parametricity を用いた control/prompt の CPS 変換の正当性の証明

浦中　花菜 (指導教員：浅井 健一)

1　はじめに

継続とは、その後の計算を表す。例えば、式「2 * 4 + 6」 において、「2 * 4」 を計算した時点で、その後に行うべき計算は 「[] + 6」であり、この部分が継続と呼ばれる (ここで [] は hole と呼ばれる)。「限定継続演算子」とは、継続を利用して計算を行う演算子であり、その名の通り、限定した継続を切り取ることも可能である。限定継続演算子では、直近の $\langle\rangle$ までを継続としている ($\langle\rangle$ は prompt と呼ばれる)。例えば、式「$\langle 2*3+1\rangle + 7$」において、「2 * 3」を計算した時点での限定継続は、「$[]+1$」となっている。もし継続を捨てるならば、例外処理を一般化した動作と同様の挙動を実現することができる。しかし、継続を捨てずに活用することも可能である。

限定継続演算子の意味論を記述しているのが、CPS 変換である。CPS 変換でその動作が模倣できることを、CPS 変換の正当性という。本論文では、限定継続演算子である control/prompt の CPS 変換の正当性の証明を行う。

2　control/prompt とは

control/prompt とは、限定継続演算子の一種である。control/prompt では、$\langle e\rangle$ で、継続の範囲を e に限定することを表し、$\langle\rangle$ を prompt と呼ぶ。control は $\mathcal{F}k.e$ と記述し、継続を変数 k に束縛し、その上で式 e を実行することを表す。

3　control/prompt の簡約方法

例を用いて、contol/prompt の簡約方法の説明を行う。(4) の式が重要である。

$\langle(\mathcal{F}k_1.3+(k_1\,2))+(\mathcal{F}k_2.2*(k_2\,1))\rangle + 30$ の場合、

$$\langle(\mathcal{F}k_1.3+(k_1\,2))+(\mathcal{F}k_2.2*(k_2\,1))\rangle+30 \quad (1)$$

$$=\langle 3+(k_1\,2)[\lambda x.\,x+(\mathcal{F}k_2.2*(k_2\,1))/k_1]\rangle+30 \quad (2)$$

$$=\langle 3+(2+(\mathcal{F}k_2.2*(k_2\,1)))\rangle+30 \quad (3)$$

$$=\langle 2*(k_2\,1)[\lambda y.\,3+(2+y)/k_2]\rangle+30 \quad (4)$$

$$=\langle 2*(3+(2+1))\rangle+30 \quad (5)$$

$$=42 \quad (6)$$

実行過程を (1) から順に見ていく。最初の実行される計算は、$\langle(\mathcal{F}k_1.3+(k_1\,2))+(\mathcal{F}k_2.2*(k_2\,1))\rangle$ であり、この時点での継続は $[]+30$ である。次に、$\langle(\mathcal{F}k_1.3+(k_1\,2))+(\mathcal{F}k_2.2*(k_2\,1))\rangle$ の中で実行される計算は、式を左から順に実行すると仮定すると、$(\mathcal{F}k_1.3+(k_1\,2))$ である。この時点での継続は、$[]+(\mathcal{F}k_2.2*(k_2\,1))$ である。この式は、$\lambda x.\,x+(\mathcal{F}k_2.2*(k_2\,1))$ と表せる。この継続を k_1 に束縛し、body 部分 $3+(k_1\,2)$ の計算に移る。これを表しているのが (2) である。

次に実行されるのは、$k_1\,2$ である。計算すると、(3) の式となる。

次の (4) の式が重要である。$\langle 3+(2+(\mathcal{F}k_2.2*(k_2\,1)))\rangle$ において、次に実行される計算は、$(\mathcal{F}k_2.2*(k_2\,1))$ である。これは、(2) の式の継続の中の計算である。また、$(\mathcal{F}k_2.2*(k_2\,1))$ は control の式である。すなわち、捕捉された継続の中で、さらに継続を捕捉しようとしている。$(\mathcal{F}k_2.2*(k_2\,1))$ 時点での継続は、$\langle 3+(2+(\mathcal{F}k_2.2*(k_2\,1)))\rangle$ の直近の $\langle\rangle$ までの継続であるため、$3+(2+[])$ である。ここで、(2) の式の継続は、計算すると $2+(\mathcal{F}k_2.2*(k_2\,1))$ となる。今、$(\mathcal{F}k_2.2*(k_2\,1))$ を見ている。しかし、継続として、2+ だけでなく 3+ も捕捉されている。これが control/prompt の特徴である。ここで、2+ を継続、3+ をトレイルと呼ぶ。これを k_2 に束縛して、$2*(k_2\,1)$ を実行する。すなわち、(4) の式となる。その後は、k_2 に $\lambda y.\,3+(2+y)$ を代入して計算を行うだけである。ここで、実行は継続の 2+ が先に行われ、次にトレイルの 3+ が行われることもポイントである。

4　control/prompt の定式化

この節では、証明を行うために、control/prompt の定式化を行う。

4.1　DS 項の定義

本論文では、単純型付きラムダ計算に control/prompt を加えたものを DS 項と呼び、この言語を扱う。先行研究である [3] で扱う DS 言語と同様のものである。

4.2　簡約規則

本節では、control/prompt の簡約規則に対する定式化を行う。なお、これらの定義は先行研究である [3] においても使用されているものである。

定義 1 簡約規則

$$\frac{e[v/x]=e'}{(\lambda x.\,e)\,v\to_\beta e'}\ (RBeta) \qquad \frac{e\to_\beta e'}{F[e]\to_\beta F[e']}\ (RFrame)$$

$$\frac{}{\langle E^p[\mathcal{F}c.e]\rangle\to_\beta\langle(\lambda c.\,e)\,(\lambda x.\,E^p[x])\rangle}\ (RControl)$$

$$\frac{}{\langle v\rangle\to_\beta v'}\ (RPrompt)$$

RControl は、継続を $(\lambda x.\,E^p[x])$ という形で取り出し、それを c に代入した後、e の実行に移ることを示している。継続とは、直近の $\langle\rangle$ までの文脈である。ここで、継続として、ピュアな評価文脈 E^p を切り取っている。ピュアな評価文脈 E^p の中には、[] を囲う $\langle\rangle$ は存在しない。そのため、直近の $\langle\rangle$ までの継続を切り取ることがで可能となっている。RPrompt は、中身が値になれば、prompt を外せることを示している。

4.3　CPS 変換

次に、control/prompt の CPS 変換を示す。本論文の目標は、この CPS 変換が、定義 1 の簡約規則と同様の挙動であると証明することである。この CPS 変換は [1] から引用したものである。ρ は環境を表し、k

は継続を、t はトレイルを表す。$\rho[\![x]\!]$ は、ρ の中の x に対応するものを表している。CPS 変換は、継続渡し形式で書かれている。最初は初期継続 id_k と空のトレイル () とともに、式を実行する。

定義 2 CPS 変換の定義

$$
\begin{aligned}
[\![n]\!]\rho &= \lambda k.\lambda t.k\ n\ t \\
[\![x]\!]\rho &= \lambda k.\lambda t.k\ (\rho[\![x]\!])\ t \\
[\![\lambda x.e]\!]\rho &= \lambda k.\lambda t.k\ (\lambda v.\lambda k_1.\lambda t_1.[\![e]\!]\rho[v/x]\ k_1\ t_1)\ t \\
[\![e_1\,e_2]\!]\rho &= \lambda k.\lambda t.[\![e_1]\!]\rho\ (\lambda v_1.\lambda t_1.[\![e_2]\!]\rho \\
&\quad (\lambda v_2.\lambda t_2.v_1\ v_2\ k\ t_2)t_1)\ t \\
[\![\mathcal{F}c.e]\!]\rho &= \lambda k.\lambda t.[\![e]\!]\rho \\
&\quad [\lambda x.\lambda k_1.\lambda t_1.k\ x\ (t@(k_1 :: t_1))/c]id_k() \\
[\![\langle e\rangle]\!]\rho &= \lambda k.\lambda t.k\ ([\![e]\!]\rho\ id_k\ ())\ t \\
id_k &= \lambda v.\lambda t.case\ t\ of\ () \Rightarrow v \mid k \Rightarrow k\ v\ () \\
\_@\_ &= \lambda t.\lambda t_1.case\ t\ of\ () \Rightarrow t_1 \mid k \Rightarrow k :: t_1 \\
\_ :: \_ &= \lambda k.\lambda t.case\ t\ of\ () \Rightarrow k \mid k_1 \Rightarrow \\
&\quad \lambda v.\lambda t_1.k\ v\ (k_1 :: t_1)
\end{aligned}
$$

5 CPS 変換の正当性

本論文の目標は、control/prompt の CPS 変換の正当性を証明することである。[3] において、control 規則を除き、正当性の証明がなされている。また、control 規則においても、$\lambda v'.\lambda k'.\lambda t'.[\![e]\!][v'/x]\ (\lambda v_0.(\lambda t_0.\mathrm{id}_k\ v_0\ t_0))\ (k' :: t') \approx \lambda v'.\lambda k'.\lambda t'.[\![e]\!][v'/x]\ \ (\lambda v_0.(\lambda t_0.\mathrm{id}_k\ v_0\ (k' :: t_0))\ \ )t')$ を示せれば、正当性の証明が完成することが分かっている。$\approx$ は 2 つの式を外から見た時の挙動が同じことを示している。

ここで、この 2 つの式の違いは、継続とトレイルである。左辺は継続が $(\lambda v_0.(\lambda t_0.\mathrm{id}_k\ v_0\ t_0))$ でトレイルが $(k' :: t')$、右辺は、継続が $(\lambda v_0.(\lambda t_0.\mathrm{id}_k\ v_0\ (k' :: t_0)))$ でトレイルが t' である。しかし、この 2 つの式は同じ入力に対し、同じ出力を返すことが分かる。継続の最後の位置で行う処理をトレイルの最初に移動しても、式の意味は変わらないからである。

よって、本論文では、継続の最後の位置で行う処理をトレイルの最初に移動しても、式の意味は変わらない、という性質を入れ込んだ論理関係を定義し、この式の証明を行う。

6 論理関係

論理関係を用いて定義することで、$\approx$ の定義を行う。論理関係を下の定義 3,4 で定義する。これは、型に基づき関連付けを行なっている。型は [1] に基づく。$(v_1, v_2) \in R_{\tau\to\tau'\langle\mu_\alpha\rangle\alpha\langle\mu_\beta\rangle\beta}$ が、v_1 と v_2 が外から見た挙動が同じであることを表している。すなわち、これが $\approx$ の定義である。これは、2 つの式に関連づいた「継続とトレイルのペア」を渡すことによって、関連づいた値を返すことを意味している。関連づいた「継続とトレイルのペア」というのは、$((k_1, t_1), (k_2, t_2)) \in KT_{(\tau\to\langle\mu_\alpha\rangle\alpha,\mu_\beta)}$ で定義している。ここで、継続とトレイルをペアにして定義を行っているのは、継続の最後の位置で行う処理をトレイルの最初に移動しても、式の意味は変わらない、という性質を入れ込むためである。

定義 3 式の論理関係の定義
$e \to_\beta^* v, e' \to_\beta^* v'$かつ $(v, v') \in R_\tau$のとき、$(e, e') \in E_\tau$

定義 4 値の論理関係の定義

$(n_1, n_2) \in R_{int} \Leftrightarrow n_1 = n_2$
$((k_1, t_1), (k_2, t_2)) \in KT_{(\tau\to\langle\mu_\alpha\rangle\alpha,\mu_\beta)} \Leftrightarrow$
$compatible(\mu_\beta, \mu_0, \mu_\alpha)$ を満たす任意のμ_0について
$\forall(v_1, v_2) \in R_\tau.\forall(t_1', t_2') \in T_{\mu_0}.$
$(k_1\ v_1\ (t_1@t_1'), k_2\ v_2\ (t_2@t_2')) \in E_\alpha$
$(v_1, v_2) \in R_{\tau\to\tau'\langle\mu_\alpha\rangle\alpha\langle\mu_\beta\rangle\beta} \Leftrightarrow$
$\forall(v_1', v_2') \in R_\tau.\forall((k_1, t_1), (k_2, t_2)) \in KT_{(\tau'\to\langle\mu_\alpha\rangle\alpha,\mu_\beta)}.$
$(v_1\, v_1'\ k_1\ t_1, v_2\, v_2'\ k_2\ t_2) \in E_\beta$
$(t_1, t_2) \in T_\bullet \Leftrightarrow t_1 = t_2 = ()$
$(t_1, t_2) \in T_{\tau\to\langle\mu_\alpha\rangle\alpha} \Leftrightarrow$
$\forall(v_1, v_2) \in R_\tau.\forall(t_1', t_2') \in T_{\mu_\alpha}.(t_1\ v_1\ t_1', t_2\ v_2\ t_2') \in E_\alpha$

7 control/proompt の CPS 変換の正当性

6 節の論理関係を用いると、示したい式は

定理 5 (k の移動) $(\lambda v'.\lambda k'.\lambda t'.[\![e]\!][v'/x]\ (\lambda v_0.(\lambda t_0.id_k\ v_0\ t_0))\ (k' :: t'), \lambda v'.\lambda k'.\lambda t'.[\![e]\!][v'/x]\ (\lambda v_0.(\lambda t_0.id_k\ v_0\ (k' :: t_0)))\ t') \in R_{\tau\to\tau'\langle\mu_\alpha\rangle\alpha\langle\mu_\beta\rangle\beta}$

と書ける。これは、DS 言語において、parametricity が成り立つことを 6 節の論理関係を用いて証明することで、証明可能である。

8 parametricity

自分自身と関連づいていることを parametricity [2]、または、論理関係の基本定理と呼ぶ。

定理 6 (parametricity) $\Gamma \vdash e : \tau\langle\mu_\alpha\rangle\alpha\langle\mu_\beta\rangle\beta$ ならば、Γ に対応する任意の ρ, ρ' と、任意の $((k, t), (k', t')) \in KT_{(\tau\to\langle\mu_\alpha\rangle\alpha,\mu_\beta)}$ について $([\![e]\!]\rho\ k\ t, [\![e]\!]\rho'\ k'\ t') \in E_\beta$

同じ式に関連づいた「継続とトレイルのペア」を渡したら、必ず関連づいた結果を返すことを示している。

9 まとめ

継続とトレイルは必ずペアで使われるという性質を取り入れて、論理関係を定義することで、parametricity を用いた control/prompt の CPS 変換の正当性の証明を可能にした。

参考文献

[1] Youyou Cong, Chiaki Ishio, Kaho Honda, and Kenichi Asai. A functional abstraction of typed invocation contexts. *Logical Methods in Computer Science*, Vol. 18, No. 3, 2022.

[2] John C. Reynols. Types, Abstraction and Parametric Polymorphism. In R.E.A. Mason, editor, *Information Processing 83*, Vol. 9 of *IFIP Congress Series*, pp. 513–523, Amsterdam, The Netherlands, 1983. Elsevier Science Publishers B.V.

[3] 本田華歩. Agda を使った限定継続演算子の性質の証明. 修士論文, お茶の水女子大学, 2023.