Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 338 (2018) 23–43

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

(In)Efficiency and Reasonable Cost Models

Beniamino Accattoli[1]

Inria & LIX, École Polytechnique

Abstract

This divulgative paper is about time cost models for the λ-calculus. To allow the definition of standard complexity classes such as `P` or `EXP` directly in the λ-calculus, a cost models has to be reasonable, that is, polynomially related to the one of Turing machines. For the λ-calculus the existence of an evaluation strategy whose number of steps is a reasonable cost model has been a long-standing open problem. Positive answers to special cases were known since 1995, but a solution for the general case has been provided only in 2014. The problem is peculiar because some of its aspects are somewhat counterintuitive. This paper is devoted to explain the subtleties of this fundamental topic. The key point that is often misunderstood, and that is here discussed at length, is that being efficient and being reasonable are two unrelated properties of evaluation strategies. A second focus of the paper is the relationship between standard and reasonable strategies.

Keywords: Lambda calculus, cost models, sharing, computational complexity, functional programming.

This work is part of a wider research effort, the COCA HOLA project [1].

1 Introduction

This unusual paper aims at explaining the problem of reasonable time cost models for the λ-calculus. *Reasonable* here is a technical word that essentially means polynomially equivalent to the time cost model of Turing machines. It is a fundamental problem, easily explainable to most computer scientists, and its solutions are technically demanding. Still, for a long time it attracted surprisingly little attention. The reason—we believe—is the fact that some of its facets are subtle, if not counterintuitive.

In the last few years the study of cost models for the λ-calculus has made considerable advances, starting in 2014 with the proof by Accattoli and Dal Lago that the leftmost-outermost (LO) evaluation strategy is reasonable [13]—a strategy is reasonable when its number of steps provides a reasonable cost model. On the one hand, that result strengthened results about weak strategies (in which evaluation

[1] Email: beniamino.accattoli@inria.fr

https://doi.org/10.1016/j.entcs.2018.10.003

does not enter into function bodies) by Blelloch and Greiner in 1995 [20], by Sands, Gustavsson, and Moran in 2002 [40], and those followed by combining the results by Dal Lago and Martini in 2009 in [34] and [26]. On the other hand, it counterbalanced Asperti and Mairson's 1998 result that Lévy's optimal strategy does not provide a reasonable cost model [15]. The advance required a new understanding of the problem, that in turn triggered a more systematic and still ongoing exploration, that clarified various points. This paper tries to sum them up and present them to a not-so-specialised audience.

Reasonable and efficient strategies. One of the motivations behind this paper is the fact that the problem is generally misunderstood, even by experts of the λ-calculus, as being about the efficient evaluation of λ-terms. In the λ-calculus evaluation is non-deterministic and different evaluation strategies may indeed behave very differently with respect to the number of steps. The situation is subtle: the λ-calculus is *confluent*, that is, the result is unique when it exists, therefore non-determinism is not about different results, but about different ways of obtaining the result. Moreover, some evaluation strategies may diverge even when the result exists, so that the way the result is computed is essential.

At first look then, the problem is about the choice of the evaluation strategy, and one would assume that a reasonable strategy must be an efficient one. Intuition, however, is misleading: *reasonable* and *efficient* are unrelated properties of strategies. Roughly, efficiency is a *comparative* property, it makes sense only if there are many strategies and one aims at comparing them. Being reasonable instead is a property of the strategy itself, independently of any other strategy, and it boils down to the fact that the strategy can be implemented with a negligible overhead.

Efficiency for reasonable strategies. Saying that *reasonable* and *efficient* are orthogonal properties of strategies is however slightly misleading, because it underestimates the value of the study of reasonable cost models. The study of efficiency, indeed, strikingly simplifies for reasonable strategies. For a reasonable strategy, one can take the number of its steps as a reasonable cost model because, roughly, every step can be considered to have cost 1, *i.e.* to be an atomic operation. Then two reasonable strategies can be compared for efficiency by simply comparing how many steps they take on the same term. When strategies are not known to be reasonable, instead, it is not clear how to compare them for efficiency, because their steps cannot be assumed to have cost 1. The idea that the efficiency of a strategy is given by its number of steps is indeed based on the hidden assumption that the strategy is reasonable.

Very few evaluation strategies have been proved reasonable, and there is at least one example of unreasonable strategy. As proved by Asperti and Mairson, a single step of Lévy's optimal strategy can have exponential cost (in the size of the initial term and the number of previous steps). For unreasonable strategies the natural way then is to compare how many steps *their implementations* take on the same term. Such a way of proceeding has however various drawbacks. First, it depends very much on the implementation of the fixed strategies, and so it hardly is a property of the strategy itself. Second, the cost is much harder to analyse,

because it depends on the many details of the fixed implementation. Last, it is an approach that somewhat clashes with the machine-independent character of the λ-calculus. There can be other ways of comparing unreasonable strategies, but far from the simplicity provided by reasonable strategies. For Lévy's optimal strategy, for instance, some works [14,17,30,31] have been able to shed some light on some aspects of its efficiency, and there are examples where it provides a considerable speed-up. Nonetheless, after almost 40 years since its introduction, it is still unclear whether in the general case it is efficient or not.

Reasonable optimisations. There is a further reason why the study of reasonable cost models turns out to be relevant for efficiency. Proving that a strategy is reasonable always requires some form of sharing, because the naive way of implementing β-reduction suffers of exponential overhead. Different strategies however require different forms of sharing and different optimisations. A close look shows that these techniques are general optimisation principles independent from the efficiency of the strategy, and composable in a modular way. In particular, some of them have been first developed for the inefficient case of LO evaluation, but they apply to more efficient strategies such as call-by-value or call-by-need. One of them, called *substituting abstractions on demand*—introduced without a name by Accattoli and Dal Lago in [13] and then studied more closely by Accattoli and Guerrieri in [12]—is essential for reasonable implementations of strong strategies, but—to the best of our knowledge—no tool based on the λ-calculus implements it. Therefore, no such tool, like for instance Coq or Isabelle, relies on a reasonable implementation: the study of cost models may thus impact on the theory of implementations, providing more efficient implementations of given strategies.

Standard and reasonable strategies. Another aim of this paper is to explain the surprising connection between the standardisation theorem of the λ-calculus and reasonable cost models. The connection has been pointed out by Accattoli and Dal Lago in [13], and it actually stems at the inception of their result. Roughly, it seems that—according to the current understanding of the problem—reasonable strategies have to be standard. This fact can be read in two ways. On the one hand, it provides a solid theoretical justification for taking the LO strategy—that is standard—as a sort of canonical reasonable strategy. On the other hand, it reveals the unexpected complexity-theoretic content of the standardisation theorem. Such a theorem is a rewriting tool introduced by Curry and Feis in 1958 to study the λ-calculus [23] but its scope has been extended and generalised to many other rewriting systems.

Related work. It may seem that the survey on the topic is unbalanced, because most cited papers on cost models are from the author and his coauthors. It is a matter of fact, however, that the topic has been, and still is, largely neglected by the λ-calculus community. A paper by Frandsen and Sturtivant [28] discussed reasonable cost model in 1991, but its content is nowadays outdated. Lawall and Mairson's [35] discusses [28], cost models, and optimal evaluation. For first-order rewriting, Dal Lago and Martini, and independently Avanzini and Moser, proved some quite general results through graph rewriting [34,16], itself a form of sharing. Other references are given in the rest of the paper, where pertinent.

2 Introducing Reasonable Cost Models

Turing machines are the standard model for complexity because their cost models are self-evident:

Time: the number of transitions;

Space: the maximum number of cells of the tape used during execution.

In order to study complexity in a different model X one has to first fix cost models for X. The basic requirement for cost models is to be *reasonable*: there should be a bidirectional simulation of Turing machines (or of another reasonable model) within a polynomially bounded overhead in time and constant factor overhead in space, as stated by Slot and van Emde Boas [42]. For instance, random access machines (RAM) are reasonable.

The importance of the concept relies on the fact that for reasonable systems the class of polynomial-time problems defined on X coincides with the one defined on Turing machines (TM, from now on): the class `P` then becomes *robust*, *i.e.* model-independent. The same is true for other (super-)polynomial classes such as `EXP` or `PSPACE`. More generally, *(in)efficiently computable* then becomes a concept similar to *(in)effectively computable*, and, unsurprisingly, it receives its own *thesis*, known alternatively as the *strong*, *extended*, *efficient*, *modern*, or *complexity-theoretic* Church(-Turing) thesis, or also as the *invariance thesis*: all models are reasonable.

It is natural to wonder whether the thesis can be strengthened even more, requiring for instance that all models are correlated by a *linear* overhead in time. It turns out that such a requirement is too strong, because simulations between models often require a non-linear overhead, for instance TM simulate RAM with a quadratic overhead, needed to simulate random access on a sequential tape. Said differently, sub-polynomial classes are not robust, which is one of the reasons why the class `P` is so relevant.

The model of interest, here, is the λ-calculus. Since the inception of computer science Turing machines were considered more effective than the λ-calculus because the cost models of the λ-calculus are not really evident. More precisely, there are natural cost models:

Time: the number of β-reductions, the computational steps of the λ-calculus, and

Space: the size of the largest λ-term produced by these steps.

There are however two problems with this naive view:

(i) *Strategy*: the λ-calculus is only *half*-deterministic: the result, if any, is unique, but there can be many ways of obtaining it, and some strategies may not terminate even when the result exists. Which strategy should be considered as providing a reasonable cost model? Is there a strategy that provides a canonical cost model, whatever that means?

(ii) *Atomicity of β*: β-reduction does not look like an atomic operation, since it

may increase the size of λ-terms at an exponential rate—this is a phenomenon called *size explosion*.

The serious issue concerns size explosion, and it suggests that the natural cost models are not reasonable. Via a detour through refined λ-calculi with sharing, however, it is possible to prove that the number of β-steps *is* a reasonable cost model for the λ-calculus (without sharing). To be precise, one should fix a specific dialect of the λ-calculus and an evaluation strategy, and then show that, somehow, the β-steps of the fixed strategy can be considered as atomic. The strongest result in the literature is that leftmost-outermost (LO) evaluation is a reasonable strategy for the (strong) λ-calculus [13].

Once it is understood how to circumvent size explosion, the strategy issue becomes more relevant, and it is somewhat the topic of this paper. We discuss here two aspects that seems to be the key features for reasonable strategies, namely *termination* and *the subterm property*, and in which sense the LO strategy is canonical.

The study of cost models is actually composed of two sub-problems, that are discussed at length in the next two sections, namely finding:

(i) a reasonable encoding of a reasonable model (usually TM) in the λ-calculus;

(ii) a reasonable encoding of the λ-calculus in a reasonable model (usually RAM).

Let us point out that we are going to consider only *unitary* cost models, that is, cost models that count β-steps according to some strategy, and simply count 1 for each one of them. This is the most natural way of thinking of a cost model for the λ-calculus, and provides easy tools to work with, for complexity analyses. Non-unitary reasonable cost model have been considered, by Dal Lago and Martini in [25] (namely, the cost of a step is the difference between the size of the redex and the size of the reduct), or suggested, by Lawall and Mairson [35] (based on the number or size of Lévy's labels).

While *space* plays a key role in our understanding of the problem, we do not address reasonable space cost models for the λ-calculus. As we shall explain, sharing is the key tool to deal with reasonable cost models. It is not, however, the right tool for space: every β-step (*i.e.* every time unit) requires a sharing annotation, implying that in such evaluation schemas space is always linear in the amount of time, that is the worst possible space behaviour, complexity-wise. As first pointed out by Schöpp [41], the geometry of interaction provides an alternative, interactive evaluation mechanism that is more parsimonious with respect to space, and that may provide a reasonable space cost model—the problem is however open.

3 From Turing Machines to the λ-Calculus

Most courses on the λ-calculus show how to represent partial recursive functions into it—see for instance the classic books by Barendregt [19] or Krivine [33]. The representation of TM can be traced back to the appendix of Turing's 1936 paper [43]. In contemporary literature this representation is however hard to find, but in itself it is not difficult. It is also not hard to show that TM can be *reasonably*

simulated in the λ-calculus. The reason is quite simple: TM are a first-order system, while the λ-calculus is higher-order, so it is expected that higher-order can simulate first-order reasonably.

At first sight, then, this direction seems not to be particularly exciting. Yet, here it lies the first counterintuitive aspect of the problem. Exactly because the higher-order world is much larger than the first-order one, it is possible to encode TM in a very simple fragment of the λ-calculus, what we like to call the *deterministic λ-calculus* $\Lambda_{\mathtt{det}}$. Let us introduce it.

The deterministic λ-calculus. The language of terms of $\Lambda_{\mathtt{det}}$ is a strict subset of the λ-calculus and is defined by:

$$\textsc{Terms} \quad t,s,u,r ::= v \mid tv \qquad\qquad \textsc{Values} \; v,v',v'' ::= \lambda x.t \mid x$$

Note that the right subterm of an application has to be a value, in contrast to what happens in the ordinary λ-calculus. Evaluation in $\Lambda_{\mathtt{det}}$ is also strictly less general than in the ordinary λ-calculus. Evaluation contexts are *weak*, *i.e.* they do not enter inside abstractions (meta-level substitution is noted $t\{x{\leftarrow}s\}$):

Weak evaluation contexts	Rule at top level	Contextual closure
$E ::= \langle\cdot\rangle \mid Ev$	$(\lambda x.t)s \mapsto_\beta t\{x{\leftarrow}s\}$	$E\langle t\rangle \to_{det} E\langle s\rangle$ if $t \mapsto_\beta s$

The deterministic λ-calculus is essentially the intersection of a continuation-passing style (CPS) calculus, where arguments can only be values, and of a weak calculus. Being CPS, call-by-name and call-by-value coincide, simply because all arguments are values. CPS calculi however usually rely on strong evaluation, while here we adopt the weak one.

The combination of the CPS and weak restrictions provides the following determinism property (proved by a simple induction on the structure of the term), that justifies the name of the calculus.

Lemma 3.1 (Determinism) *Let $t \in \Lambda_{\mathtt{det}}$. There is at most one $s \in \Lambda_{\mathtt{det}}$ such that $t \to_\beta s$, and in that case t is an application.*

In Accattoli and Dal Lago's [10] there is an encoding of TM in $\Lambda_{\mathtt{det}}$, due to Dal Lago, that is one of the steps to show that the *head λ-calculus* is reasonable. That encoding did not receive much attention, and it was confined to the technical report associated to [10], but in fact it has a very interesting consequence. This is why we reworked it to make it accessible to a more wider audience in the note [24]. The main result of that note is the following one.

Theorem 3.2 (Linear simulation, [24]) *Let Σ be an alphabet and $f : \Sigma^* \to \Sigma^*$ a function computed by a Turing machine $\mathcal{M}$ in time g. Then there is an encoding $\overline{\cdot}$ into $\Lambda_{\mathtt{det}}$ of Σ, strings, and Turing machines over Σ such that for every $s \in \Sigma^*$, $\overline{\mathcal{M}}\,\overline{s} \to^n_{det} \overline{f(s)}$ where $n = \Theta(g(|s|) + |s|)$.*

Let's now see the consequence of the existence of such an encoding.

Weak strategies. Essentially, all the weak strategies of the λ-calculus collapse

when restricted to the deterministic λ-calculus: terms have at most one weak redex and so all weak strategies coincide. Then all weak strategies provide a reasonable simulation of TM, and whenever a weak strategy (of the unrestricted λ-calculus) can be reasonably simulated on a reasonable model then it is reasonable, *independently of its efficiency*.

Let us now go back to the half-determinism of the λ-calculus. The difference in efficiency between strategies may be such that, on a given term, a strategy normalises while another one diverges. Despite its desperate inefficiency, if a diverging strategy can be implemented within a polynomial overhead then it is reasonable. And this is indeed possible: (weak) call-by-value is a reasonable strategy and yet it can diverge when (weak) call-by-name terminates. Typically, the following term $(\lambda x.y)((\lambda z.zz)(\lambda z.zz))$ normalises in one step in call-by-name but diverges in call-by-value (note that it does not belong to $\Lambda_{\mathtt{det}}$). Therefore, reasonable weak strategies need not being *terminating*, which is a quite counterintuitive fact.

Strong strategies. For strong strategies the question is subtler. The encoding of Theorem 3.2 provides a linear simulation of TM for every strong strategy reducing weak redexes or weak head redexes before any strong one. An example of the former is strong call-by-value that is usually obtained by iterating weak call-by-value under abstractions. One of the latter is leftmost-outermost evaluation.

For strong strategies, however, the behaviour with respect to termination is not irrelevant as in the weak case. The point is that the encoding relies on a fix-point operator. Fix-point operators always have, among the various possible evaluations, some diverging evaluations. In Theorem 3.2 divergence is avoided because the used fix-point diverges only if a particular strong evaluation is used, but as long as evaluation is weak everything works fine. There exist perpetual strong strategies, that is, strategies that diverge whenever possible, such as the maximal strategy, that instead always select a diverging path. These strategies would diverge on the encoding of *every* TM, at least with respect to the encoding of [24], and thus they do not allow to simulate them. It is not yet clear what is the exact termination requirement for strong strategies, nor whether there is a better encoding of TM that is termination-irrelevant also in the strong case. For what it is currently known, however, it seems that reasonable strong strategies cannot be perpetual. We shall come back to this point in Sect. 7, when discussing the standardisation theorem.

4 From λ-Calculus to Turing Machines

Encoding the λ-calculus into TM is the difficult part of showing that the λ-calculus is reasonable, because it requires to encode the higher-order world into the first-order one. Given a derivation $t_0 \to_\beta^n s$ in the λ-calculus, one wants to simulate it on TM in time polynomial in the following two key parameters:

(i) *Size of the input*: the size $|t_0|$ of the initial term, *i.e.* the number of constructors in t_0.

(ii) *Evaluation length*: the number of β-steps n.

The point is that there are *size exploding families of terms*, *i.e.* families of terms whose size is linear in n, that evaluate in n β-steps, and whose result has size exponential in n. The desired simulation, at first sight, seems impossible, because simply writing down the result requires time exponential in n (and $|t_0|$, which is itself linear in n). The way out is to switch to some refinements of the λ-calculus with sharing where evaluation produces compact, shared representations of ordinary normal forms, circumventing size explosion.

We are now going to dissect these wonderfully subtle points.

First of all, unfortunately, *size explosion affects every strategy of the λ-calculus.* This can be proved elegantly by showing that the deterministic λ-calculus already suffers of size-explosion. Since all weak strategies collapse on $\Lambda_{\mathtt{det}}$, they all are affected by size explosion. Let us show it.

First subtlety: deterministic, or strategy-independent size explosion. Consider the following size-exploding family (the family is obtained by applying t_n to the identity $I = u_0$), taken from [2]:

$$t_1 := \lambda x.\lambda y.(yxx) \qquad t_{n+1} := \lambda x.(t_n(\lambda y.(yxx))) \qquad u_0 := I \qquad u_{n+1} := \lambda y.(yu_nu_n)$$

Proposition 4.1 (Deterministic size explosion, [2]) *Let $n>0$. Then $t_nI \to^n_{det} u_n$. Moreover, $|t_nI| = O(n)$, $|u_n| = \Omega(2^n)$, t_nI is closed, and u_n is normal.*

The example also shows that $\Lambda_{\mathtt{det}}$ should not be considered as a first-order setting because, for as simple as it may look, it is already affected by size explosion, the typical disease of higher-order calculi.

The extension of the deterministic λ-calculus to strong evaluation is no longer deterministic. The given deterministic size exploding family, however, is essentially deterministic even with respect to strong evaluation. Technically speaking, t_nI has n parallel strong β-redexes, and so it does not look deterministic, but these redexes are independent: redexes cannot duplicate or erase each other (the diamond property holds for t_nI), nor there is any creation of redexes, so that all evaluations to normal form (including the Lévy optimal one) take exactly n steps, as in the weak case. Therefore, also strong and parallel strategies, and thus simply *all* strategies, suffer of size explosion.

Second subtlety: hidden assumption about space. Size explosion seems to imply that the number of β-steps cannot be a reasonable time cost model: the number of steps does not even account for the time to write down the result, that is exponentially bigger. While this is the natural way to read size explosion, this is also the wrong conclusion. In fact, the number of β-steps *is* a reasonable cost model, which means that there is a hidden wrong hypothesis in the natural reading. Let us postpone *how* to circumvent size explosion and let us focus on such a wrong hypothesis.

Essentially one is assuming that space in the λ-calculus is given by the maximum size of terms during evaluation, and since in sequential models time is greater or equal to space (because one needs a unit of time to use a unit of space), the time cost

of the size exploding family must be at least exponential. The wrong hypothesis then is that *space is the maximum size of the terms during evaluation.* It is not yet clear what accounts for space in the λ-calculus, however the study of time cost models made clear that space is *not* the size of the term.

Third subtlety: circumventing size explosion requires strategy-dependent sharing of subterms. All dialects of λ-calculus suffer from size explosion and they can usually be healed by switching to a formalism where β-reduction is decomposed in micro-steps and simulated using some form of sharing of subterms. The micro-step formalism can be a λ-calculus with explicit substitutions, an abstract machine, or some graph-rewriting mechanism. While the details certainly matters, these approaches are essentially all equivalent. We here give a sketch of the explicit substitution approach.

The basic idea is that the meta-level substitution $t\{x\leftarrow s\}$ used in the definition of β-redexes $(\lambda x.t)s \to_\beta t\{x\leftarrow s\}$ is delayed and an annotation for it, noted $t[x\leftarrow s]$ and called an *explicit substitution*, is introduced instead. Thus the language is given by (note that here we are in the general λ-calculus, and so applications have terms as right subterms):

$$\textsc{Terms with sharing} \quad t, s, u, r ::= x \mid \lambda x.t \mid ts \mid t[x\leftarrow s]$$

Explicit substitutions is the construct that accounts for sharing, and might be more familiar under the more readable but less compact form $\texttt{let}\ x = s\ \texttt{in}\ t$. The idea is that one can unfold them, obtaining the unshared underlying term. The unfolding operation is defined by:

$$\textsc{Unfolding}$$

$$x{\downarrow} := x \qquad (\lambda x.t){\downarrow} := \lambda x.t{\downarrow}$$

$$(t\ s){\downarrow} := t{\downarrow}\ s{\downarrow} \qquad (t[x\leftarrow s]){\downarrow} := t{\downarrow}\{x\leftarrow s{\downarrow}\}$$

It is easy to see that unfolding can magnify the term introducing an exponential explosion. Typically, define $t_0 := x_0$ and $t_{n+1} := t_n[x_n\leftarrow x_{n+1}x_{n+1}]$. Then t_n has size linear in n and a straightforward induction shows that $t_n{\downarrow} = \underbrace{x_n \dots x_n}_{2^n}$ has size exponential in n.

The definition of micro-step evaluation depends on the strategy under study—here we try to give a strategy-agnostic sketch. In general there is a notion of evaluation context E (whose precise definition depends on the strategy) that contains explicit substitutions and a single micro-step substitution rule:

$$\textsc{Micro-step substitution} \qquad E\langle x\rangle \to_{sub} E\langle t\rangle \qquad \text{if } E \text{ contains } [x\leftarrow t]$$

replacing the single occurrence of x in evaluation position (and not every occurrence of x as in β-reduction) with a new copy of t. Essentially, sharing is unfolded one variable occurrence at the time, and only when needed for evaluation to continue.

Let us now come back to size explosion problem. Size explosion is an inherent issue of meta-level substitution and boils down to the fact that each β-reduction may duplicate arguments of size exponential in the previous number of steps. Cir-

cumventing it via a micro-step system X usually requires the following points:

(i) *Reasonable single steps and the subterm property*: single micro-steps in X can be considered atomic in a suitable sense, that is, their cost does not explode. In all micro-steps systems we are aware of, this is a corollary of the fundamental *subterm property*: every micro-step duplication involves only a subterm of the initial term. According to our sketch, the terms t duplicated by rule $\to_{sub}$ are all subterms of the initial term. Note that deterministic size explosion implies that no strategy of the λ-calculus has the subterm property (because at least one step duplicates a subterm that is exponential in the size of the initial term). Forthcoming Sect. 7 discusses the subterm property in more depth.

(ii) *Reasonable simulation*: the simulation of β in X requires only a polynomial number of steps—in our sketch this translates as the fact that only a polynomial number of $\to_{sub}$ steps are needed to simulate the fixed strategy of the λ-calculus. Together with the subterm property then, one obtains that micro-step evaluation ends on a shared result of size polynomial in the number of β-steps, that is a compact representation of the result in the λ-calculus. This point can be tricky: sometimes (in strong and open dialects, to be discussed shortly) single steps are reasonable, and yet the simulation seems to require an exponential number of steps—refined $\to_{sub}$ rules are then needed.

(iii) *Reasonable representation*: compact representations (*i.e.* terms with sharing) may be managed (typically compared for equality) in reasonable (*i.e.* polynomial) time, without having to unfold the sharing (that would re-introduce an exponential blow-up). This fact is independent of the strategy and it has to be proven only once (for sharing as explicit substitutions or `let` expressions)—it has been done in Accattoli and Dal Lago's [10].

This schema is employed by all known proofs that a strategy is reasonable. Proving its key properties, namely the subterm property and the reasonable simulation, may however require very different techniques. Let us summarise the results in the literature:

(i) *Closed dialects*: for the weak λ-calculus with closed terms—modelling functional programming languages—standard implementations techniques like environment-based abstract machines provide reasonable simulations on RAM (a reasonable model). Morally, the strategy independent size exploding family $t_n I$ of Proposition 4.1 evaluates to

$$(\lambda y.(yxx))\underbrace{[x\leftarrow\lambda y.(yxx)]\dots[x\leftarrow\lambda y.(yxx)]}_{n-1}[x\leftarrow I] \tag{1}$$

that unfolds to the exponentially bigger result s_n but whose size is clearly linear in n. The trick is that in $\lambda y.(yxx)$ the two occurrences of x are under abstraction, but evaluation is weak and so it stops before reaching them—the duplications are avoided and size explosion is circumvented.

The first result for weak strategies is due to Blelloch and Greiner in 1995

[20] and concerns weak call-by-value evaluation. Similar results were then proved again, independently, by Sands, Gustavsson, and Moran in 2002 [40] who also addressed call-by-name and call-by-need, and by combining the results by Dal Lago and Martini in 2009 in [27] and [26], who also addressed call-by-name in [34]. Head (call-by-name) evaluation (that is not weak nor closed) is shown to be reasonable in [10] using essentially the same techniques, but recasted in Accattoli and Kesner's *linear substitution calculus* (LSC) [7], a simple and yet sharp framework refining a calculus by Robin Milner [39] with ideas coming from linear logic. The known results about weak strategies have further been dissected and refined via the LSC by Accattoli, Barenbaum, Mazza, and Sacerdoti Coen in [5,8].

Naive implementations of closed calculi such as Krivine abstract machine have an overhead quadratic in the number of β-steps and linear in the size of the initial term. With simple optimisations and an attentive choice of the data-structures the overhead can be lowered to linear in the number of β-steps and logarithmic in the size of the initial term—see Accattoli and Barras' [6].

(ii) *Strong dialects*: for *strong* strategies—at work in proof assistant engines—quite more effort and care are required because—as we hinted at before—ordinary abstract machines (such as Cregut's machine [21,22]) do provide reasonable single steps, but they do not provide reasonable simulations. The point is that micro-step evaluation no longer stops on abstractions, and so the number of duplications explodes. A second-level of sharing refining $\rightarrow_{sub}$, called *useful sharing* and to be overviewed in Sect. 5, is necessary to obtain reasonable micro-step simulations. The first such semantics has been introduced by Accattoli and Dal Lago in 2014 (and published in a journal in 2016 [13]) for the LO strategy, by relying on the LSC. Then Accattoli gave a new proof using a sophisticated abstract machine in [3].

The overhead for the strong case has a higher complexity than in the weak case. It is linear—and not logarithmic—in the size of the initial term, because even simply recognising that a term is normal requires to go through the whole term, and so cannot be done in logarithmic time (in the closed case it is a constant-time operation—just check if the first constructor is an abstraction). In the cases provided in the literature the dependency on the number of β-steps is instead *quadratic*. How to obtain a linear dependency from β-steps has been studied at length for the closed case by Accattoli and Sacerdoti Coen [8] and the technique seems to scale smoothly to strong evaluation—it was omitted from the cases in the literature only for the sake of simplicity.

(iii) *Open dialects*: by trying to better understand the result for strong evaluation, it turned out that there is an intermediate setting between closed and strong calculi, that is weak evaluation with *open* terms [29,9,11,12]. The key point is that it is possible to describe strong evaluation as an iteration under abstraction of the open case, while this is not possible with the closed case. Interestingly, for the study of cost models open dialects are strictly harder to implement than closed dialects and strictly simpler than strong ones, in the sense that

they require an optimisation that is superfluous in the closed case, and they do not require an optimisation that is needed in the strong case. We will hint at these optimisations in Sect. 5.

The key feature of the open case is a form of size explosion that is not possible in the closed setting. The easiest way to show it is with respect to right-to-left weak evaluation, noted here $\to_{rlw}$. Define:

$$t_0 := y \qquad t_{n+1} := (\lambda x.xx)t_n \qquad\qquad s_0 := y \qquad s_{n+1} := s_n s_n$$

Note that the family $\{t_n\}_{n\in\mathbb{N}}$ is open because of the first term $t_0 = y$. Then:

Proposition 4.2 (Open size explosion, [9]) *Let $n \in \mathbb{N}$. Then $t_n \to^n_{rlw} s_n$, moreover $|t_n| = O(n)$, $|s_n| = \Omega(2^n)$, and s_n is normal and not an abstraction.*

The open case is relevant for call-by-value evaluation, where Plotkin's operational semantics for the closed case is not adequate. An implementation of open call-by-value was first given by Grégoire and Leroy in [29], but without the needed optimisation to make it reasonable. Accattoli and Sacerdoti Coen provided the first such implementation in [9] by adapting useful sharing from the call-by-name case, and obtaining an implementation of overhead linear in both the number of β-steps and the size of the initial term. Accattoli and Guerrieri further simplified the machine in [9] and also showed that the open case is strictly simpler than the strong one. Last, in [11] Accattoli and Guerrieri show that different open call-by-value calculi share the same cost model.

Open call-by-name and open call-by-need have never been treated in the literature.

Fourth subtlety: new strong normal forms. Avoiding size explosion requires sharing of subterms and compact representations of normal forms. The difficulty of the strong case arises from the difficulty of stopping on a compact representation of a full normal form, given that now the substitution process can act on the whole term. Said differently, adding sharing (that is, turning to micro-step evaluation) without changing the notion of normal form would still compute the full normal form, and not a compact representation of it. The unavoidable solution is provided by Accattoli and Dal Lago's useful sharing, to be overviewed in Sect. 5, but let us here stick to an important an yet usually overlooked consequence of their result: there is *a new notion of normal form*, given by terms with sharing t whose unshared form $t{\downarrow}$ is normal, that can be computed and manipulated efficiently.

The subtlety of the consequence is better understood in logical terms, via the Curry-Howard isomorphism. Logically speaking, it means that normalisation does not need to compute cut-free proofs, but only proofs that unfold to cut-free proofs—this is a quite radical change of perspective. Linear logic provides a more precise understanding, since it provides a logical explanation of what does it mean to unfold a proof. Sharing and the substitution process are indeed taken care of by the exponential fragment of linear logic. The subtlety can then be reformulated as follows: proofs with exponential cuts whose exponential development is cut-free (and potentially exponentially bigger) can be accepted as normal (and computed

and compared efficiently).

Essentially, Accattoli and Dal Lago's result has been presented and received as a result about cost models. It is based, however, on a new point of view on what is a normal form, and such a change of perspective has mostly gone unnoticed.

Fifth subtlety: inefficiency does not help with reasonable implementations. Size explosion affects every strategy, and so it does not depend on the efficiency of the strategy. Therefore, providing a reasonable implementation is non-trivial also for inefficient strategies.

A degenerate but interesting case study is given by the maximal strategy. It is a perpetual strategy, *i.e.* it diverges whenever possible, and moreover when it terminates it takes the maximum number of steps to reach the normal form. Thus, it is as inefficient as a strategy can be, and according to the end of Sect. 3 it does not seem to be a reasonable strategy because it does not allow to simulate TM.

Nonetheless, it admits a reasonable implementation, showed in the note [4]. Such an implementation requires exactly the same technique (namely useful sharing) used for the LO strategy (the only other strong strategy for which a reasonable implementation is known), despite the strategy being less efficient, and even not providing a reasonable cost model.

5 A Glimpse of Useful Sharing

Here we attempt to explain how and why the technique of useful sharing, used for the LO strategy in [13,3], for open call-by-value in [9] and for the maximal strategy in [4], induces reasonable implementations.

Implementations always decompose β-steps in micro-steps. In particular they decompose the substitution process: substitutions are delayed via the use of sharing/explicit substitutions and variable occurrences are replaced one at a time, only when they come in evaluation position. To be reasonable, one should avoid micro substitution steps not contributing somehow to β-redexes, *i.e.* one should avoid *useless* variable replacements and perform only the *useful* ones.

Useful and useless replacements. According to the sketch given in Sect. 4, let $E\langle x\rangle$ be a variable occurrence x in an evaluation context E having an explicit substitution $[x\leftarrow t]$ for x, and consider the replacement of x with t. When is it useful for β-redexes? Two cases:

(i) *Copy of a redex*: if t contains a β-redex, then it is useful to make a copy of t and put it in evaluation position in $E\langle t\rangle$ (forget that it would be better to reduce t before substituting it, this is an optimisation that is not essential).

(ii) *Creation of a redex*: it might be that t is normal but that its substitution creates a new redex in $E\langle t\rangle$. Precisely, this happens when $t = \lambda y.s$ and E is applicative, that is $E = F\langle\langle\cdot\rangle u\rangle$, so that the replacement creates the redex $E\langle t\rangle = F\langle(\lambda y.s)u\rangle$.

Dually, a replacement is *useless* when

(i) *Neutral*: t is normal and it is not an abstraction—we say that t is neutral. In

this case copying t is of no use, because no redexes can be copied nor created;

(ii) *Normal abstraction in non-applicative context*: the term to substitute is an abstraction—thus in principle it could create a redex—but the context does not provide an argument, and so no redexes are created by the replacement.

Useful sharing is the avoidance of useless replacements. Note that it is a micro-step concept: a meta-level substitution of t for x may be useful for some occurrences of x and useless for others.

Useful sharing and closed dialects. Closed settings (weak evaluation plus closed terms) do not need useful sharing because therein replacements are always useful, thanks to the heavy restrictions.

Useful sharing and open dialects. In open setting (weak but with open terms), reasonable implementations must avoid the substitution of neutral terms (*i.e.* uselessness of the first kind) but can substitute abstractions whenever. Note indeed that in the open size exploding family (Proposition 4.2) the explosion is given by the duplication of neutral terms.

Useful sharing and strong dialects. Reasonable implementations of strong evaluation must also avoid useless replacements of the second kind, with an optimisation that is sometimes called *substituting abstractions on-demand* in [12]. Indeed, note that the key restriction for avoiding substitutions on x under abstractions in (1) at page 10 is the fact that those occurrences of x are not applied.

Such an optimisation is necessary in order to obtain reasonable implementations of strong strategies, and yet no implementation of a strong strategy (typically in proof assistants) we are aware of implements it. Therefore, the theoretical study of reasonable cost models has concrete downfalls on the implementation of higher-order tools. See Accattoli and Sacerdoti Coen's [9] and Accattoli and Guerrieri's [12].

Useful sharing is trickier, in fact. In this introduction to useful sharing we omitted a key, technical ingredient. In micro-step settings, where substitutions are delayed, it is not so easy to understand if a variable is replaced by a normal or by a neutral term, which is what is required to implement the two cases of useful sharing. The reason is that the term t that is meant to replace x might be decomposed in small pieces spread in many delayed substitutions in the environment, and the redex(es) in t may become visible only when these pieces are put together. This is why abstract machines implementing useful sharing often use labels on environment entries to trace whether, when later on *pieces will be put together* the entry will have a redex, will be an abstraction, or will be neutral. This is the approach followed in [9,3,4].

6 Efficiency and Reasonable Cost Models

Optimal sequential evaluation. It is well-known that the optimal *one-step* evaluation strategy of the λ-calculus is not recursive, see Barendregt's book [19]. And here lies another subtlety of the problem: a priori, *non-recursive* does not imply

unreasonable, as there may be a recursive sub-optimal algorithm for evaluation that is polynomial in the steps of the optimal strategy and that does not simulate the strategy itself—the existence of such an algorithm, however, is unlikely. Even more generally, note that a cost model is a measure, not a computational problem—knowing that the problem is not recursive does not immediately implies that the measure connected to it is unreasonable. Let us also point out that the proof in [19] that the optimal strategy is not recursive does not provide hints on how to show that it is unreasonable, which is an open problem.

Optimal parallel evaluation. On the other hand, Lévy's optimal *parallel* evaluation [36] is decidable, even if it takes a number of steps that is less or equal to the optimal one-step strategy. In 1998, Asperti and Mairson [15] have shown that Lévy's optimal evaluation is unreasonable, which is also the only known negative result about reasonable strategies. The key point is that Lévy's notion hides the complexity of its implementation in the cleverness of its definition. It shares too much, collapsing the complexity of too many steps into a single one, making the number of optimal steps an unreliable measure. Let us stress, once more, that the fact that the optimal strategy is unreasonable does not imply that it is inefficient, nor that its implementations are inefficient. Simply, the number of optimal steps is an unreliable metric, but it might be that optimal implementations are reasonable, if measured with respect to a reasonable cost model such as the number of LO steps to normal form. This question however is open. As pointed out in the introduction, the difficulty lies in the fact that, since the parallel optimal strategy is unreasonable, one cannot rely on counting steps in the λ-calculus, but it has to study closely the systems implementing it, that are much more involved than the ordinary λ-calculus.

There are some variants of linear logic inducing fragments of the λ-calculus with bound complexities (elementary, polynomial) with respect to ordinary evaluation techniques. These fragments provide a tool to better understand Lévy's optimality because their optimal implementation is much simpler (no need of the so-called *oracle*). In [14], Asperti, Coppola, and Martini prove that Lévy's optimality strategy is unreasonable also in the elementary fragment. Baillot, Coppola, and Dal Lago in [17] and Guerrini and Soleri in [31], showed that, on the other hand, evaluating this fragments via Lévy's optimality stays in the expected complexity of the fragment, proving then that unreasonable does not mean inefficient.

Sharing of subterms and sharing of computations. Let us compare Asperti and Mairson's result to Accattoli and Dal Lago's on a high level. The LO strategy, is a sort of *maximally unshared* normalising strategy, where redexes are duplicated whenever possible and unneeded redexes are never reduced, somehow dually with respect to optimal derivations. The reasonable implementation of the LO strategy employs useful sharing but it is important not to confuse two different levels of sharing: useful sharing shares *subterms*, but not *computations*, while Lévy's optimal derivations do the opposite. It is exactly because useful sharing does not share computations that it has the *subterm property*, the key point to prove that it is reasonable. In [13] the subterm property is shown to be deeply connected—surprisingly—to the standardisation theorem for the linear substitution calculus

and the concept of exponential box in linear logic, see also forthcoming Sect. 7.

Reasonable sharing of computations. Sharing of subterms and sharing of computations are not incompatible, if one does not go as far as Levy's optimal evaluation. In the weak realm, call-by-value and call-by-need strategies can indeed be seen as strategies sharing computations, when compared to call-by-name (that is the weak analogous of LO evaluation). For instance, on the open size exploding family of Proposition 4.2 call-by-value and call-by-need take exponentially less steps than call-by-name. And both call-by-value and call-by-need have the subterm property, allowing for sharing of subterms. Moreover, their micro-step implementation is very efficient, having an overhead that is *linear* in the number of β-steps (and logarithmic in the size of the initial term). Therefore, we *do* know strategies that are both *reasonable* and *efficient*. As already discussed in Sect. 4, there also are efficient implementations of open call-by-value. Similar techniques can also handle open call-by-need, but this case study has not been published (yet).

How to show that *strong* call-by-value and call-by-need are also reasonable is an active topic of research. The tools to deal with strong call-by-value have essentially been developed in [9,12], and we expect them to also adapt to strong call-by-need, whose operational semantics is finding its way into a published form just now [18]. The subterm property still holds, as well as the schema for proving that these strategies are reasonable. What is not immediate is the handling of the many technical details of implementing strong evaluation and useful sharing for call-by-value/need.

7 Standard Derivations and the Subterm Property

As already repeatedly pointed out, the LO strategy is reasonable. We want here to explain some further points about it and provide an abstract view on why it is reasonable.

Standard derivations and the LO strategy. The LO strategy is a *standard* strategy in the sense of the *standardisation theorem*, a key result in the theory of λ-calculus, first proved in 1958 by Curry and Feys [23]. Let us briefly recall what does it mean. The λ-calculus is non-deterministic, so that there are many different ways of computing. Is there a canonical way of computing? Luckily yes, the standard one. Roughly, a derivation (*i.e.* a sequence of rewriting steps) is standard when it selects redexes from left to right inside the term. The standardisation theorem states that given a derivation $d : t \to_\beta^* s$ it is always possible to rearrange it into a standard derivation $e : t \to_\beta^* s$ that computes the same result but selecting redexes from left to right—this is the *completeness* property of standard derivation. Let us point out that such a standardisation process is a subtle transformation that can change the length of d, by both duplicating and erasing steps, so that the length of e is in general not the same as the one of d.

The standardisation theorem says that derivations can be rearranged. It is however also easy to build standard derivations from scratch, it is enough to always pick the leftmost redex. Therefore, the LO strategy is the prototypical strategy that

always produces standard derivations, and it has the additional property of being *normalising*, that is, it reaches a normal form whenever it exists—this is the content of the normalisation theorem of the λ-calculus. Completeness and normalisation allow to consider the LO strategy as a sort of canonical evaluation strategy for the λ-calculus.

Standardisation and diagrams. A more abstract, diagrammatic view on standardisation has been first proposed by Klop in his PhD thesis [32] and then explored systematically by Melliès [37,38]. For us, the relevant idea is that the rearranging process behind standardisation can be understood as the process of anticipating *duplications* and *erasures*, and formulated as a 2-dimensional rewriting $\triangleright$ on derivations. Let us illustrate the concept with two examples. Consider the following rearrangements of non-standard into standard derivations (where $I = \lambda x.x$ is the identity combinator):

(2)

In the left diagram, the permutation of steps causes a *duplication* of the redex Iy, while in the right diagram the same redex is *erased*. Let us first discuss the right diagram: standard derivation always prepone erasing steps, and so never reduce in parts of the terms that will be erased. This is the property beyond the normalising behaviour: standard derivations do not bother reducing redexes that are not needed to reach the normal form—*i.e.* they only reduce needed redexes. The left diagram has instead another, less famous consequence. By preponing duplications, standard derivations are the *longest* needed derivations. In particular, the LO derivation is the longest needed derivation to normal form (for a normalising term). Therefore, the LO strategy is highly inefficient, and it provides a sort of worst case scenario in the realm of the normalising strategies of the λ-calculus. This inefficiency, however, is more precious than it may seem at first sight.

(Linear) standard derivations and the subterm property. By preponing duplications, and reducing from left to right, whenever a redex $(\lambda x.u)r$ is reduced along a LO derivation $d : t \to_\beta^* s$ one has that r has not been reduced by the previous steps in d (otherwise the derivation would not be LO). This property is very close to the subterm property mentioned for reasonable steps in Sect. 4 but it is not quite it: in fact, r is not necessarily a subterm of the initial term t, as it is required by the subterm property, it is only a subterm of t *where other subterms of* t *may have been substituted*—let us call this phenomenon the *iterated subterm property*, and give an example. Consider su in the standard derivation:

$$(\lambda y.((\lambda x.xx)(yu)))s \to_\beta (\lambda x.xx)(su) \to_\beta (su)(su)$$

The duplicated term su is not a subterm of the initial term, but it is obtained by substituting s for y in yu, where yu and s are both subterms of the initial term. The problem is that β-reduction is too coarse to have the subterm property, it only

has the iterated subterm property. The iterated subterm property unfortunately allows to chain substitutions of subterms of the initial term into subterms of the initial term, leading to size explosion, that is, to an exponential gap between the number of steps and the size of the duplicated objects. The plain subterm property, instead, avoids it.

A special micro-step systems is the (already mentioned) *linear substitution calculus* (LSC). What makes the LSC apart from other micro-step systems is that it admits a standardisation theorem akin to the one for the λ-calculus, as proved by Accattoli, Bonelli, Lombardi, and Kesner in [7]. Moreover, in [13] Accattoli and Dal Lago proved that standard derivations in the LSC have the subterm property, *i.e.* that every subterm duplicated along a standard derivation is a subterm of the initial term. This fact implies that standard strategies in the LSC have reasonable single steps (in the sense discussed in Sect. 4), and provide a theoretical understanding of why the LO strategy is reasonable and why one needs to pass through a micro-step system. Moreover, it establishes a highly unexpected connection between the standardisation theorem and the study of cost models.

For the sake of completeness, let us mention that the LO strategy of the LSC has reasonable steps, but it does not provide a reasonable simulation of the LO strategy of the λ-calculus, because there is an explosion of the number of steps—size explosion comes back disguised as *length explosion*. Such a length explosion can be then avoided by employing useful sharing, as briefly described in Sect. 5. See [13] for more details.

Erasure, maximality, and reasonable cost models. Let us come back to the two diagrams in (2). We have seen the fundamental role of the duplicating diagram in connection with the subterm property. It is natural to wonder what is the role of the second, erasing diagram for the study of cost models—or, equivalently, what is the role of the normalising property. Let us call *half-standard* a derivation that is standard with respect to the duplicating swaps but not with respect to the erasing swaps. The maximal strategy in Sect. 3 produces half-standard derivations and it does not (seem to) provide a simulation of TM. Thus, erasing swaps are also essential for reasonable derivations.

Standard and reasonable. Summing up, there is a nice, theoretical explanation of the relationship between standard strategies and reasonable strategies, or at least of our current understanding of the problem: standard derivations have *two* bricks, duplicating and erasing swaps, while a strategy is reasonable if there are *two* simulations, and their relationship is as follows:

- *Erasing swaps* avoid the *divergency issue* related to perpetual strong evaluation, guaranteeing a reasonably simulation of TM.
- *Duplicating swaps* avoid *size explosion*, by guarantee the subterm property, that is the main ingredient to provide a reasonable simulation of the strategy on RAM.

The theory of standard derivation then provides an explanation of why the LO strategy is reasonable and somewhat provides a canonical notion of cost model.

Let us stress a final subtlety. The insight from standardisation does not mean

that call-by-name is better than call-by-value or call-by-need (because the LO strategy is call-by-name). Here it lies the widespread ambiguity between *calculi* and *strategies*. In the call-by-name strong λ-*calculus* (that is nothing but the ordinary λ-calculus) the typical standard strategy is LO evaluation, and it provides a reasonable cost model. Call-by-value or call-by-need *strategies* can be proved standard in larger call-by-value or call-by-need *calculi* and with respect to order on redexes other than the leftmost-outermost, thus fitting in the same framework of *standard = reasonable* that we just suggested.

The current understanding of the higher-order time cost model problem therefore suggests that the reasonable strategies of a calculus are the standard ones.

Acknowledgement

This work has been partially funded by the ANR JCJC grant COCA HOLA (ANR-16-CE40-004-01).

References

[1] Accattoli, B., *COCA HOLA*, https://sites.google.com/site/beniaminoaccattoli/coca-hola (2016).

[2] Accattoli, B., *The complexity of abstract machines*, in: *WPTE@FSCD 2016*, 2016, pp. 1–15.
URL https://doi.org/10.4204/EPTCS.235.1

[3] Accattoli, B., *The Useful MAM, a Reasonable Implementation of the Strong λ-Calculus*, in: *WoLLIC 2016*, 2016, pp. 1–21.
URL http://dx.doi.org/10.1007/978-3-662-52921-8_1

[4] Accattoli, B., *The Maximal MAM, a Reasonable Implementation of the Maximal Strategy*, CoRR **abs/1711.10301** (2017).
URL https://arxiv.org/abs/1711.10301

[5] Accattoli, B., P. Barenbaum and D. Mazza, *Distilling abstract machines*, in: *ICFP 2014*, 2014, pp. 363–376.
URL http://doi.acm.org/10.1145/2628136.2628154

[6] Accattoli, B. and B. Barras, *Environments and the complexity of abstract machines*, in: *PPDP 2017*, 2017, pp. 4–16.
URL http://doi.acm.org/10.1145/3131851.3131855

[7] Accattoli, B., E. Bonelli, D. Kesner and C. Lombardi, *A nonstandard standardization theorem*, in: *POPL*, 2014, pp. 659–670.

[8] Accattoli, B. and C. S. Coen, *On the value of variables*, in: *WoLLIC 2014*, 2014, pp. 36–50.
URL http://dx.doi.org/10.1007/978-3-662-44145-9_3

[9] Accattoli, B. and C. S. Coen, *On the relative usefulness of fireballs*, in: *LICS 2015*, 2015, pp. 141–155.
URL http://dx.doi.org/10.1109/LICS.2015.23

[10] Accattoli, B. and U. Dal Lago, *On the invariance of the unitary cost model for head reduction*, in: *RTA*, 2012, pp. 22–37.

[11] Accattoli, B. and G. Guerrieri, *Open call-by-value*, in: *APLAS 2016*, 2016, pp. 206–226.
URL http://dx.doi.org/10.1007/978-3-319-47958-3_12

[12] Accattoli, B. and G. Guerrieri, *Implementing open call-by-value*, in: *FSEN 2017, Tehran, Iran, April 26-28, 2017, Revised Selected Papers*, 2017, pp. 1–19.
URL https://doi.org/10.1007/978-3-319-68972-2_1

[13] Accattoli, B. and U. D. Lago, *(Leftmost-Outermost) Beta-Reduction is Invariant, Indeed*, LMCS **12** (2016).
URL http://dx.doi.org/10.2168/LMCS-12(1:4)2016

[14] Asperti, A., P. Coppola and S. Martini, *(optimal) duplication is not elementary recursive*, in: *POPL 2000*, 2000, pp. 96–107.
URL http://doi.acm.org/10.1145/325694.325707

[15] Asperti, A. and H. G. Mairson, *Parallel beta reduction is not elementary recursive*, in: *POPL*, 1998, pp. 303–315.

[16] Avanzini, M. and G. Moser, *Closing the gap between runtime complexity and polytime computability*, in: *RTA 2010*, 2010, pp. 33–48.
URL http://dx.doi.org/10.4230/LIPIcs.RTA.2010.33

[17] Baillot, P., P. Coppola and U. Dal Lago, *Light logics and optimal reduction: Completeness and complexity*, Inf. Comput. **209** (2011), pp. 118–142.
URL https://doi.org/10.1016/j.ic.2010.10.002

[18] Balabonski, T., P. Barenbaum, E. Bonelli and D. Kesner, *Foundations of strong call by need*, PACMPL **1** (2017), pp. 20:1–20:29.
URL http://doi.acm.org/10.1145/3110264

[19] Barendregt, H. P., "The Lambda Calculus – Its Syntax and Semantics," Studies in logic and the foundations of mathematics **103**, North-Holland, 1984.

[20] Blelloch, G. E. and J. Greiner, *Parallelism in sequential functional languages*, in: *FPCA*, 1995, pp. 226–237.

[21] Crégut, P., *An abstract machine for lambda-terms normalization*, in: *LISP and Functional Programming*, 1990, pp. 333–340.

[22] Crégut, P., *Strongly reducing variants of the Krivine abstract machine*, Higher-Order and Symbolic Computation **20** (2007), pp. 209–230.

[23] Curry, H. and R. Feys, "Combinatory Logic," Number 1 in Studies in logic and the foundations of mathematics, North-Holland Publishing Company, 1958.

[24] Dal Lago, U. and B. Accattoli, *Encoding Turing Machines into the Deterministic Lambda-Calculus*, CoRR **abs/1711.10078** (2017).
URL https://arxiv.org/abs/1711.10078

[25] Dal Lago, U. and S. Martini, *An invariant cost model for the lambda calculus*, in: *CiE 2006*, 2006, pp. 105–114.
URL http://dx.doi.org/10.1007/11780342_11

[26] Dal Lago, U. and S. Martini, *Derivational complexity is an invariant cost model*, in: *FOPARA 2009*, 2009, pp. 100–113.

[27] Dal Lago, U. and S. Martini, *On Constructor Rewrite Systems and the Lambda-Calculus*, in: *ICALP (2)*, 2009, pp. 163–174.

[28] Frandsen, G. S. and C. Sturtivant, *What is an efficient implementation of the λ-calculus?*, in: *FPCA 1991*, 1991, pp. 289–312.
URL http://dx.doi.org/10.1007/3540543961_14

[29] Grégoire, B. and X. Leroy, *A compiled implementation of strong reduction*, in: *(ICFP '02).*, 2002, pp. 235–246.
URL http://doi.acm.org/10.1145/581478.581501

[30] Guerrini, S., T. Leventis and M. Solieri, *Deep into optimality – complexity and correctness of sharing implementation of bounded logics*, Proceedings of the DICE 2012 Workshop.

[31] Guerrini, S. and M. Solieri, *Is the optimal implementation inefficient? elementarily not*, in: *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, 2017, pp. 17:1–17:16.
URL https://doi.org/10.4230/LIPIcs.FSCD.2017.17

[32] Klop, J. W., "Combinatory Reduction Systems," Phd thesis, Utrecht University (1980).

[33] Krivine, J.-L., "Lambda-calculus, types and models," Ellis Horwood series in computers and their applications, Ellis Horwood, 1993.
URL https://books.google.fr/books?id=brWEAAAAIAAJ

[34] Lago, U. D. and S. Martini, *On constructor rewrite systems and the lambda calculus*, Logical Methods in Computer Science **8** (2012).
URL http://dx.doi.org/10.2168/LMCS-8(3:12)2012

[35] Lawall, J. L. and H. G. Mairson, *Optimality and inefficiency: What isn't a cost model of the lambda calculus?*, in: *(ICFP '96*, 1996, pp. 92–101.
URL http://doi.acm.org/10.1145/232627.232639

[36] Lévy, J.-J., *Réductions correctes et optimales dans le lambda-calcul*, Thése d'Etat, Univ. Paris VII, France (1978).

[37] Melliès, P.-A., "Description Abstraite de système de réécriture," PhD thesis, Paris 7 University (1996).

[38] Melliès, P.-A., *Axiomatic rewriting theory I: A diagrammatic standardization theorem*, in: *Processes, Terms and Cycles*, Lecture Notes in Computer Science **3838** (2005), pp. 554–638.

[39] Milner, R., *Local bigraphs and confluence: Two conjectures*, Electr. Notes Theor. Comput. Sci. **175** (2007), pp. 65–73.

[40] Sands, D., J. Gustavsson and A. Moran, *Lambda calculi and linear speedups*, in: *The Essence of Computation*, 2002, pp. 60–84.

[41] Schöpp, U., *Space-efficient computation by interaction*, in: *CSL 2006*, 2006, pp. 606–621.
URL https://doi.org/10.1007/11874683_40

[42] Slot, C. F. and P. van Emde Boas, *On tape versus core; an application of space efficient perfect hash functions to the invariance of space*, in: *STOC 1984*, 1984, pp. 391–400.
URL http://doi.acm.org/10.1145/800057.808705

[43] Turing, A. M., *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society **2** (1936), pp. 230–265.
URL http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf