# Light Functional Interpretation

## An Optimization of Gödel's Technique Towards the Extraction of (More) Efficient Programs from (Classical) Proofs

Mircea-Dan Hernest*

Laboratoire d'Informatique (LIX)
École Polytechnique, F-91128 Palaiseau, France
danher@lix.polytechnique.fr

**Abstract.** We give a Natural Deduction formulation of an adaptation of Gödel's functional (*Dialectica*) interpretation to the extraction of (more) efficient programs from (classical) proofs. We adapt Jørgensen's formulation of pure Dialectica translation by eliminating his "Contraction Lemma" and allowing free variables in the extracted terms (which is more suitable in a Natural Deduction setting). We also adapt Berger's *uniform* existential and universal quantifiers to the Dialectica-extraction context. The use of such quantifiers *without computational meaning* permits the identification and isolation of contraction formulas which would otherwise be redundantly included in the pure-Dialectica extracted terms. In the end we sketch the possible combination of our refinement of Gödel's Dialectica interpretation with its adaptation to the extraction of bounds due to Kohlenbach into a *light monotone* functional interpretation.

**Keywords:** Program extraction from (classical) proofs, Complexity of extracted programs, Berger's uniform quantifiers, Gödel's Functional interpretation, Proof-Carrying Code, Proof Mining.

## 1 Introduction

Important practical results have been obtained in recent years in the field of extractive Proof Theory (also dubbed *proof mining* [22]). The implemented algorithms coming from metamathematical research have yielded interesting and in many cases quite unexpected programs [7, 15, 31, 32]. Various approaches to program extraction from *classical*[1] proofs have been developed over years of research [1, 3, 6, 8, 9, 21, 24, 25, 27–30, 32]. The use of proof interpretations instead of the direct application of cut-elimination has opened the path to the obtention of better practical results by means of such modular techniques (see

* *Project LogiCal* – Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École Polytechnique, INRIA et Université Paris-Sud – FRANCE and *Graduiertenkolleg Logik in der Informatik* (GKLI) – München, GERMANY. Partly financed by Deutsche Forschungsgemeinschaft.

[1] *Constructive* proofs have a more or less explicit computational content, see, e.g., [4].

L. Ong (Ed.): CSL 2005, LNCS 3634, pp. 477–492, 2005.

[15] for a discussion on this). Gödel's functional (*Dialectica*) interpretation [2, 13] directly applies to (far) more complex proofs than (its weaker form) Kreisel's Modified Realizability [23]. Refinements [1, 6, 9] of the latter's combination with Friedman's A-translation [12] only partly repair this disparity (see [15] or [20] for discussions on this issue). However, the (rough) exact realizers yielded by the Dialectica interpretation are generally (far) more complex than those produced by, e.g., the technique of Berger, Buchholz and Schwichtenberg [6]. The main reason for such a situation seems to be the inclusion of *contraction formulas* in the rough Dialectica realizing terms. Even though in some cases the primary realizers extracted via the two techniques normalize to basically the same programs (see [15] for such an example), the normalization of programs extracted by proof interpretations is generally far more expensive than their synthesis (see [16] for a detailed complexity exposition). The simpler the rough extracted term is, the lower the overall cost of producing the final normalized realizer becomes. In order to handle this contraction problem of the Dialectica interpretation, Kohlenbach [21] devised the *monotone* functional interpretation, an adaptation of Gödel's technique to the extraction of uniform bounds for the exact realizers (see also [11] for the more recent *bounded* functional interpretation of Ferreira and Oliva). While so prolific (see [20] or [22]) in the context of mathematical Analysis, the monotone functional interpretation, used alone, is generally not as practically useful for the synthesis of exact realizers in discrete mathematics.

We propose in this paper a different kind of optimization of Gödel's functional interpretation by the elimination already from the primary extracted terms of a number of contraction formulas which are identified as *computationally redundant* by means of an adaptation of Berger's *uniform* quantifiers from [5] to the Dialectica-extraction context. These are called "quantifiers without computational content" in [34] and we will here call them quantifiers *without computational meaning* (abbreviated `ncm`, with `n` from *non*). We will here denote by $\overline{\forall}$ the `ncm` universal quantifier and by $\overline{\exists}$ the `ncm` existential quantifier. While our $\overline{\exists}$ is identical[2] to Berger's $\{\exists\}$, our $\overline{\forall}$ requires a further strengthening of the restriction set by Berger on his $\{\forall\}^+$ rule – we must take into account the inclusion of the *computationally relevant* contractions into the Dialectica realizing terms.

We build on top of Jørgensen's [19] Natural Deduction formulation of pure Dialectica interpretation which we transform by eliminating his "Contraction Lemma"[3] and by allowing free variables in the extracted terms[4]. We call *light*[5] functional interpretation this refinement of Gödel's Dialectica technique for the extraction of more efficient programs from (classical) proofs. We generally abbreviate (both regular and `ncm`) *quantifier free* by `qfr`. We will use expressions like "`qfr` formula", "`ncm` variable" or "`ncm` quantifiers" with the obvious meanings.

[2] Modulo the formulation as axioms like in [34] of Berger's rules for $\{\exists\}$ from [5].
[3] Which we find too complicated for a computer-implemented Dialectica extraction.
[4] Which is more suitable in a Natural Deduction context (see [16] for a discussion).
[5] Where "light" is to be understood as the opposite of "heavy" and not otherwise.

## 2 An Arithmetic for Gödel Functionals

We devise a weakly extensional variant WE–Z of the intuitionistic arithmetical system Z of Berger, Buchholz and Schwichtenberg [6] which restricts extensionality and adds the elements peculiar to Gödel's *Dialectica* interpretation [2, 13]. It moreover integrates the *non-computational-meaning* (abbreviated ncm) quantifiers mentioned in Section 1 above . System (WE–)Z is an extension of Gödel's **T** with the logical and arithmetical apparatus which renders it suitable to the applied program extraction from (classical) proofs (by means of Dialectica interpretation and its variants), see [6, 34, 36].

*Finite types* are inductively generated from base types by the rule that if $\sigma$ and $\tau$ are types then $(\sigma\tau)$ is a type. For simplicity we take as *base* types only the type $\iota$ for natural numbers and $o$ for booleans. We make the convention that concatenation is right associative and consequently omit unnecessary parenthesis, writing $\delta\,\sigma\,\tau$ instead of $(\delta(\sigma\tau))$. We denote tuples of types by $\underline{\sigma} :\equiv \sigma_1, \ldots, \sigma_n$. We abbreviate by $\underline{\sigma}\tau$ the type $\sigma_1 \ldots \sigma_n\tau$. It is immediate that every type $\tau$ can be written as either $\tau \equiv \underline{\sigma}\iota$ or $\tau \equiv \underline{\sigma}o$ .

The *term system* is a variant of Gödel's **T** formulated over the finite types with $\lambda$-abstraction as primitive. This is most appropriate in a Natural Deduction context. Terms are hence built from variables and term constants by $\lambda$-abstraction and application. We represent the latter as concatenation and we agree that it is left-associative in order to avoid excessive parenthesizing. All variables and constants have an *a priori* fixed type and terms have a type fixed by their formation. Written term expressions are always assumed to be well-formed in the sense that types match in all applications between sub-terms. As particular (term) *constants* we distinguish the following:

- $\mathtt{tt}^o$ and $\mathtt{ff}^o$ which denote boolean truth and falsity;
- for each type $\tau$ the *selector* $\mathtt{If}_\tau$ of type $o\,\tau\,\tau\,\tau$ which denotes choice according to a boolean condition with the usual *if-then-else* semantics;
- $0^\iota$ (zero), $\mathtt{S}^\iota$ (successor) and Gödel's recursor $\mathtt{R}_\tau$ of type $\tau\,(\iota\,\tau\,\tau)\,\iota\,\tau$;
- equality $=^{\iota\,\iota\,o}$ – a functional constant and not predicate in our system.

*Variables* are denoted by $a, b, c, p, q, u, v, x, y, z, U, V, X, Y, Z$ such that, if not otherwise specified, $a, b, c$ are free and $u, v, x, y, z$ are bound variables of type $\iota$. Also $p, q$ denote variables of type $o$ (be them free or bound) and $U, V, X, Y, Z$ are *functional* variables (i.e., not of base type). We denote terms by $r, s, t, S, T$. We use sub- or super- scripts to enlarge the classes of symbols. We use underlined letters to denote tuples of corresponding objects. Tuples are just comma-separated lists of objects. If $\underline{t} \equiv t_1, \ldots, t_n$ we denote by $s(\underline{t})$ or even $s\underline{t}$ the term $st_1 \ldots t_n$, i.e., $((st_1)\ldots)t_n$ by the left-associativity convention. Also $\underline{s}(\underline{t})$ and $\underline{s}\,\underline{t}$ denote the tuple $s_1(\underline{t}), \ldots, s_m(\underline{t})$. As particular *terms* we distinguish the following:

- Boolean conjunction and implication (with their usual semantics):

$$\begin{aligned} \mathtt{And}^{ooo} &:\equiv \lambda p, q.\, \mathtt{If}_o\ p\ q\ \mathtt{ff} \\ \mathtt{Imp}^{ooo} &:\equiv \lambda p, q.\, \mathtt{If}_o\ p\ q\ \mathtt{tt} \end{aligned}$$

- For each higher-order type $\tau \equiv \underline{\sigma}\iota$ or $\tau \equiv \underline{\sigma}o$ we define the *zero* term $\mathtt{0}_\tau$ of type $\tau$. We also let $\mathtt{0}_\iota :\equiv 0$ and $\mathtt{0}_o :\equiv \mathtt{ff}$ and thus every type is inhabited by a zero term (since we consider only $\iota$ and $o$ as base types).
- For each positive integer $n$ and type $\tau$ we define the *n-selector* $\mathtt{If}^n_\tau$ of type $\overbrace{o \dots o}^{n}\,\overbrace{\tau \dots \tau}^{n}\,\tau\,\tau$ by $\mathtt{If}^1_\tau :\equiv \mathtt{If}_\tau$ and for $n \geq 2$ the definition of $\mathtt{If}^n_\tau$ is $\lambda p_1, \dots, p_n, x_{n+1}, x_n, \dots, x_1.\, \mathtt{If}_\tau\, p_1\, (\mathtt{If}^{n-1}_\tau\, p_2 \dots p_n\, x_{n+1}\, x_n \dots x_2)\, x_1$ s.t. $\mathtt{If}^n_\tau(r_1, \dots, r_n, t_{n+1}, t_n, \dots, t_1)$ selects the first $t_i$ with $i \in \overline{1,n}$ for which $r_i$ is false, if it exists, otherwise $t_{n+1}$ – if all $\{r_i\}_{i=1}^n$ are true.

The base logical system is a Natural Deduction formulation (see [6] and [34]) of Intuitionistic Logic[6]. We use $\wedge$ (logical conjunction), $\rightarrow$ (logical implication), $\forall$ (forall) and $\exists$ (*strong*, intuitionistic exists) as base logical constants. The only predicate symbol is the unary $\mathtt{at}$ which takes a single boolean argument. If $t^o$ is a boolean term then $\mathtt{at}(t)$ is the *atomic* formula which (informally) denotes the fact that $t$ is true. We *define* the logical falsum in terms of boolean falsity by $\bot :\equiv \mathtt{at}(\mathtt{ff})$. The *weak* (classical) existential quantifier $\exists^{\mathtt{nl}}$ is defined in terms of $\forall$ by $\exists^{\mathtt{nl}} x\, A(x) :\equiv (\forall x.\, A(x) \rightarrow \bot) \rightarrow \bot$. Negation $\neg$ and equivalence $\leftrightarrow$ are defined as usual, i.e., $\neg A :\equiv A \rightarrow \bot$ and $A \leftrightarrow B :\equiv ((A \rightarrow B) \wedge (B \rightarrow A))$. Disjunction is defined by $A \vee B :\equiv \exists p^o\, ((\mathtt{at}(p) \rightarrow A) \wedge ((\neg\mathtt{at}(p)) \rightarrow B))$.

*Predicate equality* at base types is defined for boolean terms $s$ and $t$ by $s =_o t :\equiv \mathtt{at}(s) \leftrightarrow \mathtt{at}(t)$ and for natural terms $s$ and $t$ by $s =_\iota t :\equiv \mathtt{at}(= s\, t)$. Equality between terms $s$ and $t$ of type $\tau \equiv \sigma_1 \dots \sigma_n \sigma$, with $\sigma \in \{o, \iota\}$, is extensionally defined as $s =_\tau t :\equiv \forall x_1^{\sigma_1}, \dots, x_n^{\sigma_n}\, (s\, x_1 \dots x_n =_\sigma t\, x_1 \dots x_n)$. For any type $\tau$ we denote by $s \neq_\tau t :\equiv \neg(s =_\tau t)$ *non-equality* between the terms $s^\tau$ and $t^\tau$. If non-ambiguous, we often omit to specify the type $\tau$ of (non-)equality.

We also introduce in our system an adaptation of Berger's [5] *uniform* quantifiers, here denoted $\overline{\forall}$ (forall $\mathtt{ncm}$) and $\overline{\exists}$ (exists $\mathtt{ncm}$) to the extraction of (more) efficient programs by Gödel's Dialectica interpretation. From a logic viewpoint $\overline{\forall}$ and $\overline{\exists}$ behave exactly like $\forall$ and $\exists$ – a theorem stating that "the (purely syntactic) replacement of $\overline{\forall}$ and $\overline{\exists}$ with their *computationally meaningful*[7] (or *regular*) correspondents in a proof $\mathcal{P}$ yields a(nother) proof in the corresponding system without $\mathtt{ncm}$ quantifiers" can easily be established. However, the converse to this (meta)theorem does not hold (in general) because of the (necessary) restriction which is set on the introduction rules for the $\mathtt{ncm}$-universal quantifier and implication (see Sections 2.1 and 2.2 below). The special rôle of $\overline{\forall}$ and $\overline{\exists}$ is played in the program-extraction process only. There they act like some kind of labels for parts of the proof at input which are to be ignored since they are *a priory* (i.e., at the proof-building stage) distinguished as having no computational content. They also bring an important optimization with respect to the maximal type degree of programs extracted from those proofs for which the use of the com-

[6] The prominent place Johansson's Minimal Logic [18] has in the presentation of system $\mathtt{Z}$ in [6] or [34] is no longer needed for our (light) Dialectica-extraction exposition.

[7] Notice that $\forall$ is as computationally meaningful as $\exists$ in the context of program extraction by (light) Dialectica interpretation, see Definition 31 and Theorem 34.

putationally meaningful correspondents would have just brought an unjustified increase of this maximal type degree[8].

In order to avoid excessive parenthesizing we make the usual conventions that $\forall, \overline{\forall}, \exists^{\mathbf{1}}, \exists, \overline{\exists}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$ is the decreasing order of precedence and $\rightarrow$ is right associative. For (more efficient) program-extraction purposes we impose that all axioms are closed formulas and for optimization purposes – at the example of Schwichtenberg's MINLOG system [34, 36] – their closure is ensured with $\overline{\forall}$ rather than $\forall$. The only exception[9] to this rule is the *Induction Axiom* `IA`, see Section 2.2 for the various definitions of induction within system `WE–Z`. Therefore it will be understood that even though a formula presented below as axiom is literally open, in fact the axiom it denotes is the $\overline{\forall}$ closure of the respective formula.

### 2.1 The Logical Axioms and Rules of System `WE–Z`

We begin by adapting the set of rules for Minimal Logic from [34] to the setting of program-extraction by the light Dialectica interpretation (defined in Section 3). First of all we define the following two *variable conditions* which will be used to constrain the rules concerning the (`ncm-`)universal quantifier:

- $\mathtt{VC}_1(z)$: the variable $z$ does not occur free in any of the undischarged assumptions of the proof of the premise of the rule;
- $\mathtt{VC}_2(z,t)$: the term $t$ is "free for" $z$ in the conclusion, i.e., no free variable of $t$ gets quantified after substituting $\{z \leftarrow t\}$ in the conclusion.

We also define an *`ncm`-formula condition* which is required to constrain the rule of implication introduction *with contraction* (see below) in order to attain the soundness theorem for the light Dialectica interpretation. For a formula $A$, the condition `ncm-FC`$(A)$ says that if $A$ contains (at least) a positive universal or a negative existential (regular) quantifier[10], then $A$ *must not* contain any `ncm` quantifier[11]. The logical rules of our system `WE–Z` are then as follows:

- Deduction from an (arbitrary, undischarged) assumption: $A \vdash A$ .
- Conjunction elimination left: $\dfrac{A \wedge B}{A}\ \wedge_l^-$ , conjunction elimination right: $\dfrac{A \wedge B}{B}\ \wedge_r^-$ and conjunction introduction: $\dfrac{A\,,\,B}{A \wedge B}\ \wedge^+$ .
- Implication elimination: $\dfrac{A\,,\,A \rightarrow B}{B}\ \rightarrow^-$ (Modus Ponens).

[8] Upon which the run-time complexity of the normalization algorithm directly depends, regardless of the reduction strategy, see Berger's paper [5] for more on this.

[9] This *notable* exception is necessary only in the context of Dialectica extraction because the Dialectica realizers of `IA` integrate the induction formula via an $\rightarrow^+$ *with contraction*, see Section 2 of [14], Section 2.2 and the proof of Theorem 34 below.

[10] Which means that $A$ is *computationally relevant*, see "Implication introduction".

[11] So that the light `D`-interpretation of $A$ has a `qfr` *base* formula $A_{\mathtt{D}}$, see Definition 31.

– Implication introduction: $\dfrac{[A]\ \ldots\ /B}{A \to B}\ \to^{+}$ , where some particular (possibly empty) class of instances of the formula $A$ among the undischarged assumptions of the proof of $B$ gets discharged. If at least two instances of $A$ get discharged (i.e., the $\to^{+}$ is with contraction) then the $\texttt{ncm-FC}(A)$ restriction applies. If moreover the premise of $\texttt{ncm-FC}(A)$ holds, then $A$ is named *computationally relevant*, otherwise $A$ is a *computationally irrelevant* (*redundant* for Dialectica) contraction formula.

– [ncm-]ForAll elimination: $[\dfrac{\overline{\forall} z\, A(z)}{A(t)}\,\overline{\forall}^{-}_{z,t}\ ]\ \dfrac{\forall z\, A(z)}{A(t)}\,\forall^{-}_{z,t}$ , s.t. $\texttt{VC}_2(z,t)$.

– ForAll introduction: $\dfrac{A(z)}{\forall z\, A(z)}\,\forall^{+}_{z}$ , such that $\texttt{VC}_1(z)$.

– ncm-ForAll introduction: $\dfrac{\mathcal{P}\colon A(z)}{\overline{\forall} z\, A(z)}\,\overline{\forall}^{+}_{z}$, such that $\texttt{VC}_1(z)$ and $\texttt{VC}_3(z,\mathcal{P})$. The latter (third) *variable condition* applies to the $\overline{\forall}$-quantified variables only. Although basically the same as the pre-condition set by Berger on his $\{\forall\}^{+}$ rule in [5], a strengthening peculiar to light-Dialectica extraction is necessary:

  – $\texttt{VC}_3(z,\mathcal{P})$: the variable $z$ does not occur free in any of the *instantiating terms* $t$ involved by a $\forall^{-}_{\bullet,t}$ in the proof $\mathcal{P}$ (so far Berger's restriction) **and** $z$ is also not free in the computationally relevant *contraction formulas* of $\mathcal{P}$ (defined above at the Implication Introduction item).

Intuitionistic Logic is then obtained by adding the axioms defining $\exists$ and $\overline{\exists}$:

$\texttt{Ax}\exists^{-}$ : $\quad \exists z_1\, A(z_1) \wedge \forall z_2\, [\, A(z_2) \to B\,] \to B$ ($\exists$ elimination)

$\texttt{Ax}\exists^{+}$ : $\quad \forall z_1\, [\, A(z_1) \to \exists z_2\, A(z_2)\,]$ ($\exists$ introduction)

$\texttt{Ax}\overline{\exists}^{-}$ : $\quad \overline{\exists} z_1\, A(z_1) \wedge \overline{\forall} z_2\, [\, A(z_2) \to B\,] \to B$ ($\overline{\exists}$ elimination)

$\texttt{Ax}\overline{\exists}^{+}$ : $\quad \overline{\forall} z_1\, [\, A(z_1) \to \overline{\exists} z_2\, A(z_2)\,]$ ($\overline{\exists}$ introduction)

with the usual restriction that $z_2$ is not free in $B$ and, most important,

$\texttt{AxEFQ}$: $\quad \bot \to A$ (Ex-Falso-Quodlibet)

Notice that Intuitionistic Logic could have equally been formulated with rules for $\exists$ and $\vee$[12] (instead of axioms), see [5, 38]. We here follow [34] in choosing a formulation which is more suitable for computer-applied program-extraction.

### 2.2 Weakly Extensional Intuitionistic Arithmetic WE–Z

We now add the basic arithmetical apparatus to Intuitionistic Logic. We first introduce the axioms which give the (usual) behavior of (higher-order exten-

[12] The Boolean Induction axiom AxBIA from Section 2.2 is strictly required for attaining the usual logical behavior of $\vee$ in our formulation.

sional) equality, stressing from the beginning that *extensionality* must[13] be restricted in the context of Dialectica interpretation. The extensionality *axiom* $E_{\sigma,\tau} : \forall z^{\sigma\tau}, x^{\sigma}, y^{\sigma}.\ x =_{\sigma} y \to zx =_{\tau} zy$ must not be derivable in our system. We here deviate from system Z of [6, 34, 36] which derives $E_{\sigma,\tau}$ and therefore is fully extensional. We first present the more basic axioms of Reflexivity, Symmetry and Transitivity which we retain (modulo our definition of higher-order equality, hence they are no longer quantifier-free here) from system Z of [6, 34, 36] :

| | | |
|---|---|---|
| $\mathtt{AxREF}_{\tau}$ : | $x =_{\tau} x$ | (Reflexivity) |
| $\mathtt{AxSYM}_{\tau}$ : | $x =_{\tau} y \to y =_{\tau} x$ | (Symmetry) |
| $\mathtt{AxTRZ}_{\tau}$ : | $x =_{\tau} y \wedge y =_{\tau} z \to x =_{\tau} z$ | (Transitivity) |

Notice that although $\mathtt{AxSYM}_{\tau}$ and $\mathtt{AxTRZ}_{\tau}$ have no computational content under Modified Realizability, they must be provided with realizing terms under (light) Dialectica interpretation for higher-order $\tau$[14]. We thus stay as close as possible to the axiomatic of system Z of [6, 34, 36], rendering easier the task of implementing program-extraction by (light) Dialectica interpretation in Minlog [34, 36]. We do, however, have to deviate from system Z when it comes to the *Compatibility Axiom* (which implies $E_{\sigma,\tau}$): $x =_{\sigma} y \to B(x) \to B(y)$, which we replace by the following (strictly) weaker Compatibility Rule:

$$\mathtt{COMPAT}_{\sigma}: \quad \frac{\begin{array}{c} A_0 \\ \vdots \\ x =_{\sigma} y \end{array}}{B(x) \to B(y)}$$

with the restriction that all undischarged assumptions used in the proof of $x =_{\sigma} y$ (here denoted $A_0$) are quantifier-free

Had the above restriction[15] not been present, the Compatibility Axiom would be directly deducible by $\to^{+}$, hence full extensionality would be derivable and (light) Dialectica interpretation would fail to interpret all proofs of our system[16].

[13] See, e.g., the chapter on Dialectica interpretation in [20] for detailed explanations. Howard's original counterexample to the Dialectica realizability of the extensionality axiom $E_{\iota\iota,\iota}$ by Gödel primitive recursive functionals is exposed in [17]. See also [35] for a counterexample to the Dialectica realizability of $E_{\iota\iota,\iota}$ by Van de Pol – Schwichtenberg monotone majorizable functionals (a class of functionals intersecting but independent of Gödel's **T**) .

[14] We could have used only $\mathtt{AxREF}_{\iota}$, $\mathtt{AxSYM}_{\iota}$ and $\mathtt{AxTRZ}_{\iota}$ as axioms since higher-type Reflexivity, Symmetry and Transitivity can be deduced in (pure) Minimal Logic from the Reflexivity, Symmetry and respectively Transitivity of natural numbers. The latter are quantifier-free and hence realizer-free under both Realizability and Dialectica interpretations. We however chose the above presentation for practical reasons – proofs are shorter and the realizers for the (light) Dialectica interpretation of higher-order Symmetry and Transitivity are immediate (see Section 3 of [14]).

[15] This restriction is not only sufficient but also necessary. Already by allowing purely universal undischarged assumptions in the proof of $x =_{\sigma} y$ we can deduce $E_{\iota\iota,\iota}$ in our system and we therefore become subject to Howard's counterexample [17].

[16] Details of the fact that Dialectica is valid for COMPAT are given in Section 3 of [14].

We now present the *boolean axioms*. While the desired behavior of $\mathtt{ff}$ is already given by $\mathtt{AxEFQ}$, for $\mathtt{tt}$ we must introduce the Truth Axiom

$\mathtt{AxTRH}: \quad \mathtt{at}(\mathtt{tt}) \quad .$

In order to attain the usual logical behavior of $\vee$ we must complement its definition with the following[17]

$\mathtt{AxBIA}: \quad A(\mathtt{tt}) \wedge A(\mathtt{ff}) \ \to \ \forall p^o A(p) \qquad$ (Boolean Induction Axiom)

and for the selectors $\mathtt{If}_\tau$ we introduce the following expected axioms:

$$\mathtt{AxIf}_\tau: \quad \begin{cases} \mathtt{If}_\tau \ \mathtt{tt} \ x^\tau \ y^\tau =_\tau x \\ \mathtt{If}_\tau \ \mathtt{ff} \ x^\tau \ y^\tau =_\tau y \end{cases} \quad .$$

Defining axioms for the constants specific to *natural numbers* are as usual

$$\mathtt{AxS}: \quad \begin{cases} \mathtt{S}z \neq_\iota 0 \\ \mathtt{S}x =_\iota \mathtt{S}y \ \to \ x =_\iota y \end{cases} \qquad \mathtt{AxR}_\tau: \quad \begin{cases} \mathtt{R}_\tau \, x \, y \, 0 =_\tau x \\ \mathtt{R}_\tau \, x \, y \, (\mathtt{S}z) =_\tau y(z, \mathtt{R}_\tau \, x \, y \, z) \end{cases}$$

We finally arrive at integrating Induction for natural numbers in $\mathtt{WE\text{–}Z}$. There are very simple realizing terms under Kreisel's Modified Realizability for the usual Induction Axiom $\mathtt{IA}: \ A(0) \ \to \ \forall z \, (A(z) \to A(\mathtt{S}z)) \ \to \ \forall z \, A(z)$ – see [6, 34]. In contrast, the Dialectica realizing terms for $\mathtt{IA}$ are far more complicated because they include the Dialectica-translation of $\forall z(A(z) \to A(\mathtt{S}z))$ – see Section 2 of [14], Footnote 9 and the proof of Theorem 34 below. We will therefore use the following variant of the simpler induction rule employed by Jørgensen in [19]:

$\mathtt{IR}_0: \qquad \emptyset \ \cdots \ / A(0) \ , \ \emptyset \ \cdots \ / \forall z(A(z) \to A(\mathtt{S}z)) \ \vdash \ \forall z A(z)$

If one ignores the $\mathtt{ncm}$ quantifiers (and hence also the $\mathtt{ncm\text{-}FC}$ restriction), this system of intuitionistic arithmetic is in fact *the* weakly extensional version of system $\mathtt{Z}$ of [6, 34, 36], extended with $\bot \leftrightarrow \mathtt{at}(\mathtt{ff})$. We denote the system with the $\mathtt{ncm}$ quantifiers by $\mathtt{WE\text{–}Z}$ and the system without the $\mathtt{ncm}$ quantifiers (hence without the $\mathtt{ncm\text{-}FC}$ restriction) by $\mathtt{WE\text{–}Z}^-$. In $\mathtt{WE\text{–}Z}^-$ there is a full equivalence between $\mathtt{IA}$, $\mathtt{IR}_0$ and the usual Induction Rule[18] $\mathtt{IR}: \ A(0) \, , \, \forall z(A(z) \to A(\mathtt{S}z)) \vdash \forall z A(z)$.

We denote deduction in systems $\mathtt{WE\text{–}Z}$ and $\mathtt{WE\text{–}Z}^-$ by $\vdash$ and respectively $\vdash_-$.

**Remark 21** The following lemmas hold in $\mathtt{WE\text{–}Z}^-$:

$\mathtt{LmAND}: \quad \vdash_- \forall p^o, q^o \ ( \ \mathtt{at}(\mathtt{And} \, p \, q) \ \leftrightarrow \ \mathtt{at}(p) \wedge \mathtt{at}(q) \ )$

$\mathtt{LmIMP}: \quad \vdash_- \forall p^o, q^o \ ( \ \mathtt{at}(\mathtt{Imp} \, p \, q) \ \leftrightarrow \ \mathtt{at}(p) \to \mathtt{at}(q) \ )$

They establish the equivalence for terms of boolean and logical conjunction and implication, fact which permits the treatment of $\mathtt{qfr}$ formulas as prime (atomic) formulas, see Remark 22 and Section 1 of [14]. The $\mathtt{WE\text{–}Z}^-$ lemmas $\mathtt{LmO}\iota: \ \vdash_- \mathtt{O}^{\sigma\iota} \underline{z}^\sigma =_\iota 0$ and $\mathtt{LmOo}: \ \vdash_- \mathtt{O}^{\sigma o} \underline{z}^\sigma =_o \mathtt{ff}$ describe the behavior of the

[17] Only the "disjunction introduction" theorems $\vdash A \to A \vee B$ and $\vdash B \to A \vee B$ are ensured by the definition of $\vee$. The "elimination of disjunction" requires $\mathtt{AxBIA}$, see Remark 22 below.

[18] See Section 2 of [14] for details. The simulation of $\mathtt{IR}$ in terms of $\mathtt{IR}_0$ fails in $\mathtt{WE\text{–}Z}$.

zero terms. The expected behavior of the selector $\mathtt{If}^n_\tau$ is given by the following $n+1$ lemmas of $\mathtt{WE\text{–}Z}^-$ grouped under the name $\mathtt{LmIf}^n_\tau$ :

$$\begin{cases} \{\ \vdash_- \wedge_{j=1}^{i-1}\mathtt{at}(p_j) \wedge \neg\mathtt{at}(p_i) \ \to\ \mathtt{If}^n_\tau(p_1,\ldots,p_n,x_{n+1},x_n,\ldots,x_1) =_\tau x_i\ \}_{i=1}^n \\ \quad \vdash_- \wedge_{i=1}^n \mathtt{at}(p_i) \ \to\ \mathtt{If}^n_\tau(p_1,\ldots,p_n,x_{n+1},x_n,\ldots,x_1) =_\tau x_{n+1} \end{cases}$$

**Remark 22** There exists a unique bijective association of boolean terms to $\mathtt{qfr}$ formulas $A_0 \mapsto \mathtt{t}_{A_0}$ such that $\vdash_- A_0 \leftrightarrow \mathtt{at}(\mathtt{t}_{A_0})$ for all $A_0$. In particular $\vdash_- (p =_o \mathtt{tt}) \leftrightarrow \mathtt{at}(p)$ and $\vdash_- (p =_o \mathtt{ff}) \leftrightarrow \neg\mathtt{at}(p)$. It then follows that all $\mathtt{qfr}$ formulas $A_0$ of system $\mathtt{WE\text{–}Z}^-$ are decidable in the sense that $\vdash_- A_0 \vee \neg A_0$. Using $\mathtt{AxBIA}$ it further follows that we can produce $\mathtt{WE\text{–}Z}^-$ proofs by *case distinction* over $\mathtt{qfr}$ formulas, i.e., $\vdash_- (A_0 \to A) \wedge (\neg A_0 \to A) \ \to\ A$ . More general, the following schema of *disjunction elimination*[19] is deducible in system $\mathtt{WE\text{–}Z}^-$ :

$$\vdash_- \ \wedge_{i=1}^n (A_i \to B) \ \to\ (\vee_{i=1}^n A_i \ \to\ B) \tag{1}$$

*Stability* for $\mathtt{qfr}$ formulas of $\mathtt{WE\text{–}Z}$, i.e., $\vdash_- \neg\neg A_0 \to A_0$ follows as an immediate application of case distinction over $\mathtt{qfr}$ formulas with $A :\equiv \neg\neg A_0 \to A_0$ [20].

### 2.3 The Semi-classical (Plus Choice) System $\mathtt{WE\text{–}Z}^+$

We present below the extension of $\mathtt{WE\text{–}Z}$ with three axioms which have simple realizers *directly* under the (light) Dialectica interpretation (defined in Section 3). The first two principles are (logically) deducible in the (fully) classical version of $\mathtt{WE\text{–}Z}$ (obtained by adding *full* stability $\neg\neg A \to A$) but not in $\mathtt{WE\text{–}Z}$. These *semi-classical* (logical) axioms are Markov's Principle[21]

$\mathtt{AxMK}$ : $\quad \exists^{\mathrm{fl}} z\, A_0(z) \to \exists z\, A_0(z)$

and Independence of premises for universal premises

$\mathtt{AxIP}_\forall$ : $\quad [\forall x\, A_0(x) \ \to\ \exists y\, B(y)] \ \to\ \exists y\, [\forall x\, A_0(x) \ \to\ B(y)]$ .

System $\mathtt{WE\text{–}Z}^+$ is obtained by further adding to $\mathtt{WE\text{–}Z}$ (besides $\mathtt{AxMK}$ and $\mathtt{AxIP}_\forall$) the non-logical (i.e., not logically deducible in $\mathtt{WE\text{–}Z}$) Axiom of Choice:

$\mathtt{AxAC}$ : $\quad \forall x\, \exists y\, B(x,y) \ \to\ \exists Y\, \forall x\, B(x, Y(x))$ .

We will denote deduction in $\mathtt{WE\text{–}Z}^+$ by $\vdash_+$.

## 3 The Light Functional (*Light Dialectica*) Interpretation

By $\mathtt{LD}$-*interpretation* we call below our adaptation of Gödel's functional (*Dialectica*) interpretation [2, 13] to the extraction of (more) efficient programs

[19] Very useful to prove soundness for the (light) $\mathtt{D}$-interpretation of $\to^+$, in Section 3.

[20] Section 1 of [14] contains complete proofs of all the results stated above.

[21] The usual formulation of Markov's principle as $\mathtt{AxMK}_1$ : $\neg\neg\exists z\, A_0(z) \to \exists z\, A_0(z)$ is equivalent over $\mathtt{WE\text{–}Z}$ to $\mathtt{AxMK}$. Also the form $\mathtt{AxMK}_2$ : $\neg\neg\exists z\, A_0(z) \to \exists z\, \neg\neg A_0(z)$, which was preferred by the authors of [16] because of complexity considerations. Our choice of $\mathtt{AxMK}$ is motivated by the particularity of $\exists^{\mathrm{fl}}$ in [6, 34].

from (classical) proofs. It is a recursive syntactic translation from proofs in $\mathtt{WE}$–$\mathtt{Z}^+$ (or in $\mathtt{WE}$–$\mathtt{Z}^{\omega+}$, after a double-negation translation, see Section 3.2) to proofs in $\mathtt{WE}$–$\mathtt{Z}^-$ such that positive occurrences of $\exists$ and negative occurrences of $\forall$ in the proof's conclusion get actually realized by terms in Gödel's $\mathbf{T}$. These *realizing terms* are also called *the programs extracted* by the LD-interpretation and (if only the extracted programs are wanted) the translation process is also referred to as *program-extraction*. The translated proof is also called the *verifying proof* since it verifies the fact that the extracted programs actually *realize* the LD-interpretation of the conclusion of the proof at input. Gödel's Dialectica interpretation (abbreviated D-*interpretation*) is relatively (far) more complicated when it has to face *contraction*, which in Natural Deduction amounts to discharging more than one copy of an assumption in an Implication Introduction $\rightarrow^+$. Kohlenbach presents in [21] an elegant way of simplifying the treatment of contraction when the goal is to extract Howard majorizing functionals for the Dialectica realizers. He named "monotone functional interpretation" this variant of the D-interpretation which we here abbreviate by MD-*interpretation*. The MD-interpretation has been used with great success over the last years for producing mathematical proofs to important new theorems in numerical functional analysis[22].

However, when the extraction of *exact* realizers is concerned, the monotone D-interpretation does not necessarily bring an efficient answer. A different kind of optimization of Gödel's D-interpretation appears to be necessary. We here propose the LD-interpretation as a refinement of Gödel's technique which allows for the extraction of more efficient exact realizers. Moreover, the same refinement equally applies to the extraction of more efficient bounds via what might be called the *light monotone*[23] functional interpretation (abbreviated LMD-interpretation).

The D-interpretation was first introduced in [13] for a Hilbert-style formulation of Arithmetic – see also [2, 10, 20, 26, 38] for other (more modern) formulations within Hilbert-style systems. Natural Deduction formulations of the Diller-Nahm [10] variant of D-interpretation were provided by Diller's students Rath [33] and Stein [37]. Only in the year 2001 Jørgensen [19] provided a first Natural Deduction formulation of the original Gödel's functional interpretation. In the Diller-Nahm setting all choices between the potential realizers of a contraction are postponed to the very end by collecting all candidates and making a single final global choice. In contrast, Jørgensen's formulation respects Gödel's original treatment of contraction by immediate (local) choices. Jørgensen devises a so-called "Contraction Lemma" in order to handle (in the given Natural Deduction context) the discharging of more than one copy of an assumption in an Implication Introduction $\rightarrow^+$. If $n+1$ undischarged occurrences of an assumption are to be canceled in an $\rightarrow^+$, then Jørgensen uses his Contraction Lemma $n$ times, shifting partial results $n$ times back and forth over the "proof gate" $\vdash$. We consider this to be less efficient from the applied program-extraction perspec-

[22] Papers [22] and [20] contain comprehensive surveys of such to-day applications of MD-interpretation to concrete mathematical proofs from the literature.

[23] Or *light bounded* functional (*Dialectica*) interpretation, following [11] instead of [21].

tive. We also think that the soundness proof for the (L)D-interpretation is thus somewhat more complicated w.r.t. contraction. We will here use the $n$-selector $\mathtt{If}^n_\tau$ for equalizing in one single (composed) step all the (L)D-interpretations of the $n+1$ undischarged occurrences (see the proof of Theorem 34 below). While technically impossible to have a direct $n$-selector available for all $n \in \mathbb{N}$, in actual optimizing implementations $\mathtt{If}^n_\tau$ could be given a (more) direct definition for $n \leq N$ for a certain convenient[24] upper margin $N$ instead of being simulated in terms of $n$ times $\mathtt{If}^1_\tau$. The practical gain w.r.t. Jørgensen's solution is that the handling of contraction is directly moved from the proof level to the term level: back-and-forth shifting over $\vdash$ is no longer required when building the verifying proof. We also modify Jørgensen's variant of D-interpretation by allowing free variables in the extracted terms. This corresponds to the formulation of Gödel's **T** with $\lambda$-abstraction as primitive and is more natural in a Natural Deduction setting. In addition we include the treatment of our adaptation of Berger's uniform existential and universal quantifiers from [5] to the Dialectica-extraction context.

The light functional (*Dialectica*) interpretation/translation assigns a formula $A^{\mathtt{D}}(\underline{a}) \equiv \exists \underline{x}\, \forall \underline{y}\, A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a})$ with $A_{\mathtt{D}}$ not necessarily quantifier-free and $\underline{x}, \underline{y}$ tuples of fresh variables of finite type (such that $\{\underline{x}, \underline{y}, \underline{a}\}$ are all free variables of $A_{\mathtt{D}}$) to each instance of the formula $A(\underline{a})$ (with $\underline{a}$ all free variables of $A$). The types of $\underline{x}, \underline{y}$ depend only on the types of the regularly bound variables of $A$ and on the logical structure of $A$. We will also denote by $B^{\mathtt{D}}(\underline{b}) :\equiv \exists \underline{u}\, \forall \underline{v}\, B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{b})$ the LD-interpretation of $B(\underline{b})$. If non-ambiguous, we sometimes omit to display some of the free variables of the LD-translated formulas.

**Definition 31 (The light Dialectica interpretation of formulas)**

$$
\begin{aligned}
A^{\mathtt{D}} &:\equiv (A_{\mathtt{D}} :\equiv A) \text{ for prime formulas } A \\
(A \wedge B)^{\mathtt{D}} &:\equiv \exists \underline{x}, \underline{u}\, \forall \underline{y}, \underline{v}\, [\, (A \wedge B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) \wedge B_{\mathtt{D}}(\underline{u}; \underline{v}; \underline{b})\, ] \\
(\exists z A(z, \underline{a}))^{\mathtt{D}} &:\equiv \exists z^\dagger, \underline{x}\, \forall \underline{y}\, [\, (\exists z A(z, \underline{a}))_{\mathtt{D}}(z^\dagger, \underline{x}; \underline{y}; \underline{a}) :\equiv A_{\mathtt{D}}(\underline{x}; \underline{y}; z^\dagger, \underline{a})\, ] \\
(\forall z A(z, \underline{a}))^{\mathtt{D}} &:\equiv \exists \underline{X}\, \forall z^\dagger, \underline{y}\, [\, (\forall z A(z, \underline{a}))_{\mathtt{D}}(\underline{X}; z^\dagger, \underline{y}; \underline{a}) :\equiv A_{\mathtt{D}}(\underline{X}(z^\dagger); \underline{y}; z^\dagger, \underline{a})\, ] \\
(\overline{\exists} z A(z, \underline{a}))^{\mathtt{D}} &:\equiv \exists \underline{x}\, \forall \underline{y}\, [\, (\overline{\exists} z A(z, \underline{a}))_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) :\equiv \exists z\, A_{\mathtt{D}}(\underline{x}; \underline{y}; z, \underline{a})\, ] \\
(\overline{\forall} z A(z, \underline{a}))^{\mathtt{D}} &:\equiv \exists \underline{x}\, \forall \underline{y}\, [\, (\overline{\forall} z A(z, \underline{a}))_{\mathtt{D}}(\underline{x}; \underline{y}; \underline{a}) :\equiv \forall z\, A_{\mathtt{D}}(\underline{x}; \underline{y}; z, \underline{a})\, ] \\
(A \to B)^{\mathtt{D}} &:\equiv \exists \underline{Y}, \underline{U}\, \forall \underline{x}, \underline{v}\, [\, (A \to B)_{\mathtt{D}} :\equiv A_{\mathtt{D}}(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \to B_{\mathtt{D}}(\underline{U}(\underline{x}); \underline{v})\, ]
\end{aligned}
$$

Here $\cdot \mapsto \cdot^\dagger$ is a mapping which assigns to every given variable $z$ a completely new variable $z^\dagger$ which has the same type of $z$. Different variables $z^\dagger$ are returned for different applications on the same argument variable $z$ when processing a given formula $A$. This ensures that two nested quantifications of the same variable in $A$ are correctly distinguished in $A_{\mathtt{D}}$. The variables $\underline{X}, \underline{Y}, \underline{U}$ produced in the treatment of $\to$ and $\forall$ are also completely new but in contrast to the variables

[24] Only a certain limited number of $n$-selectors is needed for most practical applications.

produced by $\cdot^\dagger$, their type is strictly more complex than the type of the original variable. The free variables of $A^{\mathtt{D}}$ are exactly the free variables of $A$. If $A$ is quantifier-free then $A^{\mathtt{D}} = A_{\mathtt{D}} = A$.

**Remark 32** By abuse of notation from now on we use non-underlined letters also for tuples of objects (variables, terms, ...) and identify an individual object with the tuple containing only that object.

**Definition 33 (Dialectica terms)** For every `ncm`-quantifier free formula $A(a)$ we denote by $\mathtt{t}^{\mathtt{D}}_A[x;y;a]$ the boolean term associated to (the quantifier-free formula) $A_{\mathtt{D}}(x;y;a)$ by the mapping from Remark 22. We call it "the Dialectica term associated to" $A(a)$. The following holds:

$$\vdash_{-} A_{\mathtt{D}}(x;y;a) \leftrightarrow \mathtt{at}(\mathtt{t}^{\mathtt{D}}_A[x;y;a]) \tag{2}$$

**Theorem 34 (Exact realizer synthesis by the `LD`-interpretation)** There exists an algorithm which given at input a proof $\mathcal{P} : \{C^i\}_{i=1}^n \vdash_{+} A$ produces at output the tuples of terms $\{T_i\}_{i=1}^n$ and $T$, the tuples of variables $\{x_i\}_{i=1}^n$ and $y$ all together with the verifying proof $\mathcal{P}_{\mathtt{D}} : \{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \vdash_{-} A_{\mathtt{D}}(T(\underline{x}); y)$ – where $\underline{x} :\equiv x_1, \ldots, x_n$ . Moreover,

1. the variables $\underline{x}$ and $y$ do not occur in $\mathcal{P}$ (they are all completely new)
2. the free variables of $T$ and $\{T_i\}_{i=1}^n$ are among the free variables of $A$ and $\{C^i\}_{i=1}^n$ (we call this "the *free variable condition* `(FVC)` for programs extracted by the `D`-interpretation")

hence $\underline{x}$ and $y$ also do not occur free in the *extracted* terms $\{T_i\}_{i=1}^n$ and $T$.

**Proof:** The algorithm proceeds by recursion on the structure of the input proof $\mathcal{P}$. Realizing terms must be presented for all the axioms and then realizing terms for the conclusion of a rule must be deduced from terms which realize the premise of that rule. Since $\underline{x}, y$ are produced by the `LD`-interpretation of formulas (see Definition 31) it is immediate that they do not occur in $\mathcal{P}$. We present below only the (sub)case of Implication Introduction $\rightarrow^+$ in which at least two copies of the implicative assumption get canceled. Since it involves *contraction*, this case is far more difficult than all the other axioms and rules[25].

$\boxed{\dfrac{[A] \ldots / B}{A \rightarrow B} \rightarrow^+}$ We are given, with $n \geq 1$, $\underline{z} \equiv \overbrace{z, \ldots, z}^{n+1}$ and $\underline{x} \equiv x_{n+2}, \ldots, x_m$, that :

$$\{A_{\mathtt{D}}(z; T_i(\underline{z}, \underline{x}, y))\}_{i=1}^{n+1} \ , \ \{C^i_{\mathtt{D}}(x_i; T_i(\underline{z}, \underline{x}, y))\}_{i=n+2}^{m} \quad \vdash_{-} \quad B_{\mathtt{D}}(T(\underline{z}, \underline{x}); y) \tag{3}$$

It has been assumed that $n + 1 \leq m$, where $n + 1$ is the number of copies of the assumption $A$ which get discharged in this $\rightarrow^+$. Each of these $n + 1$ instances of $A$ produces the same tuple $z$ of existential variables under the `LD`-interpretation, see Definition 31. The $\mathtt{ncm\text{-}FC}(A)$ constraint ensures that the tuples $\{T_i\}_{i=1}^{n+1}$ are

[25] See the comments in the beginning of this section. A full proof is in Section 3 of [14].

all of length 0 (denoted $\sqcup$), i.e., $A$ is *computationally irrelevant*, orelse $A_{\mathtt{D}}$ is $\mathtt{qfr}$ (pre-condition for the association to $A$ of *Dialectica terms*, see Definition 33). Only in the former case can we directly discharge in a single $\rightarrow^+$ all $n+1 \geq 2$ copies of $A_{\mathtt{D}}(z;\sqcup)$ in the $\mathtt{LD}$-interpretation of the premise of this rule, see (3). In contrast, for the latter case we must first *equalize* the assumptions involving the $n+1$ terms $\{T_i\}_{i=1}^{n+1}$. This is because the terms $\{T_i\}_{i=1}^{n+1}$ can be mutually different since they are extracted from different sub-proofs which involve the different copies of $A$. We hereafter treat this case. We achieve the *equalizing* of $\{T_i\}_{i=1}^{n+1}$ in one single step[26], using the $n$-selector $\mathtt{If}_\tau^n$. For all $i \in \overline{1,n}$ let $T^i$ abbreviate $T_i(\underline{z},\underline{x},y)$ and let

$$\widetilde{S} \;:\equiv\; \lambda\underline{x},z,y.\,\mathtt{If}_\tau^n(\mathtt{t}_A^{\mathtt{D}}[z;T^1],\ldots,\mathtt{t}_A^{\mathtt{D}}[z;T^n],T_{n+1}(\underline{z},\underline{x},y),T^n,\ldots,T^1) \qquad (4)$$

By $n$ applications of $\overline{\forall}^-$ to $\mathtt{LmIf}_\tau^n$ with $\{p_i \leftarrow \mathtt{t}_A^{\mathtt{D}}[z;T^i]\}_{i=1}^n$ and (2) we get $\vdash_- \wedge_{j=1}^{i-1} A_{\mathtt{D}}(z;T^j) \wedge \neg A_{\mathtt{D}}(z;T^i) \rightarrow \widetilde{S}(\underline{x},z,y) = T_i(\underline{z},\underline{x},y)$ for all $i \in \overline{1,n}$ and $\vdash_- \wedge_{i=1}^n A_{\mathtt{D}}(z;T^i) \rightarrow \widetilde{S}(\underline{x},z,y) = T_{n+1}(\underline{z},\underline{x},y)$ from which we further obtain

$$\vdash_- [\wedge_{j=1}^{i-1} A_{\mathtt{D}}(z;T^j) \wedge \neg A_{\mathtt{D}}(z;T^i)] \rightarrow [A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge_{k=1}^{n+1} A_{\mathtt{D}}(z;T^k)]$$

$$\vdash_- [\wedge_{j=1}^{n} A_{\mathtt{D}}(z;T^j)] \rightarrow [A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge_{k=1}^{n+1} A_{\mathtt{D}}(z;T^k)]$$

for all $i \in \overline{1,n}$. We used one $\mathtt{COMPAT}$[27] in each of the $n+1$ above deductions and an $\mathtt{AxEFQ}$ in each of the the first $n$ of these. Due to the decidability of $\mathtt{qfr}$ formulas we have $\vdash_- \vee_{i=1}^n [\wedge_{j=1}^{i-1} A_{\mathtt{D}}(z;T^j) \wedge \neg A_{\mathtt{D}}(z;T^i)] \vee [\wedge_{j=1}^n A_{\mathtt{D}}(z;T^j)]$. It then follows by (1) that $\vdash_- A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow \wedge_{i=1}^{n+1} A_{\mathtt{D}}(z;T^i)$, hence for all $i \in \overline{1,n+1}$ we obtain $A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \vdash_- A_{\mathtt{D}}(z;T_i(\underline{z},\underline{x},y))$. We then discharge all assumptions $A_{\mathtt{D}}$ of (3) in $n+1$ applications of $\rightarrow^+$ and combine with the previously obtained proofs of $A_{\mathtt{D}}(z;T_i(\underline{z},\underline{x},y))$ in $n+1$ applications of $\rightarrow^-$ to conclude, with $S :\equiv \lambda\underline{x},z.\,T(\underline{z},\underline{x})$ and $\{S_i :\equiv \lambda\underline{x},z.\,T_i(\underline{z},\underline{x})\}_{i=n+2}^m$, that

$$\{A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y))\}_{i=1}^{n+1},\ \{C_{\mathtt{D}}^i(x_i;S_i(\underline{x},z,y))\}_{i=n+2}^m \vdash_- B_{\mathtt{D}}(S(\underline{x},z);y)\ .$$

We can now cancel all $\{A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y))\}_{i=1}^{n+1}$ in a single $\rightarrow^+$ and thus get

$$\{C_{\mathtt{D}}^i(x_i;S_i(\underline{x},z,y))\}_{i=n+2}^m \vdash_- A_{\mathtt{D}}(z;\widetilde{S}(\underline{x},z,y)) \rightarrow B_{\mathtt{D}}(S(\underline{x},z);y)\ .$$

Notice that $\mathtt{t}_A^{\mathtt{D}}$ introduces in $\widetilde{S}$ new occurrences of the free variables of $A$. We finally obtain, with the $\mathtt{FVC}$ obviously satisfied, that

$$\{C_{\mathtt{D}}^i(x_i;S_i(\underline{x},z,y))\}_{i=n+2}^m \vdash_-\ (A \rightarrow B)_{\mathtt{D}}(\,\widetilde{S}(\underline{x}),S(\underline{x})\,;z,y)\ .$$

□

**Corollary 35** There exists an algorithm which from a given proof $\vdash_+ A(a)$ produces exact realizing terms $T[a]$ with a verifying proof $\vdash_- \forall y A_{\mathtt{D}}(T;y;a)$.

[26] Jørgensen's solution uses here $n$ steps which correspond to the simulation of $\mathtt{If}_\tau^n$ in terms of $n$ instances of $\mathtt{If}_\tau^1$, see also the comments in the preamble of Section 3.

[27] Since $A_{\mathtt{D}}$ is quantifier-free, the restriction on undischarged assumptions is respected.

### 3.1 The Light Monotone (or Light Bounded) Functional Interpretation

We sketch below the combination of our refinement of Goedel's Dialectica interpretation [2, 13, 19] with its optimization for the extraction of bounds[28] due to Kohlenbach [21]. We add to systems $\mathtt{WE}$–$\mathtt{Z}^-$, $\mathtt{WE}$–$\mathtt{Z}$ and $\mathtt{WE}$–$\mathtt{Z}^+$ a functional inequality constant for naturals (of type $\iota\iota o$), denoted $\geq$. We define *predicate* inequality between terms $s^\iota$ and $t^\iota$ as an abbreviation for $\mathtt{at}(\geq s\,t)$, denoted $\geq_\iota$ . Similar to equality, predicate *inequality* at higher types $\tau \equiv \sigma_1 \dots \sigma_n \iota$ is extensionally defined as $s \geq_\tau t :\equiv \forall x_1^{\sigma_1}, \dots, x_n^{\sigma_n}\,(s\,x_1 \dots x_n \geq_\iota t\,x_1 \dots x_n)$. We also define Howard's *majorization* relation $\succeq_\tau$ by $\succeq_\iota \;:\equiv\; \geq_\iota$ and

$$x \succeq_{\sigma\tau} y \;:\equiv\; \forall z_1^\sigma, z_2^\sigma\;(z_1 \geq_\sigma z_2 \;\rightarrow\; xz_1 \geq_\tau yz_2)\;.$$

The monotonic systems $\mathtt{WE}$–$\mathtt{Z}^-_\mathtt{m}$, $\mathtt{WE}$–$\mathtt{Z}_\mathtt{m}$ and $\mathtt{WE}$–$\mathtt{Z}^+_\mathtt{m}$ are then obtained by further adding the axioms defining the usual behavior of predicate inequality on naturals: $z \geq_\iota 0\,,\; 0 \geq_\iota \mathtt{S}z \rightarrow \bot$ and $\mathtt{S}x \geq_\iota \mathtt{S}y \rightarrow x \geq_\iota y$. We denote by $\overset{\mathtt{m}}{\vdash}_-$, $\overset{\mathtt{m}}{\vdash}$ and $\overset{\mathtt{m}}{\vdash}_+$ deductions in $\mathtt{WE}$–$\mathtt{Z}^-_\mathtt{m}$, $\mathtt{WE}$–$\mathtt{Z}_\mathtt{m}$ and respectively $\mathtt{WE}$–$\mathtt{Z}^+_\mathtt{m}$.

**Conjecture 36 (Uniform bound synthesis by a LMD-interpretation)** There exists an algorithm which from a given proof $\overset{\mathtt{m}}{\vdash}_+ A(a)$ produces *closed* uniform bounds $T$ with a verifying proof $\overset{\mathtt{m}}{\vdash}_- \exists x(T \succeq x \wedge \forall a, y A_\mathtt{D}(x(a); y; a))$.

Notice that the optimization brought by the light MD-interpretation does not concern contraction (which is sufficiently handled by the pure MD-interpretation) as much as the diminishing of the maximal type degree of the extracted bounds.

### 3.2 Extensions of L(M)D-Interpretation to Extractions from Fully Classical Proofs. Systems $\mathtt{WE}$–$\mathtt{Z}^\mathtt{c}$, $\mathtt{WE}$–$\mathtt{Z}^{\mathtt{c}+}$ and $\mathtt{WE}$–$\mathtt{Z}^{\mathtt{c}+}_\mathtt{m}$

The system $\mathtt{WE}$–$\mathtt{Z}^\mathtt{c}$ of weakly extensional Classical Arithmetic is obtained by adding to $\mathtt{WE}$–$\mathtt{Z}$ the Stability principle:

$\mathtt{AxSTAB}$ : $\quad \neg\neg A \rightarrow A$ (Stability)

In fact it would be sufficient that $\mathtt{AxSTAB}$ replaces $\mathtt{AxEFQ}$ since the latter is deducible from $\mathtt{AxSTAB}$ in Minimal Logic. However we keep $\mathtt{AxEFQ}$ as axiom of $\mathtt{WE}$–$\mathtt{Z}^\mathtt{c}$ since it has simple Dialectica realizers[29]. The problem of $\mathtt{AxSTAB}$ is that it has no (direct) realizer under Dialectica interpretation – a preprocessing double negation translation, denoted $\cdot \mapsto \cdot^\mathtt{N}$ will be necessary to interpret fully classical systems (see, e.g., [16, 20, 22]).

We will denote by $\mathtt{WE}$–$\mathtt{Z}^{\mathtt{c}+}$ the system $\mathtt{WE}$–$\mathtt{Z}^\mathtt{c}$ extended with $\mathtt{AxAC}_\mathtt{o}$ (the quantifier-free version of $\mathtt{AxAC}$, obtained by restricting $B$ to $\mathtt{qfr}$ formulas). We will denote deductions in $\mathtt{WE}$–$\mathtt{Z}^\mathtt{c}$ and $\mathtt{WE}$–$\mathtt{Z}^{\mathtt{c}+}$ by $\vdash_\mathtt{c}$ and $\vdash_{\mathtt{c}+}$ respectively.

**Conjecture 37** There exists an algorithm which from a given proof $\vdash_{\mathtt{c}+} A(a)$ produces exact realizing terms $T[a]$ with a verifying proof $\vdash_- \forall y(A^\mathtt{N})_\mathtt{D}(T; y; a)$.

[28] See also [11] for a more recent adaptation of Dialectica to the extraction of bounds.
[29] Input proofs to the Dialectica-extraction algorithm may thus become shorter.

We denote by $\mathtt{WE\text{–}Z}^{c+}_{m}$, $\vdash^{m}_{c+}$ the monotonic correspondents of $\mathtt{WE\text{–}Z}^{c+}$ and $\vdash_{c+}$.

**Conjecture 38** There exists an algorithm which, given at input a proof $\vdash^{m}_{c+} A(a)$, produces at output *closed* uniform bounds $T$ and the verifying proof

$$\vdash^{m}_{-} \exists x(T \succeq x \wedge \forall a, y(A^{N})_{D}(x(a); y; a)).$$

## Acknowledgements

We would like to thank Prof. Helmut Schwichtenberg for his guidance of our interaction with MINLOG and for many useful discussions.

## References

1. J. Avigad. Interpreting classical theories in constructive ones. *The Journal of Symbolic Logic*, 65(4):1785–1812, 2000.
2. J. Avigad and S. Feferman. Gödel's functional ('Dialectica') interpretation. In S.R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 337–405. Elsevier, 1998.
3. F. Barbanera and S. Berardi. Extracting constructive content from classical logic via control–like reductions. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, pages 45–59. LNCS Vol. 664, 1993.
4. J.L. Bates and R.L. Constable. Proofs as programs. *ACM Transactions on Programming Languages and Systems*, 7(1):113–136, January 1985.
5. U. Berger. Uniform Heyting Arithmetic. *Annals of Pure and Applied Logic*, 133(1-3):125–148, May 2005. Festschrift for H. Schwichtenberg's 60th birthday.
6. U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.
7. U. Berger, H. Schwichtenberg, and M. Seisenberger. The Warshall algorithm and Dickson's lemma: Two examples of realistic program extraction. *Journal of Automated Reasoning*, 26:205–221, 2001.
8. R.L. Constable and C. Murthy. Finding computational content in classical proofs. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 341–362. Cambridge University Press, 1991.
9. T. Coquand and M. Hofmann. A new method for establishing conservativity of classical systems over their intuitionistic version. *Mathematical Structures in Computer Science*, 9(4):323–333, 1999.
10. J. Diller and W. Nahm. Eine Variante zur Dialectica Interpretation der Heyting Arithmetik endlicher Typen. *Archiv für Mathematische Logik und Grundlagenforschung*, 16:49–66, 1974.
11. F. Ferreira and P. Oliva. Bounded Functional Interpretation. *Annals of Pure and Applied Logic*, 48 pp., To appear, see Elseviers's *Science Direct* on the Internet.
12. H. Friedman. Classical and intuitionistically provably recursive functions. In G.H. Müller and D.S. Scott, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer Verlag, 1978.
13. K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958.
14. M.-D. Hernest. Technical Appendix to this paper. See the author's web-page.
15. M.-D. Hernest. A comparison between two techniques of program extraction from classical proofs. In M. Baaz, J. Makovsky, and A. Voronkov, editors, *CSL 2003: Extended Posters*, vol. VIII of *Kurt Gödel Society's Collegium Logicum*, pp. 99–102. Springer Verlag, 2004.

16. M.-D. Hernest and U. Kohlenbach. A complexity analysis of functional interpretations. *Theoretical Computer Science*, 338(1-3):200–246, 10 June 2005.
17. W.A. Howard. Hereditarily majorizable functionals of finite type. [38], pp. 454–461.
18. I. Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Matematica*, 4:119–136, 1936.
19. K.F. Jørgensen. Finite type arithmetic. Master's thesis, Departments of Mathematics and Philosophy, University of Roskilde, Roskilde, Denmark, 2001.
20. U. Kohlenbach. Proof Interpretations and the Computational Content of Proofs. *Lecture Course*, latest version in the author's web page.
21. U. Kohlenbach. Analysing proofs in Analysis. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors, *Logic: from Foundations to Applications, Keele, 1993*, European Logic Colloquium, pages 225–260. Oxford University Press, 1996.
22. U. Kohlenbach and P. Oliva. Proof mining: a systematic way of analysing proofs in Mathematics. *Proc. of the Steklov Institute of Mathematics*, 242:136–164, 2003.
23. G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland Publishing Company, 1959.
24. J.-L. Krivine. Classical logic, storage operators and second-order lambda-calculus. *Annals of Pure and Applied Logic*, 68:53–78, 1994.
25. D. Leivant. Syntactic translations and provably recursive functions. *The Journal of Symbolic Logic*, 50(3):682–688, September 1985.
26. H. Luckhardt. *Extensional Gödel Functional Interpretation*, volume 306 of *Lecture Notes in Mathematics*. Springer Verlag, 1973.
27. H. Luckhardt. Bounds extracted by Kreisel from ineffective proofs. In P. Odifreddi, editor, *Kreiseliana: About and around Georg Kreisel*, pp. 289–300. A.K. Peters, Wellesley, MA, 1996.
28. C. Murthy. Extracting constructive content from classical proofs. Tech. Report 90–1151, Dep.of Comp.Science, Cornell Univ., Ithaca, NY, U.S.A.,1990. PhD thesis.
29. G.E. Ostrin and S.S. Wainer. Elementary arithmetic. *Annals of Pure and Applied Logic*, 133(1-3):275–292, May 2005. Festschrift for H. Schwichtenberg's 60s.
30. M. Parigot. $\lambda\mu$–calculus: an algorithmic interpretation of classical natural deduction. In *Proc. of Log. Prog. and Automatic Reasoning, St. Petersburg*, volume 624 of *LNCS*, pages 190–201. Springer Verlag, Berlin, Heidelberg, New York, 1992.
31. C. Paulin-Mohring and B. Werner. Synthesis of ML programs in the system Coq. *Journal of Symbolic Computation*, 15(5/6):607–640, 1993.
32. C. Raffalli. Getting results from programs extracted from classical proofs. *Theoretical Computer Science*, 323(1-3):49–70, 2004.
33. P. Rath. *Eine verallgemeinerte Funktionalinterpretation der Heyting Arithmetik endlicher Typen*. PhD thesis, Universität Münster, Germany, 1978.
34. H. Schwichtenberg. *Minimal logic for computable functions*. Lecture course on program-extraction from (classical) proofs. Author's page or MINLOG distrib. [36].
35. H. Schwichtenberg. Monotone majorizable functionals. *Studia Logica*, 62:283–289, 1999.
36. H. Schwichtenberg and Others. Proof- and program-extraction system MINLOG. Free code and documentation at http://www.minlog-system.de.
37. M. Stein. *Interpretation der Heyting-Arithmetik endlicher Typen*. PhD thesis, Universität Münster, Germany, 1976.
38. A.S. Troelstra, editor. *Metamathematical investigation of intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, 1973.