

## Light Functional Interpretation

– 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

LAMA-Chambery Workshop, 13-14 April 2005

# Outline

- 1 **The program–extraction Problem**
  - Specifying the wanted behaviour of programs
  - A weakly extensional Arithmetic for Gödel 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 ...
  - Extensions of Light Dialectica to monotone/classical systems

# Outline

- 1 The program–extraction Problem
  - Specifying the wanted behaviour of programs
  - A weakly extensional Arithmetic for Gödel 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 ...
  - Extensions of Light Dialectica to monotone/classical systems

# Outline

- 1 The program–extraction Problem
  - Specifying the wanted behaviour of programs
  - A weakly extensional Arithmetic for Gödel 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 ...
  - Extensions of Light Dialectica to monotone/classical systems

# 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*:  
 $\neg\neg\exists z G_0(z) \rightarrow \exists z G_0(z)$  or  $\underbrace{\exists^{cl} z G_0(z)}_{\neg\neg\forall z \neg}$   $\rightarrow \exists z G_0(z)$
- 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  $\forall x \exists^{cl} y G(x, y)$  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 Gödel's T

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

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

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

3)  $\text{And}^{\circ\circ\circ} := \lambda p, q. \text{If}_\circ p q \text{ff}$        $\text{Imp}^{\circ\circ\circ} := \lambda p, q. \text{If}_\circ p q \text{tt}$

4) the  $n$ -selector  $\text{If}_\tau^n$  of type  $\overbrace{\circ \dots \circ}^n \overbrace{\tau \dots \tau}^n \tau \tau$ , s.t.  $\text{If}_\tau^1 := \text{If}_\tau$  and

$$\text{If}_\tau^n := \lambda p_1, \dots, p_n, x_{n+1}, x_n, \dots, x_1. \text{If}_\tau p_1 (\text{If}_\tau^{n-1} p_2 \dots p_n x_{n+1} x_n \dots x_2) x_1$$

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

5)  $s =_{\sigma_1 \dots \sigma_n \rightarrow \sigma} t := \forall x_1^{\sigma_1} \dots x_n^{\sigma_n} (s x_1 \dots x_n =_\sigma t x_1 \dots x_n)$ ,  $\sigma \in \{\circ, \iota\}$

$s =_\circ t := \text{at}(s) \leftrightarrow \text{at}(t)$ ,  $s =_\iota t := \text{at}(= s t)$  — extensionally

defined equality –  $\text{at}$  is the *unique* predicate symbol of WE-Z

## The logical axioms and rules of system WE-Z (1/2)

$$\begin{aligned}
 0) \quad & \text{Ax}\exists^+ : \forall z_1 [A(z_1) \rightarrow \exists z_2 A(z_2)] & \text{Ax}\text{EFQ} : \perp \rightarrow A \\
 & \text{Ax}\bar{\exists}^+ : \bar{\forall} z_1 [A(z_1) \rightarrow \bar{\exists} z_2 A(z_2)] & \text{(Ex-Falso-Quodlibet)} \\
 & \text{Ax}\exists^- : \exists z_1 A(z_1) \wedge \forall z_2 [A(z_2) \rightarrow B] \rightarrow B \\
 & \text{Ax}\bar{\exists}^- : \bar{\exists} z_1 A(z_1) \wedge \bar{\forall} z_2 [A(z_2) \rightarrow B] \rightarrow B
 \end{aligned}$$

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

$$2) \quad \frac{A \wedge B}{A} \wedge_l^-, \frac{A \wedge B}{B} \wedge_r^-, \frac{A, B}{A \wedge B} \wedge^+, \frac{A, A \rightarrow B}{B} \rightarrow^-$$

$$3) \quad \left[ \frac{\bar{\forall} z A(z)}{A(t)} \bar{\forall}_{z,t}^-, \frac{\forall z A(z)}{A(t)} \forall_{z,t}^-, \boxed{\text{VC}_2(z, t)}, \frac{A(z)}{\forall z A(z)} \forall_z^+ \boxed{\text{VC}_1(z)} \right]$$

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

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



## The logical axioms and rules of system WE-Z (2/2)

$$4) \frac{[A] \dots / B}{A \rightarrow B} \rightarrow^+, \text{ particular set of instances of } A \text{ discharged};$$

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

$$5) \frac{\mathcal{P}: A(z)}{\bar{\forall}z A(z)} \bar{\forall}z^+ \boxed{\text{VC}_1(z)} \text{ and } \text{VC}_3(z, \mathcal{P}) := z \text{ is not free in any of}$$

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

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

$A_0$

$\vdots$

$s =_\sigma t$

COMPAT $_\sigma$  – with the restriction that  
all undischarged assumptions used  
in the proof of  $s =_\sigma t$  (here denoted  $A_0$ )

$B(s) \rightarrow B(t)$

are quantifier-free

$\emptyset$

$\emptyset$

IR $_0$  – equivalent to IA, IR in WE-Z $^-$

$\vdots$

$\vdots$

$A(\text{tt}) \wedge A(\text{ff}) \rightarrow \forall p^\circ A(p)$

$A(0) \quad \forall z (A(z) \rightarrow A(Sz))$

(Boolean Induction Axiom)

$\forall z A(z)$

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

Weakly extensional Arithmetics  $WE-Z$ ,  $WE-Z^-$ ,  $WE-Z^+$ 

System  $WE-Z^-$  obtained from  $WE-Z$  by ignoring "ncm" quantif.

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

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

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

$$\vdash_- (A_0 \rightarrow A) \wedge (\neg A_0 \rightarrow A) \rightarrow A$$

3) *Disjunction Elimination*  $\vdash_- \bigwedge_{i=1}^n (A_i \rightarrow B) \rightarrow (\bigvee_{i=1}^n A_i \rightarrow B)$

System  $WE-Z^+$  obtained by adding to  $WE-Z$  the following:

$$AxMK : \quad \exists^{cl} z A_0(z) \rightarrow \exists z A_0(z)$$

$$AxIP_{\forall} : \quad [\forall x A_0(x) \rightarrow \exists y B(y)] \rightarrow \exists y [\forall x A_0(x) \rightarrow B(y)]$$

$$AxAC : \quad \forall x \exists y B(x, y) \rightarrow \exists Y \forall x B(x, Y(x))$$

## Gödel’s functional “Dialectica” interpretation

- 0) A translation of proofs which includes a translation of formulas.
- 1)  $A(\underline{a}) \mapsto A^D \equiv \exists \underline{x} \forall \underline{y} A_D(\underline{x}; \underline{y}; \underline{a})$  with  $\underline{a}$  all free vars of formula  $A$
- 2)  $A_D$  is **qfr** for Gödel’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 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 simple upper bound (majorant) of the candidates  $\Rightarrow$  extract *bounds* for the realizers.

# The Light Dialectica interpretation of formulas

$$A^D \equiv (A_D : \equiv A) \text{ for prime formulas } A$$

$$(A \wedge B)^D \equiv \exists \underline{x}, \underline{u} \forall \underline{y}, \underline{v} [(A \wedge B)_D : \equiv A_D(\underline{x}; \underline{y}; \underline{a}) \wedge B_D(\underline{u}; \underline{v}; \underline{b})]$$

$$(A \rightarrow B)^D \equiv \exists \underline{Y}, \underline{U} \forall \underline{x}, \underline{v} [(A \rightarrow B)_D : \equiv A_D(\underline{x}; \underline{Y}(\underline{x}, \underline{v})) \rightarrow B_D(\underline{U}(\underline{x}); \underline{v})]$$

$$(\exists z A(z, \underline{a}))^D \equiv \exists z^\dagger, \underline{x} \forall \underline{y} [(\exists z A(z, \underline{a}))_D(z^\dagger, \underline{x}; \underline{y}; \underline{a}) : \equiv A_D(\underline{x}; \underline{y}; z^\dagger, \underline{a})]$$

$$(\exists z A(z, \underline{a}))^D \equiv \exists \underline{x} \forall \underline{y} [(\exists z A(z, \underline{a}))_D(\underline{x}; \underline{y}; \underline{a}) : \equiv \exists z A_D(\underline{x}; \underline{y}; z, \underline{a})]$$

$$(\forall z A(z, \underline{a}))^D \equiv \exists \underline{X} \forall z^\dagger, \underline{y} [(\forall z A(z, \underline{a}))_D(\underline{X}; z^\dagger, \underline{y}; \underline{a}) : \equiv A_D(\underline{X}(z^\dagger); \underline{y}; z^\dagger, \underline{a})]$$

$$(\forall z A(z, \underline{a}))^D \equiv \exists \underline{x} \forall \underline{y} [(\forall z A(z, \underline{a}))_D(\underline{x}; \underline{y}; \underline{a}) : \equiv \forall z A_D(\underline{x}; \underline{y}; z, \underline{a})]$$

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

**Theorem:** 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  $\{\underline{x}_i\}_{i=1}^n$  and  $y$ , all together with the verifying proof

$$\mathcal{P}_D : \{C_D^i(\underline{x}_i; T_i(\underline{x}, y))\}_{i=1}^n \vdash_- A_D(T(\underline{x}); y)$$

– where  $\underline{x} \equiv x_1, \dots, x_n$ . 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

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

$$\{A_D(z; T_i(\underline{z}, \underline{x}, y))\}_{i=1}^{n+1}, \{C_D^i(x_i; T_i(\underline{z}, \underline{x}, y))\}_{i=n+2}^m \vdash_- B_D(T(\underline{z}, \underline{x}); y)$$

- 1) Same tuple  $z$  produced by  $n + 1 \leq m$  discharged instances of  $A$
- 2)  $\text{ncm-FC}(A) \implies$  tuples  $\{T_i\}_{i=1}^{n+1}$  are all of length 0 **or**  $A_D$  is **qfr**
- 3) If  $\{T_i\}_{i=1}^{n+1}$  non-null  $\implies$  their *equalization* is a must:

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

we can now cancell all  $\{A_D\}_{i=1}^{n+1}$  by a single  $\rightarrow^+$  in the verifying proof

$$\{A_D(z; \mathbf{S}(\underline{x}, z, y))\}_{i=1}^{n+1}, \{C_D^i(x_i; S_i(\underline{x}, z, y))\}_{i=n+2}^m \vdash_- B_D(\mathbf{S}(\underline{x}, z); y)$$

$$\{C_D^i(x_i; S_i(\underline{x}, z, y))\}_{i=n+2}^m \vdash_- A_D(z; \mathbf{S}(\underline{x}, z, y)) \rightarrow B_D(\mathbf{S}(\underline{x}, z); y)$$

# Computationally-redundant Contraction - an example

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

$$\boxed{\frac{A, A \rightarrow B}{B} \rightarrow^-} \quad \{C_D^i(\underline{x}'_i; S_i(\underline{x}', y'))\}_{i=1}^{n+1} \vdash^- A_D(S'(\underline{x}'); y')$$

$$\{C_D^i(\underline{x}''_i; S_i(\underline{x}'', z, y))\}_{i=n+2}^m \vdash^- A_D(z; \mathbf{S}(\underline{x}'', z, y)) \rightarrow B_D(S(\underline{x}'', z); y)$$

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

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

Soundness Theorem:  $\{C_D^i(\underline{x}_i; T_i(\underline{x}, y))\}_{i=1}^m \vdash^- B_D(T(\underline{x}); y)$



## Contraction involved by usual Induction Rule (worst)

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

$$\frac{
 \frac{
 \frac{
 C_s \quad C_b \quad C_b \rightarrow (C_s \rightarrow A(z)) \quad \rightarrow^-
 }{
 C_s \rightarrow A(z)
 }
 \rightarrow^-
 }{
 A(z)
 }
 }{
 A(sz)
 }
 \quad
 \left.
 \begin{array}{c}
 C_s \\
 \vdots \\
 \forall z(A(z) \rightarrow A(sz))
 \end{array}
 \right\}
 \mathcal{P}_s
 }{
 A(z) \rightarrow A(sz)
 }
 \quad \forall^-
 }{
 }
 \rightarrow^-$$

first cancell  $C_s$  (with contraction) and subsequently  $C_b$  (without contraction) to get  $C_b \rightarrow (C_s \rightarrow A(z)) \vdash C_b \rightarrow (C_s \rightarrow A(sz)) \implies \emptyset \vdash \forall z[(C_b \rightarrow (C_s \rightarrow A(z))) \rightarrow C_b \rightarrow (C_s \rightarrow A(sz))]$ ,  $\implies$  by  $\emptyset$ -premissed Induction Rule,  $\emptyset \vdash \forall z(C_b \rightarrow (C_s \rightarrow A(z)))$  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)

$$[n_0]\pi_1(\mathbf{R}_{\iota \rightarrow (\iota @ \iota)}(0@1)\{[n_1, p^{\circ \iota}]\pi_2(p)@(\pi_1(p) + \pi_2(p))\}n_0)$$

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

$$[G, n_1]\pi_1\pi_2(\mathbf{R}_{\iota \Rightarrow \iota @ (\iota @ \iota) @ (\iota @ \iota)}((0@0@0)@0@1)\{[n_2, p][\mathbf{If}[\mathbf{If}(G\pi_1\pi_1(p)\pi_1\pi_2\pi_1(p))$$

$$[\mathbf{If}(G(S\pi_1\pi_1(p))\pi_2\pi_2\pi_1(p))(G(S(S\pi_1\pi_1(p)))(\pi_1\pi_2\pi_1(p) + \pi_2\pi_2\pi_1(p)))\mathbf{tt}]\mathbf{tt}]$$

$$(n_2@ \pi_2(p))(\pi_1(p))\}@ \pi_2\pi_2(p)@ \pi_1\pi_2(p) + \pi_2\pi_2(p)\}n_1) \quad \text{Gödel's Dialectica}$$

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

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

$$[n_0](\mathbf{R}_{\iota \rightarrow (\iota \rightarrow \iota \rightarrow \iota) \rightarrow \iota}([f_1]f_1 01)([n_1, H_2, f_3]H_2([n_4, n_5]H_2([n_6, n_7]f_3 n_7(n_6 +$$

$$n_7))))n_0([n_1, n_2]n_1) - \text{uses the higher-type functional } H_2, \text{ within the recursion}$$

Benchmark:  $F_{15} = 610$  in 110 ms by Light Dialectica and 5918 ms

by Refined A-translation,  $F_{17} = 1597$  in 260 ms and respectively  
 30554 ms (Pentium 4, Windows XP)

## 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  $Q^l$ ):

$\forall h^l \rightarrow^l. \exists^{cl} z Q(z) \rightarrow \exists^{cl} x. (\forall y. h(y) < h(x) \rightarrow \neg Q(y)) \wedge Q(x)$

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*) ...  $\Rightarrow$  ... **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).

# The Light Monotone Dialectica interpretation

$$\begin{aligned}x \succeq_l y &::= x \geq_l y ::= \text{at}(\geq x^l y^l) \\s \geq_{\sigma_1 \dots \sigma_n} t &::= \forall x_1^{\sigma_1}, \dots, x_n^{\sigma_n} (s x_1 \dots x_n \geq_l t x_1 \dots x_n) \\x \succeq_{\sigma\tau} y &::= \forall z_1^\sigma, z_2^\sigma (z_1 \succeq_\sigma z_2 \rightarrow x z_1 \succeq_\tau y z_2)\end{aligned}$$

**Conjecture:** There exists an algorithm which from a given proof  $\vdash_+^m A(a)$  produces *closed* uniform bounds  $T$  with a verifying proof  $\vdash_-^m \exists x(T \succeq x \wedge \forall a, y A_D(x(a); y; a))$ .

The optimization brought by the Light **MD**-interpretation concerns *more* the diminishing of the maximal type degree of extracted bounds *than* contraction (which is sufficiently handled by the pure **MD**-interpretation).

## Extensions to extractions from fully classical proofs

Fully classical system  $WE-Z^c$  : $\equiv$   $WE-Z$  plus  $\neg\neg A \rightarrow A$ .

Pre-processing double negation translation  $\cdot \mapsto \cdot^N$  is necessary.

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

**Conjecture:** There exists an algorithm which, given at input a proof  $\vdash_{c+}^m A(a)$ , produces at output *closed* uniform bounds  $T$  and the verifying proof  $\vdash_{-}^m \exists x (T \succeq x \wedge \forall a, y (A^N)_D(x(a); y; a))$ .

# 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

-  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.