Confluence for untyped proof nets via parallel cut elimination

Giulio Guerrieri LIS, Aix-Marseille Université, giulio.guerrieri@lis-lab.fr
Giulia Manara IRIF, Université Paris Cité, giulia.manara@etu.univ-amu.fr
Lorenzo Tortora de Falco Dipartimento di Matematica e Fisica, Università degli Studi Roma Tre, tortora@uniroma3.it
Lionel Vaux Auclair I2M, Aix-Marseille Université, lionel.vaux@univ-amu.fr

A rewriting system is *confluent* (or Church–Rosser) if, whenever two distinct sequences of reduction steps can be applied to the same term, there is a term that is reachable from both results, by applying (possibly empty) sequences of additional reduction steps. Confluence simplifies the study of the convergence of programs (represented by the terms in the rewriting system) because it ensures that the ordering in which the reductions are chosen does not make a difference to the result: of any two possible redexes in a term, the reduction of one instead of the other will never preclude the fact of eventually reaching the same result (if any). In some sense, confluence assures modularity of computation.

The Curry-Howard correspondence [2, 6] relates proofs to programs, and cut elimination (a proof rewriting technique to show the coherence of a proof system) to program execution. Thus, confluence of cut elimination procedure assumed a computational value. It is well-known that intuitionistic sequent calculus LJ enjoys confluence while classical sequent calculus LK does not. In 1987 Girard [5] introduced linear logic (LL): a refinement of LJ and LK allowing a finer analysis of the use of hypotheses/resources in proofs/programs via the introduction of two dual modalities, the exponentials ! and ?: in LL, a formula of the form $!A$ (representing a resource available at will in cut elimination) can be proved only if each hypotheses is of the form $?B$. Girard also gave a graphical syntax to represent LL proofs via special directed graphs whose edges are labeled by LL formulas: *proof nets*. In LL proof nets, a formula $!A$ is introduced by means of a "box" marking the context of its hypotheses $?B$. Boxes are the only sub-graphs of a proof net that can be erased or duplicated at will during cut elimination.

Proof nets belong to a wider set of directed graphs labeled by LL formulas: *proof structures*. Not all proof structures are proof nets (that is, represent a proof in LL), but a geometrical criterion characterizes all and only the proof nets among the proof structures. Such a criterion, called AC, deals with acyclicity of proof structures. Cut elimination steps for LL can be defined on proof structures. And being AC (which is preserved by cut elimination steps) guarantees that the cut elimination procedure as good rewriting properties, such as *strong normalization* (that is, every reduction sequence is finite), confluence and the fact that normal forms (the proof structures where no cut elimination step can be applied) are cut-free [3, 7]. Without AC, strong normalization still holds, but confluence fails and normal forms can still have some cuts (counterexamples are in [7, 4]). Actually, cut elimination procedure naturally generalizes to *untyped* proof structures, where edges are not labeled by LL formulas: for AC untyped proof structures, cut elimination is still confluent, and hence it has a clear computational meaning for AC untyped proof structures (despite their lack of a logical content), but strong normalization fails (see Figure 1) and normal forms are not necessarily cut free [7, 4]. However, dropping AC, cut elimination for untyped proof structures is not confluent (a counterexample is in [7, Figure 12]), not even in the multiplicative-exponential fragment of LL (MELL), in which the untyped λ-calculus can be encoded.

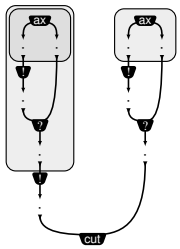

Figure 1: Example of a proof structure which is not normalizable, from [4]

Goal. In the literature, for instance [3, 7], all the proofs of confluence of cut elimination for (typed or untyped) AC proof structures rely on Newman's lemma, which states that a reduction is confluent if it is strongly normalizing and *locally confluent* (that is, whenever two distinct reduction steps can be applied to the same term, there exists a term that is reachable from both results, by applying—possibly empty—sequences of additional reductions). Newman's lemma can be applied directly to cut elimination for typed AC proof structures, but not to cut elimination for the untyped ones, since it is not strongly normalizing, as we have mentioned. In the untyped case, Newman's lemma is applied to some variant of cut elimination (finite development [3], or some subsets of the cut elimination steps that are separately strongly normalizing [7]), and confluence of cut elimination is then deduced from confluence of the variant.

Our goal is to give a proof of confluence for MELL AC untyped proof structures that is not based on Newman's lemma and strong normalization, not even indirectly. We have chosen:

- MELL because this fragment of LL is expressive enough and its proof structures can be defined in a simple way;
- AC because otherwise confluence fails, as we have mentioned;
- untyped because otherwise cut elimination is strongly normalizing, and we do not want to use this hypothesis, not even surreptitiously; of course, the fact that our proof works for untyped proof structures means that it also works in the typed case.

This way, we disentangle confluence from strong normalization for MELL AC proof structures, which is a conceptual achievement and a technical simplification of the proof of confluence.

Method. To achieve our goal we take inspiration from a basic result in rewriting theory. Given a reduction step $\to$ on a set of terms, and its reflexive-transitive closure $\to^*$ (i.e. the composition of $\to$ an arbitrary—possibly zero—number of times), to prove that $\to$ is confluent it is enough to have a reduction $\dashrightarrow$ such that:

1. $\to \subseteq \dashrightarrow \subseteq \to^*$, that is, $\dashrightarrow$ includes $\to$ and it is included in $\to^*$;
2. $\dashrightarrow$ is *diamond*, that is, whenever two distinct reduction steps can be applied to the same term, there exists a term that is reachable from both results, by applying a single reduction step.

From these two properties, confluence of $\to$ immediately follows, without using Newman's lemma or strong normalization, for any kind of reduction.

Let us see how this method had been successfully applied to the untyped λ-calculus: there, β-reduction $\to_\beta$ is neither strongly normalizing (indeed, $(\lambda x.xx)\lambda x.xx \to_\beta (\lambda x.xx)\lambda x.xx \to_\beta \dots$) nor diamond. For a counterexample to the diamond property, consider the term $t = (\lambda x.xx)((\lambda y.y)z)$, which contains two redexes. In fact, we have:

$$t \to_\beta (\lambda x.xx)z = t_1 \qquad\qquad t \to_\beta ((\lambda y.y)z)((\lambda y.y)z) = t_2$$

The terms t_1 and t_2 have a common reduct zz, but it cannot be reached in one single $\to_\beta$ step. Indeed, the step $t \to_\beta ((\lambda y.y)z)((\lambda y.y)z)$ duplicates the rightmost redex in t, so that one reduction step is not enough to close the diagram.

To prove that $\to_\beta$ is confluent, Tait and Martin-Löf solved the problem introducing a technique, later simplified by Takahashi [9], based on *parallel β-reduction* $\Rightarrow_\beta$: in a single step, parallel β-reduction can reduce an arbitrary number (possibly zero) of existing redexes in a term. This leads to some nice properties: $\Rightarrow_\beta$ is diamond and $\to_\beta \subseteq \Rightarrow_\beta \subseteq \to_\beta^*$. Note how $\Rightarrow_\beta$ solves the problem presented above in just one (parallel) step:

$$((\lambda y.y)z)((\lambda y.y)z) \Rightarrow_\beta zz \,{}_\beta\!\Leftarrow (\lambda x.xx)z$$

The Tait–Martin-Löf proof of confluence of $\to_\beta$ is one of the simplest and most elegant, it does not need to resort to Newman's lemma: it only relies on the two key properties of $\Rightarrow_\beta$.

Our idea to prove confluence of cut elimination for MELL AC untyped proof structures is then to adapt the notion of parallel reduction to this framework, in the spirit of what done in [1] for the multiplicative fragment of LL (MLL).

Challenges. To achieve our goal, we encountered some technical difficulties of different kinds.

First, choosing the right syntax for proof structures to prove the diamond property of parallel reduction turned out to be a nontrivial problem. Examples in Figures 2 and 3 show how the choice of standard syntax (with different ?-nodes for dereliction, weakening and contraction) prevents parallel reduction from being diamond. Problems arise with the exponential cut elimination step. In Figure 2 the topmost proof structure can be reduced either in the one on the left, eliminating the green box/box cut, or in the one on the right, eliminating the red and blue box/box cuts. Here boxes should cross more than one border in a single step to make parallel reduction diamond. In Figure 3 the cut chain consists of a box/box cut and a box/contraction cut. Note that in order to be diamond the reduction is needed to duplicate boxes and to transport them to the depth where the contraction premises are located. All these problems vanish with a different choice of the syntax, namely the one introduced by Regnier [8] with *generalized* ?*-nodes* (an arbitrary number—possibly zero—of premises of type A, one conclusion of $?A$). In this setting contraction, dereliction and weakening crush on one connective thus we have only one case of exponential cut. Exponential cut elimination simultaneously duplicates and opens the boxes transporting their content to arbitrary depths: exactly what we need.

Another difficulty lies in the fact that while λ-terms are built inductively, (untyped) proof structures are not. The former can be seen as trees whose root allows one to identify the last rule that was applied to construct them, whereas the latter are graphs without a clear notion of root: it is not evident how to detect a distinguished cut analogous to the head redex in a λ-term. The challenge is therefore to find a way to factorize the graph in a canonical way allowing for an unambiguous inductive definition of parallel reduction. Once again the λ-calculus comes to our aid, in particular we reflected on the following inductive step in the definition of parallel β-reduction:

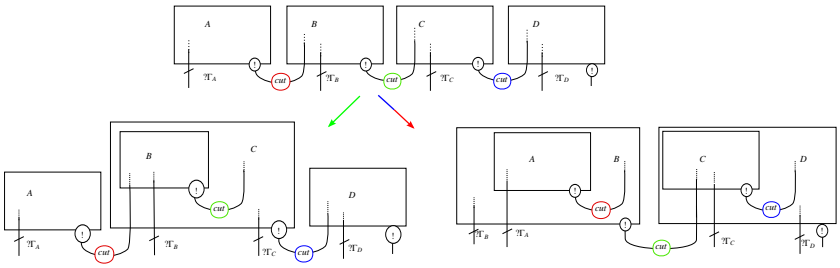

Figure 2: *Chain of exponential cuts (box/box).* The edges with a dash represent multiple edges.

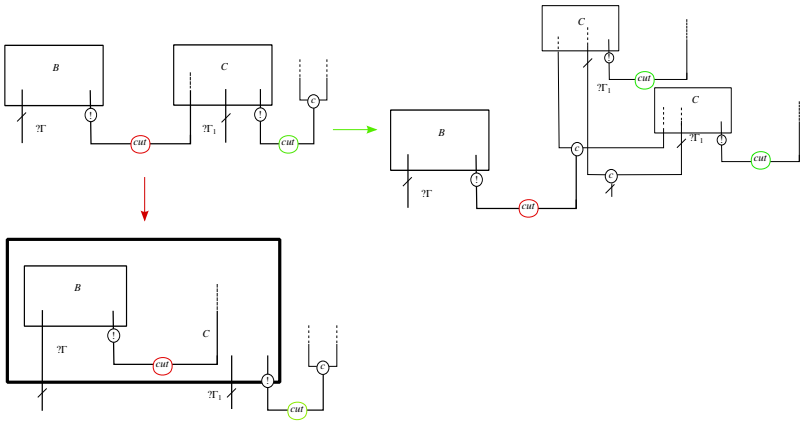

Figure 3: *Chain of exponential cuts(promotion/box border and promotion/contraction).* The edges with a dash represent multiple edges.

$$\frac{s \Rightarrow_\beta s' \qquad u \Rightarrow_\beta u'}{(\lambda x.s)u \Rightarrow_\beta s'[x/u']}\ \star$$

We denote by $\bar{t}$ the MELL untyped proof structure that encodes the λ-term t. The external redex $(\lambda x.s)u$ fired in $\star$ corresponds to a cut between a conclusion of $\overline{\lambda x.s}$ and the principal door of a box containing the translation of $\overline{u}$. This box has the property of being *external*, that is, *none of its ? conclusions is itself a premise of a cut.* For example, in the proof structure G_1 in Figure 4, the red box is external while the blue is not; in the proof structure G_2 in Figure 4, both the red and the blue boxes are not external. In every proof structure that is the image of some λ-term, there always exists an external box, which is not necessarily unique. More generally, what about MELL untyped proof structures, not necessarily in the image of the translation from the λ-calculus? A condition that guarantees the existence of the external box in a MELL untyped proof structure is *being* AC. In Figure 4, the proof structure G_1 is AC and contains one external box, while G_2 is not AC and is with no external box.

The last challenge comes with the factorization of a MELL untyped proof structure into several parts: the idea is to decompose a MELL untyped proof structure G having an external box of content b as a context H in which b was pluggable in a specific way. Note, however, that the content of a box is not a proof structure, since we are using the syntax with generalized ?-nodes. In G_1 in Figure 4, if we extract the content of the big red box, we obtain the graph b_1 in Figure 4. The red box of b_1 has a conclusion (denoted $*$ in the figure) which is not plugged to a ?-node: this is not admitted by the definition of MELL proof structures. Thus we decided to consider a more general form of MELL proof structures, where the auxiliary ports of a box are not necessarily linked to a ?-node: the MELL (*proof*) *pre-structures*. All the notions of AC, cut elimination, typed/untyped can be generalized to MELL pre-structures. We can factorize any MELL AC untyped pre-structure G, having an external box of content b plugged in a context H, as follows:

$$G = H[b/x]$$

where x is the additional information needed to know where to plug b. In Figure 4 we have $G_1 = H_1[b_1/x_1]$. Clearly, we had to prove that the result of a parallel reduction step does not depend on the choice of a particular external box.

Conclusion. Finally, we can define the analogue of $\star$ for parallel cut elimination on MELL pre-structures as follows:

$$\frac{H \Rightarrow H' \qquad b \Rightarrow b'}{H[b/x] \Rightarrow H'[b'/x]}$$

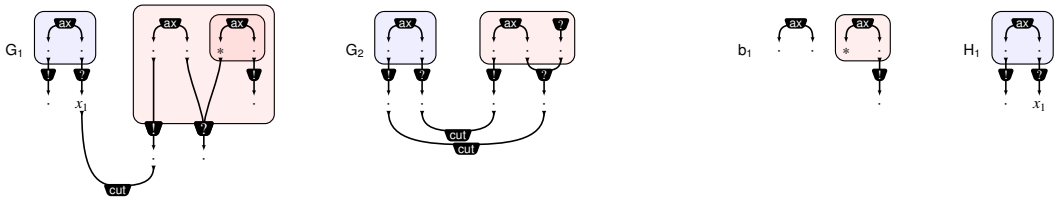

Figure 4: An acyclic proof structure:G_1. A cyclic proof structure:G_2. A pre-structure: b_1. A prestructure that is a context for G_1: H.

This rule is the key ingredient in the definition of parallel cut elimination: using it we were able to prove confluence of cut elimination for MELL AC untyped pre-structures without referring to Newman's lemma or strong normalization.

References

[1] Jules Chouquet and Lionel Vaux Auclair. An application of parallel cut elimination in multiplicative linear logic to the taylor expansion of proof nets, 2020.

[2] Haskell B. Curry and Robert. Feys. *Combinatory Logic*. Number vol. 2 in Combinatory Logic. North-Holland Publishing Company, 1958.

[3] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du Lambda-calcul)*. PhD thesis, Université Paris VII, 1990.

[4] Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in linear logic. *Theoretical Computer Science*, 412(20):1884–1902, 2011. Girard's Festschrift.

[5] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[6] William Alvin Howard. The formulae-as-types notion of construction. In Haskell Curry, Hindley B., Seldin J. Roger, and P. Jonathan, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, 1980.

[7] Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. *Theoretical Computer Science*, 411(2):410–444, 2010.

[8] Laurent Régnier. *Lambda-calcul et reseaux*. PhD thesis, Université Paris VII, 1992.

[9] Masako Takahashi. Parallel reductions in λ-calculus. *Journal of Symbolic Computation*, 7(2):113–123, 1989.