Optimal reduction of $\lambda$-expressions

Andrea Asperti

Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
asperti@cs.unibo.it

Talk given at the Tata Institute of Technology, Mumbay (INDIA)

January 2009
Content

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
Content

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
Content

1 Optimal Sharing
   • Duplication and creation
   • Copies and families
   • Virtual redexes and paths

2 Sharing and Unsharing
   • Duplication rules

3 Optimal reduction and Linear Logic
   • Linear Logic and the \(!\) comonad
   • Lambda encoding
   • Control rules

4 The Cost of Optimal Sharing
   • A Typed setting
   • Eta expansion
   • A complexity bound
Content

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
Outline

1 Optimal Sharing
   • Duplication and creation
   • Copies and families
   • Virtual redexes and paths

2 Sharing and Unsharing
   • Duplication rules

3 Optimal reduction and Linear Logic
   • Linear Logic and the ! comonad
   • Lambda encoding
   • Control rules

4 The Cost of Optimal Sharing
   • A Typed setting
   • Eta expansion
   • A complexity bound
Δ = \lambda x.(x \ x), F = \lambda x.(x \ y), I = \lambda x.x
The problem: share the reduction of redexes in a same family.
Copies and families

The problem: share the reduction of redexes in a same family.
Call by need is not optimal

We need to reduce under lambdas:

\[ n = \lambda xy. (x (x \ldots (x y))) \]
\[ 2 = \lambda xy. (x (x y)) \]
\[ I = \lambda x. x \]

\[
\begin{align*}
n \ 2 \ 1 \ 1 & \rightarrow (2 \ldots (2 \ (2 \ 1)) \ldots) I \\
& \rightarrow (2 \ldots (2 \ (\lambda y. I (I y))) \ldots) I \\
& \rightarrow (2 \ldots (\lambda y. (\lambda y. I (I y)) (\lambda y. I (I y)) y) \ldots) I
\end{align*}
\]

In Haskell or SML the reduction of \( n \ 2 \ 1 \ 1 \) is exponential in \( n \) (while innermost reduction is obviously linear).
We need to reduce under lambdas:

\[ n = \lambda xy. (x (x \ldots (x y))) \]
\[ 2 = \lambda xy. (x (x y)) \]
\[ I = \lambda x. x \]

\[
\begin{align*}
  n \ 2 \ 1 \ 1 & \rightarrow (2 \ldots (2 \ (2 \ 1)) \ldots)I \\
  & \rightarrow (2 \ldots (2 \ (\lambda y. I(I y))) \ldots)I \\
  & \rightarrow (2 \ldots (\lambda y. (\lambda y. I(I y))(\lambda y. I(I y)) y) \ldots)I
\end{align*}
\]

In Haskell or SML the reduction of \( n \ 2 \ 1 \ 1 \) is exponential in \( n \) (while innermost reduction is obviously linear).
Innermost reduction of needed redexes is not optimal

\[ \Delta = \lambda x. (x \ x) \quad M = \lambda x. (x \ I) \ \lambda y. (\Delta (y \ z)) \]

\[ \Delta = \lambda x. (x \ x) \quad M = \lambda x. (x \ I) \ \lambda y. (\Delta (y \ z)) \]

\[ (\Delta (I \ z)) \quad (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (\Delta (z)) \quad ((I \ z) (I \ z)) \]

\[ (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (I \ z) (I \ z)) \]

\[ (z (I \ z)) \quad ((I \ z) z) \]

\[ (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (z (I \ z)) \quad ((I \ z) z) \]

\[ (\Delta (z)) \quad ((I \ z) (I \ z)) \]

\[ (\Delta (I \ z)) \quad (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (z (I \ z)) \quad ((I \ z) z) \]

\[ (\Delta (z)) \quad ((I \ z) (I \ z)) \]

\[ (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (z (I \ z)) \quad ((I \ z) z) \]

\[ (\Delta (I \ z)) \quad (\lambda y. (y \ z) (y \ z) \ I) \]

\[ (z (I \ z)) \quad ((I \ z) z) \]

\[ (\Delta (z)) \quad ((I \ z) (I \ z)) \]
Virtual redexes and paths

\[
\lambda x \Delta \\
\lambda y \\
\lambda w \\
x \\
y \\
w \\
\Delta \\
dcb \\
e \\
\lambda w \\
w \\
\Delta \\
abcde \\
\lambda w \\
w \\
z
\]
Outline

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
Optimal Sharing
Sharing and Unsharing
Optimal reduction and Linear Logic
The Cost of Optimal Sharing

Sharing and Unsharing

\[ \lambda xy. (\lambda z. (z (z y)) \lambda w. (x w)) \rightarrow \lambda xy. (x (x y)) \]
Sharing and Unsharing (2)

A.Asperti

Optimal reduction of $\lambda$-expressions
Sharing and Unsharing (3)

(1) → (2) → (3)

Optimal reduction of λ-expressions
Sharing and Unsharing (4)

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ \lambda \]

\[ @ \]

\[ @ \]

\[ @ \]

\[ @ \]
Logarithmic Church Integers

The Church Integer $2^n$
Basic Rules

Problem: matching fans
Basic Rules

Problem: matching fans

A. Asperti
Optimal reduction of $\lambda$-expressions
Outline

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
Linear Logic and the ! comonad

A three-dimensional interpretation.
Lambda encoding

\[ [x] = \]

\[ [\lambda x. M] = \]

\[ [(M N)] = \]
Control rules
Outline

1. Optimal Sharing
   - Duplication and creation
   - Copies and families
   - Virtual redexes and paths

2. Sharing and Unsharing
   - Duplication rules

3. Optimal reduction and Linear Logic
   - Linear Logic and the ! comonad
   - Lambda encoding
   - Control rules

4. The Cost of Optimal Sharing
   - A Typed setting
   - Eta expansion
   - A complexity bound
the type of fans may only decrease during the reduction
the type of fans may only *decrease* during the reduction
1. $\eta_0(x) = x$

2. $\eta_{\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow o}(x) = \lambda y_1 : \alpha_1 \cdots \lambda y_n : \alpha_n . (x \eta_{\alpha_1}(y_1) \cdots \eta_{\alpha_n}(y_n))$

(a) $\eta$-expansion;  
(b) $\eta_{(o \rightarrow o) \rightarrow o \rightarrow o}(x)$
Duplication of eta-expanded variables

Fan propagation stops on atomic types
Duplication of eta-expanded variables

Fan propagation stops on atomic types
Example: duplication of $\eta(o \to o) \to o \to o(x))$
Optimal root:
For any subterm $\lambda x : \sigma.P$ of $M$:

$$(*) \quad \lambda x : \sigma.P \rightarrow \lambda x'.(\lambda x.P \eta_\sigma(x'))$$

Let $or(M)$ be the result of the transformation, and $\Delta(M)$ be Lamping’s normal form after firing all redexes of type $(*)$.

The number of $\beta$-redexes in the normalization of $\Delta(M)$ is linear in its size!
Optimal root:
For any subterm $\lambda x : \sigma . P$ of $M$:

$$\lambda x : \sigma . P \rightarrow \lambda x' . (\lambda x . P \, \eta_{\sigma}(x'))$$

Let $or(M)$ be the result of the transformation, and $\Delta(M)$ be Lamping’s normal form after firing all redexes of type ($\ast$).

The number of $\beta$-redexes in the normalization of $\Delta(M)$ is linear in its size!
The size of $\Delta(M)$

Let $n(x)$ be the multiplicity of $x$ in $M$.

**Theorem**

The number $f$ of families of $\text{or}(M)$ is less than

$$2 \times \left( |M| + \sum_{x \in M} n(x) \times |\sigma(x)| \right)$$

or also, by trivial simplifications

$$2 \times |M| \times (\max_{\sigma} + 1)$$
The cost of optimal sharing

Corollary

The cost of optimal $\beta$-reduction is not elementary recursive

Hint: A generic non-elementary function can be simulated by a “small” simply typed term, and reduced in a linear number of optimally shared redexes.

Not a negative result for Lamping’s technique! Extremely expensive because extremely powerful.

The number of families is not a good measure of the intrinsic complexity of $\lambda$-terms.
The cost of optimal sharing

Corollary

*The cost of optimal $\beta$-reduction is not elementary recursive*

Hint: A generic non-elementary function can be simulated by a “small” simply typed term, and reduced in a linear number of optimally shared redexes.

Not a negative result for Lamping’s technique! Extremely expensive because extremely powerful.

The number of families is not a good measure of the intrinsic complexity of $\lambda$-terms.
The cost of optimal sharing

Corollary

*The cost of optimal $\beta$-reduction is not elementary recursive*

Hint: A generic non-elementary function can be simulated by a “small” simply typed term, and reduced in a linear number of optimally shared redexes.

Not a negative result for Lamping’s technique! Extremely expensive because extremely powerful.

The number of families is not a good measure of the intrinsic complexity of $\lambda$-terms.
The cost of optimal sharing

Corollary

The cost of optimal $\beta$-reduction is not elementary recursive

Hint: A generic non-elementary function can be simulated by a “small” simply typed term, and reduced in a linear number of optimally shared redexes.

Not a negative result for Lamping’s technique! Extremely expensive because extremely powerful.

The number of families is not a good measure of the intrinsic complexity of $\lambda$-terms.
The cost of optimal sharing

**Corollary**

*The cost of optimal \( \beta \)-reduction is not elementary recursive*

Hint: A generic non-elementary function can be simulated by a “small” simply typed term, and reduced in a linear number of optimally shared redexes.

Not a negative result for Lamping’s technique! Extremely expensive because extremely powerful.

The number of families *is not* a good measure of the intrinsic complexity of \( \lambda \)-terms.
A. Asperti, S. Guerrini

The Optimal Implementation of Functional Programming Languages.