

# 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, Mumbai (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

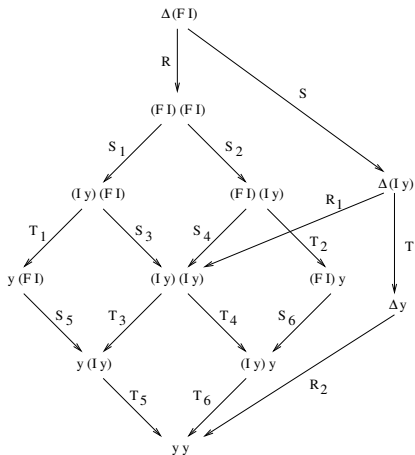
# Content

- 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

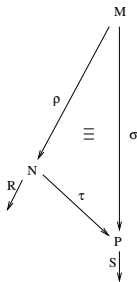
# Sharing



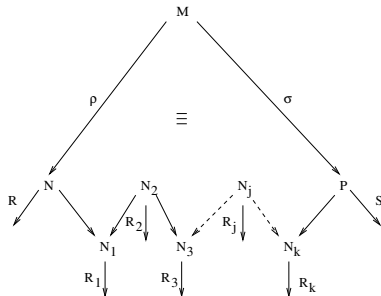
$$\Delta = \lambda x.(x x), F = \lambda x.(x y), I = \lambda x.x$$



## Copies and families



COPY



FAMILY

The problem: share the reduction of redexes in a same family.

## Call by need is not optimal

We need to reduce under lambdas:

$$\mathbf{n} = \lambda xy.(x (x \dots (x y)))$$

$$\mathbf{2} = \lambda xy.(x (x y))$$

$$\mathbf{I} = \lambda x.x$$

$$\begin{aligned} n \ 2 \ I \ I &\rightarrow (2 \dots (2 (2 \ I)) \dots) \ I \\ &\rightarrow (2 \dots (2 (\lambda y. I (I \ y))) \dots) \ I \\ &\rightarrow (2 \dots (\lambda y. (\lambda y. I (I \ y)) (\lambda y. I (I \ y)) \ y) \dots) \ I \end{aligned}$$

In Haskell or SML the reduction of  $n \ 2 \ I \ I$  is exponential in  $n$  (while innermost reduction is obviously linear).

## Call by need is not optimal

We need to reduce under lambdas:

$$\mathbf{n} = \lambda xy.(x (x \dots (x y)))$$

$$\mathbf{2} = \lambda xy.(x (x y))$$

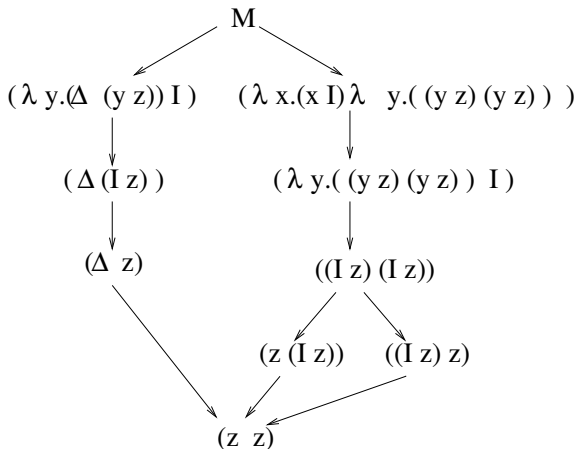
$$\mathbf{I} = \lambda x.x$$

$$\begin{aligned} n \mathbf{2} \mathbf{I} \mathbf{I} &\rightarrow (2 \dots (2 (2 \mathbf{I})) \dots) \mathbf{I} \\ &\rightarrow (2 \dots (2 (\lambda y. \mathbf{I} (\mathbf{I} y))) \dots) \mathbf{I} \\ &\rightarrow (2 \dots (\lambda y. (\lambda y. \mathbf{I} (\mathbf{I} y)) (\lambda y. \mathbf{I} (\mathbf{I} y)) y) \dots) \mathbf{I} \end{aligned}$$

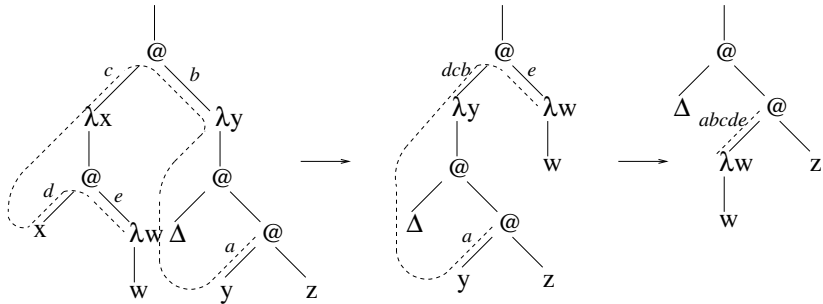
In Haskell or SML the reduction of  $n \mathbf{2} \mathbf{I} \mathbf{I}$  is exponential in  $n$  (while innermost reduction is obviously linear).

# Innermost reduction of needed redexes is not optimal

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



# Virtual redexes and paths



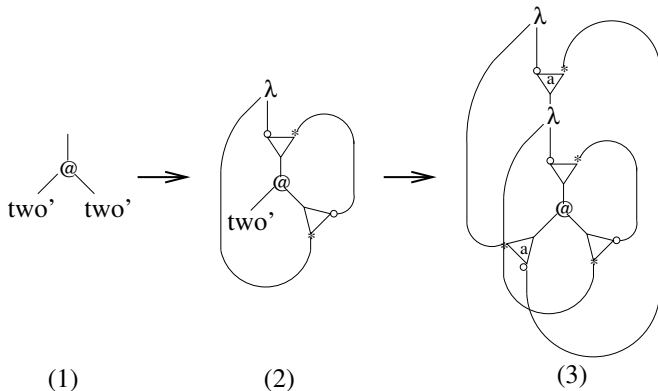
# Outline

- 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



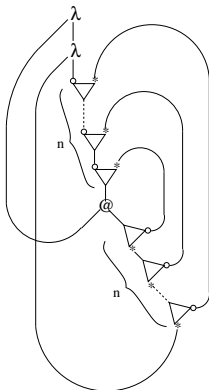


# Sharing and Unsharing (3)



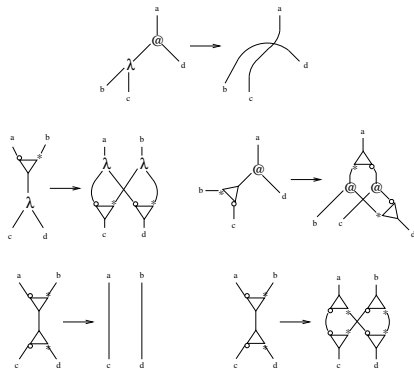


# Logarithmic Church Integers



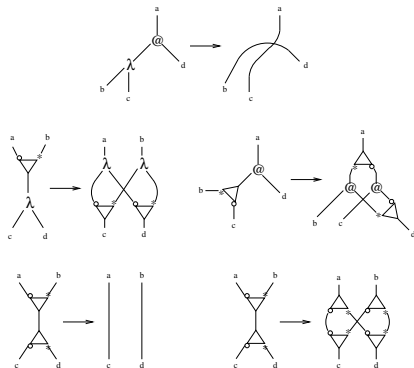
The Church Integer  $2^n$

# Basic Rules



Problem: matching fans

# Basic Rules

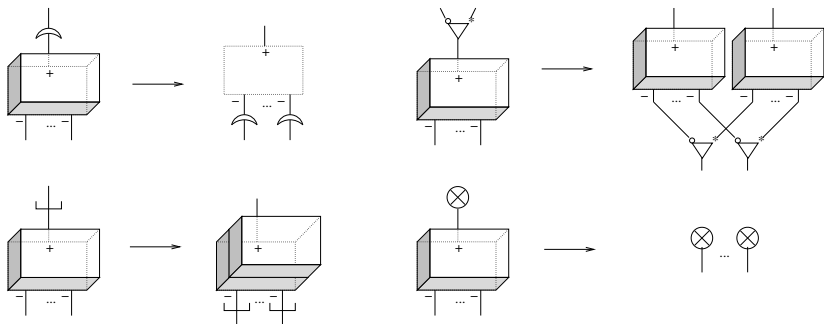


Problem: matching fans

# 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.

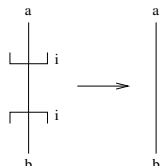
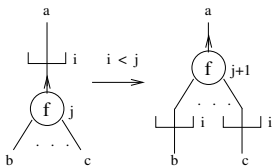
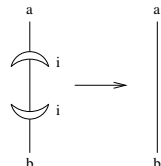
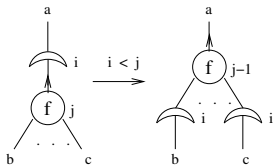
# Lambda encoding

$$[x] = \text{---} \cup \text{---}$$

$$[\lambda x.M] = \text{---} \circ \text{---} \oplus \text{---} \oplus \text{---}$$

$$[(M N)] = \text{---} \oplus \text{---} \oplus \text{---} \oplus \text{---}$$

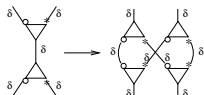
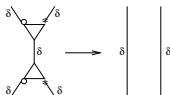
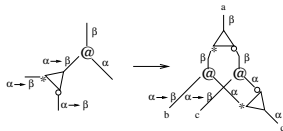
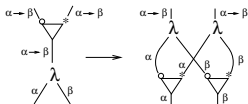
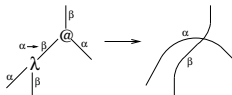
# Control rules



# Outline

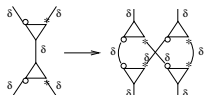
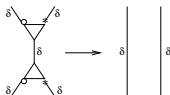
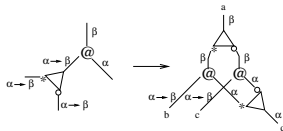
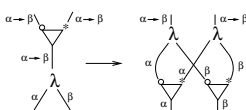
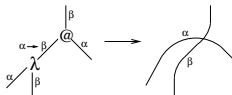
- 1 Optimal Sharing
  - Duplication and creation
  - Copies and families
  - Virtual redexes and paths
- 2 Sharing and Unsharing
  - Duplication rules
- 3 Optimal reduction and Linear Logic
  - Linear Logic and the ! comonad
  - Lambda encoding
  - Control rules
- 4 The Cost of Optimal Sharing
  - A Typed setting
  - Eta expansion
  - A complexity bound

# Typed rules



the type of fans may only *decrease* during the reduction

# Tped rules

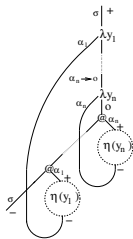


the type of fans may only *decrease* during the reduction

# Eta expansion

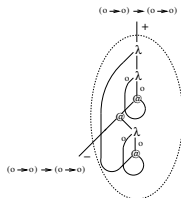
1  $\eta_o(x) = x$

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



(a)

(a)  $\eta$ -expansion;

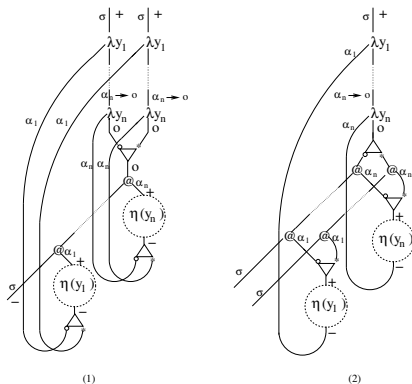


(b)

(b)  $\eta_{(o \rightarrow o) \rightarrow o \rightarrow o}(x)$

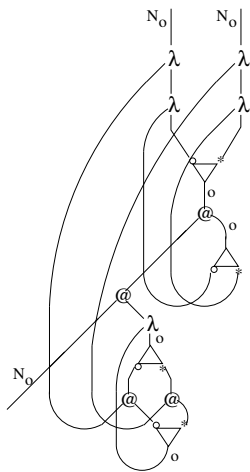


# Duplication of eta-expanded variables



Fan propagation stops on atomic types

# Example: duplication of $\eta_{(o \rightarrow o) \rightarrow o \rightarrow o}(x)$



## Linear family reduction

### 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!

## Linear family reduction

### 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!

# The size of $\Delta(M)$

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

## Theorem

*The number  $f$  of families of  $or(M)$  is less than*

$$2 * (|M| + \sum_{x \in M} n(x) * |\sigma(x)|)$$

*or also, by trivial simplifications*

$$2 * |M| * (\max_{\sigma} + 1)$$

# 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.

## To learn more

A.Asperti, S.Guerrini

### **The Optimal Implementation of Functional Programming Languages.**

Cambridge Tracts in Theoretical COmputer Science n.45,  
Cambridge University Press, 1998.