# 論理関係による shift/reset の部分評価器の正当性の証明

横関 茉衣, 浅井 健一

お茶の水女子大学

{yokozeki.mai, asai}@is.ocha.ac.jp

**概 要**　部分評価器は static な計算と dynamic な計算に分けられたプログラムを受け取り、static な計算のみを実行した結果のプログラムを返す。部分評価器の正当性を証明するためには、プログラムの意味が変換前と変換後で変わらないということを証明する。この証明を添えて shift/reset の部分評価器の定義を提案した研究がある。しかし、その正当性の証明は継続の引数の項を値でないのに値であるとして扱っている部分があった。この証明の誤りは、部分評価器が正しくないこと、つまりは変換前に存在していた dynamic な effect が変換後には消えてしまうことに繋がっていた。そこで本研究は、let-insertion を用いて dynamic な effect が消えることを防ぐように部分評価器を変更し、その新しい部分評価器に対して正当性の証明を行った。

## 1　はじめに

部分評価とは、プログラムと一部の入力データを受け取ったら、その入力データのみに依存する部分を先に計算してしまった残余プログラム (residual program) を出力するようなプログラム変換である。残りの入力データで残余プログラムを実行したとき得られる結果は、全ての入力データで元のプログラムを実行した時に得られる結果と等しい [7]。部分評価は最適化手法のひとつであり、近年においてもドメイン固有言語の開発において部分評価を取り入れてより高速なプログラムを生成する取り組みが行われている [10]。

部分評価の理論は活発に研究されていた歴史があり、研究の成果の一つとして Asai による shift/reset の部分評価器 [2, 1] がある。Asai は shift/reset [5] で拡張した値呼びの $\lambda$ 計算に対するオフライン部分評価器を定義した。オフライン部分評価器とは束縛時解析が済んだプログラムを特化するものである。つまり、オフライン部分評価器は static な計算と dynamic な計算に分けられたプログラムを受け取り、static な計算のみを実行した結果のプログラムを返す。このようにオフライン部分評価器はコンパイラと同じようにプログラム変換器であるが、プログラム変換器はプログラムの意味が変換前と変換後で変わらないことに責任を持つべきである。そこで Asai は部分評価器の定義とともに、論理関係という技法を使い、定義した部分評価器の正当性を証明した。構造的帰納法では高階関数が適用されるときの挙動を特徴づけることが難しいが、論理関係を使うと型の助けを借りて高階関数の性質を捉えることができる。しかし、部分評価器の正当性の証明を読み直したところ、証明には誤りがあることがわかり、そして部分評価器そのものも正しくないことがわかった。

本研究では、shift/reset の部分評価器の正当性の証明 [2, 1] に見つかった誤りを説明し、部分評価器が正しくないことを説明した上で、新たに部分評価器を定義する。さらに新たに定義した部分評価器に対して正当性の証明を行う。正当性の誤りは継続の引数の項を値でないのに値であるとして扱っている部分にある。この証明の誤りは、部分評価器が正しくないこと、つまりは変換前に存在していた dynamic な effect が変換後には消えてしまうことに繋がっていた。そこで本

研究は、let-insertion を用いて dynamic な effect が消えることを防ぐように部分評価器を変更し、その新しい部分評価器に対して正当性の証明を行った。

関連研究として、shift/reset の多段階計算 [9]、shift0/reset0 の多段階計算 [11] がある。多段階計算は部分評価と同じように特化によりプログラムを最適化するが、多段階計算はコードを変数に束縛したり、コードをコードに埋め込むことなどができ、自由度が高い。また、生成されるコードが型安全であることが保証されているという利点もある。一方で部分評価は、多段階計算と比較して自由度は低いが、元のプログラムと生成するプログラムで意味が変わらないため、そのままコンパイラの最適化に使用できるなどの利点がある。また、論理関係を使った証明についての関連研究として、Biernacka と Biernacki [3] による abortive な継続演算子で拡張した型付きラムダ計算に対する停止性の証明を値呼びと名前呼びの両方の体系に対して行った研究がある。本稿と同様に、項の論理関係と同時に継続の論理関係を定義している。

OCaml 5 に代数的エフェクトが導入されるなど限定継続を明示的に扱ったプログラムが増えてきている。そのようなプログラムの効率的な実行には、限定継続命令の最適化が欠かせない。しかし、限定継続命令の最適化は複雑で注意を要する。特に細かい議論をきちんとしないと間違った最適化を行うことになる。実際、最適化のひとつである先行研究の shift/reset の部分評価 [2, 1] は間違っていることがわかった。本稿ではその原因を十分に究明して、その間違いを正す。そして、正しい最適化に貢献することを目指す。

### 1.1 本稿の構成

2 節では本稿全体で使用するメタ言語の説明などの準備を行う。3 節では 4 節の前段階として $\lambda$ 計算の部分評価器を概説する。4 節では先行研究 [2, 1] の shift/reset の部分評価器を概説し、さらにその型システムと正当性を証明する定理（本稿で誤りだったことが示されたもの）を述べる。5 節では 4 節の部分評価器の正当性の証明に見つかった誤りと、部分評価器そのものに見つかった誤りを述べる。6 節では 5 節の部分評価器の誤りが修正された部分評価器を定義する。7 節では 6 節の新しい部分評価器の正当性の証明を行う。8 節ではまとめと今後の課題を述べる。

## 2 準備

### 2.1 メタ言語の定義

部分評価器は、直接形式でメタ言語の shift/reset を使って定義する。そのため、メタ言語は left-to-right の値呼びの $\lambda$ 計算を shift/reset とデータ型の構成子で拡張したものとする。メタ言語の構文は次のように表される。通常、メタ言語を表す際は $M$ を用いるが、継続の body 部分を表す際は $K$ を用いる。

$$
\begin{aligned}
M, K \quad ::= \quad & x \mid \lambda x.\, M \mid M\, M \mid \xi k.\, M \mid \langle M \rangle \mid n \mid M + 1 \mid \\
& \mathrm{Var}(n) \mid \mathrm{Lam}(n, M) \mid \mathrm{App}(M, M) \mid \mathrm{Shift}(n, M) \mid \mathrm{Reset}(M) \mid \\
& \overline{\mathrm{Lam}}(n, M) \mid \overline{\mathrm{App}}(M, M) \mid \overline{\mathrm{Shift}}(n, M) \mid \overline{\mathrm{Reset}}(M) \mid \\
& \underline{\mathrm{Lam}}(n, M) \mid \underline{\mathrm{App}}(M, M) \mid \underline{\mathrm{Shift}}(n, M)
\end{aligned}
$$

$\xi k.\, M$ は shift、$\langle M \rangle$ は reset を表す。データ型の構成子は、部分評価器への入出力項と、束縛時解析前の元のプログラムの項を表現する。上線が付いた構成子は入力項の static な項を、下線が付いた構成子は入力項の dynamic な項を表す。線が付いていない構成子は部分評価器の出力項と束縛時解析前の元のプログラムの項を表す。また、部分評価器の入出力となるプログラムの言語において、変数は自然数 $n$ で表現する。そのため、このメタ言語には自然数と 1 を加算する演算子がある。

部分評価器が出力のプログラムを作成するとき、新しい変数を生成する。出力項において、de Bruijn レベル [6] を使う。de Bruijn レベルは de Bruijn インデックスとは反対にラムダ束縛子に対して外から内へ番号を振る。そのため、例えば $\lambda x.\,(\lambda y.\,xy)\,x$ は de Bruijn レベルでは $\lambda.\,(\lambda.\,01)\,0$ となる。部分評価器の出力のプログラムを名前のついた変数を使って表すために、以下を定義する。

$$\begin{aligned}
\mathrm{var}(m) &= \lambda n.\,\mathrm{Var}(m) \\
\mathrm{lam}(f) &= \lambda n.\,\mathrm{Lam}(n, f\,(n+1)) \\
\mathrm{app}(f_1, f_2) &= \lambda n.\,\mathrm{App}(f_1\,n, f_2\,n) \\
\mathrm{shift}(f) &= \lambda n.\,\mathrm{Shift}(n, f\,(n+1)) \\
\mathrm{reset}(f) &= \lambda n.\,\mathrm{Reset}(f\,n) \\
\mathrm{let}(f_1, \mathrm{lam}(f_2)) &= \mathrm{app}(\mathrm{lam}(f_2), f_1)
\end{aligned}$$

$\mathrm{let}(f_1, \mathrm{lam}(f_2))$ は $\mathrm{app}(\mathrm{lam}(f_2), f_1)$ の略記となっており、部分評価器の定義で利用される。この定義を使って表された項 $M$ を具体的な項に変換するための操作として $\downarrow_n M$ を使う。$\downarrow_n M = M\,n$ である。例えば、$\downarrow_3 M$ は $M$ の最も外側の束縛子に 3 を振り、その一つ内側の束縛子に 4 を振るため、

$$\begin{aligned}
& \downarrow_3 (\mathrm{lam}(\lambda x.\,\mathrm{lam}(\lambda y.\,\mathrm{app}(\mathrm{var}(x), \mathrm{var}(y))))) \\
= \;& \mathrm{Lam}(3, \mathrm{Lam}(4, \mathrm{App}(\mathrm{Var}(3), \mathrm{Var}(4))))
\end{aligned}$$

となる。

メタ言語の値は次のように表される。

$$\begin{aligned}
V \;::=\; & x \mid \lambda x.\,M \mid n \mid \\
& \mathrm{Var}(n) \mid \mathrm{Lam}(n, V) \mid \mathrm{App}(V, V) \mid \mathrm{Shift}(n, V) \mid \mathrm{Reset}(V) \mid \\
& \overline{\mathrm{Lam}}(n, V) \mid \overline{\mathrm{App}}(V, V) \mid \overline{\mathrm{Shift}}(n, V) \mid \overline{\mathrm{Reset}}(V) \mid \\
& \underline{\mathrm{Lam}}(n, V) \mid \underline{\mathrm{App}}(V, V) \mid \underline{\mathrm{Shift}}(n, V)
\end{aligned}$$

## 2.2 メタ言語の項間の等価性を表す記号

メタ言語の項間の等価性を表す記号については、定義または $\alpha$ 同値を表す際は $=$、以下の Kameyama と Hasegawa [8] の shift/reset の公理による項間の等価性を表す際は $\sim$ を用いる。

### 2.2.1 shift/reset の公理

Pure な評価文脈を下記の通り定義する。

$$\begin{aligned}
F \;=\; & [\,] \mid F\,M \mid V\,F \mid \\
& \mathrm{Lam}(x, F) \mid \mathrm{App}(F, M) \mid \mathrm{App}(V, F) \mid \mathrm{Shift}(k, F) \mid \mathrm{Reset}(F) \mid \\
& \overline{\mathrm{Lam}}(x, F) \mid \overline{\mathrm{App}}(F, M) \mid \overline{\mathrm{App}}(V, F) \mid \overline{\mathrm{Shift}}(k, F) \mid \overline{\mathrm{Reset}}(F) \mid \\
& \underline{\mathrm{Lam}}(x, F) \mid \underline{\mathrm{App}}(F, M) \mid \underline{\mathrm{App}}(V, F) \mid \underline{\mathrm{Shift}}(k, F)
\end{aligned}$$

これは、hole $[\,]$ がメタ言語の reset では囲まれていないような評価文脈である。Kameyama と Hasegawa [8] による、shift/reset で拡張された $\lambda$ 計算における公理のうち本稿で使用するものは下記の通りである。

$$\begin{array}{rcll}
(\lambda x.\,M)\,V & \sim & M[V/x] & \beta_v \\
(\lambda x.\,F[x])\,M & \sim & F[M] \quad \text{if } x \notin FV(F) & \beta_\Omega \\
\langle V \rangle & \sim & V & \textit{reset-value} \\
\langle F[\xi k.\,M] \rangle & \sim & \langle (\lambda k.\,M)\,(\lambda x.\,\langle F[x] \rangle) \rangle \quad \text{if } x \notin FV(F) & \textit{reset-shift}
\end{array}$$

### 継続とは

継続とは、プログラムを実行している最中における「残りの計算」である。例えば、$1+2*3$ という式の実行において、$2*3$ を計算しようとしている際の継続は「$2*3$ の結果を受け取り、それに 1 を加える」となる。

### shift/reset とは

shift/reset[5] とは限定継続、つまり限定された継続をプログラム中で扱うための演算子の一種である。reset で継続を区切り、shift で区切られた継続を取り除いて変数に束縛して利用する。本稿では reset を $\langle M \rangle$ と書き、継続を $\langle\rangle$ の位置で区切る。shift を $\xi k.\, M$ と書き、その時点の継続を取り除いて変数 $k$ に束縛して $M$ を実行する。例えば、$3 + \langle 2 + \xi k.\, k\,(k\,5)\rangle$ は以下のように実行される。shift/reset の正確な意味論は、通常 CPS インタプリタまたは CPS 変換によって与えられる。

$$
\begin{array}{rl}
 & 3 + \langle 2 + \xi k.\, k\,(k\,5)\rangle \\
\rightsquigarrow & 3 + \langle (k\,(k\,5))[\lambda v.\, \langle 2 + v\rangle / k]\rangle \\
\rightsquigarrow & 3 + \langle (k\,7)[\lambda v.\, \langle 2 + v\rangle / k]\rangle \\
\rightsquigarrow & 3 + 9 \\
\rightsquigarrow & 12
\end{array}
$$

$\xi k.\, k\,(k\,5)$ を計算しようとするとき、reset によって区切られた継続は $2 + [\cdot]$ である。そのため shift の実行によりその継続が取って来られ、$k$ に $\lambda v.\, \langle 2 + v\rangle$ が束縛される。

## 3 $\lambda$ 計算の部分評価器

left-to-right の値呼びの $\lambda$ 計算に対する部分評価器は以下のように定義される。$\rho$ は部分評価時の環境である。基本的には、static な入力はインタプリタと同じように実行し、dynamic な入力はコードに残している。

$$
\begin{array}{rcl}
\mathcal{P}_1\,[\![\mathrm{Var}(n)]\!]\,\rho & = & \rho\,(n) \\
\mathcal{P}_1\,[\![\overline{\mathrm{Lam}}(n, W)]\!]\,\rho & = & \lambda x.\, \mathcal{P}_1\,[\![W]\!]\,\rho[x/n] \\
\mathcal{P}_1\,[\![\underline{\mathrm{Lam}}(n, W)]\!]\,\rho & = & \mathrm{lam}(\lambda x.\, \langle \mathcal{P}_1\,[\![W]\!]\,\rho[\mathrm{var}(x)/n]\rangle) \\
\mathcal{P}_1\,[\![\overline{\mathrm{App}}(W_1, W_2)]\!]\,\rho & = & (\mathcal{P}_1\,[\![W_1]\!]\,\rho)\,(\mathcal{P}_1\,[\![W_2]\!]\,\rho) \\
\mathcal{P}_1\,[\![\underline{\mathrm{App}}(W_1, W_2)]\!]\,\rho & = & \xi k.\, \mathrm{let}(\mathrm{app}(\mathcal{P}_1\,[\![W_1]\!]\,\rho, \mathcal{P}_1\,[\![W_2]\!]\,\rho), \mathrm{lam}(\lambda t.\, k\,(\mathrm{var}(t))))
\end{array}
$$

dynamic な関数呼び出しの特化において、let-insertion [4] を行っている。let-insertion を行わなければ、部分評価器の正当性は言えない。dynamic な関数呼び出しの特化が

$$\mathcal{P}_1\,[\![\underline{\mathrm{App}}(W_1, W_2)]\!]\,\rho = \mathrm{app}(\mathcal{P}_1\,[\![W_1]\!]\,\rho, \mathcal{P}_1\,[\![W_2]\!]\,\rho)$$

と定義されていた場合を考える。このときに部分評価器が正しくない動作をする例として例 1 を見る。$W_{diverge}$ は $M_{diverge}$ を、$W$ は $M$ を束縛時解析したものとなっている。

**例 1.**

$$
\begin{array}{rcl}
W & = & \overline{\mathrm{App}}(\overline{\mathrm{Lam}}(x, \underline{\mathrm{Lam}}(y, \mathrm{Var}(y))), W_{diverge}) \\
M & = & \mathrm{App}(\mathrm{Lam}(x, \mathrm{Lam}(y, \mathrm{Var}(y))), M_{diverge}) \\
 & & \lambda \text{ 式の表記：}\ (\lambda x.\, \lambda y.\, y)\, M_{diverge} \\
W_{diverge} & = & \underline{\mathrm{App}}(\underline{\mathrm{Lam}}(x, \underline{\mathrm{App}}(\mathrm{Var}(x), \mathrm{Var}(x))), \underline{\mathrm{Lam}}(x, \underline{\mathrm{App}}(\mathrm{Var}(x), \mathrm{Var}(x)))) \\
M_{diverge} & = & \mathrm{App}(\mathrm{Lam}(x, \mathrm{App}(\mathrm{Var}(x), \mathrm{Var}(x))), \mathrm{Lam}(x, \mathrm{App}(\mathrm{Var}(x), \mathrm{Var}(x)))) \\
 & & \lambda \text{ 式の表記：}\ (\lambda x.\, x\, x)\, (\lambda x.\, x\, x)
\end{array}
$$

$\mathcal{P}_1[\![W]\!]\rho$ を計算すると、$\mathcal{P}_1[\![W_{diverge}]\!]\rho$ の周りの継続が $\mathcal{P}_1[\![W_{diverge}]\!]\rho$ の結果を捨てるため、$\mathrm{lam}(\lambda y.\,\mathrm{var}(y))$ となる。一方で、$M_{diverge}$ は発散する項のため、$M$ はインタプリタで評価すると発散する。

$W_{diverge}$ に dynamic な入出力、副作用、例外などの他の effect が入っていた場合も同様に、変換前に存在していた effect が変換後には消えてしまう結果となる。部分評価器が正しいということは、プログラムの意味が変換前と変換後で変化しないことである。変換前に存在していた effect が変換後には消えてしまう場合は部分評価器が正しくない。

$W_{diverge}$ の dynamic な関数呼び出しのように、dynamic な関数呼び出しは停止しない dynamic な計算を作っている可能性があり、let-insertion で出力項に残留されなければならない。

そこで dynamic な関数呼び出しの特化では、そのときの継続 $k$ を shift で捉えた上で、関数呼び出しを残留させ、それを $t$ という変数に束縛し、継続 $k$ に $t$ を渡している。

$$\mathcal{P}_1[\![\underline{\mathrm{App}}(W_1,W_2)]\!]\rho = \xi k.\,\mathrm{let}(\mathrm{app}(\mathcal{P}_1[\![W_1]\!]\rho,\mathcal{P}_1[\![W_2]\!]\rho),\mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t))))$$

これにより、継続 $k$ が引数を捨ててしまうような継続だったとしても、dynamic な関数呼び出しは出力されるプログラムに残る。例 1 の $W$ 部分評価では、$W_{diverge}$ の 3 つの dynamic な関数呼び出しが let-insertion により残留し、以下のような出力項を得る。

$$\begin{aligned}&\mathrm{reset}(\mathrm{let}(\mathrm{app}(\mathrm{lam}(\lambda x_2.\,\mathrm{reset}(\mathrm{let}(\mathrm{app}(\mathrm{var}(x_2),\mathrm{var}(x_2)),\lambda t_2.\,\mathrm{var}(t_2)))),\\ &\qquad\mathrm{lam}(\lambda x_3.\,\mathrm{reset}(\mathrm{let}(\mathrm{app}(\mathrm{var}(x_3),\mathrm{var}(x_3)),\lambda t_3.\,\mathrm{var}(t_3))))),\\ &\qquad\lambda t_1.\,\mathrm{lam}(\lambda y_1.\,\mathrm{var}(y_1))))\end{aligned}$$

$\lambda$ 式の表記：$\langle \mathbf{let}\ \boldsymbol{t_1} = ((\lambda x_2.\,\langle \mathbf{let}\ \boldsymbol{t_2} = x_2\,x_2\ \mathbf{in}\ t_2\rangle)\,(\lambda x_3.\,\langle \mathbf{let}\ \boldsymbol{t_3} = x_3\,x_3\ \mathbf{in}\ t_3\rangle))\ \mathbf{in}\ \lambda y_1.\,y_1\rangle$

# 4 Asai による shift/reset の部分評価器 [2, 1]

## 4.1 部分評価器

left-to-right の値呼びの shift/reset に対する部分評価器も、基本的には $\lambda$ 計算に対する部分評価器と同じ考え方で定義される。

$$\begin{aligned}\mathcal{P}_2[\![\mathrm{Var}(n)]\!]\rho &= \rho\,(n)\\ \mathcal{P}_2[\![\overline{\mathrm{Lam}}(n,W)]\!]\rho &= \lambda x.\,\mathcal{P}_2[\![W]\!]\rho[x/n]\\ \mathcal{P}_2[\![\underline{\mathrm{Lam}}(n,W)]\!]\rho &= \mathrm{lam}(\lambda x.\,\mathrm{shift}(\lambda k.\,\langle\mathrm{reset}(\mathrm{app}(\mathrm{var}(k),\mathcal{P}_2[\![W]\!]\rho[\mathrm{var}(x)/n]))\rangle))\\ \mathcal{P}_2[\![\overline{\mathrm{App}}(W_1,W_2)]\!]\rho &= (\mathcal{P}_2[\![W_1]\!]\rho)\,(\mathcal{P}_2[\![W_2]\!]\rho)\\ \mathcal{P}_2[\![\underline{\mathrm{App}}(W_1,W_2)]\!]\rho &= \xi k.\,\mathrm{reset}(\mathrm{let}(\mathrm{app}(\mathcal{P}_2[\![W_1]\!]\rho,\mathcal{P}_2[\![W_2]\!]\rho),\mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t)))))\\ \mathcal{P}_2[\![\overline{\mathrm{Shift}}(n,W)]\!]\rho &= \xi k.\,\mathcal{P}_2[\![W]\!]\rho[k/n]\\ \mathcal{P}_2[\![\underline{\mathrm{Shift}}(n,W)]\!]\rho &= \xi k.\,\mathcal{P}_2[\![W]\!]\rho[\mathrm{lam}(\lambda v.\,\langle k\,(\mathrm{var}(v))\rangle)/n]\\ \mathcal{P}_2[\![\overline{\mathrm{Reset}}(W)]\!]\rho &= \langle\mathcal{P}_2[\![W]\!]\rho\rangle\end{aligned}$$

ただし、特化している最中の継続と、束縛時解析前の元のプログラムを実行している最中の継続の間の整合性が保たれるように書かれている。特に dynamic な関数の特化にはそのための工夫が施されている。

dynamic な関数の特化が

$$\mathcal{P}_2[\![\underline{\mathrm{Lam}}(n,W)]\!]\rho = \mathrm{lam}(\lambda x.\,\langle\mathcal{P}_2[\![W]\!]\rho[\mathrm{var}(x)/n]\rangle)$$

と定義されていたときを考える。$W$ に static な shift があった場合、その static な shift を特化しようとすると、shift で補足する継続は特化の最中に分からないという状態に陥る。そのため、dynamic な関数の body 部分の static な shift を特化できなくなってしまう。

そこで、特に dynamic な関数 $\underline{\mathrm{Lam}}(n, W)$ を特化する際には、入力を $\underline{\mathrm{Lam}}(n, \underline{\mathrm{Shift}}(k, \underline{\mathrm{App}}(k, W)))$ ととらえ、出力されるコードの継続 $k$ を明示した上で、その継続の下で $W$ を特化している。（詳しくは [2] を参照。）このようにすることで、入力の reset は常に static に処理できるようになるとともに、特化時に static に shift をすると、常に正しい継続を取ることができるようになっている。

### 4.2 型システム

型は以下のように定義される。

$$\tau \quad = \quad d \mid \tau/\tau \to \tau/\tau$$

$d$ は dynamic であること表し、$\tau/\tau \to \tau/\tau$ は static な関数であることを表す。

型システムは以下のように定義される。これは束縛時解析を兼ねていて $A, \alpha \vdash M : \tau, \beta\ [W]$ の前半部分は「型環境 $A$ の下で（上線や下線のない入力の）項 $M$ は $\tau$ 型を持ち、answer type を $\alpha$ から $\beta$ に変化させる」ことを意味する。さらに束縛時解析の結果 $M$ は $W$ のように static/dynamic に区分されることを示す。

$$A[n : \tau], \alpha \vdash \mathrm{Var}(n) : \tau, \alpha\ [\mathrm{Var}(n)]$$

$$\frac{A[n : \sigma], \alpha \vdash M : \tau, \beta\ [W]}{A, \delta \vdash \mathrm{Lam}(n, M) : \sigma/\alpha \to \tau/\beta, \delta\ [\overline{\mathrm{Lam}}(n, W)]} \qquad \frac{\begin{array}{c} A, \delta \vdash M_1 : \sigma/\alpha \to \tau/\epsilon, \beta\ [W_1] \\ A, \epsilon \vdash M_2 : \sigma, \delta\ [W_2] \end{array}}{A, \alpha \vdash \mathrm{App}(M_1, M_2) : \tau, \beta\ [\overline{\mathrm{App}}(W_1, W_2)]}$$

$$\frac{A[n : \tau/\delta \to \alpha/\delta], \sigma \vdash M : \sigma, \beta\ [W]}{A, \alpha \vdash \mathrm{Shift}(n, M) : \tau, \beta\ [\overline{\mathrm{Shift}}(n, W)]} \qquad \frac{A, \sigma \vdash M : \sigma, \tau\ [W]}{A, \alpha \vdash \mathrm{Reset}(M) : \tau, \alpha\ [\overline{\mathrm{Reset}}(W)]}$$

$$\frac{A[n : d], d \vdash M : d, d\ [W]}{A, \delta \vdash \mathrm{Lam}(n, M) : d, \delta\ [\underline{\mathrm{Lam}}(n, W)]} \qquad \frac{A, \delta \vdash M_1 : d, \beta\ [W_1] \quad A, d \vdash M_2 : d, \delta\ [W_2]}{A, d \vdash \mathrm{App}(M_1, M_2) : d, \beta\ [\underline{\mathrm{App}}(W_1, W_2)]}$$

$$\frac{A[n : d], \sigma \vdash M : \sigma, \beta\ [W]}{A, d \vdash \mathrm{Shift}(n, M) : d, \beta\ [\underline{\mathrm{Shift}}(n, W)]}$$

static な項の型付け規則に関しては、単純型付きの $\lambda$ 計算と同じである。一方、dynamic な項の型付け規則では、dynamic な型は一括して $d$ としている。すなわち、dynamic な項は untyped であり、停止しない項も型がつく。

### 4.3 インタプリタ

正当性の証明で必要となるインタープリタは以下のように定義される。

$$\begin{array}{rcl} \mathcal{I}[\![\mathrm{Var}(n)]\!]\,\rho & = & \rho\,(n) \\ \mathcal{I}[\![\mathrm{Lam}(n, M)]\!]\,\rho & = & \lambda x.\mathcal{I}[\![M]\!]\,\rho[x/n] \\ \mathcal{I}[\![\mathrm{App}(M_1, M_2)]\!]\,\rho & = & (\mathcal{I}[\![M_1]\!]\,\rho)\,(\mathcal{I}[\![M_2]\!]\,\rho) \\ \mathcal{I}[\![\mathrm{Shift}(n, M)]\!]\,\rho & = & \xi k.\mathcal{I}[\![M]\!]\,\rho[k/n] \\ \mathcal{I}[\![\mathrm{Reset}(M)]\!]\,\rho & = & \langle \mathcal{I}[\![M]\!]\,\rho \rangle \end{array}$$

### 4.4 shift/reset の部分評価器の正当性の証明

値呼びのメタ言語の項間の論理関係 [12] を型の構造に対する帰納法により、以下のように定義する。

$$\begin{array}{lll}(M, M') \in R_d & \iff & \mathcal{I}[\![\downarrow_n M]\!]\,\rho_{id} \sim M' \quad (n \text{ は十分に大きい}) \\ (M, M') \in R_{\sigma/\alpha\to\tau/\beta} & \iff & \forall (V, V') \in R_\sigma.\, \forall (\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha. \\ & & (\langle (\lambda v.\, K)\,(M\,V)\rangle, \langle (\lambda v'.\, K')\,(M'\,V')\rangle) \in R_\beta\end{array}$$

ここで、$\rho_{id}(n)$ は任意の $n$ に対して $\rho_{id}(n) = z_n$ と定義する。これは部分評価器の出力項あるいは束縛時解析前の元のプログラムの自由変数と、メタ言語の自由変数を関係付けるものである。前者の自由変数が $\mathrm{Var}(n)$ のとき、後者の関係付けられる自由変数を $z_n$ と表す。この論理関係は開いた項も対象に定義しているため、このような関係付けが必要となる。

また、$M$ は自由変数を持つ可能性があるため、$R_d$ の定義における $n$ は $M$ に現れる自由変数の整数より大きい必要がある。

そして、$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ は以下のように定義する。

$$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha \iff \forall (V, V') \in R_\tau.\, (\langle (\lambda v.\, K)\,V\rangle, \langle (\lambda v'.\, K')\,V'\rangle) \in R_\alpha$$

論理関係は再帰的に定義されているが、型に関する再帰的な定義となっており、定義の右辺に現れる型は元の型より小さくなっている。

また、簡単のため対象言語とメタ言語では base type を省いているが、例えば $int$ 型を追加する場合は、型の定義に static な $int$ 型を加え、以下のように static な $int$ 型の論理関係の定義を追加する。

$$(M, M') \in R_{int} \iff M \sim M'$$

このようにして定義された論理関係を使い、先行研究 [2, 1] では以下の言明が定理として述べられていた。5 節で見る通り、部分評価器 $\mathcal{P}_2$ は正しくないため、部分評価器 $\mathcal{P}_2$ を使ったこの言明は成り立たない。

**先行研究 [2, 1] では定理とされていた言明.** $A, \alpha \vdash M : \tau, \beta\ [W]$ かつ $(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ ならば $(\langle (\lambda v.\, K)\,(\mathcal{P}_2[\![W]\!]\,\rho)\rangle, \langle (\lambda v'.\, K')\,(\mathcal{I}[\![M]\!]\,\rho')\rangle) \in R_\beta$.

ここで $(\rho, \rho') \models A$ は、$(\rho, \rho') \models A \iff$ 任意の $n \in dom(A)$ について $(\rho(n), \rho'(n)) \in R_{A(n)}$ と定義する。ただし、$dom(A)$ は $A$ の定義域。

この言明の継続を初期継続にし、環境を空にすると、部分評価器の正当性を示す言明が得られる。しかしこの言明もまた、先行研究では系とされていたが、$\mathcal{P}_2$ を使っているため成り立たない。

**先行研究 [2, 1] では系とされていた言明.** $d \vdash M : d, d\ [W]$ ならば $\mathcal{I}[\![\downarrow_0 \langle \mathcal{P}_2[\![W]\!]\,\emptyset\rangle]\!]\,\rho_{id} \sim \langle \mathcal{I}[\![M]\!]\,\emptyset\rangle$.

## 5 shift/reset の部分評価器 [2, 1] の正当性の誤り

### 5.1 正当性の証明の誤り

Asai による shift/reset の部分評価器の正当性の証明 [2] は、$A \vdash M : \tau\ [W]$ の証明の構造に沿った帰納法によるものである。$A \vdash M : \tau\ [W]$ に対して適用される型規則は 7 通りであるため、証明は 7 つの場合分けからなる。このうち、static な reset の場合と static な shift の場合において、証明に誤りがあった。この二つの誤りはどちらも、継続の引数の項を値でないのに値であるとして扱っているという誤りである。

### 5.1.1 static な reset

static な reset の場合は以下のように証明されていた。補題 1、補題 2 は 7.1 節で定義されている。

**(Static Reset)**

$$\frac{A, \sigma \vdash M : \sigma, \tau \ [W]}{A, \alpha \vdash \mathrm{Reset}(M) : \tau, \alpha \ [\overline{\mathrm{Reset}}(W)]}$$

が成り立っている。示したいのは、$(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ ならば

$$(\langle(\lambda v.\, K)\,(\mathcal{P}_2\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Reset}(M)]\!]\,\rho')\rangle) \in R_\alpha$$

である。 $\mathcal{P}_2$ と $\mathcal{I}$ の定義と補題 1 より、

$$\begin{aligned}
&(\langle(\lambda v.\, K)\,(\mathcal{P}_2\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Reset}(M)]\!]\,\rho')\rangle) \in R_\alpha \\
\iff\ &(\langle(\lambda v.\, K)\,\langle\mathcal{P}_2\,[\![W]\!]\,\rho\rangle\rangle, \langle(\lambda v'.\, K')\,\langle\mathcal{I}\,[\![M]\!]\,\rho'\rangle\rangle) \in R_\alpha \\
\iff\ &(\langle(\lambda v.\, K)\,\langle(\lambda x.\, x)\,(\mathcal{P}_2\,[\![W]\!]\,\rho)\rangle\rangle, \langle(\lambda v'.\, K')\,\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle\rangle) \in R_\alpha
\end{aligned}$$

$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ より

$$(\langle(\lambda x.\, x)\,(\mathcal{P}_2\,[\![W]\!]\,\rho)\rangle, \langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle) \in R_\tau$$

を示せばよい。これは補題 2 から $(\lambda x.\, x, \lambda x'.\, x') \models \sigma \rightsquigarrow \sigma$ が成り立つため、帰納法の仮定より成り立つ。 □

この赤文字の部分が誤りであった。4.4 節でも定義した通り、$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ かつ $(V, V') \in R_\tau$ ならば $(\langle(\lambda v.\, K)\,V\rangle, (\lambda v'.\, K')\,V') \in R_\alpha$ である。つまり、$\langle(\lambda x.\, x)\,(\mathcal{P}_2\,[\![W]\!]\,\rho)\rangle$ と $\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ が値ならば、赤文字の部分は正しい。

しかし、$\langle(\lambda x.\, x)\,(\mathcal{P}_2\,[\![W]\!]\,\rho)\rangle$ と $\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ は値ではなく、さらには $\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ が正規化可能であるとは限らない。なぜならば全ての dynamic な項の型は $d$ であるため、例えば

$$\begin{aligned}
W_{diverge} \quad = \quad & \underline{\mathrm{App}}(\underline{\mathrm{Lam}}(x, \underline{\mathrm{App}}(\mathrm{Var}(x), \mathrm{Var}(x))), \underline{\mathrm{Lam}}(x, \underline{\mathrm{App}}(\mathrm{Var}(x), \mathrm{Var}(x)))) \\
& \lambda \text{ 式の表記：}\ (\lambda x.\, x\, x)\,(\lambda x.\, x\, x)
\end{aligned}$$

といった発散する項も型がつく。そのため、$\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ の $M$ は発散する項である可能性があり、$\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ が正規化可能であるとは限らない。

### 5.1.2 static な shift

static な shift の場合は以下のように証明されていた。補題 1、補題 2 は 7.1 節で定義されている。

**(Static Shift)**

$$\frac{A[n : \tau/\delta \to \alpha/\delta], \sigma \vdash M : \sigma, \beta \ [W]}{A, \alpha \vdash \mathrm{Shift}(n, M) : \tau, \beta \ [\overline{\mathrm{Shift}}(n, W)]}$$

が成り立っている。示したいのは、$(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ ならば

$$(\langle(\lambda v.\, K)\,(\mathcal{P}_2\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Shift}(n, M)]\!]\,\rho')\rangle) \in R_\beta$$

である。補題 1 と、

$$\begin{aligned}
& \langle(\lambda v.\, K)\,(\mathcal{P}_2\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho)\rangle \\
= \ & \{\text{definition of } \mathcal{P}_2\} \\
& \langle(\lambda v.\, K)\,(\xi k.\, \mathcal{P}_2\,[\![W]\!]\,\rho[k/n])\rangle \\
\sim \ & \{\textit{reset-shift}\} \\
& \langle(\lambda k.\, \mathcal{P}_2\,[\![W]\!]\,\rho[k/n])\,(\lambda a.\, \langle(\lambda v.\, K)\,a\rangle)\rangle \\
\sim \ & \{\beta_v\} \\
& \langle\mathcal{P}_2\,[\![W]\!]\,\rho[\lambda a.\, \langle(\lambda v.\, K)\,a\rangle/n]\rangle \\
\sim \ & \{\beta_\Omega\} \\
& \langle(\lambda x.\, x)\,(\mathcal{P}_2\,[\![W]\!]\,\rho[\lambda a.\, \langle(\lambda v.\, K)\,a\rangle/n])\rangle
\end{aligned}$$

と

$$
\begin{aligned}
& \langle (\lambda v'.\, K')\, (\mathcal{I}\, [\![\mathrm{Shift}(n, M)]\!]\, \rho') \rangle \\
= \quad & \{\text{definition of } \mathcal{I}\} \\
& \langle (\lambda v'.\, K')\, (\xi k.\, \mathcal{I}\, [\![M]\!]\, \rho'[k/n]) \rangle \\
\sim \quad & \{\textit{reset-shift}\} \\
& \langle (\lambda k.\, \mathcal{I}\, [\![M]\!]\, \rho'[k/n])\, (\lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle) \rangle \\
\sim \quad & \{\beta_v\} \\
& \langle \mathcal{I}\, [\![M]\!]\, \rho'[\lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle / n] \rangle \\
\sim \quad & \{\beta_\Omega\} \\
& \langle (\lambda x'.\, x')\, (\mathcal{I}\, [\![M]\!]\, \rho'[\lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle / n]) \rangle
\end{aligned}
$$

から

$$
\begin{aligned}
& (\langle (\lambda v.\, K)\, (\mathcal{P}_2\, [\![\overline{\mathrm{Shift}}(n, W)]\!]\, \rho) \rangle, \langle (\lambda v'.\, K')\, (\mathcal{I}\, [\![\mathrm{Shift}(n, M)]\!]\, \rho') \rangle) \in R_\beta \\
\iff \quad & (\langle (\lambda x.\, x)\, (\mathcal{P}_2\, [\![W]\!]\, \rho[\lambda a.\, \langle (\lambda v.\, K)\, a \rangle / n]) \rangle, \langle (\lambda x'.\, x')\, (\mathcal{I}\, [\![M]\!]\, \rho'[\lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle / n]) \rangle) \in R_\beta
\end{aligned}
$$

を示せば良い。補題 2 から $(\lambda x.\, x, \lambda x'.\, x') \models \sigma \rightsquigarrow \sigma$ であるから、

$$
(\rho[\lambda a.\, \langle (\lambda v.\, K)\, a \rangle / n], \rho'[\lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle / n]) \models A[n : \tau/\delta \to \alpha/\delta]
$$

すなわち

$$
(\lambda a.\, \langle (\lambda v.\, K)\, a \rangle, \lambda a'.\, \langle (\lambda v'.\, K')\, a' \rangle) \in R_{\tau/\delta \to \alpha/\delta} \quad .
$$

が成り立てば、帰納法の仮定より証明が終わる。

そのためには任意の $(V, V') \in R_\tau$ と $(\lambda u.\, L, \lambda u'.\, L') \models \alpha \rightsquigarrow \delta$ について

$$
(\langle (\lambda u.\, L)\, \langle (\lambda v.\, K)\, V \rangle \rangle, \langle (\lambda u'.\, L')\, \langle (\lambda v'.\, K')\, V' \rangle \rangle) \in R_\delta
$$

を示せばよい。これは、$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ であるため成り立つ。 □

この赤文字の部分が誤りであった。4.4 節でも定義した通り、$(\lambda u.\, L, \lambda u'.\, L') \models \alpha \rightsquigarrow \delta$ かつ $(V, V') \in R_\alpha$ ならば $(\langle (\lambda u.\, L)\, V \rangle, (\lambda u.\, L')\, V') \in R_\delta$ である。つまり、$\langle (\lambda v.\, K)\, V \rangle$ と $\langle (\lambda v'.\, K')\, V' \rangle$ が値ならば、赤文字の部分は正しい。しかし、$\langle (\lambda v.\, K)\, V \rangle$ と $\langle (\lambda v'.\, K')\, V' \rangle$ は値ではなく、さらには $\langle (\lambda v'.\, K')\, V' \rangle$ が正規化可能であるとは限らない。そのため、例えば $\lambda v'.\, K'$ が $\lambda v'.\, \mathcal{I}\, [\![M]\!]\, \rho'$ と表せて $M$ が発散する項だった場合、$\langle (\lambda v'.\, K')\, V' \rangle$ は発散する。つまり $\langle (\lambda v'.\, K')\, V' \rangle$ が正規化可能であるとは限らない。

## 5.2 部分評価器の正当性の誤り

5.1 節の通り部分評価器の正当性の証明に誤りがあったが、実は証明しようとしていたもの、つまり部分評価器そのものが正当ではなかった。部分評価器の入力項に dynamic な effect が含まれていた場合、その effect が出力項では消滅しているケースがあり、その場合は部分評価器が正しくない。effect が部分評価により消滅するケースは、5.1 節の正当性の証明が破綻するケースと対応しており、static な reset に由来するものと static な shift に由来するものがある。

### 5.2.1 static な reset

5.1.1 節で説明した通り、

$$
(\langle (\lambda v.\, K)\, (\mathcal{P}_2\, [\![\overline{\mathrm{Reset}}(W)]\!]\, \rho) \rangle, \langle (\lambda v'.\, K')\, (\mathcal{I}\, [\![\mathrm{Reset}(M)]\!]\, \rho') \rangle) \in R_\alpha
$$

の証明が破綻するのは、$\langle \mathcal{I}\, [\![M]\!]\, \rho' \rangle$ が発散するときである。そのような場合の例として例 2 を考える。$\mathcal{I}\, [\![M_2]\!]\, \rho'$ の計算において、$\langle \mathcal{I}\, [\![M_{diverge}]\!]\, \rho' \rangle$ は発散する。$W_2$ は $M_2$ を束縛時解析したものである。また、$W_{diverge}$ と $M_{diverge}$ は例 1 と同じである。

**例 2.**

$$\begin{aligned} W_2 &= \overline{\mathrm{App}}(\overline{\mathrm{Lam}}(x, \overline{\mathrm{Lam}}(y, \mathrm{Var}(y))), \overline{\mathrm{Reset}}(W_{diverge})) \\ M_2 &= \mathrm{App}(\mathrm{Lam}(x, \mathrm{Lam}(y, \mathrm{Var}(y))), \mathrm{Reset}(M_{diverge})) \\ &\quad \lambda \text{ 式の表記：} (\lambda x.\, \lambda y.\, y)\, \langle M_{diverge} \rangle \end{aligned}$$

$\mathcal{P}_2 [\![W_2]\!]\, \rho$ を計算すると、$\mathcal{P}_2 [\![W_{diverge}]\!]\, \rho$ の周りの継続が $\mathcal{P}_2 [\![W_{diverge}]\!]\, \rho$ の結果を捨てるため、$\mathrm{lam}(\lambda y.\, \mathrm{var}(y))$ となる。一方で $\mathcal{I} [\![M_2]\!]\, \rho'$ を計算すると発散する。変換前に存在していた停止しない計算が変換後には消えているため、部分評価器は正しくない。

3 節で見た通り、dynamic な関数呼び出しの部分評価の定義において、停止しない計算が消滅してしまうのを防ぐために、let-insertion を使って dynamic な関数呼び出しを let 文に残している。なぜ $W_{diverge}$ に含まれていた dynamic な関数呼び出しは消滅してしまったのだろうか。$W_2$ を部分評価すると次のような経過をたどる。

$$\mathcal{P}_2 [\![W_2]\!]\, \rho \quad = \quad (\lambda x'.\, \mathcal{P}_2 [\![\overline{\mathrm{Lam}}(y, \mathrm{Var}(y))]\!]\, \rho[x'/x])\, \langle \mathcal{P}_2 [\![W_{diverge}]\!]\, \rho \rangle$$

$W_{diverge}$ の dynamic な関数呼び出しを部分評価する際、let-insertion が行われるが、let 文は $\mathcal{P}_2 [\![W_{diverge}]\!]\, \rho$ を囲う reset の下に置かれる。そのため、のちのステップで let 文ごと消滅してしまう。

### 5.2.2 static な shift

5.1.2 節で説明した通り、

$$(\langle (\lambda v.\, K)\, (\mathcal{P}_2 [\![\overline{\mathrm{Shift}}(n, W)]\!]\, \rho) \rangle, \langle (\lambda v'.\, K')\, (\mathcal{I} [\![\mathrm{Shift}(n, M)]\!]\, \rho') \rangle) \in R_\beta$$

の証明が破綻するのは、捕捉した継続に渡される値を $V'$ として、$\langle (\lambda v'.\, K')\, V' \rangle$ が発散するときである。そのような場合の例として例 3 を考える。$\mathcal{I} [\![M_3']\!]\, \rho'$ の計算において、$\mathrm{Shift}(k, M_3)$ により捕捉される継続に渡す値を $V'$ として、$\langle (\lambda y'.\, \mathcal{I} [\![M_{diverge}]\!]\, \rho'[y'/y])\, V' \rangle$ は発散する。$W_3$ は $M_3$ を、$W_3'$ は $M_3'$ を束縛時解析したものである。また、$W_{diverge}$ と $M_{diverge}$ は例 1 と同じである。

**例 3.**

$$\begin{aligned} W_3' &= \overline{\mathrm{Reset}}(\overline{\mathrm{App}}(\overline{\mathrm{Lam}}(y, W_{diverge}), \overline{\mathrm{Shift}}(k, W_3))) \\ M_3' &= \mathrm{Reset}(\mathrm{App}(\mathrm{Lam}(y, M_{diverge}), \mathrm{Shift}(k, M_3))) \\ &\quad \lambda \text{ 式の表記：} \langle (\lambda y.\, M_{diverge})\, (\xi k.\, M_3) \rangle \\ W_3 &= \overline{\mathrm{App}}(\overline{\mathrm{Lam}}(x, \underline{\mathrm{Lam}}(y, \mathrm{Var}(y))), \overline{\mathrm{App}}(\mathrm{Var}(k), \overline{\mathrm{Lam}}(z, \mathrm{Var}(z)))) \\ M_3 &= \mathrm{App}(\mathrm{Lam}(x, \mathrm{Lam}(y, \mathrm{Var}(y))), \mathrm{App}(\mathrm{Var}(k), \mathrm{Lam}(z, \mathrm{Var}(z)))) \\ &\quad \lambda \text{ 式の表記：} (\lambda x.\, \lambda y.\, y)\, (k\, (\lambda z.\, z)) \end{aligned}$$

$\mathcal{P}_2 [\![W_3']\!]\, \rho$ を計算すると、$\mathrm{lam}(\lambda y.\, \mathrm{var}(y))$ となる。一方で $\mathcal{I} [\![M_3']\!]\, \rho'$ を計算すると発散する。変換前に存在していた停止しない計算が変換後には消えているため、部分評価器は正しくない。

なぜ $W_3$ に含まれていた dynamic な関数呼び出しは、let-insertion を使って let 文に残したはずであるのに消滅してしまったのだろうか。$W_3'$ を部分評価すると次のような経過をたどる。

$$\begin{aligned} & \mathcal{P}_2 [\![W_3']\!]\, \rho \\ = \; & \langle (\lambda y'.\, \mathcal{P}_2 [\![W_{diverge}]\!]\, \rho[y'/y])\, (\xi k'.\, \mathcal{P}_2 [\![W_3]\!]\, \rho[k'/k]) \rangle \\ = \; & \langle (\lambda y'.\, \mathcal{P}_2 [\![W_{diverge}]\!]\, \rho[y'/y])\, (\xi k'.\, (\lambda x'.\, \mathcal{P}_2 [\![\underline{\mathrm{Lam}}(y, \mathrm{Var}(y))]\!]\, \rho[x'/x, k'/k])\, (k'\, (\lambda z'.\, z'))) \rangle \\ \sim \; & (\lambda x'.\, \mathcal{P}_2 [\![\underline{\mathrm{Lam}}(y, \mathrm{Var}(y))]\!]\, \rho[x'/x, \lambda v.\, \langle (\lambda y'.\, \mathcal{P}_2 [\![W_{diverge}]\!]\, \rho[y'/y])\, v \rangle / k])\, (\langle \mathcal{P}_2 [\![W_{diverge}]\!]\, \rho[y'/y] \rangle) \end{aligned}$$

$W_{diverge}$ の dynamic な関数呼び出しを部分評価する際、let-insertion が行われるが、let 文は $\mathcal{P}_2 [\![W_{diverge}]\!]\, \rho[y'/y]$ を囲う reset の下に置かれる。そのため、のちのステップで let 文ごと消滅してしまう。

# 6　本稿の shift/reset の部分評価器

本稿が提案する部分評価器は以下の通りである。部分評価器・型システム共に、static な reset と static な shift 以外は先行研究 [2, 1] と変わらない。先行研究と異なる部分を赤文字で示す。

部分評価器：

$$\begin{aligned}
\mathcal{P}_3\,[\![\mathrm{Var}(n)]\!]\,\rho &= \rho\,(n)\\
\mathcal{P}_3\,[\![\overline{\mathrm{Lam}}(n,W)]\!]\,\rho &= \lambda x.\,\mathcal{P}_3\,[\![W]\!]\,\rho[x/n]\\
\mathcal{P}_3\,[\![\underline{\mathrm{Lam}}(n,W)]\!]\,\rho &= \mathrm{lam}(\lambda x.\,\mathrm{shift}(\lambda k.\,\langle\mathrm{reset}(\mathrm{app}(\mathrm{var}(k),\mathcal{P}_3\,[\![W]\!]\,\rho[\mathrm{var}(x)/n]))\rangle))\\
\mathcal{P}_3\,[\![\overline{\mathrm{App}}(W_1,W_2)]\!]\,\rho &= (\mathcal{P}_3\,[\![W_1]\!]\,\rho)\,(\mathcal{P}_3\,[\![W_2]\!]\,\rho)\\
\mathcal{P}_3\,[\![\underline{\mathrm{App}}(W_1,W_2)]\!]\,\rho &= \xi k.\,\mathrm{reset}(\mathrm{let}(\mathrm{app}(\mathcal{P}_3\,[\![W_1]\!]\,\rho,\mathcal{P}_3\,[\![W_2]\!]\,\rho),\mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t)))))\\
\mathcal{P}_3\,[\![\overline{\mathrm{Shift}}(n,W)]\!]\,\rho &= \xi k.\,\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda v.\,\xi k'.\,\mathrm{let}(k\,v,\mathrm{lam}(\lambda t.\,k'\,(\mathrm{var}(t))))/n]\\
\mathcal{P}_3\,[\![\underline{\mathrm{Shift}}(n,W)]\!]\,\rho &= \xi k.\,\mathcal{P}_3\,[\![W]\!]\,\rho[\mathrm{lam}(\lambda v.\,\langle k\,(\mathrm{var}(v))\rangle)/n]\\
\mathcal{P}_3\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho &= \xi k.\,\mathrm{let}(\langle\mathcal{P}_3\,[\![W]\!]\,\rho\rangle,\mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t))))
\end{aligned}$$

型：

$$\tau \quad = \quad d \mid \tau/\tau \to \tau/\tau$$

型システム：

$$A[n:\tau],\alpha \vdash \mathrm{Var}(n) : \tau,\alpha\ [\mathrm{Var}(n)]$$

$$\frac{A[n:\sigma],\alpha \vdash M : \tau,\beta\ [W]}{A,\delta \vdash \mathrm{Lam}(n,M) : \sigma/\alpha \to \tau/\beta,\delta\ [\overline{\mathrm{Lam}}(n,W)]} \qquad \frac{\begin{array}{c}A,\delta \vdash M_1 : \sigma/\alpha \to \tau/\epsilon,\beta\ [W_1]\\ A,\epsilon \vdash M_2 : \sigma,\delta\ [W_2]\end{array}}{A,\alpha \vdash \mathrm{App}(M_1,M_2) : \tau,\beta\ [\overline{\mathrm{App}}(W_1,W_2)]}$$

$$\frac{A[n:\tau/d \to d/d],\sigma \vdash M : \sigma,\beta\ [W]}{A,d \vdash \mathrm{Shift}(n,M) : \tau,\beta\ [\overline{\mathrm{Shift}}(n,W)]} \qquad \frac{A,\sigma \vdash M : \sigma,d\ [W]}{A,d \vdash \mathrm{Reset}(M) : d,d\ [\overline{\mathrm{Reset}}(W)]}$$

$$\frac{A[n:d],d \vdash M : d,d\ [W]}{A,\delta \vdash \mathrm{Lam}(n,M) : d,\delta\ [\underline{\mathrm{Lam}}(n,W)]} \qquad \frac{A,\delta \vdash M_1 : d,\beta\ [W_1] \quad A,d \vdash M_2 : d,\delta\ [W_2]}{A,d \vdash \mathrm{App}(M_1,M_2) : d,\beta\ [\underline{\mathrm{App}}(W_1,W_2)]}$$

$$\frac{A[n:d],\sigma \vdash M : \sigma,\beta\ [W]}{A,d \vdash \mathrm{Shift}(n,M) : d,\beta\ [\underline{\mathrm{Shift}}(n,W)]}$$

## 6.1　static な reset

5.2.1 節で見た dynamic な effect の消滅が行われないように、定義を変更する。3 節の dynamic な関数呼び出しの部分評価で見た通り、effect を捨ててしまうことを防ぐためによく用いられる手法は、let-insertion を行うことである。そこで、reset の部分評価の定義を以下のように定義することを考える。

$$\mathcal{P}_3\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho = \xi k.\,\mathrm{let}(\langle\mathcal{P}_3\,[\![W]\!]\,\rho\rangle,\mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t))))$$

こうすると、継続 $k$ に渡されるものは値 $\mathrm{var}(t)$ となる。そのため 7.2.1 節で見る通り、正当性の証明を行うことができる。また、$W$ が effect を持つ項だったとしても、let-insertion により effect が消えないようになる。

## 6.2 static な shift

5.2.2 節で見た dynamic な effect の消滅が行われないように、定義を変更する。static な reset と同様に、effect を捨ててしまうことを防ぐために let-insertion を行う。static な shift の部分評価の定義を以下のように定義することを考える。

$$\mathcal{P}_3\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho = \xi k.\,\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda v.\,\xi k'.\,\mathrm{let}(k\,v, \mathrm{lam}(\lambda t.\,k'\,(\mathrm{var}(t))))/n]$$

こうすると、継続 $k'$ に渡されるものは値 $\mathrm{var}(t)$ となる。そのため 7.2.2 節で見る通り、正当性の証明を行うことができる。また、$W$ が effect を持つ項だったとしても、let-insertion により effect が消えないようになる。

## 6.3 型付け規則における変更

### 6.3.1 static な reset

先行研究の static な reset の型付け規則は

$$\frac{A, \sigma \vdash M : \sigma, \tau\ [W]}{A, \alpha \vdash \mathrm{Reset}(M) : \tau, \alpha\ [\overline{\mathrm{Reset}}(W)]}$$

であったが、本稿の static な reset の型付け規則は次のようになっている。

$$\frac{A, \sigma \vdash M : \sigma, d\ [W]}{A, d \vdash \mathrm{Reset}(M) : d, d\ [\overline{\mathrm{Reset}}(W)]}$$

これは static な reset の特化の定義

$$\mathcal{P}_3\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho = \xi k.\,\mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\,k\,(\mathrm{var}(t))))$$

に由来している。$\tau$ が $d$ となった理由は、特化の定義において $\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle$ が let の引数になっているためである。また、$\alpha$ が $d$ となった理由は、特化の定義において $\lambda t.\,k\,(\mathrm{var}(t))$ が lam の引数になっているためである。

### 6.3.2 static な shift

先行研究の static な reset の型付け規則は

$$\frac{A[n : \tau/\delta \to \alpha/\delta], \sigma \vdash M : \sigma, \beta\ [W]}{A, \alpha \vdash \mathrm{Shift}(n, M) : \tau, \beta\ [\overline{\mathrm{Shift}}(n, W)]}$$

であったが、本稿の static な shift の型付け規則は次のようになっている。

$$\frac{A[n : \tau/d \to d/d], \sigma \vdash M : \sigma, \beta\ [W]}{A, d \vdash \mathrm{Shift}(n, M) : \tau, \beta\ [\overline{\mathrm{Shift}}(n, W)]}$$

これは static な shift の特化の定義

$$\mathcal{P}_3\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho = \xi k.\,\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda v.\,\xi k'.\,\mathrm{let}(k\,v, \mathrm{lam}(\lambda t.\,k'\,(\mathrm{var}(t))))/n]$$

に由来している。$\alpha$ が $d$ となった理由は、特化の定義において $k\,v$ が let の引数になったいるためである。また、$\delta$ が $d$ となった理由は、特化の定義において $\lambda t.\,k'\,(\mathrm{var}(t))$ が lam の引数になっているためである。

### 6.4 型の制約が厳しくなったことによる影響

let-insertion により正しい部分評価器を得ることができたが、その代わりに let-insertion は 6.3 節の通り、型の制約を厳しくした。先行研究の型システムでは型が付くが、本研究の型システムでは厳しくなった制約により型が付かない例として例 4 が考えられる。

**例 4.**

$$\begin{aligned} W_4 &= \overline{\mathrm{Reset}}(\overline{\mathrm{App}}(\overline{\mathrm{Lam}}(x, \mathrm{Var}(x)), \overline{\mathrm{Shift}}(k, \mathrm{Var}(k)))) \\ &\quad \lambda \text{ 式の表記：} \langle (\lambda x.\, x)\, (\xi k.\, k) \rangle \end{aligned}$$

6.3.1 節で見た通り、先行研究の static な reset の型付け規則の $\tau$ は $d$ となった。そのため、static な reset に囲われた項の部分評価結果が static な関数となる例 4 は型が付かない。また、6.3.2 節で見た通り、先行研究の static な shift の型付け規則の $\alpha$ と $\delta$ は $d$ となった。この点でも例 4 は型が付かない。

例 4 のように、dynamic な関数呼び出しが存在せず、let-insertion が必要ない項にも型の制約がかかってしまう。

## 7 正当性の証明

### 7.1 いくつかの性質

**補題 1** (Admissibility, Wand [12])**.** もし $M_1 \sim M_1'$ かつ $M_2 \sim M_2'$ ならば $(M_1, M_2) \in R_\tau \iff (M_1', M_2') \in R_\tau$

**証明** $\tau$ の長さ $|\tau|$ の帰納法による。$|\tau|$ は $|d| = 0$ と $|\sigma/\alpha \to \tau/\beta| = |\beta| + 1$ によって定義される。型が $d$ の場合、十分に大きい $n$ に対し、

$$\begin{aligned} (M_1, M_2) \in R_d &\iff \mathcal{I}\,[\![\downarrow_n M_1]\!]\,\rho_{id} \sim M_2 \\ &\iff \mathcal{I}\,[\![\downarrow_n M_1']\!]\,\rho_{id} \sim M_2' \\ &\iff (M_1', M_2') \in R_d \end{aligned}$$

型が $\sigma/\alpha \to \tau/\beta$ の場合、$|\beta| < |\sigma/\alpha \to \tau/\beta|$ であるから

$$\begin{aligned} & (M_1, M_2) \in R_{\sigma/\alpha \to \tau/\beta} \\ \iff\ & \forall (V_1, V_2) \in R_\sigma.\, \forall (\lambda v_1.\, K_1, \lambda v_2.\, K_2) \models \tau \rightsquigarrow \alpha.\, (\langle (\lambda v_1.\, K_1)\, (M_1\, V_1) \rangle, \langle (\lambda v_2.\, K_2)\, (M_2\, V_2) \rangle) \in R_\beta \\ \iff\ & \forall (V_1, V_2) \in R_\sigma.\, \forall (\lambda v_1.\, K_1, \lambda v_2.\, K_2) \models \tau \rightsquigarrow \alpha.\, (\langle (\lambda v_1.\, K_1)\, (M_1'\, V_1) \rangle, \langle (\lambda v_2.\, K_2)\, (M_2'\, V_2) \rangle) \in R_\beta \\ \iff\ & (M_1', M_2') \in R_{\sigma/\alpha \to \tau/\beta} \end{aligned}$$

□

**補題 2.** 任意の $\tau$ に対し、$(\lambda x.\, x, \lambda x'.\, x') \models \tau \rightsquigarrow \tau$.

**証明** $(V, V') \in R_\tau$ とする。$(\langle (\lambda x.\, x)\, V \rangle, \langle (\lambda x'.\, x')\, V' \rangle) \in R_\tau$ であることを示す。これは

$$\langle (\lambda x.\, x)\, V \rangle \ \sim\ \langle V \rangle \ \sim\ V$$

であり、$V'$ についても同じことが言えるため、補題 1 によって証明される。 □

### 7.2 正当性の証明

**定理 1.** $A, \alpha \vdash M : \tau, \beta\ [W]$ かつ $(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$, ならば
$(\langle(\lambda v.\, K)\,(\mathcal{P}_3\,[\![W]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle) \in R_\beta$.

この定理が成り立てば、継続を初期継続にし、環境を空にすることによって、欲しかった部分評価器の正当性を証明できる。

**系 1.** $d \vdash M : d, d\ [W]$ ならば $\mathcal{I}\,[\![\downarrow_0 \langle \mathcal{P}_3\,[\![W]\!]\,\emptyset\rangle]\!]\,\rho_{id} \sim \langle \mathcal{I}\,[\![M]\!]\,\emptyset\rangle$.

定理 1 と系 1 の文言は、以前の成り立っていなかった文言の $\mathcal{P}_2$ を $\mathcal{P}_3$ に変えただけである。つまり以前の文言は $\mathcal{P}_3$ に対しては成り立つことを意味している。

定理 1 の証明は、$A \vdash M : \tau\ [W]$ の証明の構造に沿った帰納法によるものである。$A \vdash M : \tau\ [W]$ に対して適用される型規則は 7 通りであるため、証明は 7 つの場合分けからなる。static な reset と static な shift の場合以外、証明は先行研究 [2] と変わらない。よって、この 2 つの場合のみ部分評価の正当性を証明する。

#### 7.2.1 static な reset

**($A, d \vdash \mathrm{Reset}(M) : d, d\ [\overline{\mathrm{Reset}}(W)]$ の場合)**

$$\frac{A, \sigma \vdash M : \sigma, d\ [W]}{A, d \vdash \mathrm{Reset}(M) : d, d\ [\overline{\mathrm{Reset}}(W)]}$$

が成り立っている。示したいのは、$(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models d \rightsquigarrow d$ ならば

$$(\langle(\lambda v.\, K)\,(\mathcal{P}_3\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Reset}(M)]\!]\,\rho')\rangle) \in R_d$$

である。 補題 1 と、

$$\begin{array}{ll} & \langle(\lambda v.\, K)\,(\mathcal{P}_3\,[\![\overline{\mathrm{Reset}}(W)]\!]\,\rho)\rangle \\ = & \{\mathcal{P}_3 \text{ の定義 }\} \\ & \langle(\lambda v.\, K)\,(\xi k.\, \mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, k\,(\mathrm{var}(t)))))\rangle \\ \sim & \{\textit{reset-shift}\} \\ & \langle(\lambda k.\, \mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, k\,(\mathrm{var}(t))))\,(\lambda x.\, \langle(\lambda v.\, K)\,x\rangle)\rangle \\ \sim & \{\beta_v\} \\ & \langle\mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, (\lambda x.\, \langle(\lambda v.\, K)\,x\rangle)\,(\mathrm{var}(t))))\rangle \\ \sim & \{\beta_v\} \\ & \langle\mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda v.\, K)\,(\mathrm{var}(t))\rangle))\rangle \\ \sim & \{\textit{reset-value}\} \\ & \mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda v.\, K)\,(\mathrm{var}(t))\rangle)) \end{array}$$

と

$$\begin{array}{ll} & \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Reset}(M)]\!]\,\rho')\rangle \\ = & \{\mathcal{I} \text{ の定義 }\} \\ & \langle(\lambda v'.\, K')\,\langle \mathcal{I}\,[\![M]\!]\,\rho'\rangle\rangle \end{array}$$

より、

$$(\mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda v.\, K)\,(\mathrm{var}(t))\rangle)),\ \langle(\lambda v'.\, K')\,\langle \mathcal{I}\,[\![M]\!]\,\rho'\rangle\rangle) \in R_d$$

を示せば良い。

ここで、補題 2 から $(\lambda x.\, x, \lambda x'.\, x') \models \sigma \rightsquigarrow \sigma$ であり、これと帰納法の仮定から $(\langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho)\rangle, \langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle) \in R_d$ が成り立つ。これを使うと上の欲しかった論理関係は、十分に大きい $n$ に対して以下のように導出できる。

$$
\begin{array}{ll}
& \mathcal{I}\,[\![\downarrow_n (\mathrm{let}(\langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda v.\, K)\,(\mathrm{var}(t))\rangle)))]\!]\,\rho_{id} \\
= & \{\text{let の定義}\} \\
& \mathcal{I}\,[\![\downarrow_n (\mathrm{app}(\mathrm{lam}(\lambda t.\, \langle(\lambda v.\, K)\,(\mathrm{var}(t))\rangle), \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle))]\!]\,\rho_{id} \\
= & \{\downarrow_n \text{の定義}\} \\
& \mathcal{I}\,[\![\mathrm{App}(\mathrm{Lam}(n, \downarrow_{n+1} \langle(\lambda v.\, K)\,(\mathrm{var}(n))\rangle), \downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle)]\!]\,\rho_{id} \\
= & \{\mathcal{I} \text{ の定義}\} \\
& (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda v.\, K)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id}[z_n/n])\,(\mathcal{I}\,[\![\downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle]\!]\,\rho_{id}) \\
= & \{\rho_{id} \text{ の定義}\} \\
& (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda v.\, K)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id})\,(\mathcal{I}\,[\![\downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle]\!]\,\rho_{id}) \\
\sim & \{(\lambda v.\, K, \lambda v'.\, K') \models d \rightsquigarrow d \text{ かつ } (\mathrm{var}(n), z_n) \in R_d\} \qquad \cdots (*) \\
& (\lambda z_n.\, \langle(\lambda v'.\, K')\, z_n\rangle)\,(\mathcal{I}\,[\![\downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle]\!]\,\rho_{id}) \\
\sim & \{\beta_\Omega\} \\
& (\lambda z_n.\, \langle(\lambda v'.\, K')\, z_n\rangle)\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho)\rangle]\!]\,\rho_{id}) \\
\sim & \{(\langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho)\rangle, \langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle) \in R_d\} \\
& (\lambda z_n.\, \langle(\lambda v'.\, K')\, z_n\rangle)\,(\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle) \\
\sim & \{\beta_\Omega\} \\
& (\lambda z_n.\, \langle(\lambda v'.\, K')\, z_n\rangle)\,(\langle \mathcal{I}\,[\![M]\!]\,\rho'\rangle) \\
\sim & \{\beta_\Omega\} \\
& \langle(\lambda v'.\, K')\,\langle \mathcal{I}\,[\![M]\!]\,\rho'\rangle\rangle
\end{array}
$$

□

最後の $\langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho)\rangle$ と $\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho')\rangle$ が関係付いていることの証明の途中にある

$$
\begin{array}{ll}
& (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda v.\, K)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id})\,(\mathcal{I}\,[\![\downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle]\!]\,\rho_{id}) \\
\sim & \{(\lambda v.\, K, \lambda v'.\, K') \models d \rightsquigarrow d \text{ かつ } (\mathrm{var}(n), z_n) \in R_d\} \qquad \cdots (*) \\
& (\lambda z_n.\, \langle(\lambda v'.\, K')\, z_n\rangle)\,(\mathcal{I}\,[\![\downarrow_n \langle \mathcal{P}_3\,[\![W]\!]\,\rho\rangle]\!]\,\rho_{id})
\end{array}
$$

という変形は、定義を変更したために行えたことである。let-insertion により、継続の引数が値、つまり変数 $\mathrm{var}(n)$ もしくは $z_n$ となり、継続が関係付いていることを利用することができた。

### 7.2.2 static な shift

$$\frac{A[n : \tau/d \to d/d], \sigma \vdash M : \sigma, \beta\ [W]}{A, d \vdash \mathrm{Shift}(n, M) : \tau, \beta\ [\overline{\mathrm{Shift}}(n, W)]}$$

が成り立っている。示したいのは、$(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow d$ ならば

$$(\langle(\lambda v.\, K)\,(\mathcal{P}_3\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho)\rangle, \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Shift}(n, M)]\!]\,\rho')\rangle) \in R_\beta$$

である。補題 1 と、

$$
\begin{aligned}
& \langle(\lambda v.\, K)\,(\mathcal{P}_3\,[\![\overline{\mathrm{Shift}}(n, W)]\!]\,\rho)\rangle \\
= \;& \{\mathcal{P}_3 \text{ の定義 }\} \\
& \langle(\lambda v.\, K)\,(\xi k.\, \mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}(k\, w, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n])\rangle \\
\sim \;& \{\textit{reset-shift}\} \\
& \langle(\lambda k.\, \mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}(k\, w, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n])\,(\lambda a.\, \langle(\lambda v.\, K)\, a\rangle)\rangle \\
\sim \;& \{\beta_v\} \\
& \langle\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}((\lambda a.\, \langle(\lambda v.\, K)\, a\rangle)\, w, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n]\rangle \\
\sim \;& \{\beta_v\} \\
& \langle\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n]\rangle \\
\sim \;& \{\beta_\Omega\} \\
& \langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n])\rangle
\end{aligned}
$$

と

$$
\begin{aligned}
& \langle(\lambda v'.\, K')\,(\mathcal{I}\,[\![\mathrm{Shift}(n, M)]\!]\,\rho')\rangle \\
= \;& \{\mathcal{I} \text{ の定義 }\} \\
& \langle(\lambda v'.\, K')\,(\xi k.\, \mathcal{I}\,[\![M]\!]\,\rho'[k/n])\rangle \\
\sim \;& \{\textit{reset-shift}\} \\
& \langle(\lambda k.\, \mathcal{I}\,[\![M]\!]\,\rho'[k/n])\,(\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle)\rangle \\
\sim \;& \{\beta_v\} \\
& \langle\mathcal{I}\,[\![M]\!]\,\rho'[\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle/n]\rangle \\
\sim \;& \{\beta_\Omega\} \\
& \langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho'[\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle/n])\rangle
\end{aligned}
$$

より、

$$
\begin{aligned}
&(\langle(\lambda x.\, x)\,(\mathcal{P}_3\,[\![W]\!]\,\rho[\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n])\rangle, \\
&\langle(\lambda x'.\, x')\,(\mathcal{I}\,[\![M]\!]\,\rho'[\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle/n])\rangle) \in R_\beta
\end{aligned}
$$

を示せば良い。

補題 2 から $(\lambda x.\, x, \lambda x'.\, x') \models \sigma \rightsquigarrow \sigma$ であるから、

$$
(\rho[\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))/n], \rho'[\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle/n]) \models A[n : \tau/d \to d/d]
$$

つまり

$$
(\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t)))), \lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle) \in R_{\tau/d \to d/d}
$$

が成り立てば、帰納法の仮定より証明が終わる。

そこで、これを示す。そのためには、任意の $(V, V') \in R_\tau$ と $(\lambda u.\, L, \lambda u'.\, L') \models d \rightsquigarrow d$ について、

$$
(\langle(\lambda u.\, L)\,((\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t)))))\, V)\rangle, \langle(\lambda u'.\, L')\,((\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle)\, V')\rangle) \in R_d
$$

を示せば良い。補題 1 と、

$$\begin{array}{ll} & \langle(\lambda u.\, L)\,((\lambda w.\, \xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, w\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t)))))\, V)\rangle \\ \sim & \{\beta_v\} \\ & \langle(\lambda u.\, L)\,(\xi k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t)))))\rangle \\ \sim & \{\mathit{reset\text{-}shift}\} \\ & \langle(\lambda k'.\, \mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, k'\,(\mathrm{var}(t))))\,(\lambda a.\, \langle(\lambda u.\, L)\, a\rangle)\rangle \\ \sim & \{\beta_v\} \\ & \langle\mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, (\lambda a.\, \langle(\lambda u.\, L)\, a\rangle)\,(\mathrm{var}(t))))\rangle \\ \sim & \{\beta_v\} \\ & \langle\mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle))\rangle \\ \sim & \{\mathit{reset\text{-}value}\} \\ & \mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle)) \end{array}$$

と

$$\begin{array}{ll} & \langle(\lambda u'.\, L')\,((\lambda a'.\, \langle(\lambda v'.\, K')\, a'\rangle)\, V')\rangle \\ \sim & \{\beta_v\} \\ & \langle(\lambda u'.\, L')\,\langle(\lambda v'.\, K')\, V'\rangle\rangle \end{array}$$

より、

$$(\mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle)), \langle(\lambda u'.\, L')\,\langle(\lambda v'.\, K')\, V\rangle\rangle) \in R_d$$

を示せば良い。これは、十分に大きい $n$ に対して以下のように導出できる。

$$\begin{array}{ll} & \mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle)) \\ = & \{\text{let の定義 }\} \\ & \mathcal{I}\,[\![\downarrow_n (\mathrm{app}(\mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle), \langle(\lambda v.\, K)\, V\rangle))]\!]\,\rho_{id} \\ = & \{\downarrow_n \text{ の定義 }\} \\ & \mathcal{I}\,[\![\mathrm{App}(\mathrm{Lam}(n, \downarrow_{n+1} \langle(\lambda u.\, L)\,(\mathrm{var}(n))\rangle), \downarrow_n \langle(\lambda v.\, K)\, V\rangle)]\!]\,\rho_{id} \\ = & \{\mathcal{I} \text{ の定義 }\} \\ & (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda u.\, L)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id}[z_n/n])\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda v.\, K)\, V\rangle]\!]\,\rho_{id}) \\ = & \{\rho_{id} \text{ の定義 }\} \\ & (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda u.\, L)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id})\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda v.\, K)\, V\rangle]\!]\,\rho_{id}) \\ \sim & \{(\lambda u.\, L, \lambda u'.\, L') \models d \rightsquigarrow d \text{ かつ } (\mathrm{var}(n), z_n) \in R_d\} \qquad \cdots (*) \\ & (\lambda z_n.\, \langle(\lambda u'.\, L')\, z_n\rangle)\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda v.\, K)\, V\rangle]\!]\,\rho_{id}) \\ \sim & \{(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow d \text{ かつ } (V, V') \in R_\tau\} \\ & (\lambda z_n.\, \langle(\lambda u'.\, L')\, z_n\rangle)\,\langle(\lambda v'.\, K')\, V'\rangle \\ \sim & \{\beta_\Omega\} \\ & \langle(\lambda u'.\, L')\,\langle(\lambda v'.\, K')\, V'\rangle\rangle \end{array}$$

□

最後の $\mathrm{let}(\langle(\lambda v.\, K)\, V\rangle, \mathrm{lam}(\lambda t.\, \langle(\lambda u.\, L)\,(\mathrm{var}(t))\rangle))$ と $\langle(\lambda u'.\, L')\,\langle(\lambda v'.\, K')\, V\rangle\rangle$ が関係付いていることの証明の途中にある

$$\begin{array}{ll} & (\lambda z_n.\, \mathcal{I}\,[\![\downarrow_{n+1} \langle(\lambda u.\, L)\,(\mathrm{var}(n))\rangle]\!]\,\rho_{id})\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda v.\, K)\, V\rangle]\!]\,\rho_{id}) \\ \sim & \{(\lambda u.\, L, \lambda u'.\, L') \models d \rightsquigarrow d \text{ かつ } (\mathrm{var}(n), z_n) \in R_d\} \qquad \cdots (*) \\ & (\lambda z_n.\, \langle(\lambda u'.\, L')\, z_n\rangle)\,(\mathcal{I}\,[\![\downarrow_n \langle(\lambda v.\, K)\, V\rangle]\!]\,\rho_{id}) \end{array}$$

という変形は、定義を変更したために行えたことである。let-insertion により、継続の引数が値、つまり変数 $\mathrm{var}(n)$ もしくは $z_n$ となり、継続が関係付いていることを利用することができた。

## 8　まとめと今後の課題

本研究では先行研究 [2, 1] の shift/reset の部分評価器の正当性の証明において、継続の引数の項を値でないのに値であるとして扱っている部分を発見した。これは、部分評価器が正しくないこと、つまりは変換前に存在していた dynamic な effect が変換後には消えてしまうことに繋がっていた。そこで、let-insertion を用いて定義を修正し、effect が消えることを防いだ新しい部分評価器を定義し、それに対して正当性の証明を行った。

今後の課題としては、厳しくなりすぎている型の制約を緩和することが挙げられる。6.3 節で説明した通り、先行研究の static な reset の型付け規則は

$$\frac{A, \sigma \vdash M : \sigma, \tau \ [W]}{A, \alpha \vdash \mathrm{Reset}(M) : \tau, \alpha \ [\overline{\mathrm{Reset}}(W)]}$$

であったが、本稿の型付け規則は次のようになっている。

$$\frac{A, \sigma \vdash M : \sigma, d \ [W]}{A, d \vdash \mathrm{Reset}(M) : d, d \ [\overline{\mathrm{Reset}}(W)]}$$

これは static な reset の特化の定義

$$\mathcal{P}_3 \, [\![\overline{\mathrm{Reset}}(W)]\!] \, \rho = \xi k. \, \mathrm{let}(\langle \mathcal{P}_3 \, [\![W]\!] \, \rho \rangle, \mathrm{lam}(\lambda t. \, k \, (\mathrm{var}(t))))$$

に由来している。しかし、static な reset の項 $\overline{\mathrm{Reset}}(W)$ に対して、$\langle \mathcal{P}_3 \, [\![W]\!] \, \rho \rangle$ の型が dynamic な型となるように制限してしまうことは本来なら避けたい。型が dynamic になってしまっているのは、入力言語の shift/reset の実行と let-insertion の実現に、どちらもメタ言語の同じ shift/reset を使っているためである。そうではなく入力言語の shift/reset を超えた位置に let-insertion を行えればこの問題は解消できる可能性がある。これを目指して、現在は CPS を 2 度行った 2CPS の体系を検討中である。

また今後の展望として、代数的エフェクトを含む他の限定継続演算子と部分評価器の組み合わせが考えられる。shift/reset はいろいろな限定継続演算子の中では扱いやすい演算子であり、他の限定継続演算子の部分評価を考える上での土台となると考えられる。

## 謝辞

有益なコメントをくださった査読者の皆様に感謝申し上げます。また、本研究は一部 JSPS 科研費 22H03563 の助成を受けたものです。

## 参考文献

[1] K. Asai. Logical relations for call-by-value delimited continuations. In *Symposium on Trends in Functional Programming*, 2005.

[2] K. Asai. Logical relations for call-by-value delimited continuations. In *Technical Report of Department of Information Science, Ochanomizu University, OCHA-IS 06-1*, April 2006.

[3] M. Biernacka and D. Biernacki. A context-based approach to proving termination of evaluation. *Electronic Notes in Theoretical Computer Science*, 249:169–192, 2009. Proceedings of the 25th Conference on Mathematical Foundations of Programming Semantics (MFPS 2009).

[4] A. Bondorf and O. Danvy. Automatic autoprojection of recursive equations with global variables and abstract data types. *Science of Computer Programming*, 16(2):151–195, 1991.

[5] O. Danvy and A. Filinski. Abstracting control. In *Proceedings of the 1990 ACM Conference on LISP and Functional Programming*, LFP '90, page 151–160, New York, NY, USA, 1990. Association for Computing Machinery.

[6] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, 1972.

[7] N. D. Jones, C. K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation.* Englewood Cliffs, NJ: Prentice Hall, 1993.

[8] Y. Kameyama and M. Hasegawa. A sound and complete axiomatization of delimited continuations. In *Proceedings of the eighth ACM SIGPLAN International Conference on Functional Programming (ICFP'03)*, August 2003.

[9] Y. Kameyama, O. Kiselyov, and C. chieh Shan. Shifting the stage: Staging with delimited control. *Journal of Functional Programming*, 21(6):617–662, 2011.

[10] R. Leißa, K. Boesche, S. Hack, A. Pérard-Gayot, R. Membarth, P. Slusallek, A. Müller, and B. Schmidt. Anydsl: a partial evaluation framework for programming high-performance libraries. *Proceedings of the ACM on Programming Languages*, 2:1 – 30, 2018.

[11] J. Oishi and Y. Kameyama. Staging with control: Type-safe multi-stage programming with control operators. In *Proceedings of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, GPCE 2017, page 29–40, New York, NY, USA, 2017. Association for Computing Machinery.

[12] M. Wand. Specifying the correctness of binding-time analysis. In *Journal of Functional Programming, Vol. 3, No. 3*, pages 365–387. Cambridge University Press, July 1993.