多相関数を含むプログラムの抽象解釈を用いた最適化

美馬 久行　上野 雄大　大堀 淳

関数型プログラミング言語において，多相関数はその記述性に大きく貢献している．しかし，現在のコンピュータアーキテクチャでは多相関数の存在を想定していないため，多相関数の実装には本質的なコストがかかる．そこで，本研究の最終目標として，ユーザが定義した多相関数には単一の型が型適用されている場合が多いという洞察に基づき，単一の型のみが型適用されている多相関数を単相関数に変換することにより，プログラムの最適化を行うことを目指す．本研究では，目標への最初の段階として，多相関数がどのように型適用されるか，という情報を収集する抽象解釈を用いた静的解析を定義し，その性質を証明した．

1　はじめに

多くの関数型プログラミング言語では多相関数を定義することができる．しかし，多相関数をもつ関数型プログラミング言語を現在のコンピュータアーキテクチャ上で動作する機械語にコンパイルするのは容易ではない．型システムや操作的意味論の段階では型情報が保持されているため，多相型を持つ値に型適用を行う際には，単純に型を適用することができる．機械語にコンパイルされる段階では，型情報が破棄される．したがって，多相関数を機械語にコンパイルする際，引数のサイズがわからない，ポインターかどうかが分からないので GC が適切に処理できない，などの問題が発生する．この問題に対して，機械語にコンパイルする際に型情報も機械語に落とし込み，多相関数を呼ぶ際にその型情報を適用する方法，型情報自体ではなく機械語の実行に必要な情報のみを機械語に落とし込む方法，などの解決方法がある．どの方法であれ，現在のコンピュータアーキテクチャ上では機械語に型情報が付加されていないので，多相関数を持つ関数型プログラミング言語をコンパイルする際には，型に関する何らかの情報を機械語に埋め込む必要がある．したがって，多相関数を呼び指す際には，機械語に埋め込まれた型に関する何らかの情報を渡すという，単相関数にはない本質的なコストがかかる．

OCaml や SML などの関数型プログラミング言語では，その強力な型システムによって，プログラマがほとんど型注釈を記述することなくプログラムを記述することができる．しかし，型注釈を記述しないことによって，本来は単相関数であるべき関数が多相関数として解釈されることがある．そのような多相関数は，単一の型のみが型適用されていると考えられる．そこで，本研究の最終目標として，単一の型のみが型適用されている多相関数を単相関数に変換することにより，本来不要であった型抽象を取り除き，プログラムの最適化を行うことを目指す．本研究では，目標への最初の段階として，let 多相を持つ ML 型システムの項に対して，多相関数がどのように型適用されるか，という情報を収集しつつ項にその情報を付加する，抽象解釈を用いたアルゴリズムを定義した．さらに，変換された型適用情報が付加された項に対して，型適用情報を型適用制約と捉え，型適用制約に違反しないこと保証する型システムを定義し，変換が正しい

For the Optimization of Program with Polymorphism
Hisayuki Mima, 東北大学情報科学研究科, Graduate School of Information Sciences, Tohoku University.
Katsuhiro Ueno, Atsushi Ohori, 東北大学電気通信研究所, Research Institute of Electrical Communication, Tohoku University.

ことを証明した．

ML 型システムは Rank0 多相性を持つ型システムである．Rank1 またはそれ以上の多相性を持つ型システムでは，値が多相型を持つことができるので，値に対して型適用することができる．したがって，そのような型システムを持つ言語に対して，型システムを用いた解析によって型適用情報を収集するのは容易ではない．抽象解釈を用いて値を抽象的に扱うことで，Rank1 多相性を持つ型システムに対しても型適用情報を収集することできると考えられる．そこで，Rank1 多相性を持つ型システムへの拡張を考慮し，本論文でも抽象解釈を用いてアルゴリズムを記述した．

本論文は以下のような構成である．2 節では，変換対象となる言語 ML とその型システムを定義する．3 節では，言語 ML に対して型適用制約を付加した言語 ML^{A} を定義し，その言語に対する型適用制約に違反しないことも保証する型システムを定義する．4 節では，言語 ML の項から言語 ML^{A} の項に変換するアルゴリズムを定義し，変換後の項が型適用制約に違反しないことを証明する．5 節では，まとめと今後の課題について述べる．

2 対象言語の定義

変換の対象として，以下の構文を持つ明示的型付き項から成る言語 ML を考える．

$$\begin{aligned}
\tau &::= \iota \mid t \mid \tau \to \tau \mid \tau \times \tau \\
\sigma &::= \tau \mid \forall(t,\ldots,t).\tau \\
M &::= c^{\iota} \mid x \mid \lambda x:\tau.M \mid M\ M \\
&\quad \mid (M,M) \mid \#1\ M \mid \#2\ M \\
&\quad \mid \text{let } x:\forall(t,\ldots,t).\tau = M \text{ in } M \mid (x\ (\tau,\ldots,\tau))^{l}
\end{aligned}$$

通常の明示的型付き ML 項と異なり，型適用を追跡するため，型適用の項にラベル l が付加されている．

言語 ML に対して，$\mathrm{ML} \vdash K, \Gamma \triangleright M : \tau$ の形で ML 型システムを定義する．K は型変数の有限集合であり，型制約付き言語との対応のために導入した．

3 型適用制約付き言語 ML^{A}

2 節で定義した言語 ML に型適用制約を付加した言語 ML^{A} を考える．

$$\begin{aligned}
M^{\mathrm{A}} &::= c^{\iota} \mid x \mid \lambda x:\tau.M^{\mathrm{A}} \mid M^{\mathrm{A}}\ M^{\mathrm{A}} \\
&\quad \mid (M^{\mathrm{A}},M^{\mathrm{A}}) \mid \#1\ M^{\mathrm{A}} \mid \#2\ M^{\mathrm{A}} \\
&\quad \mid \text{let } x:\forall(t::P,\ldots,t::P).\tau = M^{\mathrm{A}} \text{ in } M^{\mathrm{A}} \\
&\quad \mid (x\ (\tau,\ldots,\tau))^{l}
\end{aligned}$$

言語 ML^{A} に対して，$\mathrm{ML}^{\mathrm{A}} \vdash K^{\mathrm{A}}, \Gamma^{\mathrm{A}} \triangleright M^{\mathrm{A}} : \tau$ の形で，型適用制約を満たすことも保証する ML^{A} 型システムを定義する．

4 変換アルゴリズム

2 節で定義した言語 ML の項を 3 節で定義した言語 ML^{A} の項に変換するアルゴリズムを $K^{\mathrm{A}}, \Gamma^{\mathrm{A}}, M \rightsquigarrow \Gamma^{\mathrm{A}\prime}, M^{\mathrm{A}}$ の形で定義する．

この変換アルゴリズムに対して，変換の妥当性，つまり変換後の項が型適用制約に違反しないことを示す定理が成り立つ．

定理 1 (変換の妥当性)．$\mathrm{ML} \vdash K, \Gamma \triangleright M : \tau$ かつ $\vdash \Gamma^{\mathrm{A}} : \Gamma$ かつ $dom(K^{\mathrm{A}}) = K$ かつ $K^{\mathrm{A}}, \Gamma^{\mathrm{A}}, M \rightsquigarrow \Gamma^{\mathrm{A}\prime}, M^{\mathrm{A}}$ ならば，$\mathrm{ML}^{\mathrm{A}} \vdash K^{\mathrm{A}}, \Gamma^{\mathrm{A}\prime} \triangleright M^{\mathrm{A}} : \tau$ が成り立つ．

5 まとめと今後の課題

言語 ML の項 M に対して，M の中の型適用の情報を収集し，その情報を付加した言語 ML^{A} の項 M^{A} を得る変換アルゴリズムを定義し，そのアルゴリズムの妥当性を証明することができた．

本論文で用いた ML 型システムの多相性はランク 0 に分類される．多くの実用的な関数型プログラミング言語ではランク 1 かまたはそれ以上のランクの多相性を持っている．そこで，本論文で定義した変換アルゴリズムをランク 1 多相性を持つ型システムに拡張することが今後の課題としてあげられる．

謝辞 本研究の一部は JSPS 科研費 25280019「ML 系多相型言語 SML#の実用化技術に関する基礎研究」および 15K15964「実用プログラミング言語のための系統的言語開発基盤の実現」の助成を受けて実施されたものである．