論理関係による shift/reset の部分評価器の正当性の証明

理学専攻　情報科学コース　2240676　横関 茉衣　(指導教員：浅井 健一)

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残余プログラム (residual program) に変換することである。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ときに得られる結果と等しい [3]。残余プログラムは元のプログラムより計算が進んでおり効率が良いため、部分評価は最適化手法のひとつと言える。近年においては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 [4]。

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 [1] がある。Asai は shift/reset [2] で拡張した値呼びの λ 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示した。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。

本研究では、shift/reset の部分評価器の正当性の証明 [1] に見つかった誤りを説明し、そこから定義の誤りを発見し、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。

2　準備

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 λ 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。

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

$\xi k.\, M$ 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、変数は整数 n で表現する。

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。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、$\mathrm{var}(t)$、$\mathrm{lam}(f)$ などを代わりに用いる。

メタ言語の項と項の関係を表す記号については、定義または α 同値を表す際は $=$、β 同値を表す際は $\sim$ を用いる。

3　Asai による shift/reset の部分評価器 [1]

3.1　部分評価器

shift/reset に対する部分評価器は次のように定義される。

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

3.2　shift/reset の部分評価器の正当性の証明

メタ言語の項間の論理関係 [5] 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。なお、$\mathcal{I}[\![M]\!]\,\rho$ は環境 ρ のもとで M がインタプリタの定義によって変換される項を表す。

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

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

$$
\begin{array}{l}
(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha \\
\quad \Longleftrightarrow \forall (V, V') \in R_\tau.\, (\langle (\lambda v.\, K)\, V \rangle, \langle (\lambda v'.\, K')\, V' \rangle) \in R_\alpha
\end{array}
$$

そして、この論理関係を使い以下の定理を作る。ただし、$A, \alpha \vdash M : \tau, \beta\ [W]$ は「型環境 A の下で（上線や下線のない入力の）項 M は τ 型を持ち、answer type を α から β に変化させる。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」ことを意味する。

定理 3.1 $A, \alpha \vdash M : \tau, \beta\ [W]$ かつ $(\rho, \rho') \models A$ かつ $(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$, ならば
$(\langle (\lambda v.\, K)\, (\mathcal{P}\, [\![W]\!]\, \rho)\rangle, \langle (\lambda v'.\, K')\, (\mathcal{I}\, [\![M]\!]\, \rho')\rangle) \in R_\beta$

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。

4 shift/reset の部分評価器の正当性の証明における誤り

Asai による shift/reset の部分評価器の正当性の証明 [1]、つまり定理 3.1 の証明は、型の導出に関する帰納法によるものである。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。

具体的に static な reset における証明の誤りを見る。証明の誤りは

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

を示すためには、「$(\lambda v.\, K, \lambda v'.\, K') \models \tau \rightsquigarrow \alpha$ より $(\langle \mathcal{P}\, [\![W]\!]\, \rho\rangle, \langle \mathcal{I}\, [\![M]\!]\, \rho'\rangle) \in R_\tau$ を示せばよい。」としている部分である。

3.2 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、$\langle \mathcal{P}\, [\![W]\!]\, \rho\rangle$ と $\langle \mathcal{I}\, [\![M]\!]\, \rho'\rangle$ が値ならば証明に誤りはない。

しかし、これらは値ではなく、さらには $\langle \mathcal{I}\, [\![M]\!]\, \rho'\rangle$ 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 $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$ といった発散する項を表す dynamic な項も型がつく。そのため、$\langle \mathcal{I}\, [\![M]\!]\, \rho'\rangle$ の M は発散する項である可能性がある。

5 本稿の shift/reset の部分評価器

本稿が提案する部分評価器は、先行研究 [1] の static な reset と static な shift を以下の定義に変えたものである。

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

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

以下に static な reset の定義を導いた過程を示す。4 節で説明した通り、証明が破綻するのは、reset の中身の項が発散するときである。よって、その場合を考える。$W_{diverge}$、$M_{diverge}$ をそれぞれ $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$ を（下線が引かれている）dynamic な項で表したもの、（上線や下線が引かれていない）部分評価の入力の項で表したものとする。

$$\begin{aligned} W &= \overline{\mathrm{App}}(\overline{\mathrm{Lam}}(n, \overline{\mathrm{Lam}}(m, \mathrm{Var}(m))), \overline{\mathrm{Reset}}(W_{diverge})) \\ M &= \mathrm{App}(\mathrm{Lam}(n, \mathrm{Lam}(m, \mathrm{Var}(m))), \mathrm{Reset}(M_{diverge})) \\ &\quad \lambda\text{式の書式：}\ (\lambda x.\, \lambda y.\, y)\, \langle M_{diverge}\rangle \end{aligned}$$

のとき、$W_{diverge}$ は停止しない項であるが、W を部分評価すると $\mathrm{lam}(\lambda y.\, \mathrm{var}(y))$ となる。一方で M をインタープリタに渡して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。

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。

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

こうすると、継続 k に渡されるものは値 $\mathrm{var}(t)$ となる。これにより、変更後の部分評価器の正当性の証明において、

$$(\lambda z_n.\, \mathcal{I}\, [\![\downarrow_{n+1} \langle (\lambda v.\, K)\, (\mathrm{var}(n))\rangle]\!]\, \rho_{id})\, (\mathcal{I}\, [\![\downarrow_n \langle \mathcal{P}\, [\![W]\!]\, \rho\rangle]\!]\, \rho_{id})$$

が

$$(\lambda z_n.\, \langle (\lambda v'.\, K')\, z_n\rangle)\, (\mathcal{I}\, [\![\downarrow_n \langle \mathcal{P}\, [\![W]\!]\, \rho\rangle]\!]\, \rho_{id})$$

と β 同値であることが、$(\lambda v.\, K, \lambda v'.\, K') \models d \rightsquigarrow d$ かつ $(\mathrm{var}(n), z_n) \in R_d$ から言える。これは 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ためである。

6 まとめと今後の課題

本研究では先行研究 [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。現在、static な reset の項 $\overline{\mathrm{Reset}}(W)$ に対して、$\langle \mathcal{P}\, [\![W]\!]\, \rho\rangle$ の型が dynamic な型となるように制限してしまっている。この制限をなくすために、 CPS を 2 度行った 2CPS の体系を検討中である。

参考文献

[1] K. Asai. Logical relations for call-by-value delimited continuations. In *A chapter of Trends in Functional Programming*, volume 6, pages 63–78. Intellect, 2007.

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

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

[4] 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.

[5] M. Wand. Specifying the correctness of binding-time analysis. *Journal of Functional Programming*, 3:365–387, 1993.