Diller-Nahm Bar Recursion

Valentin Blot

Diller-Nahm Bar Recursion

Valentin Blot
Inria, Laboratoire Méthodes Formelles, Université Paris-Saclay, France

Abstract

We present a generalization of Spector's bar recursion to the Diller-Nahm variant of Gödel's Dialectica interpretation. This generalized bar recursion collects witnesses of universal formulas in sets of approximation sequences to provide an interpretation to the double-negation shift principle. The interpretation is presented in a fully computational way, implementing sets via lists. We also present a demand-driven version of this extended bar recursion manipulating partial sequences rather than initial segments. We explain why in a Diller-Nahm context there seems to be several versions of this demand-driven bar recursion, but no canonical one.

2012 ACM Subject Classification Theory of computation → Proof theory

Keywords and phrases Dialectica, Bar recursion

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.32

1 Introduction

Gödel's functional interpretation [5], also known as the Dialectica interpretation (from the name of the journal it was published in) is a translation from intuitionistic arithmetic into the Σ_2^0 fragment of intuitionistic arithmetic in finite types. If π is a proof of arithmetical formula A, then the functional interpretation of π is a proof of a formula $A^D \equiv \exists \vec{x} \forall \vec{y} A_D(\vec{x}, \vec{y})$ where A_D is quantifier-free. This formula can be understood as asserting that some two-player game has a winning strategy: there exists a strategy $\vec{x}$ such that for all strategy $\vec{y}$, $\vec{x}$ wins against $\vec{y}$, that is, $A_D(\vec{x}, \vec{y})$ holds. By the witness property, the proof of A^D yields a proof of $\forall \vec{y} A_D(\vec{t}, \vec{y})$ for some sequence of terms $\vec{t}$ in system T: simply-typed λ-calculus with recursion over natural numbers at all finite types. This sequence $\vec{t}$ of programs is the computational content of π under the Dialectica interpretation.

Since the negative translation of every axiom of arithmetic is provable in intuitionistic arithmetic, the Dialectica interpretation combined with a negative translation provides an interpretation of classical arithmetic. When it comes to classical analysis (classical arithmetic plus the axiom of countable choice) this is not true anymore, as the negative translation of the axiom of choice fails to be an intuitionistic consequence of the axiom of choice. Spector's bar recursion operator [11] provides a Dialectica interpretation of the double-negation shift (DNS) principle, from which one can derive intuitionistically any formula from its negative translation. Applying this to the axiom of countable choice, Spector obtains an interpretation of classical analysis.

Interpreting the contraction rule $A \Rightarrow A \wedge A$ in Gödel's original interpretation requires (besides the $\lambda x. \langle x, x \rangle$ component) a program that, given a witness M and two potential counterwitnesses x and y of A such that either x or y wins against M, answers with a single counterwitness that wins against M. Doing that relies on the decidability of winningness, which ultimately relies on the decidability of atomic formulas of the source logic (which is true in arithmetic). In order to get rid of this decidability requirement, Diller and Nahm [2] defined a variant of Gödel's interpretation where the programs provide a finite set of counterwitnesses, with the requirement that at least one is correct. In the previous example, the program interpreting the contraction rule answers with the set $\{x; y\}$ and does not have to decide which one is correct. The first contribution of this paper is the extension of Spector's bar

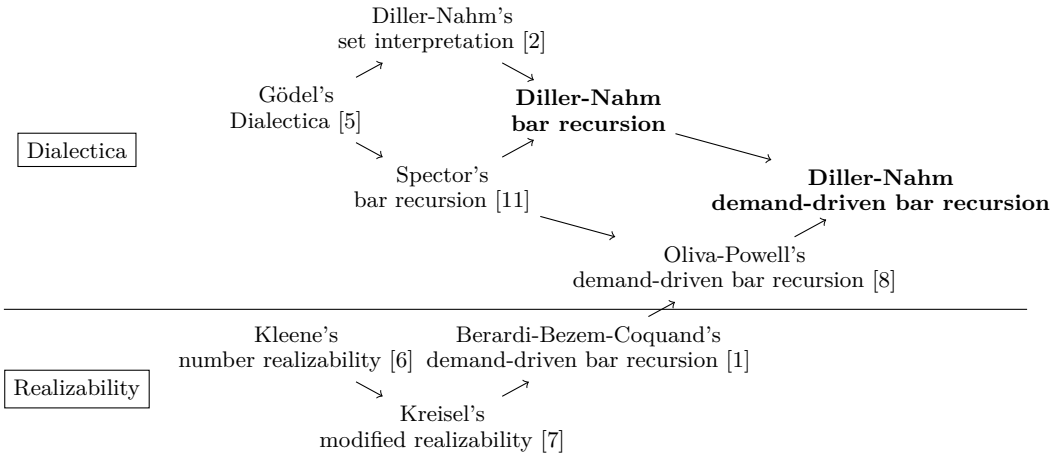

Figure 1 Contributions of the paper, **in bold font**.

recursion to the Diller-Nahm setting. Our operator has a lot in common with the extension of bar recursion to the Herbrand functional interpretation of non-standard arithmetic [3], though there are notable differences which are discussed.

Berardi, Bezem and Coquand [1] adapted Spector's bar recursion from Gödel's Dialectica to Kreisel's modified realizability [7]. Their operator also behaves differently from Spector's original bar recursion as it is demand-driven: it computes the choice sequence in an order that is driven by the environment, rather than in the natural order on natural numbers. This provides a more natural computational interpretation to the axiom of countable choice. More recently, Oliva and Powell [8] adapted Berardi-Bezem-Coquand's operator to Gödel's Dialecica interpretation and obtained a demand-driven bar-recursive interpretation of the axiom of countable choice in this setting. The second contribution of this paper is the definition of a demand-driven bar recursion operator in the Diller-Nahm setting.

Figure 1 summarizes the two contributions of this paper as well as their relationship to the state of the art. An arrow from X to Y means that Y is an extension/refinement/variant of X, and we distinguish elements that take place in the framework of realizability from those that take place in the framework of Dialectica-style functional interpretations.

First, we introduce the logical system and programming language we are working with. Then, we define precisely the Dialectica interpretation of the former into the latter. Finally, we present the two contributions of the paper : first, a variant of Spector's bar recusion in the Diller-Nahm setting, and then a demand-driven version of this operator.

2 Logical System and Programming Language

This section contains the definitions of the logical system and programming language that we will use throughout the paper.

2.1 Logical System

The Dialectica interpretation was originally formulated in a "Hilbert-style" presentation of arithmetic. Moreover, nowadays it is still often presented as a translation on formulas that satisfies a soundness theorem. In this setting, the construction of the witness $\vec{x}$ of the interpretation $\exists\vec{x}\forall\vec{y}A_D(\vec{x},\vec{y})$ of a formula A is hidden in the proof of the soundness theorem.

$$\frac{}{\Gamma \vdash p_i : A_i} \qquad \frac{\Gamma \vdash \pi : \bot}{\Gamma \vdash \bot_A(\pi) : A}$$

$$\frac{\Gamma \vdash \pi : A \Rightarrow \bot}{\Gamma \vdash \neg_i(\pi) : \neg A} \qquad \frac{\Gamma, p_{n+1} : A_{n+1} \vdash \pi : B}{\Gamma \vdash \lambda p_{n+1}.\pi : A_{n+1} \Rightarrow B} \qquad \frac{\Gamma \vdash \pi : A}{\Gamma \vdash \lambda x.\pi : \forall x\, A}\ x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \pi : \neg A}{\Gamma \vdash \neg_e(\pi) : A \Rightarrow \bot} \qquad \frac{\Gamma \vdash \pi_1 : A \Rightarrow B \qquad \Gamma \vdash \pi_2 : A}{\Gamma \vdash \pi_1\, \pi_2 : B} \qquad \frac{\Gamma \vdash \pi : \forall x\, A}{\Gamma \vdash \pi\, t : A\,[t/x]}$$

Figure 2 Rules of first-order logic.

In contrast we present here the Dialectica interpretation as a proof translation, in the line of Pédrot [9, 10]. For that, we define precisely the logical system under consideration as a version of natural deduction in a sequent-calculus presentation (with explicit listing of hypotheses). This system is equipped with proof terms in order to ease the definition of the Dialectica interpretation, but these should be seen as purely syntactic artefacts without any notion of β-reduction or cut-elimination. We then give an inductive definition of the program witnessing the interpretation of a formula.

We work in first-order logic parameterized by a signature that is a set of function symbols (ranged over by metaviariable f) and predicate symbols (ranged over by metaviariable P) with arities:

$$t, u ::= x \mid f(t_1, \dots, t_n) \qquad A, B ::= P(t_1, \dots, t_n) \mid \bot \mid \neg A \mid A \Rightarrow B \mid \forall x\, A$$

We use a primitive negation operator rather than a definition in terms of implication and absurdity. This is completely equivalent but will simplify a lot the functional interpretation in the next sections, especially that of the double-negation shift principle. We consider only a reduced set of connectives with no conjunction or existential quantification in order to simplify the presentation, but these could easily be handled. The rules of the system are given in figure 2, where Γ denotes a context of the form $\vec{p} : \vec{A} \equiv p_1 : A_1, \dots, p_n : A_n$, where we use $\equiv$ throughout the paper for definitions at the meta-level.

Arithmetic

In the Dialectica interpretation, the theory under consideration is that of arithmetic. There are 4 function symbols: `0` or arity 0, `S` of arity 1, $+$ and $\times$ of arity 2, for which we use infix notation. The only predicate symbol is $=$ of arity 2, for which we use infix notation as well.

The axioms of arithmetic are, as usual, those of equality (reflexivity and Leibniz's scheme), the 4 axioms defining addition and multiplication, non-confusion, injectivity of `S` and the induction scheme. They are given in figure 3.

2.2 Programming Language

Gödel's initial interpretation targeted quantifier-free arithmetic at finite types: system T. Nowadays, system T is usually described as primitive recursive functionals, or simply-typed lambda-calculus with natural numbers.

We use a programming language that is a variant of system T where the type of natural numbers is replaced with types σ^* of lists of elements of type σ, for each σ. This type is extensively used for encoding sets that are manipulated in the Diller-Nahm variant of Dialectica. Also, we use it to encode the types of booleans and natural numbers, and the initial segments and partial functions manipulated in our versions of bar recursion.

$$\forall x\,(x = x) \qquad \forall xy\,(x = y \Rightarrow A\,[x/z] \Rightarrow A\,[y/z])$$
$$\forall x\,(x + 0 = x) \qquad \forall xy\,(x + \mathtt{S}\,y = \mathtt{S}\,(x + y))$$
$$\forall x\,(x \times 0 = 0) \qquad \forall xy\,(x \times \mathtt{S}\,y = x \times y + x)$$
$$\forall x\,\neg\,(\mathtt{S}\,x = 0) \qquad \forall xy\,(\mathtt{S}\,x = \mathtt{S}\,y \Rightarrow x = y)$$
$$A\,[\mathtt{0}/x] \Rightarrow \forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x]) \Rightarrow \forall x\,A$$

Figure 3 Axioms of arithmetic.

$$\frac{}{\Gamma \vdash x : \sigma}\ {\scriptstyle x:\sigma\in\Gamma} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}$$
$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash M.1 : \sigma} \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash M.2 : \tau} \qquad \frac{}{\Gamma \vdash \langle\rangle : 1}$$
$$\frac{}{\Gamma \vdash \mathtt{nil} : \sigma^*} \qquad \frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \sigma^*}{\Gamma \vdash M :: N : \sigma^*} \qquad \frac{}{\Gamma \vdash \mathtt{rec} : \tau \to (\sigma \to \sigma^* \to \tau \to \tau) \to \sigma^* \to \tau}$$

Figure 4 Typing rules of the programming language.

For simplicity we also include unit and product types in our programming language. The types and terms are as follows, the typing rules are given in figure 4:

$$\sigma, \tau ::= \sigma \to \tau \,|\, \sigma \times \tau \,|\, 1 \,|\, \sigma^* \qquad M, N ::= x \,|\, \lambda x.M \,|\, M\,N \,|\, \langle M, N\rangle \,|\, M.1 \,|\, M.2 \,|\, \langle\rangle \,|\, \mathtt{nil} \,|\, M :: N \,|\, \mathtt{rec}$$

Term constructions are, in order: variable, abstraction, application, pairing, first and second projections, unit, empty list, list consing and recursion on lists.

The operational semantics of this language is given by the following β-reduction system:

$$(\lambda x.M)\,N \to_\beta M\,[N/x] \qquad \langle M, N\rangle.1 \to_\beta M \qquad \langle M, N\rangle.2 \to_\beta N$$
$$\mathtt{rec}\,P\,Q\,\mathtt{nil} \to_\beta P \qquad \mathtt{rec}\,P\,Q\,(M :: N) \to_\beta Q\,M\,N\,(\mathtt{rec}\,P\,Q\,N)$$

As with Gödel's system T, every program of this simple programming language terminates.

As usual we combine variables bound by λ-abstraction, so $\lambda xy.y\,x$ is a notation for $\lambda x.\lambda y.y\,x$. Also, if some variable is bound in an expression, we allow the use of _ in place of the variable if the variable does not appear free in the term. For example, $\lambda x\_.x$ is a notation for $\lambda xy.x$.

We encode Booleans in our programming language as follows:

$$\mathtt{bool} \equiv 1^* \qquad \mathtt{false} \equiv \mathtt{nil} \qquad \mathtt{true} \equiv \langle\rangle :: \mathtt{nil}$$

With these definitions we can derive the following typing rules (a dashed line means that a rule is admissible):

$$\frac{}{\Gamma \vdash \mathtt{false} : \mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{true} : \mathtt{bool}}$$

We define case analysis on Booleans via case analysis on lists, for which we can derive the expected typing rule:

$$\mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ P \equiv \mathtt{rec}\ P\ (\lambda\_\ \_\ \_.N)\ M$$

$$\frac{\Gamma \vdash M : \mathtt{bool} \qquad \Gamma \vdash N : \sigma \qquad \Gamma \vdash P : \sigma}{\Gamma \vdash \mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ P : \sigma}$$

We also define a few basic operations on Booleans:

$$M \,\&\, N \equiv \mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ \mathtt{false} \qquad \frac{\Gamma \vdash M : \mathtt{bool} \qquad \Gamma \vdash N : \mathtt{bool}}{\Gamma \vdash M \,\&\, N : \mathtt{bool}}$$

$$\neg M \equiv \mathtt{if}\ M\ \mathtt{then}\ \mathtt{false}\ \mathtt{else}\ \mathtt{true} \qquad \frac{\Gamma \vdash M : \mathtt{bool}}{\Gamma \vdash \neg M : \mathtt{bool}}$$

We also encode natural numbers as follows:

$$\mathtt{nat} \equiv 1^* \qquad 0 \equiv \mathtt{nil} \qquad \mathtt{S}\ M \equiv \langle\rangle :: M \qquad \frac{}{\Gamma \vdash 0 : \mathtt{nat}} \qquad \frac{\Gamma \vdash M : \mathtt{nat}}{\Gamma \vdash \mathtt{S}\ M : \mathtt{nat}}$$

Recursion on natural numbers and its expected derivable typing rule are as follows:

$$\mathtt{rec}^{\mathbb{N}}\ M\ N \equiv \mathtt{rec}\ M\ (\lambda\_.N) \qquad \frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \mathtt{nat} \to \sigma \to \sigma}{\Gamma \vdash \mathtt{rec}^{\mathbb{N}}\ M\ N : \mathtt{nat} \to \sigma}$$

We also define a few basic operations on natural numbers:

$$M = N \equiv \mathtt{rec}^{\mathbb{N}}\ \left(\mathtt{rec}^{\mathbb{N}}\ \mathtt{true}\ (\lambda\_\ \_.\mathtt{false})\right)\ \left(\lambda\_ f.\mathtt{rec}^{\mathbb{N}}\ \mathtt{false}\ (\lambda x\_.f\ x)\right)\ M\ N$$

$$\frac{\Gamma \vdash M : \mathtt{nat} \qquad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash M = N : \mathtt{bool}}$$

$$M - N \equiv \mathtt{rec}^{\mathbb{N}}\ (\lambda x.x)\ \left(\lambda\_ f.\mathtt{rec}^{\mathbb{N}}\ 0\ (\lambda x\_.f\ x)\right)\ N\ M \qquad \frac{\Gamma \vdash M : \mathtt{nat} \qquad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash M - N : \mathtt{nat}}$$

$$M < N \equiv \mathtt{S}\ M - N = 0 \qquad \frac{\Gamma \vdash M : \mathtt{nat} \qquad \Gamma \vdash N : \mathtt{nat}}{\Gamma \vdash M < N : \mathtt{bool}}$$

3 Diller-Nahm interpretation

We define in this section the Diller-Nahm variant [2] of the Dialectica interpretation [5]. We present the interpretation as an explicit translation from proofs to programs and since we work with natural deduction we handle contexts similarly to Pédrot [9, 10].

We define some notations that will make the interpretation easier to read. First, since we need an interpretation for the *ex falso quodlibet* principle, we introduce dummy terms at each type:

$$\square_{\sigma \to \tau} \equiv \lambda\_.\square_\tau \qquad \square_{\sigma \times \tau} \equiv \langle \square_\sigma, \square_\tau \rangle \qquad \square_1 \equiv \langle\rangle \qquad \square_{\sigma^*} \equiv \mathtt{nil}$$

Second, since we work in the Diller-Nahm variant of the Dialectica interpretation in which several witnesses can be collected for a single formula, we extensively use an encoding of sets as lists. Note that we impose no restriction on this encoding, so that a single element can appear several times and the order of elements is of no importance. We use set notations as follows:

$$\{\sigma\} \equiv \sigma^* \qquad \{M_1; M_2; \dots; M_n\} \equiv M_1 :: M_2 :: \cdots :: M_n :: \mathtt{nil}$$

$$\frac{\Gamma \vdash M_1 : \sigma \qquad \Gamma \vdash M_2 : \sigma \qquad \cdots \qquad \Gamma \vdash M_n : \sigma}{\Gamma \vdash \{M_1; M_2; \dots; M_n\} : \{\sigma\}}$$

$$M \cup N \equiv \mathtt{rec}\ M\ (\lambda x\_ z.x :: z)\ N \qquad \frac{\Gamma \vdash M : \{\sigma\} \qquad \Gamma \vdash N : \{\sigma\}}{\Gamma \vdash M \cup N : \{\sigma\}}$$

$$\{N \mid x \in M \wedge P\} \equiv \mathtt{rec}\,\mathtt{nil}\,(\lambda x\_z.\mathtt{if}\,P\,\mathtt{then}\,N :: z\,\mathtt{else}\,z)\,M$$

$$\frac{\Gamma \vdash M : \{\sigma\} \qquad \Gamma, x : \sigma \vdash N : \tau \qquad \Gamma, x : \sigma \vdash P : \mathtt{bool}}{\Gamma \vdash \{N \mid x \in M \wedge P\} : \{\tau\}}$$

$$\{N \mid x \in M\} \equiv \{N \mid x \in M \wedge \mathtt{true}\} \qquad \frac{\Gamma \vdash M : \{\sigma\} \qquad \Gamma, x : \sigma \vdash N : \tau}{\Gamma \vdash \{N \mid x \in M\} : \{\tau\}}$$

$$\bigcup M \equiv \mathtt{rec}\,\mathtt{nil}\,(\lambda x\_z.x \cup z)\,M \qquad \frac{\Gamma \vdash M : \{\{\sigma\}\}}{\Gamma \vdash \bigcup M : \{\sigma\}}$$

$$\forall x \in M,\, N \equiv \mathtt{rec}\,\mathtt{true}\,(\lambda x\_z.N \,\&\, z)\,M \qquad \frac{\Gamma \vdash M : \{\sigma\} \qquad \Gamma, x : \sigma \vdash N : \mathtt{bool}}{\Gamma \vdash \forall x \in M,\, N : \mathtt{bool}}$$

$$[0; M[\equiv \mathtt{rec}^{\mathbb{N}}\,\mathtt{nil}\,(\lambda xy.x :: y)\,M \qquad \frac{\Gamma \vdash M : \mathtt{nat}}{\Gamma \vdash [0; M[: \{\mathtt{nat}\}}$$

In order to ensure the preservation of computations via the Dialectica interpretation, Pédrot axiomatized abstract multisets via a series of laws that these should satisfy. Here we work with a concrete implementation of these abstract multisets as lists. To simplify the presentation we do not define any reduction on proofs, and therefore we do not aim at preservation of computations. Consequently we do not define our implementation of sets in such a way that they satisfy the monadic and distributive laws of Pédrot. Choosing carefully the implementation should allow to satisfy some of these laws. Some other (in particular commutativity of union) would be more technical to implement in a terminating language.

The Dialectica interpretation provides a computational interpretation of proofs. Therefore, the meaning of such an interpretation is defined up to equivalence of programs: witnesses of formulas are equivalence classes of programs. In our case this equivalence is β-equivalence $=_\beta$, defined as the reflexive symmetric transitive contextual closure of β-reduction $\to_\beta$. When this is clear from the context we will describe an equivalence class by one of its representatives.

Since we work in the context of first-order logic, the whole interpretation is parameterized by an interpretation of the function and predicate symbols. In the general case, the type of programs interpreting first-order terms should be a parameter of the interpretation as well. However, since in the present work we only consider arithmetic and its extensions, we fix this type to be $\mathtt{nat}$, the type of natural numbers. The interpretation is therefore parameterized by the interpretations of function and predicate symbols. The interpretation of a function symbol f of arity n is a program:

$$f^\bullet : \mathtt{nat} \to \ldots \to \mathtt{nat} \to \mathtt{nat} \ (n \text{ arguments})$$

and the interpretation of a predicate symbol P of arity n is a set:

$$P^\bullet \subseteq \mathtt{nat}^n_{/=_\beta}$$

of n-tuples of (equivalence classes of) programs of type $\mathtt{nat}$, where $\mathtt{nat}_{/=_\beta}$ denotes the set of equivalence classes of terms of type $\mathtt{nat}$. This interpretation is extended to all first-order terms as follows, so that $FV\,(t) = FV\,(t^\bullet)$:

$$x^\bullet \equiv x \qquad (f\,(t_1, \ldots, t_n))^\bullet \equiv f^\bullet\, t_1^\bullet \ldots t_n^\bullet$$

Following Pédrot we define for each formula A the type of its witnesses $\underline{A}$ (Pédrot's $\mathbb{W}\,(A)$) and the type of its counterwitnesses $\overline{A}$ (Pédrot's $\mathbb{C}\,(A)$). Since we use a non-dependently-typed language, the types of witnesses and counterwitnesses of atomic predicates $P\,(t_1, \ldots, t_n)$ cannot depend on $t_1^\bullet, \ldots, t_n^\bullet$, even though the truth of these predicates does depend on them.

witnesses	counterwitnesses
$\underline{P\,(t_1,\dots,t_n)} \equiv 1$	$\overline{P\,(t_1,\dots,t_n)} \equiv 1$
$\underline{\bot} \equiv 1$	$\overline{\bot} \equiv 1$
$\underline{\neg A} \equiv \underline{A} \to \{\overline{A}\}$	$\overline{\neg A} \equiv \underline{A}$
$\underline{A \Rightarrow B} \equiv (\underline{A} \to \underline{B}) \times (\underline{A} \to \overline{B} \to \{\overline{A}\})$	$\overline{A \Rightarrow B} \equiv \underline{A} \times \overline{B}$
$\underline{\forall x\, A} \equiv \mathtt{nat} \to \underline{A}$	$\overline{\forall x\, A} \equiv \mathtt{nat} \times \overline{A}$

Figure 5 Types of witnesses and counterwitnesses of formulas.

The interpretation itself therefore does not depend on the interpretations $P^\bullet$ of predicates P, but the correctness of the interpretation will be expressed via an orthogonality relation afterwards. The types of witnesses and counterwitnesses of formulas are defined inductively in figure 5.

We now lift this interpretation of formulas to an interpretation of proofs, so that a proof of the sequent $\vec{p} : \vec{A} \vdash \pi : A$ (where $\vec{p} : \vec{A} \equiv p_1 : A_1, \dots, p_n : A_n$) is interpreted as a collection of programs with the following types, where $FV\left(\vec{A}, A\right) \subseteq \vec{x}$:

$$\begin{array}{cc} & \vec{x} : \mathtt{nat}, \vec{p} : \underline{\vec{A}} \vdash \pi_{p_1} : \overline{A} \to \{\overline{A_1}\} \\ \vec{x} : \mathtt{nat}, \vec{p} : \underline{\vec{A}} \vdash \pi^\bullet : \underline{A} & \vdots \\ & \vec{x} : \mathtt{nat}, \vec{p} : \underline{\vec{A}} \vdash \pi_{p_n} : \overline{A} \to \{\overline{A_n}\} \end{array}$$

The interpretation of proofs is given in figure 6, where some type superscripts have been added to ease the reading. Note that $(\pi_1\,\pi_2)_{p_j}$ involves the union of counterwitnesses comming from both π_1 and π_2.

$p_i^\bullet \equiv p_i$	$p_{i_{p_j}} \equiv \begin{cases} \lambda q^{\overline{A_i}}.\,\{q\} & \text{if } j = i \\ \lambda \_^{\overline{A_i}}.\,\{\} & \text{if } j \neq i \end{cases}$
$(\bot_A\,(\pi))^\bullet \equiv \Box_{\underline{A}}$	$(\bot_A\,(\pi))_{p_j} \equiv \lambda \_^{\overline{A}}.\pi_{p_j}\,\langle\rangle$
$(\neg_i\,(\pi))^\bullet \equiv \lambda x.\pi^\bullet.2\,x\,\langle\rangle$	$(\neg_i\,(\pi))_{p_j} \equiv \lambda q^{\overline{\neg A}}.\pi_{p_j}\,\langle q, \langle\rangle\rangle$
$(\neg_e\,(\pi))^\bullet \equiv \langle \lambda \_.\,\langle\rangle\,, \lambda x.\lambda \_.\pi^\bullet\,x\rangle$	$(\neg_e\,(\pi))_{p_j} \equiv \lambda q^{\overline{A \Rightarrow \bot}}.\pi_{p_j}\,q.1$
$(\lambda p_{n+1}.\pi)^\bullet \equiv \left\langle \lambda p_{n+1}.\pi^\bullet, \lambda p_{n+1}.\pi_{p_{n+1}} \right\rangle$	$(\lambda p_{n+1}.\pi)_{p_j} \equiv \lambda q^{\overline{A_{n+1} \Rightarrow B}}.\left(\lambda p_{n+1}.\pi_{p_j}\right) q.1\,q.2$
$(\pi_1\,\pi_2)^\bullet \equiv \pi_1^\bullet.1\,\pi_2^\bullet$	$(\pi_1\,\pi_2)_{p_j} \equiv \lambda q^{\overline{B}}.\left(\pi_{1_{p_j}}\,\langle \pi_2^\bullet, q\rangle\right) \cup \left(\bigcup \left\{ \pi_{2_{p_j}}\,r \,\middle|\, r \in \pi_1^\bullet.2\,\pi_2^\bullet\,q \right\}\right)$
$(\lambda x.\pi)^\bullet \equiv \lambda x.\pi^\bullet$	$(\lambda x.\pi)_{p_j} \equiv \lambda q^{\overline{\forall x\, A}}.\left(\lambda x.\pi_{p_j}\right) q.1\,q.2$
$(\pi\,t)^\bullet \equiv \pi^\bullet\,t^\bullet$	$(\pi\,t)_{p_j} \equiv \lambda q^{\overline{A[t/x]}}.\pi_{p_j}\,\langle t^\bullet, q\rangle$

Figure 6 Diller-Nahm interpretation of proofs.

$$
\begin{aligned}
M \bot_{P(t_1,\dots,t_n)} N &\text{ iff } (t_1^\bullet, \dots, t_n^\bullet) \in P^\bullet \\
M \bot_{\bot} N &\text{ is false} \\
M \bot_{\neg A} N &\text{ iff } N \bot_A P \text{ is false for some } P \in M\,N \\
M \bot_{A \Rightarrow B} N &\text{ iff } N.1 \bot_A P \text{ for all } P \in M.2\,N.1\,N.2 \text{ implies } M.1\,N.1 \bot_B N.2 \\
M \bot_{\forall x\, A} N &\text{ iff } M\,N.1 \bot_{A[N.1/x]} N.2
\end{aligned}
$$

Figure 7 Definition of the orthogonality relation.

As explained before, the first-order structure is not reflected in this interpretation, as $\langle\rangle$ has the type interpreting any atomic formula, including $\bot$. Similarly, $\lambda\_.\,\{\}$ has the type interpreting $\neg A$ for any formula A. Therefore, typing is not sufficient to ensure correctness of the interpretation, both because of the first-order aspect of our logic and because the Diller-Nahm variant manipulates "false" witnesses and counterwitnesses ($M \bot_{\neg A} N$ only requires $\neg\,(N \bot_A P)$ for *some* $P \in M\,N$). In order to express correctness, we define an orthogonality relation between (equivalence classes of) witnesses and (equivalence classes of) counterwitnesses of formulas that corresponds Gödel's A_D. When viewing the interpretation as games, this orthogonality relation represents the outcome of the game: a witness is orthogonal to a counterwitness whenever the witness wins against the counterwitness. In the following, if $N : \{\sigma\}$ then we write "$M \in N$" as a shorthand for "$N =_\beta \{N_1; \dots; N_n\}$ and $M =_\beta N_i$ for some i". If $M : \underline{A}$ and $N : \overline{A}$, then $M \bot_A N$ is defined in figure 7.

With this orthogonality relation at hand we can now formulate the correctness theorem of the interpretation:

▶ **Theorem 1.** *If $\vec{p} : \vec{A} \vdash \pi : B$ is derivable in first-order logic and $FV\left(\vec{A}, B\right) \subseteq \vec{x}$, then for all $\vec{X} : \vec{\mathtt{nat}}$, $\vec{P} : \underline{\vec{A}}$ and $Q : \overline{B}$:*

if for all i and for all $R \in \pi_{p_i}\left[\vec{P}, \vec{X}/\vec{p}, \vec{x}\right] Q$ we have $P_i \bot_{A_i} R$, then $\pi^\bullet\left[\vec{P}, \vec{X}/\vec{p}, \vec{x}\right] \bot_B Q$

Proof. By induction on π, see Pédrot [10]. ◀

This theorem states that whenever $\vec{P}$ are witnesses of $\vec{A}$ and Q is a counterwitness of B, if for each i P_i wins against every counterwitness of A_i computed by π_{p_i} (using $\vec{P}$ and Q), then the witness of B computed by $\pi^\bullet$ (using $\vec{P}$) wins against Q.

Arithmetic

The interpretations of function symbols of arithmetic are the expected implementations of operations on natural numbers in our programming language, and the only predicate of arithmetic, equality, is interpreted as:

$$=^\bullet \equiv \{(M, M) \mid M : \mathtt{nat}\}$$

The interpretation of the axioms of arithmetic, except for induction, is relatively straightforward and is given in figure 8. The case of induction is more complex. Its interpretation is of the form $\langle \lambda p.\, \langle \lambda q.a, \lambda qr.c\,[r.1, r.2/z, r]\rangle, \lambda ps.d\,[s.1, s.2.1, s.2.2/q, z, r]\rangle$, where we define a, b, c and d as follows:

$\forall x\,(x = x)$	$\lambda\_.\,\langle\rangle$
$\forall xy\,(x = y \Rightarrow A\,[x/z] \Rightarrow A\,[y/z])$	$\lambda\_\,\_.\,\langle\lambda\_.\,(\lambda p.p)^\bullet, \lambda\_\,\_.\,\{\langle\rangle\}\rangle$
$\forall x\,(x + 0 = x)$	$\lambda\_.\,\langle\rangle$
$\forall xy\,(x + \mathtt{S}\,y = \mathtt{S}\,(x + y))$	$\lambda\_.\,\langle\rangle$
$\forall x\,(x \times 0 = 0)$	$\lambda\_.\,\langle\rangle$
$\forall xy\,(x \times \mathtt{S}\,y = x \times y + x)$	$\lambda\_.\,\langle\rangle$
$\forall x \neg\,(\mathtt{S}\,x = 0)$	$\lambda\_\,\_.\,\{\langle\rangle\}$
$\forall xy\,(\mathtt{S}\,x = \mathtt{S}\,y \Rightarrow x = y)$	$\lambda\_\,\_.\,\langle\lambda\_.\,\langle\rangle, \lambda\_\,\_.\,\{\langle\rangle\}\rangle$

Figure 8 Interpretation of the axioms of arithmetic, except induction.

$$
\begin{aligned}
&p : \underline{A\,[0/x]}, q : \underline{\forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x])} && \vdash a : \underline{\forall x\,A}\\
&p : \underline{A\,[0/x]}, q : \underline{\forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x])}, z : \mathtt{nat}, r : \overline{A} && \vdash b : \mathtt{nat} \to \left\{\overline{A}\right\}\\
&p : \underline{A\,[0/x]}, q : \underline{\forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x])}, z : \mathtt{nat}, r : \overline{A} && \vdash c : \left\{\overline{A\,[0/x]}\right\}\\
&p : \underline{A\,[0/x]}, q : \underline{\forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x])}, z : \mathtt{nat}, r : \overline{A} && \vdash d : \left\{\overline{\forall x\,(A \Rightarrow A\,[\mathtt{S}\,x/x])}\right\}
\end{aligned}
$$

$$
\begin{aligned}
a &\equiv \mathtt{rec}^{\mathbb{N}}\,p\,(\lambda xy.\,(q\,x)\,.1\,y)\\
b &\equiv \mathtt{rec}^{\mathbb{N}}\,\{r\}\,\left(\lambda xy.\,\bigcup\{(q\,(z - (\mathtt{S}\,x)))\,.2\,(a\,(z - (\mathtt{S}\,x)))\,g \mid g \in y\}\right)\\
c &\equiv b\,z\\
d &\equiv \bigcup\{\{\langle n, \langle a\,n, g\rangle\rangle \mid g \in b\,(z - (\mathtt{S}\,n))\} \mid n \in [0; z[\}
\end{aligned}
$$

a performs standard recursion on natural numbers, while $b\,0, \ldots, b\,z$ compute sets of counterwitnesses to $A\,[n/x]$ for n from z down to 0, starting with counterwitness r for $A\,[z/x]$. The following lemma is the core of correctness of this interpretation:

▶ **Lemma 2.** *Let* $P : \underline{A\,[0/z]}$, $Q : \underline{\forall x\,(A\,[x/z] \Rightarrow A\,[\mathtt{S}\,x/z])}$, $Z : \mathtt{nat}$ *and* $R : \overline{A}$, *and let* $A \equiv a\,[P, Q/p, q]$, $B \equiv b\,[P, Q, Z, R/p, q, z, r]$, $C \equiv c\,[P, Q, Z, R/p, q, z, r]$ *and* $D \equiv d\,[P, Q, Z, R/p, q, z, r]$.

if for any $S \in C$ *we have* $P \perp_{A[0/z]} S$
and if for any $S \in D$ *we have* $Q \perp_{\forall x(A[x/z] \Rightarrow A[\mathtt{S}\,x/z])} S$
then for any $N \in [0; \mathtt{S}\,Z[$ *and* $S \in B\,(Z - N)$ *we have* $A\,N \perp_{A[N/z]} S$

Proof. By induction on N from 0 to Z. ◀

With this lemma at hand it becomes easy to prove correctness of the interpretation of arithmetic:

▶ **Theorem 3.** *For any axiom* A *of arithmetic interpreted as* $a : \underline{A}$, *for all* $M : \overline{A}$, $a \perp_A M$.

4 Double-Negation Shift

This section is devoted to the main contributions of this paper: bar recursive interpretations of the double-negation shift (DNS) principle in the Diller-Nahm variant of the Dialectica interpretation.

The negative translation provides an interpretation of classical logic into intuitionistic logic, at the expense of changing the formula proven: if A is provable in classical logic then $A^\neg$ is provable in intuitionistic logic, where $A^\neg$ is the negative translation of A defined inductively on the structure of A. This negative translation can be extended to arithmetic because the negative translation of every axiom of arithmetic is derivable in intuitionistic arithmetic. Combined with Friedman's translation [4] one can even eliminate classical logic without changing the formula in the case of Π^0_2 formulas, which means that classical arithmetic is conservative over intuitionistic arithmetic for this class of formulas.

When it comes to richer theories, however, negative translation can fail. In particular, the negative translation of the axiom of choice fails to be an intuitionistic consequence of the axiom of choice. In order to recover the negative translation on such theories, one can use the DNS principle:

$$\forall x \neg\neg A \Rightarrow \neg\neg \forall x\, A$$

Using this principle, $A^\neg$ becomes an intuitionistic consequence of A for any formula A. This allows the extension of the negative translation from plain logic to any theory. In particular, negative translation can then be applied to classical analysis using DNS on natural numbers. Combining the negative translation with an extension of the Dialectica interpretation to the DNS principle on natural numbers, one obtains a computational interpretation of classical analysis.

In the past, several versions of bar recursion have been used to provide a Dialectica interpretation of the DNS principle, including Spector's original bar recursion [11], as well as Oliva-Powell's demand-driven variant [8], inspired by Berardi-Bezem-Coquand's demand-driven operator [1]. In the following sections we give new versions of bar recursion that are compatible with the Diller-Nahm variant of the Dialectica interpretation. This extends the Dialectica interpretation to classical theories with non-decidable atomic predicates.

4.1 Diller-Nahm Bar Recursion

In this section we define an adaptation of Spector's original bar recursion to the Diller-Nahm setting. Bar recursion builds incrementally finite aproximations to a witness of $\forall x\, A(x)$. These finite aproximations are sequences of witnesses for $A(0), \ldots, A(n-1)$ for some $n \in \mathbb{N}$. We encode these approximations as lists of programs of type $\underline{A}$, so that if for $i < n$, $a_i : \underline{A}$ is a witness for $A(i)$, then $a_{n-1} :: \ldots :: a_1 :: a_0 :: \mathtt{nil}$ is such an approximation. We define the following notation for the length of an approximation:

$$|M| \equiv \mathtt{rec}\,0\,(\lambda\_\,\_\,z.\mathtt{S}\,z)\,M \qquad \frac{\Gamma \vdash M : \sigma^*}{\Gamma \vdash |M| : \mathtt{nat}}$$

We also define the "dummy" completion of an approximation $a_{n-1} :: \ldots :: a_1 :: a_0 :: \mathtt{nil}$ into a full sequence mapping $i < n$ to a_i and $i \geq n$ to $\square_\sigma$:

$$M^+ \equiv \lambda n.\mathtt{rec}\,\square_\sigma\,(\lambda xyz.\mathtt{if}\,n = |y|\,\mathtt{then}\,x\,\mathtt{else}\,z)\,M \qquad \frac{\Gamma \vdash M : \sigma^*}{\Gamma \vdash M^+ : \mathtt{nat} \to \sigma}$$

Finally, each relation between $\mathtt{nat}$ and σ can be turned into a function from $\mathtt{nat}$ to $\{\sigma\}$:

$$\widehat{M} \equiv \lambda x.\, \{y.2 \,|\, y \in M \wedge y.1 = x\} \qquad \frac{\Gamma \vdash M : \{\mathtt{nat} \times \sigma\}}{\Gamma \vdash \widehat{M} : \mathtt{nat} \to \{\sigma\}}$$

Computing the required type for a program interpreting the DNS principle gives:

$$\underline{\forall x\, \neg\neg A \Rightarrow \neg\neg \forall x\, A} \equiv (\underline{\forall x\, \neg\neg A} \to \underline{\neg\forall x\, A} \to \{\mathtt{nat} \to \underline{A}\}) \times \left(\underline{\forall x\, \neg\neg A} \to \underline{\neg\forall x\, A} \to \left\{\mathtt{nat} \times \left(\underline{A} \to \left\{\overline{A}\right\}\right)\right\}\right)$$

$$\underline{\forall x\, \neg\neg A} \equiv \mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\} \qquad \underline{\neg\forall x\, A} \equiv (\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}$$

The interpretation is therefore a pair of two programs a and b with respective types:

$$a : \left(\mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}\right) \to \left((\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}\right) \to \{\mathtt{nat} \to \underline{A}\}$$
$$b : \left(\mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}\right) \to \left((\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}\right) \to \left\{\mathtt{nat} \times \left(\underline{A} \to \left\{\overline{A}\right\}\right)\right\}$$

Unfolding the definitions we see that in order to satisfy correctness, a and b should satisfy for any arguments $P : \mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}$ and $Q : (\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}$:

if for any $M \in b\, P\, Q, P \bot_{\forall x \neg\neg A} M$
then there is some $M \in a\, P\, Q$ such that for all $N \in Q\, M, M \bot_{\forall x\, A} N$

In other words, we should put in $b\, P\, Q$ counterwitnesses of $\forall x \neg\neg A$ such that whenever P wins against all of them, we can use that hypothesis to ensure that $a\, P\, Q$ contains at least an element M that wins against every element of $Q\, M$. The elements of $a\, P\, Q$ will be "dummy" completions of finite approximations, that is, they will be of the form S^+ for $S : \underline{A}^*$. Given such a finite approximation S, if $Q\, S^+$ contains some $\langle N, R\rangle$ then $S^+\, N$ should win against R. But if $N \geq |S|$ then $S^+\, N$ is the dummy value $\square_{\underline{A}}$, meaning that S is not a sufficiently precise approximation in order to be correct. The idea is then to extend S with a value at point $|S|$, using $P\, |S|$ (which is a witness for $\neg\neg A\, [|S|\, /x]$) and continue the process. Computation stops when a sufficiently precise approximation is found, that is, an approximation S such that for all $\langle N, R\rangle \in Q\, S^+$, we have $N < |S|$.

Our variant $\mathtt{dnbr}$ of bar recursion is an operator that, given an approximation in $\underline{A}^*$, computes extensions of this approximation in such a way that at least one of these extensions is correct. Then a is obtained by running this operator on the empty approximation. In order to distinguish sufficiently precise approximations we use the notation:

$$S \Subset Q \equiv \forall z \in Q\, S^+,\, z.1 < |S| \qquad \frac{\Gamma \vdash Q : (\mathtt{nat} \to \sigma) \to \{\mathtt{nat} \times \tau\} \qquad \Gamma \vdash S : \sigma^*}{\Gamma \vdash S \Subset Q : \mathtt{bool}}$$

In order to facilitate the reading we also use the notation:

$$\mathtt{let}\, f = M\, \mathtt{in}\, N \equiv (\lambda f.N)\, M \qquad \frac{\Gamma \vdash M : \sigma \qquad \Gamma, f : \sigma \vdash N : \tau}{\Gamma \vdash \mathtt{let}\, f = M\, \mathtt{in}\, N \equiv (\lambda f.N)\, M : \tau}$$

We now extend our programming language with this operator and its reduction rule:

$$\mathtt{dnbr} : \left(\mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}\right) \to \left((\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}\right) \to \underline{A}^* \to \{\underline{A}^*\}$$
$$\mathtt{dnbr}\, P\, Q\, S \to_\beta \{S\} \cup \mathtt{if}\, S \Subset Q\, \mathtt{then}\, \{\}$$
$$\mathtt{else}\, \mathtt{let}\, f = \lambda x.\mathtt{dnbr}\, P\, Q\, (x :: S)\, \mathtt{in}$$
$$\bigcup \left\{ f\, x \,\middle|\, x \in P\, |S| \left(\lambda y. \bigcup \left\{ \widehat{Q\, t^+}\, |S| \,\middle|\, t \in f\, y \wedge t \Subset Q \right\}\right)\right\}$$

Observe that given P, Q and an approximation S as input, if for every $\langle N, R\rangle \in Q\,S^+$ we have $N < |S|$ then $S \Subset Q$ so S is a potentially correct approximation and the recursive calls stop. Otherwise, new recursive calls are made on extensions of S with values obtained from $P\,|S|$.

A fundamental result about this operator is that for any P, Q and S, $\mathtt{dnbr}\,P\,Q\,S$ has a normal form. Otherwise, there would be an infinite sequence of programs $M_0, M_1, \ldots$ such that for any k there exists $\langle N, R\rangle \in Q\,(M_{k-1} :: \ldots :: M_1 :: M_0 :: \mathtt{nil})^+$ with $N > k$. But continuity of $\lambda f.\,\{x.1 \mid x \in Q\,f\} : (\mathtt{nat} \to \underline{A}) \to \{\mathtt{nat}\}$ implies that the sequence:

$$\left\{x.1 \mid x \in Q\,\mathtt{nil}^+\right\}, \left\{x.1 \;\middle|\; x \in Q\,(M_0 :: \mathtt{nil})^+\right\}, \ldots, \left\{x.1 \;\middle|\; x \in Q\,(M_{k-1} :: \ldots :: M_1 :: M_0 :: \mathtt{nil})^+\right\}, \ldots$$

is ultimately constant, hence the contradiction.

With this operator at hand we can finally define a and b, and therefore the interpretation of the DNS principle in the Diller-Nahm variant of the Dialectica interpretation:

$$a \equiv \lambda pq.\,\left\{s^+ \mid s \in \mathtt{dnbr}\,p\,q\,\mathtt{nil} \wedge s \Subset q\right\}$$

$$b \equiv \lambda pq.\,\left\{\left\langle |s|, \lambda x.\bigcup\left\{\widehat{q\,t^+}\,|s| \;\middle|\; t \in \mathtt{dnbr}\,p\,q\,(x :: s) \wedge t \Subset q\right\}\right\rangle \;\middle|\; \begin{array}{l} s \in \mathtt{dnbr}\,p\,q\,\mathtt{nil} \\ \quad \wedge \neg\, s \Subset q \end{array}\right\}$$

Correctness of this interpretation relies on the following lemma:

▶ **Lemma 4.** *Let* $P : \mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}$ *and* $Q : (\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}$ *and let* $D \equiv \mathtt{dnbr}\,P\,Q$. *If for any* $M \in b\,P\,Q$ *we have* $P \bot_{\forall x \neg\neg A} M$, *then there exists a sequence of programs* $M_0, \ldots, M_{n-1}$ *such that if we write* $S_i \equiv M_{i-1} :: \ldots :: M_0 :: \mathtt{nil}$*:*

(a) $D\,\mathtt{nil} = D\,S_0 \supseteq D\,S_1 \supseteq \ldots \supseteq D\,S_n$

(b) *for all* $i \leq n$, $S \in D\,S_i$ *and* $N \in Q\,S^+$, *if* $S \Subset Q$ *and* $N.1 < i$ *then* $S_i^+\,N.1 \bot_{A[N.1/x]} N.2$

(c) *for all* $N \in Q\,S_n^+$, $N.1 < n$

Proof. $S_0 = \mathtt{nil}$ trivially satisfies (b). Suppose now $M_0, \ldots, M_{k-1}$ already defined, and satisfying (a) and (b). If for all $N \in Q\,S_k^+$, $N.1 < k$, then we choose $n = k$ and we are done since (c) is verified. Otherwise we have:

$$D\,S_k =_\beta \{S_k\} \cup \bigcup\left\{D\,(x :: S_k) \;\middle|\; x \in P\,k\left(\lambda y.\bigcup\left\{\widehat{Q\,t^+}k \;\middle|\; t \in D\,(y :: S_k) \wedge t \Subset Q\right\}\right)\right\}$$

and we have $S_k \in D\,S_k \subseteq C\,\mathtt{nil}$, so:

$$\left\langle k, \lambda x.\bigcup\left\{\widehat{Q\,t^+}k \;\middle|\; t \in D\,(x :: S_k) \wedge t \Subset Q\right\}\right\rangle \in b\,P\,Q$$

and therefore by hypothesis:

$$P \bot_{\forall x \neg\neg A} \left\langle k, \lambda x.\bigcup\left\{\widehat{Q\,t^+}k \;\middle|\; t \in D\,(x :: S_k) \wedge t \Subset Q\right\}\right\rangle$$

which means that there must be some:

$$M \in P\,k\left(\lambda x.\bigcup\left\{\widehat{Q\,t^+}k \;\middle|\; t \in D\,(x :: S_k) \wedge t \Subset Q\right\}\right)$$

such that for all:

$$N \in \bigcup\left\{\widehat{Q\,t^+}k \;\middle|\; t \in D\,(M :: S_k) \wedge t \Subset Q\right\}$$

we have $M \bot_{A[k/x]} N$. Define $M_k \equiv M$. We have (a) since $D\, S_{k+1} \equiv D\,(M_k :: S_k) \subseteq D\, S_k$. For (b), let $S \in D\, S_{k+1}$ and $N \in Q\, S^+$ such that $S \Subset Q$ and $N.1 < \mathtt{S}\, k$. If $N.1 < k$ then $S^+_{k+1}\, N.1 =_\beta S^+_k\, N.1$ and we are done by hypothesis on S_k. Otherwise $N.1 =_\beta k$ and $S^+_{k+1}\, N.1 =_\beta M_k$. But then we can prove that:

$$N.2 \in \bigcup \left\{ \widehat{Q\, t^+}\, k \;\middle|\; t \in D\,(M_k :: S_k) \wedge t \Subset Q \right\}$$

so we obtain $M_k \bot_{A[k/x]} N.2$ by property of M_k. ◀

Finally we can conclude that our interpretation of the DNS principle is correct:

▶ **Theorem 5.** *For any* $P : \mathtt{nat} \to (\underline{A} \to \{\overline{A}\}) \to \{\underline{A}\}$ *and* $Q : (\mathtt{nat} \to \underline{A}) \to \{\mathtt{nat} \times \overline{A}\}$*:*

if for any $M \in b\, P\, Q$ *we have* $P \bot_{\forall x \neg\neg A} M$

then there is some $M' \in a\, P\, Q$ *such that for all* $N \in Q\, M'$ *we have* $M' \bot_{\forall x\, A} N$

Proof. Apply the previous lemma and take $M' = S^+_n$. The conclusion follows from properties 2 and 3. ◀

We now discuss the relationship between our operator and Herbrand bar recursion [3] (`hBR`). In the Diller-Nahm interpretation, finite sets appear only in the return type of the "reverse component" of the witnesses of implication. In the Herbrand functional interpretation, handling of standard and non-standard elements requires the use of finite sets in several places. In particular both components of the witnesses of implication are finite sets. Nevertheless, `hBR` is very similar in principle to `dnbr`. But besides technical differences there is also a conceptual difference: `dnbr` handles the argument Q of type $(\mathtt{nat} \to \underline{A}) \to \{\mathtt{nat} \times \overline{A}\}$ more carefully than `hBR`. Indeed, for each t, $Q\, t^+$ is a set of pairs $\langle N, R\rangle$ where R is a potential counterwitness of $A\,[N/x]$. However, since `dnbr` on S extends S at point $|S|$, the only useful counterwitnesses are those of the form $\langle |S|, R\rangle$, which are those we get via $\widehat{Q\, t^+}\, |S|$. Conversely, in `hBR`, Q is split in two components: $q \equiv \lambda f.\, \{x.2 \,|\, x \in q\, f\}$ and $\omega \equiv \lambda f.\mathtt{max}\, \{x.1 \,|\, x \in q\, f\}$. Therefore in $q\, f$ the information about the N in $A\,[N/x]$ for which the element of $q\, f$ is a counterwitness is lost. When extending S at point $|S|$, `hBR` considers all the elements of $q\, S^+$, that is, $\{x.2 \,|\, x \in Q\, S^+\}$, instead of only $\{x.2 \,|\, x \in Q\, S^+ \wedge x.1 = |S|\}$ (as `dnbr` does). In that sense `hBR` is less optimal that `dnbr`.

4.2 Diller-Nahm Demand-Driven Bar Recursion

In this last section we present the second contribution of the paper: a demand-driven version of bar recursion in the context of the Diller-Nahm variant of the Dialectica interpretation. The first demand-driven bar recursion was defined by Berardi, Bezem and Coquand in the context of Kreisel's realizability, but this was only recently adapted to the Dialectica interpretation by Oliva and Powell. Here we take inspiration both from Oliva and Powell's operator, and from the version of bar recursion presented in the previous section.

Instead of working on initial segments, demand-driven bar recursion works on partial sequences that may be defined at arbitrary points. We encode these partial functions as lists of points, that is, lists of pairs of a natural number and its image. We define the empty function and the extension of a function with a new value as follows:

$$\epsilon \equiv \mathtt{nil} \qquad \frac{}{\Gamma \vdash \epsilon : (\mathtt{nat} \times \sigma)^*}$$

$$M\,[N \mapsto P] \equiv \langle N, P\rangle :: M \qquad \frac{\Gamma \vdash M : (\mathtt{nat} \times \sigma)^* \qquad \Gamma \vdash N : \mathtt{nat} \qquad \Gamma \vdash P : \sigma}{\Gamma \vdash M\,[N \mapsto P] : (\mathtt{nat} \times \sigma)^*}$$

Similarly to the $\_^+$ operator on initial segments, we define the "dummy" completion of a partial function:

$$M^\dagger \equiv \lambda n.\mathtt{rec}\,\square_\sigma\,(\lambda x\_z.\mathtt{if}\,x.1 = n\,\mathtt{then}\,x.2\,\mathtt{else}\,z)\,M \qquad \frac{\Gamma \vdash M : (\mathtt{nat} \times \sigma)^*}{\Gamma \vdash M^\dagger : \mathtt{nat} \to \sigma}$$

We also define the domain of definition of a partial function as follows:

$$\mathtt{dom}\,(M) \equiv \mathtt{rec}\,\{\}\,(\lambda x\_z.\,\{x.1\} \cup z)\,M \qquad \frac{\Gamma \vdash M : (\mathtt{nat} \times \sigma)^*}{\Gamma \vdash \mathtt{dom}\,(M) : \{\mathtt{nat}\}}$$

We will also need some more operations on sets:

$$M \setminus N \equiv \{x \,|\, x \in M \wedge \forall y \in N, \neg\,(x = y)\} \qquad \frac{\Gamma \vdash M : \{\mathtt{nat}\} \qquad \Gamma \vdash N : \{\mathtt{nat}\}}{\Gamma \vdash M \setminus N : \{\mathtt{nat}\}}$$

$$M \subseteq N \equiv \forall x \in M, \neg\forall y \in N, \neg\,(x = y) \qquad \frac{\Gamma \vdash M : \{\mathtt{nat}\} \qquad \Gamma \vdash N : \{\mathtt{nat}\}}{\Gamma \vdash M \subseteq N : \mathtt{bool}}$$

Similarly to the $\Subset$ relation on initial segments, sufficiently precise partial functions are the ones satisfying:

$$S \propto Q \equiv \mathtt{dom}\left(Q\,S^\dagger\right) \subseteq \mathtt{dom}\,(S) \qquad \frac{\Gamma \vdash Q : (\mathtt{nat} \to \sigma) \to \{\mathtt{nat} \times \tau\} \qquad \Gamma \vdash S : (\mathtt{nat} \times \sigma)^*}{\Gamma \vdash S \propto Q : \mathtt{bool}}$$

Finally, we define an operator that picks an element in a non-empty set:

$$\mathtt{choose}\,(M) \equiv M\,?\,\{\mathtt{nil} \mapsto \square_\sigma \,|\, x :: \_ \mapsto x\} \qquad \frac{\Gamma \vdash M : \{\sigma\}}{\Gamma \vdash \mathtt{choose}\,(M) : \sigma}$$

The new demand-driven bar recursion operator is very similar to the previous one, except that if some $N \in Q\,S^\dagger$ is such that $N.1$ falls outside the domain of definition of S we do not discard N and extend the function in a linear order as before, but we use this $N.1$ as a hint and extend the current partial function at this point.

We extend our programming language with the following demand-driven bar recursion operator and its reduction rule:

$$\begin{aligned}
&\mathtt{dnddbr} : \left(\mathtt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}\right) \to \left((\mathtt{nat} \to \underline{A}) \to \left\{\mathtt{nat} \times \overline{A}\right\}\right) \\
&\qquad\qquad \to (\mathtt{nat} \times \underline{A})^* \to \left\{(\mathtt{nat} \times \underline{A})^*\right\} \\
&\mathtt{dnddbr}\,P\,Q\,S \to_\beta \{S\} \cup \mathtt{if}\,S \propto Q\,\mathtt{then}\,\{\} \\
&\qquad\qquad \mathtt{else}\,\mathtt{let}\,f = \lambda z x.\mathtt{dnddbr}\,P\,Q\,(S\,[z \mapsto x])\,\mathtt{in} \\
&\qquad\qquad\quad \mathtt{let}\,z = \mathtt{choose}\left(\mathtt{dom}\left(Q\,S^\dagger\right) \setminus \mathtt{dom}\,(S)\right)\,\mathtt{in} \\
&\qquad\qquad\quad \bigcup\left\{f\,z\,x \,\middle|\, x \in P\,z\left(\lambda y.\bigcup\left\{\widehat{Q\,t^\dagger}\,z \,\middle|\, t \in f\,z\,y \wedge t \propto Q\right\}\right)\right\}
\end{aligned}$$

Note that while in the previous section the expansions were always performed at $|S|$, here we expand the function at a point chosen in the domain of $Q\,S^\dagger$ but outside the domain of S. Since the final goal is to obtain a partial function S such that $S \propto Q$, choosing such a point is more natural and may avoid some useless recursive calls.

As before, the continuity of $\lambda f.\mathtt{dom}\,(Q\,f)$ ensures that $\mathtt{dnddbr}\,P\,Q\,S$ has a normal form for every arguments P, Q and S.

Using this new operator we define the demand-driven interpretation of the DNS principle as follows:

$$a \equiv \lambda pq.\left\{s^\dagger \,\middle|\, s \in \mathtt{dnddbr}\,p\,q\,\epsilon \wedge s \propto q\right\}$$

$$b \equiv \lambda pq. \left\{ \begin{array}{l} \texttt{let}\, z = \texttt{choose}\left(\texttt{dom}\left(q\, s^\dagger\right) \setminus \texttt{dom}\left(s\right)\right)\, \texttt{in} \\ \left(\lambda z. \left\langle z, \lambda x. \bigcup \left\{ \widehat{q\, t^\dagger}\, z \;\middle|\; \begin{array}{l} t \in \texttt{dnddbr}\, p\, q\, (s\,[z \mapsto x]) \\ \wedge\, t \propto q \end{array} \right\} \right\rangle \right) \end{array} \;\middle|\; \begin{array}{l} s \in \texttt{dnddbr}\, p\, q\, \epsilon \\ \wedge \neg\, s \propto q \end{array} \right\}$$

The following lemma is an adaptation of the one in the previous section:

▶ **Lemma 6.** *Let* $P : \texttt{nat} \to \left(\underline{A} \to \left\{\overline{A}\right\}\right) \to \{\underline{A}\}$ *and* $Q : (\texttt{nat} \to \underline{A}) \to \left\{\texttt{nat} \times \overline{A}\right\}$ *and let* $D \equiv \texttt{dnddbr}\, P\, Q$*. If for any* $M \in b\, P\, Q$ *we have* $P \bot_{\forall x \neg\neg A} M$*, then there exists a sequence of programs* $N_0, M_0, \ldots, N_{n-1}, M_{n-1}$ *such that if we write* $S_i \equiv \epsilon\,[N_0 \mapsto M_0] \ldots [N_{i-1} \mapsto M_{i-1}]$*:*

(a) $D\, \epsilon = D\, S_0 \supseteq D\, S_1 \supseteq \ldots \supseteq D\, S_n$

(b) *for all* $i \leq n$*,* $S \in D\, S_i$ *and* $N \in Q\, S^\dagger$
if $S \propto Q$ *and* $N.1 \in \texttt{dom}\left(S_i\right)$ *then* $S_i^\dagger\, N.1 \bot_{A[N.1/x]} N.2$

(c) *for all* $N \in Q\, S_n^\dagger$*,* $N.1 \in \texttt{dom}\left(S_n\right)$

Proof. $S_0 = \epsilon$ trivially satisfies (b). Suppose now $N_0, M_0, \ldots, N_{k-1}, M_{k-1}$ already defined, and satisfying (a) and (b). If for all $N \in Q\, S_k^\dagger$, $N.1 \in \texttt{dom}\left(S_k\right)$, then we choose $n = k$ and we are done since (c) is verified. Otherwise, let $N_k \equiv \texttt{choose}\left(\texttt{dom}\left(Q\, S_k^\dagger\right) \setminus \texttt{dom}\left(S_k\right)\right)$. We have:

$$D\, S_k =_\beta \{S_k\} \cup \bigcup \left\{ D\left(S_k\,[N_k \mapsto x]\right) \;\middle|\; x \in P\, k \left(\lambda x. \bigcup \left\{ \widehat{Q\, t^\dagger}\, k \;\middle|\; \begin{array}{l} t \in D\left(S_k\,[N_k \mapsto x]\right) \\ \wedge\, t \propto Q \end{array} \right\}\right) \right\}$$

and we have $S_k \in D\, S_k \subseteq D\, \epsilon$, so:

$$\left\langle N_k, \lambda x. \bigcup \left\{ \widehat{Q\, t^\dagger}\, N_k \;\middle|\; t \in D\left(S_k\,[N_k \mapsto x]\right) \wedge t \propto Q \right\} \right\rangle \in b\, P\, Q$$

and therefore by hypothesis:

$$P \bot_{\forall x \neg\neg A} \left\langle N_k, \lambda x. \bigcup \left\{ \widehat{Q\, t^\dagger}\, N_k \;\middle|\; t \in D\left(S_k\,[N_k \mapsto x]\right) \wedge t \propto Q \right\} \right\rangle$$

which means that there must be some:

$$M \in P\, N_k \left(\lambda x. \bigcup \left\{ \widehat{Q\, t^\dagger} N_k \;\middle|\; t \in D\left(S_k\,[N_k \mapsto x]\right) \wedge t \propto Q \right\}\right)$$

such that for all:

$$N \in \bigcup \left\{ \widehat{Q\, t^\dagger} N_k \;\middle|\; t \in D\left(S_k\,[N_k \mapsto M]\right) \wedge t \propto Q \right\}$$

we have $M \bot_{A[N_k/x]} N$. Define $M_k \equiv M$. We have (a) since $D\, S_{k+1} \equiv D\left(S_k\,[N_k \mapsto M_k]\right) \subseteq D\, S_k$. For (b), let $S \in D\, S_{k+1}$ and $N \in Q\, S^\dagger$ such that $N.1 \in \texttt{dom}\left(S_{k+1}\right) = \{N_k\} \cup \texttt{dom}\left(S_k\right)$. If $N.1 \in \texttt{dom}\left(S_k\right)$ then $S_{k+1}^\dagger\, N.1 =_\beta S_k^\dagger\, N.1$ and we are done by hypothesis on S_k. Otherwise $N.1 =_\beta N_k$ and $S_{k+1}^\dagger\, N.1 =_\beta M_k$. But then we can prove that:

$$N.2 \in \bigcup \left\{ \widehat{Q\, t^\dagger} N_k \;\middle|\; t \in D\left(S_k\,[N_k \mapsto M_k]\right) \wedge t \propto Q \right\}$$

so we obtain $M_k \bot_{A[N_k/x]} N.2$ by property of M_k. ◀

Correctness of this new interpretation of the DNS principle then follows from this lemma as in the previous section.

Having a demand-driven operator allows for a potentially simpler interpretation, as well as a more natural one. Indeed, there is no reason why the bar recursion operator should rely on a particular ordering of natural numbers (as the non-demand-driven one does) and the

intuitive interpretation should be that the operator only computes the witnesses that are necessary, rather than collecting blindly witnesses until there are enough of them. This also opens the possibility of interpreting theories other than arithmetic, where basic objects are not natural numbers and may not have any natural ordering.

It should be noted that while $\texttt{choose}\,(M)$ picks the first element of (the list encoding) M, this property is never used in the proof. The only required property is that if M has at least one element, then $\texttt{choose}\,(M)$ picks an element of M (hence its name). Picking any other element would work as well, and the choice we made seems arbitrary. In the context of the standard Dialectica interpretation, Oliva and Powell did not encounter this because in that case $Q\,S^\dagger$ returns a single pair, and there is only one possible point at which S can be extended. In the Diller-Nahm variant, however, there may be several points in $\texttt{dom}\left(Q\,S^\dagger\right) \setminus \texttt{dom}\,(S)$ and there seems to be no canonical choice. Depending on the case it may be interesting to use some heuristic to choose the point at which S is extended. One may also wonder whether S could be extended at every point in $\texttt{dom}\left(Q\,S^\dagger\right) \setminus \texttt{dom}\,(S)$ simultaneously, but this seems impossible since the correctness proof requires that the sequence of partial functions $S_0, S_1, \ldots$ is totally ordered.

References

1. Stefano Berardi, Marc Bezem, and Thierry Coquand. On the Computational Content of the Axiom of Choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.
2. Justus Diller and Werner Nahm. Eine Variante zur Dialectica-Interpretation der Heyting-Arithmetik endlicher Typen. *Archiv für mathematische Logik und Grundlagenforschung*, 16:49–66, 1974.
3. Martín Escardó and Paulo Oliva. The herbrand functional interpretation of the double negation shift. *Journal of Symbolic Logic*, 82(2):590–607, 2017.
4. Harvey Friedman. Classically and intuitionistically provably recursive functions. In Gert Müller and Dana Scott, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer, 1978.
5. Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958.
6. Stephen Cole Kleene. On the Interpretation of Intuitionistic Number Theory. *Journal of Symbolic Logic*, 10(4):109–124, 1945.
7. Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957*, Studies in Logic and the Foundations of Mathematics, pages 101–128. North-Holland Publishing Company, 1959.
8. Paulo Oliva and Thomas Powell. Bar recursion over finite partial functions. *Annals of Pure and Applied Logic*, 168(5):887–921, 2017.
9. Pierre-Marie Pédrot. A functional functional interpretation. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 77:1–77:10. ACM, 2014.
10. Pierre-Marie Pédrot. *A Materialist Dialectica. (Une Dialectica matérialiste)*. PhD thesis, Paris Diderot University, France, 2015.
11. Clifford Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, 1962.