研究室ゼミ

A Natural Semantics for Lazy Evaluation

水野雅之

2016 年 11 月 30 日

これまでの研究

入出力を含む純粋でない関数型言語の
K 正規化を検証

⇓

CakeML(ICFP2016) **や** Pilsner(ICFP2015)
→call by value **な関数型言語処理系の検証は**
red ocean **っぽい**

⇓

call by need(**以後** cbn) **な関数型言語へ**

今回紹介する論文

- A Natural Semantics for Lazy Evaluation
- A locally nameless representation for natural semantics for lazy evaluation
  - ↑↑**に** locally nameless representation **を導入したもの**

アウトライン

1. A Natural Semantics for Lazy Evaluation
2. A locally nameless representation for natural semantics for lazy evaluation

A Natural Semantics for Lazy Evaluation

- cbn な λ 計算の拡張の大ステップ操作的意味論を定義する論文
  - λ 計算に相互再帰的な let を入れた言語が対象
  - 変数名を thunk の location として流用
    - 時々束縛変数の renaming が必要

論文のモチベーション

- 実際の処理系の実装を模した意味論を定義したい
  - 値の sharing
  - 一度評価された関数引数は再び評価されない
- 先行研究は too concrete
  - 継続、コードポインタ、スタックポインタ、間接参照を含むノード等
  - 証明に使うには大変

対象言語

- λ 計算に相互再帰な let を導入した言語
- 評価に先立ち構文を制限

$$
\begin{array}{rcl}
e & ::= & \lambda x.\ e \\
& | & e\ x \\
& | & x \\
& | & \mathbf{let}\ x_1 = e_1, \cdots, x_n = e_n\ \mathbf{in}\ e
\end{array}
$$

評価規則

$$\boxed{\Gamma : e \Downarrow \Delta : z}$$

$$\frac{}{\Gamma : \lambda x.e \Downarrow \Gamma : \lambda x.e}$$

$$\frac{\Gamma : e \Downarrow \Delta : \lambda y.e' \qquad \Delta : e'[x/y] \Downarrow \Theta : z}{\Gamma : e\ x \Downarrow \Theta : z}$$

$$\frac{\Gamma : e \Downarrow \Delta : z}{(\Gamma, x \mapsto e) : x \Downarrow (\Delta, x \mapsto z) : z}$$

$$\frac{(\Gamma, x_1 \mapsto e_1 \cdots x_n \mapsto e_n) : e \Downarrow \Delta : z}{\Gamma : \mathbf{let}\ x_1 = e_1, \cdots, x_n = e_n\ \mathbf{in}\ x \Downarrow \Delta : z}$$

評価の例

必要なら白板で…

- 評価関係は帰納的に定義するため、無限回の規則適用は許さない
- 一見無限回評価規則を適用できそうな式の評価でも、有限回の適用で終了する場合がある
- 評価の途中で renaming が必要になる場合がある

表示的意味論との対応

Coq での検証に使えるか微妙なので、あまり詳しく追ってません

- 最小不動点を使う、ありがちな表示的意味論
- Correctness はいわゆる健全性
- Computational adequacy はいわゆる健全性+完全性

意味論の拡張

- コンストラクタの追加
  - コンストラクタの中身は変数に制限
  - コンストラクタも値とする
- GC の実装
  - 環境がヒープに相当
  - 全体の式から到達できない変数の束縛を削除
- 評価のコストのカウント
  - 計算に要したステップ数の評価ができる

アウトライン

1. A Natural Semantics for Lazy Evaluation
2. A locally nameless representation for natural semantics for lazy evaluation

A locally nameless representation for natural semantics for lazy evaluation

- **先に解説した論文に** locally nameless representation **を導入する論文**
- Correctness **や** computational adequacy **は証明していない**
- **(この時点では)**Coq **を使っていない**
  - **後に** Coq **で書いた論文も出ている模様**

論文のモチベーション

- 先に挙げた論文の意味論は変数名に深く依存している
  - 評価前に renaming が必要
  - 評価中に必要になる場合も
- 名前で束縛を表現するため定理証明支援系での扱いが煩雑

これらを解決したい

Locally nameless representation

- **自由変数を名前で、束縛変数を** de Bruijn index **で表現する手法**
  - **シフト “は” 不要になる**

$$
\begin{array}{rcl}
v & ::= & \mathbf{bvar}\ i\ j \\
 & | & \mathbf{fvar}\ x \\
t & ::= & v \\
 & | & \mathbf{abs}\ t \\
 & | & \mathbf{app}\ t\ v \\
 & | & \mathbf{let}\ \{t_i\}_{i=1}^{n}\ \mathbf{in}\ t
\end{array}
$$

Variable opening

束縛の中身を処理する場合、束縛されていた変数が自由変数になる
$\rightarrow$ インデックスで表現された変数を名前による表現に直す

$$
\begin{aligned}
\{k \to \overline{x}\}(\mathbf{bvar}\ i\ j) &= \begin{cases} \mathbf{fvar}\ (\mathbf{List.nth}\ j\ \overline{x} & \texttt{if } i = k \wedge j < |\overline{x}| \\ \mathbf{bvar}\ i\ j & \texttt{otherwise} \end{cases} \\
\{k \to \overline{x}\}(\mathbf{fvar}\ x) &= \mathbf{fvar}\ x \\
\{k \to \overline{x}\}(\mathbf{abs}\ t) &= \mathbf{abs}\ (\{k+1 \to \overline{x}\}\ t) \\
\{k \to \overline{x}\}(\mathbf{app}\ t\ v) &= \mathbf{app}\ (\{k \to \overline{x}\}\ t)\ (\{k \to \overline{x}\}\ v) \\
\{k \to \overline{x}\}(\mathbf{let}\ \overline{t}\ \mathbf{in}\ t) &= \mathbf{let}\ (\{k+1 \to \overline{x}\}\overline{t}\ \mathbf{in}\ (\{k+1 \to \overline{x}\}\ t)
\end{aligned}
$$

Variable closing

変数を束縛する際、束縛された変数をインデックスでの表現に直す
→Opening の逆の操作

$$
\begin{aligned}
\{k \leftarrow \overline{x}\}(\mathbf{bvar}\ i\ j) &= \mathbf{bvar}\ i\ j \\
\{k \leftarrow \overline{x}\}(\mathbf{fvar}\ x) &= \begin{cases} \mathbf{bvar}\ k\ j & \texttt{if } \exists j : 0 \leq j < |\overline{x}|.x = \mathbf{List.nth}\ j\ \overline{x} \\ \mathbf{fvar}\ x & \texttt{otherwise} \end{cases} \\
\{k \leftarrow \overline{x}\}(\mathbf{abs}\ t) &= \mathbf{abs}\ (\{k+1 \leftarrow \overline{x}\}\ t) \\
\{k \leftarrow \overline{x}\}(\mathbf{app}\ t\ v) &= \mathbf{app}\ (\{k \leftarrow \overline{x}\}\ t)\ (\{k \leftarrow \overline{x}\}\ v) \\
\{k \leftarrow \overline{x}\}(\mathbf{let}\ \overline{t}\ \mathbf{in}\ t) &= \mathbf{let}\ (\{k+1 \leftarrow \overline{x}\}\overline{t}\ \mathbf{in}\ (\{k+1 \leftarrow \overline{x}\}\ t)
\end{aligned}
$$

Local closure

- **自由変数は名前で、束縛変数はインデックスで正しく表現されているかを表す述語**