Dialectica Interpretation of Well-Founded Induction

Helmut Schwichtenberg[*1]

[1] Mathematisches Institut der LMU, Theresienstr. 39, D-80333 München

Received 11. April 2007, revised 9. November 2007, accepted ?
Published online ?

Key words Dialectica interpretation, Euclid's theorem, program extraction.
MSC (2000) 03F10, 03F25, 03F35

From a classical proof that the gcd of natural numbers a_1 and a_2 is a linear combination of the two, we extract by Gödel's Dialectica interpretation an algorithm computing the coefficients. The proof uses the minimum principle. We show generally how well-founded recursion can be used to Dialectica interpret well-founded induction, which is needed in the proof of the minimum principle. In the special case of the example above it turns out that we obtain a reasonable extracted term, representing an algorithm close to Euclid's.

Finding and extracting computational content in existence proofs is a challenging subject, particularly so when the proofs do not seem to have such content. This is regularly the case when what is proved is only a "weak" existential formula, that is, a formula $\forall_x \tilde{\exists}_y A_0(x,y)$ with $\tilde{\exists}$ the weak existential quantifier defined by $\tilde{\exists}_y A_0(x,y) := \neg\forall_y \neg A_0(x,y)$, and A_0 quantifier-free.

Yiannis Moschovakis suggested the following example: the gcd of natural numbers a_1 and a_2 is a linear combination of the two. This proof uses the minimum principle. Here we treat that example as a case study for program extraction from classical proofs by Gödel's Dialectica interpretation. We show generally how well-founded recursion can be used to Dialectica interpret well-founded induction, which is needed in the proof of the minimum principle. In the special case of the example above it turns out that we obtain a reasonable extracted term, representing an algorithm close to Euclid's.

In [3] the same example has already been treated, but with a different method: a refined [1] form of the "A-translation" [5, 4]. The extracted algorithm was again close to Euclid's. The work in [3] would probably benefit as well from the use of well-founded induction. This and also a detailed comparison of the two methods is left for future work.

1 Arithmetic in finite types

We use a standard formalization HA^ω of arithmetic in finite types, based on natural deduction; cf. [10]. In fact, its "negative" fragment suffices, because we will only need implication and universal quantification to build formulas. For simplicity we take as the only base types the type $\mathbf{N}$ of natural numbers (generated by the constructors zero 0 and successor S), and $\mathbf{B}$ of booleans (with constructors $\mathtt{tt}$ and $\mathtt{ff}$).

1.1 Language

Types are built from base types $\mathbf{N}$ and $\mathbf{B}$ by arrows $\rho \to \sigma$ and products $\rho \times \sigma$. (Typed) *terms* are formed from typed variables and constants by (type correct) lambda abstraction, application, pairing and projections; for the latter we use the notation $r0$, $r1$ (for the left/right component of r). *Constants* (including the structural and well-founded recursion operators; see below) are defined by *computation rules*. We assume that the constants denote total functions only. It is well known that under the standard conversion rules plus the computation rules every term posesses a unique normal form, which in fact can be computed via *normalization by evaluation*; see [2]. To avoid equality reasoning in formal derivations we identify terms with the same normal form. The only predicates

* Corresponding author: e-mail: schwicht@math.lmu.de, Phone: +49 89 2180 4413, Fax: +49 89 2180 4466

we admit are Leibniz equality Eq_ι, for our two base types $\iota = \mathbf{N}$ and $\iota = \mathbf{B}$. The axioms are

$$\mathrm{Eq}^+ \colon \forall_x \mathrm{Eq}(x,x),$$
$$\mathrm{Eq}^- \colon \forall_{x,y}\big(\mathrm{Eq}(x,y) \to \forall_x C(x,x) \to C(x,y)\big).$$

One easily proves symmetry, transitivity and

Lemma (Compatibility) $\forall_{x,y}\big(\mathrm{Eq}(x,y) \to A(x) \to A(y)\big)$.

Proof. Use Eq^-, with $C(x,y) := A(x) \to A(y)$. □

Lemma (Ex-Falso-Quodlibet) *Define* falsity *by* $F := \mathrm{Eq}_{\mathbf{B}}(\mathtt{ff}, \mathtt{tt})$. *Then*

$$F \to A.$$

Proof. We first show that $F \to \mathrm{Eq}(x,y)$. To see this, notice that from $\mathrm{Eq}(\mathtt{ff},\mathtt{tt})$ we obtain

$$\mathrm{Eq}[\textbf{if } \mathtt{tt} \textbf{ then } x \textbf{ else } y][\textbf{if } \mathtt{ff} \textbf{ then } x \textbf{ else } y]$$

by compatibility. Hence $\mathrm{Eq}(x,y)$. The claim follows by induction on formulas. □

A further crucial use of the equality predicate Eq is that it allows to lift a boolean term $r^{\mathbf{B}}$ to a formula, using $\mathrm{atom}(r^{\mathbf{B}}) := \mathrm{Eq}(r^{\mathbf{B}}, \mathtt{tt})$. This opens up a convenient way to deal with equality on $\mathbf{N}$: notice that we can define decidable equality as a boolean-valued function $=_{\mathbf{N}} \colon \mathbf{N} \to \mathbf{N} \to \mathbf{B}$. The computation rules ensure that for instance the boolean term $\mathrm{S}(r) =_{\mathbf{N}} \mathrm{S}(s)$ is identified with $r =_{\mathbf{N}} s$. We can now turn this boolean term into the formula $\mathrm{Eq}(\mathrm{S}(r) =_{\mathbf{N}} \mathrm{S}(s), \mathtt{tt})$, which again is abbreviated by $\mathrm{S}(r) =_{\mathbf{N}} \mathrm{S}(s)$, but this time with the understanding that it is a formula. Then the two formulas $\mathrm{S}(r) =_{\mathbf{N}} \mathrm{S}(s)$ and $r =_{\mathbf{N}} s$ are identified, and consequently there is no need to prove such trivial propositions explicitely.

Negation is defined by $\neg A := A \to F$.

1.2 Derivation terms

It will be convenient to write derivations as terms, where the derived formula is viewed as the type of the term. This representation is known under the name *Curry-Howard correspondence*.

We give an inductive definition of derivation terms in Table 1, where for clarity we have written the corresponding derivations to the left. For the universal quantifier $\forall$ there is an introduction rule $\forall^+ x$ and an elimination rule $\forall^-$, whose right premise is the term r to be substituted. The rule $\forall^+ x$ is subject to the standard *(Eigen-) variable condition*: The derivation term M of the premise A should not contain any open assumption with x as a free variable.

1.3 Well-founded induction and recursion

1.3.1 Well-founded induction

Structural induction is naturally connected with the inductive generation of free algebras: at each point one recurs to its immediate predecessors. The reason for the validity of this induction principle is of course the fact that free algebras are well-founded. We now study a more general form of induction, called "well-founded induction", which allows recurrence to *all* points "strictly below" the present one. For applications it is best to make the necessary comparisons w.r.t. a "measure function" μ. Then it suffices to use an initial segment of the ordinals instead of a well-founded set. For simplicity we here restrict ourselves to the segment given by ω, so the ordering we refer to is just the standard $<$-relation on the natural numbers. The principle of well-founded induction is

$$\mathrm{GInd}_{n,A} \colon \forall_{\mu,x}\big(\mathrm{Prog}^\mu_x A(x) \to A(x)\big) \tag{1}$$

where $\mathrm{Prog}^\mu_x A(x)$ expresses "progressiveness" w.r.t. the measure function μ and the ordering $<$:

$$\mathrm{Prog}^\mu_x A(x) := \forall_x\big(\forall_{y;\mu y<\mu x} A(y) \to A(x)\big).$$

It is easy to see that in our special case of the $<$-relation we can *prove* (1) from structural induction. However, using well-founded induction as a primitive axiom has an advantage when we consider its computational content, which is well-founded recursion.

derivation	term
$u\colon A$	u^A
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A\to B}\to^+u$	$(\lambda_{u^A} M^B)^{A\to B}$
$\dfrac{\mid M \qquad \mid N}{A\to B \qquad A}\to^-$	$(M^{A\to B}N^A)^B$
$\dfrac{\mid M}{A}\forall^+x$ (with var.cond.)	$(\lambda_x M^A)^{\forall_x A}$ (with var.cond.)
$\dfrac{\mid M}{\forall_x A(x) \qquad r}\forall^-$	$(M^{\forall_x A(x)}r)^{A(r)}$

Table 1 Derivation terms for $\to$ and $\forall$.

1.3.2 Well-founded recursion

What was said above for *proof* by induction holds mutatis mutandis for *definition* by recursion, as a principle to define *total* functions. As in [9], we define the constant $\mathcal{F}$ of *well-founded recursion* by

$$\mathcal{F}\mu x G = Gx\big(\lambda_y[\textbf{if } \mu y < \mu x \textbf{ then } \mathcal{F}\mu y G \textbf{ else } \varepsilon]\big), \tag{2}$$

where ε denotes a canonical inhabitant of the value type. In our special case of the $<$-relation well-founded recursion is easily definable from structural recursion; the details are spelled out in [9, p.399–400]. However, well-founded recursion is preferable from an efficiency point of view.

2 Gödel's Dialectica interpretation

In his original functional interpretation [6], Gödel assigned to every formula A a new one $\exists_{\vec{x}}\forall_{\vec{y}} A_D(\vec{x},\vec{y})$ with $A_D(\vec{x},\vec{y})$ quantifier-free. Here $\vec{x}$, $\vec{y}$ are lists of variables of finite types; the use of higher types is necessary even when the original formula A is first-order. He did this in such a way that whenever a proof of A in arithmetic is given, one could produce closed terms $\vec{r}$ such that the quantifier-free formula $A_D(\vec{r},\vec{y})$ is provable in T.

In [6] Gödel used a Hilbert-style proof calculus. However, since the realizers will be formed in a λ-calculus formulation of system T, Gödel's interpretation becomes a lot more perspicious when it is done for natural deduction, as in the present exposition. A difference to the earlier treatments of Jørgensen [8] and Hernest [7] is that we view open assumptions not as formulas, but as assumption variables. The well-known need for contractions then comes up in the (only) logical rule with two premises: modus ponens (or implication elimination

$\to^-$). We will need that for every quantifier-free formula C there is a boolean term r_C such that $C \leftrightarrow r_C = \mathrm{tt}$; but this clearly is the case for our language.

2.1 Positive and negative types

To determine the types of $\vec{x}$ and $\vec{y}$, we assign to every formula A objects $\tau^+(A)$, $\tau^-(A)$ (a type or the "nulltype" symbol ε). $\tau^+(A)$ is for the realizer, $\tau^-(A)$ for the challenge. We also extend the use of $\rho \to \sigma$ and $\rho \times \sigma$ to the nulltype symbol ε:

$$\begin{aligned}
(\rho \to \varepsilon) &:= \varepsilon, & (\rho \times \varepsilon) &:= \rho,\\
(\varepsilon \to \sigma) &:= \sigma, & (\varepsilon \times \sigma) &:= \sigma,\\
(\varepsilon \to \varepsilon) &:= \varepsilon, & (\varepsilon \times \varepsilon) &:= \varepsilon.
\end{aligned}$$

Then

$$\begin{aligned}
\tau^+(\mathrm{Eq}(r,s)) &:= \varepsilon, & \tau^-(\mathrm{Eq}(r,s)) &:= \varepsilon,\\
\tau^+(\forall_{x^\rho} A) &:= \rho \to \tau^+(A), & \tau^-(\forall_{x^\rho} A) &:= \rho \times \tau^-(A)
\end{aligned}$$

and for implication

$$\begin{aligned}
\tau^+(A \to B) &:= \big(\tau^+(A) \to \tau^+(B)\big) \times \big(\tau^+(A) \to \tau^-(B) \to \tau^-(A)\big),\\
\tau^-(A \to B) &:= \tau^+(A) \times \tau^-(B).
\end{aligned}$$

In case $\tau^+(A)$ ($\tau^-(A)$) is $\neq \varepsilon$ we say that A has *positive (negative) computational content*.

2.2 Gödel translation

For every formula A and terms r, s of type $\tau^+(A)$, $\tau^-(A)$ we define a new quantifier-free formula $|A|^r_s$ by induction on A. It is convenient here to allow a "nullterm" symbol ε, in case one of $\tau^+(A)$, $\tau^-(A)$ is the nulltype symbol, and to extend the use of term operations to it: $\varepsilon r := \varepsilon\varepsilon := \varepsilon 0 := \varepsilon 1 := \varepsilon$, $r\varepsilon := r$ and $\langle r, \varepsilon\rangle := \langle \varepsilon, r\rangle := r$.

$$\begin{aligned}
|\mathrm{Eq}(t_1,t_2)|^\varepsilon_\varepsilon &:= \mathrm{Eq}(t_1,t_2),\\
|\forall_x A(x)|^r_s &:= |A(s0)|^{r(s0)}_{s1},\\
|A \to B|^r_s &:= |A|^{s0}_{r1(s0)(s1)} \to |B|^{r0(s0)}_{s1}.
\end{aligned}$$

If r, s in $|A|^r_s$ are formed by the pair constructor in case they are of pair type, we have the easier-to-memorize equations

$$|\forall_x A(x)|^f_{s,t} = |A(s)|^{fs}_t, \quad |A \to B|^{f,g}_{s,t} = |A|^s_{gst} \to |B|^{fs}_t.$$

The formula $\exists_x \forall_y |A|^x_y$ is called the *Gödel translation* of A, where $\exists_x$, $\forall_y$ is missing if $\tau^+(A)$, $\tau^-(A)$ is the nulltype symbol, respectively.

Theorem (Soundness) *Let M be a derivation of A from assumptions $u_i\colon C_i$ (where $i = 1, \dots, n$). Let x_i of type $\tau^+(C_i)$ be variables for realizers of the assumptions, and y be a variable of type $\tau^-(A)$ for a challenge of the goal. Then we can find terms*

- $[\![M]\!]^+ =: t$ *of type* $\tau^+(A)$ *with* $y \notin \mathrm{FV}(t)$ *and*
- $[\![M]\!]^-_i =: r_i$ *of type* $\tau^-(C_i)$,

and a derivation of $|A|^t_y$ from assumptions $\bar{u}_i\colon |C_i|^{x_i}_{r_i}$.

The proof is by induction on M. It will be given in the following three sections: for the logic rules, for (ordinary) induction and for well-founded induction.

2.3 Soundness of logic

Case $u\colon A$. Let x of type $\tau^+(A)$ be the variable for a realizer of the assumption u. Define $[\![u]\!]^+ := x$ and $[\![u]\!]^- := y$.

Case $\lambda_{u^A} M^B$. By IH (induction hypothesis) we have a derivation of $|B|^t_z$ from $\bar{u}\colon |A|^x_r$ and $\bar{u}_i\colon |C_i|^{x_i}_{r_i}$, where $\bar{u}\colon |A|^x_r$ may be absent. Substitute $y0$ for x and $y1$ for z. By $(\to^+)$ we obtain $|A|^{y0}_{r[x,z:=y0,y1]} \to |B|^{t[x:=y0]}_{y1}$, which is (up to β-conversion)

$$|A \to B|^{\lambda_x t, \lambda_{x,z} r}_y, \quad \text{from} \quad \bar{u}'_i\colon |C_i|^{x_i}_{r_i[x,z:=y0,y1]}.$$

Here r is the canonical inhabitant of the type $\tau^-(A)$ in case $\bar{u}\colon |A|^x_r$ is absent. Hence we can define the required terms by (assuming that u^A is u_1)

$$[\![\lambda_u M]\!]^+ := (\lambda_x [\![M]\!]^+, \lambda_{x,z} [\![M]\!]^-_1), \qquad [\![\lambda_u M]\!]^-_i := [\![M]\!]^-_{i+1}[x, z := y0, y1].$$

Case $M^{A\to B} N^A$. By IH we have a derivation of

$$\begin{array}{ll} |A \to B|^t_x = |A|^{x0}_{t1(x0)(x1)} \to |B|^{t0(x0)}_{x1} & \text{from } |C_i|^{x_i}_{p_i}, |C_k|^{x_k}_{p_k}, \text{ and of} \\ |A|^s_z & \text{from } |C_j|^{x_j}_{q_j}, |C_k|^{x_k}_{q_k}. \end{array}$$

Substituting $\langle s, y\rangle$ for x in the first derivation and of $t1sy$ for z in the second derivation gives

$$\begin{array}{ll} |A|^s_{t1sy} \to |B|^{t0s}_y & \text{from } |C_i|^{x_i}_{p'_i}, |C_k|^{x_k}_{p'_k}, \text{ and} \\ |A|^s_{t1sy} & \text{from } |C_j|^{x_j}_{q'_j}, |C_k|^{x_k}_{q'_k}. \end{array}$$

Now we contract $|C_k|^{x_k}_{p'_k}$ and $|C_k|^{x_k}_{q'_k}$: since $|C_k|^{x_k}_w$ is quantifier-free, there is a boolean term r_{C_k} such that

$$|C_k|^{x_k}_w \leftrightarrow r_{C_k} w = \mathtt{tt}. \tag{3}$$

Hence with $r_k := [\textbf{if } r_{C_k} p'_k \textbf{ then } q'_k \textbf{ else } p'_k]$ we can derive both $|C_k|^{x_k}_{p'_k}$ and $|C_k|^{x_k}_{q'_k}$ from $|C_k|^{x_k}_{r_k}$. The derivation proceeds by cases on the boolean term $r_{C_k} p'_k$. If it is true, then r_k converts into q'_k, and we only need to derive $|C_k|^{x_k}_{p'_k}$. But this follows by substituting p'_k for w in (3). If $r_{C_k} p'_k$ is false, then r_k converts into p'_k, and we only need to derive $|C_k|^{x_k}_{q'_k}$, from $|C_k|^{x_k}_{p'_k}$. But the latter implies $\mathtt{ff} = \mathtt{tt}$ (substitute again p'_k for w in (3)) and therefore every quantifier-free formula, in particular $|C_k|^{x_k}_{q'_k}$.

Using $(\to^-)$ we obtain

$$|B|^{t0s}_y \quad \text{from } |C_i|^{x_i}_{p'_i}, |C_j|^{x_j}_{q'_j}, |C_k|^{x_k}_{r_k}.$$

Let $[\![MN]\!]^+ := t0s$ and $[\![MN]\!]^-_i := p'_i$, $[\![MN]\!]^-_j := q'_j$, $[\![MN]\!]^-_k := r_k$.

Case $\lambda_x M^{A(x)}$. By IH we have a derivation of $|A(x)|^t_z$ from $\bar{u}_i\colon |C_i|^{x_i}_{r_i}$. Substitute $y0$ for x and $y1$ for z. We obtain $|A(y0)|^{t[x:=y0]}_{y1}$, which is (up to β-conversion)

$$|\forall_x A(x)|^{\lambda_x t}_y, \quad \text{from} \quad \bar{u}'_i\colon |C_i|^{x_i}_{r_i[x,z:=y0,y1]}.$$

Hence we can define the required terms by

$$\begin{aligned} &[\![\lambda_x M]\!]^+ := \lambda_x [\![M]\!]^+, \\ &[\![\lambda_x M]\!]^-_i := [\![M]\!]^-_i [x, z := y0, y1]. \end{aligned}$$

Case $M^{\forall_x A(x)} s$. By IH we have a derivation of $|\forall_x A(x)|^t_z = |A(z0)|^{t(z0)}_{z1}$ from $|C_i|^{x_i}_{r_i}$. Substituting $\langle s, y\rangle$ for z gives

$$|A(s)|^{ts}_y \quad \text{from } |C_i|^{x_i}_{r_i[z:=\langle s,y\rangle]}.$$

Let $[\![Ms]\!]^+ := ts$ and $[\![Ms]\!]^-_i := r_i[z := \langle s, y\rangle]$.

2.4 Soundness of ordinary induction

We consider induction for the natural numbers, given by constructors 0 and S; for boolean induction the argument is similar (and simpler). The induction schema then reads

$$\mathrm{Ind}_{n,A}\colon \forall_n\big(A(0) \to \forall_m(A(m) \to A(m+1)) \to A(n)\big).$$

Let $B(n) := A(0) \to \forall_m(A(m) \to A(m+1)) \to A(n)$. Clearly we can derive $B(0)$ and $B(n) \to B(n+1)$. By those parts of the proof of the Soundness Theorem that we have dealt with already, we obtain realizing terms s and t, r and derivations of $|B(0)|^s_y$ and of $|B(n) \to B(n+1)|^{t,r}_{x,u}$, hence of

$$\begin{aligned}
&|B(n)|^x_{rxu} \to |B(n+1)|^{tx}_u\\
&\forall_y|B(n)|^x_y \to |B(n+1)|^{tx}_u\\
&\forall_y|B(n)|^x_y \to \forall_y|B(n+1)|^{tx}_y.
\end{aligned}$$

So if we define $g(0) := s$ and $g(n+1) := t(g(n))$, then we have proved by induction that $\forall_y|B(n)|^{g(n)}_y$, which is $\forall_y|\forall_n B(n)|^g_{n,y}$.

However, for an implementation of the Dialectica interpretation it is advisable to replace axioms by rules whenever possible. In particular, more perspicious realizers for proofs involving induction can be obtained if the induction axiom appears with sufficiently many arguments, so that it can be seen as an application of an induction rule. Note that this can always be achieved by means of η-expansion. Moreover, in this way we are able to stay within a quantifier-free setup.

Consider $\mathrm{Ind}_{n,A}\vec{a}aM_0^{A(0)}M_1^{\forall_n(A(n)\to A(n+1))}$. By IH we have derivations of

$$\begin{aligned}
&|\forall_n(A(n) \to A(n+1))|^t_{n,f,y} =\\
&|A(n) \to A(n+1)|^{tn}_{f,y} =\\
&|A(n)|^f_{tn1fy} \to |A(n+1)|^{tn0f}_y \quad \text{from } |C_i|^{x_i}_{r_{i1}(n,f,y)}
\end{aligned}$$

and of

$$|A(0)|^{t_0}_{x_0} \quad \text{from } |C_i|^{x_i}_{r_{i0}(x_0)}.$$

i ranges over all assumption variables in $\mathrm{Ind}_{n,A}\vec{a}aM_0M_1$ (if necessary choose canonical terms r_{i0} and r_{i1}). It suffices to construct terms (involving recursion operators) $\tilde{t}$, $\tilde{r}_i$ with free variables among $\vec{x}$ such that

$$\forall_{m,y}\big((|C_i|^{x_i}_{\tilde{r}_i m y})_i \to |A(m)|^{\tilde{t}m}_y\big), \tag{4}$$

where $(C_i)_i \to A$ is short for $C_1 \to \cdots \to C_n \to A$. For then we can define $[\![\mathrm{Ind}_{n,A}\vec{a}aM_0M_1]\!]^+ := \tilde{t}a$ and $[\![\mathrm{Ind}_{n,A}\vec{a}aM_0M_1]\!]^-_i := \tilde{r}_i ay$. The recursion equations for $\tilde{t}$ are

$$\tilde{t}0 = t_0, \quad \tilde{t}(n+1) = tn0(\tilde{t}n)$$

and for $\tilde{r}_i$

$$\tilde{r}_i 0y = r_{i0}(y), \quad \tilde{r}_i(n+1)y = \begin{cases} r_{i1}(n, \tilde{t}n, y) =: s_i(y) & \text{if } \neg|C_i|^{x_i}_{s_i(y)},\\ \tilde{r}_i n(tn1(\tilde{t}n)y) & \text{otherwise.}\end{cases}$$

$\tilde{t}$, $\tilde{r}_i$ can be written explicitely with recursion operators:

$$\begin{aligned}
&\tilde{t}m = \mathcal{R}mt_0(\lambda_n(tn0)),\\
&\tilde{r}_i m = \mathcal{R}m(\lambda_y r_{i0})\big(\lambda_{n,p,y}[\textbf{if } r_{C_i}s_i(y) \textbf{ then } p(tn1(\tilde{t}n)y) \textbf{ else } s_i(y)]\big)
\end{aligned}$$

with $s_i(y)$ as above. It remains to prove (4). We shall do this by quantifier-free induction. To this end, define

$$\tilde{s}0ym := y, \quad \tilde{s}(k+1)ym := t(m \dot{-} k \dot{-} 1)1\big(\tilde{t}(m \dot{-} k \dot{-} 1)\big)(\tilde{s}kym).$$

We prove by induction on n that

$$n \le m \to \big(|C_i|^{x_i}_{\tilde{r}_i n(\tilde{s}(m \dot{-} n)ym)}\big)_i \to |A(n)|^{\tilde{t}n}_{\tilde{s}(m \dot{-} n)ym}. \tag{5}$$

Then (4) will follow with $n := m$. For the base case $n = 0$ we must show

$$\big(|C_i|^{x_i}_{\tilde{r}_i 0(\tilde{s}mym)}\big)_i \to |A(0)|^{\tilde{t}0}_{\tilde{s}mym}.$$

Recall that the global IH (for the base derivation) gives with $x_0 := \tilde{s}mym$

$$\big(|C_i|^{x_i}_{r_{i0}(\tilde{s}mym)}\big)_i \to |A(0)|^{t_0}_{\tilde{s}mym}.$$

By definition of $\tilde{t}$ and $\tilde{r}_i$ this is what we want. Now consider the successor case. Assume $n + 1 \le m$. We write $\tilde{s}k$ for $\tilde{s}kym$. Notice that for $k + 1 = m \dot{-} n$ by definition of $\tilde{s}$ we have $\tilde{s}(m \dot{-} n) = tn1(\tilde{t}n)(\tilde{s}(m \dot{-} n \dot{-} 1))$. Assume $|C_i|^{x_i}_{\tilde{r}_i(n+1)(\tilde{s}(m \dot{-} n \dot{-} 1))}$ for all i. We must show $|A(n+1)|^{\tilde{t}(n+1)}_{\tilde{s}(m \dot{-} n \dot{-} 1)}$. Let y_0 abbreviate $\tilde{s}(m \dot{-} n \dot{-} 1)$. If $\neg|C_i|^{x_i}_{s_i(y_0)}$ for some i, then by definition $\tilde{r}_i(n+1)y_0 = s_i(y_0)$ and we have $|C_i|^{x_i}_{s_i(y_0)}$, a contradiction. Hence $|C_i|^{x_i}_{s_i(y_0)}$ for all i, and therefore $\tilde{r}_i(n+1)y_0 = \tilde{r}_i n(tn1(\tilde{t}n)y_0) = \tilde{r}_i n(\tilde{s}(m \dot{-} n))$. The IH (5) therefore gives $|A(n)|^{\tilde{t}n}_{\tilde{s}(m \dot{-} n)}$, because of our initial assumptions $|C_i|^{x_i}_{\tilde{r}_i(n+1)y_0}$. Recall that the global IH (for the step derivation) gives with $f := \tilde{t}n$ and $y := \tilde{s}(m \dot{-} n \dot{-} 1)$

$$\big(|C_i|^{x_i}_{s_i(\tilde{s}(m \dot{-} n \dot{-} 1))}\big)_i \to |A(n)|^{\tilde{t}n}_{\tilde{s}(m \dot{-} n)} \to |A(n+1)|^{\tilde{t}(n+1)}_{\tilde{s}(m \dot{-} n \dot{-} 1)}$$

and we are done.

2.5 Soundness of well-founded induction

We now treat well-founded induction. Consider $\mathrm{GInd}_{n,A}\vec{a}hkM^{\mathrm{Prog}^h_n A(n)} \colon A(n)$. By IH we can derive

$$\begin{aligned}
&|\mathrm{Prog}^h_n A(n)|^t_{n,f,z} = \\
&|\forall_n(\forall_{m;hm<hn} A(m) \to A(n))|^t_{n,f,z} = \\
&|\forall_{m;hm<hn} A(m) \to A(n)|^{tn}_{f,z} = \\
&|\forall_{m;hm<hn} A(m)|^f_{tn1fz} \to |A(n)|^{tn0f}_z = \\
&\big(h(tn1fz0) < hn \to |A(tn1fz0)|^{f(tn1fz0)}_{tn1fz1}\big) \to |A(n)|^{tn0f}_z \quad \text{from } |C_i|^{x_i}_{r_i(n,f,z)},
\end{aligned}$$

where i ranges over all assumption variables in $\mathrm{GInd}_{n,A}\vec{a}hkM$ (if necessary choose canonical terms r_i). It suffices to construct terms (involving well-founded recursion operators) $\tilde{t}$, $\tilde{r}_i$ with free variables among $\vec{x}$ such that

$$\forall_{n,z}\big((|C_i|^{x_i}_{\tilde{r}_i nz})_i \to |A(n)|^{\tilde{t}n}_z\big), \tag{6}$$

for then we can define $[\![\mathrm{GInd}_{n,A}\vec{a}hkM]\!]^+ = \tilde{t}k$ and $[\![\mathrm{GInd}_{n,A}\vec{a}hkM]\!]^-_i = \tilde{r}_i kz$. The recursion equations for $\tilde{t}$ and $\tilde{r}_i$ are

$$\tilde{t}n = tn0[\tilde{t}]_{<hn}, \quad \tilde{r}_i nz = \begin{cases} r_i(n, [\tilde{t}]_{<hn}, z) =: s & \text{if } \neg|C_i|^{x_i}_s, \\ [\tilde{r}_i]_{<hn}(t'0)(t'1) & \text{otherwise,} \end{cases}$$

with the abbreviations

$$[r]_{<hn} := \lambda_m[\textbf{if } hm < hn \textbf{ then } rm \textbf{ else } \varepsilon], \quad t' := tn1[\tilde{t}]_{<hn}z.$$

It remains to prove (6). For its proof we use well-founded induction. Fix n. We can assume

$$\forall_{m;hm<hn}\forall_z\big((|C_i|^{x_i}_{\tilde{r}_i mz})_i \to |A(m)|^{\tilde{t}m}_z\big). \tag{7}$$

Fix z and assume $|C_i|^{x_i}_{\tilde{r}_i nz}$ for all i. We must show $|A(n)|^{\tilde{t}n}_z$. If $\neg|C_i|^{x_i}_s$ for some i, then by definition $\tilde{r}_i nz = s$ and we have $|C_i|^{x_i}_s$, a contradiction. Hence $|C_i|^{x_i}_s$ for all i, and therefore $\tilde{r}_i nz = [\tilde{r}_i]_{<hn}(t'0)(t'1)$. The IH (7) with $m := t'0$ and $z := t'1$ gives

$$h(t'0) < hn \to (|C_i|^{x_i}_{\tilde{r}_i(t'0)(t'1)})_i \to |A(t'0)|^{\tilde{t}(t'0)}_{t'1}.$$

Recall that the global IH (for the derivation of progressiveness) gives with $f := [\tilde{t}]_{<hn}$

$$(|C_i|^{x_i}_s)_i \to \big(h(t'0) < hn \to |A(t'0)|^{[\tilde{t}]_{<hn}(t'0)}_{t'1}\big) \to |A(n)|^{tn0[\tilde{t}]_{<hn}}_z.$$

Since $\tilde{t}(t'0) = [\tilde{t}]_{<hn}(t'0)$ and $\tilde{r}_i nz = [\tilde{r}_i]_{<hn}(t'0)(t'1) = \tilde{r}_i(t'0)(t'1)$ we are done.

Notice that we can view this proof as an application of *quantifier-free* well-founded induction, where the formula $(|C_i|^{x_i}_{\tilde{r}_i nz})_i \to |A(n)|^{\tilde{t}n}_z$ is proved w.r.t. the measure function $h'nz := hn$.

3 An application: Euclid's theorem

3.1 Informal proof

Theorem *Assume $0 < a_2$. Then there must exist natural numbers k_1, k_2 such that $0 < |k_1a_1 - k_2a_2|$ and $\mathrm{Rem}(a_i, |k_1a_1 - k_2a_2|) = 0$ $(i = 1, 2)$.*

Proof. Assume $0 < a_2$. Denote $|k_1a_1 - k_2a_2|$ by $h\vec{k}$ and $\mathrm{Rem}(a, b) = 0$ by $b \mid a$, and $(b \mid a_1) \wedge (b \mid a_2)$ by $b \mid a_1, a_2$. Since the theorem claims existence in the weak (or "classical") sense, from the "false" assumption

$$u\colon \forall_{\vec{k};0<h\vec{k}}(h\vec{k} \mid a_1, a_2 \to F)$$

we need to derive F. Assume u. It suffices to prove $\forall_{\vec{k};0<h\vec{k}}\, h\vec{k} \mid a_1, a_2$, for then the desired contradiction follows with $k_1 = 0$ and $k_2 = 1$, using the assumption $0 < a_2$. For the proof we use well-founded induction with measure h and formula $A(\vec{k}) := 0 < h\vec{k} \to h\vec{k} \mid a_1, a_2$. Therefore it suffices to prove

$$\begin{aligned}&\mathrm{Prog}^h := \mathrm{Prog}^h_{\vec{k}}\, A(\vec{k}) = \\ &\forall_{\vec{k}}\big(\forall_{\vec{l};0<h\vec{l}<h\vec{k}}\, h\vec{l} \mid a_1, a_2 \to 0 < h\vec{k} \to h\vec{k} \mid a_1, a_2\big).\end{aligned} \tag{8}$$

Fix $\vec{k}$ and assume

$$\begin{aligned}&u_1\colon \forall_{\vec{l};0<h\vec{l}<h\vec{k}}\, h\vec{l} \mid a_1, a_2,\\ &u_2\colon 0 < h\vec{k}.\end{aligned}$$

We must show $h\vec{k} \mid a_i$ for $i = 1, 2$. By symmetry it suffices to consider $i = 1$. Denote $\mathrm{Quot}(a_1, h\vec{k})$ by q and $\mathrm{Rem}(a_1, h\vec{k})$ by r. We must show $r = 0$. Because of $0 < h\vec{k}$ general properties of Quot and Rem ensure

$$a_1 = q \cdot h\vec{k} + r, \quad r < h\vec{k}.$$

From this we obtain

$$r = |\underbrace{\mathrm{Step}(a_1, a_2, k_1, k_2, q)}_{=:l_1} a_1 - \underbrace{qk_2}_{=:l_2} a_2| = h\vec{l} < h\vec{k},$$

where

$$\mathrm{Step}(a_1, a_2, k_1, k_2, q) := \begin{cases} qk_1 - 1 & \text{if } k_2a_2 < k_1a_1 \text{ and } 0 < q,\\ qk_1 + 1 & \text{otherwise.}\end{cases}$$

Assume $0 < h\vec{l}$. Then $h\vec{l} \mid a_1, a_2$ by u_1. Now u applied to $\vec{l}$ gives F. Therefore $0 < h\vec{l} \to F$ and hence $h\vec{l} = 0$. Now $r = h\vec{l}$ gives $r = 0$, as desired. □

3.2 Formalization

The informal proof has been given in such detail that it is now easy to formalize it completely. Let

$$M := \lambda_{\vec{a}}\lambda v_0^{0<a_2}\lambda u^{\forall_{\vec{k};0<h\vec{k}}(h\vec{k}|a_1,a_2\to F)}.$$
$$u01v_0\big(\mathrm{GInd}^{\forall_{\vec{a},\mu,\vec{k}}(\mathrm{Prog}^\mu\to\forall_{\vec{k};0<\mu\vec{k}}\mu\vec{k}|a_1,a_2)}\vec{a}h01M_{\mathrm{Prog}}v_0\big)$$

where

$$M_{\mathrm{Prog}} := \lambda_{\vec{k}}\lambda u_1^{\forall_{\vec{l};0<h\vec{l}<h\vec{k}}\,h\vec{l}|a_1,a_2}\lambda u_2^{0<h\vec{k}}\langle M_{h\vec{k}|a_1}, M_{h\vec{k}|a_2}\rangle,$$
$$M_{h\vec{k}|a_1} := L_1 r_1\big(h\vec{l_1}\big)M_{1,=}^{r_1=h\vec{l_1}}M_{1,\not<}^{0<h\vec{l_1}\to F},$$
$$M_{1,=} := L_{S1}\vec{a}\vec{k}q_1r_1(L_Qa_1(h\vec{k})u_2),$$
$$M_{1,\not<} := \lambda w^{0<h\vec{l_1}}.u\vec{l_1}wM_{1,\mathrm{div}}^{h\vec{l_1}|a_1,a_2},$$
$$M_{1,\mathrm{div}} := u_1\vec{l_1}w\big(L_2r_1(h\vec{l_1})(h\vec{k})M_{1,=}(L_Ra_1(h\vec{k})u_2)\big)$$

and similarly $M_{h\vec{k}|a_2}$, $M_{2,=}$, $M_{2,\not<}$, $M_{2,\mathrm{div}}$. We have used abbreviations

$$q_i := \mathrm{Quot}(a_i, h\vec{k}) \quad (i = 1, 2),$$
$$r_i := \mathrm{Rem}(a_i, h\vec{k}) \quad (i = 1, 2),$$
$$\vec{l_1} := (\mathrm{Step}(a_1, a_2, k_1, k_2, q_1), q_1k_2),$$
$$\vec{l_2} := (q_2k_1, \mathrm{Step}(a_2, a_1, k_2, k_1, q_2))$$

and lemmata

$$L_1\colon \forall_{r,l}(r = l \to (0 < l \to F) \to r = 0),$$
$$L_2\colon \forall_{r,l,k}(r = l \to r < k \to l < k),$$
$$L_Q\colon \forall_{a,b}(0 < b \to a = \mathrm{Quot}(a, b) \cdot b + \mathrm{Rem}(a, b)),$$
$$L_R\colon \forall_{a,b}(0 < b \to \mathrm{Rem}(a, b) < b),$$
$$L_{S1}\colon \forall_{\vec{a},\vec{k},q,r}(a_1 = q \cdot h\vec{k} + r \to r = h\vec{l_1}),$$
$$L_{S2}\colon \forall_{\vec{a},\vec{k},q,r}(a_2 = q \cdot h\vec{k} + r \to r = h\vec{l_2}).$$

3.3 Term extraction for M_{Prog}

We begin with some observations concerning special situations of extraction of terms from proofs, as treated generally in the proof of the Soundness Theorem.

1. Lemmata without positive content – for instance, purely universal ones – can be added as axioms in the statement of the Soundness Theorem, both in the premise and the conclusion.

2. Abstraction of an assumption variable for a quantifier-free formula does not affect the positive or negative content.

We now compute the Dialectica realizers and challenges for the derivations above. $M_{1,=}$ has neither positive nor negative content. $M_{h\vec{k}|a_1}$, $M_{1,\not<}$, $M_{1,\mathrm{div}}$ all have no positive content, and their negative content w.r.t. the free assumptions u and/or u_1 are always $\vec{l_1}$. $\langle M_{h\vec{k}|a_1}, M_{h\vec{k}|a_2}\rangle$ again has no positive content. Its negative content w.r.t. the shared assumption u_1 is to be formed by contraction:

$$[\textbf{if } 0 < h\vec{l_1} \to h\vec{l_1} < h\vec{k} \to h\vec{l_1} \mid a_1, a_2 \textbf{ then } \vec{l_2} \textbf{ else } \vec{l_1}]$$

and the negative content w.r.t. the shared assumption u is

$$[\textbf{if } 0 < h\vec{l_1} \to h\vec{l_1} \mid a_1, a_2 \to F \textbf{ then } \vec{l_2} \textbf{ else } \vec{l_1}].$$

Therefore for M_{Prog} we obtain

$$\begin{aligned} t := [\![M_{\mathrm{Prog}}]\!]^+ &= \lambda_{\vec{k}}[\textbf{if } 0<h\vec{l_1} \to h\vec{l_1}<h\vec{k} \to h\vec{l_1} \mid a_1, a_2 \textbf{ then } \vec{l_2} \textbf{ else } \vec{l_1}], \\ r(\vec{k}) := [\![M_{\mathrm{Prog}}]\!]^- &= [\textbf{if } 0 < h\vec{l_1} \to h\vec{l_1} \mid a_1, a_2 \to F \textbf{ then } \vec{l_2} \textbf{ else } \vec{l_1}] \\ &\quad\ [\textbf{if } 0 < h\vec{l_1} \wedge h\vec{l_1} \mid a_1, a_2 \textbf{ then } \vec{l_1} \textbf{ else } \vec{l_2}]. \end{aligned}$$

3.4 Term extraction for $\mathrm{GInd}\,\vec{a}h01M_{\mathrm{Prog}}$

We now specialize the general term extraction procedure for well-founded induction (cf. 2.5) to the present case. From the definition of Prog^h in (8) it is easy to see that $\tau^+(\mathrm{Prog}^h) = \mathbf{N}\times\mathbf{N} \to \mathbf{N}\times\mathbf{N}$ and $\tau^-(\mathrm{Prog}^h) = \mathbf{N}\times\mathbf{N}$. Using the notation form 2.5, f, z are not present here, and we have derived

$$|\mathrm{Prog}^h_{\vec{k}}\, A(\vec{k})|^t_{\vec{k}} = \big(h(t\vec{k}) < h\vec{k} \to A(t\vec{k})\big) \to A(\vec{k}) \quad \text{from } |C|^{\varepsilon}_{r(\vec{k})}.$$

Here $C := \forall_{\vec{k};0<h\vec{k}}(h\vec{k} \mid a_1, a_2 \to F)$ is the formula of the assumption variable u. Note that

$$\begin{aligned} &|C|^{\varepsilon}_{r(\vec{k})} = (0 < h(r(\vec{k})) \to h(r(\vec{k})) \mid a_1, a_2 \to F), \\ &\neg|C|^{\varepsilon}_{r(\vec{k})} \leftrightarrow 0 < h(r(\vec{k})) \wedge h(r(\vec{k})) \mid a_1, a_2. \end{aligned}$$

Therefore $A(\vec{k})$ is derivable from $|C|^{\varepsilon}_{\tilde{r}\vec{k}}$, with $\tilde{r}$ defined by well-founded recursion:

$$\tilde{r}\vec{k} = \begin{cases} r(\vec{k}) & \text{if } 0 < h(r(\vec{k})) \wedge h(r(\vec{k})) \mid a_1, a_2 \\ [\tilde{r}]_{<h\vec{k}}(t\vec{k}) & \text{otherwise.} \end{cases}$$

According to the general definition we have $[\![\mathrm{GInd}\,\vec{a}h\vec{k}M_{\mathrm{Prog}}]\!]^- = \tilde{r}\vec{k}$.

3.5 Term extraction for M

Since u has another occurrence outside M_{Prog}, a further contraction is necessary. We obtain

$$\begin{aligned} &[\![u01v_0\big(\mathrm{GInd}\,\vec{a}h01M_{\mathrm{Prog}}v_0\big)]\!]^- = \\ &[\textbf{if } 0 < h\langle 0,1\rangle \to h\langle 0,1\rangle \mid a_1, a_2 \to F \textbf{ then } \tilde{r}\langle 0,1\rangle \textbf{ else } \langle 0,1\rangle] = \\ &[\textbf{if } 0 < h\langle 0,1\rangle \wedge h\langle 0,1\rangle \mid a_1, a_2 \textbf{ then } \langle 0,1\rangle \textbf{ else } \tilde{r}\langle 0,1\rangle] \end{aligned}$$

and finally

$$[\![M]\!]^+ = \lambda_{\vec{a}}.[\textbf{if } 0 < h\langle 0,1\rangle \wedge h\langle 0,1\rangle \mid a_1, a_2 \textbf{ then } \langle 0,1\rangle \textbf{ else } \tilde{r}\langle 0,1\rangle].$$

To understand how $[\![M]\!]^+$ operates, recall the abbreviations $\vec{l_1}$, $\vec{l_2}$ (which use the Step function) and $h\vec{k} := |k_1a_1 - k_2a_2|$. After an initial check whether 0, 1 are already the desired coefficients, $[\![M]\!]^+$ calls $\tilde{r}$ with $\langle 0,1\rangle$. $\tilde{r}\vec{k}$ checks whether $r(\vec{k})$ (which is one of $\vec{l_1}$ and $\vec{l_2}$) are the coefficients needed, and if not, steps down via $t\vec{k}$ (which again is one of $\vec{l_1}$ and $\vec{l_2}$) and then recursively calls itself.

This extracted algorithm is rather close to Euclid's. The difference is that $[\![M]\!]^+$ keeps a_1 and a_2 fixed, whereas in Euclid's algorithm a_1 and a_2 are replaced by a_2 and the remainder of dividing a_1 by a_2. The gcd is not affected by this change, but the numbers get smaller, which helps for calculations.

Acknowledgements Critical comments of Simon Huber, Diana Ratiu, Trifon Trifonov and an anonymous referee have contributed a lot to the present paper. By incorporating well-founded recursion and induction into an implementation of Gödel's Dialectica interpretation in the proof assistant and program extraction system Minlog[1], Simon Huber and Trifon Trifonov have been able to mechanically extract the algorithm above from a formalized proof of Euclid's theorem.

References

[1] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.
[2] Ulrich Berger, Matthias Eberl, and Helmut Schwichtenberg. Term rewriting for normalization by evaluation. *Information and Computation*, 183:19–42, 2003.
[3] Ulrich Berger and Helmut Schwichtenberg. The greatest common divisor: a case study for program extraction from classical proofs. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs. International Workshop TYPES '95, Torino, Italy, June 1995. Selected Papers*, volume 1158 of *LNCS*, pages 36–46. Springer Verlag, Berlin, Heidelberg, New York, 1996.
[4] Albert Dragalin. New kinds of realizability. In *Abstracts of the 6th International Congress of Logic, Methodology and Philosophy of Sciences*, pages 20–24, Hannover, Germany, 1979.
[5] Harvey Friedman. Classically and intuitionistically provably recursive functions. In D. Scott and G. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978.
[6] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.
[7] Mircea-Dan Hernest. *Feasible programs from (non-constructive) proofs by the light (monotone) Dialectica interpretation*. PhD thesis, Ecole Polytechnique Paris and LMU München, 2006.
[8] Klaus Frovin Jørgensen. Finite type arithmetic. Master's thesis, University of Roskilde, 2001.
[9] Helmut Schwichtenberg and Stanley S. Wainer. Ordinal bounds for programs. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 387–406. Birkhäuser, Boston, 1995.
[10] Anne S. Troelstra, editor. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1973.

[1] See `http://www.minlog-system.de`