エフェクトハンドラを用いた型安全なコード生成

礒田 華成　亀山 幸義

代数的エフェクトおよびエフェクトハンドラは，様々な計算効果を統一的かつモジュラーに扱うための言語機能であり，コード生成の文脈においても，let-insertion などを表現することができ，有用であると考えられる．先行研究において，可変セルなどのある種の言語機能は，コード生成と組み合わせたとき，生成されるコード中の変数がスコープ逸脱を起こす場合があることが知られており，エフェクトハンドラにおいても同様の問題が発生し得る．本研究では，エフェクトハンドラを利用したコード生成のための 2 段階の計算体系と型システムを提案する．Kiselyov らが提案した，生成されるコードに含まれる自由変数の集合を追跡するための手法である refined environment classifier に基づき，ハンドラを未来のステージの変数の束縛子として取り扱う．型システムの健全性に加え，生成されたコードがスコープ逸脱を起こすことがないことを示した．

1　研究の背景と目的

代数的エフェクトとエフェクトハンドラ [4] は，プログラムにおける様々な計算効果を統一的かつモジュラーに扱うための言語機能であり，コード生成においても有用な利用例が考えられる．図 1 に示すのは，生成されるコードの重複を防ぐための let-insertion とよばれる手法を表現する例である．

$$H_{\mathsf{id}} \coloneqq \{\mathbf{return}\ x \mapsto \mathbf{return}\ x\}$$
$$H_{\mathsf{ins}} \coloneqq \{\mathsf{ins}\ x\ k \mapsto \underline{\mathbf{let}}\ y \leftarrow \mathbf{return}\ x\ \underline{\mathbf{in}}\ \mathbf{throw}\ k\ y\}$$

$$\begin{aligned}&\mathbf{with}\ H_{\mathsf{id}} \uplus H_{\mathsf{ins}}\ \mathbf{handle}\\ &\quad \mathbf{let}\ x \leftarrow \mathbf{do}\ \mathsf{ins}\ \langle M\rangle\ \mathbf{in}\ x \mathbin{\underline{+}} x\\ \rightsquigarrow^*\ &\langle \mathbf{let}\ y \leftarrow M\ \mathbf{in}\ y + y\rangle\end{aligned}$$

図 1　エフェクトハンドラによる let-insertion の例

let のような下線つきの構文は，評価されると対応するコードを生成する演算子である．生成された式 M のコードは $\langle M\rangle$ と書く．この例では，生成されたコードに式 M が重複して出現することを防ぐために，ins というオペレーションによって M を一度変数 y に束縛するというコードを生成している．

他方で，可変セル [2] や限定継続演算子 [3] のように，式の位置を自由に移動させることのできる言語機能は，コード生成と組み合わせたとき，変数のスコープ逸脱を引き起こすことが知られており，図 2 に示すように，エフェクトハンドラにおいても同様の問題が発生し得る．

$$\begin{aligned}&\underline{\lambda}x.\,\mathbf{with}\ H_{\mathsf{id}} \uplus H_{\mathsf{ins}}\ \mathbf{handle}\\ &\quad \mathbf{let}\ z \leftarrow \mathbf{do}\ \mathsf{ins}\ \langle \mathbf{return}\ x\rangle\ \mathbf{in}\ \mathbf{return}\ z\\ \rightsquigarrow^*\ &\langle \mathbf{let}\ y \leftarrow \mathbf{return}\ x'\ \mathbf{in}\ \mathbf{return}\ \lambda x'.\,\mathbf{return}\ y\rangle\end{aligned}$$

図 2　エフェクトハンドラによるスコープ逸脱の例

オペレーション ins の呼び出しの引数 $\langle \mathbf{return}\ x\rangle$ は，with-handle 式の外側で束縛された変数 x を含んでいる．オペレーション呼び出しによって with-handle 式の直前の位置に let が挿入されるため，生成された

---

* Type-Safe Code Generation with Effect Handlers. This is an unrefereed paper. Copyrights belong to the Authors.

Kanaru Isoda, Yukiyoshi Kameyama, 筑波大学 情報理工学位プログラム, Department of Computer Science, University of Tsukuba.

コードでスコープ逸脱が起きている．

本研究では，エフェクトハンドラを用いて型安全なコード生成を実現できるようにすることを目的とし，エフェクトハンドラを含む型安全な多段階計算の体系を提案する．

本稿は，GPCE 2024 で発表予定の内容 [1] を日本語でまとめたものである．

2 先行研究

Taha と Nielsen は，environment classifiers とよばれる手法に基づき，生成されたコードを実行する演算子である run を含む型安全な多段階計算体系 λ_α を提案した [5]．

Kiselyov らによる，可変セルを含む型安全な 2 段階の計算体系 <NJ> [2] では，この手法を元にした refined environment classifiers (REC) とよばれる新たな手法を提案し，生成されるコードに含まれ得る自由変数の集合を型システムで追跡することで，スコープ逸脱が起きないことを静的に保証している．<NJ> では，型 τ のコードを表す型は $\langle\tau\rangle^\gamma$ と書かれ，式に含まれ得る未来のステージの変数の集合を表す classifier γ の注釈がついている．図 3 に示すのは，ラムダ抽象を生成する式 $\underline{\lambda}x.\,M$ の型付け規則である (構文は提案体系と合わせているため一部異なっている部分がある)．

$$\frac{\gamma \in \Delta \uplus \Gamma \qquad \gamma_1 \notin \Delta \uplus \Gamma \qquad \Delta; \Gamma, \gamma_1, (\gamma_1 \succeq \gamma), (x : \langle A^1 \rangle^{\gamma_1}) \vdash M : \langle B^1 \rangle^{\gamma_1}}{\Delta; \Gamma \vdash \underline{\lambda}x.\,M : \langle A^1 \to B^1 \rangle^{\gamma}}$$

図 3　<NJ> における式 $\underline{\lambda}x.\,M$ の型付け規則

この式は，評価されると未来のステージの新しい変数を導入するため，型付けの際に，型環境に新しい classifier γ_1 と，γ_1 が現在の classifier γ より深いスコープで導入されたことを表す，classifier 同士の部分型関係 $\gamma_1 \succeq \gamma$ を追加する．この仕組みによって，スコープ逸脱を起こし得るコードの型付けを試みると，classifier の不整合によってそれが検出される．

大石と亀山は，<NJ> に限定継続演算子 shift0/reset0 を追加し，REC に，classifier 同士の join をとることのできる構造を導入した型システムを提案した [3]．

3 提案する計算体系と型システム

本研究で提案する，エフェクトハンドラを含む 2 段階の計算体系の型システムの概観を述べる．計算体系に関しては，およそ <NJ> にエフェクトハンドラを加えた体系になっており，型システムに関しては，先述した大石と亀山の先行研究 [3] に基づいている．ただし，エフェクトハンドラの取り扱いに関しては特記するべき点があるため，以下で議論する．

3.1 エフェクトハンドラの取り扱い

オペレーションの呼び出し，および対応するハンドラの実行によって，図 4 に示すように with-handle 式の外側に未来のステージの変数の新しい束縛が生じる場合がある．

$$\begin{aligned} &\textsf{with}\ H_{\mathsf{id}} \uplus H_{\mathsf{ins}}\ \textsf{handle do}\ \mathsf{ins}\ \langle 1 \rangle \\ \rightsquigarrow\ &\underline{\textsf{let}}\ y \leftarrow \textsf{return}\ \langle 1 \rangle\ \underline{\textsf{in}}\ \textsf{throw}\ k\ y \\ &\text{where } k = \kappa z.\,\textsf{with}\ H_{\mathsf{id}} \uplus H_{\mathsf{ins}}\ \textsf{handle return}\ z \end{aligned}$$

図 4　with-handle 式が束縛子として振る舞う例

そのため，型システムにおいて with-handle 式は未来のステージの変数の束縛子と同様の扱いを受ける．図 5 に示すのは with-handle 式の型付け規則である．

$$\frac{\gamma \in \Delta \uplus \Gamma \qquad \gamma_1 \notin \Delta \uplus \Gamma \qquad \Delta; \Gamma \vdash H : \forall \upsilon.\, \langle A^1 \rangle^{\upsilon}\, !\, E \rightrightarrows \langle B^1 \rangle^{\upsilon}\, !\, E' \qquad \Delta; \Gamma, \gamma_1, (\gamma_1 \succeq \gamma) \vdash M : \langle A^1 \rangle^{\gamma_1}\, !\, E[\gamma_1/\upsilon]}{\Delta; \Gamma \vdash \textsf{with}\ H\ \textsf{handle}\ M : \langle B^1 \rangle^{\gamma}\, !\, E'[\gamma/\upsilon]}$$

図 5　with-handle 式の型付け規則

図 3 で見た，未来のステージの変数を新しく導入する式である $\underline{\lambda}x.\,M$ と同様に，with-handle 式の型付けの際に，新しい classifier γ_1 を導入し型環境に追加している．

ハンドラの型は，図 6 に示すように classifier に関

する多相性をもち，図 5 で示した with-handle 式の型付けの際に具体的な classifier によってインスタンス化される．これは，ハンドラが実際に利用されるときまでハンドラ本体に含まれる式がもつべき classifier を決定することができないことに起因している．

$$
\begin{aligned}
H &= \{\textsf{return}\ x \mapsto M\} \uplus \{op_i\ x\ k \mapsto N_i\}_i \\
E &= \{op_i : A_i^{\mathsf{op}} \twoheadrightarrow \langle B_i^{\mathsf{op}} \rangle^{\upsilon}\}_i \uplus E', \\
\Gamma_i' &= (x : A_i^{\mathsf{op}}), (k : \langle B_i^{\mathsf{op}} \rangle^{\upsilon} \rightarrowtail \langle B^1 \rangle^{\upsilon} ! E')
\end{aligned}
$$

$$
\frac{\begin{array}{c}\upsilon \notin \Delta \uplus \Gamma \\ \Delta; \Gamma, \upsilon, (x : \langle A^1 \rangle^{\upsilon}) \vdash M : \langle B^1 \rangle^{\upsilon} ! E' \\ \left[\Delta; \Gamma, \upsilon, \Gamma_i' \vdash N_i : \langle B^1 \rangle^{\upsilon} ! E'\right]_i\end{array}}{\Delta; \Gamma \vdash H : \forall \upsilon. \langle A^1 \rangle^{\upsilon} ! E \Rightarrow \langle B^1 \rangle^{\upsilon} ! E'}
$$

図 6　ハンドラの型付け規則

3.2　型システムの健全性

提案する型システムが健全であることを，以下の 2 つの定理を証明することによって示した．ただし，$\overline{\Gamma}$ は classifier とそれらの部分型関係のみを含む型環境を表す．

定理 1 (進行)．$\Delta; \overline{\Gamma} \vdash M : C$ ならば，M は正規形または $M \rightsquigarrow N$.

定理 2 (主部簡約)．$\Delta; \overline{\Gamma} \vdash M : C$ かつ $M \rightsquigarrow N$ ならば，$\Delta, \Delta'; \overline{\Gamma} \vdash N : C$.

また，重要な帰結として，生成されたコードがスコープ逸脱を起こさないことを表す以下の系を示した．

系 3. $\gamma; \vdash M : \langle A^1 \rangle^{\gamma}$ かつ $M \rightsquigarrow^* \textsf{return}\ V$ ならば，$V = \langle N^1 \rangle$ かつ $\gamma; \vdash^{\gamma} N^1 : A^1$.

4　まとめと今後の展望

本研究では，エフェクトハンドラを用いた型安全なコード生成の実現を目的として，エフェクトハンドラを含む型安全な 2 段階の計算体系を提案し，型システムの健全性を証明した．今後の展望を以下に述べる．

- 体系や型システムの拡張・変更．生成されたコードを実行する演算子 run を追加したり，コード型以外をもつハンドラを書けるような拡張が可能であるかどうか，また，classifier に関する多相性をどの程度導入することができるか検討する．
- 型推論アルゴリズム．Classifier を明示しなければならない体系は，実用的なプログラミング言語としてみると煩雑である．これを避けるために型推論がどの程度可能であるか検討する．

参 考 文 献

[1] Isoda, K., Yokoyama, A., and Kameyama, Y.: Type-Safe Code Generation with Algebraic Effects and Handlers, *Proceedings of the 23rd ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, GPCE 2024, Pasadena, California, United States, October 21-22, 2024*, to appear.

[2] Kiselyov, O., Kameyama, Y., and Sudo, Y.: Refined Environment Classifiers - Type- and Scope-Safe Code Generation with Mutable Cells, *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings*, Igarashi, A.(ed.), Lecture Notes in Computer Science, Vol. 10017, 2016, pp. 271–291.

[3] Oishi, J. and Kameyama, Y.: Staging with control: type-safe multi-stage programming with control operators, *Proceedings of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, GPCE 2017, Vancouver, BC, Canada, October 23-24, 2017*, Flatt, M. and Erdweg, S.(eds.), ACM, 2017, pp. 29–40.

[4] Plotkin, G. D. and Pretnar, M.: Handlers of Algebraic Effects, *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, Castagna, G.(ed.), Lecture Notes in Computer Science, Vol. 5502, Springer, 2009, pp. 80–94.

[5] Taha, W. and Nielsen, M. F.: Environment classifiers, *Conference Record of POPL 2003: The 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New Orleans, Louisisana, USA, January 15-17, 2003*, Aiken, A. and Morrisett, G.(eds.), ACM, 2003, pp. 26–37.