Optimality and intrinsic computational complexity

Andrea Asperti

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

talk offered at the

Anniversary Workshop in honour of Gérard Berry et Jean-Jacques Lévy

Gerardmer, France, 2-4 February, 2011

Content

Optimal sharing [Levy 78]

The problem

- optimize reduction of lambda terms by maximazing sharing
- provide an intrinsic measure of the computational cost of lambda terms

Optimal sharing [Levy 78]

The problem

- optimize reduction of lambda terms by maximazing sharing
- provide an intrinsic measure of the computational cost of lambda terms

Call by need is not optimal

We need to reduce under lambdas:

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

$$\begin{array}{lcl} n\ 2\ I\ I & \rightarrow & (2 \ldots (2\ (2\ I)) \ldots)I \\ & \rightarrow & (2 \ldots (2\ (\lambda y.I(I\ y))) \ldots)I \\ & \rightarrow & (2 \ldots (\lambda y.(\lambda y.I(I\ y))(\lambda y.I(I\ y))\ y) \ldots)I \end{array}$$

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

time to go beyond weak reduction?

Call by need is not optimal

We need to reduce under lambdas:

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

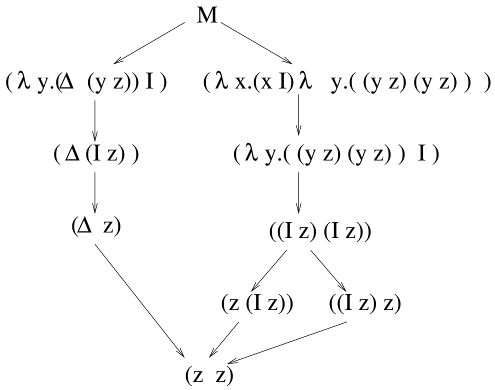
$$\begin{array}{lcl} n\ 2\ I\ I & \rightarrow & (2 \ldots (2\ (2\ I)) \ldots)I \\ & \rightarrow & (2 \ldots (2\ (\lambda y.I(I\ y))) \ldots)I \\ & \rightarrow & (2 \ldots (\lambda y.(\lambda y.I(I\ y))(\lambda y.I(I\ y))\ y) \ldots)I \end{array}$$

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

time to go beyond weak reduction?

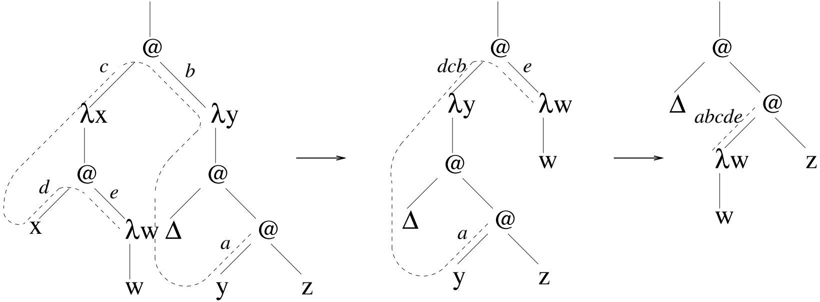
Innermost reduction of needed redexes is not optimal

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



If M contains multiple occurrences of x we have a tension between the outermost and the innermost strategy.

Virtual redexes and paths

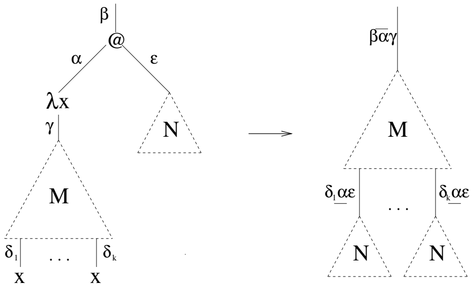


Labeling [Levy 76]

Every subterm gets a label

$$((\lambda x.M)^{\alpha}\ N)^{\beta} \rightarrow \beta \cdot \overline{\alpha} \cdot M[\underline{\alpha} \cdot N/x]$$

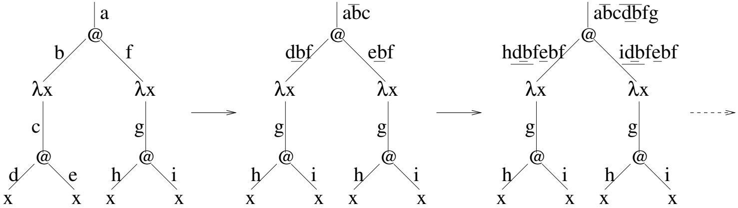
where $\alpha \cdot M^{\beta} = M^{\alpha\beta}$



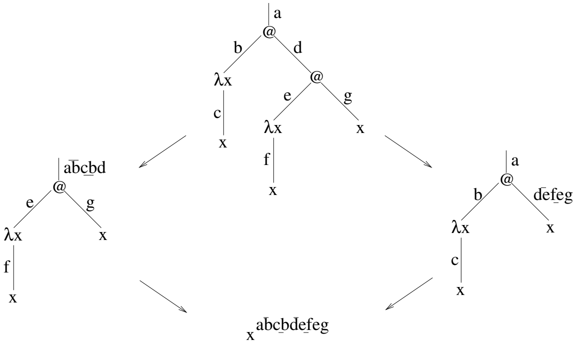
The label keeps a trace of the creation history of each edge (in fact, a path in the original term)

Two redexes belong to the same family if they have the same label.

Example 1

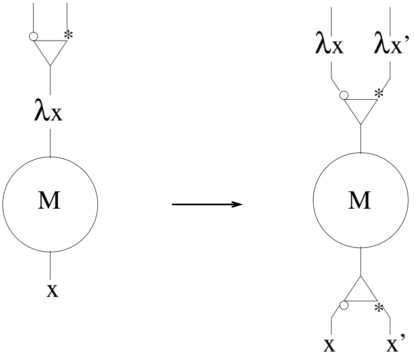


Example 2



Sharing and Unsharing [Lamping 90]

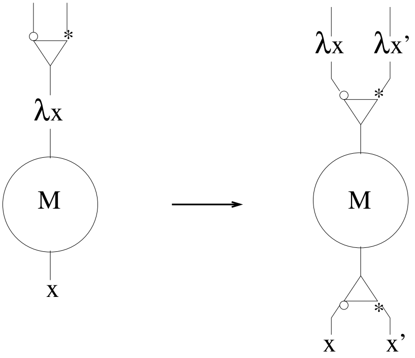
Optimal sharing is feasible!



At what price?

Sharing and Unsharing [Lamping 90]

Optimal sharing is feasible!



At what price?

The cost of family reduction

Asperti, Mairson [AM98]. **Parallel beta reduction is not elementary recursive**. Inf. & Comp. 2001 (short version at POPL'98)
the cost of reducing a redex family cannot be bounded by any elementary function.

The result is composed of two different parts:

- a technique (eta-expansion technique) aimed to prove that, with a bit of preprocessing (few eta and beta expansion steps) each simply typed lambda terms can be reduced in a linear number of family reduction steps.
- a clever exploitation of an old result by Meyer stating that the reduction of simply typed lambda terms cannot be bounded by any elementary function.

Ten years of oblivion

- Negative Perception of the result in [AM98], **but why**?
  Like to say that Meyer's result is a negative result for the simply typed lambda calculus . . .

In particular [AM98]

- does not say that pursuing optimal sharing is a bad idea
- says **nothing at all** (and hence nothing bad) about Lamping's algorithm.

It says indeed that the mere **number** of redex families is not a good measure of the "intrinsic complexity" of lambda terms. But this is **intriguing**: it means that the problem is still open.

We must **understand what's wrong** with the notion of family, **improve it** or **propose an alternative notion of cost**.

The cost of redexes

creation the redex is created by firing other redexes: the cost should be attributed to them

firing we create a large number of new connections (recall we are firing a whole family in a single step). The connections are only relevant as fragments of new redexes: they will be taken into account when these redexes will be fired.

So, what's so drastically wrong in considering the firing of a family of redexes as a finitistic operation?

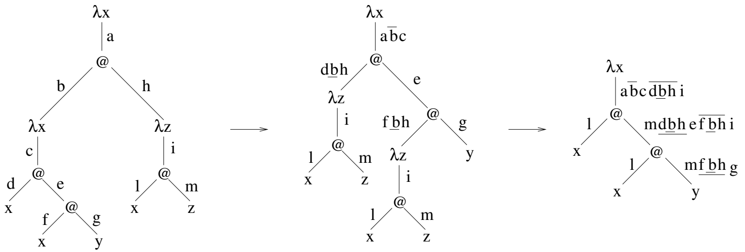
Why can we reduce a simply typed lambda terms in such a small number of family steps? Let's have a closer at the eta expansion technique.

The eta-expansion technique

Consider the term

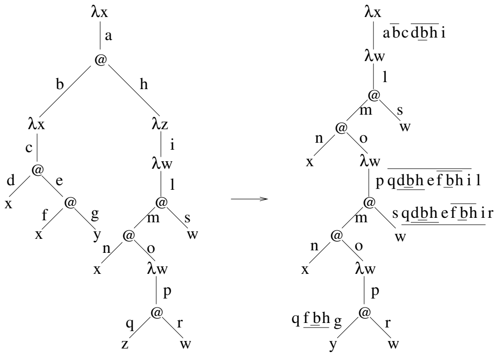
$$\lambda x.x\ (x\ y)$$

When x is instantiated we create two different redexes. But x is a function, and can be eta-expanded into $\lambda z.x\ z$: if instead of sharing x we *share* its eta-expansion (with a let-in or a beta expansion), we shall be able to share the redex created by the instantiation of x.



The eta-expansion technique (2)

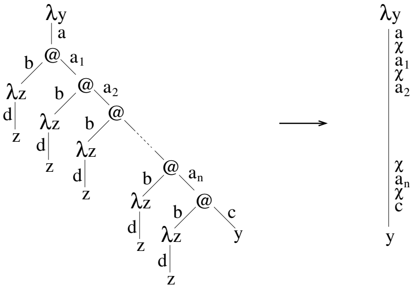
We can adapt the technique to the type of x.
For instance, if $x : (o \to o) \to (o \to o)$ we obtain



We not only share the redex at n, but also all redexes created by its firing.

Firing a family is not a finitistic operation

Consider a church integer $n = \lambda x.\lambda y.x(\ldots(x\ y))$ where all x have a same label b (as can be obtained by eta-expansion). Suppose now to instantiate x with an identity an to fire the unique redex family.



We fired a single redex, but the cost of connecting all edges a_i (all different!) is clearly linear in n.

So little to do, so much time

Although it improves optimal sharing, the eta-expansion technique is, clearly, **not a real optimization**.

It is a **trick** explicitly conceived to scoff at the notion of family:

- redexes have simple labels: the actual cost is not part of their creation history
- all work is "postponed" in new connections that are never going to be part of a full redex
- a huge number of connections are required to generate a single link

The optimal cost

Labeling is OK.

We should just take it more seriously. Instead of counting the number of different labels for redexes, we should count

the whole number of different labels

generated along a standard reduction sequence.
(already suggested by Lawall and Mairson in '96).

Let us call this quantity the optimal cost $o(t)$ of a λ-term t.

A lot of interesting problems

- compute $o(t)$ for a large number of terms
- try to characterize the class of terms for which traditional implementation techniques are indeed optimal
- try to understand what can be the performance loss of traditional (weak) techniques (worse than exponential?)
- try to understand what can be the performance loss w.r.t. (say) the best reduction strategy using explicit substitutions.
- compare $o(t)$ with the number of duplication steps in Lamping's algorithm
- ...

A child of five would understand this.
Send someone to fetch a child of five.
– Groucho Marx

Essential bibliography (1)

- A.Asperti. $\delta \circ !\epsilon = 1$: Optimizing optimal λ-calculus implementations. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95*, volume 914 of *Lecture Notes in Computer Science*, pp. 102–116, Kaiserslautern, Germany, 1995. Springer-Verlag.
- A.Asperti. On the complexity of beta-reduction. In *Proc. of the Twentythird Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'96)*, St. Petersburg Beach, Florida, 1996.
- A.Asperti, P.Coppola and S.Martini. Optimal duplication is not elementary recursive. Information and Computation V.193, n.1, pp. 21-56; 2004.
- A.Asperti, V.Danos, C.Laneve, and L.Regnier. Paths in the λ-calculus. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, Paris, France, 1994.
- A.Asperti, S.Guerrini. The Optimal Implementation of functional programming languages. Cambridge Tracts in Theoretical Computer Science, 45. Cambridge University Press, 1998.
- A. Asperti, H. Mairson. Parallel beta reduction is not elementary recursive. Information and Computation, V.170, n.1, pp.49-80; 2001.

Essential bibliography (2)

- G.Gonthier, M.Abadi, and J.J.Lévy. The geometry of optimal lambda reduction. In *Conference record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'92)*, pp. 15–26, Albequerque, New Mexico, January 1992.
- J.Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. of Seventeenth Annual ACM Symposyum on Principles of Programming Languages*, pp. 16–30, San Francisco, California, January 1990.
- J.L.Lawall, H.G.Mairson. Optimality and Inefficiency: What Isn't a Cost Model of the Lambda Calculus? ICFP, 1996, pp. 92-101.
- J.J. Lévy. An algebraic interpretation of the $\lambda\beta K$-calculus and an application of labelled λ-calculus. *Theoretical Computer Science*, 2(1):97–114, 1976.
- J.J. Lévy. Réductions Correctes et Optimales dans le lambda-calcul. PhD Thesis, Université Paris VII, 1978.
- J.J. Lévy. Optimal reductions in the lambda-calculus. In *To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*, Seldin and Hindley eds., Academic Press, 1980, pp. 159–191.
- A.R.Meyer. *The inherent computational complexity of theories of ordered sets.* Proceedings of the International Congress of Mathematicians, 1974, pp. 477–482.