2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science

The cost of usage in the λ-calculus

Andrea Asperti
University of Bologna
Bologna, Italy
Email: asperti@cs.unibo.it

Jean-Jacques Lévy
INRIA Paris-Rocquencourt
Le Chesnay, France
Email: jean-jacques.levy@inria.fr

***Abstract*—A new "inductive" approach to standardization for the λ-calculus has been recently introduced by Xi, allowing him to establish a double-exponential upper bound $|M|^{2^{|\sigma|}}$ for the length of the standard reduction relative to an arbitrary reduction σ originated in M. In this paper we refine Xi's analysis, obtaining much better bounds, especially for computations producing small normal forms. For instance, for terms reducing to a boolean, we are able to prove that the length of the standard reduction is at most a mere *factorial* of the length of the shortest reduction sequence. The methodological innovation of our approach is that instead of counting the cost for *producing* something, as is customary, we count the cost of *consuming* things. The key observation is that the part of a λ-term that is *needed* to produce the normal form (or an arbitrary rigid prefix) may rapidly augment along a computation, but can only *decrease* very slowly (actually, linearly).**

I. Introduction

The standardization theorem, stated and proved for the first time by Curry and Feys back in 1958 [11], is one of the major results of λ-calculus. It states that for any reduction $\sigma : M \twoheadrightarrow N$ there exists a *standard* reduction between the same terms that avoids to reduce residuals of previous redexes: in other words, redexes are fired, if ever, as soon as they are met in leftmost outermost order. As a consequence, a term M has a normal form *if and only if* the leftmost reduction sequence originated by M is finite (normalization theorem), explaining the relevance of lazy strategies in functional programming languages.

There are also important relations between standardization and sequentiality, as first observed by Plotkin in his seminal work on PCF [23]. For instance, we can easily argue that parallel-or cannot be expressed in the λ-calculus: there is no context $C[\_,\_]$ such that $C[\Omega,\Omega]$ has no normal form, but $C[\Omega, I]$ and $C[I,\Omega]$ have normal forms, where $\Delta = \lambda x.\, xx$, $\Omega = \Delta\Delta$ and $I = \lambda x.x$. Indeed, with Curry's standardization theorem, we know that a normal form is always reached by the leftmost-outermost (normal) reduction. There are three possibilities: if the normal reduction of $C[M,N]$ ignores M and N then $C[\Omega,\Omega]$ has a normal form too; if the normal reduction starts looking at M, then $C[\Omega, I]$ cannot have a normal form; conversely, if it starts looking at N, then $C[I,\Omega]$ cannot have a normal form. Hence, the deterministic feature of the normal reduction enforces the stability property of the predicate "has a normal form".

Many proofs of the standardization theorem can be found in the literature such as [22], [16], and [8]; abstract properties of the standardization process have been investigated in [13]. A novel approach, that avoids the notion of development and is entirely based on structural induction on λ-terms and derivations, has been recently proposed by Xi [26] (see also [15]). Xi's approach not only results in a very concise proof, but it also establishes an upper bound to the number of steps in the standard reduction sequence obtained from a given reduction sequence.

Let us define the multiplicity $m(R)$ of a redex $R = (\lambda x.M)N$ as the number of occurrences of x in M. Let $\sigma = R_0 R_1 \dots R_n$ be an arbitrary reduction. Xi proves that there exists a standard reduction σ_s such that

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

where $|\sigma_s|$ is the length of σ_s.

Since the multiplicity of a redex R_i is obviously 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 $M = M_0$, we also have

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

In Xi's own words, the previous bounds are "not very tight, but intelligible". However, it is possible to provide concrete example of terms exhibiting a double exponential explosion, such as the following one.

Example 1: Let $\mathsf{I} = \lambda x.x$ and $\mathsf{two} = \lambda x.\lambda y.x(xy)$. Let us define

$$M_0 = xx$$
$$M_{n+1} = \mathsf{two}(\lambda x.M_n)x$$

In four steps

$$M_{n+1} \to M_n[M_n/x]$$

The (normal form of the) term M_n grows as a double exponential in n, and an innermost reduction get to the normal form in a linear number of steps. Consider now the term

$$E_n = \lambda x.M_n(Iz)$$

If we reduce the argument first, we can still compute the normal form in a linear number of steps; if we reduce M_n first, we will eventually create a double exponential number of copies of (Iz).

$\Box$

Since we have indeed examples of λ-terms whose standard reduction grows as a double exponential of the length of the shortest reduction, the problem seems to be essentially settled.

In fact, it is not. The point is that the only known examples are based, as in the previous case, on fast growing λ-terms producing at the end huge results, where just the cost of their *reading* dwarfs the number of steps required for their construction. As a matter of fact, we are never really interested in *computing* functions but, possibly, just to partially evaluate them in view of their future application. At the end, we are merely interested in atomic results (data types that, like variables, or booleans, take a fixed amount of space and can be read in a constant amount of time).

So, the natural question is if we can refine Xi's analysis in case we restrict the attention to computations leading to a known rigid structure. The answer is surprising and quite unexpected: not only it is possible to generally give a better bound, but this bound is *sensibly* smaller than Xi's one. For instance, in case of computations producing a variable (or a boolean) we are able to prove that the length of the standard (hence normal) reduction σ_s is at most a factorial of the length of the shortest reduction sequence σ:

$$|\sigma_s| \leq |\sigma|! \qquad (3)$$

As far as we know, our approach is original and innovative too. The general idea is that instead of trying to count the cost for *producing* something, as customary, we count the cost of *consuming* things, where by consuming, we do not mean *wasting* (that is a very easy and quite inexpensive task) but really *exhausting* things through they concrete usage. In the case of redexes, the difference should be clear: something is to throw away a redex inside garbage, and something else is to consume it by its firing.

In order to distinguish between the two previous forms of consumption, we need to be able to distinguish what is live (needed) or dead (useless) inside a lambda term, along a given computation. The technical tool that we shall use to this aim is offered by prefixes: partially specified terms, where some subterms have been replaced by a dummy constant _ (a hole). For a lambda term M producing a normal form A, there exists a minimal prefix $\lfloor M \rfloor_A$ of M that is enough for producing A, and we say that a subterm of M is live for A if it belongs to the minimal prefix.

The main technical contribution of the paper is a refinement of Xi'x bound where we replace the notion of multiplicity of a redex with that of "live multiplicity": we only need to count the occurrences of the variable that are live for the given reduction.

The other important observation is that the live prefix of a term consumes *very slowly* (at most linearly) along any reduction. Since the size of the live prefix is an upper bound to the live multiplicity, is then very easy to get our factorial bound 3.

The structure of the paper is the following. Section II is devoted to the formal definition of prefixes and their theory; the main result in this section is the Slow Consumption Lemma 7. In Section III, we rephrase Xi's approach focusing on live prefixes of lambda terms; the main result is Theorem 2 that refines Xi's bound of Equation (1) by just considering the "live" multiplicity of variables. Finally, in Section IV we put together these two results to get the factorial bound of Theorem 3.

II. Prefixes

We consider λ-terms augmented with a constant $\_$. Formally, this gives a theory of contexts, but we are more interested to look at them as *prefixes*, or incomplete terms, where some part of the term is not specified. Let Λ^c be the set of these terms. The notions of substitution, β-reduction, and so on are completely standard, so we omit them.

Definition 1: Given two terms M and N in Λ^c we say that M is a prefix of N ($M \preceq N$) if N matches M except on $\_$ subterms.

Lemma 1: (lifting) Let $P \preceq M$; if $P \to Q$, then there exists N such that $Q \preceq N$ and $M \to N$.

Proof: Trivial. ∎

A. Rigid prefixes

Following [17], we say that a λ-term is a 0-term if it cannot reduce to an abstraction.

Definition 2: A prefix A of M is rigid if A is in normal form and morevoer all occurrences of _ in head position of an application correspond to 0-subterms of M.

If A is a rigid prefix of M and $M \to M'$, then A is a rigid prefix of M', that explains the terminology. Any redex ever fired along a reduction rooted in M is "below" A, in the sense that if $M \twoheadrightarrow N$ and $R : N \to N'$, then the occurrence of R is N (expressed as a path) is of the form $u \cdot v$ where u is occurrence of _ in A.

Rigid prefixes are finite approximants of Berarducci trees[9]. They can only be of form $\lambda x.A_1$, $xA_1A_2\dots A_n$ or $\_A_1A_2\dots A_n$ where A_i are rigid prefixes and $n \geq 0$. For instance $\_(\lambda x.x\_)\_$ is a rigid prefix of $M = \Omega(\lambda x.x(Ix))(IIx)$. But $\_(\lambda x.x\_)(\_Ix)$ is not rigid prefix of M.

Definition 3 (production): Given a prefix A, we say that M *produces* rigid A if there exists N such that $M \twoheadrightarrow N$ and A is rigid prefix of N.

It is easy to show that rigid prefixes enjoy a Church-Rosser property. But more important for us, they respect the stability property[10].

Lemma 2: For any term M producing a rigid prefix A, there exists a unique minimum prefix $\lfloor M \rfloor_A$ of M producing A.

Proof: The proof is the analogous for Berarducci trees of the proof for Bohm (or Lévy-Longo) trees in [10]. It is in fact the one by Plotkin using the standardization theorem in PCF [23]. ∎

Remark 1: The notion of minimal prefix can also be easily understood in terms of the labeled calculus [19] or paths [5]. Suppose M produces A. In the former cases, given an initial labeling for M, the minimal prefix $\lfloor M \rfloor_A$ is the prefix of M containing all labels appearing in A. In the latter case, $\lfloor M \rfloor_A$ can be understood as the minimal prefix of M containing all ancestors of paths in A.

Definition 4 (live and dead): Let M be a term producing a rigid A:

- a subterm P of M is *live for* A in M if (the root of) P belongs to $\lfloor M \rfloor_A$
- a redex $(\lambda x.P)Q$ is *live for* A in M if its function part $(\lambda x.P)$ is live for 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 $\lfloor M \rfloor_A$).

For instance $M = (\lambda f.\, ff(f\Delta))(\lambda x.I)$ reduces to I, and

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

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.

Notice that, in general, a minimal prefix of a term does not reduce to a minimal prefix. For instance, let $P = \lambda x.xII$, $\mathsf{fst} = \lambda x.\lambda y.x$, $\mathsf{snd} = \lambda x.\lambda y.y$ and let us consider the term

$$M = (\lambda x.x\,\mathsf{fst}\,(x\,\mathsf{snd}))P$$

that reduces to I. Then, $\lfloor M \rfloor_I = M$; however M reduces to

$$N = (\lambda x.xII)\,\mathsf{fst}\,((\lambda x.xII)\,\mathsf{snd})$$

and

$$\lfloor N \rfloor_I = (\lambda x.xI\,\_)\,\mathsf{fst}\,((\lambda x.x\,\_\,I))\,\mathsf{snd})$$

However, if $M \to N$, the minimal prefix of N is a prefix of the reduct of the minimal prefix of M. Formally:

Lemma 3: For any rigid prefix A produced by M and any redex $R : M \to N$ then either R is dead for A and hence $\lfloor M \rfloor_A = \lfloor N \rfloor_A$, or R is live for A and $R : \lfloor M \rfloor_A \to N'$ where $\lfloor N \rfloor_A = \lfloor N' \rfloor_A \preceq N' \preceq N$.

Proof: By the minimality of the prefix. Slight subtlety: the top pair application and abstractions nodes of R cannot overlap the boundary of $\lfloor M \rfloor_A$ since A is rigid. ∎

As a corollary of the previous result, we cannot have zombies around: a live subterm can only be a residual of a live subterm (or, stated the other way round, deads remain deads).

Lemma 4: (no zombies) For any rigid prefix A produced by M, any redex $R : M \to N$, and any subterm P of N, if P is live for A in N then it is a residual of a live subterm for A in M.

Proof: Obvious corollary of Lemma 3. ∎

Corollary 1: (live multiplicity may only increase) Let M be a term producing a rigid prefix A, and suppose $M \to M'$. Then, for any redex R' in M' residual of a redex R in M, $m_A(R) \leq m_A(R')$.

Proof: Since by Lemma 4 any live occurrence must be a residual of a live occurrence. ∎

Of course, it is not true that the any residual of a live subterm remains live. Nevertheless, a live subterm, if it is not consumed along the reduction, must have at least a live residual.

Lemma 5: (survival) For any rigid prefix A produced by M, any redex $R : M \to N$, and any subterm P of M, if $P \neq R$ is live for A in M, then it has at least one live residual in N.

Proof: Replacing P with $\_$ in $\lfloor M \rfloor_A$ would still reduce to a term larger then $\lfloor N \rfloor_A$ and hence would still produce A, contradicting the minimality of $\lfloor M \rfloor_A$. ∎

B. Sizes of rigid prefixes

Definition 5: (size) We define the size of a term (with holes) in the following way:

- $|x| = |\_| = 0$
- $|(M\,N)| = |M| + |N| + 1$
- $|\lambda x.M| = |M| + 1.$

As a consequence of the previous lemma we have the following major corollary concerning the size of the minimal prefixes:

Lemma 6: (slow consumption) For any rigid prefix A produced by M, if $R : M \to N$, then

$$|\lfloor N \rfloor_A| \geq |\lfloor M \rfloor_A| - 2$$

Proof: By lemma 5 any live subterm of P of M has *at least* on live residual, but for the application-lambda pair of nodes fired by R. ∎

The previous lemma, and its relation with the live multiplicity can be better stated if, in the size of the terms, we just count applications.

Definition 6: (applicative size) The applicative size $|M|_@$ of a λ-term M is the number of applications in it.

Then, the slow consumption lemma can be rephrased in the following terms:

Lemma 7: (@ slow consumption) For any rigid prefix A produced by M, if $R : M \to N$, then

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

Lemma 8: (rigid prefix bound) Let A be a rigid prefix, and let $\sigma : M \twoheadrightarrow N$ be any reduction producing A (that is such that $A \preceq N$). Then, for any variable x in M, if $m(x)$ is its live multiplicity (for A) we have

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

Proof: The live multiplicity for A of any variable in M is obviously bound by $|\lfloor M \rfloor_A|_@ + 1$. On the other side, by Lemma 7 the size of $|\lfloor M \rfloor_A|_@$ is at most equal to $|\sigma| + |A|_@$, and so we are done. ∎

Example 2: In Figure 1, we give a couple of examples of reductions, computing at each step the minimal prefix and its @-size.

C. Equivalence for production

Definition 7: Given two terms M and N producing A, we say that they are *equivalent for* A if they coincide on their minimal prefixes for A:

$$M \approx_A N \Leftrightarrow \lfloor M \rfloor_A = \lfloor N \rfloor_A$$

An important property of the previous equivalence relation is that it is preserved by reduction:

Lemma 9: If $M \approx_A N$ and $R : M \to M_1$, then either $M_1 \approx_A N$ (if R is dead for A) or there exists $N \to N_1$, $M_1 \approx_A N_1$.

Proof: When R is dead for A, then $\lfloor M \rfloor_A = \lfloor M_1 \rfloor_A$ by Lemma 3. When R is live for A, then (top part of) R also exists in N at same occurrence and we apply again Lemma 3. ∎

reduction	$(\lambda f.f\,f\,(f\,\Delta))(\lambda x.I)$	$\to$	$(\lambda x.I)(\lambda x.I)((\lambda x.I)\Delta))$	$\to$	$(\lambda x.I)(\lambda x.I)I$	$\to$	II	$\to$	I
minimal prefixes	$(\lambda f.f\,\_\,(f\,\_))(\lambda x.I)$		$(\lambda x.I)\,\_\,((\lambda x.I)\,\_))$		$(\lambda x.I)\,\_\,I$		II		I
@-size	4		3		2		1		0
reduction	$(\lambda x.xx\,(xx))(IK)$	$\to$	$(\lambda x.xx(xx))K$	$\to$	$KK(KK)$	$\to$	$(\lambda y.K)(KK)$	$\to$	K
minimal prefixes	$(\lambda x.xx\,\_)(IK)$		$(\lambda x.xx\,\_)K$		$KK\,\_$		$(\lambda y.K)\,\_$		K
@-size	4		3		2		1		0
reduction	$(\lambda x.xxx)(KI)$	$\to$	$(\lambda x.xxx)(\lambda y.I)$	$\to$	$(\lambda y.I)(\lambda y.I)(\lambda y.I)$	$\to$	$I(\lambda y.I)$	$\to$	$\lambda y.I$
minimal prefixes	$(\lambda x.x\,\_\,x)(KI)$		$(\lambda x.x\,\_\,x)(\lambda y.I)$		$(\lambda y.I)\,\_\,(\lambda y.I)$		$I(\lambda y.I)$		$\lambda y.I$
@-size	4		3		2		1		0

Fig. 1. @-size of minimal prefixes along some reductions

III. Standard and normal reductions

This section contains our revisitation of Xi's "inductive" approach to standardization [26], taking into into account the notion of live multiplicity for variables of Definition 4. In particular, we shall follow Kashima's version of the proof [15], that is particularly neat and clear (see also [14] for a completely formalized version of Kashima's standardization proof for the Matita Proof Assistant).

Definition 8: (Standard and normal reduction)

- A reduction $M_0 \overset{R_1}{\to} M_1 \overset{R_2}{\to} \cdots \overset{R_k}{\to} M_k \overset{R_{k+1}}{\to} \cdots$ 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.
- A reduction $M_0 \overset{R_1}{\to} M_1 \overset{R_2}{\to} \cdots \overset{R_k}{\to} M_k \overset{R_{k+1}}{\to} \cdots$ is *normal* (or leftmost-outermost) when for any $i \geq 1$ the redex R_i is the leftmost redex in M_i.

In the following, we shall use $M \overset{\mathsf{st}}{\twoheadrightarrow} N$ and $M \overset{\mathsf{norm}}{\twoheadrightarrow} N$ to respectively denote standard and normal reductions.

Definition 9: A normal reduction is *rigid* if it never reduces under a λ or at the right of a head variable.

Our notion of rigid normal reduction is called "head reduction in application" (abbreviated with "hap") in [15], so we shall keep this name in order to avoid useless confusion, and we shall use the notation $\overset{\mathsf{hap}}{\twoheadrightarrow}$ for it. Formally, $\overset{\mathsf{hap}}{\twoheadrightarrow}$ is the reflexive and transitive closure of the following generalization of the β-rule:

$$(\beta_{app}) \qquad (\lambda x.A_0)A_1 \ldots A_n \overset{\mathsf{hap}}{\twoheadrightarrow} A_0[A_1/x] \ldots A_n$$

It is immediate that any normal reduction is also standard and that a normal reduction is uniquely determined: if $\rho : M \overset{\mathsf{norm}}{\twoheadrightarrow} N$ and $\rho' : M \overset{\mathsf{norm}}{\twoheadrightarrow} N'$, then ρ is a prefix of ρ', or vice versa (the same is also true for $\overset{\mathsf{hap}}{\twoheadrightarrow}$, since $\overset{\mathsf{hap}}{\twoheadrightarrow}$ is a special case of normal reduction).

Any reduction of a term M may end up in a variable, in an abstraction or an application; if the reduction is standard, it can be decomposed according to the following lemma:

Lemma 10: (Decomposition)

- if $M \overset{\mathsf{st}}{\twoheadrightarrow} x$, then $M \overset{\mathsf{hap}}{\twoheadrightarrow} x$
- if $M \overset{\mathsf{st}}{\twoheadrightarrow} \lambda x.N$ then $M \overset{\mathsf{hap}}{\twoheadrightarrow} \lambda x.N'$ and $N' \overset{\mathsf{st}}{\twoheadrightarrow} N$
- if $M \overset{\mathsf{st}}{\twoheadrightarrow} (P\ Q)$ then $M \overset{\mathsf{hap}}{\twoheadrightarrow} (P_1\ Q_1)$, $P_1 \overset{\mathsf{st}}{\twoheadrightarrow} P$ and $Q_1 \overset{\mathsf{st}}{\twoheadrightarrow} Q$

  Proof: See [15]. ■

In fact, any reduction satisfying the conditions of the decomposition lemma is a standard reduction, hence "standard" is the smallest relation defined over reductions satisfying the decomposition lemma, that allows us to reason by cases on such a decomposition.

We recall some properties of the above reductions.

Lemma 11: If $M \overset{\mathsf{hap}}{\twoheadrightarrow} N$ then $M\,P \overset{\mathsf{st}}{\twoheadrightarrow} N\,P$.

Proof: By the definition of β_{app}. ■

Lemma 12: If $M \overset{\mathsf{hap}}{\twoheadrightarrow} N$ and $N \overset{\mathsf{st}}{\twoheadrightarrow} P$ then $M \overset{\mathsf{st}}{\twoheadrightarrow} N$.

Proof: By induction on the possible decompositions of $N \overset{\mathsf{st}}{\twoheadrightarrow} P$. ■

Lemma 13: If $M \overset{\mathsf{hap}}{\twoheadrightarrow} N$ then $M[P/x] \overset{\mathsf{hap}}{\twoheadrightarrow} N[P/x]$.

Proof: By induction on the length of $M \overset{\mathsf{hap}}{\twoheadrightarrow} N$. ■

Note that the previous result does not hold for normal reductions. Suppose that x appears as head variable of some applicative subterm $x\,A_1 \ldots A_n$; then the normal reduction can reduce a redex R in some A_i, but this redex would be at the right of any redex in P when we substitute P for x.

Lemma 14: If $\sigma : M \overset{\mathsf{st}}{\twoheadrightarrow} N$ and $\tau : P \overset{\mathsf{st}}{\twoheadrightarrow} Q$, there exists

$$\gamma : M[P/x] \overset{\mathsf{st}}{\twoheadrightarrow} N[Q/x]$$

moreover, $|\gamma| \leq |\sigma| + m|\tau|$ where m is the multiplicity of x in N.

Proof: By induction on the decomposition of the standard reduction, using lemma 13 (see e.g. [15]) ■

The main idea behind our results is that the previous lemma remains "morally" true even if we restrict the substitution to the live occurrences of x. To this aim, we shall exploit the following generalization of Lemma 14:

Lemma 15: Suppose $\sigma : M \overset{\mathsf{st}}{\twoheadrightarrow} N$ and $\tau : P \overset{\mathsf{st}}{\twoheadrightarrow} Q$. Let us split the occurrences of x in N in two sets x_{true} and x_{false}. Then

$$\gamma : M[P/x] \overset{\mathsf{st}}{\twoheadrightarrow} N[Q/x_{true}; P/x_{false}]$$

where $|\gamma| \leq |\sigma| + m|\tau|$ where m is the multiplicity of x_{true} in N.

Proof: By induction on the structure of $\sigma : M \overset{\mathsf{st}}{\twoheadrightarrow} N$, according to the decomposition lemma.

- If $\sigma \overset{\mathsf{hap}}{\twoheadrightarrow} y$, then we have two subcases:
  - $y \neq x_{true}$. Then, $\gamma = \sigma$.
  - $y = x_{true}$. Then $\gamma = \sigma\tau$, that is a standard reduction by Lemma 12.

- if $N = \lambda y.N'$, then σ can be decomposed in the following way: $M \overset{\mathsf{hap}}{\twoheadrightarrow} \lambda x.N'$ and $N' \overset{\mathsf{st}}{\twoheadrightarrow} N$. By induction hypothesis
$$\gamma_1 : N'[P/x] \overset{\mathsf{st}}{\twoheadrightarrow} N[Q/x_{true}; P/x_{false}]$$
hence
$$\gamma_2 : (\lambda y.N')[P/x] \overset{\mathsf{st}}{\twoheadrightarrow} (\lambda y.N)[Q/x_{true}; P/x_{false}]$$
Since by Lemma 13 we have a normal reduction
$$\gamma_3 : M[P/x] \overset{\mathsf{hap}}{\twoheadrightarrow} (\lambda y.N')[P/x]$$
the composition of $\gamma_3\gamma_2$ is standard and it readily satisfies the length conditions.
- similar to the previous one.

■

The main lemma is traditionally stated in the following way (see e.g. [26], [15])

Lemma 16: If $\sigma : A \overset{\mathsf{st}}{\twoheadrightarrow} (\lambda x.M)N$, there exists a standard reduction $\sigma' : A \overset{\mathsf{st}}{\twoheadrightarrow} M[N/x]$ such that
$$|\sigma'| \leq 1 + \max\{m(x), 1\}|\sigma|$$
where $m(x)$ is the multiplicity of x in M.

We want to rephrase the same lemma just taking into account the *live multiplicity* of x. In order to do so, we should first of all fix some rigid prefix produced by our computation. However, in order to get the right level of generality, we should embed our computation inside an arbitrary context $\mathbf{C}[\_]$, and consider some rigid prefix A_{O} produced inside this context. Moreover, at the end, we do not expect to precisely get $\mathbf{C}[M[N/x]]$, but just a term equivalent to it for the production of A_0.

So, this is the new statement:

Lemma 17: Let $\sigma : A \overset{\mathsf{st}}{\twoheadrightarrow} (\lambda x.M)N$, let $\mathbf{C}[\_]$ be an arbitrary context, and suppose $\mathbf{C}[A]$ (and hence $\mathbf{C}[(\lambda x.M)N]$) produces A_0. Then there exists a standard reduction $\sigma' : A \overset{\mathsf{st}}{\twoheadrightarrow} B$ where $\mathbf{C}[B] \approx_{A_0} \mathbf{C}[M[N/x]]$ and $|\sigma'| \leq 1 + \max\{m(x), 1\}|\sigma|$, where $m(x)$ is the live multiplicity of x in $\mathbf{C}[(\lambda x.M)N]$ for A_0.

Proof: By the decomposition Lemma 10, σ can be split in the following way:
- $\sigma_0 : A \overset{\mathsf{hap}}{\twoheadrightarrow} P\,Q$
- $\sigma_1 : P \overset{\mathsf{st}}{\twoheadrightarrow} \lambda x.M$
- $\sigma_2 : Q \overset{\mathsf{st}}{\twoheadrightarrow} N$

Moreover, for the same lemma, $\sigma_1 : P \overset{\mathsf{st}}{\twoheadrightarrow} \lambda x.M$ decomposes into
- $\sigma_{10} : P \overset{\mathsf{hap}}{\twoheadrightarrow} \lambda x.M'$
- $\sigma_{11} : M' \overset{\mathsf{st}}{\twoheadrightarrow} M$

Let us split the occurrences of x in M in two sets x_{live} and x_{dead} according to the fact they are live for A_o in $\mathbf{C}[(\lambda x.M)N]$ or not, (formally, if they belong or not to $\lfloor \mathbf{C}[(\lambda x.M)N] \rfloor_{A_{\mathrm{O}}}$). Then
$$\begin{array}{lll} A & \overset{\mathsf{hap}}{\twoheadrightarrow} P\,Q & \text{via } \sigma_0 \\ & \overset{\mathsf{hap}}{\twoheadrightarrow} (\lambda x.M')Q & \text{via } \sigma_{10} \\ & \overset{\mathsf{hap}}{\twoheadrightarrow} M'[Q/x] & \text{via } \beta_{app} \\ & \overset{\mathsf{st}}{\twoheadrightarrow} M[N/x_{live}, Q/x_{dead}] & \text{by Lemma 15 with } \sigma_{11}, \sigma_2 \end{array}$$

The length of this reduction is
$$\begin{array}{l} |\sigma_0| + |\sigma_{10}| + 1 + |\sigma_{11}| + m(x)|\sigma_2| \\ \quad \leq 1 + \max\{m(x), 1\}(|\sigma_0| + |\sigma_{10}| + |\sigma_{11}| + |\sigma_2|) \\ \quad = 1 + \max\{m(x), 1\}|\sigma| \end{array}$$
where $m(x)$ is the multiplicity of x_{true} in M.

Moreover,
$$\mathbf{C}[M[N/x_{live}, Q/x_{dead}]] \approx_{A_{\mathrm{O}}} \mathbf{C}[M[N/x]]$$
is a direct consequence of the way we defined live and dead occurrences.

■

We are ready to state and prove the general result.

Theorem 1: Let $\sigma : A \overset{\mathsf{st}}{\twoheadrightarrow} B$ and $R : B \to C$. Let $\mathbf{C}[\_]$ be an arbitrary context, and suppose $\mathbf{C}[A]$ (and hence $\mathbf{C}[B]$ and $\mathbf{C}[C]$) produces A_0. Then there exists a standard reduction $\sigma' : A \overset{\mathsf{st}}{\twoheadrightarrow} D$ where $\mathbf{C}[D] \approx_{A_0} \mathbf{C}[C]$ and $|\sigma'| \leq 1 + \max\{m, 1\}|\sigma|$ where m is the live multiplicity of R in $\mathbf{C}[B]$ (for A_{O}).

Proof: The proof is by induction on the one-step reduction $R : B \to C$.
- if $B = R = (\lambda x.M)N$, this is just Lemma 17.
- suppose $B = M\,N$ and $R' : M \to M'$. By the decomposition lemma, $\sigma : A \overset{\mathsf{st}}{\twoheadrightarrow} M\,N$ can be decomposed into
  - $\sigma_0 : A \overset{\mathsf{hap}}{\twoheadrightarrow} P\,Q$
  - $\sigma_1 : P \overset{\mathsf{st}}{\twoheadrightarrow} M$
  - $\sigma_2 : Q \overset{\mathsf{st}}{\twoheadrightarrow} N$

  By the induction hypothesis relative to σ_1, R' and the context $\mathbf{C}[(\_\,N)]$) (let us remark the essential use of the context here), there exists a term D' and a standard reduction $\sigma_1' : P \overset{\mathsf{st}}{\twoheadrightarrow} D'$ such that
$$(1) \qquad \mathbf{C}[(D'\,N)] \approx_{A_0} \mathbf{C}[(M'\,N)]$$
  and
$$(2) \qquad |\sigma_1'| \leq 1 + \max\{m, 1\}|\sigma_1|$$
  where m is the live multiplicity of R' in
  By composing $\sigma_0 : A \overset{\mathsf{hap}}{\twoheadrightarrow} P\,Q$, $\sigma_1' : P \overset{\mathsf{st}}{\twoheadrightarrow} D'$ and $\sigma_2 : Q \overset{\mathsf{st}}{\twoheadrightarrow} N$ we get a standard reduction $\sigma' : A \overset{\mathsf{st}}{\twoheadrightarrow} (D'N)$, where, by (1), $\mathbf{C}[(D'\,N)] \approx_{A_0} \mathbf{C}[(M'\,N)]$.
  As for the length, let us note first that the live multiplicity of R' in $\mathbf{C}[(M\,N)]$ is the same as the live multiplicity m of R in $\mathbf{C}[(M\,N)]$, since they are the same redex. So, the length of σ' is
$$\begin{array}{rl} |\sigma'| = & |\sigma_0| + |\sigma_1'| + |\sigma_2| \\ & \leq |\sigma_0| + 1 + \max\{m, 1\}|\sigma_1| + |\sigma_2| \\ & \leq 1 + \max\{m, 1\}(|\sigma_0| + \sigma_1| + |\sigma_2|) \\ & = 1 + \max\{m, 1\}(|\sigma|) \end{array}$$
- the case $B = M\,N$ and R is internal to N, or $B = \lambda x.M$, with R internal to M are analogous to the previous one, taking the suitable context.

■

Theorem 2: (standardization with bounds) Given a reduction $\sigma : M \twoheadrightarrow N$, where $\sigma = R_0R_1 \ldots R_n$ and any term

A produced by M (and N), there exists a standard reduction $std(\sigma) : M \overset{\text{st}}{\twoheadrightarrow} P$ 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).

Proof: Let $\sigma_i = R_0 R_1 \dots R_i$, and let $\ell_i = |std(\sigma_i)|$. We reason by induction on i. If $i = 0$ then $std(\sigma_i) = \sigma$, $\ell_0 = 1$ and we are done. Let us suppose, by induction hypothesis, that there exists $std(\sigma_{i-1}) : M \overset{\text{st}}{\twoheadrightarrow} P_i$ such that $P_i \approx_A M_i$

$$\ell_{i-1} \leq (1 + \max\{m(R_1), 1\}) \dots (1 + \max\{m(R_{i-1}), 1\})$$

If R_i is dead for A, we can just take σ_{i-1}. Otherwise, R_i is in $\lfloor P_i \rfloor_A = \lfloor M_i \rfloor_A$. Suppose $R_i : P_i \to P_{i+1}$. Since $P_i \approx_A M_i$, then $P_{i+1} \approx_A M_{i+1}$. By applying Lemma 17 to $std(\sigma_{i-1})$ and R_i we get a standard reduction $std(\sigma_i) : M \overset{\text{st}}{\twoheadrightarrow} Q$ such that $Q \approx_A P_i \approx_A M_i$, and

$$|std(\sigma_{i-1})| \leq 1 + \max\{m(R_i), 1\} \cdot \ell_i \leq \ell_i \cdot (1 + \max\{m(R_i), 1\})$$

as expected. ■

Note that, if N is in normal form, then by taking $A = N$ we get a standard reduction $M \overset{\text{st}}{\twoheadrightarrow} N$ in the traditional sense (but with an improved bound).

Example 3: Let $S = \lambda x.xx(xx)$ and $P = IK$. Then,

$$SP \to SK \to KK(KK) \to (\lambda y.K)(KK) \to K$$

The multiplicity of the four redexes (in Xi's sense) is respectively $1, 4, 1, 0$, so we would get a bound of $5 \cdot 2 \cdot 2 = 20$ steps for the standard reduction (the first redex does not count). However the *live* multiplicity of the second redex is just 2 (the relevant prefix of S is $\lambda x.xx\_$ that allows us to reduce the bound to 12 steps. Neither of the two bounds is very tight: the actual length of the standard reduction is of just 5 steps:

$$\begin{aligned} SP &\to IK(IK)(IK(IK)) \to K(IK)(IK(IK)) \\ &\to (\lambda y.IK)(IK(IK)) \to IK \to K \end{aligned}$$

The actual interest of the notion of live multiplicity is not that it offers a better bound to the length of the standard reduction, but to provide a conceptual explanation of the reasons why, even in presence of terms with huge multiplicities, a normal reduction may still be relatively short. The point is that the leftmost strategy, being safe, is also *parsimonious*: if part of the term is dead, it will never try to reduce inside it.

In the next section we shall see a surprising corollary of this fact.

IV. Sinking to a rigid prefix

We can now put together our two main results, namely Lemma 8 and Theorem 2.

Theorem 3: Let M be a term producing a rigid prefix A, and let $\sigma : M \overset{\text{st}}{\twoheadrightarrow} N$ be an arbitrary reduction producing A (that is such that $A \preceq N$). Then there exists a standard reduction $\sigma_s : M \to B$ such that $B \approx_A N$ and

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

Proof: Let $\sigma = R_0 R_1 \dots R_n$, so that $|\sigma| = n + 1$. Let $\tau_i = R_{i+1} \dots R_n$ be the tail of R_i in σ, so that $|\tau_i| = n - i$ By Theorem 2,

$$|\sigma_s| \leq \prod_{1 \leq i \leq n} (1 + \max\{m(R_i), 1\})$$

By Lemma 8, the multiplicity $m(R_i)$ is bound by

$$m(R_i) \leq |\tau_i| + |A|_@ + 1 = n - i + 1 + |A|_@$$

Hence,

$$|\sigma_s| \leq \prod_{1 \leq i \leq n} (n - i + 2 + |A|_@) = (n + 1 + |A|_@)!/(1 + |A|_@)!$$

■

Corollary 2: Let M be a term having N has a normal form. Let $\sigma : M \overset{\text{st}}{\twoheadrightarrow} N$ be an arbitrary reduction. Then there exists a standard reduction $\sigma_s : M \to N$ such

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

Proof: By Theorem 3, taking $A = N$. Note that $B \approx_A A$ if and only if $B = A$. ■

The previous theorem is particularly significant when A is small. For instance, if A is just a variable of a boolean $|A|_@ = 0$, so we get the following striking result:

Corollary 3: 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

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

Proof: Just observe that the applicative size of those terms is 0. ■

V. Conclusion

The standardization theorem essentially tells us that the "lazy" reduction strategy is more "precise" than any other reduction strategy: an eager step of evaluation can always be simulated by performing several times the "same redex" in a more lazy way. At the same time, this accuracy also allows us to avoid useless steps: lazy operations are only performed on demand, if strictly needed. For this reason, a precise investigation of what is needed (live) and what is useless (dead) inside a lambda for producing a given output seems to be a mandatory step in order to make a precise comparison between an arbitrary reduction and its standard version.

This is precisely what we did in this paper. As a technical tool to formalize live and dead subterms we used prefixes, that are partially specified terms, where some subterms have been replaced by a dummy constant $\_$ (a hole). An *output* A is a *rigid* prefix, that is a finite approximant of a Berarducci tree; then, for any lambda term M producing an output A, there exists a minimal prefix $\lfloor M \rfloor_A$ of M that is enough for producing A. Finally, a subterm of M is live for A if it belongs to the minimal prefix.

The first important observation is very simple: by firing a redex, the size of the minimal prefix can only decrease by the two @–λ nodes involved in the redex. As a consequence, the size of the minimal prefix relative to some output A cannot decrease more than linearly along a reduction producing A.

The other technical contribution of the paper is a refinement of Xi's bound where we replace the notion of multiplicity of a redex with that of "live multiplicity": we only need to count the occurrences of the variable that are live for the given reduction.

Since the live mutiplicity of variables is obviously bound by the size of the minimal prefix, it is easy to compose together the previous results to prove that the length of the standard reduction is at most a mere *factorial* of the length of the shortest reduction sequence (plus the size of the output).

It is probably worth to stress that the existence of *minimal* prefixes requires standardization: hence we do not have an alternative standardization proof. Like Baron Munchausen, pulling himself out the swamp by his hairs, we are boostrapping standardization, improving the analysis in a self-sustained process.

There is probably some space for a further improvement of the result, making a deeper analysis on the structure of terms. In particular, there are at least a couple of places where our upper bounds seem to provide only a rough approximation of reality: the first one is when we approximate the live multiplicity of variables with the size of the minimal prefix, and the second one is in Lemma 17 where one is forced to use the live multiplicity of the redex in a somewhat rough way, applying it on segments of derivations that could not need to be duplicated during standardization.

Since the analysis looks rough, it is natural to wonder if there are concrete examples of a factorial explosion. Let us consider the following term:

$$\mathbf{nn}II$$

where $n = \lambda x.\lambda y.x(x \ldots (xy))$ with n occurrences of x is a Church integer. Pursuing an innermost reduction strategy, this term can be reduced to I in a quadratic number of steps; however any weak strategy takes at least n^n steps (see [4] for a detailed analysis of a similar term). Hence, we are not very far from an optimal bound.

The actual interpretation of our result from the point of view of the complexity of beta-reduction is more problematic. In fact, it only tells us that the length of the best sequential reduction strategy is at most an arc-factorial of the normal reduction *as long as we remain in the syntactical realm of λ-terms*. However, the λ-calculus is not so good in representing *sharing*: we can give examples of terms where *any* reduction strategy ends up duplicating work, that is executing several times residuals of redexes that could be fired a single time along a different reduction (see [20]). More complex syntactical frameworks like explicit substitutions [1] or sharing graphs ([18], [4]) are needed to exploit the concrete potentialities of sharing in λ-terms. Alternatively, we can *mimic* sharing on λ-terms through a *parallel* notion of reduction, firing at each step all redexes belonging to a same family according to a given notion of sharing. Of course, firing redexes in parallel allows to decrease the applicative size of a live prefix in a much faster way than a sequential reduction. Since in λ-terms we can easily get enormous degrees of parallelism/sharing (see the appendix), we can still hope to be able to compute the normal form of a term in a number of steps smaller than the arc-factorial of the length of the standard reduction. The delicate point is that, in the case of parallel/shared reductions, contrarily to what happens for first order term rewriting systems [7], and for (some) sequential reduction strategies in the lambda-calculus [3], the length of the computation cannot be considered as a fair measure of its actual computational cost [6] (see also the appendix to this work).

REFERENCES

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *J. Funct. Program.*, 1(4):375–416, 1991.

[2] 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.

[3] 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.

[4] 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.

[5] Andrea Asperti and Cosimo Laneve. Paths, computations and labels in the lambda-calculus. *Theor. Comput. Sci.*, 142(2):277–297, 1995.

[6] Andrea Asperti and Harry G. Mairson. Parallel beta reduction is not elementary recursive. *Inf. Comput.*, 170(1):49–80, 2001.

[7] Martin Avanzini and Georg Moser. Complexity analysis by graph rewriting. In *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan*, LNCS 6009, pp 257–271, Springer, 2010.

[8] Henk Barendregt. *The Lambda Calculus, its Syntax and Semantics*. North Holland, 1984.

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

[10] Gérard Berry. *Modèles stables du lambda-calcul*. PhD thesis, Paris 7, 1978.

[11] H.B.Curry and R.Feys. *Combinatory Logic*. North Holland Publishing Company, 1958.

[12] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[13] 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.

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

[15] 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.

[16] Jan Willem Klop. *Combinatory Reduction Systems*. PhD thesis, CWI, Amsterdam, 1980.

[17] Richard Kennaway, Jan Willem Klop, M. Ronan Sleep and Fer-Jan de Vries. Infinitary Lambda Calculus. Theor. Comput. Sci., 175(1):93-125, 1997.

[18] 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.

[19] 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.

[20] Jean-Jacques Lévy. *Réductions corrcectes et optimales dans le lambda calcul*. PhD thesis, University of Paris 7, 1978.

[21] 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.

[22] Gerd Mitschke. The standardisation theorem for the λ-calculus. *Z. Math. Logik. Grundlag. Math*, 25:29–31, 1979.

[23] Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):223–255, 1977.

[24] 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.

[25] Masako Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118(1):120–127, 1995.

[26] Hongwei Xi. Upper bounds for standardizations and an application. *J. Symb. Log.*, 64(1):291–303, 1999.

Appendix

A remark on parallelism and sharing The potential parallelism inherent in λ-terms can be easily understood by considering parallel β-reduction in Takahashi's sense [25] (parallel reductions in [20] are a *refinements* of this notion, focused on *sharing*). We recall this argument here since, as far as we know, it has never been spelled out before.

In the simply typed case, it is traditional to define a notion of *degree* of a redex R in the following way (see e.g.[12]).

Definition 10 (degree): The *degree* $\partial(T)$ of a type T is defined by:

- $\partial(A) = 1$ if A is atomic
- $\partial(U \to V) = \max\{\partial(U), \partial(V)\} + 1$

The *degree* of a redex $(\lambda x : U.M)N$ is $\mathsf{degree}(U \to V)$, where V is the type of M.
The *degree* $\partial(M)$ of a term M is the maximum among the degrees of all its redexes.

A crucial property of the simply typed lambda calculus is that a redex R of type $U \to V$ may only create redexes of type U or of type V, hence with a degree strictly less than that of R. As a consequence, each simply typed lambda term M can be reduced to its normal form with a number of parallel reduction steps bound by its degree $\partial(M)$. On the other side, we can encode complex (Kalmar-elementary) computations in λ-terms with low-degrees (see [21], [24]), proving that the cost of a parallel reduction (in Takahashi's sense) is not elementary recursive. In [6] it is proved that most of these parallel redexes are actually *sharable* in Lévy's optimal sense, and hence the cost of *sharing* a single redex cannot be bound by any elementary function.