型付き対称λ計算と古典論理

Symmetric λ-Calculus and Classical Logic

上田 やよい　　浅井 健一

ueda.yayoi@is.ocha.ac.jp　　asai@is.ocha.ac.jp

お茶の水女子大学

平成 22 年 2 月 13 日

概 要

対称λ計算 (Symmetric λ-Calculus, SLC) [Filinski 1989] は、項と継続が完全に対称な形をしており、項を扱うのと全く同じ様に継続を扱うことができる。阪上らの型付き対称λ計算 [2008] では、簡約規則を small step semantics で定式化し直し、λ_C 計算 [Felleisen and Hieb 1992] や $\lambda\mu$ 計算 [Parigot 1992] を SLC に変換できることが示された。

上田の先行研究 [2009] では、SLC の型に論理積と論理和を導入した。しかしこの体系では、型として論理式を与えているものの、base case が整数型であるために論理学的な意味を考えることが難しかった。

本研究では、SLC の型定義に false (⊥) 型を加えることで、SLC の型規則が古典論理学に対応していることを示す。またその一例として排中律や二重否定の型を持つ項を導出する。

1　序論

計算理論において、λ計算と直観主義論理が対応しており、カリーハワード同型が示せることは周知の事実である。しかし古典論理と計算理論の関係は、長らく解明されなかった。

最初に古典論理とのカリーハワード同型を示したのは Griffin である。1990 年に [4] において call/cc が Peirce's Law に、コントロールオペレータ $\mathcal{C}$ が二重否定の法則と対応することが示された。その後 Parigot が λ_μ 計算と古典論理の自然演繹が対応することを示した [6]。さらに古典論理の sequent 計算と対応する体系を Curien, Herbelin [1] や Wadler[10, 11] が提唱した。

一方、近年では例外処理など非局所的な制御を扱うために「プログラムの残りの計算」を表す継続が使われるようになっている。CPS 変換や評価文脈などの手法が提案されてきているが、CPS 変換はプログラムの構造が大きく変化し、また評価文脈はプログラムの構造自体は大きく変えないが「評価文脈」という新しい概念を扱えるようにする必要がある。

残りの計算である継続と、現在計算している項とを統一的に扱う理論としては、Fillinski の対称λ計算 (Symmetric λ-calculus, SLC) がある [3]。阪上らは SLC を small-step で定式化し直し、Felleisen の λ_C 計算 [2] と Parigot の λ_μ 計算 [6] が SLC に変換できることを示した。上田らの先行研究 [9] では阪上らの SLC に対して論理和型と論理積型を導入した。SLC では関数が対偶の型を同時に持っていることが特徴的である。このため SLC は古典論理だと思われてきたが、その証明は成されていなかった。

本研究では、まず SLC の定式化をし、Progress, Preservation の証明や Call-By-Value の導入及びその簡約の一意性、停止性などの性質を紹介する。さらに SLC に ⊥ 型を導入した体系 SLC$_\perp$ について、その型規則を古典論理で表すことができ、健全であることを示す。その上で SLC$_\perp$ の型規則を用いて、古典論理の形式体系 NK の推論規則を `RAA` (背理法) を含めて表現できることを示す。さらに、二重否定の除去 ($\neg\neg A \supset A$) や排中律 ($A \vee \neg A$) を実際に SLC$_\perp$ で導出する。

本論文の構成は以下の通りである。まず 2 章では古典論理と形式体系 NK を紹介する。3 章では SLC の定式化及び Call-By-Value の導入と停止性などの各種性質の紹介をする。4 章では SLC に対して型 ⊥ を加えた SLC$_\perp$ を導入する。5 章では SLC$_\perp$ の型規則を古典論理で表し、SLC$_\perp$ の健全性を示す。最後に 6 章では SLC$_\perp$ で古典論理の推論規則が表せることを示し、その例として二重否定や排中律を実際に SLC$_\perp$ で導出する。

$$A, B, C ::= \bot \mid X \mid \neg A \mid A \wedge B \mid A \vee B \mid A \supset B$$

図 1: 古典論理式の構文

2 古典論理

古典論理の論理式は図 1 の定義で表される。

A、B、C を任意の論理式とする。X は命題変数である。$\neg A$ は A の否定を表す。$A \wedge B$ は A と B の論理積である。$A \vee B$ は A と B の論理和である。$A \supset B$ は含意であり「A ならば B」と読む。各論理演算子は図 1 で左にあるものほど結合力が強い。

推論規則として、古典論理を自然演繹で表した形式体系 NK を用いる。図 2 にその推論規則を示した。ここでは [5] による定式化を用いている。

仮定となる論理式の集合を Γ で表し、そのもとで論理式 A が成り立つとき $\Gamma \vdash A$ と表す。Γ は空集合でもよい。また A とは関係のない論理式を含んでいても構わない。図 2 の最下段は仮定 Γ に関する規則である。Weak は B とは関係のない仮定 A を加える規則、Cont は仮定に同じものが含まれる場合に片方を消去する規則である。Γ を集合として扱うため、Interchange の規則はない。$\Gamma, A \vdash B$ では A の位置は関係なく、仮定に A が含まれていればよい。

各規則は「上段が成り立つならば、下段が推論される」と読む。特に Axm は上段に何もないので、前提なしに $\Gamma, A \vdash A$ を導くことができる。

以下、各規則について注釈を加える。

Axm は、仮定に論理式 A が含まれているならば、A は無条件に成り立つとする規則である。古典論理の証明の base case となる規則である。

$\wedge$I は、同じ仮定 Γ のもとで A と B が成り立つなら、Γ のもとで論理積 $A \wedge B$ が成り立つと推論する規則である。$\wedge\mathrm{E}_1$ はその逆で、$A \wedge B$ が成り立つのなら A も B も真であるはずなので、A が成り立つと推論する規則である。$\wedge\mathrm{E}_2$ も 同様である。

$\vee\mathrm{I}_2$ は、A が成り立つならば任意の論理式 B について $A \vee B$ が成り立つと推論する規則である。$\vee\mathrm{I}_1$ も同様である。$\vee$E は、ある仮定 Γ のもとで $A \vee B$ が成り立ち、かつ Γ と A のもとで C が、Γ と B のもとで C が成り立つならば、Γ のみのもとで C が成り立つと推論する規則である。

$\supset$ I は、ある仮定のもとで B が成り立つならば、仮定に含まれる任意の論理式 (図 2 では A) を取り出し、残りの仮定 (図 2 では Γ) のもとで $A \supset B$ が成り立つと推論する規則である。$\supset$ E は、同じ仮定 Γ のもとで $A \supset B$, A が成り立つならば B が成り立つと推論する規則である。モーダスポネンスとも呼ばれる。

$$\frac{}{\Gamma, A \vdash A}\ \mathtt{Axm}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\mathtt{I} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\mathtt{E}_1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\mathtt{E}_2$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\mathtt{I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\mathtt{I}_2$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\mathtt{E}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\mathtt{I} \qquad \frac{\Gamma \vdash A \supset B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\mathtt{E}$$

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \mathtt{RAA} \qquad \left(\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\mathtt{E}\right)$$

$$\frac{\Gamma \vdash B}{\Gamma,\ A \vdash B}\ \mathtt{Weak} \qquad \frac{\Gamma,\ A,\ A \vdash B}{\Gamma,\ A \vdash B}\ \mathtt{Cont}$$

図 2: 古典論理の形式体系 NK

RAA は、推論の仮定で矛盾が生じた場合、仮定に含まれる $\neg A$ を消去し、A を推論する規則である。一般には背理法と呼ばれ、「$\sqrt{2}$ は無理数」の証明に使われる背理法と全く同じものである。

() をつけて書かれている規則 $\bot$E は、直観主義論理の推論規則である。古典論理は「直観主義論理の体系に背理法 (RAA) を加えたもの」が定義であるため、厳密には $\bot$E も古典論理の導出規則に含まれる。しかし実際には、$\bot$E の上段の仮定に Weak で $\neg A$ を加えると RAA と全く同じ推論を表していると言えるので、本論文では $\bot$E を RAA で代用する。(余談であるが、RAA を $\bot$E で代用することはできない。またその点が古典論理と直観主義論理の違いでもある。)

論理演算子 $\neg A$ は、適宜矛盾を表す $\bot$ を用いて $A \supset \bot$ と読み替え、$\supset$ I 及び $\supset$ E を用いて推論する。$\supset$ I において B を $\bot$ と考えると、ある仮定のもとで矛盾 $\bot$ が導かれたら仮定のどれかが間違っている、つまり $A \supset \bot$ ($\neg A$) を推論する。$\supset$ E でも同様に B を $\bot$ と考え、同じ仮定のもとで A と $A \supset \bot$ ($\neg A$) が導かれたら矛盾する、つまり $\bot$ を推論する。

$$
\begin{array}{rl}
\neg(A \wedge B) \equiv \neg A \vee \neg B & \text{ドモルガンの法則}_1 \\
\neg(A \vee B) \equiv \neg A \wedge \neg B & \text{ドモルガンの法則}_2 \\
A \supset B \equiv \neg B \supset \neg A & \text{対偶律} \\
A \supset (B \supset C) \equiv (A \wedge B) \supset C & \text{移入律} \\
A \equiv \neg\neg A & \text{二重否定の法則} \\
A \supset A & \text{同一律} \\
A \vee \neg A & \text{排中律}
\end{array}
$$

図 3: 恒真式の例

$$
\begin{array}{rcl}
 & & \lceil x \Rightarrow x \rceil \uparrow (\lceil g \rceil \Rightarrow 1 \uparrow g) \\
(begin) & \rightsquigarrow & \langle\ \lceil x \Rightarrow x \rceil \uparrow (\lceil g \rceil \Rightarrow 1 \uparrow g) \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{pop}) & \rightsquigarrow & \langle\ \lceil x \Rightarrow x \rceil \mid \lceil g \rceil \Rightarrow 1 \uparrow g \mid \bullet_{\mathsf{int}}\ \rangle \\
(\beta_f) & \rightsquigarrow & \langle\ 1 \uparrow (x \Rightarrow x) \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{pop}) & \rightsquigarrow & \langle\ 1 \mid x \Rightarrow x \mid \bullet_{\mathsf{int}}\ \rangle \\
(\beta) & \rightsquigarrow & \langle\ 1 \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{end}) & \rightsquigarrow & 1
\end{array}
$$

図 5: SLC 実行例

2.1 恒真式

図 2 で推論される論理式のうち Γ が空のものを恒真式と呼ぶ。$A \vee \neg A$、$\neg\neg A \supset A$ などがそれにあたる。本論文で使用する恒真式を図 3 に示した。$\equiv$ は「同値」を表す論理演算子で、$A \equiv B$ の定義は $(A \supset B) \wedge (B \supset A)$ である。ここでは、左右の論理式が等価であることを意味するために使用している。

恒真式に関して、次の補題を特に断りなく使用する。

Lemma 2.1 論理式 $C[A]$（C 中に A が現れる）について、$A \equiv B$ ならば、C に現れる任意の A を B に置き換えた論理式 $C[B]$ は、$C[A] \equiv C[B]$ を満たす。

3 型付き対称λ計算

対称λ計算 (Symmetric λ-Calculus, SLC) は「項 e」「関数 f」「継続 c」から成り、項と継続を双対的に計算する計算言語理論である。その型 (図 4) も双対的に定義されており、項を $+T$ 型と表すのに対して、それを受け取る継続を $\neg T$ 型と表す。関数は項と継続をつなぐ中立の型で、論理学での対偶の関係にある型を同時に持つ。Non-Deterministic SLC では実行規則も完全に対称形である。本論文では阪上の体系 [8] に論理積型と論理和型、すなわち項の組と継続の組を導入したものを扱っている。

SLC では項と継続を双対的に計算するため、実行の際には現在の「項」「継続」と注目している「関数」を保持している必要がある。そのため、SLC の計算状態は $\langle$ 項 | 継続 $\rangle$ の二つ組か $\langle$ 項 | 関数 | 継続 $\rangle$ の三つ組で記述される。

$$
\begin{array}{rcll}
S & ::= & +T & (\text{項 } e \text{ の型}) \\
 & \mid & \begin{smallmatrix}+A \to +B \\ \neg A \leftarrow \neg B\end{smallmatrix} & (\text{関数 } f \text{ の型}) \\
 & \mid & \neg T & (\text{継続 } c \text{ の型}) \\
T, A, B & ::= & \mathsf{int} \mid A \wedge B \mid A \vee B \mid A \to B \mid A - B &
\end{array}
$$

図 4: 型付き対称λ計算 SLC 型定義

図 5 に SLC 式の例とその実行例を示した。一行目は $(\lambda f.f1)\lambda x.x$ を SLC の項で表したものである。λ抽象 $\lambda f.f1$ は SLC の項の関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ に、$\lambda x.x$ は $\lceil x \Rightarrow x \rceil$ に対応する。g は SLC での関数用の変数で、λ計算では一般に変数 f で表されるものである。$\uparrow$ は SLC での項の関数適用を表している。$1 \uparrow g$ はλ計算で変数 f に 1 を渡す $f1$ に、$\lceil x \Rightarrow x \rceil \uparrow (\lceil g \rceil \Rightarrow 1 \uparrow g)$ はλ抽象 $\lambda f.f1$ に $\lambda x.x$ を渡す $(\lambda f.f1)\lambda x.x$ に相当する。なお、SLC では関数適用の関数と引数の位置が入れ替わっている。

二行目では、一行目の項と初期継続 $\bullet_{\mathsf{int}}$ で SLC の計算状態の二つ組を作っている。$\bullet_{\mathsf{int}}$ は直感的には空の継続に相当する。三行目では、項から関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ を取り出して注目し、関数を中心にその時を項を右側に、継続を左側に分けている。四行目では関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ のβ簡約が起こり、項にあった関数 $x \Rightarrow x$ が変数 g に代入されている。五行目では、再び項から関数を取り出している。取り出された関数 $x \Rightarrow x$ は恒等関数であるので 1 をそのまま返す。

3.1 構文

型付き対称λ計算の構文は、項 e、継続 c、関数 f から成り、$n, \bullet_T$ を除くと関数を中心とする対称形である (図 6)。他のλ計算に比べて関数が項から独立している点が特徴的であり、それに伴う構文や規則が存在する。

例えば、通常のλ計算では関数適用を $M\ M$ のように式を並べて書くが、SLC では $e \uparrow f$ のように、関数 f に項 e を渡すと記述する。従って、関数に関数を渡したい場合には、$\lceil f \rceil$ のように関数にタグをつけ、一時的に項として扱って関数に渡す。受け取った

$$
\begin{array}{rcl}
e & ::= & n \mid x \mid (e,\, e) \mid \lceil f \rceil \mid e \uparrow f \\
f & ::= & g \mid x \Rightarrow e \mid (x_1,\, x_2) \Rightarrow e \mid \lceil g \rceil \Rightarrow e \mid \overline{e} \mid \\
 & & h \mid c \Leftarrow y \mid c \Leftarrow (y_1,\, y_2) \mid c \Leftarrow \lfloor h \rfloor \mid \underline{c} \\
c & ::= & \bullet_T \mid y \mid (c,\, c) \mid \lfloor f \rfloor \mid f \downarrow c \\
 & & \\
\mathsf{var} & ::= & x \mid y \mid g \mid h
\end{array}
$$

図 6: SLC 構文

$$1 \uparrow \overline{\lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x)}$$

図 7: SLC 式の例

関数を項ではなく関数として使えるようにするには、$\lceil g \rceil \Rightarrow e$ のように関数の仮引数部分にパターンマッチを書く。e の中で変数 g を用いることで、渡された関数を使えるようになる。図 5 では、これらの式を用いて関数 $x \Rightarrow x$ を関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ の本体の中で使えるようにしている。

さらにこの構文では、関数適用 $e \uparrow f$ の f の部分に関数適用をそのまま書くことは出来ないため、関数を返す項 e を一時的に関数として扱うためのタグ $\overline{e}$ を導入している。後述する実行規則 (exc) (3.3 節) を用いて、必要なときに e の実行に移る。これらを用いた SLC 式の例を図 7 に示した。

以下、SLC の構文について注釈を加える。

SLC の構文は、項 e、関数 f、継続 c から成る。

項 e は、整数 n、変数 x、項の組 $(e,\, e)$、関数の項扱い $\lceil f \rceil$、関数適用 $e \uparrow f$ から成る。関数 f には、関数用の変数 g、λ 抽象 $\lambda x.M$ に相当する $x \Rightarrow e$、それに関数のパターンマッチをつけた $\lceil g \rceil \Rightarrow e$、項の組のパターンマッチをつけた $(x_1,\, x_2) \Rightarrow e$、項の関数扱い $\overline{e}$ が含まれる。

ここまでが項に関する構文である。以下は継続についての構文であり、項の構文と対称形を成している。

継続 c は、初期継続 $\bullet_T$、変数 y、継続の組 $(c,\, c)$、関数の継続扱い $\lfloor f \rfloor$、関数適用 $f \downarrow c$ から成る。初期継続 $\bullet_T$ には、任意の型 T を与えてよい。継続の関数適用は、項の関数適用とは関数と引数の位置が逆になっており $\downarrow$ で表される。継続を扱う関数として、変数 h、λ 抽象 $x \Rightarrow e$ の双対にあたる $c \Leftarrow y$、それに関数のパターンマッチをつけた $c \Leftarrow \lfloor h \rfloor$、継続の組のパターンマッチをつけた $c \Leftarrow (y_1,\, y_2)$、継続の関数扱い $\underline{c}$ がある。継続を扱う関数 $c \Leftarrow y$ は、項を扱う関数とは逆に、仮引数 y を右側に、関数の本体の継続 c を左側に書く。

簡便のため、SLC で扱う変数 $x,\, y,\, g,\, h$ をまとめて var という記号で表している。なお、関数用の変数 g と h は SLC の双対性を示すために 2 種類記述しているが、どちらも関数 f を表す変数なので実質的には同じものである。

3.2 型規則

SLC の型は図 4 のように定義される。SLC の型には大きく 3 種類あり、項の型 $+T$、継続の型 $\neg T$、関数の型 $\{^{+A \to +B}_{\neg A \leftarrow \neg B}$ から成る。$+T$ は「$+T$ 型を返す項」という意味である。それに対し $\neg T$ は「$+T$ 型の項を受け取る継続」という意味を持つ。項の型についている記号 $+$ は項型であることを明示するためのもので、論理学上の意味はない。関数の型 $\{^{+A \to +B}_{\neg A \leftarrow \neg B}$ で使われている矢印 $\to$、$\leftarrow$ は、どちらも SLC 上での含意を表す。2 章の $\supset$ と意味は同じで、「$+A$ ならば $+B$」「$\neg B$ ならば $\neg A$」と読む。その意味は「$+A$ 型の項を受け取り $+B$ 型の項を返す関数」あるいは「$\neg B$ 型の継続を受け取り $\neg A$ 型の継続を返す関数」である。両者は論理学での対偶の関係にあたる。

図 4 の T などで定義されている型は、SLC の式の形を表すものである。int は整数 n の型で、SLC の型の base case である。$A \wedge B$ は項の組 $(e,\, e)$ の型であり、論理学の論理積に相当する。$A \vee B$ は継続の組 $(c,\, c)$ の型であり、論理学の論理和に相当する。$A \to B$ は関数の項扱い $\lceil f \rceil$ の型であり、論理学の含意に相当する。$A - B$ は関数の継続扱い $\lfloor f \rfloor$ の型である。$-$ は「含意の対偶の否定」を表す論理演算子で、$A - B$ は $\neg(\neg B \supset \neg A)$ と定義される。

SLC の型導出規則を図 8 に示した。これらの型規則は論理学の推論規則に則ったものである。環境 Γ のもとで式 M が型 S を持つとき、$\Gamma \vdash M : S$ と表す。Γ は変数の型定義 $\mathsf{var} : S$ の集合である。図中には示していないが、Weak と Cont は自由に用いてよい。

TVar は変数の導出規則である。SLC で扱う変数は全て、この規則により導出される。

TAnd は項の組を導出する規則、$\overline{\texttt{TOr}}$ は継続の組を導出する規則である。論理学で $\wedge$ を推論する規則に則ったものである。SLC の型定義 (図 4) より継続の型は $\neg T$ の形でなければならないので、$\overline{\texttt{TOr}}$ で導出された型は $\neg A \wedge \neg B$ からドモルガンの法則を用いて $\neg$ を前に出した形となっている。継続の組の型が $\vee$ で表現されるのはこのためである。

$$\frac{}{\Gamma \vdash n : +\mathsf{int}}\ \mathtt{TInt} \qquad \frac{}{\Gamma,\, \mathsf{var} : S \vdash \mathsf{var} : S}\ \mathtt{TVar} \qquad \frac{}{\Gamma \vdash \bullet_T : \neg T}\ \overline{\mathtt{TInit}}$$

$$\frac{\Gamma \vdash e_1 : +A \quad \Gamma \vdash e_2 : +B}{\Gamma \vdash (e_1,\, e_2) : +(A \wedge B)}\ \mathtt{TAnd} \qquad \frac{\Gamma \vdash c_1 : \neg A \quad \Gamma \vdash c_2 : \neg B}{\Gamma \vdash (c_1,\, c_2) : \neg(A \vee B)}\ \overline{\mathtt{TOr}}$$

$$\frac{\Gamma,\, x : +A \vdash e : +B}{\Gamma \vdash x \Rightarrow e : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}\ \mathtt{TFun} \qquad \frac{\Gamma,\, y : \neg B \vdash c : \neg A}{\Gamma \vdash c \Leftarrow y : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}\ \overline{\mathtt{TFun}}$$

$$\frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \quad \Gamma \vdash e : +A}{\Gamma \vdash e \uparrow f : +B}\ \mathtt{TApp} \qquad \frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \quad \Gamma \vdash c : \neg B}{\Gamma \vdash f \downarrow c : \neg A}\ \overline{\mathtt{TApp}}$$

$$\frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}{\Gamma \vdash \lceil f \rceil : +(A \to B)}\ \mathtt{TFClo} \qquad \frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}{\Gamma \vdash \lfloor f \rfloor : \neg(A - B)}\ \overline{\mathtt{TFClo}}$$

$$\frac{\Gamma \vdash e : +(A \to B)}{\Gamma \vdash \overline{e} : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}\ \mathtt{TCFun} \qquad \frac{\Gamma \vdash c : \neg(A - B)}{\Gamma \vdash \underline{c} : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}\ \overline{\mathtt{TCFun}}$$

$$\frac{\Gamma,\, x_1 : +A,\, x_2 : +B \vdash e : +T}{\Gamma \vdash (x_1,\, x_2) \Rightarrow e : \{^{+(A \wedge B) \to +T}_{\neg(A \wedge B) \leftarrow \neg T}}\ \mathtt{TPFun} \qquad \frac{\Gamma,\, y_1 : \neg A,\, y_2 : \neg B \vdash c : \neg T}{\Gamma \vdash c \Leftarrow (y_1,\, y_2) : \{^{+T \to +(A \vee B)}_{\neg T \leftarrow \neg(A \vee B)}}\ \overline{\mathtt{TPFun}}$$

$$\frac{\Gamma,\, g : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \vdash e : +T}{\Gamma \vdash \lceil g \rceil \Rightarrow e : \{^{+(A \to B) \to +T}_{\neg(A \to B) \leftarrow \neg T}}\ \mathtt{TFFun} \qquad \frac{\Gamma,\, h : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \vdash c : \neg T}{\Gamma \vdash c \Leftarrow \lfloor h \rfloor : \{^{+T \to +(A - B)}_{\neg T \leftarrow \neg(A - B)}}\ \overline{\mathtt{TFFun}}$$

図 8: SLC 型規則

$$\begin{array}{rrcl}
(begin) & e : +T & \rightsquigarrow & \langle\ e \mid \bullet_T\ \rangle \\
(left) & \langle\ (e_1,\, e_2) \mid c\ \rangle & \rightsquigarrow & \langle\ e_1 \mid x_1 \Rightarrow (x_1,\, e_2) \mid c\ \rangle \\
(right) & \langle\ (e_1,\, e_2) \mid c\ \rangle & \rightsquigarrow & \langle\ e_2 \mid x_2 \Rightarrow (e_1,\, x_2) \mid c\ \rangle \\
(\overline{pop}) & \langle\ e \uparrow f \mid c\ \rangle & \rightsquigarrow & \langle\ e \mid f \mid c\ \rangle \\
(push) & \langle\ e \mid f \mid c\ \rangle & \rightsquigarrow & \langle\ e \mid f \downarrow c\ \rangle \\
(exc) & \langle\ e \mid \overline{e'} \mid c\ \rangle & \rightsquigarrow & \langle\ e' \mid \lceil g \rceil \Rightarrow e \uparrow g \mid c\ \rangle \\
(\beta) & \langle\ e \mid x \Rightarrow e' \mid c\ \rangle & \rightsquigarrow & \langle\ e'[e/x] \mid c\ \rangle \\
(\beta_p) & \langle\ (e_1,\, e_2) \mid (x_1,\, x_2) \Rightarrow e' \mid c\ \rangle & \rightsquigarrow & \langle\ e'[e_1/x_1,\, e_2/x_2] \mid c\ \rangle \\
(\beta_f) & \langle\ \lceil f \rceil \mid \lceil g \rceil \Rightarrow e' \mid c\ \rangle & \rightsquigarrow & \langle\ e'[f/g] \mid c\ \rangle \\
(\overline{\beta_f}) & \langle\ e \mid c' \Leftarrow \lfloor h \rfloor \mid \lfloor f \rfloor\ \rangle & \rightsquigarrow & \langle\ e \mid c'[f/h]\ \rangle \\
(\overline{\beta_p}) & \langle\ e \mid c' \Leftarrow (y_1,\, y_2) \mid (c_1,\, c_2)\ \rangle & \rightsquigarrow & \langle\ e \mid c'[c_1/y_1,\, c_2/y_2]\ \rangle \\
(\overline{\beta}) & \langle\ e \mid c' \Leftarrow y \mid c\ \rangle & \rightsquigarrow & \langle\ e \mid c'[c/y]\ \rangle \\
(\overline{exc}) & \langle\ e \mid \underline{c'} \mid c\ \rangle & \rightsquigarrow & \langle\ e \mid h \downarrow c \Leftarrow \lfloor h \rfloor \mid c'\ \rangle \\
(\overline{push}) & \langle\ e \mid f \mid c\ \rangle & \rightsquigarrow & \langle\ e \uparrow f \mid c\ \rangle \\
(pop) & \langle\ e \mid f \downarrow c\ \rangle & \rightsquigarrow & \langle\ e \mid f \mid c\ \rangle \\
(\overline{right}) & \langle\ e \mid (c_1,\, c_2)\ \rangle & \rightsquigarrow & \langle\ e \mid (c_1,\, y_2) \Leftarrow y_2 \mid c_2\ \rangle \\
(\overline{left}) & \langle\ e \mid (c_1,\, c_2)\ \rangle & \rightsquigarrow & \langle\ e \mid (y_1,\, c_2) \Leftarrow y_1 \mid c_1\ \rangle \\
(\overline{end}) & \langle\ e \mid \bullet_T\ \rangle & \rightsquigarrow & e : +T
\end{array}$$

図 9: SLC 実行規則 (Non-Deterministic)

TFun は項の関数を導出する規則、$\overline{\texttt{TFun}}$ は継続の関数を導出する規則である。論理学の含意の推論に相当する。項の関数、継続の関数ともに、項中心に書かれた型 $+A \to +B$ と継続中心に書かれた型 $\neg A \leftarrow \neg B$ の両方を表示する。

TApp は項の関数適用を導出する規則である。論理学ではモーダスポネンスに相当する。TApp では関数の型を項中心に考え $+A \to +B$ 型の関数に $+A$ 型の項を渡すと $+B$ 型の項を返すと考える。$\overline{\texttt{TApp}}$ も同様だが、関数の型を継続中心に考えている点で異なる。

TFClo は関数の項扱いを導出するための規則である。関数 f に関数の項扱いを表すタグ $\lceil\ \rceil$ をつけている他、関数型 $\{^{+A\to+B}_{\neg A\leftarrow\neg B}$ から項型 $+(A \to B)$ に変わっている。TCFun はその逆をする規則である。これらの型規則は、型の形及び SLC 式の形を変えているだけである。

$\overline{\texttt{TFClo}}$ は関数の継続扱いを導出するための規則である。関数 f に関数の継続扱いを表すタグ $\lfloor\ \rfloor$ をつけている他、型が関数型 $\{^{+A\to+B}_{\neg A\leftarrow\neg B}$ から継続型 $\neg(A - B)$ に変わっている。$\neg(A - B)$ は論理演算子 $-$ の定義より $\neg\neg(\neg B \to \neg A)$ に等しい。従って、この型規則は $\neg B \to \neg A$ からその二重否定を推論する規則と言える。$\overline{\texttt{TCFun}}$ はその逆の導出を行う規則である。

TPFun は項の組のパターンマッチがついた関数を導出するための規則である。直感的には、環境に含まれる変数 $x_1 : +A$, $x_2 : +B$ で項の組 $(x_1, x_2) : +(A \wedge B)$ をつくり、それを TFun で関数の形にしたものに相当する。$\overline{\texttt{TPFun}}$ は TPFun の継続版で、継続の組のパターンマッチがついた関数を導出するための規則である。

TFFun は関数の項扱いのパターンマッチがついた関数を導出するための規則である。直感的には、環境に含まれる変数 $g : \{^{+A\to+B}_{\neg A\leftarrow\neg B}$ を TFClo で $\lceil g \rceil : +(A \to B)$ の形に変えて、TFun を適用したものに相当する。$\overline{\texttt{TFFun}}$ は TFFun の継続版で、関数の継続扱いのパターンマッチがついた関数を導出するための規則である。

TInt は整数 n を導出する規則、$\overline{\texttt{TInit}}$ は任意の型の初期継続を導出する規則である。

図 5, 図 7 に示した式は、図 8 の規則に従って型を導出することができ、どちらも $+$int 型となる。

3.3 実行規則

項と継続を Dual に計算する SLC では、その実行に際してその時点での項と継続、関数を保持している必要がある。従って SLC の計算状態は ⟨ 項 | 継続 ⟩ の二つ組か、⟨ 項 | 関数 | 継続 ⟩ の三つ組で記述される。三つ組は、中央の関数に注目して SLC 式全体を項と継続に分け、関数の実行に移ろうとしている状態を表す。二つ組は、項または継続の中から新たに中央に関数を取り出そうとしている状態を表す。

二つ組や三つ組をつくる際、項、継続、関数の型は図 10 の型規則を満たさなければならない。二つ組は $+T$ 型を持つ項 e と $+T$ 型を受け取る継続 $c : \neg T$ でつくられる。三つ組は $+A$ 型を持つ項 e と $+A$ 型の項を受け取って $+B$ 型の項を返す (または $\neg B$ 型の継続を受け取って $\neg A$ 型の継続を返す) 関数 $f : \{^{+A\to+B}_{\neg A\leftarrow\neg B}$、$+B$ 型を受け取る継続 $c : \neg B$ からつくられる。

図 9 に SLC の実行規則を示した。以下注釈を加える。

$(begin)$ は $+T$ 型を持つ項 e に対して $\neg T$ 型の初期継続 $\bullet_T$ を与え、SLC の実行を始める規則である。これは二つ組でも三つ組でもない特殊な規則で、後述する $(\overline{end})$ で取り出された項に対して用いることは許さない。

$(left)$ は項の組から左側を取り出して実行するための規則である。実行が終わったら、右辺の中央に現れている関数 $x_1 \Rightarrow (x_1, e_2)$ で元の組の形に戻す。$(right)$ は項の組の左側を取り出して実行するための規則である。$(\overline{pop})$ は項の中から関数を取り出し、その実行に移ろうとする規則である。$(push)$ は取り出した関数を実行せずに継続側に押しやり、さらに項の中を実行するための規則である。(exc) [1]は関数扱いされている項を左側に取り出し、実行しようとする規則である。実行が終わったら、右辺の中央に現れている関数で元の関数適用の形に戻す。(β) は λ 抽象の β 簡約と全く同じである。その時の項 e を変数 x で束縛し関数の本体 e' に代入する。これを図中では $e'[e/x]$ と表している。(β_p) は項の組のパターンマッチのついた関数を β 簡約するための規則である。e_1, e_2 をそれぞれ x_1, x_2 で束縛し、e' に代入する。この簡約により、組になっていたふたつの項を e' の中で別々に使えるようになる。(β_f) は関数の値扱いのパターン

$$\frac{\vdash e : +T \quad \vdash c : \neg T}{\vdash \langle\ e \mid c\ \rangle}\ \texttt{TProg1}$$

$$\frac{\vdash e : +T_1 \quad \vdash f : \{^{+T_1\to+T_2}_{\neg T_1\leftarrow\neg T_2} \quad \vdash c : \neg T_2}{\vdash \langle\ e \mid f \mid c\ \rangle}\ \texttt{TProg2}$$

図 10: SLC 計算状態

[1] $(exchange)$ の省略形

$$
\begin{array}{rl}
 & 1 \uparrow \overline{\lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x)} \\
(begin) & \rightsquigarrow \langle\ 1 \uparrow \overline{\lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x)} \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{pop}) & \rightsquigarrow \langle\ 1 \mid \overline{\lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x)} \mid \bullet_{\mathsf{int}}\ \rangle \\
(exc) & \rightsquigarrow \langle\ \lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x) \mid \lceil g \rceil \Rightarrow 1 \uparrow g \mid \bullet_{\mathsf{int}}\ \rangle \\
(push) & \rightsquigarrow \langle\ \lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x) \mid (\lceil g \rceil \Rightarrow 1 \uparrow g) \downarrow \bullet_{\mathsf{int}} \rangle \\
(\overline{pop}) & \rightsquigarrow \langle\ \lceil y \Leftarrow y \rceil \mid x \Rightarrow x \mid (\lceil g \rceil \Rightarrow 1 \uparrow g) \downarrow \bullet_{\mathsf{int}}\ \rangle \\
(\beta) & \rightsquigarrow \langle\ \lceil y \Leftarrow y \rceil \mid (\lceil g \rceil \Rightarrow 1 \uparrow g) \downarrow \bullet_{\mathsf{int}}\ \rangle \\
(pop) & \rightsquigarrow \langle\ \lceil y \Leftarrow y \rceil \mid \lceil g \rceil \Rightarrow 1 \uparrow g \mid \bullet_{\mathsf{int}}\ \rangle \\
(\beta_f) & \rightsquigarrow \langle\ 1 \uparrow (y \Leftarrow y) \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{pop}) & \rightsquigarrow \langle\ 1 \mid y \Leftarrow y \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{\beta}) & \rightsquigarrow \langle\ 1 \mid \bullet_{\mathsf{int}}\ \rangle \\
(\overline{end}) & \rightsquigarrow 1
\end{array}
$$

図 11: SLC 実行例

マッチをつけた関数を β 簡約するための規則である。引数の $\lceil f \rceil$ のタグを外した状態で関数の本体 e' 中の変数 g に代入する。e' の中では f を項ではなく関数として扱うことができるようになる。

(β_p) と (β_f) は、項の形と仮引数のパターンマッチが一致しなければ適用できない。項がパターンと一致しない場合は $(push)$ で中央の関数を継続へ押しやり、項からさらに関数を取り出して実行する。$(push)$ の代わりに $(\overline{push})$ を用いて中央の関数を項側へ押しやり、継続から新たな関数を取り出してもよい。

以上が項に関する実行規則である。残りは継続に関するものであるが、これらは全て項の実行規則を継続側に移したものである。基本的に、対応する項の規則名に $\overline{\ }$ をつけたものが継続の規則名になっている。

一つだけ異なる規則 $(\overline{end})$ は、継続が初期継続のみの場合に初期継続に項を渡して実行を終了する規則である。これも $(begin)$ と同様特殊な規則であり、$(\overline{end})$ を用いた後は $(begin)$ で再び実行を始めることは許さない。

図 9 の実行規則を用いて、図 7 の SLC 式を実際に実行すると、図 11 のようになる。

$1 \uparrow \overline{\lceil y \Leftarrow y \rceil \uparrow (x \Rightarrow x)}$ は $+\mathsf{int}$ 型であるので、まず $(begin)$ で $\neg\mathsf{int}$ 型 (int 型を受け取る継続の型) の初期継続を与えて SLC の計算状態をつくり、$(\overline{pop})$ で項から関数を取り出す。この関数はこのままではβ簡約できないので、(exc) で $\overline{\ }$ の中身の実行に移る。中央に現れた関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ は、項の形と仮引数のパターンマッチが合っていないので $(push)$ で一旦継続に移動し、さらに項の内側の実行に移る。恒等関数 $x \Rightarrow x$ のβ簡約が終わったら、先程継続に移した関数 $\lceil g \rceil \Rightarrow 1 \uparrow g$ を (pop) で中央に呼び戻す。今度は項の形とパターンマッチが一致しているので、(β_f) でβ簡約を行うことができる。パターンマッチによって引数 $\lceil y \Leftarrow y \rceil$ のタグ $\lceil\ \rceil$ が外れ、以降の式で $y \Leftarrow y$ を関数として扱えるようになっている。$y \Leftarrow y$ は項の関数 $x \Rightarrow x$ と対を成すもので、継続の恒等関数である。従って $(\overline{\beta})$ でこの関数のβ簡約を行うと、そのときの継続 $\bullet_{\mathsf{int}}$ がそのまま返ってくる。

図 9 の実行規則について以下の定理が成り立つことを、c, f, e 及び簡約規則による場合分けでそれぞれ示すことができる。

Theorem 3.1 (Progress)
図 9 の実行規則について

1. $\vdash \langle\ e \mid f \mid c\ \rangle$ であるならば、ある c', f', e' が存在し、$\langle\ e \mid f \mid c\ \rangle \rightsquigarrow \langle\ e' \mid f' \mid c'\ \rangle$ または $\langle\ e \mid f \mid c\ \rangle \rightsquigarrow \langle\ e' \mid c'\ \rangle$ である。

2. $\vdash \langle\ e \mid c\ \rangle$ であるならば、それは $\langle\ e \mid \bullet_T\ \rangle$ $(e : +T)$ という形であるか、ある c', f', e' が存在して、$\langle\ e \mid c\ \rangle \rightsquigarrow \langle\ e' \mid f' \mid c'\ \rangle$ である。

Theorem 3.2 (Preservation)
図 9 の実行規則について

1. $\vdash \langle\ e \mid f \mid c\ \rangle$ のとき、$\langle\ e \mid f \mid c\ \rangle \rightsquigarrow \langle\ e' \mid f' \mid c'\ \rangle$ ならば $\vdash \langle\ e' \mid f' \mid c'\ \rangle$ であり、$\langle\ e \mid f \mid c\ \rangle \rightsquigarrow \langle\ e' \mid c'\ \rangle$ ならば $\vdash \langle\ e' \mid c'\ \rangle$ である。

2. $\vdash \langle\ e \mid c\ \rangle$ のとき、$\langle\ e \mid c\ \rangle \rightsquigarrow \langle\ e' \mid f' \mid c'\ \rangle$ ならば $\vdash \langle\ e' \mid f' \mid c'\ \rangle$ である。

なお、図 9 に示した実行規則は非決定的である。この規則は自由度がかなり高く、$(\overline{pop})$ と $(\overline{push})$ を無限に繰り返したり、$(begin)$ で実行を始めた直後に $(\overline{end})$ で実行を終了することができる。

3.4 Call-By-Value SLC

3.3 節で示した実行規則は非決定的であり、自由度が極めて高かった。ここでは SLC に評価戦略 Call-By-Value (CBV) を導入し、SLC の性質についてより詳しく検証していく。

一般に Call-By-Value とは、β 簡約を行う際にその引数をそれ以上簡約できない形、つまり値 (value) まで簡約してから、β 簡約を行うものである。SLC では、β 簡約が項と継続の 2 種類存在する。Call-By-Value SLC は、項のβ簡約を行う際には引数を値まで計算

$$
\begin{array}{rrcll}
(begin) & e : +T & \rightsquigarrow & \langle\, e \mid \bullet_T \,\rangle & \\
(left_v) & \langle\, (e_1, e_2) \mid c \,\rangle & \rightsquigarrow & \langle\, e_1 \mid x_1 \Rightarrow (x_1, e_2) \mid c \,\rangle & \text{if } e_1 \neq v \\
(right_v) & \langle\, (v_1, e_2) \mid c \,\rangle & \rightsquigarrow & \langle\, e_2 \mid x_2 \Rightarrow (v_1, x_2) \mid c \,\rangle & \text{if } e_2 \neq v \\
(\overline{pop}) & \langle\, e \uparrow f \mid c \,\rangle & \rightsquigarrow & \langle\, e \mid f \mid c \,\rangle & \\
(\overline{pop'_{pv}}) & \langle\, [v \uparrow (c' \Leftarrow (y_1, y_2))] \mid (c_1, c_2) \,\rangle & \rightsquigarrow & \langle\, v \mid c' \Leftarrow (y_1, y_2) \mid (c_1, c_2) \,\rangle & \\
(\overline{pop'_{fv}}) & \langle\, [v \uparrow (c' \Leftarrow \lfloor h \rfloor)] \mid \lfloor f \rfloor \,\rangle & \rightsquigarrow & \langle\, v \mid c' \Leftarrow \lfloor h \rfloor \mid \lfloor f \rfloor \,\rangle & \\
(push_v) & \langle\, e \mid f \mid c \,\rangle & \rightsquigarrow & \langle\, e \mid f \downarrow c \,\rangle & \text{if } e \neq v \\
(exc_v) & \langle\, v \mid \overline{e'} \mid c \,\rangle & \rightsquigarrow & \langle\, e' \mid \lceil g \rceil \Rightarrow v \uparrow g \mid c \,\rangle & \\
(\beta_v) & \langle\, v \mid x \Rightarrow e' \mid c \,\rangle & \rightsquigarrow & \langle\, e'[v/x] \mid c \,\rangle & \\
(\beta_{pv}) & \langle\, (v_1, v_2) \mid (x_1, x_2) \Rightarrow e' \mid c \,\rangle & \rightsquigarrow & \langle\, e'[v_1/x_1, v_2/x_2] \mid c \,\rangle & \\
(\beta_f) & \langle\, \lceil f \rceil \mid \lceil g \rceil \Rightarrow e' \mid c \,\rangle & \rightsquigarrow & \langle\, e'[f/g] \mid c \,\rangle & \\
(\overline{\beta_{fv}}) & \langle\, v \mid c' \Leftarrow \lfloor h \rfloor \mid \lfloor f \rfloor \,\rangle & \rightsquigarrow & \langle\, v \mid c'[f/h] \,\rangle & \\
(\overline{\beta_{pv}}) & \langle\, v \mid c' \Leftarrow (y_1, y_2) \mid (c_1, c_2) \,\rangle & \rightsquigarrow & \langle\, v \mid c'[c_1/y_1, c_2/y_2] \,\rangle & \\
(\overline{\beta_v}) & \langle\, v \mid c' \Leftarrow y \mid c \,\rangle & \rightsquigarrow & \langle\, v \mid c'[c/y] \,\rangle & \\
(\overline{exc_v}) & \langle\, v \mid \underline{c'} \mid c \,\rangle & \rightsquigarrow & \langle\, v \mid h \downarrow c \Leftarrow \lfloor h \rfloor \mid c' \,\rangle & \\
(\overline{push'_{fv}}) & \langle\, v \mid c' \Leftarrow \lfloor h \rfloor \mid c \,\rangle & \rightsquigarrow & \langle\, [v \uparrow (c' \Leftarrow \lfloor h \rfloor)] \mid c \,\rangle & \text{if } c \neq \lfloor f \rfloor \\
(\overline{push'_{pv}}) & \langle\, v \mid c' \Leftarrow (y_1, y_2) \mid c \,\rangle & \rightsquigarrow & \langle\, [v \uparrow (c' \Leftarrow (y_1, y_2))] \mid c \,\rangle & \text{if } c \neq (c_1, c_2) \\
(pop_v) & \langle\, v \mid f \downarrow c \,\rangle & \rightsquigarrow & \langle\, v \mid f \mid c \,\rangle & \\
(\overline{end_v}) & \langle\, v \mid \bullet_T \,\rangle & \rightsquigarrow & v : +T &
\end{array}
$$

図 12: Call-By-Value SLC 実行規則

$$
\begin{array}{rcl}
v & ::= & x \mid n \mid (v, v) \mid \lceil f \rceil \mid \\
 & & [v \uparrow (c \Leftarrow (y_1, y_2))] \mid [v \uparrow (c \Leftarrow \lfloor h \rfloor)] \\
e & ::= & v \mid (e, e) \mid e \uparrow f \\
f & ::= & g \mid x \Rightarrow e \mid (x_1, x_2) \Rightarrow e \mid \lceil g \rceil \Rightarrow e \mid \overline{e} \mid \\
 & & h \mid c \Leftarrow y \mid c \Leftarrow (y_1, y_2) \mid c \Leftarrow \lfloor h \rfloor \mid \underline{c} \\
c & ::= & \bullet_T \mid y \mid (c, c) \mid \lfloor f \rfloor \mid f \downarrow c \\
 & & \\
\mathsf{var} & ::= & x \mid y \mid g \mid h
\end{array}
$$

図 13: Call-By-Value SLC 構文

し、継続の β 簡約は（継続の）引数が値継続になっていなくても β 簡約を行うものとする。また、継続の計算よりも項の計算が優先される。つまり、継続の計算は項が値になってから行う。通常、λ 計算の Call-By-Value と言うと Right-to-Left と Left-to-Right の 2 種類存在するが、本論文の CBV は正確には Right-to-Left CBV にあたる。なお SLC では項の関数適用の位置が通常の λ 計算とは逆になっているため、実際には SLC 式の左側から順に実行することになる。

CBV SLC の構文を図 13 に示した。CBV では項の関数に与える引数をそれ以上簡約できない形にする。

$$
\frac{\Gamma \vdash c \Leftarrow (y_1, y_2) : \{^{+T \to +(A \vee B)}_{\neg T \leftarrow \neg(A \vee B)} \quad \Gamma \vdash v : +T}{\Gamma \vdash [v \uparrow (c \Leftarrow (y_1, y_2))] : +(A \vee B)}\,\texttt{TPCont}
$$

$$
\frac{\Gamma \vdash c \Leftarrow \lfloor h \rfloor : \{^{+T \to +(A - B)}_{\neg T \leftarrow \neg(A - B)} \quad \Gamma \vdash v : +T}{\Gamma \vdash [v \uparrow (c \Leftarrow \lfloor h \rfloor)] : +(A - B)}\,\texttt{TFCont}
$$

図 14: Call-By-Value SLC 追加型規則

従って、項 e とは別にそれ以上簡約できない式、値 v を定義する。値 v は、整数 n、変数 x、(e, e) の関数適用を含まない形 (v, v)、関数の項 (値) 扱い $\lceil f \rceil$、コンテキスト $[\,]$ から成る。関数適用を含む組 (e, e) は項として扱う。項の関数の β 簡約は引数が値 v になってから行われるので、項の変数 x は値 v に含める。本節で追加されているコンテキスト $[\,]$ は、継続の式の形と関数のパターンマッチが合うまで、その関数の実行を凍結するためのものである。コンテキストの型規則を図 14 に示した。関数が継続のパターンマッチ付き関数に、引数が値に制限されていることと、導出された SLC 式に $[\,]$ がついていること以外は、項の関数適用 `TApp` と全く同じである。コンテキストの具

体的な役割は、実行規則と併せて解説する。

CBV SLC の実行規則を図 12 に示した。図 9 の実行規則が元になっているが、項の β 簡約と継続の計算は項が v になってから行うよう、さらに実行手順が一意になるよう変更してある。図 9 から変更のある規則は、規則名に添字 $_v$ をつけて表している。

$(left_v)$ 及び $(right_v)$ は e_1 及び e_2 が値 v でないときに限り適用する。$(right_v)$ は組の左側が値である場合のみ適用できるものとし、左側が値でない場合は左側を先に計算する。

$(push_v)$ は項が値でないときに限り適用する。値ならば、β 簡約をするか継続の計算に移る。(exc_v) で $\overline{e'}$ の中身の計算に移るのも、項が値の場合である。

β 簡約はすべて項が値 v になってから行う。継続の β 簡約でも同じである。なお (β_f) のみはもともと項が値 $\lceil f \rceil$ のときのみ適用される規則なので、Call-By-Value でも規則に変化がない。

以降、継続に関する規則は全て、項が v である状態で実行を行う。Call-By-Value SLC では、一度項から取り出した関数を再び項に戻すことは許されないため、$(\overline{push})$ はコンテキストに関わる特別な条件下でのみ使用することになる (後述)。また継続の組の中身を計算する規則 $(\overline{left_v})$ 及び $(\overline{right})$ は Call-By-Value SLC では使用しない。これは、この二つの規則を加えると簡約の一意性 (定理 3.3) が失われるためである。$(\overline{left_v})$ と $(\overline{right})$ がないので、継続の組の中身はパターンマッチ付き関数で組から取り出されない限り計算されることはない。

新たに追加された規則 $(\overline{pop'_{fv}})$, $(\overline{pop'_{pv}})$, $(\overline{push'_{fv}})$, $(\overline{push'_{pv}})$ はコンテキスト [] に関わる規則である。$(\overline{\beta_{fv}})$ 及び $(\overline{\beta_{pv}})$ で継続の式の形がパターンと一致しなければ、継続 c の実行に移る。Call-By-Value では関数を項に戻すことは許されないため、現在の値 v と関数をまとめてコンテキスト [] として値扱いし、c の実行に移れるようにする規則が $(\overline{push'_{fv}})$ 及び $(\overline{push'_{pv}})$ である。c を実行してパターンマッチと継続の式の形が合うようになったら、$(\overline{pop'_{fv}})$ 及び $(\overline{pop'_{pv}})$ でコンテキスト中の継続の関数を取り出し、その実行に移る。

Call-By-Value SLC について、以下の定理が簡約規則による場合分けで得られる。

Theorem 3.3 (簡約の一意性)
Call-By-Value SLC において、

1. $\vdash \langle\, e \mid f \mid c \,\rangle$ ならば、その簡約の仕方は一意である。
2. $\vdash \langle\, e \mid c \,\rangle$ ならば、その簡約の仕方は一意である。

定理 3.3 から次の系が得られる。

Corollary 3.4 Call-By-Value SLC において、$\vdash e : +T$ ならばその簡約列は一意である。

また、Call-By-Value SLC について以下の定理が得られる。これは論理関係 (Logical Relation) の手法を用いて示すことができる。(紙面の都合上ここでは詳しく触れない。)

Theorem 3.5 (CBV SLC の停止性)
Call-By-Value SLC において、$\vdash e : +T$ であるならば $\langle\, e \mid \bullet_T \,\rangle \rightsquigarrow v$ と有限ステップで到達できる v が存在する。

3.5 慣用表現

一般に、論理式の型を与える計算理論には、論理和型の項をつくるプリミティブ inl, inr が存在する。SLC にはそのようなプリミティブが存在せず、明示的に論理和型の項を定義してはいないが、$\overline{\texttt{TPFun}}$ で導出される関数に `TApp` で項を関数適用させることで、論理和型の項を導出することができる。特に関数 $y_1 \Leftarrow (y_1, y_2)$ は、前提なしで導出できる上、項中心に型を考えると $+A$ 型の項を受け取って $+(A \vee B)$ 型の項を返す関数であることから、SLC では慣用的に inl と表記している。

このような慣用表現を図 15 にまとめた。

fst 及び snd は項の組の片方を取り出す関数である。$+(A \wedge B)$ 型を受け取って $+A$ 型または $+B$ 型を返す。継続を中心に考えると $\neg A$ 型または $\neg B$ 型を受け取って $\neg(A \wedge B)$ 型を返す関数になる。項とは対称に $\neg(A \wedge B)$ 型を持つ継続は明示的に定義されていないが、fst や snd を用いて論理積型の継続を導出することも可能である。

inl 及び inr は fst、snd と対を成す関数であり、本来は継続の組から片方を取り出す関数である。従って

$$
\begin{array}{rcl}
inl & = & y_1 \Leftarrow (y_1, y_2) : \{^{+A \to +(A \vee B)}_{\neg A \leftarrow \neg(A \vee B)} \\
inr & = & y_2 \Leftarrow (y_1, y_2) : \{^{+B \to +(A \vee B)}_{\neg B \leftarrow \neg(A \vee B)} \\
fst & = & (x_1, x_2) \Rightarrow x_1 : \{^{+(A \wedge B) \to +A}_{\neg(A \wedge B) \leftarrow \neg A} \\
snd & = & (x_1, x_2) \Rightarrow x_1 : \{^{+(A \wedge B) \to +B}_{\neg(A \wedge B) \leftarrow \neg B} \\
[f_1, f_2] & = & (f_1 \downarrow y, f_2 \downarrow y) \Leftarrow y : \{^{+(A \vee B) \to +T}_{\neg(A \vee B) \leftarrow \neg T} \\
\{f_1, f_2\} & = & x \Rightarrow (x \uparrow f_1, x \uparrow f_2) : \{^{+T \to +(A \wedge B)}_{\neg T \leftarrow \neg(A \wedge B)}
\end{array}
$$

図 15: SLC 慣用表現

$$\frac{\Gamma \vdash e : +(A \wedge B)}{\Gamma \vdash e \uparrow fst : +A}\ \texttt{Fst} \qquad \frac{\Gamma \vdash e : +A}{\Gamma \vdash e \uparrow inl : +(A \vee B)}\ \texttt{Inl}$$

$$\frac{\Gamma \vdash e : +(A \wedge B)}{\Gamma \vdash e \uparrow snd : +B}\ \texttt{Snd} \qquad \frac{\Gamma \vdash e : +B}{\Gamma \vdash e \uparrow inr : +(A \vee B)}\ \texttt{Inr}$$

$$\frac{\Gamma \vdash f_1 : \{^{+A \to +T}_{\neg A \leftarrow \neg T} \quad \Gamma \vdash f_2 : \{^{+B \to +T}_{\neg B \leftarrow \neg T}}{\Gamma \vdash [f_1,\, f_2] : \{^{+(A \vee B) \to +T}_{\neg (A \vee B) \leftarrow \neg T}}\ \texttt{FPair}$$

$$\frac{\Gamma \vdash f_1 : \{^{+T \to +A}_{\neg T \leftarrow \neg A} \quad \Gamma \vdash f_2 : \{^{+T \to +A}_{\neg T \leftarrow \neg A}}{\Gamma \vdash \{f_1,\, f_2\} : \{^{+T \to +(A \wedge B)}_{\neg T \leftarrow \neg (A \wedge B)}}\ \overline{\texttt{FPair}}$$

図 16: SLC 慣用表現 型規則

その型は $\neg(A \vee B)$ 型を受け取って $\neg A$ 型または $\neg B$ 型を返す形になる。この対偶を取って項を中心に考えると、$+A$ 型または $+B$ 型を受け取って $+(A \vee B)$ 型を返す関数になる。これらを項の関数適用で用いると $+(A \vee B)$ 型の項を導出することができる。

$[f_1, f_2]$ はいわゆる match 文に相当する関数である。項を中心に考えると f_1、f_2 は受け取る型は違うが返す型は同じであるので、一つの継続を共有することができる。それを表した関数が $(f_1 \downarrow y,\, f_2 \downarrow y) \Leftarrow y$ である。この関数は、現在の継続 ($\neg T$ 型) を切り取って二つの関数の後ろにコピーする。従って $+A$ 型の項を受け取っても $+B$ 型の項を受け取っても、この関数でつくられた継続の組のうち適当な方を選べば実行を完了することができる。$[f_1, f_2]$ の型は $\neg T$ 型の継続を受け取って $\neg(A \vee B)$ 型の継続を返す関数である。これを項中心の型で読み替えると $+(A \vee B)$ 型の項を受け取って $+T$ 型の項を返す関数になる。このことからも $[f_1, f_2]$ を match 文と見なすことができる。

$\{f_1, f_2\}$ は $[f_1, f_2]$ の継続版である。$[f_1, f_2]$ は項に対する match 文であったが、$\{f_1, f_2\}$ は継続に対する match 文と考えることができる。

慣用表現を用いた SLC 式の導出規則を図 16 にまとめた。inl らについては紙面の都合上項の規則のみの掲載だが、継続に対する規則も考えることができる。

4 $\mathrm{SLC}_\perp$ の導入

SLC の型定義は base case が整数型であり、論理学的な意味を与えることが難しい。また、否定記号 $\neg$ が継続の型にしか与えられない、つまり $\neg T$ 型は書けるが $+(\neg T \vee T)$ 型は SLC では書くことができない。

そこで、本論文では SLC の型定義に $\perp$ を加えた体系 $\mathrm{SLC}_\perp$ を導入し、$\mathrm{SLC}_\perp$ と古典論理との関係について考える。$\perp$ は「その型を持つ値は存在しない」という型を表す。従って $\perp$ 型の値は定義されない。

$\mathrm{SLC}_\perp$ の型定義を図 17 に示した。SLC (図 4) の定義に $\perp$ と命題変数 X が加わっている。古典論理との関連を調べることが目的であるので、整数型 int は除いてある。$\mathrm{SLC}_\perp$ の構文は図 6 から整数 n を除いたものである。型規則は図 8 から TInt を除いたものになる。実行規則は図 9 と変わらない。

$\mathrm{SLC}_\perp$ では SLC と同様にして次の定理が成り立つ。

Theorem 4.1
$\mathrm{SLC}_\perp$ において Progress が成り立つ。

Theorem 4.2
$\mathrm{SLC}_\perp$ において Preservation が成り立つ。

SLC と同様、$\mathrm{SLC}_\perp$ についても CBV を定義することができる。CBV $\mathrm{SLC}_\perp$ について、CBV SLC と同様にして次が成り立つ。

Theorem 4.3
CBV $\mathrm{SLC}_\perp$ の簡約の仕方は一意である。

Corollary 4.4
CBV $\mathrm{SLC}_\perp$ で $\vdash e : +T$ の簡約列は一意である。

Theorem 4.5
CBV $\mathrm{SLC}_\perp$ で $\vdash e : +T$ ならばその実行は停止する。

$\mathrm{SLC}_\perp$ と古典論理の関係を調べる際には、$\mathrm{SLC}_\perp$ の項型 $+T$ の T を古典論理による論理式と見なす。$\mathrm{SLC}_\perp$ の型定義では項型の中に $\neg$ は現れないが、関数を項扱いするときに使う型 $T \to \perp$ を $\neg T$ と解釈する。これは古典論理や直観主義論理において、$\neg T$ を $T \supset \perp$ と見なすのと全く同じことである。ただし、関数の型 $\{^{+T \to +\perp}_{\neg T \leftarrow \neg \perp}$ を $\neg T$ と解釈はしない。あくまでも、論理式の $\neg T$ と見なすのは「項」型の $T \to \perp$ である。このように厳格な区別をするのは、型規則中に表れるすべての $\{^{+T \to +\perp}_{\neg T \leftarrow \neg \perp}$ を $\neg T$ と解釈すると、論理

$$\begin{aligned} S ::= {} & +T && \text{(項 } e \text{ の型)} \\ \mid {} & \{^{+T \to +T}_{\neg T \leftarrow \neg T} && \text{(関数 } f \text{ の型)} \\ \mid {} & \neg T && \text{(継続 } c \text{ の型)} \\ T, A, B ::= {} & \perp \mid X \mid A \wedge B \mid A \vee B \mid A \to B \mid A - B \end{aligned}$$

図 17: $\mathrm{SLC}_\perp$ 型定義

学上は正当な推論でも $\mathrm{SLC}_\bot$ の型規則には当てはまらないことがあるからである。

$\mathrm{SLC}_\bot$ と古典論理の関係を調べるにあたって、$\mathrm{SLC}_\bot$ に対していくつか制限を加える。

まず、初期継続は $\bullet_\bot : \neg\bot$ のみ導出するものとする。$\mathrm{SLC}_\bot$ (SLC) では本来、$\overline{\texttt{TInit}}$ ではどのような型の初期継続を導出しても構わないが、論理学的には任意の論理式の否定を導出することはできないからである。さらに、導出する項 $e : +T$ は実行する前の項のみついて考える。これは $(begin)$ で実行を始めると継続に初期継続 $\bullet_T$ が現れ、項 e に代入される可能性を避けるためである。

これらの制限により、項 e が $+\bot$ 型を持つならば、$\Gamma, x : +\bot \vdash x : +\bot$ または $\Gamma \vdash e \uparrow f : +\bot$ $(\Gamma \neq \Phi)$ であることが言える。

5 $\mathrm{SLC}_\bot$ の論理学的解釈

この章では 3.2 節の $\mathrm{SLC}_\bot$ の型規則 (図 10 から TInt を除いたもの) が古典論理で表せることを見ていく。2 章の形式体系 NK 及び補題 2.1 を用いて、$\mathrm{SLC}_\bot$ の型規則が表現できることを示す。補題で用いる恒真式を図 18 に再掲した。証明図中では図 18 に示した規則名を用いている。$\mathrm{SLC}_\bot$ の型規則を NK で表したものを図 19 に示した。各証明図の配置は図 8 に則っている。

$\mathrm{SLC}_\bot$ の関数には 2 種類の型（含意とその対偶）が存在するが、図 19 では、項の規則（左側）では関数の型として「$A \supset B$」を用い、継続の規則（右側）では「$\neg B \supset \neg A$」を用いている。このため $\mathrm{SLC}_\bot$ の型導出木を NK で表す際には、図 19 の証明図の他に、CNT を用いて含意を対偶に変えたり、対偶を含意に戻したりという操作を加える必要がある。

TVar は NK の Axm で表される。$\overline{\texttt{TInit}}$ は NK で $\bot$ に関する Axm と $\supset$I で表される。TAnd は NK の $\wedge$I で表される。$\overline{\texttt{TOr}}$ は NK では $\wedge$I で $\neg A \wedge \neg B$ を導出した後、DM2 を適用して表すことができる。TFun、

$$\begin{aligned}
\neg(A \vee B) &\equiv \neg A \wedge \neg B && \texttt{DM2}\\
A \supset B &\equiv \neg B \supset \neg A && \texttt{CNT}\\
A \supset (B \supset C) &\equiv (A \wedge B) \supset C && \texttt{IMP}\\
A &\equiv \neg\neg A && \texttt{DNG}\\
A - B &\equiv \neg(\neg B \supset \neg A) && \texttt{MIN}
\end{aligned}$$

図 18: 恒真式

$$\frac{}{\Gamma, A \vdash A}\,\texttt{Axm} \qquad \frac{\overline{\Gamma, \bot \vdash \bot}\,\texttt{Axm}}{\Gamma \vdash \neg\bot}\,\supset\texttt{I}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,\wedge\texttt{I} \qquad \frac{\dfrac{\Gamma \vdash \neg A \quad \Gamma \vdash \neg B}{\Gamma \vdash \neg A \wedge \neg B}\,\wedge\texttt{I}}{\Gamma \vdash \neg(A \vee B)}\,\texttt{DM2}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\,\supset\texttt{I} \qquad \frac{\Gamma, \neg B \vdash \neg A}{\Gamma \vdash \neg B \supset \neg A}\,\supset\texttt{I}$$

$$\frac{\Gamma \vdash A \supset B \quad \Gamma \vdash A}{\Gamma \vdash B}\,\supset\texttt{E} \qquad \frac{\Gamma \vdash \neg B \supset \neg A \quad \Gamma \vdash \neg B}{\Gamma \vdash \neg A}\,\supset\texttt{E}$$

$$\frac{\Gamma \vdash A \supset B}{\Gamma \vdash A \supset B}\,(\text{なし}) \qquad \frac{\dfrac{\Gamma \vdash \neg B \supset \neg A}{\Gamma \vdash \neg\neg(\neg B \supset \neg A)}\,\texttt{DNG}}{\Gamma \vdash \neg(A - B)}\,\texttt{MIN}$$

$$\frac{\Gamma \vdash A \supset B}{\Gamma \vdash A \supset B}\,(\text{なし}) \qquad \frac{\dfrac{\Gamma \vdash \neg(A - B)}{\Gamma \vdash \neg\neg(\neg B \supset \neg A)}\,\texttt{MIN}}{\Gamma \vdash \neg B \supset \neg A}\,\texttt{DNG}$$

$$\frac{\dfrac{\Gamma, A, B \vdash T}{\Gamma, A \vdash B \supset T}\,\supset\texttt{I}}{\Gamma \vdash A \supset B \supset T}\,\supset\texttt{I}\;\texttt{IMP} \qquad \frac{\dfrac{\dfrac{\Gamma, \neg A, \neg B \vdash \neg T}{\Gamma, \neg A \vdash \neg B \supset \neg T}\,\supset\texttt{I}}{\Gamma \vdash \neg A \supset \neg B \supset \neg T}\,\supset\texttt{I}\;\texttt{IMP}}{\Gamma \vdash \neg A \wedge \neg B \supset \neg T}\,\texttt{DM2}$$

$$\frac{\Gamma, A \supset B \vdash T}{\Gamma \vdash (A \supset B) \supset T}\,\supset\texttt{I} \qquad \frac{\dfrac{\dfrac{\Gamma, \neg B \supset \neg A \vdash \neg T}{\Gamma \vdash (\neg B \supset \neg A) \supset \neg T}\,\supset\texttt{I}}{\Gamma \vdash \neg\neg(\neg B \supset \neg A) \supset \neg T}\,\texttt{DNG}}{\Gamma \vdash \neg(A - B) \supset \neg T}\,\texttt{MIN}$$

図 19: NK による $\mathrm{SLC}_\bot$ の型規則

$\overline{\texttt{TFun}}$ は NK の $\supset$I で表される。TApp、$\overline{\texttt{TApp}}$ は NK の $\supset$E で表される。TFClo、TCFun は $\mathrm{SLC}_\bot$ 上の式変形であり、論理学的な意味はない。$\overline{\texttt{TFClo}}$、$\overline{\texttt{TCFun}}$ は DNG と MIN で表すことができる。TPFun、$\overline{\texttt{TPFun}}$ は NK で $\supset$I を二回用いて $A \supset B \supset T$ を導出した後、移入律 で引数部分を $A \wedge B$ に変えたものと考えられる。$\overline{\texttt{TPFun}}$ ではさらに DM2 を用いる。TFFun は NK の $\supset$I の特別な場合と考えられる。引数は $\mathrm{SLC}_\bot$ の型定義に合わせて TFClo と同様の変換を行っている。$\overline{\texttt{TFFun}}$ も同様である。

以上より、次の定理が得られる。

Theorem 5.1 $\mathrm{SLC}_\bot$ の型規則は、古典論理の形式体系 NK で表すことができる。

6 SLC$_\perp$ における古典論理

5 章では、SLC$_\perp$ の型規則を古典論理の形式体系 NK で表せることを示した。本章では、古典論理の形式体系 NK を SLC$_\perp$ の型規則で表せることを示す。

6.1 SLC$_\perp$ 項での古典論理

古典論理の形式体系 NK (図 2) を SLC$_\perp$ の型規則で表したものを図 20 に示した。ここでは古典論理の型を持つ「項」を導出することを考えるので、型規則も項中心に記述している。

$\overline{\texttt{TInit}}$ は言わば $\top$ を導出するための規則である。論理学において $\top$ を導出することには特には意味を成さない。しかし SLC$_\perp$ の構文上、ダミー値 (ダミー変数) が必要となる場合があるので、その際にはこの規則を用いる。

TVar は NK の Axm を表す。TAnd は NK の $\wedge$I を、Fst、Snd は $\wedge$E$_1$、$\wedge$E$_2$ を表す。Inl、Inr は NK の $\vee$I$_1$、$\vee$I$_2$ を表す。

FPair が用いられている中段の証明図は、NK の $\vee$E を表したものである。$\Gamma, x_1 : +A \vdash e_1 : +C$ と $\Gamma, x_2 : +B \vdash e_2 : +C$ からそれぞれ関数を導出し、それらを関数のペアにして $e : +(A \vee B)$ を渡している。計算理論的には match 文に式を渡す様子を表しているものと言える。

TFun、TFFun は NK の $\supset$I を、TApp は $\supset$E を表す。$\supset$ I を表す規則が二種類あるのは、SLC$_\perp$ では項と関数を明確に区別しており、それらに対する構文や規則が異なるためである。

TFClo は主に $+(A \to \perp)$ 型の項、つまり古典論理での $\neg A$ 型を導出するために用いる。DNeg は RAA を表すための公理である。$\Gamma, f_x : \{^{+A\to+\perp}_{\neg A\leftarrow\neg\perp} \vdash e : +\perp$ から TFFun と TFClo で $\Gamma \vdash \lceil\lceil f_x\rceil \Rightarrow e\rceil : +((A \to \perp) \to \perp)$ を導出し、DNeg の関数と TApp で関数適用させることで $+A$ 型の項を導出できる。ここで用いられている命題変数 A は図 17 の T の定義を満たしていれば、どんな論理式であってもよい。なお DNeg の関数は inl などと同様、SLC$_\perp$ の型規則 (図 8 から TInt をのぞいたもの) で前提なしで (Γ が空集合でも) 導出することが可能である。この関数は式が長いので、以降 f_{DN} と略記する。

以上より、次の定理を示すことができる。

Theorem 6.1
SLC$_\perp$ の型規則で、古典論理の形式体系 NK を表すことができる。

Theorem 6.2
SLC$_\perp$ において $\vdash e : +T \Leftrightarrow T$ は古典論理での恒真式である。

例として、図 21 に $+(A \vee (A \to \perp))$ の導出木を示した。この図は、古典論理での $A \vee \neg A$ の証明図 (図 22) と図 20 に従ったものである。

6.2 考察：継続の役割

6.1 節では項の型に注目し、その型が古典論理の推論規則を表せることで古典論理との対応を示した。しかし 6.1 節ではほとんど項に関する規則しか使っていない。残りの継続に関する規則には、どのような意味があるのだろうか。

6.1 節では特に触れなかったが、図 20 の DNeg の導出には継続の規則が使われている。図 25 にその証明図を示した。式中でも継続の関数 $\cdots \Leftarrow y$ や継続の関数適用 $\downarrow$ が用いられているが、導出木中でも当然、それらに関する規則が使われている。また、これも特に言明していないが inl 及び inr の本体は継続のパターンマッチ付き関数であるので、厳密にはこの関数を導出するために $\overline{\texttt{TPFun}}$ が必要である。

古典論理式を SLC$_\perp$ で証明するにあたり、必ずしも図 20 の規則に則る必要はない。例えば、図 20 では RAA を関数 f_{DN} で表しているが、継続を用いると図 23 のように表せる。導出木左上の $\Gamma, y : \neg A \vdash e : +\perp$ から、最下段の $+A$ 型の項を導く形である。ここでは、変数 y の型 $\neg A$ は古典論理の $\neg A$ として扱う。こちらの方が直感的には NK の導出規則に近い。

図中の id は恒等関数 $x \Rightarrow x : \{^{+A\to+A}_{\neg A\leftarrow\neg A}$ ($y \Leftarrow y$ でもよい) を値扱いした $\lceil x \Rightarrow x\rceil : +(A \to A)$ のことであり、論理学的には $\top$ と等価である。型 ID は関数の値扱い $A \to A$ を表す。紙面の都合上、このような略記をしている。これらは、項 e を変数 y の有効範囲に入れるためのダミー値である。また、図 23 の RAA を用いるには継続の変数 y を用いて $+\perp$ 型の項 e を導くことになるので、図 20 に従うだけでは導出はできず、継続に関する規則を用いることになる。

本論文ではあまり扱っていないが、直観主義論理の導出規則 $\perp$E を SLC$_\perp$ で表すこともできる。図 24 にそれを示した。右下にある $\Gamma \vdash e : +\perp$ から、最下段の $+A$ 型の項を導出している。これは、図 23 の変数 y が項 e に現れない場合と考えることができる。導出された項では、ダミー値やダミー変数がなく、e が y の有効範囲から外れている。つまり、図 24 では $e : +\perp$ の導出に仮定 $y : \neg A$ を使っていない。この

$$\frac{}{\Gamma,\, x : S \vdash x : S}\ \texttt{TVar} \qquad \frac{}{\Gamma \vdash \bullet_\bot : \neg\bot}\ \texttt{TInit}$$

$$\frac{\Gamma \vdash e_1 : +A \quad \Gamma \vdash e_2 : +B}{\Gamma \vdash (e_1,\, e_2) : +(A \wedge B)}\ \texttt{TAnd} \qquad \frac{\Gamma \vdash e : +(A \wedge B)}{\Gamma \vdash e \uparrow fst : +A}\ \texttt{Fst} \qquad \frac{\Gamma \vdash e : +(A \wedge B)}{\Gamma \vdash e \uparrow snd : +B}\ \texttt{Snd}$$

$$\frac{\Gamma \vdash e : +A}{\Gamma \vdash e \uparrow inl : +(A \vee B)}\ \texttt{Inl} \qquad \frac{\Gamma \vdash e : +B}{\Gamma \vdash e \uparrow inr : +(A \vee B)}\ \texttt{Inr}$$

$$\dfrac{\dfrac{\dfrac{\Gamma,\, x_1 : +A \vdash e_1 : +C}{\Gamma \vdash x_1 \Rightarrow e_1 : \{^{+A \to +C}_{\neg A \leftarrow \neg C}}\ \texttt{TFun} \quad \dfrac{\Gamma,\, x_2 : +B \vdash e_2 : +C}{\Gamma \vdash x_2 \Rightarrow e_2 : \{^{+B \to +C}_{\neg B \leftarrow \neg C}}\ \texttt{TFun}}{\Gamma \vdash [x_1 \Rightarrow e_1,\, x_2 \Rightarrow e_2] : \{^{+(A \vee B) \to +C}_{\neg(A \vee B) \leftarrow \neg C}}\ \texttt{FPair} \qquad \Gamma \vdash e : +(A \vee B)}{\Gamma \vdash e \uparrow [x_1 \Rightarrow e_1,\, x_2 \Rightarrow e_2] : +C}\ \texttt{TApp}$$

$$\frac{\Gamma,\, x : +A \vdash e : +B}{\Gamma \vdash x \Rightarrow e : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}\ \texttt{TFun} \qquad \frac{\Gamma,\, g : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \vdash e : +C}{\Gamma \vdash \lceil g \rceil \Rightarrow e : \{^{+(A \to B) \to +C}_{\neg(A \to B) \leftarrow \neg C}}\ \texttt{TFFun} \qquad \frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B} \quad \Gamma \vdash e : +A}{\Gamma \vdash e \uparrow f : +B}\ \texttt{TApp}$$

$$\frac{\Gamma \vdash f : \{^{+A \to +B}_{\neg A \leftarrow \neg B}}{\Gamma \vdash \lceil f \rceil : +(A \to B)}\ \texttt{TFClo} \qquad \frac{}{\Gamma \vdash (\lceil g \rceil \Rightarrow \lceil y \Leftarrow y' \rceil \uparrow g) \downarrow \bullet_\bot \Leftarrow y : \{^{+((A \to \bot) \to \bot) \to +A}_{\neg((A \to \bot) \to \bot) \leftarrow \neg A}}\ \texttt{DNeg}$$

図 20: $\mathrm{SLC}_\bot$ における古典論理 NK の導出規則

※紙面の都合上 $\Gamma = g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot},\, x : +A$ と略記している箇所がある

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot}}\ \texttt{TVar} \quad \dfrac{\dfrac{}{\Gamma \vdash x : +A}\ \texttt{TVar}}{\Gamma \vdash x \uparrow inl : +(A \vee (A \to \bot))}\ \texttt{Inl}}{g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot},\, x : +A \vdash x \uparrow inl \uparrow g : +\bot}\ \texttt{TApp}}{g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot} \vdash x \Rightarrow x \uparrow inl \uparrow g : \{^{+A \to +\bot}_{\neg A \leftarrow \neg\bot}}\ \texttt{TFun}}{g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot} \vdash \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil : +((A \vee (A \to \bot)) \to \bot)}\ \texttt{TFClo}}{g : \{^{+(A \vee (A \to \bot)) \to +\bot}_{\neg(A \vee (A \to \bot)) \leftarrow \neg\bot} \vdash \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil \uparrow inr : +(A \vee (A \to \bot))}\ \texttt{Inr} \qquad \cdots *$$

$$\frac{}{\vdash f_{DN} : \{^{+(((A \vee (A \to \bot)) \to \bot) \to \bot) \to +(A \vee (A \to \bot))}_{\neg(((A \vee (A \to \bot)) \to \bot) \to \bot) \leftarrow \neg(A \vee (A \to \bot))}}\ \texttt{DNeg} \qquad \cdots **$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{g : \{^{+(A \vee \neg A) \to +\bot}_{\neg(A \vee \neg A) \leftarrow \neg\bot} \vdash g : \{^{+(A \vee \neg A) \to +\bot}_{\neg(A \vee \neg A) \leftarrow \neg\bot}}\ \texttt{TVar} \qquad *}{g : \{^{+(A \vee \neg A) \to +\bot}_{\neg(A \vee \neg A) \leftarrow \neg\bot} \vdash \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil \uparrow inr \uparrow g : +\bot}\ \texttt{TApp}}{\vdash \lceil g \rceil \Rightarrow \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil \uparrow inr \uparrow g : \{^{+((A \vee (A \to \bot)) \to \bot) \to +\bot}_{\neg((A \vee (A \to \bot)) \to \bot) \leftarrow \neg\bot}}\ \texttt{TFFun}}{\vdash \lceil \lceil g \rceil \Rightarrow \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil \uparrow inr \uparrow g \rceil : +(((A \vee (A \to \bot)) \to \bot) \to \bot)}\ \texttt{TFClo} \qquad **}{\vdash \lceil \lceil g \rceil \Rightarrow \lceil x \Rightarrow x \uparrow inl \uparrow g \rceil \uparrow inr \uparrow g \rceil \uparrow f_{DN} : +(A \vee \neg A)}\ \texttt{TApp}$$

図 21: $\mathrm{SLC}_\bot$ における $A \vee \neg A$ の導出

$$\dfrac{\dfrac{}{\neg(A\vee\neg A)\vdash\neg(A\vee\neg A)}\,\mathtt{Axm}}{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A\vee\neg A),\,A\vdash\neg(A\vee\neg A)}\,\mathtt{Axm}\qquad\dfrac{\dfrac{}{\neg(A\vee\neg A),\,A\vdash A}\,\mathtt{Axm}}{\neg(A\vee\neg A),\,A\vdash A\vee\neg A}\,\vee\mathtt{I}_1}{\neg(A\vee\neg A),\,A\vdash\bot}\,\supset\mathtt{E}}{\neg(A\vee\neg A)\vdash\neg A}\,\supset\mathtt{I}}{\neg(A\vee\neg A)\vdash A\vee\neg A}\,\vee\mathtt{I}_2}{\neg(A\vee\neg A)\vdash\bot}\,\supset\mathtt{E}}{\vdash A\vee\neg A}\,\mathtt{RAA}$$

図 22: NK における $A \vee \neg A$ の証明図

２つの導出木は、古典論理と直観主義論理の違いを $\mathrm{SLC}_\perp$ で端的に示しているものと思われる。

背理法 RAA は、本来成り立たないはずの仮定のもとで推論を行い、矛盾が生じるならその仮定を誤りと結論づけるものである。計算理論で言うと、導出できない（型エラーとなる）はずの式や型を用いて推論を進めなければならない。しかしそのような式や型は事実存在しないのであるから、背理法を用いるための推論自体ができないことになる。

$\mathrm{SLC}_\perp$ においても項のみで古典論理を表すことは不可能である。しかし項では表せない背理法の仮定を継続から借りることにより、背理法による推論を可能にしているように思える。図 23 がそれにあたる。

図 23 の左上の $\Gamma, y : \neg A \vdash e : +\perp$ は、項では表せない背理法の仮定を継続の変数 $y : \neg A$ で表し、それを用いてたどり着いた結論と考えられる。継続の型 $\neg A$ の意味が「$+A$ 型の項を受け取る継続」であることも軽視できない。つまり、$\neg A$ 型の継続の存在はすなわち、$+A$ 型の項の存在を示唆しているように思える。実際、この項に $\neg A$ 型の継続を与えると、それが e 中の変数 y に代入され、いずれは $\neg A$ 型の継続に $+A$ 型の項が渡るであろうことが予想される。

一方図 24 は、図 23 と似た項を導出するが、$e : +\perp$ と変数 $y : \neg A$ は全く関係がなく、導出された項は与えられた $\neg A$ 型の継続を完全に捨ててしまう。この挙動は、$+A$ 型の項の存在に対して非常に否定的であるように見える。

$\mathrm{SLC}_\perp$ と古典論理の関係における継続の意義は、$\neg A$ 型でありながら $+A$ 型に対して肯定的な仮定を与えることにあると思われる。古典論理と関係する計算理論は、$\mathrm{SLC}_\perp$ の他にも多く存在するが、これらにも $\neg A$ でありながら A を許容するような構造があるのではないかと推測する。

参考文献

[1] Curien, P. and Herbelin, H. “The duality of computation,” *Proceedings of the fifth ACM SIGPLAN International Conference on Functional Programming (ICFP’00)*, pp. 233–243 (2000).

[2] Felleisen, M., and Hieb, R. “The Revised Report on the Syntactic Theories of Sequential Control and State,” *Theoretical Computer Science*, Vol. 103, No. 2, pp. 235–271 (September 1992).

[3] Filinski, A. “Declarative Continuations and Categorical Duality,” Master’s thesis, DIKU Report 89/11, University of Copenhagen (August 1989).

[4] Griffin, T. “A Formulae-as-Types Notion of Control,” *17’th Symposium on Principles of Programming Languages (POPL’90)*, pp. 47–58 (January 1990).

[5] 小野寛晰「情報科学における論理」日本評論社 (April 1994).

[6] Parigot, M. “$\lambda\mu$-calculus: An Algorithmic Interpretation of Classical Natural Deduction,” In A. Voronkov, editor, *Logic Programming and Automated Reasoning (LNCS 624)*, pp. 190–201 (July 1992).

[7] Pierce, B. C. *Types and Programming Languages*, Cambridge: MIT Press (2002).

[8] 阪上紗里, 浅井健一「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 (May 2009).

[9] 上田やよい, 浅井健一「型付き対称λ計算における論理積と論理和の導入」第 11 回 プログラミング及びプログラミング言語ワークショップ, ポスター発表 (March 2009).

[10] Wadler, P. “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 (August 2003).

[11] Wadler, P. “Call-by-value is dual to call-by-name, Reloaded,” In J. Giesl, editor, *Term Rewriting and Applications (LNCS 3467)*, pp. 185–203 (April 2005).

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma,\, y:\neg A \vdash e:+\bot}{\Gamma,\, y:\neg A,\, x:+\mathsf{ID} \vdash e:+\bot}\,\mathtt{Weak}}{\Gamma,\, y:\neg A \vdash x \Rightarrow e:\{^{+\mathsf{ID}\to+\bot}_{\neg\mathsf{ID}\leftarrow\neg\bot}}\,\mathtt{TFun} \quad \dfrac{}{\Gamma,\, y:\neg\bot \vdash \bullet_\bot:\neg\bot}\,\overline{\mathtt{TInit}}}{\Gamma,\, y:\neg A \vdash (x \Rightarrow e)\downarrow \bullet_\bot:\neg\mathsf{ID}}\,\overline{\mathtt{TApp}} \quad \dfrac{}{\Gamma \vdash id:+\mathsf{ID}}\,\mathtt{IDFun}}{\Gamma \vdash id \uparrow ((x \Rightarrow e)\downarrow \bullet_\bot \Leftarrow y):+A}\,\mathtt{TApp}$$

図 23: RAA の $\mathrm{SLC}_\bot$ での導出木

$$\dfrac{\dfrac{}{\Gamma,\, y:\neg A \vdash \bullet_\bot:}\,\overline{\mathtt{TInit}}}{\Gamma \vdash \bullet_\bot \Leftarrow y:\{^{+\bot\to+A}_{\neg\bot\leftarrow\neg A}}\,\overline{\mathtt{TFun}} \quad \Gamma \vdash e:+\bot}{\Gamma \vdash e \uparrow (\bullet_\bot \Leftarrow y):+A}\,\mathtt{TApp}$$

図 24: ⊥E の $\mathrm{SLC}_\bot$ での導出木

$$\dfrac{\dfrac{}{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot} \vdash g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot}}\,\mathtt{TVar} \quad \dfrac{\dfrac{}{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot},\, y':\neg\bot \vdash y:\neg A}\,\mathtt{TVar}}{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot} \vdash y \Leftarrow y':\{^{+A\to+\bot}_{\neg A\leftarrow\neg\bot}}\,\overline{\mathtt{TFun}}}{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot} \vdash \lceil y \Leftarrow y'\rceil:+(A\to\bot)}\,\mathtt{TFClo}}{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot} \vdash \lceil y \Leftarrow y'\rceil \uparrow g:+\bot}\,\mathtt{TApp}$$

$$\dfrac{\dfrac{\dfrac{y:\neg A,\, g:\{^{+(A\to\bot)\to+\bot}_{\neg(A\to\bot)\leftarrow\neg\bot} \vdash \lceil y \Leftarrow y'\rceil \uparrow g:+\bot}{y:\neg A \vdash \lceil g\rceil \Rightarrow \lceil y \Leftarrow y'\rceil \uparrow g:\{^{+((A\to\bot)\to\bot)\to+\bot}_{\neg((A\to\bot)\to\bot)\leftarrow\neg\bot}}\,\mathtt{TFFun} \quad \dfrac{}{y:\neg A \vdash \bullet_\bot:\neg\bot}\,\overline{\mathtt{TInit}}}{y:\neg A \vdash (\lceil g\rceil \Rightarrow \lceil y \Leftarrow y'\rceil \uparrow g)\downarrow \bullet_\bot:\neg((A\to\bot)\to\bot)}\,\overline{\mathtt{TApp}}}{\vdash (\lceil g\rceil \Rightarrow \lceil y \Leftarrow y'\rceil \uparrow g)\downarrow \bullet_\bot \Leftarrow y:\{^{+((A\to\bot)\to\bot)\to+A}_{\neg((A\to\bot)\to\bot)\leftarrow\neg A}}\,\overline{\mathtt{TFun}}$$

図 25: $\mathrm{SLC}_\bot$ における $\neg\neg A \supset A$ の導出

※紙面の都合上 $\Gamma = y_1:\neg A,\, y_2:\neg(A\to\bot)$ と略記している箇所がある

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,\, x:+\mathsf{ID},\, y:\neg\bot \vdash y_1:\neg A}\,\mathtt{TVar}}{\Gamma,\, x:+\mathsf{ID} \vdash y_1 \Leftarrow y:\{^{+A\to+\bot}_{\neg A\leftarrow\neg\bot}}\,\overline{\mathtt{TFun}}}{\Gamma,\, x:+\mathsf{ID} \vdash \lceil y_1 \Leftarrow y\rceil:+(A\to\bot)}\,\mathtt{TFClo}}{\Gamma \vdash x \Rightarrow \lceil y_1 \Leftarrow y\rceil:\{^{+\mathsf{ID}\to+(A\to\bot)}_{\neg\mathsf{ID}\leftarrow\neg(A\to\bot)}}\,\mathtt{TFun} \quad \dfrac{}{\Gamma \vdash y_2:\neg(A\to\bot)}\,\mathtt{TVar}}{\Gamma \vdash (x \Rightarrow \lceil y_1 \Leftarrow y\rceil)\downarrow y_2:\neg\mathsf{ID}}\,\overline{\mathtt{TApp}}$$

$$\dfrac{\dfrac{y_1:\neg A,\, y_2:\neg(A\to\bot) \vdash (x \Rightarrow \lceil y_1 \Leftarrow y\rceil)\downarrow y_2:\neg\mathsf{ID}}{\vdash (x \Rightarrow \lceil y_1 \Leftarrow y\rceil)\downarrow y_2 \Leftarrow (y_1,\, y_2):\{^{+\mathsf{ID}\to+(A\vee(A\to\bot))}_{\neg\mathsf{ID}\leftarrow\neg(A\vee(A\to\bot))}}\,\overline{\mathtt{TPFun}} \quad \dfrac{}{\vdash id:+\mathsf{ID}}\,\mathtt{IDFun}}{\vdash id \uparrow ((x \Rightarrow \lceil y_1 \Leftarrow y\rceil)\downarrow y_2 \Leftarrow (y_1,\, y_2)):+(A\vee(A\to\bot))}\,\mathtt{TApp}$$

図 26: $\mathrm{SLC}_\bot$ における $A \vee \neg A$ の導出 (2)