Sequential algorithms as a Dialectica interpretation

Valentin Blot

INRIA, Laboratoire des Sciences du Numérique de Nantes, Nantes Université

We provide the first formal connection between Gödel's Dialectica interpretation and Berry-Curien's sequential algorithms model, via a parameterized version of Gödel's Dialectica interpretation which allows us to recast both the Diller-Nahm variant of the interpretation and Berry-Curien's sequential algorithms model. This illustrates that the Dialectica interpretation has a strong intensional flavour, as the reverse component of the Dialectica interpretation of a function (from environments in the output to environments in the input) corresponds precisely to the computation indices of sequential algorithms (which specify what is needed in the input in order to produce a given output). While a straightforward instanciation gives the usual Diller-Nahm variant of the Dialectica inerpretation, a more involved instanciation provides exactly the sequential algorithms model. We believe that this is a first step towards a convergence between Dialectica-style computational interpretations of logic on one hand, and game semantics flavoured denotational models of various programming languages on the other hand.

Gödel's Dialectica Gödel's functional interpretation [Göd58], also known as the Dialectica interpretation (from the name of the journal it was published in) is an interpretation of intuitionistic arithmetic into system T (simply-typed λ-calculus with recursion over natural numbers at all finite types). In the Dialectica interpretation, a proof of $A \Rightarrow B$ is interpreted as a two-component object: the direct component maps witnesses of A to witnesses of B, while the reverse component maps (given a witness of A) counter-witnesses of B to counter-witnesses of A. More precisely, if we write $\underline{A}$ for the set of witnesses of A and $\overline{A}$ for the set of counter-witnesses of A, then a witness of $A \Rightarrow B$ is an element of:

$$\underline{A \Rightarrow B} = (\underline{A} \to \underline{B}) \times \left(\underline{A} \to \overline{B} \to \overline{A}\right)$$

The direct component in $\underline{A} \to \underline{B}$ corresponds to the usual Curry-Howard computational interpretation of the proof, while the reverse component in $\underline{A} \to \overline{B} \to \overline{A}$ maps (given some witness of A in $\underline{A}$) counter-witnesses of B (in $\overline{B}$) to counter-witnesses of A (in $\overline{A}$).

The correctness condition of some $a \in \underline{A} \to \underline{B}$ in the usual Curry-Howard case with only the direct component can be stated as "$\forall x \in \underline{A}\,(x$ correct for A implies $a(x) \in \underline{B}$ correct for $B)$". Contrastingly, in the Dialectica interpretation, the correctness condition of some witness in $\underline{A}$ is always relative to some counter-witness in $\overline{A}$, and the correctness of some:

$$\langle a, b\rangle \in \underline{A \Rightarrow B} = (\underline{A} \to \underline{B}) \times \left(\underline{A} \to \overline{B} \to \overline{A}\right) \text{ relatively to some } \langle x, y\rangle \in \overline{A \Rightarrow B} = \underline{A} \times \overline{B}$$

can be stated as "$x \in \underline{A}$ correct for A with respect to $b\,(x)\,(y)$ implies $a\,(x) \in \underline{B}$ correct for B with respect to y". This correctness condition can be seen as a refinement of the Curry-Howard interpretation, in the sense that the reverse component b of such a witness of $A \Rightarrow B$ can provide specific counterwitnesses $b\,(x)\,(y)$, which are the only ones for which the witness x of A (which is part of the counter-witness $\langle x, y\rangle$ of $A \Rightarrow B$) can be considered to be correct. Contrastingly, in the Curry-Howard case the universally quantified x is "universally correct", which is a much stronger condition. Since these correctness conditions propagates inside the whole hierarchy of implications that a formula can contain, the Dialectica interpretation can arguably be considered as a refinement of the usual Curry-Howard interpretation.

Berry-Curien's sequential algorithms Scott continuous functions between complete partial orders (cpo) provide a very convincing abstract notion of computation, based on the paradigm that a computer program exploits only a finite part of his input in order to provide a finite amount of output. But the inherent sequentiality of computer programs is not reflected in Scott's model: a program not only looks at a finite part of its input, it also looks at this finite part step by step in a fixed order. Capturing this notion of sequentiality has proved to be a very challenging problem, formulated by Milner [Mil77] and Plotkin [Plo77] and known as the "full abstraction problem for PCF", where PCF is a simply-typed λ-calculus extended with natural numbers and general recursion. In a fully abstract model of PCF, one has to forbid the non-sequential "parallel or" program (of type $\texttt{bool} \times \texttt{bool} \to \texttt{bool}$) that would evaluate both arguments in parallel and return `true` as soon as one of the arguments returns `true`. As simple as it may seem, such a thing proved to be very difficult, and was ultimately shown to be impossible (in a decidable way) by Loader [Loa01].

In this quest for sequential models of PCF, Kahn and Plotkin [KP93] defined concrete domains, concrete data structures (csd, initially called "information matrices") and sequential functions between these structures. However, their construction lacked a cartesian closed structure, and therefore did not provide models of λ-calculus. The breakthrough came later with Berry and Curien's sequential algorithms model [BC82], who described the construction of a function space cds, whose elements' underlying extensional functions are exactly sequential functions between two given cds. This model was also a great source of inspiration for the very fruitful field of game semantics that emerged in the 90's and 2000's. The sequential algorithms model provides a satisfactory answer to the problem of finding a fully abstract model of a purely sequential language, as it is a fully abstract model of an extension of PCF with a `catch` operator [CF92, CCF94] (similar to Scheme's `call/cc` call-with-current-continuation operator), which is arguably sequential. Morphisms in the sequential algorithms model can be described as "abstract algorithms": pairs of an extensional sequential function together with its "computation strategy". Formally, an abstract algorithm between cds A and B is the pair of a sequential function in $D(A) \to D(B)$ (where $D(A)$ is the Scott domains generated by cds A), together with a computation strategy for it, given as a partial function in $D(A) \to C_B \rightharpoonup C_A$, where C_A is the set of "cells" of cds A, that is, the set of "locations" in A that can be filled with values during the computation process. This computation strategy should be understood as providing, given some current state of the input (the element in $D(A)$), the next location in the input (the element in C_A) that needs to be filled in order to compute the required location in the output (the element in C_B). A sequential algorithm from cds A to cds B can therefore be represented as an element of:

$$(D\,(A) \to D\,(B)) \times (D\,(A) \to C_B \rightharpoonup C_A)$$

Contributions Note of course the similarity between the set of abstract algorithms from cds A to cds B and the set of Dialectica witnesses of $A \Rightarrow B$. We identify two main specificities of the sequential algorithms model with respect to the dialectica interpretation: first, the computation strategy of sequential algorithms (which corresponds to the reverse component of the Dialectica interpretation) must return cells which are accessible in the current input. Second, in the uncurrying of a morphism from T to $U \to V$ in the sequential algorithms case we have to inspect "subcells" of the current cell on the output $U \to V$, that is, cells for which the argument on U may be smaller than the argument in the current cell that is asked on the output $U \to V$. To take these specificities into account, we parameterize the Dialectica interpretation so that a straightforward instantiation gives the usual Diller–Nahm variant of the Dialectica interpretation, and another instantiation provides exactly the sequential algorithms model.

References

[BC82] Gérard Berry and Pierre-Louis Curien. Sequential Algorithms on Concrete Data Structures. *Theoretical Computer Science*, 20(3):265–321, 1982.

[Ber78] Gérard Berry. Stable models of typed lambda-calculi. In *5th Colloquium on Automata, Languages and Programming*, pages 72–89. Springer, 1978.

[CCF94] Robert Cartwright, Pierre-Louis Curien, and Matthias Felleisen. Fully Abstract Semantics for Observably Sequential Languages. *Information and Computation*, 111(2):297–401, 1994.

[CF92] Robert Cartwright and Matthias Felleisen. Observable Sequentiality and Full Abstraction. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 19-22, 1992*, pages 328–342. ACM Press, 1992.

[Göd58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958.

[KP93] Gilles Kahn and Gordon D. Plotkin. Concrete Domains. *Theoretical Computer Science*, 121(1&2):187–277, 1993.

[Loa01] Ralph Loader. Finitary PCF is not decidable. *Theoretical Computer Science*, 266(1-2):341–364, 2001.

[Mil77] Robin Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.

[Plo77] Gordon Plotkin. LCF Considered as a Programming Language. *Theoretical Computer Science*, 5(3):223–255, 1977.

[Vui73] Jean Vuillemin. *Proof techniques for recursive programs.* PhD thesis, Stanford University, USA, 1973.