なぜ古典論理に計算的意味を与えるのか
—SLC にもとづく Dummett 的観点の拡張の試み—
Why giving computational interpretation to classical logic?
-an attempt at extending Dummett's conception of meaning based on SLC-

角田健太郎

Abstract

Recently, computational interpretations of propositional logical calculi have been extended from intuitionistic logic to classical logic. That seems to prompt us to rethink Dummett's notion of meaning and make it more generous so that not only intuitionistic logic but also (at least the elementary parts of) classical logic could be made sense of. For this purpose, it would be indispensable to introduce one new technical notion from the semantics of programming languages: the notion of *continuation*, which designates, roughly speaking, 'that part of the computational process that has not been executed yet'. I conclude this note by firstly surveying the computational system named SLC (*symmetric lambda calculus*) developed by Filinski and Asai et al., in which the notion of continuation is fully made use of, and secondly proposing several important improvements that resulted from my own research which is still under development.

1 研究テーマ

論理学とはすなわち妥当な推論形式の研究であるという見方は、哲学者たちの間に広く共有されている。つまり論理学とは、モデル等の道具立てによって一定の妥当性の基準を据えて、どのような推論形式が妥当であるのかを述べる研究であるという見方である。実際それは間違いではないであろう。しかし近年ますます進展しているのは、そうした推論形式の妥当性を述べることに留まらないような、より詳細な、論理的推論の各ステップに含まれている計算的な推論的原理の解明である。例えば BHK 解釈は、実効的演算や幾何学的作図などに典型的に含まれる構成 (construction) の概念—これはすなわち、素材となるものの構造を常に保存する形で新たな対象を作り出すような対象形成手続きを特徴づける概念である—によって直観主義論理の各推論規則を特徴づけることができ、そして直観主義論理の証明プロセスにより導出された論理式は「その論理式が諸仮定からどのような実効的手続きを経て導出されたか」という証明の構造をその表現内容とすると解釈できることを明らかにした [7]。また、この BHK 解釈のひとつの洗練と見ることができる Curry-Howard 対応は—以下で詳しく説明するように—直観主義論理の証明プロセスが λ 式と呼ばれるコンピュータプログラムの抽象的表現の構築プロセスと対応付けられることを明らかにした [1][6]。

そうした近年の論理学の展開は、哲学にどういうことを教えるだろうか。すでにこうした情勢と結びついている哲学のひとつとして、Dummett の直観主義哲学がある。彼は 1973 年の論文 [2][12] で、命題の意味内容はその命題の「使い方・使用 (use)」に訴えて説明されるべきであり、そして BHK 解釈は直観主義論理に対して「使い方」による意味説明を与えていると論じた。ここで彼は直観主義論理による証明こそが命題の意味の源泉であると主張し、他方で古典論理によって証明された命題は正当化を欠いた無意味な命題であるとする過激な主張を行った。というのも、任意の式 $\mathfrak{A}$ について $\mathfrak{A} \vee \neg\mathfrak{A}$ が無仮定で成り立つ古典論理では、どの選言文も選言肢の少なくとも一方が実効的手続きを経た上で導出されるという BHK 解釈の条件が崩れている―つまり古典論理は「使い方」による説明を超えている―と彼は考えたからである。

Dummett は、BHK 解釈の観点が論理結合子延いては言語の意味の問題でもあることを深く明らかにした点で参考になる。しかし、すでに様々な応用が行われている数学の定理を導出した古典論理を一切無意味として切り捨ててしまうのは奇異だと思われる。直観主義論理の証明プロセスが含む実効的手続きを経る導出の概念には帰着できずとも、しかし依然としてある種の合理化と言えるような部分が古典論理の証明プロセスには含まれており、Dummett はこれを見過ごしてしまっているのではなかろうか。本研究は、古典論理に特有なこの合理化概念を究明し古典論理的命題の意味を与え、そして Dummett の哲学の (忠実ではないかもしれないが) ひとつの健全な発展の方向として古典論理ベースの哲学を展開することをテーマとしている。

2 研究の背景・先行研究

Dummett が指摘するような古典論理のいわゆる非構成性を引き受けた上で、しかしながら近年の古典論理研究の進展の中では、古典論理の証明プロセスは導出される論理式の内容に対してある本質的な関わりを持っているということが明らかとなってきた。参考となるのは Curry-Howard 対応の古典論理への拡張の研究である。この研究は、古典論理の証明プロセスが、計算の順序を制御するある特別な演算子を含むようなプログラムの構築プロセスと対応付けられることを明らかにした。以下ではこのことを順を追って説明する。なお以下では紙幅の節約のため直観主義論理を IL と呼び、古典論理を CL と呼ぶ。

出発点となるのは、Gentzen の 1935 年の論文 [4] である。Gentzen はここで IL の体系として直観主義自然演繹 NJ を、CL の体系として古典自然演繹 NK を与えた。Gentzen の体系では、ある論理式が証明可能であるとき、かつその時に限って、推論プロセスの 1 ステップを表現する図式を有限木構造状に連鎖させた証明図と呼ばれる図式が構成可能である。例えば IL で証明可能ではな

く、かつ CL で証明可能なパースの法則と呼ばれる論理式 $((\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A}) \supset \mathfrak{A}$ をとろう。パースの法則を導出する NK-証明図は例えば次の形をしている。[1]

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathfrak{A} \supset \bot \vdash \mathfrak{A} \supset \bot}\text{公理}}{(\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A} \vdash (\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A}}\text{公理} \quad \dfrac{\dfrac{\dfrac{}{\mathfrak{A} \supset \bot, \mathfrak{A} \vdash \bot}\text{公理}}{\mathfrak{A} \supset \bot, \mathfrak{A} \vdash \mathfrak{B}}\text{爆発則}}{\mathfrak{A} \supset \bot \vdash \mathfrak{A} \supset \mathfrak{B}}\supset\text{導入則}}{(\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A}, \mathfrak{A} \supset \bot \vdash \mathfrak{A}}\supset\text{除去則}}{(\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A}, \mathfrak{A} \supset \bot \vdash \bot}\supset\text{導入則}}{(\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A} \vdash (\mathfrak{A} \supset \bot) \supset \bot}\text{二重否定除去則}}{(\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A} \vdash \mathfrak{A}}\supset\text{導入則}}{\vdash ((\mathfrak{A} \supset \mathfrak{B}) \supset \mathfrak{A}) \supset \mathfrak{A}}$$

「$\mathfrak{A}_1, ..., \mathfrak{A}_n \vdash \mathfrak{B}$」(ただし $0 \leq n$) という表現はシーケントと呼ばれ、命題の集合「$\mathfrak{A}_1, ..., \mathfrak{A}_n$」を構成するすべての命題が仮定されたとき「$\mathfrak{B}$」という命題が結論されるという仮言的判断を意味する。各規則は、「水平線の上の判断が成り立つとき、水平線の横に表示された推論規則によって水平線の下の判断が成り立つ」という意味である。NJ と NK の違いは、推論規則に二重否定除去則を前者は含まない一方で後者は含む、という点のみにある。つまりパースの法則の NK-証明図中の二重否定除去則より上の証明図は NK-証明図でありかつ NJ-証明図である。ポイントは、シーケントたちが形成する木構造によって、証明図の一番下のシーケント (終式と呼ばれる) に対しては、それが表す判断へと至る証明プロセスが与えられるということである。

さて 1960 年代に明らかとなったのは、NJ と型付き λ 計算と呼ばれるプログラム構築体系との間に、ある同型性が与えられるという事実である [1][6]。この対応は Curry-Howard 対応 (以下 CH 対応) と呼ばれる。

まず λ 計算とは、計算機科学・情報科学の分野において計算機やプログラムの一般的性質に関する理論的研究を行うために、プログラムの構築とその実行という概念を記号列表現の形成とそれに対する変形操作という観点から抽象化した形式体系である。特に、各 λ 式に対して型の概念が与えられている λ 計算が型付き λ 計算と呼ばれる。型付き λ 計算の基本的な定義は次の通りである。変数記号として「$x, y, z, ...$」、束縛演算子として「λ」、型変数記号として「$A, B, C, ...$」、型演算子として「$\to$」、補助記号として「$), (, .$」を用いる。また以下で x は任意の変数記号ちょうど 1 文字を表し、$\mathcal{A}$ は任意の型変数記号ちょうど 1 文字を表す。

- λ 式の定義　$\mathsf{L}, \mathsf{M}, \mathsf{N} ::= \quad \mathsf{x} \quad | \quad \lambda\mathsf{x}.\mathsf{L} \quad | \quad \mathsf{MN}$
- 型の定義　$\alpha, \beta ::= \quad \mathcal{A} \quad | \quad \alpha \to \beta$
- 簡約規則の定義　$(\lambda\mathsf{x}.\mathsf{M})\mathsf{N} \triangleright_\beta \mathsf{M}[\mathsf{N}/\mathsf{x}]$
  ある λ 式中の「$\triangleright_\beta$」の左辺の形をした部分式を可約部と呼ぶ。また左辺の

形の λ 式を変形して右辺の形の λ 式を得ることを β 変換と呼ぶ。$\mathsf{M}[\mathsf{N}/\mathsf{x}]$ という表現は M 中の自由な x をすべて N で置き換えた表現を表す。

- λ 式型付け規則の定義

$$\frac{}{\mathsf{x}:\tau,\Gamma\vdash\mathsf{x}:\tau}\,id \qquad \frac{\Gamma\vdash\mathsf{L}:\tau}{\mathsf{x}:\alpha,\Gamma\vdash\mathsf{L}:\tau}\,wk \qquad \frac{\mathsf{y}:\alpha,\mathsf{x}:\alpha,\Gamma\vdash\mathsf{L}:\tau}{\mathsf{x}:\alpha,\Gamma\vdash\mathsf{L}[\mathsf{x}/\mathsf{y}]:\tau}\,ctr$$

$$\frac{\mathsf{x}:\alpha,\Gamma\vdash\mathsf{L}:\beta}{\Gamma\vdash\lambda\mathsf{x}.\mathsf{L}:\alpha\to\beta}\,FUN \qquad \frac{\Gamma\vdash\mathsf{M}:\alpha\to\beta \qquad \Delta\vdash\mathsf{N}:\alpha}{\Gamma,\Delta\vdash\mathsf{MN}:\beta}\,APP$$

「$\mathsf{x}_1:\alpha_1,...,\mathsf{x}_n:\alpha_n\vdash\mathsf{M}:\tau$」(ただし $0\leq n$) という表現は、「変数 $\mathsf{x}_1,...,\mathsf{x}_n$ の型がそれぞれ $\alpha_1,...,\alpha_n$ であるとき、λ 式 M の型は τ である」という型判断を意味する。各規則は、「水平線の上の型判断が成り立つとき、水平線の下の型判断が水平線の横に表示された λ 式型付け規則によって成り立つ」という意味である。また大文字のギリシア文字は $\mathsf{x}_1:\alpha_1,...,\mathsf{x}_n:\alpha_n$(ただし $0\leq n$) を表す。

ポイントは、定義により $\lambda\mathsf{x}.\mathsf{M}:\alpha\to\beta$ という λ 式は「型 α の λ 式が右から入力として結合したら (すなわち $\mathsf{N}:\alpha$ として、$(\lambda\mathsf{x}.\mathsf{M})\mathsf{N}$ という λ 式が形成されたら) 型 β の λ 式 $\mathsf{M}[\mathsf{N}/\mathsf{x}]$ を出力として返すような関数」を表現しているという点である。つまり、λ 式型付け規則によって型を与えられる λ 式は、まさにその型が表す仕様通りの振る舞いをする関数となっているということである。具体例をとろう。自然数および自然数上の後者関数を定め、後者関数の計算を表現することを考えよう。自然数を定める一つのやり方は、型 nat を $nat ::= (A\to A)\to(A\to A)$ で定め、さらに型 nat を持つ λ 式を、$y:A\to A$、$x:A$ として $0 ::\equiv \lambda y.\lambda x.x$、$1 ::\equiv \lambda y.\lambda x.yx$、$2 ::\equiv \lambda y.\lambda x.y(yx)$、...、$n ::\equiv \lambda y.\lambda x.y(\cdots(y(yx))\cdots)$ と定めるというものである [2]。さらにこれを踏まえ後者関数 suc を、$z:nat$、$y:A\to A$、$x:A$ として $suc ::\equiv \lambda z.\lambda y.\lambda x.y((zy)x)$ で定めれば、型付き λ 計算で後者関数の計算を実現できる。例えば $suc0$ から 1 を得る計算は、$suc0\equiv(\lambda z.\lambda y.\lambda x.y((zy)x))\lambda y.\lambda x.x\rhd_\beta\lambda y.\lambda x.y(((\lambda y.\lambda x.x)y)x)\rhd_\beta\lambda y.\lambda x.y((\lambda x.x)x)\rhd_\beta\lambda y.\lambda x.yx\equiv 1$ のようになる。重要な点は、nat を上のように定義したときに、後者関数 suc に当たる λ 式が意図した通り nat 上の 1 変数関数となっていることが次の λ 式型付け規則の連鎖的表現 (型付けの図式と呼ばれる) によって保証されるということである。

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y:A\to A\vdash y:A\to A}\,id \qquad \dfrac{\dfrac{\dfrac{}{z:nat\vdash z:nat}\,id \qquad \dfrac{}{y:A\to A\vdash y:A\to A}\,id}{z:nat,y:a\vdash zy:A\to A}\,APP \qquad \dfrac{}{x:A\vdash x:A}\,id}{z:nat,y:a,x:A\vdash(zy)x:A}\,APP}{z:nat,y:a,x:A\vdash y((zy)x):A}\,APP}{z:nat,y:a\vdash\lambda x.y((zy)x):A\to A}\,FUN}{z:nat\vdash\lambda y.\lambda x.y((zy)x):nat}\,FUN}{\vdash\lambda z.\lambda y.\lambda x.y((zy)x):nat\to nat}\,FUN$$

型付けの図式は NJ-証明図とそっくりである。特に FUN と APP の図式中の型の配置は、それぞれ先ほどの証明図の $\supset$ 導入則 と $\supset$ 除去則 の図式の論理式の配置と (書体の違いを無視し型演算子 $\rightarrow$ と論理結合子 $\supset$ とを同一視すれば) 一致することがわかる。CH 対応とは、型付けの図式と NJ-証明図の間に見られる、この対応関係のことに他ならない（なお $\supset$ 以外の結合子についても特定の演算子を計算体系に導入し型付け規則を定めて対応を与えることができる）[1][6]。ただしこの対応は単なる類比ではない。CH 対応は、BHK 解釈によって NJ の推論規則に見出される構成の概念と、型付き λ 計算の型付け規則の λ 式形成手続き (すなわち素材となる λ 式の構造を保存しながら型の制約に則って新たな λ 式を構築する手続き) が持つ構成の概念とが同型だということを明らかにしているのである [1][6]。NJ-証明図と対応する型付け図式によって構築される λ 式は、まさにその証明図の終式の結論の BHK 解釈の具体的な形式を与えていると捉えられるのである。ところで CH 対応についてもう一つポイントがある。察しの通り、実は Gentzen の証明図は型付き λ 計算での β 変換と対応する証明図変形機構を備えている [1][4][6]。この変形は―紙幅の都合によりここでは詳細を述べられないが、ともかく―余剰な推論規則の適用箇所 (証明の回り道と呼ばれる) を除去し証明を正規化する操作にあたる。NJ-証明図と対応する型付け図式によって構築される λ 式は、BHK 解釈の体現であると同時に、証明の正規化の手順を与えているのである。[3]

さて近年、CH 対応の「証明に対してその正規化手順を与える」という側面を CL へと拡張する方法が明らかとなった。これを初めて定式化したのは 1990 年の Griffin の論文 [5] である。ここで明らかとなったのは、型付き λ 計算に call/cc と呼ばれる継続呼び出し演算子を導入した拡張体系で構築される式が、NK(詳しくは NK と同等な古典自然演繹) の正規化手順を与えるということであった。

継続 (continuation) とは、ひとつの λ 式のある 1 箇所または 0 箇所の可約部に焦点を当てたときにその可約部にとっての背景となる (いわば無視される)、その λ 式中のその可約部を除いた部分を指す概念である。「計算の残りの部分」とも呼ばれる。たとえば、先の $suc0$ を表す λ 式の計算過程で現れた λ 式 $\lambda y.\lambda x.y(((\lambda y.\lambda x.x)y)x)$ について、$\lambda y.\lambda x.y(\underset{\star}{((\lambda y.\lambda x.x)y)}x)$ の★で示した可約部に焦点を当てたとき、$\underbrace{\lambda y.\lambda x.y((}_{\heartsuit}\underbrace{\underset{\star}{(\lambda y.\lambda x.x)y}}_{\star}\underbrace{)x)}_{\heartsuit}$ の ♡ で示した部分が★にとっての継続である。$\lambda y.\lambda x.y(((\lambda y.\lambda x.x)y)x)$ について、0 箇所の可約部すなわち式全体に焦点を当てたときは、空な表現がその可約部にとっての継続である。簡約における可約部と継続の移り変わりは、先の $suc0$ についての簡

約を例にとれば、$\underbrace{\overbrace{(\lambda z.\lambda y.\lambda x.y((zy)x))\lambda y.\lambda x.x}^{\text{可約部}}}_{\text{継続}} \triangleright_\beta \underbrace{\lambda y.\lambda x.y(}_{\text{継続}}\overbrace{(\lambda y.\lambda x.x)y}^{\text{可約部}}\underbrace{)x)}_{\text{継続}} \triangleright_\beta \underbrace{\lambda y.\lambda x.y(}_{\text{継続}}\overbrace{(\lambda x.x)x}^{\text{可約部}}\underbrace{)}_{\text{継続}} \triangleright_\beta \underbrace{\lambda y.\lambda x.yx}_{\text{継続}\ (0\ \text{箇所の})\ \text{可約部}}$ のようになる。さて継続呼び出しとは、「焦点が当たっている可約部の内部の特定の継続表現をその可約部にとっての継続で置き換える操作」に当たる。β 変換が「ある λ 式の内部の特定の変数をその式に右から入力として結合している λ 式で置き換える」操作であることを踏まえると、継続呼び出しとは言わば β 変換の裏返しである。つまり継続呼び出しを含む計算とは、仕組みとしては、$\underbrace{(\lambda \mathsf{x}.\mathsf{L})}_{\text{継続}_A}(\overbrace{(\lambda \mathsf{y}.\mathsf{M})(\underbrace{\ldots}_{\text{継続}_B}\underbrace{\mathsf{N}}_{\text{可約部}_B}\underbrace{\ldots}_{\text{継続}_B})}^{\text{可約部}_A}\underbrace{)}_{\text{継続}_A}$ という形の λ 式の 可約部$_A$ の内部の 継続$_B$ を 継続$_A$ で置き換えて $(\lambda \mathsf{y}.\mathsf{M})((\lambda \mathsf{x}.\mathsf{L})\mathsf{N})$ という λ 式を得るような「計算の順序を変更するような計算」である。

つまるところ Griffin により、CL の証明プロセスは、証明中の背景や文脈に相当する部分の補足・制御を含むような、IL よりも複雑な証明正規化手順を与える式についての―しかしそれが BHK 解釈の体現とはなっていない式についての―構築プロセスと対応付けられるということが明らかとなった。

3 筆者の主張

BHK 解釈の条件が崩れている CL に対して上のような CH 対応の拡張が与えられるという事実は、次の見方を導くと筆者は考える。それは、まず IL についての CH 対応においては証明によって構築される λ 式は、第一に BHK 解釈を体現し、第二に証明の正規化手順を与えるのであったが、実はこの二つのことが本来は別のことであり、そして CL の CH 対応においては後者のみが実現している、という見方である。CL によって導出される論理式に対しては、一方で確かに BHK 解釈を与えることはできないのであるが、しかし他方で構築される式が証明正規化手順を与えているという点では単なる二値性に帰着できないような証明の構造に関する概念 [4] をその表現内容としていると見ることができよう。

さて Griffin によって明らかとなったのは、CL で導出される論理式が表現内容としている事柄すなわち証明正規化手順の概念は、IL の証明正規化の際には扱われない証明の文脈・背景に対する操作を含んでいるという事実であったが、さらに近年鮮明となってきたのは、その正規化手順が (IL では失われているような) 対称性を備えているという事実である。特に、CL と CH 対応する体系である Wadler[8] の双対計算 (dual calculus) と Filinski[3]・浅井ら [11][9] の対称 λ 計算 (symmetric lambda calculus、以下 SLC) では、可約部とその操作を定める構文的道具立てと、継続とその操作を定める構文的道具立てとが

すべて互いに対称的な形で与えられ、証明正規化手順を表現する式はそれら対称な両者の結合表現として構築される。つまり、CL によって導出される論理式が表現内容とする証明正規化手順には、IL において扱われる証明に対する抽象／適用操作に加えて、これと対称的な証明の文脈・背景に対する抽象／適用操作が組み込まれているという事実が鮮明となった。この対称性を如何に解釈するかについては様々な余地があると考えられるが、一つのやり方としては、CL により導出される論理式は IL の BHK 解釈的な内容とその双対すなわち「ある論理式が反証についての諸仮定からどのような実効的手続きを経て反証されたか」という BHK 解釈の双対的内容を部分的に併せ持っていると解釈することができるはずだと筆者は考える。そして特にこの点で、⊃とその双対の機能を持つ結合子とさらにそれらについての両義性の機能を持つ結合子の三者が組み込まれていることを特徴とする SLC[11][9] が構築する式は、CL の証明プロセスが構築する証明正規化手順が証明の構造と反証の構造の並存やそれらの間の相互作用をまさに表現していると筆者は考える。

4　今後の展望

SLC が構築する式の形を踏まえて、CL により導出される論理式が表現する証明の構造に関する内容が BHK 解釈的な意味とどのような隔たりを持ったものであるのかを明らかにし、そしてここで CL の命題が持つ本質的な概念とは何かを特定することが今後の課題である。また筆者による研究の途上ではあるが、SLC に対してある拡張―詳しくは〈継続 | 関数 | 項〉という構文的構造の入れ子を許すという拡張―を行った体系 入れ子対称 λ 計算 (nested symmetric lambda calculus) は、これまで与えられていなかった SLC の述語論理の CH 対応への拡張を与えられる体系であると見込まれる。この体系の整備を進め述語論理レベルの証明の分析も行う予定である。

注

[1] ここでは説明のしやすさの都合により、自然演繹を Gentzen[4] のオリジナルの記法ではなくシーケントスタイルと呼ばれる記法によって表している。

[2] 何故このような形の λ 式によって自然数を定めることができるかについては、[10] で論じられている。

[3]CH 対応に対しては「論理の内実を計算が明らかにしている」という見方と「計算の内実を論理が明らかにしている」という見方のいずれもが与えられると考えられるが、本稿では計算論理学の分野で広く認められる作業仮設に従って前者の見方のみを取り上げて議論を進めている。

[4]Frege や Frege を踏まえている Dummett であればこの概念を意義 (Sinn) と呼ぶだろう。

文献

[1] Curry,H.B. and Feys,R. (1958) *Combinatory Logic*, Studies in Logic and the Foundations of Mathematics, Vol.I, 1st edition, Amsterdam: North-Holland

[2] Dummett,M (1973) The Philosophical Basis of Intuitionistic Logic. *Truth and Other Enigmas*, Cambridge: Harvard UP, 1978, pp.215-247

[3] Filinski,A (1989) Declarative continuations and categorical duality. Master's thesis, DIKU Seport 89/11, University of Copenhagen

[4] Gentzen,G (1935) Investigations into Logical Deduction. *Mathematische Zeitschrift 39*, pp.176-210,405-431. Reprinted in M. E. Szabo, editor *The Collected Papers of Gerhard Gentzen*, North-Holland, 1969

[5] Griffin,T (1990) A Formulae-as-Types Notion of Control. *17'th symposium on Principles of Programming Languages (POPL'90)*, pp.47-58

[6] Howard,W.A. (1969) The Formulae-as-Types Notion of Construction. *To H.B.Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, pp.479-490

[7] Troelstra,A.S. (1991) *History of constructivism in the twentieth century*, ITLI Prepublication series for Mathematical Logic and Foundations ML-91-05, University of Amsterdam

[8] Wadler,P (2003) Call-by-value is dual to call-by-name. *Proceedings of the eighth ACM SIGPLAN International Conference on Functional Programming (ICFP'03)*, pp.189–201

[9] 上田 やよい、浅井 健一 (2010)「型付き対称 λ 計算と古典論理」 第 12 回プログラミングおよびプログラミング言語ワークショップ論文集, pp.34-48

[10] 岡本 賢吾 (2003) 「命題を集合と同一視すること——包括原理からカリー＝ハワード対応へ」 『科学哲学』 第 36 巻 2 号, pp.103-118

[11] 阪上 紗里、浅井 健一 (2009)「対称 λ 計算の基礎理論」 『コンピュータソフトウェア』 Vol.26 No. 2, pp.3-17

[12] マイケル・ダメット (1973)「直観主義論理の哲学的基底」 藤田 晋吾 訳 (1986) 『真理という謎』 勁草書房 pp.212-266

（東京都立大学）