ソフトウェア基礎論　35－2
(1990. 5. 24)

高階の強く型付けられた言語における Evaluationの対称性について

三好 博之

慶應義塾大学理工学部数理科学科
横浜市港北区日吉3-14-1

本稿の目的は、高階の理論にdeclarativeなcontinuationを扱える枠組を導入し、これとcategoricalな構造との関係を示すことにある。Filinskiはvalueとcontinuationが双対的に解釈できることを指摘し、これを表現する言語とそのcategoricalなsemanticsを与えた。しかしこれは一階の言語であり型理論としては表現力の点で不満足なものであった。ここではこの手法を拡張し、高階の型理論の下でfirst-class continuationにdeclarativeな意味を与える。

On the Symmetry of Evaluation in a Higher-Order Strongly Typed Language

Hiroyuki Miyoshi

Department of Mathematics,
Keio University,
3-14-1 Hiyoshi, Yokohama 223, Japan

The purpose of this thesis is to introduce the higher-order framework to treat delcarative continuations, and to show the relation to some categorical structure. Filinski pointed out that the relation between values and continuations is dual, and provided the corresponding language and its categorical semantics. However that language is first-order and is not satisfiable as type theory. We extend this idea, and show that in higher-order type theory it is possible to give a declarative semantics to first-class continuations.

1 はじめに

本稿の目的は、高階の理論に declarative な continuation を扱える枠組を導入し、これと categorical な構造との関係を示すことにある。具体的には、まず Girard の F_ω[Gir72]を Filinski が[Fil89]において提案した Symmetric Lambda Calculus (SLC)と融合して Higher-order Symmetric Lambda Calculus (HSλ)という体系を構成する。

そして、base category として Cartecian Closed Category (CCC)を持ち、fibre category として Symmetric Combinatory Logic (SCL) [Fil89]を持つ indexed category である Higher-order Symmetric Categorical Combinatory Logic (HSCCL)を構成し、それを用いて HSλ の categorical semantics を与える interpretation を定義して、その soundness を証明する。

1.1 declarative continuation

近年の型理論の発展に伴って、category theory は理論計算機科学の様々な分野で利用されてきている。しかし、それらは主にデータ構造に関することであって、制御構造についてはあまり適切に扱われてこなかった。特に例外処理や非局所的脱出等の（宣言的でなく）命令的な特徴は categorical な概念とはそぐわなかった。

一方で continuation に基づいた denotational semantics はこれらの挙動を説明するのに適している。また、実行時の error handling や backtracking や coroutine の明確な基礎を与え、記述言語の評価順序を気にせずにプログラムやその停止性を形式的に取り扱うことができる。しかしそれは、意味記述に余分な抽象のレベルを導入するため、変換や正当性の証明のための、プログラムについての自動的な推論を困難にする面がある。

Filinski は[Fil89]において value と continuation を双対的に解釈できることを指摘し、実際にそれを陽に表わした言語（Symmetric Lambda Calculus (SLC)）とその categorical な semantics を与える表現（Symmetric Combinatory Logic (SCL)）を構成した。

しかし SLC は simply typed lambda calculus に基づいた一階の言語であり、表現力の点では不満足なものであった。我々はこの手法を全面的に高階の言語に拡張し、強力な型システムの下で宣言的に continuation を扱える体系を示す。

1.2 HSλ

宣言的な continuation について考えるための高階の言語として、我々は Higher-order Symmetric Lambda Calculus (HSλ)を導入する。この言語は、type のレベルでは simply typed lambda calculus と同様の構文を持ち、term のレベルでは value と continuation が対応した対称的な構文を持つ。このことにより型チェックの停止性を保証しつつ、プログラムの実行において停止性が保証されない様々な動作について解析することが可能になる。

また高階であることの利点としては非常に柔軟な表現力を持つことがある。Böhm と Berarducci は second-order λ-calculus において、polymorphic な型を用いることによって、自然数等の「一様な」再帰的データ型を表現できることを示した[BB85]。Pfenning を中心とするグループはこれをさらに拡張し、F_ω や Calculus of Constructions [CH88]等の高階の型理論においては、list や cons のような「一様でない」再帰的データ型を表現できることを示した[PDM89]。HSλ はこの F_ω の拡張とみなすことができ、上に示したような構成を同様に行なうことができる。これらは conversion rule として普通の $\beta\eta$-conversion に従う。

1.3 HS category と HSCCL

さらに Seely の意味での categorical semantics[See87]を与えるために Higher-order Symmetric Category (HS category)と呼ばれる category とその equational な表現である Higher-order Symmetric Categorical Combinatory Logic (HSCCL)を導入する。

HS category は、base category として CCC を持ち、fibre category として SCL を持つ indexed category である。HSCCL は Lambek らによって導入された category を equational な deductive system として表現する方法[LS86]に基づいて HS category を形式化したものである。

HSCCL は P.-L. Curien の CCL (Categorical Combinatory Logic)を高階に拡張したものとみなすことができる。"categorical"という意味は、その combinator と rule が、ある category の object を forget して arrow のみに注目することにより、category 構造から型のついてない情報を抽出することによって得られるということである[Cur86]。逆にいえば combinator は、型が与えられて arrow を決定するという意味で implicit な polymorphism になっている。

これらの手法は Coquand らによって F_ω に適用されている[CE87]が、我々はさらにこれを拡張し、symmetric な構造を持った HSλ に対してもこの手法を用いることができることを示す。そしてそれを用いて HSλ の categorical semantics を与える interpretation を定義して、その soundness を証明する。

1.4 本稿の構成

以後の構成について簡単に述べよう。まず第 2 章では value と continuation の対称性について述べ、対応する言語を非形式的に導入する。第 3 章では、言語 HSλ の形式的な定義を述べる。第 4 章では Seely の意味で HSλ に categorical semantics を与える indexed category である HS category と、その対応を概説する。第 5 章では HS category の equational な表現である HSCCL を構成する。第 6 章では categorical semantics を与える interpretation を定義し、その soundness を示す。それから、これらの体系に関連する話題についてそのいくつかを第 7 章で考察し、最後に結論を述べる。

本論文では category theory の一般的と思われる用語については一切説明を省いた。詳しくは[Mac71,LS86]等を参照されたい。

2 ValueとContinuationの対称性

ここではFilinskiによって指摘されたvalueとcontinuationの双対性と、本稿で行う高階への拡張について簡単に述べる。

関数は普通、valueの変換を行なうものと考えられてきた。例えば初等数学でいう関数はすべてこのようなものである。しかし別の見方も考えられる。それは要求の変換を行なうものとみなすことである。例えば関数 $odd : int \to bool$ は、普通整数値を真偽値に変換するものと考えられるが、真偽値の要求を整数値の要求に変換するものと考えることができる。

実際、例えば実行戦略によってvalueと要求の役割は変わる。データ駆動実行では制御の流れはvalueによって決定され、要求は陽には現れない。要求駆動実行では要求により実行が進み、valueは受動的である。

この見方はいわゆるcontinuation semanticsに拡張することができる。ここではcontinuationがある意味で要求として動作する。lazyな言語においては、continuationはその結果が尋ねられるまでは評価されないものとして計算を進めてゆく。eagerな言語においては、continuationは既に計算された結果が受け取られる。しかし、どちらの場合もcontinuationはvalueの欠如とか不在とみなすことができる。

この考え方を元に、valueとcontinuationが対称的な高階の言語を非形式的に考えてみよう。まずtypeについてはtypecheckの停止性を保証したい。そのために F_ω と同じ構造とする。つまりorderとoperatorという範疇を考え、これらはsimply typed lambda calculusのtypeとtermの関係と、定数を除いて同じ構造とする。orderのなかにはある特別なorder Ω があって、これに属するoperatorをtypeと呼ぶ。

つぎにtermについて考える。termはvalueとcontinuationという二つの範疇からなり、それぞれ一つのtypeに属するものとする。関数はvalueを変換するものとしてとして扱うこともできるし、continuationを変換するものとして扱うこともできる。

valueへのfunction applicationは $e \nearrow f$ と書く。これは e が f によって変換された結果のvalueと考える。対称的にcontinuationへのfunction applictionは $f \swarrow c$ と表わし、c が f によって変換されたと考える。これは入力をまず f に通すような新しいcontinuationを表わす。

関数はvalue abstraction $\sigma \rhd e$ もしくはcontinuation abstraction $c \lhd \tau$ で表わされる。前者は入力された値をどのようにして結果の値に変換するかであり、後者は結果の要求がどのようにして入力の要求に変換されるかである。

conversion ruleは $\beta\eta$-conversionをvalueとcontinuation双方に用いる。また変数については α-conversionの問題を回避するためにde Bruijn indexing [de 72]を用いて自然数で表現するものとする。

SLCではfunctionという中間的な範疇を更に用いているが、われわれは単純にするためこれは省略する。そのためfunctionの合成についてのconversion ruleも省略し、valueもしくはcontinuationでのfunction applictionの合成によって表わすものとする。また異なる型のidentityを同じものとみなすというSLCの規則も我々は導入しない。これらによりここで定義される言語は $\beta\eta$-theoryになる。

最後により category 的な見方を示しておこう。typeをobjectと見ると関数はmorphismとみなせる。type σ のvalueは morphism $\top \to \sigma$ に対応する。ここで $\top$ は(weak) termial objectである。これは入力を取らないdegenerateな関数と見ることができる。双対的に、型 σ の値を受け取るcontinuationはmorphism $\sigma \to \bot$ とみなせる。ここで、$\bot$ は(weak) initial objectである。これは出力をしない関数と見ることができる。

ただしHSλの場合、functionという範疇がないので関数 $\sigma \to \tau$ は、morphism $\top \to \sigma \Rightarrow \tau$ で表わされるvalue、もしくはmorphism $\sigma \Leftarrow \tau \to \bot$ で表わされるcontinuationで表現される。これらのmorphismの間は cu もしくは uc というsignetureで関係づけられる。

3 Higher-order Symmetric λ-Calculus (HSλ)

3.1 syntax

ここではde Bruijn indexとよばれるname-free encodingを用いて変数を自然数で表現し、α-conversionの問題を回避している。de Bruijn indexについては[de 72]を参照せよ。

以下で A, B はorder、σ, τ はoperator、e はexpression (value)、c はcontinuationと呼ばれるsyntax categoryを表す。e, c をまとめてtermと呼ぶ。

$$
\begin{aligned}
A &= \Omega \mid A \Rightarrow B \\
\sigma &= i_o \mid \top \mid \bot \mid \lambda A\sigma \mid \sigma\tau \mid \sigma \Rightarrow \tau \mid \sigma \Leftarrow \tau \mid \forall A\sigma \\
e &= j_x \mid \langle\rangle \mid \sigma \rhd e \mid e_1 \nearrow e_2 \mid \Lambda Ae \mid e\{\sigma\} \mid c \\
c &= k_y \mid [] \mid c \lhd \sigma \mid c_1 \swarrow c_2 \mid \Lambda Ac \mid c\{\sigma\} \mid e
\end{aligned}
$$

ここで i_o, j_x, k_y はde Bruijn indexと呼ばれる自然数である。

operator σ に対して、それに対応する通常の記法のoperatorのfree variableの数を $FV(\sigma)$ と表す。同様に、term t に対して $FV_x(t)$ をfree value variableの数、$FV_y(t)$ をfree continuation variableの数とし、$FV_o(t)$ をfree operator variableの数とする。

また、lifting $\_[\_\uparrow\_\to]$, $\_[\_\uparrow\_\to]_o$, $\_[\_\uparrow\_\to]_x$, $\_[\_\uparrow\_\to]_y$, $\_\uparrow$, $\_\uparrow_o$, $\_\uparrow_x$, $\_\uparrow_y$ とsubstitution $\_[\_\leftarrow\_]$, $\_[\_\leftarrow\_]_o$, $\_[\_\leftarrow\_]_x$, $\_[\_\leftarrow\_]_y$ を[CE87]と同様に定義する[Miy90]。

3.2 typing rules and equality

3.2.1 context of order

orderのcontextはorderの有限リスト $\Delta = (K_1, K_2, \cdots, K_l)$ である。$\mid \Delta \mid$ は Δ の長さ l を表し、$A \cdot \Delta$ はcontext $(A, K_1, K_2, \cdots, K_l)$ を表す。一般に $i \leq l$ に対して $\Delta[i \leftarrow A]$ はcontext $(K_1, K_2, \cdots, K_{i-1}, A, K_i, \cdots, K_{l-1}, K_l)$ を表す。

3.2.2 typing rules for operator

$\Delta = (K_1, K_2, \cdots, K_l)$ が order の context のとき、operator の typing judgement は $\Delta \vdash \sigma : A$ である。ここで、σ は $FV(\sigma) \leq |\Delta|$ であるような operator であり、A は order であり、これは以下の規則によって帰納的に定義される。

$$\frac{i \leq |\Delta|}{\Delta \vdash i_\Delta : K_i}$$

$$\frac{A \cdot \Delta \vdash \sigma : B}{\Delta \vdash \lambda A\sigma : A \Rightarrow B} \qquad \frac{\Delta \vdash \sigma : A \Rightarrow B \quad \Delta \vdash \tau : A}{\Delta \vdash \sigma\tau : B}$$

$$\Delta \vdash \top : A \to \Omega \qquad \Delta \vdash \bot : A \to \Omega$$

$$\frac{\Delta \vdash \sigma : \Omega \quad \Delta \vdash \tau : \Omega}{\Delta \vdash \sigma \Rightarrow \tau : \Omega} \qquad \frac{\Delta \vdash \sigma : \Omega \quad \Delta \vdash \tau : \Omega}{\Delta \vdash \sigma \Leftarrow \tau : \Omega}$$

$$\frac{A \cdot \Delta \vdash \sigma : \Omega}{\Delta \vdash \forall A\sigma : \Omega}$$

order の contextΔ が与えられたとき、$\Delta \vdash \sigma : \Omega$ である operator σ を type という。

operator の equality については、$\beta\eta$-theory を考える。$=$ は equality modulo $\beta\eta$-conversion であり、対応する等式は次のようになる。

$$(\lambda A\sigma)\tau = \sigma[1_o \leftarrow \tau]$$
$$\lambda A(\sigma \uparrow 1_o) = \sigma$$

3.2.3 context of type

与えられた order の context Δ について、value と continuation に対する valid な type の context は、それぞれ type の有限リスト $X = (\nu_1, \nu_2, \cdots, \nu_m)$, $Y = (\xi_1, \xi_2, \cdots, \xi_n)$ である。すなわち、$j \leq m$, $k \leq n$ に対して $\Delta \vdash \nu_j : \Omega$, $\Delta \vdash \xi_k : \Omega$ が成り立っている。

value に対する type の context について、$|X|$ は X の長さ m を表し、$\sigma \cdot X$ は context $(\sigma, \nu_1, \nu_2, \cdots, \nu_m)$ を表す。一般に $j \leq m$ に対して $X[j \leftarrow \sigma]$ は context $(\nu_1, \nu_2, \cdots, \nu_{j-1}, \sigma, \nu_j, \cdots, \nu_{m-1}, \nu_m)$ を表す。また、type に準じて、$X \uparrow = (\nu_1 \uparrow, \nu_2 \uparrow, \cdots, \nu_m \uparrow)$ と定義する。continuation に対する type の context についても同様の定義がなされる。

3.2.4 typing rules for term

Δ, X, Y が上のような条件を満たすとき、value の typing judgement は $\Delta, X, Y \vdash_e e : \sigma$、continuation の typing judgement は $\Delta, X, Y \vdash_c c : \sigma$ である。ここで、e は $FV_x(e) \leq |X|$, $FV_y(e) \leq |Y|$, $FV_o(e) \leq |\Delta|$ を満たす expression、c は $FV_x(c) \leq |X|$, $FV_y(c) \leq |Y|$, $FV_o(c) \leq |\Delta|$ を満たす continuation、σ は operator であり、これらは以下の規則によって帰納的に定義される。

$$\frac{j \leq |X|}{\Delta, X, Y \vdash_e j_x : \nu_j} \qquad \frac{k \leq |Y|}{\Delta, X, Y \vdash_c k_y : \xi_k}$$

$$\Delta, X, Y \vdash_e \langle\rangle : \top \qquad \Delta, X, Y \vdash_c [] : \bot$$

$$\frac{\Delta, \sigma \cdot X, Y \vdash_e e : \tau}{\Delta, X, Y \vdash_e \sigma \rhd e : \sigma \Rightarrow \tau} \qquad \frac{\Delta, X, \tau \cdot Y \vdash_c c : \sigma}{\Delta, X, Y \vdash_c c \lhd \tau : \sigma \Leftarrow \tau}$$

$$\frac{\Delta, X, Y \vdash_e e_1 : \sigma \Rightarrow \tau \quad \Delta, X, Y \vdash_e e_2 : \sigma}{\Delta, X, Y \vdash_e e_2 \nearrow e_1 : \tau} \qquad \frac{\Delta, X, Y \vdash_c c_1 : \sigma \Leftarrow \tau \quad \Delta, X, Y \vdash_c c_2 : \tau}{\Delta, X, Y \vdash_c c_1 \swarrow c_2 : \sigma}$$

$$\frac{\Delta, X, Y \vdash_e e : \forall A\sigma \quad \Delta \vdash \tau : A}{\Delta, X, Y \vdash_e e\{\tau\} : \sigma[1 \leftarrow \tau]_x} \qquad \frac{\Delta, X, Y \vdash_c c : \forall A\sigma \quad \Delta \vdash \tau : A}{\Delta, X, Y \vdash_c c\{\tau\} : \sigma[1 \leftarrow \tau]_y}$$

$$\frac{\Delta, X\uparrow, Y\uparrow \vdash_e e : \sigma}{\Delta, X, Y \vdash_e \Lambda A e : \forall A\sigma} \qquad \frac{\Delta, X\uparrow, Y\uparrow \vdash_c c : \sigma}{\Delta, X, Y \vdash_c \Lambda A c : \forall A\sigma}$$

$$\frac{\Delta, X, Y \vdash_e e : \sigma \quad \sigma = \sigma'}{\Delta, X, Y \vdash_e e : \sigma'} \qquad \frac{\Delta, X, Y \vdash_c c : \sigma \quad \sigma = \sigma'}{\Delta, X, Y \vdash_c c : \sigma'}$$

$$\frac{\Delta, X, Y \vdash_c c : \sigma \Leftarrow \tau}{\Delta, X, Y \vdash_e c : \sigma \Rightarrow \tau} \qquad \frac{\Delta, X, Y \vdash_e e : \sigma \Rightarrow \tau}{\Delta, X, Y \vdash_c e : \sigma \Leftarrow \tau}$$

term の equality に関しては、term の application と operater の application についての $\beta\eta$-theory を考える。上の形式化では対応する等式は次のようになる。

$$e_2 \nearrow (\sigma \rhd e_1) = e_1[1_x \leftarrow e_2]_x \qquad (c_1 \lhd \tau) \swarrow c_2 = c_1[1_y \leftarrow c_2]_y$$
$$\sigma \rhd (1_x \nearrow e \uparrow_x) = e \qquad (c \uparrow_y \swarrow 1_y) \lhd \tau = c$$

$$(\Lambda A e)\{\sigma\} = e[1_o \leftarrow \sigma]_o \qquad (\Lambda A c)\{\sigma\} = c[1_o \leftarrow \sigma]_o$$
$$\Lambda A(e \uparrow_o \{1_o\}) = e \qquad \Lambda A(c \uparrow_o \{1_o\}) = c$$

4 Higher-order Symmetric Category (HS category)

ここでは F_ω に対する Seely の方法に基づいて、HSλ に対応する category である HS category の構成を概説する。これは次章で与えるこの category の equational な表現 HSCCL に対する直観を与えるのが目的である。

4.1 HS category

HS category (S, G) は以下のものから構成される：

1. category S を CCC とし、ある object Ω を区別する。
2. S の上の indexed category G、すなわち functor $G : S_{op} \to Cat$ を考える。ここで Cat は(small) categories の category である。このとき G は次の条件を満足するものである。

   (a) S の任意の object A について、$Obj(G(A))\mathrm{Hom}_S \cong (A, \Omega)$ である。このとき、$G(A)$ の object と S の arrow $A \to \Omega$ を同一視する。

   (b) S の各 object A に対して、$G(A)$ は Symmetric Combinatory Logic (SCL) [Fil89]によって表される category であり、各 arrow $\sigma : A \to B$ に対して、functor $G(\sigma)$ は SCL の構造を保存する。

   (c) S の morphism $\sigma : A \to B$ について、σ^* は、functor $G(\sigma) : G(B) \to G(A)$ であり、この functor は $G(B)$ の object に対して、σ にこの object を右から合成することによって作用するものとする。すなわち、$\sigma^*(\tau) = \sigma; \tau$ である。

   いま S の object B に対して、G^B を $G^B(A) = G(A \times B)$ と $G^B(\sigma) = G(\sigma \times Id)$ で定義される functor とする。G から G^B への natural transformation Fst^*_B を、S の object A に対して、$Fst^*_B = G(Fst_{A,B})$ と定義する。ここで $Fst_{A,B}$ は $A \times B$ から A への projection である。このとき次の条件を満たす。

   i. 各 $Fst^*_B(A)$ が right adjoint $\forall_B(A)$ を持つ。つまり、ここで $\Lambda_B(A)$ をその adjunction における hom-set の bijection とすると、
   $$\Lambda_B(A) : \mathrm{Hom}_{G^B(A)}((Fst; \sigma), \tau) \cong \mathrm{Hom}_{G(A)}(\sigma, \forall_B(A)(\tau))$$
   となる。

   ii. $\forall_B$ は natural transformation になる。そして $\Lambda_B(A)$ は A において natural である。

eqational な表現で書くためには、adjunction $\forall_B(A)$ を bijection $\Lambda_B(A)$ と counit morphism $proj$ によって定義することにする。このとき $proj$ は $proj = \Lambda^{-1}(Id_{\forall(\tau)})$ で定義される $(Fst; \forall(\tau))$ から τ への $G_B(A)$ の morphism である。

4.2 category of HS categories

HS category の functor $F : (G, S) \to (G, S)$ は次のものからなる：

1. finite product と exponent を保存する functor $F_0 : S \to S'$
2. SCL の構造を保存する S-indexed functor $F_1 : G \to F_0^* G'$

ここで $F_0^* G'$ は $F_0^* G'(A) = G'(F_0(A))$ によって定義される S 上の indexed category である。adjunction を保存するというのは以下の図式が可換になるということである。

F_1 が S-indexed であるという条件は F_1 が f^* と交換できることを保証している。G と G' を functor としてみると、このことは F_1 が natural transformation であることを意味する。HS categories と上で定義した functor からなる category を HSCat と呼ぶ。

5 Higher-order Symmetric CCL (HSCCL)

category を equational な deductive system として定義する方法は Lambek によって導入された[LS86]。ここで用いた方法は[CE87]で F_ω に適用されたものをさらに拡張したものである。

5.1 order and operator

order (kind)は、BNF 記法を用いると次のように表される。ここでメタ変数 A,B は order である。

$$A := \Omega \mid 1 \mid A \Rightarrow B \mid A \times B$$

これは base category における object に対応する。operator も同様に次のように表される。ここでメタ変数 σ, τ は operator である。

$$\begin{aligned} \sigma \quad := \quad & Id \mid \sigma; \tau \mid \bigcirc \mid \langle \sigma, \tau \rangle \mid Fst \mid Snd \mid Cur(\sigma) \mid App \mid \sigma \otimes \tau \mid \\ & \top \mid \bot \mid \times \mid + \mid \Rightarrow \mid \Leftarrow \mid \amalg \end{aligned}$$

これは base category における morphism に対応する。

operator の order による typing judgement は次のようなものである。

$$\vdash \sigma : A \to B$$

ここで、σ は operator 、A と B は order である。なお自明な場合、次のように省略する。

$$\sigma : A \to B$$

5.2 typing rule for operators

これらは cartesian closed categories の公理に、type constructor を表す補助定数のための typing rule を加えたものである。

$$\bigcirc : A \to 1 \qquad Id : A \to A$$

$$\frac{\sigma : A \to B \quad \tau : B \to C}{\sigma;\tau : A \to C} \qquad \frac{\sigma : C \to A \quad \tau : C \to B}{\langle \sigma, \tau \rangle : C \to A \times B}$$

$$Fst : A \times B \to A \qquad Snd : A \times B \to B$$

$$\frac{\sigma : A \times B \to C}{Cur(\sigma) : A \to (B \Rightarrow C)} \qquad App : (B \Rightarrow C) \times B \to C$$

補助定数のための typing rule は次のようなものである。

$$\top : A \to \Omega \qquad \bot : A \to \Omega$$
$$\times : (\Omega \times \Omega) \to \Omega \qquad + : (\Omega \times \Omega) \to \Omega$$
$$\Rightarrow : (\Omega \times \Omega) \to \Omega \qquad \Leftarrow : (\Omega \times \Omega) \to \Omega$$
$$\Pi : (A \Rightarrow \Omega) \to \Omega$$

5.3 equations on operators

$$\bigcirc = Id$$

$$\sigma;(\tau;\rho) = (\sigma;\tau);\rho \qquad \sigma;Id = \sigma \qquad Id;\sigma = \sigma$$
$$\langle \sigma,\tau \rangle;Fst = \sigma \qquad \langle \sigma,\tau \rangle;Snd = \tau \qquad \langle Fst, Snd \rangle = Id$$
$$\rho;\langle \sigma,\tau \rangle = \langle \rho;\sigma, \rho;\tau \rangle$$
$$\sigma \otimes \tau = \langle Fst;\sigma, Snd;\tau \rangle \qquad (Cur(\sigma) \otimes Id);App = \sigma \qquad Cur((\sigma \otimes Id);App) = \sigma$$

5.4 type and term

operater のなかで $t : 1 \to \Omega$ であるものを type 、$t : A \to \Omega$ を type scheme と呼ぶ。これらをまとめて type と呼ぶこともある。これらは index category の object に対応する。

term の type による typing judgement は次のようなものである。

$$\vdash t : \sigma \to \tau$$

ここで、t は term 、σ と τ は type である。なお自明な場合、次のように省略する。

$$t : \sigma \to \tau$$

以下は type についての等式である。

$$\top = \bigcirc;\top \qquad \bot = \bigcirc;\bot$$
$$\sigma \times \tau = \langle \sigma,\tau \rangle;\times \qquad \sigma + \tau = \langle \sigma,\tau \rangle;+$$
$$\sigma \Rightarrow \tau = \langle \sigma,\tau \rangle;\Rightarrow \qquad \sigma \Leftarrow \tau = \langle \sigma,\tau \rangle;\Leftarrow$$
$$\forall(\sigma) = cu(\sigma);\Pi$$

5.5 typing rules for terms

まず、SCL [Fil89]の typing rule に対応する規則を示す。

$$\frac{\sigma : A \to \Omega}{id : \sigma \to \sigma}$$

$$\frac{t : \sigma \to \tau \quad u : \tau \to \rho}{t;u : \sigma \to \rho}$$

$$\frac{\tau : A \to \Omega}{\bigcirc : \tau \to \top} \qquad \frac{\tau : A \to \Omega}{\Box : \bot \to \tau}$$

$$\frac{t : \rho \to \sigma \quad u : \rho \to \tau}{\langle t, u \rangle : \rho \to \sigma \times \tau} \qquad \frac{t : \sigma \to \rho \quad u : \tau \to \rho}{[t, u] : \sigma + \tau \to \rho}$$

$$\frac{\sigma : A \to \Omega \quad \tau : A \to \Omega}{\pi_1 : \sigma \times \tau \to \sigma} \qquad \frac{\sigma : A \to \Omega \quad \tau : A \to \Omega}{\iota_1 : \sigma \to \sigma + \tau}$$

$$\frac{\sigma : A \to \Omega \quad \tau : A \to \Omega}{\pi_2 : \sigma \times \tau \to \tau} \qquad \frac{\sigma : A \to \Omega \quad \tau : A \to \Omega}{\iota_2 : \tau \to \sigma + \tau}$$

$$\frac{t : \sigma \times \tau \to \rho}{cu\,(t) : \sigma \to (\tau \Rightarrow \rho)} \qquad \frac{t : \rho \to \sigma + \tau}{uc\,(t) : (\rho \Leftarrow \tau) \to \sigma}$$

$$\frac{\tau : A \to \Omega \quad \rho : A \to \Omega}{ap : (\tau \Rightarrow \rho) \times \tau \to \rho} \qquad \frac{\tau : A \to \Omega \quad \rho : A \to \Omega}{pa : \rho \to (\rho \Leftarrow \tau) + \tau}$$

$$\frac{\sigma : A \to \Omega \quad \tau : A \to \Omega \quad \rho : A \to \Omega}{\phi : \sigma \times (\tau \Leftarrow \rho) \to ((\sigma \times \tau) \Leftarrow \rho)} \qquad \frac{\sigma : A \to \Omega \quad \tau : A \to \Omega \quad \rho : A \to \Omega}{\theta : (\rho \Rightarrow (\sigma + \tau)) \to \sigma + (\rho \Rightarrow \tau)}$$

$\forall$のために以下の規則を付け加える。

$$\frac{t : (Fst;\sigma) \to \tau}{\Lambda(t) : \sigma \to \forall(\tau)}$$

$$proj : (Fst;\forall(\sigma)) \to \sigma$$

最後に type equality の規則を付け加える。

$$\frac{t : \sigma \to \tau \quad \sigma = \sigma_1 \quad \tau = \tau_1}{t : \sigma_1 \to \tau_1}$$

operater の term への action の typing rule は次のようなものである。

$$\frac{\sigma : A \to B \quad \tau : B \to \Omega \quad \rho : B \to \Omega \quad t : \tau \to \rho}{\sigma * t : (\sigma;\tau) \to (\sigma;\rho)}$$

5.6 equations for terms

term を表す morphism $t : \sigma \to \tau$ が strict であるとは、$\Box;t = \Box$ を満たすことをいい、total (costrict)であるとは $t;\bigcirc = \bigcirc$ を満たすことをいう。

ここで、a は strict な arrow であり、b は total (costrict)な arrow とする。

$$s;(t;u) = (s;t);u$$
$$s;id = s$$
$$id;s = s$$

$$\bigcirc_{1_A} = id_{1_A} \qquad \Box_{0_A} = id_{0_A}$$
$$\langle s,t\rangle;\pi_1 = s \qquad \iota_1;[s,t] = s$$
$$\langle s,t\rangle;\pi_2 = t \qquad \iota_2;[s,t] = t$$
$$\langle \pi_1,\pi_2\rangle = id \qquad [\iota_1,\iota_2] = id$$
$$a;\langle s,t\rangle = \langle a;s,a;t\rangle \qquad [s,t];b = [s;b,t;b]$$
$$a_1 \times a_2 = \langle \pi_1;a_1,\pi_2;a_2\rangle \qquad (b_1 + b_2) = [b_1;\iota_1,b_2;\iota_2]$$
$$(cu\,(s) \times id\,);ap = s \qquad pa\,;(uc\,(s) + id\,) = s$$
$$cu\,((a \times id\,);ap\,) = a \qquad uc\,(pa\,;(b + id\,)) = b$$
$$\langle \bigcirc;s,id\rangle;\phi = uc\,(\langle \bigcirc;s,id\rangle;pa\,) \qquad \theta;[s;\Box,id\,] = cu\,(ap\,;[s;\Box,id\,])$$
$$\phi;uc\,(\pi_2;s) = \pi_2;uc\,(s) \qquad cu\,(s;\iota_2);\theta = cu\,(s);\iota_2$$

$*$ が contravariant functor であることを表わす等式を加える

$$(\sigma;\tau) * t = \sigma * (\tau * t)$$
$$id * t = t$$

以下の等式はoperatorが、termに対応しているこのHSCCLの構造に$*$によって作用することを表わしている。正確にいえば、最初の2式は$\sigma*$がcovariant functorとして作用することを表わし、残りの式はSCLの構造を同型の意味でなくそのまま維持することを述べている。

$$\sigma*(t;u)=(\sigma*t);(\sigma*u)$$
$$\sigma*id=id$$

$$\begin{array}{ll}\sigma*\bigcirc=\bigcirc & \sigma*\Box=\Box\\ \sigma*\langle t,u\rangle=\langle\sigma*t,\sigma*u\rangle & \sigma*[t,u]=[\sigma*t,\sigma*u]\\ \sigma*cu(t)=cu(\sigma*t) & \sigma*uc(t)=uc(\sigma*t)\\ \sigma*\pi_1=\pi_1 & \sigma*\iota_1=\iota_1\\ \sigma*\pi_2=\pi_2 & \sigma*\iota_2=\iota_2\\ \sigma*ap=ap & \sigma*pa=pa\\ \sigma*\phi=\phi & \sigma*\theta=\theta\end{array}$$

次に$\Lambda(_)$と$proj$が$Fst*_$のright adjunction $\forall(_)$を定義することを表現する等式を与える。これらは$Fst*_$についてのcouniversal mapping problemのsolutionが$(\forall,proj,\Lambda)$となることを示している。（これがadjunctionを構成することは[LS86]を参照せよ）。なお括弧の省略のため、'$*$'の優先順位を';'よりも上とする。

$$Fst*\Lambda(s);proj=s$$
$$s;\Lambda(t)=\Lambda(Fst*s;t)$$
$$\Lambda(proj)=id$$

さらにこれらのadjunctionがnaturalであることを表現する等式を与える。正確にいえば$\Lambda(_)$と$proj$がnaturalであるということである。

$$\sigma*\Lambda(s)=\Lambda((\sigma\otimes Snd)*s)$$
$$(\sigma\otimes Snd)*proj=proj$$

最後の式と$\Lambda(proj)=id$から次の式が求まる。

$$\Lambda(\langle Id,Snd\rangle*proj)=Id$$

これは、termのoperatorへのapplicationのextensionalityを表している。

6 Interpretation

6.1 order and operator

HSλのorderはHSCCLのorderにそのままinterpretされる。$[A]$はorder Aのinterpretationを表すものとする。

$$[\Omega]=\Omega$$
$$[A\Rightarrow B]=[A]\Rightarrow[B]$$

$\Delta=(K_1,\cdots,K_l)$がHSλのorderのcontextであるとき、Δのinterpretation $K=[\Delta]$をorder $(\cdots(1\times[K_l])\cdots)\times[K_1]$と定義する。そして$\Delta\vdash\sigma:A$を満たす全ての$\sigma$について、$\mathcal{O}[\sigma]_K$を以下で帰納的に構成されるHSCCLのoperator $\mathcal{O}[\sigma]_K:K\rightarrow[A]$であると定義する。

$$\mathcal{O}[i_o]_K=\delta Fst^{i-1};Snd$$
$$\mathcal{O}[\top]_K=\top$$
$$\mathcal{O}[\bot]_K=\bot$$
$$\mathcal{O}[\lambda A\sigma]_K=Cur(\mathcal{O}[\sigma]_{K\times[A]})$$
$$\mathcal{O}[\sigma\tau]_K=\langle\mathcal{O}[\sigma]_K,\mathcal{O}[\tau]_K\rangle;App$$
$$\mathcal{O}[\sigma\Rightarrow\tau]_K=\mathcal{O}[\sigma]_K\Rightarrow\mathcal{O}[\tau]_K$$
$$\mathcal{O}[\sigma\Leftarrow\tau]_K=\mathcal{O}[\sigma]_K\Leftarrow\mathcal{O}[\tau]_K$$
$$\mathcal{O}[\forall A\sigma]_K=\mathcal{O}[\lambda A\sigma]_K;\Pi$$

ここでδは環境の変数であり、評価時に与えられる。

最後の式はΠがtype schemeをtypeに変換することを示している。またこの定義によりHSλのtypeはHSCCLのtypeにinterpretされる。このoperatorのinterpretationは本質的に、simply typed λ-calculusのtermのCCLへのinterpretationと同様である[Cur86]。

6.2 value and continuation

Δをorderのcontextとし、$X=(\nu_1,\cdots,\nu_m)$, $Y=(\xi_1,\cdots,\xi_n)$をそれぞれその orderのcontextにおけるvalueとcontinuationのtypeのcontextとする。$K=[\Delta]$であるとき、Xのinterpretation $\nu=[X]_K$をtypeのproduct $(\cdots(\top\times\mathcal{O}[\nu_m]_K)\cdots)\times\mathcal{O}[\nu_1]_K$と定義し、$Y$のinterpretation $\xi=[Y]_K$をtypeのsum $(\cdots(\bot+\mathcal{O}[\xi_n]_K)\cdots)+\mathcal{O}[\xi_1]_K$と定義する。ここで$\nu_j$, ξ_kはtypeであるので、$\mathcal{O}[\nu_j]_K:K\rightarrow\Omega$, $\mathcal{O}[\xi_k]_K:K\rightarrow\Omega$となることに注意せよ。そして、$\Delta,X,Y\vdash_e e:\sigma$を満たす$e$に対し

て、$\mathcal{E}[e]^{K}_{\nu\ \xi}$ を以下で帰納的に構成される HSCCL の term $\mathcal{E}[e]^{K}_{\nu\ \xi} : \nu \to \mathcal{O}[\sigma]_K$ と定義し、$\Delta, X, Y \vdash_c c : \tau$ を満たす e に対して、$\mathcal{C}[c]^{K}_{\nu\ \xi}$ を同様に HSCCL の term $\mathcal{C}[c]^{K}_{\nu\ \xi} : \mathcal{O}[\tau]_K \to \xi$ と定義する。

$$\mathcal{E}[j_x]^{K}_{\nu\ \xi} = \delta * (x; \pi_1^{j-1}; \pi_2) \qquad \mathcal{C}[k_y]^{K}_{\nu\ \xi} = \delta * (\iota_2; \iota_1^{k-1}; y)$$
$$\mathcal{E}[\langle\rangle]^{K}_{\nu\ \xi} = \delta * \top \qquad \mathcal{C}[[]]^{K}_{\nu\ \xi} = \delta * \bot$$
$$\mathcal{E}[\sigma \rhd e]^{K}_{\nu\ \xi} = cu(\mathcal{E}[e]^{K}_{\nu \times \mathcal{O}[\sigma]_K\ \xi}) \qquad \mathcal{C}[c \lhd \sigma]^{K}_{\nu\ \xi} = uc(\mathcal{C}[c]^{K}_{\nu\ \xi + \mathcal{O}[\sigma]_K})$$
$$\mathcal{E}[e_1 \nearrow e_2]^{K}_{\nu\ \xi} = (\mathcal{E}[e_2]^{K}_{\nu\ \xi}, \mathcal{E}[e_1]^{K}_{\nu\ \xi}); ap \qquad \mathcal{C}[c_1 \swarrow c_2]^{K}_{\nu\ \xi} = pa; [\mathcal{C}[c_1]^{K}_{\nu\ \xi}, \mathcal{C}[c_2]^{K}_{\nu\ \xi}]$$
$$\mathcal{E}[\Lambda A e]^{K}_{\nu\ \xi} = \Lambda(\mathcal{E}[e]^{K \times [A]}_{Fst;\nu\ Fst;\xi}) \qquad \mathcal{C}[\Lambda A c]^{K}_{\nu\ \xi} = \Lambda(\mathcal{C}[c]^{K \times [A]}_{Fst;\nu\ Fst;\xi})$$
$$\mathcal{E}[e\{\sigma\}]^{K}_{\nu\ \xi} = \mathcal{E}[e]^{K}_{\nu\ \xi}; (\langle Id, \mathcal{O}[\sigma]_K\rangle * proj) \qquad \mathcal{C}[c\{\sigma\}]^{K}_{\nu\ \xi} = \mathcal{C}[c]^{K}_{\nu\ \xi}; (\langle Id, \mathcal{O}[\sigma]_K\rangle * proj)$$
$$\mathcal{E}[c]^{K}_{\nu\ \xi} = cu(\pi_2; pa; [\mathcal{C}[c]^{K}_{\nu\ \xi}; y; \Box, id]) \qquad \mathcal{C}[e]^{K}_{\nu\ \xi} = uc(\langle \bigcirc; x; \mathcal{E}[e]^{K}_{\nu\ \xi}, id\rangle; ap; \iota_2)$$

ここで δ, x, y は環境の変数であり、評価時に与えられる。最後の2式は functional closure もしくは contextual continuation にしてから変換することに対応する。

6.3 soundness

スペースの都合上、詳しい証明は省く。詳しくは[Miy90]を参照。

Lemma 6.1 (substitution lemma 1) *order* の *context* を $\Delta = (K_1, \cdots, K_l)$ とし、その *interpretation* を $K = [\Delta]$ とする。このとき次の命題が成り立つ。

1. $\Delta \vdash \sigma : B$ のとき、$A \cdot \Delta \vdash \sigma\uparrow : B$ であり、$\mathcal{O}[\sigma\uparrow]_{K \times [A]} = Fst; \mathcal{O}[\sigma]_K$
2. $A \cdot \Delta \vdash \sigma : B$ かつ $\Delta \vdash : A$ のとき、$\mathcal{O}[\sigma[1_o \leftarrow \tau]]_K = \langle Id, \mathcal{O}[\tau]_K\rangle; \mathcal{O}[\sigma]_{K \times [A]}$

Lemma 6.2 (substitution lemma 2) *order*、*value* の *type*、*continuation* の *type* の *context* をそれぞれ $\Delta = (K_1, \cdots, K_l)$, $X = (\nu_1, \cdots, \nu_m)$, $Y = (\xi_1, \cdots, \xi_n)$ とし、その *interpretation* を $K = [\Delta]$, $\nu = [X]_K$, $\xi = [Y]_K$ とする。このとき次の命題が成り立つ。

1. $\Delta, X, Y \vdash_e e : \sigma$ のとき、$A \cdot \Delta, X\uparrow, Y\uparrow \vdash_e e\uparrow_o : \sigma\uparrow$ であり、$\mathcal{E}[e\uparrow_o]^{K \times [A]}_{Fst;\nu\ Fst;\xi} = Fst * \mathcal{E}[e]^{K}_{\nu\ \xi}$
2. $A \cdot \Delta, X\uparrow, Y\uparrow \vdash_e e : \sigma$ かつ $\Delta \vdash \sigma : A$ のとき、$\mathcal{E}[e[1_o \leftarrow \sigma]_o]^{K}_{\nu\ \xi} = \langle Id, \mathcal{O}[\sigma]_K\rangle * \mathcal{E}[e]^{K \times [A]}_{Fst;\nu\ Fst;\xi}$
3. $\Delta, X, Y \vdash_e e : \sigma$ のとき、$\Delta, \tau \cdot X, Y \vdash_e e\uparrow_x : \sigma$ であり、$\mathcal{E}[e\uparrow_x]^{\nu \times \mathcal{O}[\tau]_K}_{\xi} = \pi_1; \mathcal{E}[e]^{K}_{\nu\ \xi}$
4. $\Delta, \tau \cdot X, Y \vdash_e e : \sigma$ かつ $\Delta, X, Y \vdash_e e_1 : \tau$ のとき、$\mathcal{E}[e[1_x \leftarrow e_1]_x]^{K}_{\nu\ \xi} = \langle id, \mathcal{E}[e_1]^{K}_{\nu\ \xi}\rangle; \mathcal{E}[e]^{\nu \times \mathcal{O}[\tau]_K}_{\xi}$

$\mathcal{C}[\ ]$ についても同様の結果が成り立つ。

Proof $\_\uparrow$ と $\_[\_ \leftarrow \_]$ の定義から operator, value, continuation 上の構造帰納法によって求まる。 □

Lemma 6.3 (soundness for β- and η-conversion 1) *order* の *context* を $\Delta = (K_1, \cdots, K_l)$ とし、その *interpretation* を $K = [\Delta]$ とする。このとき次の命題が成り立つ。

1. $\mathcal{O}[(\lambda A \sigma)\tau]_K = \mathcal{O}[\sigma[1_o \leftarrow \tau]]_K$
2. $\mathcal{O}[\lambda A(\sigma\uparrow 1_o)]_K = \mathcal{O}[\sigma]_K$

Lemma 6.4 (soundness for β- and η-conversion 2) *order*、*value* の *type*、*continuation* の *type* の *context* をそれぞれ $\Delta = (K_1, \cdots, K_l)$, $X = (\nu_1, \cdots, \nu_m)$, $Y = (\xi_1, \cdots, \xi_n)$ とし、その *interpretation* を $K = [\Delta]$, $\nu = [X]_K$, $\xi = [Y]_K$ とする。このとき次の命題が成り立つ。

1. $\mathcal{E}[e_2 \nearrow (\sigma \rhd e_1)]^{K}_{\nu\ \xi} = \mathcal{E}[e_1[1_x \leftarrow e_2]_x]^{K}_{\nu\ \xi}$
2. $\mathcal{E}[\sigma \rhd (1_x \nearrow e\uparrow_x)]^{K}_{\nu\ \xi} = \mathcal{E}[e]^{K}_{\nu\ \xi}$
3. $\mathcal{E}[(\Lambda A e)\{\sigma\}]^{K}_{\nu\ \xi} = \mathcal{E}[e[1_o \leftarrow \sigma]_o]^{K}_{\nu\ \xi}$
4. $\mathcal{E}[\Lambda A(e\uparrow_o \{1_o\})]^{K}_{\nu\ \xi} = \mathcal{E}[e]^{K}_{\nu\ \xi}$

$\mathcal{C}[\ ]$ についても同様の結果が成り立つ。

Theorem 6.5 (soundness theorem) σ_1, σ_2 を *operator*、e_1, e_2 を *value*、c_1, c_2 を *continuation* とする。

1. $\sigma_1 = \sigma_2$ ならば、$\mathcal{O}[\sigma_1]_K = \mathcal{O}[\sigma_2]_K$
2. $e_1 = e_2$ ならば、$\mathcal{E}[e_1]^{K}_{\nu\ \xi} = \mathcal{E}[e_2]^{K}_{\nu\ \xi}$
3. $c_1 = c_2$ ならば、$\mathcal{C}[c_1]^{K}_{\nu\ \xi} = \mathcal{C}[c_2]^{K}_{\nu\ \xi}$

Proof HSλ は $\beta\eta$-thory であるので、$\beta\eta$-conversion についての soundness より成り立つ。 □

7 議論

本節ではHSlambdaやHSCCLに関係するいくつかの話題について述べる。

論理そのものを記述する言語は自動証明のためのメタ言語として古くはAUTOMATH projectにまでさかのぼるが、最近型理論の発展により刺激を受け、再びgeneral logicという見方で見直されてきた。その主なものにはLogical Framework[HHP87], Calculus of Constructions[CH88], ECC[Luo89]等がある。HSλはこのような目的にも用いることができる可能性がある。その利点としては論理システムとして矛盾のある体系についても部分的にしろ統一的に取り扱えることである。また計算による学習のための枠組に用いることも考えられる。たとえば例によるプログラム合成[Hag89]において停止しないプログラムについてもまとめて取り扱える可能性がある。

高階の型付き言語においては言語そのものをcompile-time part (typechecking)とrun-time part (execution)を明確に分離してindexed categoryとしてそれらを関係付けることにより、categorical semanticsにそれらの概念を導入することができる[Mog89]。このような言語では型の部分がcompile時に計算できる（これは部分計算と考えることもできる）ため、compileの最適化処理等に応用することも考えられる。

linear logic [Gir87]とSLCの類似もFilinskiによって指摘されている。intuitionistic linear logicはsymmetric monoidal categoryとの対応があり、CCCにおけるCAMのようなabstract machine (Linear Abstract Machine [Laf88])も考えられている。その意味でも本論文で述べたような高階のcategorical logicとの関係についての研究が期待される。

8 結論

我々は高階の型理論にvalueとcontinuationの双対性を導入した体系HSλを構成し、それに対してcategorical semanticsを与えるために、HS categoryと呼ばれるcategoryとそのeqational representationであるHSCCLを構成した。そしてsemanticsを与えるinterpretationを定義し、そのsoundnessを証明した。さらにいくつかの話題について簡単に考察した。

References

[BB85] Corrado Böhm and Alessandro Berarducci. Automatic synthsis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39, 1985.

[CE87] Thierry Coquand and Thomas Ehrhard. An equational presentation of higher order logic. In *Category Theory and Computer Science, LNCS283*, Edinburgh, U.K., Sep. 1987.

[CH88] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[Cur86] P.-L. Curien. Categorical combinators. *Information and Control*, 69:188–255, 1986.

[de 72] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34:381–392, 1972.

[Fil89] Andrzej Filinski. Declarative continuations: an investigation of duality in programming language semantics. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, Pitts A. M., and Poigné, editors, *Category Theory and Computer Science, LNCS 389*, pages 224–249, Springer-Verlag, Manchester, UK, September 1989.

[Gir72] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Universite Paris VII, 1972.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–120, 1987.

[Hag89] Masami Hagiya. Programming by example and proving by example (in Japanese). manuscript, 1989.

[HHP87] Robert Harper, Fuiro Honsell, and Gordon Plotkin. A framework for defining logics. In *Symposium on Logic in Computer Science*, pages 194–204, IEEE, Ithaca, New York, USA, 1987.

[Laf88] Y. Lafont. The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.

[LS86] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic.* Volume 7 of *Cambridge Studies in Advanced Mathematics*, Cambridge University Press, 1986.

[Luo89] Zhaohui Luo. ECC, an extended calculus of constructions. In *Fourth Annual Symposium on Logic in Computer Science*, IEEE, 1989.

[Mac71] S. MacLane. *Categories for Working Mathematian.* Volume 5 of *Graduate Texts in Mathematics*, Springer-Verlag, 1971.

[Miy90] Hiroyuki Miyoshi. *On the Symmetry of Evaluation in a Higher-Order Strongly Typed Language (in Japanese).* Master's thesis, Department of Mathematics, Keio University, 1990.

[Mog89] Eugenio Moggi. A category-theoretic account of program modules. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, Pitts A. M., and Poigné, editors, *Category Theory and Computer Science*, pages 101–117, Springer-Verlag, Manchester, UK, September 1989.

[PDM89] Benjamin Pierce, Scott Dietzen, and Spiro Michaylov. *Programming in Higher-Order Typed Lambda-Calculi.* Technical Report CMU-CS-89-111, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, March 1989.

[See87] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *The Journal of Symbolic Logic*, 52(4), Dec. 1987.