論理関係によるスタック導入の正当性の証明

新井祐美　(指導教員：浅井健一)

1　はじめに

本研究では、先行研究 [1] で提案されているプログラム変換の一手法であるスタック導入について、論理関係を用いてその前後の等価性を証明する。

スタック導入とは、より現実の機械に近い形式の抽象機械を導出するためのプログラム変換であり、これによってプログラム中の関数呼出しは、値の受け渡しをスタックの先頭で行うように書き換えられる。証明の方針は、まずスタック導入前後の値についての対応を論理関係 [2] として定義し、入力データの構造についての帰納法によって、両者の出力結果が（論理関係の意味で）等しいことを示していく。

これにより、CPS の λ 計算のインタプリタにスタック導入することの正当性が保証される。

2　スタック導入

先行研究は、限定継続処理のオペレータ shift / reset に関して、正当性の保証された処理系を与えることを目的としている。その手法として、CPS の λ 計算のインタプリタにスタック導入を含む各種プログラム変換を用いて、より抽象機械の動作に近いインタプリタを導出し、機械語での限定継続処理実装のための正当性の保証のあるモデルを提案しようとしている。

スタック導入前のインタプリタでは、値はそれを取る継続ごとに保持されているが、一般的な機械語ではスタックに中間結果を保存するようになっている。このため先行研究では、新たに評価器に引数としてスタックを持たせ、値をスタックに積んで受け渡すように変更し、このプログラム変換をスタック導入と呼んでいる。

スタック導入前後のインタプリタについて、型無しの言語を入力した場合、両者で同じように実行が進むことが先行研究において証明されているが、本研究では、型付きの言語を入力した場合、結果が等しいということを証明する。

3　論理関係

本研究では、論理関係を用いた証明の手法を採用した。この手法は、型付きの言語を対象としている。

まず型に関する帰納法によって、等価性を保証するための関係を定義する。そして、関係性を示したい2つの項がともに定義した関係を満たしていれば、それらは等価であるということが証明できる、というものである。

4　CPS の λ 計算のインタプリタ

本論で扱った CPS の λ 計算のインタプリタ $eval_1$ および $eval_1$ にスタック導入を施した $eval_2$ の振る舞いを図 1 に示した。

4.1　CPS（継続渡し形式）

継続とは、計算が終わったあとにする処理のことである。継続渡し形式は、継続を引数として渡すことで、すべての関数を末尾呼出しにすることを言う。$eval_1$ と $eval_2$ はともに CPS で書かれている。

4.2　λ 計算

λ 計算は、関数の計算に関する理論である。

本研究では、整数型を基本型とし、以下のように定義される型付きの λ 式を入力言語としている。

項： $t := n$ (整数) | x (変数)
　　| $\lambda x.M$ (λ抽象) | MN (関数適用)
型： $T := int \mid T_1 \to T_2$
また、型付け規則は以下の通りである。

$\Gamma \vdash n : int$　　(整数)

$\Gamma,\ x : T_1 \vdash x : T_1$　　(変数)

$$\frac{\Gamma, x : T_1 \vdash M : T_2}{\Gamma \vdash \lambda x.M : T_1 \to T_2}\ (\lambda\text{ 抽象})$$

$$\frac{\Gamma \vdash M : T_1 \to T_2 \qquad \Gamma \vdash N : T_1}{\Gamma \vdash MN : T_2}\ (\text{関数適用})$$

5　定理

図 1 に示した $eval_1$ と $eval_2$ の等価性を示すために、まず、次の定理を証明する。

定理： $\Gamma \vdash t : T$ と型付けされた λ 式 t について、

$R_A((eval_1\ t\ env_1\ cont_1),\ fst(eval_2\ t\ env_2\ stack\ cont_2))$

ただし、
1. $\forall x \in dom(\Gamma)$ について、$R_{\Gamma(x)^*}(env_1(x),\ env_2(x))$
2. $stack$ は任意のリスト
3. $R_{T^* \to A}(cont_1,\ cont_2)$
4. T^* は継続の型であり、以下の定義に従う。
   - $(int)^* = int$
   - $(T_1 \to T_2)^* = T_1^* \to (T_2^* \to A) \to A$
     $A : Answer\ Type$
5. $fst()$ はリストを受け取ったらその先頭の要素を返す関数とする。

また、論理関係 R_T は、次のように定める。

1. $R_{(int)^*}(t_1,\ t_2) \Leftrightarrow \forall n : int$ について、
   $t_1 = VInt_1(n)$ かつ $t_2 = VInt_2(n)$

2. $R_{T_1^* \to A}(k_1,\ k_2) \Leftrightarrow \forall (v_1,\ v_2) \in R_{T_1^*}$、$\forall s$ について、$R_A(k_1 v_1,\ fst(k_2(v_2 :: s)))$

3. $R_{(T_1 \to T_2)^*}(VFun_1(f_1),\ VFun_2(f_2))$
   $\Leftrightarrow \forall (v_1,\ v_2) \in R_{T_1^*}$、$\forall (k_1,\ k_2) \in R_{T_2^* \to A}$、$\forall s$ について、$R_A((f_1\ v_1\ k_1),\ fst(f_2(v_2 :: s)\ k_2))$

R_T は型 T に関して帰納的に定義されている。これは前節の T の構造に従っているので、すべての入力言語について論理関係が定義できていることがわかる。

$eval_1\ n\ env_1\ cont_1 \rightsquigarrow cont_1(VInt_1(n))$	$eval_2\ n\ env_2\ stack\ cont_2 \rightsquigarrow cont_2(VInt_2(n) :: stack)$
$eval_1\ x\ env_1\ cont_1 \rightsquigarrow cont_1(env_1(x))$	$eval_2\ x\ env_2\ stack\ cont_2 \rightsquigarrow cont_2(env_2(x) :: stack)$
$eval_1\ (\lambda x.M)\ env_1\ cont_1$	$eval_2\ (\lambda x.M)\ env_2\ stack\ cont_2$
$\rightsquigarrow cont_1(VFun_1(fun\ v_1\ k_1 \rightarrow$	$\rightsquigarrow cont_2(VFun_2(fun\ (v_2 :: s)\ k_2 \rightarrow$
$\quad eval_1\ M\ (x,\ v_1) :: env_1\ k_1)$	$\quad eval_2\ M\ (x,\ v_2) :: env_2\ s\ k_2)$
$eval_1\ MN\ env_1\ cont_1$	$eval_2\ MN\ env_2\ cont_2$
$\rightsquigarrow eval_1\ N\ env_1\ (fun\ v_2 \rightarrow$	$\rightsquigarrow eval_2\ N\ env_2\ stack\ (fun\ s_1 \rightarrow$
$\quad eval_1\ M\ env_1\ (fun\ v_1 \rightarrow$	$\quad eval_2\ M\ env_2\ s_1\ (fun\ s_2 \rightarrow$
$\quad match\ v_1\ with$	$\quad match\ s_2\ with$
$\quad VFun_1(f_1) \rightarrow f_1\ v_2\ cont_1$	$\quad VFun_2(f_2) :: v_2' :: s \rightarrow f_2\ (v_2' :: s)\ cont_2$
$\mid \_ \rightarrow raise\ Error$	$\mid \_ \rightarrow raise\ Error$

図 1: $eval_1, eval_2$ の振る舞い

$eval_1$ と $eval_2$ では継続の型が違うので、等価性を考えることが難しくなる。しかし、両者の関係性を定義する論理関係を用いることにより、入力言語の性質そのものについて議論できるようになるため、この問題は解消される。

6 証明

t の構造についての帰納法により、各入力に対する出力が定義した論理関係を満たしていることを示す。

(1) $t = n,\ T = int$（整数）のとき

論理関係 1 および env_1, env_2 についての前提より、論理関係 2 が適用できて、
$R_A(cont_1(VInt_1(n)), fst(cont_2(VInt_2(n) :: stack)))$

(2) $t = x,\ T = \Gamma(x)$（変数）のとき

env_1, env_2 および $cont_1, cont_2$ についての前提より、論理関係 2 が適用できて、
$R_A(cont_1(env_1(x)),\ fst(cont_2(env_2(x) :: stack))))$

(3) $t = \lambda x.M,\ T = T_1 \rightarrow T_2$（$\lambda$ 抽象）のとき

実行結果を $cont_1(VFun_1(f_1))$、$cont_2(VFun_2(f_2) :: stack)$ とおく。

$f_1,\ f_2$ に $\forall(v_1,\ v_2) \in R_{T_1^*}$、$\forall(k_1,\ k_2) \in R_{T^* \rightarrow A}$、$\forall s$ を渡すことを考えれば、型に関する帰納法の仮定が使える。

論理関係 3 より、$R_{(T_1 \rightarrow T_2)^*}(VFun_1(f_1),\ VFun_2(f_2))$ である。

また、$cont_1, cont_2$ についての前提より、論理関係 2 が適用できて、
$R_A(cont_1(VFun_1(f_1)), fst(cont_2(VFun_2(f_2)::stack)))$

(4) $t = MN,\ T = T_2$（関数適用）のとき

型に関する帰納法の仮定より、$\Gamma \vdash M : T_1 \rightarrow T_2$、$\Gamma \vdash N : T_1$ が得られる。各継続に $\forall(x_1,\ x_2) \in R_{T_1^*}$、$\forall s$ を渡すことを考えれば、帰納法の仮定を使うための前提条件が満たされるので、
$R_A((f_1\ v_2\ cont_1),\ fst(f_2\ (v_2' :: s)\ cont_2))$

以上より、$\Gamma \vdash t : T$ と型付けされた入力に対して、$R_A((eval_1\ t\ env_1\ cont_1),\ fst(eval_2\ t\ env_2\ stack\ cont_2))$ が成り立つ。

7 スタック導入の正当性の証明

5 節の定理を証明できたことにより、次の系を得る。

系： $\vdash t : int$ と型付けされたプログラムについて、

$$eval_1\ t\ \phi\ id = fst(eval_2\ t\ \phi\ []\ id)$$

ただし、id は恒等関数である。

証明

命題は、前定理に関して、型環境 Γ、環境 env_1、env_2 および $stack$ が空、初期継続が id の場合であると言える。

今、Γ が空であるから、$x \in dom(\Gamma)$ を満たす x が存在しないため、前定理の環境についての前提を満たす。

また、$\forall(v_1,\ v_2) \in R_{(int)^*}$、$\forall s$ について、
$R_{int}((id\ \ v_1),\ \ fst(id\ \ (v_2\ \ ::\ \ s)))$ が成り立つので、$R_{(int)^* \rightarrow int}(id,\ id)$ となり、これは前定理の継続についての前提を満たす。

よって、前定理よりこれは成り立ち、スタック導入の正当性が証明された。

8 まとめと今後の課題

CPS の λ 計算のインタプリタにスタック導入を用いるとき、入力される λ 式が型付けされており、値が整数であれば結果は等しいということを証明した。今後は、先行研究の本題である限定継続命令を含めたインタプリタへのスタック導入の正当性を確かめたい。今回見てきたインタプリタで限定継続命令を扱えるようにするためには、対象言語に shift と reset を付け足すことになるため、本論で定義した論理関係ではすべての入力を網羅することができない。しかし、限定継続を表す項の性質をよく観察し、スタック導入前後の対応を記述する論理関係を定義することができれば、本論と同様に証明できると予想される。

参考文献

[1] 木谷有沙, 浅井健一. 「仮想機械導出のためのプログラム変換」. 第 12 回プログラミングおよびプログラミング言語ワークショップ (PPL2010) 掲載予定, January 2010.

[2] 萩谷昌己, 西崎真也. 論理と計算のしくみ. 岩波書店, 2007.