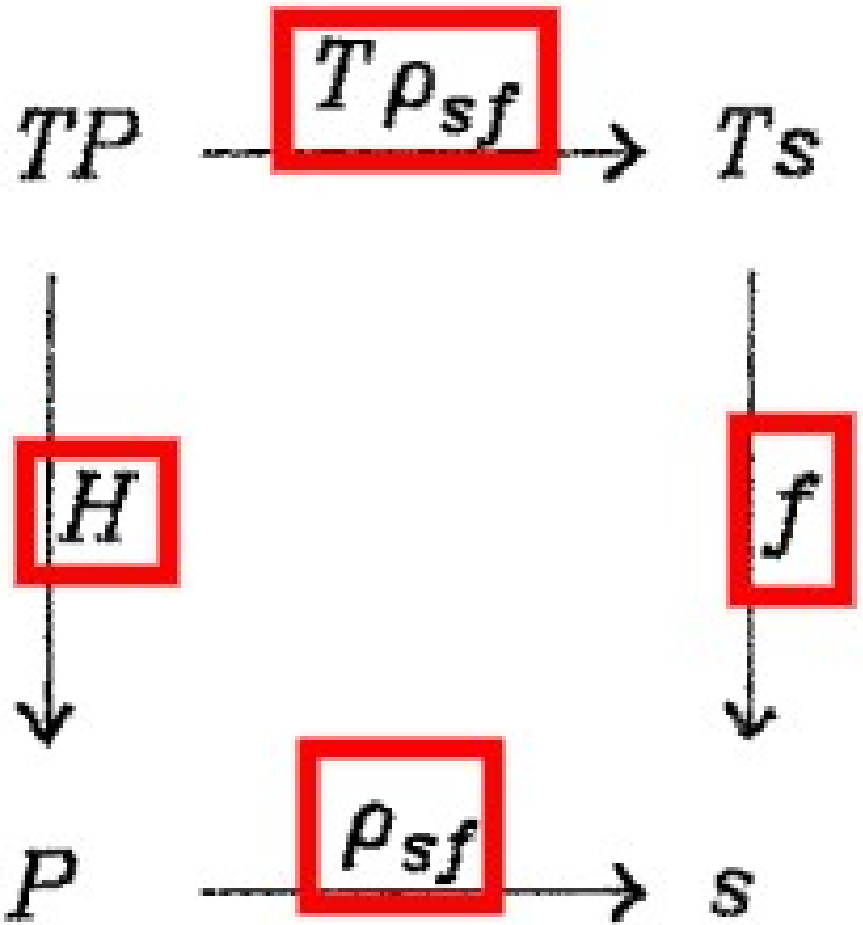
Lemma 2: Defining H and P



Lemma 2: Defining f and ρ

- $f \in ((s \to B) \to B) \to s = Ts \to s$
- $\rho_{sf} p = \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}} [\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{p} : p]f$

Lemma 2: Defining f and ρ



Lemma 2: Proof

Now we want to proof for any set s, $f \in ((s \to B) \to B) \to s$, and $h \in (P \to B) \to B = TP$

$$(\rho_{sf} \circ H)h = (f \circ T(\rho_{sf}))h$$

Lemma 2: Proof

$$\rho_{sf}(Hh) = \rho_{sf}(\mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S[]h)$$

Definition of H

$$H = \mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S$$

Lemma 2: Proof

$$\rho_{sf}(\mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S[]h)$$

$$= \rho_{sf}(\mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\lambda\mathbf{h} : (\mathbf{P}\to\mathbf{B})\to\mathbf{B}.\Lambda\mathbf{s}.\mathbf{M}]\!]S[]h)$$

$$= \rho_{sf}((\lambda h : (P\to B)\to B.\mu_{[],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S[])h)$$

$$= \rho_{sf}(\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S[\mathbf{h} : h])$$

Definition of **H**

$\mathbf{H} = \lambda\mathbf{h} : (\mathbf{P}\to\mathbf{B})\to\mathbf{B}.\Lambda\mathbf{s}.\mathbf{M}$

Semantic Rule 9: Semantics of λ

If $e \in E_{[\pi|v:\omega],\omega'}$

Then $\mu_{\pi,\omega\to\omega'}[\![\lambda v : \omega.e]\!]S\eta = \lambda x \in S^{\#}\omega.\mu_{[\pi|v:\omega],\omega'}[\![e]\!]S[\eta|v : x]$

Lemma 2: Proof

$$\rho_{sf}(\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S[\mathbf{h} : h])$$

$$= \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{p} : \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S[\mathbf{h} : h]]f$$

Definition of ρ_{sf}

$\rho_{sf}p = \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{p} : p]f$

Lemma 2: Proof

$$\mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s}:s][\mathbf{p}:\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\mathbf{M}]\!]S[\mathbf{h}:h]]f$$
$$= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![(\lambda\mathbf{p}:\mathbf{P}.\mathbf{p}[\mathbf{s}])(\Lambda\mathbf{s}.\mathbf{M})]\!][S|\mathbf{s}:s][\mathbf{h}:h]f$$

Semantic rule 9: Semantics of λ

$\mu_{\pi,\omega\to\omega'}[\![\lambda v:\omega.e]\!]S\eta = \lambda x \in S^{\#}\omega.\mu_{[\pi|v:\omega],\omega'}[\![e]\!]S[\eta|v:x]$

Semantic rule 5: Environment extension

if $\pi \subseteq \pi'$ and $\eta \subseteq \eta'$ then $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi'\omega}[\![e]\!]S\eta'$

Semantic rule 4: Free variable dependence

if $S\tau = S'\tau$ for all $\tau \in \mathrm{FTV}(e) \cup \mathrm{FTV}(\omega) \cup \bigcup_{v\in dom(\pi)} \mathrm{FTV}(\pi v)$, then $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi\omega}[\![e]\!]S'\eta$

Semantic rule 8: Application

$\mu_{\pi\omega'}[\![e_1(e_2)]\!]S\eta = \mu_{\pi,\omega\to\omega'}[\![e_1]\!]S\eta(\mu_{\pi\omega}[\![e_2]\!]S\eta)$

Lemma 2: Proof

$$\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![(\lambda \mathbf{p} : \mathbf{P}.\mathbf{p}[\mathbf{s}])(\Lambda \mathbf{s}.\mathbf{M})]\!][S|\mathbf{s} : s][\mathbf{h} : h]f$$
$$= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![\Lambda \mathbf{s}.\mathbf{M}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{h} : h]f$$
$$= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![\mathbf{M}]\!][S|\mathbf{s} : s][\mathbf{h} : h]f$$

$$= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![\lambda \mathbf{f} : ((\mathbf{s}\to\mathbf{B})\to\mathbf{B})\to\mathbf{s}.\mathbf{f}(\lambda \mathbf{g} : \mathbf{s}\to\mathbf{B}.\mathbf{h}(\lambda \mathbf{p} : \mathbf{P}.\mathbf{g}(\mathbf{p}[\mathbf{s}]\mathbf{f})))]\!][S|\mathbf{s} : s][\mathbf{h} : h]f$$

Semantic rule 10: Value reduction

$\mu_{\pi\omega'}[\![(\lambda v : \omega.e_1)(e_2)]\!]S\eta = \mu_{\pi\omega'}[\![e_3]\!]S\eta$

Semantic rule 12: Type reduction

If $e \in E_{\pi-\tau,\omega}$ Then we have $\mu_{\pi-\tau,\omega}[\![(\Lambda\tau.e)[\tau]]\!] = \mu_{\pi-\tau,\omega}[\![e]\!]$

Definition of **M**

$\mathbf{M} = \lambda \mathbf{f} : ((\mathbf{s}\to\mathbf{B})\to\mathbf{B})\to\mathbf{s}.\mathbf{f}(\lambda \mathbf{g} : \mathbf{s}\to\mathbf{B}.\mathbf{h}(\lambda \mathbf{p} : \mathbf{P}.\mathbf{g}(\mathbf{p}[\mathbf{s}]\mathbf{f})))$

Lemma 2: Proof

$$\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![\lambda\mathbf{f}:((\mathbf{s}\to\mathbf{B})\to\mathbf{B})\to\mathbf{s}.\mathbf{f}(\lambda\mathbf{g}:\mathbf{s}\to\mathbf{B}.\mathbf{h}(\lambda\mathbf{p}:\mathbf{P}.\mathbf{g}(\mathbf{p}[\mathbf{s}]\mathbf{f})))]\!][S|\mathbf{s}:s][\mathbf{h}:h]f$$

$$= f(\lambda g \in s \to B.h(\lambda p \in P.g(\mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s}:s][\mathbf{p}:p]f)))$$

Semantic rule 9: Semantics of λ

$\mu_{\pi,\omega\to\omega'}[\![\lambda v:\omega.e]\!]S\eta = \lambda x \in S^{\#}\omega.\mu_{[\pi|v:\omega],\omega'}[\![e]\!]S[\eta|v:x]$

Semantic rule 8: Application

$\mu_{\pi\omega'}[\![e_1(e_2)]\!]S\eta = \mu_{\pi,\omega\to\omega'}[\![e_1]\!]S\eta(\mu_{\pi\omega}[\![e_2]\!]S\eta)$

Semantic rule 7: Semantics of ordinary variables

If $v \in dom(\pi)$ then $\mu_{\pi,\pi v}[\![v]\!]S\eta = \eta v$

Semantic rule 5: Environment extension

if $\pi \subseteq \pi'$ and $\eta \subseteq \eta'$ then $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi'\omega}[\![e]\!]S\eta'$

Lemma 2: Proof

$$f(\lambda g \in s \to B.h(\lambda p \in P.g(\mu_{[\mathbf{p}:\mathbf{P}],\mathbf{w}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{p} : p]f)))$$

Definition of ρ_{sf}

$$\rho_{sf}p = \mu_{[p:P],\mathbf{w}}[\![\mathbf{p}[\mathbf{s}]]\!][S|\mathbf{s} : s][\mathbf{p} : p]f$$

$$= f(\lambda g \in s \to B.h(\lambda p \in P.g(\rho_{sf}p)))$$

Definition of T

$$T(\rho) = \lambda h : (s \to B) \to B.\lambda g : s' \to B.h(g \circ \rho)$$

$$= f(T\,\rho_{sf}\,h)$$

Lemma 2: Conclusion

We have proved for any set s, $f \in ((s \to B) \to B) \to s$, and $h \in (P \to B) \to B = TP$

$$(\rho_{sf} \circ H)h = (f \circ T(\rho_{sf}))h$$

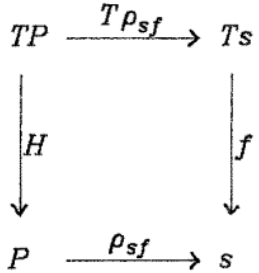


Figure: Morphism between 2 T-algebra's

The End

Questions? Comments?