Light Dialectica program extraction from a classical Fibonacci proof

– using an optimization of Gödel's technique towards the extraction of more efficient programs from classical proofs –

Mircea Dan HERNEST

Project LogiCal – Paris, FRANCE and GKLI – Munich, GERMANY

DCM'06 Talk in Venice, 16 July 2006

Outline

1. The program–extraction Problem
   - Specifying the wanted behaviour of programs
   - A weakly extensional Arithmetic for Göedel functionals

2. The Light Functional “Dialectica” Interpretation
   - From Gödel’s Dialectica to the Light Dialectica
   - The Contraction Problem

3. Conclusions and Further Work
   - Applicability of Light Dialectica ??? Open research ...

Outline

1. The program–extraction Problem
    - Specifying the wanted behaviour of programs
    - A weakly extensional Arithmetic for Göedel functionals
2. The Light Functional “Dialectica” Interpretation
    - From Gödel’s Dialectica to the Light Dialectica
    - The Contraction Problem
3. Conclusions and Further Work
    - Applicability of Light Dialectica ??? Open research ...

Outline

1. The program–extraction Problem
    - Specifying the wanted behaviour of programs
    - A weakly extensional Arithmetic for Göedel functionals
2. The Light Functional “Dialectica” Interpretation
    - From Gödel’s Dialectica to the Light Dialectica
    - The Contraction Problem
3. Conclusions and Further Work
    - Applicability of Light Dialectica ??? Open research ...

Why $\forall x \exists y\, G(x, y)$ specifications ? [$G \equiv$ Goal Formula]

- Specifications describe wanted behavior for our Program.
- Programs have inputs – x and outputs – y .
- Therefore specifications are formulas $\forall x\, \exists y\, G(x, y)$ where
- $G(x, y)$ is a formula describing the desired relationship between the given input x and the desired output y .
- Exists proof $\mathcal{P}$ of $\forall x\, \exists y\, G(x, y)$ in some logical system $\mathcal{S}$.
- We want to be able to *uniformly* produce by an Algorithm a program t which *realizes* the given specification, i.e., $\forall x\, G(x, t(x))$ is provable in some (other) logical system $\mathcal{S}'$.
- Such Algorithms taking inputs $\mathcal{P}$ are *Program Extraction procedures* and the term t is called *extracted program*.

Classical versus Constructive proofs

Recall that $\mathcal{P}$ is a proof of $\forall x\, \exists y\, G(x,y)$ in the logical system $\mathcal{S}$!

- *G* can be arbitrary only when $\mathcal{S}$ is constructive, otherwise ...
- *Constructive* means *intuitionistic* plus *Markov's Principle* :

$$\neg\neg\exists z\, G_0(z) \rightarrow \exists z\, G_0(z) \quad \underline{\text{or}} \quad \underbrace{\exists^{cl} z}_{\neg\forall z\neg}\, G_0(z) \rightarrow \exists z\, G_0(z)$$

- Only extraction techniques based on Gödel's **Dialectica** interpretation allow Markov's Principle (as axiom) in $\mathcal{S}$.
- *Refined* ***A****–translation* first brings the *Minimal Logic* proof [*minimal* $\equiv$ intuitionistic $\setminus \perp \rightarrow F$] of $\underline{\forall x\, \exists^{cl} y\, G(x,y)}$ to a corresponding intuitionistic proof of $\forall x\, \exists y\, G(x,y)$
- Kreisel's **Modified Realizability** applies to the latter .
- *G* can be a *goal formula* – i.e., more than quantifier-free .

The term system – a lambda-variant of Göedel's **T**

0) All finite types generated from ι and o by the rule $\sigma, \tau \mapsto (\sigma\tau)$

1) $\mathtt{tt}^{o}$, $\mathtt{ff}^{o}$, the *selector* $\mathbf{If}_\tau^{o\,\tau\,\tau\,\tau}$ (usual *if-then-else*), equality $=^{\iota\,\iota\,o}$

2) 0^ι (zero), $\mathtt{S}^\iota$ (successor) and Gödel's recursor $\mathbf{R}_\tau^{\tau\,(\iota\,\tau\,\tau)\,\iota\,\tau}$

3) $\mathtt{And}^{ooo} \;:\equiv\; \lambda p, q.\,\mathbf{If}_o\; p\; q\; \mathtt{ff}$ $\qquad \mathtt{Imp}^{ooo} \;:\equiv\; \lambda p, q.\,\mathbf{If}_o\; p\; q\; \mathtt{tt}$

4) the *n-selector* $\mathbf{If}_\tau^n$ of type $\overbrace{o\ldots o}^{n}\;\overbrace{\tau\ldots\tau}^{n}\;\tau\;\tau$, s.t. $\mathbf{If}_\tau^1 :\equiv \mathbf{If}_\tau$ and

$$\mathbf{If}_\tau^n \;:\equiv\; \lambda p_1, \ldots, p_n, x_{n+1}, x_n, \ldots, x_1.\,\mathbf{If}_\tau\; p_1\; (\mathbf{If}_\tau^{n-1}\; p_2\; \ldots\; p_n\; x_{n+1}\; x_n\; \ldots\; x_2)\; x_1$$

$\mathbf{If}_\tau^n(r_1, \ldots, r_n, t_{n+1}, t_n, \ldots, t_1)$ 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

5) $s =_{\sigma_1\ldots\sigma_n\to\sigma} t :\equiv \forall x_1^{\sigma_1} \ldots x_n^{\sigma_n}\,(s\,x_1 \ldots x_n =_\sigma t\,x_1 \ldots x_n)$, $\sigma \in \{o, \iota\}$

$s =_o t :\equiv \mathtt{at}(s) \leftrightarrow \mathtt{at}(t)$, $s =_\iota t :\equiv \mathtt{at}(= s\, t)$ — *extensionally* defined equality – $\mathtt{at}$ is the *unique* predicate symbol of $\mathtt{WE{-}Z}$

The logical axioms and rules of system $\mathtt{WE\text{–}Z}$ (1/2)

0) $\mathtt{Ax}\exists^+ :\ \forall z_1\,[\,A(z_1) \to \exists z_2\,A(z_2)\,]$

$\mathtt{Ax}\bar{\exists}^+ :\ \bar{\forall} z_1\,[\,A(z_1) \to \bar{\exists} z_2\,A(z_2)\,]$

$$\boxed{\begin{array}{l}\mathtt{AxEFQ}:\ \bot \to A\\ \text{(Ex-Falso-Quodlibet)}\end{array}}$$

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

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

1) Deduction from (arbitrary, undischarged) assumption: $A \vdash A$

2) $$\frac{A \wedge B}{A}\ \wedge^-_{\mathbf{l}}\ ,\quad \frac{A \wedge B}{B}\ \wedge^-_{\mathbf{r}}\ ,\quad \frac{A\,,\ B}{A \wedge B}\ \wedge^+,\quad \frac{A\,,\ A \to B}{B}\ \to^-$$

3) $$\Big[\frac{\bar{\forall} z\,A(z)}{A(t)}\ \bar{\forall}^-_{z,t}\Big]\ \frac{\forall z\,A(z)}{A(t)}\ \forall^-_{z,t}\ \boxed{\mathtt{VC}_2(z,t)},\quad \frac{A(z)}{\forall z\,A(z)}\ \forall^+_z\ \boxed{\mathtt{VC}_1(z)}$$

$\mathtt{VC}_1(z)$: z does not occur free in any undischarged assumption

$\mathtt{VC}_2(z,t)$: no free variable of t gets quantified in A after substit.

The logical axioms and rules of system $\mathtt{WE{-}Z}$ (2/2)

4) $\dfrac{[A] \ldots / B}{A \to B} \to^{+}$, particular set of instances of A discharged;

if at least two A get discharged (contraction) then $\boxed{\mathtt{ncm{-}FC}(A)}$ restriction applies: "if A contains (at least) a positive universal or a negative existential (regular) quantifier then A *must not* contain any $\mathtt{ncm}$ quantifier ($\overline{\exists}$, $\overline{\forall}$)" – case when we say that A is *computationally relevant* (otherwise A is *comput. irrelevant*)

5) $\dfrac{\mathcal{P}\colon A(z)}{\overline{\forall} z\, A(z)} \overline{\forall}_z^{+}$ $\boxed{\mathtt{VC}_1(z)}$ and $\mathtt{VC}_3(z, \mathcal{P}) :\equiv z$ is not free in any of the t involved by $\forall^{-}_{\bullet,t}$ in the proof $\mathcal{P}$ (Berger) and z is also not free in the computationally relevant *contraction formulas* of $\mathcal{P}$

Extensionality/Compatibility and Induction rules

$E_{\sigma,\tau}$: $\forall z^{\sigma\tau}, x^\sigma, y^\sigma.\ x =_\sigma y \to zx =_\tau zy$ – must be forbidden

$$\dfrac{\begin{array}{c} A_0 \\ \vdots \\ s =_\sigma t \end{array}}{B(s) \to B(t)}$$

$\mathtt{COMPAT}_\sigma$ – with the restriction that all undischarged assumptions used in the proof of $s =_\sigma t$ (here denoted A_0) are quantifier-free

$$\dfrac{\begin{array}{c} \emptyset \\ \vdots \\ A(0) \end{array} \qquad \begin{array}{c} \emptyset \\ \vdots \\ \forall z\,(A(z) \to A(\mathtt{S}z)) \end{array}}{\forall z\, A(z)}$$

$\mathtt{IR}_0$ – equivalent to $\mathtt{IA}$, $\mathtt{IR}$ in $\mathtt{WE{-}Z}^-$

$$A(\mathtt{tt}) \wedge A(\mathtt{ff}) \ \to\ \forall p^o A(p)$$

(Boolean Induction Axiom)

$$\boxed{\left.\begin{array}{l} \mathbf{R}_\tau\, x\, y\, 0 =_\tau x \\ \mathbf{R}_\tau\, x\, y\, (\mathtt{S}z) =_\tau y(z, \mathbf{R}_\tau\, x\, y\, z) \end{array}\right\} : \mathtt{Ax}\mathbf{R}_\tau}$$

Weakly extensional Arithmetics $\mathtt{WE{-}Z}$, $\mathtt{WE{-}Z}^-$, $\mathtt{WE{-}Z}^+$

System $\mathtt{WE{-}Z}^-$ obtained from $\mathtt{WE{-}Z}$ by ignoring "ncm" quantifiers

0) Prime formulas decidable: $\vdash_{-} \mathtt{at}(t) \vee \neg\mathtt{at}(t)$ by definition of $\vee$

1) Exists unique bijective association of boolean terms to **qfr** formulas $A_0 \mapsto \mathtt{t}_{A_0}$ such that $\vdash_{-} A_0 \leftrightarrow \mathtt{at}(\mathtt{t}_{A_0})$

2) *Case Distinction* over **qfr** formulas:

$$\vdash_{-} (A_0 \to A) \wedge (\neg A_0 \to A) \;\to\; A$$

3) *Disjunction Elimination* $\vdash_{-} \wedge_{i=1}^{n} (A_i \to B) \;\to\; (\vee_{i=1}^{n} A_i \;\to\; B)$

System $\mathtt{WE{-}Z}^+$ obtained by adding to $\mathtt{WE{-}Z}$ the following:

$\mathtt{AxMK}$: $\exists^{cl} z\, A_0(z) \to \exists z\, A_0(z)$

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

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

Göedel's functional "Dialectica" interpretation

0) A translation of proofs which includes a translation of formulas.

1) $A(\underline{a}) \mapsto A^{\mathbf{D}} \equiv \exists \underline{x}\, \forall \underline{y}\, A_{\mathbf{D}}(\underline{x}; \underline{y}; \underline{a})$ with $\underline{a}$ all free vars of formula A

2) $A_{\mathbf{D}}$ is **qfr** for Göedel's Dialectica, not necess for Light Dialectica

3) Recursive syntactic translation from proofs in Constructive Arithmetic (or Classical Arithmetic, modulo the double-negation translation) to proofs in Intuitionistic Arithmetic such that positive occurrences of $\exists$ and negative occurrences of $\forall$ in the proof's conclusion formula get actually realized by terms in Gödel's **T**.

4) *Contraction Problem:* choose between a number of realizers according to a boolean term associated to the contraction formula; *Diller-Nahm*: postpone all choices to the very end by collecting all candidates and making a single final global choice; *Monotone* (or *Bounded*) *Dialectica*: use a Howard (or Bezem) majorant of the candidates ⇒ extract *majorants* for the realizers.

The Light Dialectica interpretation of formulas

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

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.

Exact realizer synthesis by the **LD**-interpretation

Extraction and Soundness Theorem: There exists an algorithm which, given at input a proof $\mathcal{P} : \{C^i\}_{i=1}^n \vdash_+ A$ will produce at output

1) the tuples of terms T and $\{T_i\}_{i=1}^n$
2) the tuples of variables $\{x_i\}_{i=1}^n$ and y
3) the verifying proof

$$\mathcal{P}_{\mathbf{D}} : \{C^i_{\mathbf{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \vdash_- A_{\mathbf{D}}(T(\underline{x}); y)$$

– where $\underline{x} :\equiv x_1, \ldots, x_n$. Moreover,

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

[$\Rightarrow \underline{x}, y$ not occur free in the *extracted* terms $\{T_i\}_{i=1}^n$ and T]

Implication Introduction with Contraction

$$\boxed{\cfrac{[A]\ \dots\ /B}{A \to B}\ \to^{+}} \qquad n \geq 1, \quad \underline{z} \equiv \overbrace{z, \dots, z}^{n+1} \text{ and } \underline{x} \equiv x_{n+2}, \dots, x_m :$$

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

1) Same tuple z produced by $n + 1 \leq m$ discharged instances of A

2) $\mathtt{ncm\text{-}FC}(A) \Longrightarrow$ tuples $\{T_i\}_{i=1}^{n+1}$ are all of length 0 or $A_{\mathbf{D}}$ is **qfr**

3) If $\{T_i\}_{i=1}^{n+1}$ non-null $\Longrightarrow$ their *equalization* is a must:

$$\mathbf{S} :\equiv \lambda \underline{x}, z, y.\, \mathbf{If}^{n}_{\tau}(\mathtt{t}^{\mathbf{D}}_{A}[z; T^1], \dots, \mathtt{t}^{\mathbf{D}}_{A}[z; T^n], T_{n+1}(\underline{z}, \underline{x}, y), T^n, \dots, T^1)$$

we can now cancell all $\{A_{\mathbf{D}}\}_{i=1}^{n+1}$ *by a single* $\to^{+}$ *in the verifying proof*

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

$$\{C^i_{\mathbf{D}}(x_i; S_i(\underline{x}, z, y))\}_{i=n+2}^{m} \vdash_{-} A_{\mathbf{D}}(z; \mathbf{S}(\underline{x}, z, y)) \to B_{\mathbf{D}}(S(\underline{x}, z); y)$$

Computationally-redundant Contraction - an example

Warning: A and B are here the same as on the previous slide !!!

$$\boxed{\dfrac{A\,,\ A \to B}{B}\ \to^{-}} \qquad \{C^i_{\mathbf{D}}(x'_i;\, S_i(\underline{x'}, y'))\}_{i=1}^{n+1} \vdash_{-} A_{\mathbf{D}}(S'(\underline{x'});\, y')$$

$$\{C^i_{\mathbf{D}}(x''_i;\, S_i(\underline{x''}, z, y))\}_{i=n+2}^{m} \vdash_{-} A_{\mathbf{D}}(z;\, \mathbf{S}(\underline{x''}, z, y)) \to B_{\mathbf{D}}(S(\underline{x''}, z);\, y)$$

$$T_i \ :\equiv \ \begin{cases} \lambda \underline{x}, y.\, S_i(\underline{x'}, \mathbf{S}(\underline{x''}, S'(\underline{x'}), y)) \ , & \text{if } 1 \leq i \leq n+1 \\ \lambda \underline{x}, y.\, S_i(\underline{x''}, S'(\underline{x'}), y) \ , & \text{if } n+1 < i \leq m \end{cases}$$

$$T \ :\equiv \ \lambda \underline{x}.\, S(\underline{x''}, S'(\underline{x'}))$$

Extraction and Soundness Theorem:

$$\{C^i_{\mathbf{D}}(x_i;\, T_i(\underline{x}, y))\}_{i=1}^{m} \vdash_{-} B_{\mathbf{D}}(T(\underline{x});\, y)$$

Contraction involved by usual Induction Rule (worst)

We are given $\mathcal{P}_b : \ C_b \vdash A(0)$ and $\mathcal{P}_s : \ C_s \vdash \forall z(A(z) \to A(\mathrm{S}z))$

$$\dfrac{\dfrac{C_s \qquad \dfrac{C_b \quad C_b \to (C_s \to A(z))}{C_s \to A(z)}\to^-}{A(z)}\to^- \qquad \dfrac{\left.\begin{matrix} C_s \\ \vdots \\ \forall z(A(z) \to A(\mathrm{S}z)) \end{matrix}\right\} \mathcal{P}_s}{A(z) \to A(\mathrm{S}z)}\forall^-}{A(\mathrm{S}z)}\to$$

first cancell C_s (with contraction) and subsequently C_b (without contraction) to get $C_b \to (C_s \to A(z)) \vdash C_b \to (C_s \to A(\mathrm{S}z)) \quad \Longrightarrow$

$\emptyset \vdash \forall z[(C_b \to (C_s \to A(z))) \to C_b \to (C_s \to A(\mathrm{S}z))] \quad \Longrightarrow$

by $\emptyset$-premised Induction Rule that $\emptyset \vdash \forall z(C_b \to (C_s \to A(z)))$

from which we recover the usual conclusion $C_b, C_c \vdash \forall z A(z)$

Clear-cut practical example - classical Fibonacci

MINLOG program extracted by Light Dialectica (after normalization)

$$[n_0]\pi_1(\mathbf{R}_{\mathbb{N}\to(\mathbb{N}@\mathbb{N})}(0@1)\{[n_1, p^{\mathbb{N}@\mathbb{N}}]\pi_2(p)@(\pi_1(p)+\pi_2(p))\}n_0)$$

Exactly the usual algorithm computing the *n*-th Fibonacci number.

$$[G, n_1]\pi_1\pi_2(\mathbf{R}_{\mathbb{N}=>\mathbb{N}@(\mathbb{N}@\mathbb{N})@(\mathbb{N}@\mathbb{N})}((0@0@0)@0@1)\{[n_2, p][\mathbf{If}[\mathbf{If}(G\pi_1\pi_1(p)\pi_1\pi_2\pi_1(p))$$
$$[\mathbf{If}(G(\mathtt{S}\pi_1\pi_1(p))\pi_2\pi_2\pi_1(p))(G(\mathtt{S}(\mathtt{S}\pi_1\pi_1(p)))(\pi_1\pi_2\pi_1(p)+\pi_2\pi_2\pi_1(p)))\mathtt{tt}]\mathtt{tt}]$$
$$(n_2@\pi_2(p))(\pi_1(p))]@\pi_2\pi_2(p)@\pi_1\pi_2(p)+\pi_2\pi_2(p)\}n_1)$$

This by Gödel's Dialectica

If-tests are only due to the Contraction formula, integrated in extracted program

The BBS Refined A-translation yields a more complex program:

$$[n]\ \mathbf{R}_{\mathbb{N}\to(\mathbb{N}\to\mathbb{N}\to\mathbb{N})\to\mathbb{N}}\quad([f^{\mathbb{N}\to\mathbb{N}\to\mathbb{N}}]\,f\,0\,1)$$
$$([m, \mathbf{H}, g^{\mathbb{N}\to\mathbb{N}\to\mathbb{N}}]\ \mathbf{H}\ ([p, q]\ \mathbf{H}\ ([k, l]\ g\ l\ (k+l))))\quad n\quad([k, l]k)$$

– uses the type-2 functional $\mathbf{H}^{(\mathbb{N}\to\mathbb{N}\to\mathbb{N})\to\mathbb{N}}$, within the recursion

Attempts to apply Light Dialectica - a negative result

Dickson-2-2 Lemma: “For any two functions $f, g : \mathbb{N} \mapsto \mathbb{N}$ defined within the set of natural numbers $\mathbb{N}$ there exist indexes $i < j$ such that both $f(i) \leq f(j)$ and $g(i) \leq g(j)$.”

Classical proof within Minimal Arithmetic uses 3 times the Minimum Principle (relative to unary predicate variable $Q^{\mathbb{N}}$):

$$\forall h^{\mathbb{N}\to\mathbb{N}}.\ \exists^{cl} z\, Q(z) \to \exists^{cl} x.\ (\forall y.\ h(y) < h(x) \to \neg Q(y)) \wedge Q(x)$$

Not possible to use an “ncm” version of $\exists^{cl} x$ in none of the 3 instances of Minimum Principle ... but x is positive universal quantified in a Contraction formula (thus *computationally relevant*) ... ⇒ ... impossible to avoid any of the 3 contractions.

These contractions are involved by **Induction** and the realizing program **must** contain 3 times Gödel’s recursor $\mathbf{R}_\tau$ (as iterator).

Short List of related Papers I

M.-D. Hernest and U. Kohlenbach.
A complexity analysis of functional interpretations.
Theoretical Computer Science, 338(1-3):200–246, 2005.

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.

U. Kohlenbach and P. Oliva.
Proof mining: a systematic way of analysing proofs in Mathematics.
Proc. of the Steklov Inst. of Mathem., 242:136–164, 2003.

Short List of related Papers II

- M.-D. Hernest, "Light Functional Interpretation",
  LNCS **3634** (2005), pp. 477 – 492, CSL 2005.
- U. Berger, W. Buchholz, and H. Schwichtenberg.
  Refined program extraction from classical proofs.
  Annals of Pure and Applied Logic, 114:3–25, 2002.
- C. Raffalli.
  Getting results from programs extracted from classical proofs.
  Theoretical Computer Science, 323(1-3):49–70, 2004.
- C. Paulin-Mohring and B. Werner.
  Synthesis of ML programs in the system Coq.
  Journal of Symbolic Computation, 15(5/6):607–640, 1993.