The cost of usage in the λ-calculus

Andrea Asperti[1] Jean-Jacques Lévy[2]

[1]DISI, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
Email: asperti@cs.unibo.it

[2]INRIA Paris-Rocquencourt
Le Chesnay, France
Email: jean-jacques.levy@inria.fr

LICS 2013

June 25-28, 2013, New Orleans, USA

Abstract

- A λ-term has a normal form if and only if its leftmost outermost (normal) reduction is finite.
- The normal reduction σ_s can be much longer (a double-exponential, in the worst case) than the shortest reduction σ:

$$|\sigma_s| \leq |M|^{2^{|\sigma|}}$$

  where M is the initial term
- In this paper, we prove that, in many interesting cases, we can reduce this bound to a mere factorial:

$$|\sigma_s| \leq !|\sigma|$$

Content

Standardization

A reduction $M_0 \stackrel{R_1}{\to} M_1 \stackrel{R_2}{\to} \cdots \stackrel{R_k}{\to} M_k \stackrel{R_{k+1}}{\to} \cdots$ is standard when for any i, j such that $1 \leq i < j$, the redex R_j is not residual of a redex in M_{i-1} to the left of R_i.

Standardization theorem [Curry & Feys [10]]

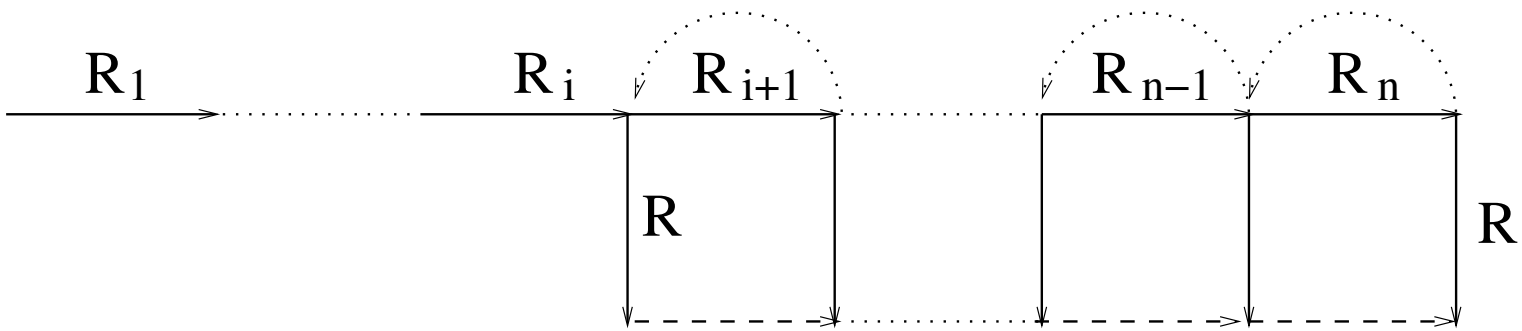
For any reduction $\sigma : M \twoheadrightarrow N$ there exists a *standard* reduction between the same terms.

Corollary: a term M has a normal form *if and only if* the leftmost reduction sequence originated by M is finite (normalization theorem).

Xi's approach

Lots of proofs: Mitschke [21], Klop [15], Barendregt [7]; Gonthier et al. [12], ...

A recent "inductive" approach by Xi [25] allows to compute an upper bound for the lenght of the standard reduction.



See also Kashima [14] and Guidi [13].

Xi's upper bound

The multiplicity $m(R)$ of a redex $R = (\lambda x.M)N$ is the number of occurrences of x in M.

Standardization - Xi [25]

Let $\sigma = R_0 R_1 \dots R_n$ be an arbitrary reduction; then there exists a standard reduction σ_s such that

$$|\sigma_s| \leq (1 + \max\{m(R_1), 1\}) \cdots (1 + \max\{m(R_n), 1\}) \qquad (1)$$

The multiplicity of a redex R_i is bound by the size $|M_i|$ of the term it belongs to, and the size of M_i is at most a double exponential $|M|^{2^i}$ of the size of the initial term $M = M_0$, so

$$|\sigma_s| \leq |M|^{2^1} \cdots |M|^{2^{n-1}} < |M|^{2^n} \qquad (2)$$

Example

Let $S = \lambda x.xx(xx)$, $I = \lambda x.x$, $K = \lambda x, y.x$, $P = IK$. Then,

$$SP \xrightarrow{R_0} SK \xrightarrow{R_1} KK(KK) \xrightarrow{R_2} (\lambda y.K)(KK) \xrightarrow{R_3} K$$

$m(R_0) = 1, m(R_1) = 4, m(R_2) = 1, m(R_3) = 0$

Xi's-bound: $5 \times 2 \times 2 = 20$.

The multiplicity of SK is too big:

we should only count the occurrences of live variables (live multiplicity)

Example

Let $S = \lambda x.xx(xx)$, $I = \lambda x.x$, $K = \lambda x, y.x$, $P = IK$. Then,

$$SP \xrightarrow{R_0} SK \xrightarrow{R_1} KK(KK) \xrightarrow{R_2} (\lambda y.K)(KK) \xrightarrow{R_3} K$$

$m(R_0) = 1, m(R_1) = 4, m(R_2) = 1, m(R_3) = 0$

Xi's-bound: $5 \times 2 \times 2 = 20$.

The multiplicity of SK is too big:

we should only count the occurrences of live variables (live multiplicity)

Main ideas

- a framework for studying what is live (needed) or dead (useless) inside a lambda term:

  minimal prefixes

- a bound for the live multiplicity of redexes.
  The size of minimal prefixes can grow very fast, but can only

  decrease very slowly

  Do not count the cost for **producing** a term, but for **consuming** it.

Minimal prefixes

We consider terms with holes _.

Given two terms M and N, M is a prefix of N ($M \preceq N$) if N matches M except on _ subterms.

Stability theorem [Berry[9],Plotkin[22]]

For any term M producing a normal form A, there exists a unique minimum prefix $\lfloor M \rfloor_A$ of M producing A.

(can be generalized to an arbitrary rigid prefix A: see the paper).

Live multiplicity

Let M be a term producing A.

- a subterm P of M is live (for A) in M if (the root of) P belongs to $\lfloor M \rfloor_A$
- the live multiplicity $m(R)$ of a redex $(\lambda x.P)Q$ in a term M is the number of live occurrences of x (that is the number of occurrences of x in $\lfloor M \rfloor_A$).

Example Let $M = (\lambda f.\, ff(f\Delta))(\lambda x.I) \twoheadrightarrow I$; then

$$\lfloor M \rfloor_I = (\lambda f.f\_(f\_))(\lambda x.I)$$

The subterm Δ is dead for I in M, as well as the second occurrence of f; so the live multiplicity of f is 2.

Slow consumption

Definition

(applicative size) The applicative size $|M|_@$ of a λ-term M is the number of applications in it.

Lemma

(@-survival) Suppose $R : M \to N$; if P is a live applicative subterm of M, and $P \neq R$, then it has at least one live residual in N.

Lemma

(@ slow consumption) For any rigid prefix A produced by M, if $R : M \to N$, then

$$|\lfloor N \rfloor_A|_@ \geq |\lfloor M \rfloor_A|_@ - 1$$

Examples: @-size of minimal prefixes

red.	$(\lambda f.f\ f\ (f\ \Delta))(\lambda x.I)$	$\to$	$(\lambda x.I)(\lambda x.I)((\lambda x.I)\Delta))$	$\to$	$(\lambda x.I)(\lambda x.I)I$	$\to$	II	$\to$	I
min.pref.	$(\lambda f.f\ \_\ (f\ \_))(\lambda x.I)$		$(\lambda x.I)\ \_\ ((\lambda x.I)\ \_))$		$(\lambda x.I)\ \_\ I$		II		I
@-size	4		3		2		1		0

red.	$(\lambda x.xx\,(xx))(IK)$	$\to$	$(\lambda x.xx(xx))K$	$\to$	$KK(KK)$	$\to$	$(\lambda y.K)(KK)$	$\to$	K
min.pref.	$(\lambda x.xx\ \_)(IK)$		$(\lambda x.xx\ \_)K$		$KK\ \_$		$(\lambda y.K)\ \_$		K
@-size	4		3		2		1		0

reduction	$(\lambda x.xxx)(KI)$	$\to$	$(\lambda x.xxx)(\lambda y.I)$	$\to$	$(\lambda y.I)(\lambda y.I)(\lambda y.I)$	$\to$	$I(\lambda y.I)$	$\to$	$\lambda y.I$
min.pref.	$(\lambda x.x\ \_\ x)(KI)$		$(\lambda x.x\ \_\ x)(\lambda y.I)$		$(\lambda y.I)\ \_\ (\lambda y.I)$		$I(\lambda y.I)$		$\lambda y.I$
@-size	4		3		2		1		0

Standardization with bounds

Theorem

Given a reduction $\sigma : M \twoheadrightarrow N$, *where* $\sigma = R_0 R_1 \dots R_n$ *and any term* A *produced by* M *(and* N*), there exists a standard reduction* $std(\sigma) : M \overset{st}{\twoheadrightarrow} P$ *such that* $P \approx_A N$ *and*

$$|std(\sigma)| \leq (1 + max\{m(R_1), 1\}) \dots (1 + max\{m(R_n), 1\})$$

where $m(R_i)$ *is the multiplicity of* $R_i : M_i \to M_{i+1}$ *in* $\lfloor M_i \rfloor_A$ *(that is, we only count the variables which are live for* A*).*

Sinking to a rigid prefix

Lemma

Let $\sigma : M \twoheadrightarrow N$ and let A be a rigid prefix of N. Then, for any variable x in M, if $m(x)$ is its live multiplicity (for A) we have

$$m(x) \leq |\sigma| + |A|_{@} + 1$$

Theorem

Let $\sigma : M \overset{st}{\twoheadrightarrow} N$ and let A be a rigid prefix of N. Then there exists a standard reduction $\sigma_s : M \to B$ such that $B \approx_A N$ and

$$|\sigma_s| \leq \frac{(|\sigma| + |A|_{@})!}{(1 + |A|_{@})!}$$

Corollaries

Corollary

Let M be a term having N has a normal form. Let $\sigma : M \stackrel{st}{\twoheadrightarrow} N$ be an arbitrary reduction. Then there exists a standard reduction $\sigma_s : M \to N$ such

$$|\sigma_s| \leq \frac{(|\sigma| + |N|_@)!}{(1 + |N|_@)!}$$

The previous theorem is particularly significant when N is small:

Corollary

If M is a term reducing to a variable or a boolean ($\lambda x.\lambda y.x$ or $\lambda x.\lambda y.y$), the length of the normal reduction σ_n is at worst a factorial of the length of the best reduction σ, that is

$$|\sigma_n| \leq (|\sigma|)!$$

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Conclusions

Preliminary considerations

- weak reduction is very similar to first order reduction;
- the computational interest of beta-reduction (if any) lies in deep reduction (need to *share* inside lambdas);
- we know examples of lambda terms where deep reduction works exponentially better than the best weak reduction strategy (see Asperti&Guerrini[4]);

By the results in this paper:

- either we cannot expect much better speed-ups
- or these cannot be observed sticking to the realm of lambda terms and its sequential reduction strategies (need more parallelisms/sharing).

Bibliography

Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy.

Explicit substitutions.
J. Funct. Program., 1(4):375–416, 1991.

Martín Abadi, Butler W. Lampson, and Jean-Jacques Lévy.

Analysis and Caching of Dependencies.
In *Proceedings of the 1996 ACM International Conference on Functional Programming (ICFP'96), Philadelphia*, pp 83–91, 1996.

Beniamino Accattoli, Ugo Dal Lago.

On the Invariance of the Unitary Cost Model for Head Reduction.
In *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , Nagoya, Japan*, pp 22–37. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

Andrea Asperti and Stefano Guerrini.

The Optimal Implementation of Functional Programming Languages, volume 45 of *Cambridge Tracts in Theoretical Computer Science*.
Cambridge University Press, 1998.

Andrea Asperti and Cosimo Laneve.

Paths, computations and labels in the lambda-calculus.
Theor. Comput. Sci., 142(2):277–297, 1995.

Andrea Asperti and Harry G. Mairson.

Parallel beta reduction is not elementary recursive.
Inf. Comput., 170(1):49–80, 2001.

Bibliography

Henk Barendregt.

The Lambda Calculus, its Syntax and Semantics.
North Holland, 1984.

Alessandro Berarducci.

Infinite λ-calculus and non-sensible models.
In *Logic and algebra (Pontignano, 1994)*, pp 339–377. Dekker, New York, 1996.

Gérard Berry.

Modèles stables du lambda-calcul.
PhD thesis, Paris 7, 1978.

H.B.Curry and R.Feys.

Combinatory Logic.
North Holland Publishing Company, 1958.

Jean-Yves Girard, Yves Lafont, and Paul Taylor.

Proofs and Types, volume 7 of *Cambridge Tracts in Theoretical Computer Science*.
Cambridge University Press, 1989.

Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès.

An abstract standardisation theorem.
In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA*, pages 72–81. IEEE Computer Society, 1992.

Ferruccio Guidi.

Standardization and Confluence in Pure Lambda-Calculus Formalized for the Matita Theorem Prover.
Journal of Formalized Reasoning, 5(1):1–25, 2012.

Bibliography

Ryo Kashima.

A proof of the standardization theorem in lambda-calculus.
Technical Report Research Reports on Mathematical and Computing Sciences, C-145,, Tokyo Institute of Technology, 2000.

Jan Willem Klop.

Combinatory Reduction Systems.
PhD thesis, CWI, Amsterdam, 1980.

Richard Kennaway, Jan Willem Klop, M. Ronan Sleep and Fer-Jan de Vries.

Infinitary Lambda Calculus.
Theor. Comput. Sci., 175(1):93-125, 1997.

John Lamping.

An Algorithm for Optimal Lambda Calculus Reduction.
Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages (POPL'90), San Francisco, California, USA, ACM press, pages 16-30, 1990.

Jean-Jacques Lévy.

An algebraic interpretation of the lambda beta - calculus and a labeled lambda - calculus.
In *Lambda-Calculus and Computer Science Theory, Proceedings of the Symposium Held in Rome, March 25-27, 1975*, volume 37 of *Lecture Notes in Computer Science*, pages 147–165, 1975.

Jean-Jacques Lévy.

Réductions corrcectes et optimales dans le lambda calcul.
PhD thesis, University of Paris 7, 1978.

Bibliography

Albert R. Meyer.

The inherent computational complexity of theories of ordered sets.
In *Proceedings of the International Congress of Mathematicians, Vancouver*, pages 477–482, 1974.

Gerd Mitschke.

The standardisation theorem for the λ-calculus.
Z. Math. Logik. Grundlag. Math, 25:29–31, 1979.

Gordon D. Plotkin.

LCF considered as a programming language.
Theor. Comput. Sci., 5(3):223–255, 1977.

Richard Statman.

The typed lambda-calculus is not elementary recursive.
In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA*, pages 90–94. IEEE Computer Society, 1977.

Masako Takahashi.

Parallel reductions in λ-calculus.
Information and Computation, 118(1):120–127, 1995.

Hongwei Xi.

Upper bounds for standardizations and an application.
J. Symb. Log., 64(1):291–303, 1999.