研究室ゼミ

A Survey of Inlining for Call-by-Need Purely Functional Language

Glasgow Haskell Compiler's Case

水野 雅之

東北大学 大学院情報科学研究科
住井・松田研究室

2016 年 7 月 14 日

Outline

Outline

1. Motivation
2. Call-by-need

Current My Research

Formal verification of MinCaml (esp. K-normalization)

MinCaml : impure higher-order functional language

- Call-by-value
- Pair
- Array
- External function call (possibly cause I/O)

Good case study for verification of functional programs performing infinite I/O

Previous Works

Verification of call-by-value languages contain I/O

- Cake ML (ICFP 2016)
- Pilsner (ICFP 2015)

It was red ocean...

$\Rightarrow$Switch to call-by-need (purely) functional languages

Outline

アウトライン

What's Call-By-Need?

Improvement of call-by-name

	normalize	duplicate redex
call-by-value	×	×
call-by-name	◯	◯
call-by-need	◯	×

Call-By-Name

Yields normal form if exists

$$(\lambda xy.\ y)\ ((\lambda x.\ xx)\ (\lambda x.\ xx))$$
$$\longrightarrow_* \lambda y.\ y$$

Duplicate redex

$$(\lambda x.\ xx)\ ((\lambda x.\ x)\ (\lambda x.\ x))$$
$$\longrightarrow (\lambda x.\ x)\ (\lambda x.\ x)\ ((\lambda x.\ x)\ (\lambda x.\ x))$$

Call-By-Need

Yields normal form if exists

$$(\lambda xy.\ y)\ ((\lambda x.\ xx)\ (\lambda x.\ xx))$$
$$\longrightarrow_* \lambda y.\ y$$

Don't duplicate redex

$$(\lambda x.\ xx)\ ((\lambda x.\ x)\ (\lambda x.\ x))$$
$$\longrightarrow \mathbf{let}\ x = (\lambda x.\ x)\ (\lambda x.\ x)\ \mathbf{in}\ x\ x$$
$$\longrightarrow_* \mathbf{let}\ x = \lambda x.\ x\ \mathbf{in}\ x\ x$$

Memoize argument (thunk)

アウトライン

Call-By-Name Lambda Calculus

Syntax

$$
\begin{array}{llcl}
\texttt{Values} & V & ::= & \lambda x.\ M \\
\texttt{Term} & M, N & ::= & x \mid \lambda x.\ M \mid M\,N \\
\texttt{Contexts} & E & ::= & [] \mid E M
\end{array}
$$

Evaluation rule

$$(\lambda x.\ M)N \longrightarrow [x \mapsto N]M$$

Introducing Let

Syntax

$$\begin{array}{lll}
\texttt{Values} & V & ::= \lambda x.\ M \\
\texttt{Answers} & A & ::= V \mid \textbf{let } x = M \textbf{ in } A \\
\texttt{Terms} & L, M, N & ::= x \mid V \mid M N \mid \textbf{let } x = M \textbf{ in } N
\end{array}$$

Evaluation rule

$$\begin{array}{lll}
(\lambda x.\ M)N & \longrightarrow & \textbf{let } x = N \textbf{ in } M \\
\textbf{let } x = V \textbf{ in } C[x] & \longrightarrow & \textbf{let } x = V \textbf{ in } C[V] \\
(\textbf{let } x = L \textbf{ in } M)N & \longrightarrow & \textbf{let } x = L \textbf{ in } M N \\
\textbf{let } y = (\textbf{let } x = L \textbf{ in } M) \textbf{ in } N & \longrightarrow & \textbf{let } x = L \textbf{ in let } y = M \textbf{ in } N
\end{array}$$

Problem

Church-Rosser, but nondeterministic

$$
\begin{aligned}
&(\mathbf{let}\ x = \lambda y.\ y\ \mathbf{in}\ x)\ (\lambda z.\ z) \\
\longrightarrow\ &(\mathbf{let}\ x = \lambda y.\ y\ \mathbf{in}\ \lambda y.\ y)\ (\lambda z.\ z)
\end{aligned}
$$

$$
\begin{aligned}
&(\mathbf{let}\ x = \lambda y.\ y\ \mathbf{in}\ x)\ (\lambda z.\ z) \\
\longrightarrow\ &\mathbf{let}\ x = \lambda y.\ y\ \mathbf{in}\ x\ (\lambda z.\ z)
\end{aligned}
$$

Standard Reduction

Syntax

$$
\begin{array}{llcl}
\texttt{Values} & V & ::= & \lambda x.\ M \\
\texttt{Answers} & A & ::= & V \mid \mathbf{let}\ x = M\ \mathbf{in}\ A \\
\texttt{Terms} & L, M, N & ::= & x \mid V \mid M N \mid \mathbf{let}\ x = M\ \mathbf{in}\ N \\
\texttt{Evaluation} & L, M, N & ::= & [] \mid E M \mid \mathbf{let}\ x = M\ \mathbf{in}\ E \\
\texttt{Contexts} & & \mid & \mathbf{let}\ x = E\ \mathbf{in}\ E[x]
\end{array}
$$

Evaluation rule

$$
\begin{array}{lcl}
(\lambda x.\ M)N & \longrightarrow & \mathbf{let}\ x = N\ \mathbf{in}\ M \\
\mathbf{let}\ x = V\ \mathbf{in}\ E[x] & \longrightarrow & \mathbf{let}\ x = V\ \mathbf{in}\ E[V] \\
(\mathbf{let}\ x = M\ \mathbf{in}\ A)N & \longrightarrow & \mathbf{let}\ x = M\ \mathbf{in}\ A N \\
\mathbf{let}\ y = (\mathbf{let}\ x = M\ \mathbf{in}\ A)\ \mathbf{in}\ E[y] & & \\
\longrightarrow \mathbf{let}\ x = M\ \mathbf{in}\ \mathbf{let}\ y = A\ \mathbf{in}\ E[y] & &
\end{array}
$$

Metatheory

Lemma 4.2

Given a program M, either M is an answer, or there exists a unique evaluation context E and a standard redex N such that $M \equiv E[N]$

Theorem 5.8 (correctness)

If $M \underset{\texttt{name}}{\longrightarrow}_* V$ then, there exists A such that $M \underset{\texttt{need}}{\longrightarrow}_* A$

Let-Less Formulation

Syntax

`Values`	V	$::=$	$\lambda x.\ M$
`Answers`	A	$::=$	$V \mid (\lambda x.\ A)M$
`Terms`	L, M, N	$::=$	$x \mid V \mid MN$
`Evaluation`	L, M, N	$::=$	$[] \mid EM \mid (\lambda x.\ M)E$
`Contexts`		$\mid$	$(\lambda x.\ E[x])E$

Evaluation rule

$$
\begin{array}{lcl}
(\lambda x.\ C[x])V & \longrightarrow & (\lambda x.\ C[V])V \\
(\lambda x.\ L)MN & \longrightarrow & (\lambda x.\ LN)M \\
(\lambda x.\ L)((\lambda y.\ M)N) & \longrightarrow & (\lambda y.\ (\lambda x.\ L)M)N
\end{array}
$$

アウトライン

1. Motivation
2. Call-by-need
   - What's Call-By-Need
   - Small-Step Operational Semantic
   - Relation to Big-Step Operational Semantics

Big-Step Operational Semantics