Non-associative composition in string diagrams and sequential proof-nets for calculi with effects

Éléonore Mangel *Univ. Paris Cité, CNRS, INRIA*
Paul-André Melliès *Univ. Paris Cité, CNRS, INRIA*
Guillaume Munch-Maccagnoni *INRIA, LS2N CNRS, Nantes*

May 10, 2026

In this talk, we will describe two different and equivalent diagrammatic ways to describe how effectul programs compose in a possibly non-associative way. The first formalism is a variant of functorial game semantics depicted in the 2-categorical language of string diagrams [1, 2, 3]. The second formalism is based on *sequential proof-nets*, a new graphical representation of proofs with non-associative composition recently introduced by the three authors for the classical L-calculus [4].

In order to understand where non-associativity of composition comes from, consider a language with two different **let** constructors:

- ▷ a $\mathbf{let}^{\mathsf{cbv}}$ constructor in call-by-value, where $\mathbf{let}^{\mathsf{cbv}}\, x = u \;\mathbf{in}\; t$ evaluates u before t,
- ▷ a $\mathbf{let}^{\mathsf{cbn}}$ constructor in call-by-name, where $\mathbf{let}^{\mathsf{cbn}}\, x = u \;\mathbf{in}\; t$ evaluates t before u.

Then, consider the two effectful programs u and v defined as follows:

$$\begin{aligned} u &\equiv \mathbf{let}^{\mathsf{cbv}}\, x = f \;\mathbf{in}\; (\mathbf{let}^{\mathsf{cbn}}\, y = g \;\mathbf{in}\; h) \\ v &\equiv \mathbf{let}^{\mathsf{cbn}}\, y = (\mathbf{let}^{\mathsf{cbv}}\, x = f \;\mathbf{in}\; g) \;\mathbf{in}\; h \end{aligned} \tag{1}$$

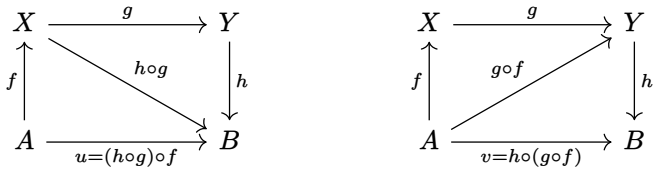
where f, g and h are themselves effectful programs and where x does not occur in h. The three programs f, g and h define a path of length 3

$$A \xrightarrow{\;f\;} X \xrightarrow{\;g\;} Y \xrightarrow{\;h\;} B$$

where the two effectful programs

$$u = (h \circ g) \circ f \qquad v = h \circ (g \circ f)$$

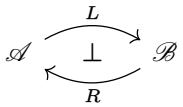
are obtained as composite with different orders of composition:



The difference between the two programs is only in the position of the parentheses. However, the left-hand side will evaluate first f, then h and finally g, whereas the right-hand side will evaluate first h, then f and finally g. Since the programs are effectful, the order of evaluation matters and the two programs are not necessarily equal. Thus non-associativity emerges.

This phenomenon of non-associativity was first observed by Girard in his work on classical logic [5] and then independently by Abramsky in his work on the Blass game model of linear logic [6, 7]. The categorical explanation of the phenomenon based on an adjunction $L \dashv R$ between positive types for call-by-value and negative types for call-by-name was given by the second author in [8]. The theory of duploids was then developed on these observations by the third author [9] as a direct and non-associative categorical semantics of the L-calculus, a polarized variant of the $\lambda\mu\tilde{\mu}$-calculus [10].

The main idea of [8, 9] is that since call-by-value is usually modelled by the Kleisli category of a monad and call-by-name by the co-Kleisli of a comonad, to mix them together, it makes sense to start from an adjunction:



The collage category $\mathbf{collage}_{L,R}$ of the adjunction has objects of the form $(0, A)$ where A is an object of $\mathscr{A}$ and $(1, B)$ where B is an object of the category $\mathscr{B}$, with hom-sets defined as follows:

$$\begin{array}{ccc} \mathbf{collage}_{L,R}(A, A') & := & \mathscr{A}(A, A') \\ \mathbf{collage}_{L,R}(B, B') & := & \mathscr{B}(B, B') \\ \mathbf{collage}_{L,R}(A, B) & := & \mathscr{A}(A, RB) \\ \mathbf{collage}_{L,R}(B, A) & := & \text{empty} \end{array}$$

where we write A instead of $(0, A)$ and B instead of $(1, B)$ for notational simplicity and clarity. Note that by definition of the adjunction $L \dashv R$ there is a family of bijections

$$\mathscr{A}(A, RB) \quad \cong \quad \mathscr{B}(LA, B)$$

We could have thus equivalenty defined

$$\mathbf{collage}_{L,R}(A, B) \quad := \quad \mathscr{B}(LA, B)$$

The collage category of the adjunction $L \dashv R$ may be seen as a Grothendieck construction associated to the functor

$$(L, R) \quad : \quad \mathbf{2} \longrightarrow \mathbf{Adj}$$

from the category $\mathbf{2}$ with two objects 0 and 1 and one morphism $0 \to 1$ to the 2-category $\mathbf{Adj}$ of adjunctions.

In the same way as the collage category "glues" together the categories $\mathscr{A}$ and $\mathscr{B}$, we would like to "glue" together the Kleisli category of the monad $R \circ L$ and the co-Kleisli category of the comonad $L \circ R$. However, as observed in [8] in the case of the category of Blass games, this construction generally induces a non-associative category. The construction works for any adjunction $L \dashv R$ and was coined the *duploid construction* in [9]. The non-associative category noted $\mathbf{dupl}_{L,R}$ has the same objects as the collage category, and its hom-sets are defined as:

- For A and A', both in $\mathscr{A}$, $\mathbf{dupl}_{L,R}(A, A') = \mathscr{A}(A, RLA)$, i.e. the Kleisli morphisms,
- For B and B', both in $\mathscr{B}$, $\mathbf{dupl}_{L,R}(B, B') = \mathscr{B}(LRB, B')$, i.e. the coKleisli morphisms,
- For $A \in \mathscr{A}$ and $B' \in \mathscr{B}$, $\mathbf{dupl}_{L,R}(A, B') = \mathscr{A}(A, RB')$,
- For $B \in \mathscr{B}$ and $A' \in \mathscr{A}$, $\mathbf{dupl}_{L,R}(B, A') = \mathscr{A}(RB, RLA')$.

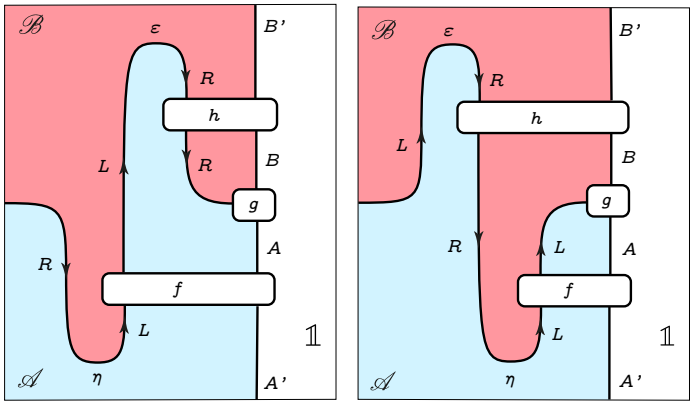
This construction has identities and composition, but contrary to the collage category, which is an associative category in the usual sense, the duploid $\mathbf{dupl}_{L,R}$ is not associative.

First account: non-associativity in 2-categorical string diagrams

This phenomenon of non-associativity can be depicted in the language of 2-categorical string diagrams using the ideas of functorial game semantics [1, 2] and of string diagrams for premonoidal categories [11]. Suppose given two objects A, A' of the category $\mathscr{A}$ and two objects B, B' of the category $\mathscr{B}$ as well as three morphisms

$$\begin{aligned} f &\in \mathbf{dupl}_{L,R}(A', A) \cong \mathscr{B}(LA', LA) \\ g &\in \mathbf{dupl}_{L,R}(A', B) = \mathscr{A}(A, RB') \cong \mathscr{B}(LA, B) \\ h &\in \mathbf{dupl}_{L,R}(B, B') \cong \mathscr{A}(RB, RB') \end{aligned}$$

The two possible compositions are represented as follows in the language of 2-categorical string diagrams:



The flow of control, determined by the trajectories of the functors L and R, indicates the order of evaluation. On the left-hand side, f is the first to be evaluated, then h and finally g, whereas on the right-hand side, it starts by h, then f and finally g, demonstrating the lack of associativity. Cells with a horizontal input such as the one labelled g conveniently represent oblique morphisms of the adjunction $L \dashv R$.

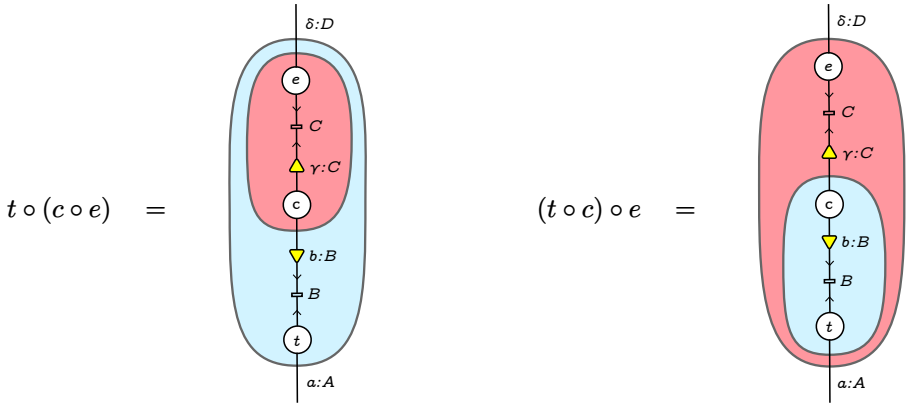
Second account: non-associativity in sequential proof-nets

A second way to picture this phenomenon relies on the notion of *sequential proof-nets* recently introduced by the three authors [12]. In these sequential proof-nets, every cut between two programs is located inside a dedicated "effect area" whose purpose is to encapsulate the computational effects produced by the cut. The colour of the effect area depends on the order of evaluation: light blue for call-by-value and dark red for call-by-name.

Instead of being explicitly depicted with a string as in functorial game semantics, the control flow remains implicit and is represented by these effect areas, which describe the order of compositions in the same way as parenthesis in the syntax. Consider for example the two possible compositions of three programs t, c and e as follows

$$t \circ (c \circ e) \qquad\qquad (t \circ c) \circ e$$

and their respective representation as sequential proof-nets:



As a matter of fact, these diagrams based on sequential proof-nets are in correspondence with the linear classical L-calculus, introduced in [4] by the three authors, which can be understood as an abstract-machine calculus for polarised multiplicative linear logic. In particular, removing the effect areas (and the $\mu\tilde{\mu}$-binders) of a sequential proof-net induces a multiplicative proof-nets (depicted in the style of Lafont interaction nets [13]). This "depolarization process" transports a proof of system L and polarized classical linear logic into linear logic and is not reversible as there are multiple different sequentializations of a single multiplicative proof-nets of (depolarized) multiplicative linear logic.

References

[1] P.-A. Melliès, "Game semantics in string diagrams," in *Proceedings of the Twenty-Seventh Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS)*, ACM SIGACT and IEEE Computer Society. IEEE Computer Society, 2012, pp. 481 – 490.

[2] P.-A. Melliès, "Une étude micrologique de la négation," Habilitation à Diriger des Recherches, Université Paris Diderot, 2017.

[3] P.-A. Melliès, "Ribbon tensorial logic," in *Proceedings of the Thirty-Third Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS)*, ACM SIGACT and IEEE Computer Society. IEEE Computer Society, 2018.

[4] E. Mangel, P.-A. Melliès, and G. Munch-Maccagnoni, "Classical Notions of Computation and the Hasegawa-Thielecke Theorem," *Proc. ACM Program. Lang.*, vol. 10, no. POPL, Jan. 2026. [Online]. Available: https://doi.org/10.1145/3776715

[5] J.-Y. Girard, "On the Unity of Logic," *Ann. Pure Appl. Logic*, vol. 59, no. 3, pp. 201–217, 1993.

[6] A. Blass, "A game semantics for linear logic," *Annals of Pure and Applied Logic*, vol. 56, no. 1, pp. 183–220, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0168007292900739

[7] S. Abramsky, "Sequentiality vs. concurrency in games and logic," *Mathematical Structures in Computer Science*, vol. 13, no. 4, p. 531565, 2003.

[8] P.-A. Melliès, "Asynchronous games 3: An innocent model of linear logic," in *Proceedings of the 10th Conference on Categorical Structures in Computer Science (CTCS 2004)*, vol. 122. Electronic Notes in Theoretical Computer Science, 2005, pp. 171–192.

[9] G. Munch-Maccagnoni, "Models of a non-associative composition," in *Proc. FoSSaCS 2014*. Springer, 2014, pp. 396–410.

[10] P.-L. Curien and H. Herbelin, "The duality of computation," *ACM sigplan notices*, vol. 35, no. 9, pp. 233–243, 2000.

[11] A. Jeffrey, "Premonoidal categories and a graphical view of programs," 1998.

[12] E. Mangel, P.-A. Melliès, and G. Munch-Maccagnoni, "The free dialogue duploid as a non-associative category of sequential proof-nets," 2025, unpublished manuscript.

[13] Y. Lafont, *From proof nets to interaction nets*. Cambridge University Press, Jun. 1995, pp. 225–248.