Deep into Optimality
Complexity and correctness of sharing implementation of bounded logics

(Extended abstract)

Stefano Guerrini Thomas Leventis Marco Solieri

LIPN, Université Paris 13 Nord

Introduction

Sharing graphs are a tool for the implementation of graph rewriting systems whose rules may require the duplication, the displacement or the erasing of a whole subgraph, in the style of the β-rule of λ-calculus or of the exponential rules of linear logic [1, 8]. Sharing graphs were introduced by Lamping [10] as an implementation of Lévy's optimal reduction for λ-calculus [11]. Later, Gonthier, Abadi, and Lévy [6, 7] showed that sharing graphs could be understood in terms of Girard's Geometry of Interaction and that they could be used to implement cut-elimination of linear logic proof nets.

Sharing graphs are a key tool for the analysis of the cost of λ-calculus β-reduction. In particular, they have allowed to prove that the cost of β-reduction can be decomposed in two parts: the cost of sharing β-rules, which correspond to the cost of atomic steps which contract in parallel a whole set of β-redexes and can be brought to be equal to the number of Lévy's redex families, and the cost of the structural rules required for the actual replacement of the variable of an applied function with the actual value of its argument.

Asperti and Mairson [2] have shown that, in the case of the simply typed λ-calculus, the cost of reduction can be moved to the cost of the structural part of reduction, showing at the same time that the number of Lévy's redex families of a term cannot be taken as a cost model for λ-calculus. Such a result has been frequently interpreted as an evidence of the inefficiency of sharing graphs implementations; nevertheless, since it rests on a lower bound result for the β-reduction of simply typed λ-terms, it does not say anything about the comparison of sharing graph reduction w.r.t. to other standard implementations.

Indeed, the analysis of the complexity of sharing reduction is far from complete, and the only comparative result in the literature considers the case of Girard's bounded linear logics [5] only; in particular, the affine systems EAL (Elementary Affine Logic) and LAL (Light Affine Logic) presented by Asperti and Roversi [3]. The restrictions that these logics require on the nesting of exponential boxes allow a much simpler implementation of the matching of sharing (and unsharing) operators required for the implementation of sharing reduction. In the general case, one needs a dynamic naming of sharing operators, which is usually implemented by additional operators and which may lead to an important additional cost. In the case of EAL and LAL instead, we can exploit the fact that the box nesting of (the residuals of) graph operators is invariant along the reduction; the matching of sharing operators can then be implemented via a static naming which does not need the introduction of any additional operator or reduction rule, and does not introduce then any additional cost.

The cut elimination of the proofs of EAL and LAL provide an implicit characterization of the Kalmar-elementar and of the polynomial complexity classes, respectively. By a careful analysis of reduction based on Geometry of Interaction, Baillot, Coppola, and Dal Lago [4] have given a semantical proof of soundness of sharing reduction w.r.t. the complexity bounds of EAL and LAL. In other words, they have proven that the corresponding sharing implementations respect the asymptotic Kalmar-elementar or polynomial time constraint.

Complexity and soundness

We propose, for the first time, a complete and stronger comparison between the cost of cut elimination for EAL and LAL proof nets, denoted by CE, and the cost of their corresponding sharing graphs implementation, denoted by SG.

We compare then the functions $\mathcal{C}_{\mathsf{CE}}$ and $\mathcal{C}_{\mathsf{SG}}$ which give the cost of a reduction sequence in CE and SG, respectively. As usual, $\mathcal{C}_{\mathsf{CE}}$ is defined as the sum of the variations in the size of the proof net after each reduction step. Because of the atomicity of the reduction steps of SG, for $\mathcal{C}_{\mathsf{SG}}$ instead, it is enough to account the length of the reduction sequence.

For any couple of corresponding reductions, we show that the cost of the sharing reduction of a proof net T is bound by the cost of the corresponding proof net cut elimination reduction. More formally, for every optimal sharing reduction $\sigma_o : T \longrightarrow^* N'$ that reduces the proof net T to the optimal sharing representation N' of its normal form N, if $\rho : T \longrightarrow^* N$ is the corresponding cut elimination reduction, then

$$\mathcal{C}_{\mathsf{CE}}(\rho) \geq \mathcal{C}_{\mathsf{SG}}(\sigma_o \sigma_r)$$

where $\sigma_r : N' \longrightarrow^* N$ is the read-back reduction of N'. On the other hand, given a normalizing standard proof net reduction ρ, we have a corresponding optimal sharing reduction σ_o followed by a read-back reduction σ_r s.t. the above inequality holds.

Let us call recall that Baillot, Coppola, and Dal Lago just proved that, if $\mathcal{C}_{\mathsf{RB}}(N')$ is the cost of the external algorithm that recover a proof net in normal form from its sharing representation, then $\mathcal{C}_{\mathsf{SG}}(\sigma_o) + \mathcal{C}_{\mathsf{RB}}(N')$ is in the polytime (Kalmar-elementary) function class. Therefore, since our result holds for every SG-reduction, and not only for those ending with an normal form, we generalize the already known result in two ways, making clear that sharing reductions perform not worst (and certainly better as soon us one reduces a non-linear term) than cut elimination, and that this holds even if one changes the notion of normal form; for instance, by taking some kind of lazy reduction that does not reduce under λ-abstractions.

We show then, in full detail and for the first time, an explicit account of the expected benefits of sharing implementation and lazy duplication.

Syntactical approach

Following an approach already exploited in [8, 9], the proof of our result makes use of an intermediate graph rewriting system without sharing, denoted by ED, where the points in which sharing superposition might take place are explicitly marked. The key properties of the intermediate system ED are that there is a trivial simulation relation between its reductions and those of the standard system CE (just ignore the operators used to mark sharing points), and that every sharing reduction in SG can be simulated in ED by a series of parallel moves. The composition of these two simulation relations leads then to a syntactical correspondence between the shared reduction of SG and the cut elimination of proof nets. In the above mentioned papers, this simulation allows to prove the correctness of sharing reductions, here we show that it also allows to track computational complexity back to proof net cut elimination.

The following diagram depicts an overview of the simulation relation.

$$\begin{tikzcd}
 & G' \arrow[r, "\mathsf{SG}"] \arrow[d, mapsto, dashed, "\twoheadleftarrow"'] & G'' \arrow[r, phantom, "\cdots"] \arrow[d, mapsto, dashed, "\twoheadleftarrow"'] & G^i \arrow[d, mapsto, dashed, "\twoheadleftarrow"'] \arrow[ddr, dashed, "\mathsf{RB} \subset \mathsf{SG}"] & \\
 & \overline{T}' \arrow[r, dashed, "\mathsf{ED}"] \arrow[d, mapsto, dashed] & \overline{T}'' \arrow[r, phantom, "\cdots"] \arrow[d, mapsto, dashed] & \overline{T}^i \arrow[d, mapsto, dashed] \arrow[dr, dashed, "\mathsf{ED}"] & \\
T \arrow[uur, "\mathsf{SG}"] \arrow[ur, dashed, "\mathsf{ED}"] \arrow[r, dashed, "\mathsf{CE}"'] & T' \arrow[r, dashed, "\mathsf{CE}"] & T'' \arrow[r, phantom, "\cdots"] & T^i \arrow[r, dashed, "\mathsf{CE}"] & T^{i+1}
\end{tikzcd}$$

We recall that SG includes a set of reduction rules RB that performs the read-back procedure, i.e., which extracts the proof net hidden in a possibly tangled sharing graph. We also recall that the above simulation diagram holds even if we restrict the upper reduction to a subset of optimal rules $\mathsf{SG}_o \subset \mathsf{SG}$ which avoids useless unfolding of sharing and which corresponds to an implementation of Lévy's families (optimal) reduction.

We also remark that in the case of proof nets, the initial translation function $\mathcal{T}$ mapping proof nets to sharing graphs reduces to the identity. Therefore, since RB internalize read-back into SG, correctness of sharing reduction follows from the fact that for every sharing reduction ρ_s^{i+1} from a proof net T to some proof net T^{i+1}, there is a proof net reduction $\rho^{i+1} : T \longrightarrow^* T^{i+1}$. Moreover, the above correspondence holds all along the reduction of T, i.e., given an SG reduction $\rho^i : T \longrightarrow^* T^i$, there is a corresponding ED reduction $\rho_u^i : T \longrightarrow^* \overline{T}^i$ and a corresponding proof net reduction $\rho^i : T \longrightarrow^* T^i$, where $\overline{T}^i$ is the unfolding of G^i and T^i is the proof net obtained from $\overline{T}^i$ by erasing the operators marking sharing point. Finally, even for these intermediate results we can relate cost of reduction

$$\mathcal{C}_{\mathsf{CE}}(\rho^i) \geq \mathcal{C}_{\mathsf{ED}}(\rho_u^i) \geq \mathcal{C}_{\mathsf{SG}}(\rho_s^i)$$

where $\mathcal{C}_{\mathsf{ED}}$ is a suitable cost function defined on ED reductions.

Sharing insight

The syntactical approach described above enlightens the relation between the two styles in which the systems SG and CE perform duplication—global and synchronous in proof net cut elimination, local and asynchronous in sharing graph reduction.

More importantly, it offers a novel insight on the sharing technique. Indeed, the main role of the intermediate system ED is to link together the complementary behaviors of CE and SG. In particular, on one side it reveals the work of duplication in CE, by means of reduction steps; on the other side it makes explicit and accountable the amount of shared links of a sharing graph.

For these reasons, our work appears a suitable starting point for two, more general, related complexity problems which are still open: the cost model for the lambda calculus itself, and the cost of its sharing implementation.

References

[1] Asperti, Andrea and Stefano Guerrini: *The Optimal Implementation of Functional Programming Languages*, volume 45 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.

[2] Asperti, Andrea and Harry G. Mairson: *Parallel beta reduction is not elementary recursive*. In *Conference record of POPL '98*, pages 303–315. ACM Press, 1998.

[3] Asperti, Andrea and Luca Roversi: *Intuitionistic light affine logic*. ACM Transactions on Computational Logic (TOCL), 3(1):137–175, 2002.

[4] Baillot, Patrick, Paolo Coppola, and Ugo Dal Lago: *Light logics and optimal reduction: Completeness and complexity*. Information and Computation, 209(2):118–142, 2011.

[5] Girard, Jean Yves: *Light linear logic*. Inform. and Comput., 143(2):175–204, 1998, ISSN 0890-5401.

[6] Gonthier, Georges, Martín Abadi, and Jean Jacques Lèvy: *The geometry of optimal lambda reduction*. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 15–26, Albequerque, New Mexico, 1992.

[7] Gonthier, Georges, Martín Abadi, and Jean Jacques Lévy: *Linear Logic without boxes*. In *LICS*, pages 223–234. IEEE Computer Society, 1992.

[8] Guerrini, Stefano: *A general theory of sharing graphs*. TCS: Theoretical Computer Science, 227, 1999.

[9] Guerrini, Stefano, Simone Martini, and Andrea Masini: *Coherence for sharing proof-nets*. Theoretical computer science, 294(3):379–409, 2003.

[10] Lamping, John: *An algorithm for optimal lambda calculus reduction*. In *17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 16–30. ACM, 1989.

[11] Lévy, Jean Jacques: *Réductions Correctes et Optimales dans le Lambda Calcul*. PhD thesis, Université Paris VII, 1978.