対称ラムダ計算の定式化

理学専攻　情報科学コース　新井祐美（指導教員：浅井健一）

1　はじめに

継続とは，計算のある時点における残りの計算を表す．対称ラムダ計算 (Symmetric Lambda Calculus，以下 SLC) とは，項と継続が対称になっており，項を扱うのと同様に継続を扱うことができる計算体系である．1989 年に Filinski によって初めて提唱され [1]，阪上らによって small-step semantics で定式化し直された [2]．また，上田ら [3] によって，種々の性質が示されつつある．本稿では，これまでに示されてきた SLC の性質の正しさを裏付けする一手法として，定理証明系 Coq による定式化を試みている．

具体的には，上田らによる非決定性 SLC の syntax，types，typing rule，reduction rule を Coq で定義し，型付きの言語が正当であるために必要な性質の 1 つである preservation の証明を定式化した．次に，call-by-value SLC の停止性を示すため，値，項，関数，継続のそれぞれが停止するという論理述語を定義し，論理関係の手法によるこの証明が Coq によって定式化可能であるかを検証している．

2　非決定性 SLC

本稿で定式化した非決定性 SLC について述べる．

2.1　Syntax

項	e	::=	$n \mid x \mid e \uparrow f \mid \lceil f \rceil$
関数	f	::=	$g \mid p \Rightarrow e \mid c \Leftarrow q \mid \overline{e} \mid \underline{c}$
継続	c	::=	$\bullet \mid y \mid f \downarrow c \mid \lfloor f \rfloor$
項のパターン	p	::=	$x \mid \lceil g \rceil$
継続のパターン	q	::=	$y \mid \lfloor g \rfloor$

SLC では項，関数，継続の 3 つを組み合わせて計算状態を表しており，それらは相互に再帰する構造になっている．項についての λ 抽象を表す $p \Rightarrow e$，継続についての λ 抽象を表す $c \Leftarrow q$ に現れる p, q は束縛変数に相当し，項（継続）型の変数または項（継続）として扱う関数型の変数になる．

本稿では，まず p, q をデータ型として定義し，それを用いて e, f, c を相互再帰させる形で定義している．

2.2　Types

T	::=	$+A$	(types of expressions e)
	|	$A \supset B$	(types of functions f)
	|	$\neg B$	(types of continuations c)
A, B	::=	$\bot \mid \top \mid \mathbf{int} \mid A \wedge B \mid A \vee B \mid A \to B \mid A - B$	

SLC では，型も e, f, c それぞれについて定義されている．A, B は neutral type とし，3 つの型をより細分化する．例えば，項として扱う関数 $\lceil f \rceil$ は，型規則によると $+(A \to B)$ のように表される．本稿では，e, f, c の型と neutral type を互いに相互再帰するデータ型として定義し，次に述べる型規則を，項型・関数型・継続型が相互再帰する，Prop 型を返す述語として定義した．

2.3　Typing rule

本稿で定式化した SLC の型規則を一部抜粋する．

$$\frac{}{\Gamma_g \cup \{g : A \supset B\} \vdash g : A \supset B}\ \textbf{TFVar}$$

$$\frac{}{\Gamma_x \cup \{x : +A\} \vdash x : +A}\ \textbf{TVar} \qquad \frac{}{\Gamma_y \cup \{y : \neg B\} \vdash y : \neg B}\ \overline{\textbf{TVar}}$$

$$\frac{\Gamma \vdash_x x : +A \quad \Gamma \vdash e : +B}{\Gamma_x \vdash x \Leftarrow e : A \supset B}\ \textbf{TFun}_1 \qquad \frac{\Gamma \vdash c : \neg A \quad \Gamma \vdash_y y : \neg B}{\Gamma_y \vdash c \Rightarrow y : A \supset B}\ \overline{\textbf{TFun}_1}$$

$$\frac{\Gamma \vdash e : +A \quad \Gamma \vdash f : A \supset B}{\Gamma \vdash e \uparrow f : +B}\ \textbf{TApp} \qquad \frac{\Gamma \vdash f : A \supset B \quad \Gamma \vdash c : \neg B}{\Gamma \vdash c \downarrow f : \neg B}\ \overline{\textbf{TApp}}$$

$$\frac{\vdash e : +A \quad c : \neg A}{\vdash \langle e \mid c \rangle}\ \textbf{TProg1} \qquad \frac{\vdash e : +A \quad \vdash f : A \supset B \quad c : \neg B}{\vdash \langle e \mid f \mid c \rangle}\ \textbf{TProg2}$$

型付け規則も項と継続が対称形を成すように定義されている．ある SLC 式に型がつくということは，それを構成する e, f, c すべてに型がつくということで表現される．

SLC では，環境は上記のように 1 つであり，Γ には項，関数，継続すべての変数が定義されているが，Coq による定式化にあたり，環境は項，関数，継続それぞれについて 1 つずつ用意した．これにより定式化の複雑さが緩和されている．なお，本稿において Γ_x の表記は，環境の中から x の束縛を除いてあることを意味する．また，2.5 節に記した各種の定理・補題の証明を定式化する際，簡約等の操作の前後における環境の中身を比較する必要があったため，変数は EVar（項の変数を表す）等のタグを付けた自然数で表し，昇順になるように定義している．

2.4　reduction rule

small-step semantics による表記．

(*begin*)	$e : +\mathbf{int}$	$\rightsquigarrow$	$\langle e \mid \bullet \rangle$
(*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*)	$\langle e \mid \overline{e'} \mid c \rangle$	$\rightsquigarrow$	$\langle e' \mid (\lceil g \rceil \Rightarrow e \uparrow g) \downarrow c \rangle$
(βR_1)	$\langle e \mid x \Rightarrow e' \mid c \rangle$	$\rightsquigarrow$	$\langle e'[\mathcal{R}[\![x]\!] \mapsto e] \mid c \rangle$
(βR_2)	$\langle e \mid \lceil g \rceil \Rightarrow e' \mid c \rangle$	$\rightsquigarrow$	$\langle e'[\mathcal{R}[\![\lceil g \rceil]\!] \mapsto e] \mid c \rangle$
($\overline{\beta R_1}$)	$\langle e \mid c' \Leftarrow y \mid c \rangle$	$\rightsquigarrow$	$\langle e \mid c'[\mathcal{R}[\![y]\!] \mapsto c] \rangle$
($\overline{\beta R_2}$)	$\langle e \mid c' \Leftarrow \lfloor g \rfloor \mid c \rangle$	$\rightsquigarrow$	$\langle e \mid c'[\mathcal{R}[\![\lfloor g \rfloor]\!] \mapsto c] \rangle$
($\overline{exc}$)	$\langle e \mid \underline{c'} \mid c \rangle$	$\rightsquigarrow$	$\langle e \uparrow (g \downarrow c \Leftarrow \lfloor g \rfloor) \mid c' \rangle$
($\overline{push}$)	$\langle e \mid f \mid c \rangle$	$\rightsquigarrow$	$\langle e \uparrow f \mid c \rangle$
($\overline{pop}$)	$\langle e \mid f \downarrow c \rangle$	$\rightsquigarrow$	$\langle e \mid f \mid c \rangle$
($\overline{end}$)	$\langle n \mid \bullet \rangle$	$\rightsquigarrow$	n

簡約規則は，SLC の計算状態を 2 つ受け取り Prop 型を返す述語として定義した．

2.5　定理・補題

補題 2.1 *(weakening)*（継続については省略）

$$\frac{\Gamma_x \vdash e : +T}{\Gamma_x \cup \{x : +A\} \vdash e : +T} \qquad \frac{\Gamma_g \vdash f : T_1 \supset T_2}{\Gamma_g \cup \{g : A \supset B\} \vdash f : T_1 \supset T_2}$$

証明　e, f, c の型の導出に関する mutual recursion による．

補題 2.2 *(*項に関する代入補題*)*

$\Gamma_x \cup \{x : +A\} \vdash e : +T_0$, $f : T_1 \supset T_2$, $c : \neg T_3$ のとき，$\vdash e' : +A$ ならば $\Gamma_x \vdash e[e'/x] : +T_0, f[e'/x] : T_1 \supset T_2$, $c[e'/x] : \neg T_3$.

証明　e, f, c の型の導出に関する mutual recursion による．

補題 2.1 および 2.2 を用いて，以下の定理が証明できる．

定理 2.1 *(Preservation)*
SLC 式 $\langle\ldots\rangle_1$ について，$\vdash \langle\ldots\rangle_1$ が成り立つとき，$\langle\ldots\rangle_1 \rightsquigarrow \langle\ldots\rangle_2$ ならば，$\vdash \langle\ldots\rangle_2$ である．

証明　簡約規則についての場合分けによる．

定理 2.1 の証明が Coq によって定式化可能であることが示されたことにより，非決定性 SLC が well-typed な言語であることを保証する 1 要素が得られた．

3　call-by-value SLC

前節で定義した SLC に call-by-value 評価戦略を導入する．これによって，ある SLC 式の簡約結果は一意に定まることになる．

はじめに構文について述べる．call-by-value では，関数に渡される引数はすべて値になるまで簡約されてから渡される．よって，構文には値 v が導入され，v, e, f, c の 4 つが相互再帰するデータ型へと変更した．非決定性 SLC では項として扱われていた自然数 n，変数 x，項型の関数 $\lceil f \rceil$ が値として扱われるようになり，さらに context $[v \uparrow (g \downarrow c \Leftarrow \lfloor g \rfloor)]$ が値として新たに定義されている．

次に，簡約規則の変更点であるが，call-by-value では，引数は β 簡約の前に評価されているはずであるから，SLC の簡約規則もそれに従い変更される．例えば，非決定性 SLC の簡約規則のひとつである *(exc)* は，項の部分が値の形をしていなければ適用されなくなる，という具合である．
また，構文に context が追加されたことに伴い，簡約規則にもそれに関する規則が追加されている．

3.1　定理・補題・系

定義 3.1　（*CBV SLC* の停止性論理述語）
call-by-value SLC において，停止性を示す論理述語は以下のようになっている．

1. (a) $R^v_{+\mathbf{int}}(n)$ は自明
   (b) $R^v_{+(A\to B)}(\lceil f \rceil) \iff R^f_{A\supset B}(f)$
   (c) $R^v_{+(A-B)}([v \uparrow (g \downarrow c \Leftarrow \lfloor g \rfloor)])$
   $\iff R^v_{+A}(v)$ かつ $R^c_{\neg B}(c)$
2. $R^e_{+T}(e) \iff \forall R^c_{\neg T}(c).\langle e \mid c\rangle \rightsquigarrow^* n$
3. $R^f_{A\supset B}(f) \iff \forall R^v_{+A}(v).\forall R^c_{\neg B}(c).\langle v \mid f \mid c\rangle \rightsquigarrow^* n$
4. $R^c_{\neg T}(c) \iff \forall R^v_{+T}(v).\langle v \mid c\rangle \rightsquigarrow^* n$

本稿では，定義 3.1 の Coq による定式化が可能であるかを検証している．上田らによってその正当性は十分に検証されているが，定義が複雑なため，定理証明器による定式化が実現されれば，その正当性がより強力に保証されることになる．
定義 3.1 は型に関する論理述語によるため，ある述語を定義する述語は，もとの述語よりも小さな型に関するものでなければならない．しかし，定義 3.1.1〜4 は互いに再帰する形で記されており，例えば 3.1.2 の $+T$ 型の項についての論理述語は，$\neg T$ 型が $+T$ 型よりも小さな型でなければ，停止性を表す述語として適当であると言えない．
一方 Coq では，停止することが保証されない定義は定式化することができないようになっている．つまり，定義 3.1 を Coq によって定式化し，次に述べる主定理の証明を Coq 上で行うことができれば，CBV SLC の停止性を保証することが可能となる．

3.2　CBV SLC の停止性

定理 3.1 *(Termination)*
$\Gamma = x_i : +T_i,\ y_j : \neg T_j,\ g_k : A_k \supset B_k$ であり，$R^v_{+T_i}(v_i)$, $R^c_{\neg T_j}(c)$, $R^f_{A_k\supset B_k}(f_k)$ について，$\rho = [v_i/x_i,\ c_j/y_j,\ f_k/g_k]$ とする．このとき，以下が成り立つ．

1. $\Gamma \vdash v : +T \Rightarrow R^v_{+T}(v\rho)$
2. $\Gamma \vdash e : +T \Rightarrow R^e_{+T}(e\rho)$
3. $\Gamma \vdash f : A \supset B \Rightarrow R^f_{A\supset B}(f\rho)$
4. $\Gamma \vdash c : \neg T \Rightarrow R^c_{\neg T}(c\rho)$

系 3.1 *(Normalization)*
call-by-value SLC において，$\vdash e : +\mathbf{int}$ ならば，$e \rightsquigarrow^* n$ であるような n が一意に存在する．

4　まとめと今後の課題

項と継続を同じように扱うことが可能である対称ラムダ計算の Coq による定式化を試みた．非決定性 SLC については，定理 2.1 および定理 2.2 の証明の定式化を行い，SLC が well-typed な言語であることを保証する条件の 1 つを得た．
また，非決定性 SLC を，評価戦略 call-by-value を導入した call-by-value SLC に書き換え，停止性論理述語の定式化を試みた．

今後の課題としては，CBV SLC の停止性論理述語の定式化手法の検討が第一に挙げられる．
定義 3.1 は，値，項，関数，継続についての論理述語が非常に複雑な相互再帰によって定義されている．単純型付きラムダ計算とは異なり，SLC では常に項と継続の 2 つ組，または項，関数，継続の 3 つ組で計算状態が表されるため，簡約が行われる際のそれらの状態遷移の関係を含めて正しく定式化しなければならない．ただし，定義 3.1 の定式化さえ実現できれば，定理 3.1 の証明を定式化することは可能であると予測される．

参考文献

[1] Filinski, A.: Declarative Continuations and Categorical Duality, in *DIKU report 89/11*, University of Copenhagen (1989).

[2] 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, 第 26(2) 巻 (2009).

[3] Ueda, Y. and Asai, K.: Reinvestigation of Symmetric Lambda Calculus, in *DIKU report 11/01* (2011).