

多相型付きCPS計算および 暗黙的多相付き入計算からの変換

中村光希^{*1}

関山太郎^{*2}

五十嵐淳^{*1}

^{*1}：京都大学 ^{*2}：国立情報学研究所

継続渡し形式 (CPS)

コンパイラの間中表現 (IR) として望ましい性質を持つ形式：

- 関数呼び出しをジャンプとして扱える
- 自然にコントロールオペレータを扱える

関数呼び出しが全て末尾呼び出しのため

CPS計算 [Thielecke, 1997]

- CPSの項のみを記述できる計算体系
- IRの議論に適した計算体系 [Torrens et al, 2024]
- CPS計算上での簡約が IR 上での最適化とみなせる

動機：型付きIRが欲しい

例えば OCaml が該当

コンパイラの生成する IR の正当性検証などで有効

①暗黙的多相があり②値制限がなく③値呼びである言語に対する型付きIRを理論建てしたい。

課題：現在CPS計算で議論されている型システムは単純型システムのみ → 多相の追加を試みる

予備知識：暗黙的多相付き入計算からの型保存CPS変換

[Sekiyama and Tsukada, 2021]

①暗黙的多相があり②値制限がなく③値呼びである言語からのCPS変換には、ターゲット言語に特殊な機能が必要。

機能A. 型変数の束縛 ($\nu\alpha$)

機能B. 型変数を用いた型の一般化 ($\Lambda^\circ\langle\alpha, M\rangle$)

- ただし同じ型変数は一度しか型の一般化に使えない
- また、 Λ° の内側に現れる M は簡約して良い

この制約抜きでは、型安全性が壊れるため

型抽象の内側を簡約できる体系とみなせる
暗黙的多相の挙動を再現するため

本研究：CPS計算+多相型 (暗黙的多相)

型

κ は線形型の use で、
値の使用可能回数を示す
0...使えない, 1...1回, ω ...任意回

$A ::= \neg^\kappa \vec{A} \mid \forall\alpha.A \mid \alpha$

関数型 $\vec{A} \rightarrow \perp$ のこと

構文

κ は線形型の use

$M ::= k(\vec{x}) \mid \text{let } x^\kappa = R \text{ in } M$

$R ::= (\vec{x}).M \mid \Lambda^\circ$

関数

型の一般化をするオペレーター

拡張の概要

- 型に多相型を追加
- 暗黙的多相付き入計算からのCPS変換に必要な機能を実現する特殊オペレーター Λ° を追加

型付け規則

機能Aに対応
型変数 α を
型環境へ導入

機能Bに対応
環境に導入された型変数 α を使って
変数の型を一般化できる関数 x
(※この関数は一度だけ利用可能)

受け取った変数の型を
一般化して継続に渡す

$\Gamma, \alpha, x : \neg^1(A, \neg^1(\forall\alpha.A)) \vdash M$

$\Gamma \vdash \text{let } x^1 = \Lambda^\circ \text{ in } M$

簡約規則

受け取った変数を

同じく受け取った継続に
そのまま渡す

$\text{let } \dots \text{ in let } x^1 = \Lambda^\circ \text{ in let } \dots \text{ in } x(f, k)$
 $\longrightarrow \text{let } \dots \text{ in let } x^0 = \Lambda^\circ \text{ in let } \dots \text{ in } k(f)$

x はこれ以上使えなくなる

※CPS変換については、[Sekiyama and Tsukada, 2021] を移植

取組み中の証明

□ 型システムの健全性

項に型がつくなら、その項は実行時エラーを起こさない

今後の応用・展望

- コンパイラの正当性検証
- 型保存コンパイルの総括的な理論建て
- 多段階計算機能を持つ言語からのCPS変換