∂ is for Dialectica

Marie Kerjean and Pierre-Marie Pédrot

[1] CNRS, LIPN, Université Sorbonne Paris Nord kerjean@lipn.fr
[2] Inria pedrot@inria.fr

Dialectica was originally introduced by Gödel in a famous paper [7] as a way to constructively interpret an extension of HA [1], but turned out to be a very fertile object of its own. Judged too complex, it was quickly simplified by Kreisel into the well-known realizability interpretation that now bears his name. Soon after the inception of Linear Logic (LL), Dialectica was shown to factorize through Girard's embedding of LJ into LL, purveying an expressive technique to build categorical models of LL [13]. In its logical outfit, Dialectica led to numerous applications and was tweaked into an unending array of variations in the proof mining community [10].

The modern way to look at Dialectica is however to consider it as a program translation, or more precisely *two* mutually defined translations of the λ-calculus exposing intensional information [14].

In a different scientific universe, Automatic Differentiation [8] (AD) is the field that studies the design and implementation of *efficient* algorithms computing the differentiation of mathematical expression and numerical programs. Indeed, due to the chain rule, computing the differential of a sequence of expressions involves a choice, namely when to compute the value of a given expression and when to compute the value of its derivative. Two extremal algorithms coexist. On the one hand, forward differentiation [16] computes functions and their derivatives pairwise in the order they are provided, while on the other hand reverse differentiation [12] computes all functions first and then their derivative in reverse order. Depending on the setting, one can behave more efficiently than the other. Notably, reverse differentiation has been critically used in the fashionable context of deep learning.

Differentiable programming is a rather new and lively research domain aiming at expressing automatic differentiation techniques through the prism of the traditional tools of the programming language theory community. As such, it has been studied through continuations [15], functoriality [6], and linear types [4]. It led to a myriad of implementation over rich programming languages, proven correct through semantics of higher-order differentiable functions [11]. Surprisingly, these various principled explorations of automatic differentiation are what allows us to draw a link between Dialectica and differentiation in logic.

The simple, albeit fundamental claim of this talk is that, behind its different logical avatars, the Dialectica translation is in fact a reverse differentiation algorithm, where the linearity and involutivity of differentiation have been forgotten. In the domain of proof theory, differentiation has been very much studied from the point of view of *linear logic*. This led to Differential Linear Logic [5] (DiLL), differential categories [3], or the differential λ-calculus. To support our thesis

with evidence, we will formally state a correspondence between each of these objects and the corresponding Dialectica interpretation.

More generally, Dialectica is known for extracting *quantitative* information from proofs [10], and this relates very much with the quantitative point of view that differentiation has brought to λ-calculus [2]. Herbelin also notices at the end of its paper realizing Markov's rule through delimited continuations that this axiom has the type of a differentiation operator [9]. If time permits, we will explore the possible consequences of formally relating reverse differentiation and Dialectica to proof mining and Herbelin's work in the conclusion.

References

1. Avigad, J., Feferman, S.: Gödel's functional ('dialectica') interpretation. In: Buss, S.R. (ed.) Handbook of Proof Theory. Elsevier Science Publishers, (1998)
2. Barbarossa, D., Manzonetto, G.: Taylor subsumes scott, berry, kahn and plotkin. Proc. ACM Program. Lang. **4**(POPL), 1:1–1:23 (2020).
3. Blute, R.F., Cockett, J.R.B., Seely, R.A.G.: Differential categories. Math. Structures Comput. Sci. **16**(6) (2006).
4. Brunel, A., Mazza, D., Pagani, M.: Backpropagation in the simply typed lambda-calculus with linear negation. POPL (2020).
5. Ehrhard, T., Regnier, L.: Differential interaction nets. Theoretical Computer Science **364**(2) (2006)
6. Elliott, C.: The simple essence of automatic differentiation. In: Proceedings of the ACM on Programming Languages (ICFP) (2018).
7. Gödel, K.: Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. Dialectica **12**, 280–287 (1958)
8. Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Society for Industrial and Applied Mathematics, USA, second edn. (2008)
9. Herbelin, H.: An intuitionistic logic that proves markov's principle. LICS 2010.
10. Kohlenbach, U.: Applied Proof Theory - Proof Interpretations and their Use in Mathematics. Springer Monographs in Mathematics, Springer (2008).
11. Krawiec, F., Peyton Jones, S., Krishnaswami, N., Ellis, T., Eisenberg, R.A., Fitzgibbon, A.: Provably correct, asymptotically efficient, higher-order reverse-mode automatic differentiation. Proc. ACM Program. Lang. **6**(POPL) (jan 2022).
12. Linnainmaa, S.: Taylor expansion of the accumulated rounding error. BIT Numerical Mathematics **16**, 146–160 (1976)
13. de Paiva, V.: A dialectica-like model of linear logic. In: Category Theory and Computer Science, (1989)
14. Pédrot, P.: A functional functional interpretation. CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014.
15. Wang, F., Zheng, D., Decker, J., Wu, X., Essertel, G.M., Rompf, T.: Demystifying differentiable programming: Shift/reset the penultimate backpropagator. **3**(ICFP), (2019).
16. Wengert, R.E.: A simple automatic derivative evaluation program. Commun. ACM **7**(8), 463–464 (1964).