Light Dialectica program extraction from a classical Fibonacci proof

Mircea-Dan Hernest[1,2]

Laboratoire d'Informatique (LIX)
École Polytechnique
F-91128 Palaiseau - FRANCE

Abstract

We demonstrate program extraction by the Light Dialectica Interpretation (LDI) on a minimal logic proof of the classical existence of Fibonacci numbers. This semi-classical proof is available in MinLog's library of examples. The term of Gödel's **T** extracted by the LDI is, after strong normalization, exactly the usual recursive algorithm which defines the Fibonacci numbers (in pairs). This outcome of the Light Dialectica meta-algorithm is much better than the **T**-program extracted by means of the pure Gödel Dialectica Interpretation. It is also strictly less complex than the result obtained by means of the refined A-translation technique of Berger, Buchholz and Schwichtenberg on an artificially distorted variant of the input proof, but otherwise it is identical with the term yielded by Berger's Kripke-style refined A-translation. Although syntactically different, it also has the same computational complexity as the original program yielded by the refined A-translation from the undistorted input classical Fibonacci proof.

Keywords: Proof Mining, Program extraction from (classical) proofs, Complexity of extracted programs, Refined A-translations, Quantifiers without computational meaning, Light Dialectica Interpretation, Computationally redundant contractions, Gödel's functional "Dialectica" interpretation

1 Introduction

There has been quite some work in the last years in the field of program extraction from *classical* proofs. Although strong mathematical results have recently been obtained in the Proof Mining of classical analytical proofs (see, e.g., [17,15,18,20,22]), the computer-implemented program extraction meta-algorithms were able to produce only limited results, for rather small test-cases and even then, the extracted program is not the optimal one.

Such a situation one partly encounters in the extraction of a rather unusual, distorted algorithm for the computation of Fibonacci numbers by means of the Berger-Buchholz-Schwichtenberg (BBS) refined A-translation of [3]. The term $\mathtt{t}_{\mathtt{BBS}}$

[1] *Project LogiCal* - Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École Polytechnique, INRIA et Université Paris-Sud - FRANCE

[2] Email: danher@lix.polytechnique.fr

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

of Gödel's **T** extracted via this BBS refined A-translation from an artificially distorted [3] variant of the MinLog minimal logic proof of the *weak* (classical) existence of the Fibonacci numbers, followed by Kreisel's Modified Realizability [19] and finally strongly normalized [4,5] not only makes necessarily use of a type-2 Gödel recursor (present also in the original extraction from [3]), but also uses two times the corresponding type-2 functional, fact which strictly increases its computational complexity. The program $\mathtt{t}_{\mathtt{BBS}}$ has an unexpected exponential time complexity, see the end of Section 4. On the other hand, the program extracted by the BBS technique from the original classical Fibonacci proof outlined in [3] is nonetheless linear-time in the unary representation of natural numbers, see [3] for full technical details.

The aforementioned type-2 recursor is $\mathtt{R}_{(\iota\to\iota\to\iota)\to\iota}$, where the type level (degree) of $(\iota \to \iota \to \iota) \to \iota$ is 2. Here ι is the base type which denotes the set of natural numbers $\mathbb{N}$ and $\mathtt{R}_\rho$ is the denotation for the so-called "type-ρ Gödel recursor", which actually has the type $\rho \to (\iota \to \rho \to \rho) \to \iota \to \rho$ in Gödel's **T** [4]. This situation is quite unexpected since the usual recursive definition of Fibonacci numbers (in pairs) can be expressed in Gödel's **T** by means of a type-0 Gödel recursor only, namely $\mathtt{R}_{\iota\times\iota}$. Here $\sigma \times \tau$ denotes the pairing of types σ and τ. In fact such a **T**-term was actually extracted in MinLog [25] by pure Modified Realizability, from the usual pure intuitionistic proof of the *strong* (intuitionistic) existence of Fibonacci numbers, see [3] [5].

The point of the endeavour of extracting programs from classical rather than constructive or even purely intuitionistic proofs is that (semi-)classical proofs are much easier and more direct to build, both by human brain and also in the various computer-implemented proof-systems. It is therefore desirable that the algorithms synthesised from classical proofs by means of the more complex program extraction meta-algorithms [6] are at least as good as those yielded by the more common extraction techniques [7] from the corresponding constructive/purely intuitionistic proofs. When applied to the semi-classical MinLog Fibonacci proof (by this we hereon mean the distorted variant obtained by automated proof-search, present in [25,11], and not the manual one, originally introduced in [3]), this is not the case, neither for the BBS refined A-translation, nor for the pure Gödel Dialectica Interpretation, as we show later in the sequel. A repair of this situation can be provided for the BBS refined A-translation by eliminating the distortion from the proof at input [8] or by using its Kripke-style variant due to Berger in [2] [9]. The latter extraction technique

[3] This artificial distortion is due to the automated proof-search mechanism of MinLog, relative to the more manually given input proof which was originally used in the classical Fibonacci extraction reported in [3].

[4] See paper [3] for more such technical details.

[5] On the other hand, this linear - in the unary representation of natural numbers - algorithm is outperformed by other logarithmic algorithms, see [24] for such an example.

[6] Here we think particularly (but not exclusively) at those from the Dialectica family (see [23] for a nice unification work) and the Refined A-translation family.

[7] Basically variants of Kreisel's Modified Realizability [19], which is a simpler but weaker form of Gödel's functional (*Dialectica*) interpretation [1,10].

[8] See [3,24] or the end of Section 4 for a display of the original program that is obtained by BBS from the undistorted classical Fibonacci proof, which is also of linear-time complexity in the unary representation of the natural-number input. See also Footnote 18.

[9] Berger's *Kripke-style* refined A-translation introduced in [2] nicely combines the optimizing (in the sense of the efficiency of programs extracted from classical proofs) features of both the BBS [3] and the Coquand-Hofmann [7] refined A-translations. It also furthermore adds the so-called *uniform* quantifiers, which are used to "label" and thus isolate parts of the input proof which are meant not to have a computational content under such a translation.

actually produces exactly the same program as our Light Dialectica interpretation (originally introduced in [13], but see also [14] for a much larger and more unified exposition).

On the other hand, none of the *monotone* [16] or *bounded* [8] optimizations of Gödel's technique can handle such an exact realizer extraction problem. It is the Light Dialectica interpretation (abbreviated `LDI`) which gives the solution. The term of Gödel's **T** extracted by the `LDI` is, after strong normalization, exactly the usual recursive algorithm which defines the Fibonacci numbers (in pairs).

2 The semi-classical Fibonacci proof in `MinLog`

`MinLog` is an interactive proof- and program-extraction system developed by H. Schwichtenberg and members of the logic group at the University of Munich. It is based on first order Natural Deduction calculus and uses as primitive *minimal* rather than classical or intuitionistic logic. See [11,25] for full details.

Definition 2.1 [Fibonacci Numbers] The inductive definition is as usual

$$\texttt{Base}: \; F_0 :\equiv 0, \; F_1 :\equiv 1 \qquad \texttt{Step}: \; F_{n+1} :\equiv F_n + F_{n-1} \;\text{ for }\; n \geq 1, \, n \in \mathbb{N}$$

The Fibonacci Numbers example was implemented in `MinLog` and it was comparatively analysed in [3] by both pure Modified Realizability (from the usual pure intuitionistic proof) and also by the `BBS` refined `A`-translation (from a minimal logic proof of the weak, classical existence of Fibonacci Numbers; we dub such proofs as "semi-classical") followed by Modified Realizability.

The semi-classical Fibonacci proof in `MinLog` is a Natural Deduction proof of $\forall n \exists^{\mathtt{cl}} k \, G(n,k)$ – where $\exists^{\mathtt{cl}} k \, G(n,k) :\equiv (\forall k. \, G(n,k) \to \bot) \to \bot$ – from assumptions expressing that G is the graph of the Fibonacci function, i.e.,

$$G(0,0) \;\; \texttt{AND} \;\; G(1,1) \;\; \texttt{AND} \;\; \forall n,k,l. \, [G(n,k) \wedge G(n+1,l)] \to G(n+2,k+l) \, .$$

The best source for reading and analysing this proof is the `MinLog` distribution [11] (or [25]), particularly that this differs, due to the use of automated proof-search, from the more manually given proof from Section 6 of [3]. See also Footnote 18 for some hints on how these semi-classical proofs can be constructed in `MinLog`. Notice that in the context of program extraction by the Light Dialectica interpretation, presented in Section 3 below, the assumption on G is rather expressed as

$$G(0,0) \;\; \texttt{AND} \;\; G(1,1) \;\; \texttt{AND} \;\; \overline{\forall} n,k,l. \, [G(n,k) \wedge G(n+1,l)] \to G(n+2,k+l) \; ,$$

where $\overline{\forall}$ is the universal quantifier without computational meaning, see below.

3 The *light* functional *Dialectica* interpretation

The "light" variant of Gödel's functional "Dialectica" Interpretation was introduced in [13] as an optimization for term-extraction of Gödel's original technique [10]

[10] Paper [1] provides a nice survey in English which includes the extensions to full Analysis.

from [10]. The main feature of "Dialectica Light" is the elimination *already at extraction time* of a number of *relevant* (for the Dialectica program extraction) Contractions which are identified as *redundant* and in consequence are isolated by means of an adaptation of Berger's quantifiers without computational content [11] (introduced in [2] as "uniform quantifiers"), here denoted $\overline{\forall}$ and $\overline{\exists}$, like in [13,14].

Dialectica Light (abbreviated LDI) is a recursive syntactic translation from proofs in a semi-classical [12] weakly extensional arithmetical system in all finite types [13] (denoted $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$) to proofs in the corresponding purely intuitionistic system [14] (denoted $\mathtt{WeZ}^{\exists}$) such that the positive occurrences of the strong $\exists$ and the negative occurrences of $\forall$ in the proof's conclusion formula get actually realized by terms in Gödel's **T**. These *realizing terms* are also called the *programs extracted* by the LDI and (if only the extracted programs are wanted) the translation process is also referred to as "program extraction". The LDI translation of proofs includes the following translation of formulas:

Definition 3.1 By quantifier-free (qfr) formula we understand a formula built from prime formulas $\mathtt{at}(t^o)$ and $\bot$ by means of $\wedge$, $\to$ and, if $\exists$ is available, also $\vee$. The qfr formulas are all decidable in our systems. There exists a unique bijective association of boolean terms to quantifier-free formulas $A_0 \mapsto \mathtt{t}_{A_0}$ such that $\vdash A_0 \leftrightarrow \mathtt{at}(\mathtt{t}_{A_0})$. Then the LDI translation of formulas is:

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

where $\cdot \mapsto \cdot^{\dagger}$ is a mapping which assigns to every given variable z a completely new variable $z^{\dagger}$ which has the same type of z. The free variables of $A^{\mathtt{D}}$ are exactly the free variables of A.

Remark 3.2 For the light Dialectica interpretation, the *radical* (or "root") formula $A_{\mathtt{D}}$ (which is LDI associated to A) is not necessarily quantifier-free, like it is for the pure Gödel's functional interpretation. It actually contains the translation of all ncm quantifiers to the corresponding regular quantifiers.

---

[11] In [13] we named these special existential and universal quantifiers "without (or non-) computational meaning", abbreviated ncm. We here continue to use our own terminology.

[12] This can be extended to fully classical proofs, modulo some double-negation translation, see [14].

[13] System $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$ was denoted $\mathtt{WE{-}Z}^{+}$ in [13]. It is nevertheless much better presented, with complete comparative details in [14], just like its corresponding $\mathtt{WeZ}^{\exists}$, see below.

[14] System $\mathtt{WeZ}^{\exists}$, which was denoted $\mathtt{WE{-}Z}^{-}$ in [13], is a Natural Deduction formulation of the weakly extensional Heyting Arithmetic in all finite types $\mathtt{WE{-}HA}^{\omega}$ from Section 1.6.12 of [26]. See also [3,24] for the original corresponding fully extensional variant $\mathtt{Z}^{\exists} \equiv \mathtt{Z} + \exists$.

Theorem 3.3 (Exact realizer synthesis by the Light Dialectica [13])
There exists an algorithm which, given at input a Natural Deduction proof $\mathcal{P} : \{C^i\}_{i=1}^n \vdash A$ [15] in $\mathtt{WeZ}^{\exists,\mathtt{nc}+}$, it eventually produces at output the following:

(i) the tuples of terms $\{T_i\}_{i=1}^n$ and T,

(ii) the tuples of variables $\{x_i\}_{i=1}^n$ and y, all together with

(iii) the verifying proof $\mathcal{P}_{\mathtt{D}} : \{C^i_{\mathtt{D}}(x_i; T_i(\underline{x}, y))\}_{i=1}^n \vdash A_{\mathtt{D}}(T(\underline{x}); y)$ in $\mathtt{WeZ}^{\exists}$, where $\underline{x} :\equiv x_1, \ldots, x_n$.

Moreover,

- the variables $\underline{x}$ and y do not occur in $\mathcal{P}$ (they are all completely new)
- 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 LDI".

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

Remark 3.4 Gödel's functional "Dialectica" interpretation becomes relatively (far) more complicated at the moment when it has to face *contraction*. In the Natural Deduction setting, Contraction amounts to the discharging of at least two copies (from the same parcel [16]) of an open assumption formula A during an Implication Introduction $\dfrac{[A] \ldots / B}{A \to B}$. This is because, for the so-called "Dialectica-relevant" contractions [17], A becomes part of the (raw, i.e., not yet normalized) realizing term. Therefore, the *a priori* (i.e., already at the extraction stage) elimination of some of these D-relevant contractions, rather than *a posteriori* (i.e., during the subsequent strong normalization process), represents an important complexity improvement of the extracted program. We exemplify our statement in the following Section 4.

4 A comparison of the three extraction techniques

It can be immediately seen, also from the machine benchmarks below, that the program yielded by the Light Dialectica interpretation clearly outperforms the algorithm given by the BBS refined A-translation [18]. The latter is at its turn much more efficient than the term extracted by means of the pure Gödel Dialectica interpretation, which contains an important quantity of redundant information. All three extracted (by the three program-synthesis techniques) terms are presented below in a human-processed adaptation of the raw MinLog output. See [11] for the pure machine-extracted programs. We stress the fact that the outcomes of the pure and the light Dialectica meta-algorithms would remain the same even if the input

[15] Hence of the formula A from the *open* assumption formulas $C^1, \ldots, C^n$. Here "open" is to be understood as "un-cancelled" or "un-discharged" and not necessarily as "un-closed".

[16] In the sense of the terminology from [9]. This is the same notion as "assumption variable" in [24].

[17] See [14] for full details on this terminology and generally for a large and unified exposition of the Light Dialectica extraction.

[18] This situation holds only for the more artificial input proof. When its distortions are eliminated by using a (search 1) restricted proof-search command instead of just (search), the run-time performance of the two extracted programs is quite equal. Here (search 1) means that the open assumptions may only be used at most once in the wanted searched proof. See [24] and the MinLog manual [25] for more details.

classical Fibonacci proof would be the original, undistorted one from [3]. Only the output of the BBS A-translation would get better when using its original input, see Footnote 18. Our point here is that if the user is unable or unwilling to optimize the input proof, then it is the responsibility of the extraction technique to deal with such practically very possible artificial situations and overcome the complexity loss.

It appears that the BBS refined A-translation is more directly dependent on the shape of the input proof and hence its performance decreases with the artificial distortions. This is because the BBS interpretation is based on an initial proof translation which literally includes the translation of the distortions. The witness is subsequently literally read from such a translated proof by Modified Realizability, which cannot avoid to preserve the distortions. In the case of the distorted classical Fibonacci proof, the redundant use twice of the (basically the induction hypothesis) assumption $\exists^{\mathtt{d}}k,l.\,G(n,k)\wedge G(n+1,l)$ during the automated search for a proof of the induction step will yield the double appearance of the type-2 functional H in the BBS-extracted program, see it below at 2). On the contrary, for both the D-interpretation and the LDI, the artificial distortion is harmless w.r.t. already the raw extracted program. Only a purely logical contraction, irrelevant already for the pure Dialectica, over the open assumption $\exists^{\mathtt{d}}k,l.\,G(n,k)\wedge G(n+1,l)$ will occur. This contraction has no computational content anyway, already in the case of the D-interpretation, because its formula translation has an empty universal side, see Definition 3.1 and [13,14] for full technical details. For the LDI the situation is identical, without any use of the special quantifiers without computational meaning. In fact not only this extra contraction, but the whole redundant proof-branch produced by the artificial distortion is without computational content under both the D-interpretation and the LDI. This is why the raw programs extracted by the two techniques are unchanged by the redundant distortion in the proof at input, i.e., regardless of whether the afore-mentioned assumption had been used once or twice, etc. Such an invariant situation was not possible for the BBS refined A-translation because this extraction technique lacks the full modularity of the Dialectica interpretations (see also [12] for an extended comment on this issue) and is more proof-dependent (as explained above).

We now attempt a theoretical explanation of why the program extracted by the LDI outperforms so neatly the one given by the pure D-interpretation. As hinted by Remark 3.4, the difference in performance is yielded by (the elimination of) a computationally D-redundant contraction. This contraction is given by the fact that the assumption $u:\forall n,k,l.\,[G(n,k)\wedge G(n+1,l)]\to G(n+2,k+l)$ is open in the proof of the induction step of the classical Fibonacci proof. The contraction is inserted in the proof to be mined independently of the number of open occurrences of u in the original proof at input. The mechanism of the Dialectica interpretation in Natural Deduction will actually double the number of open occurrences of u, hence a logical contraction appears anyway. See [14] for the technical details of such a contraction yielded by the simulation of the general Induction Rule (and thus also of the Induction Axiom) in terms of the more particular rule of induction restricted to assumption-less base and step input proofs. Now, what happens as a consequence of our "light" optimization? Because of the use of the quantifier without computational meaning $\overline{\forall}$ instead of the regular $\forall$ in u, this open assumption looses its Dialectica

computational content, which existed only due to the presence of (the three) regular $\forall$ in a positive position. See [14] for this terminology and full technical details. The number of open occurrences (in the original input proof) of the computationally D-redundant assumption u becomes irrelevant since this assumption is ignored anyway by the program extraction via the Light Dialectica Interpretation.

The subsequent computer benchmarks were performed on a DELL laptop (model X1, hence powered by an Intel Centrino CPU) running the Windows XP Prof. operating system. We used the more special MinLog distribution [11], which is not yet integrated with the official MinLog [25]. As Scheme interpreter we used the Petite Chez Scheme 7.0a, see [21]. The quantitative measures of computing time and space overhead were obtained by means of the Scheme "time" procedure.

1) The (MinLog, adapted) outcome of pure Gödel's Dialectica interpretation:

```
.........................
(add-var-name "n" "m" (py "nat"))
(add-var-name "G" (py "nat=>nat=>boole"))
(add-var-name "H" (py "(nat@@(nat@@nat)@@(nat@@nat))"))
.........................
> t_{PDI} == [G,n] left right
((Rec nat=>nat@@(nat@@nat)@@(nat@@nat))((0@0@0)@0@1)
  ([m,H] [if
     [if (G left left H left right left H)
        [if (G (Succ left left H) right right left H)
              (G (Succ(Succ left left H))
                   (left right left H + right right left H))
                   True]
             True]
          (m @ right H) (left H)] @ right right H@
                                  left right H + right right H)
          n)
> (time (nt (mk-term-in-app-form  t_{PDI} (pt "G") (pt "5"))))
   314 collections
   6031 ms elapsed cpu time, including 676 ms collecting
   6110 ms elapsed real time, including 687 ms collecting
   341280176 bytes allocated, including 337674848 bytes reclaimed
"5"
> (time (nt (make-term-in-app-form  t_{PDI} (pt "G") (pt "6"))))
  2700 collections
  56750 ms elapsed cpu time, including 9676 ms collecting
  58375 ms elapsed real time, including 10008 ms collecting
  2937460672 bytes allocated, including 2933419728 bytes reclaimed
"8"
```

2) The outcome of the BBS refined A-translation (MinLog output, adapted):

```
.........................
(add-var-name "i" "j" "k" "l" "m" "n" (py "nat=>nat=>nat"))
```

```
(add-var-name "f" (py "nat=>nat=>nat"))
(add-var-name "H" (py "(nat=>nat=>nat)=>nat"))
.........................
> > > t_{BBS} == "[k](Rec nat=>(nat=>nat=>nat)=>nat)
([f] f 0 1) ([l,H,f] H ([i,j] H ([n,m] f m (n+m))))
k ([n,m] n)"
> (time (nt (make-term-in-app-form t_{BBS} (pt "12"))))
  39 collections
  813 ms elapsed cpu time, including 109 ms collecting
  813 ms elapsed real time, including 107 ms collecting
  42919528 bytes allocated, including 39266296 bytes reclaimed
"144"
> (time (nt (make-term-in-app-form t_{BBS} (pt "15"))))
  321 collections
  7094 ms elapsed cpu time, including 1153 ms collecting
  7203 ms elapsed real time, including 1246 ms collecting
  348911096 bytes allocated, including 326154920 bytes reclaimed
"610"
```

3) The outcome of Light Dialectica interpretation (MinLog output, adapted):

```
.........................
(add-var-name "n" "m" (py "nat"))
(add-var-name "G" (py "nat=>nat=>boole"))
(add-var-name "H" (py "(nat@@nat)"))
.........................
> t_{LDI} == "[G,n] left ((Rec nat=>nat@@nat) (0@1)
([m,H] right H @ left H + right H) n)"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "15"))))
 6 collections
 125 ms elapsed cpu time, including 0 ms collecting
 140 ms elapsed real time, including 0 ms collecting
 6802576 bytes allocated, including 6383624 bytes reclaimed
"610"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "20"))))
 68 collections
 1343 ms elapsed cpu time, including 62 ms collecting
 1344 ms elapsed real time, including 63 ms collecting
 73584536 bytes allocated, including 71466424 bytes reclaimed
"6765"
> (time (nt (mk-term-in-app-form t_{LDI} (pt "G") (pt "25"))))
 750 collections
 16219 ms elapsed cpu time, including 2279 ms collecting
 16657 ms elapsed real time, including 2331 ms collecting
 816525224 bytes allocated, including 803991296 bytes reclaimed
"75025"
```

Notice that the above concrete quantitative measurements of time and space

overhead correspond to the distorted classical Fibonacci proof. For both Dialectica interpretation and the `LDI` they would be the same also for the original input proof from [3], or the proof obtained by limited automated search via `(search 1)` (instead of the unlimited `(search)`). On the contrary, for the `BBS` refined `A`-translation the difference would be rather big, since from the cleaner input proof a linear-time program is obtained, with run-time performance fairly equal to that of the output of the `LDI` technique (despite the difference of syntactic shape). The program $\mathtt{t}_{\mathtt{BBS}}$ displayed above at 2) can be written as a Scheme [21] program as follows:

```
(define (FiboBis n)
  (fibo2 n (lambda (k l) k)))
(define (fibo2 n1 f)
  (if (= n1 0)  (f 0 1)
      (fibo2 (- n1 1) (lambda (kk ll)
             (fibo2 (- n1 1) (lambda (k l) (f l (+ k l))))))))
```

Recall that the algorithm originally obtained in [3] could be spelled in Scheme as:

```
(define (Fibo n)
  (fibo1 n (lambda (k l) k)))
(define (fibo1 n1 f)
  (if (= n1 0)  (f 0 1)
        (fibo1 (- n1 1) (lambda (k l) (f l (+ k l))))))
```

We immediately figure out that the price to pay for the distortion in the input proof is rather big when using the `BBS` technique. The algorithm `FiboBis` is exponential in `n` because the call of `fibo2` on `n1` induces two recursive calls of `fibo2` on `n1-1`.

5 Conclusions and future work

More practical examples should be found for the application of the "light" optimization of Gödel's Dialectica interpretation. A negative result exists for the case of the `MinLog`-implemented semi-classical proof of Dickson's Lemma (see [6]). Here three nested Inductions give rise at three Contractions which are thus all three included in the extracted term(s), within the triply nested recursion. It is hence immediate to figure out that such a program would be *very* complex. Unfortunately, the Light Dialectica cannot repair this situation.

References

[1] Avigad, J. and S. Feferman, *Gödel's functional ('Dialectica') interpretation*, in: S. Buss, editor, *Handbook of Proof Theory*, Studies in Logic and the Foundations of Mathematics **137**, Elsevier, 1998 pp. 337–405.

[2] Berger, U., *Uniform Heyting Arithmetic*, Annals of Pure and Applied Logic **133** (2005), pp. 125–148, Festschrift for H. Schwichtenberg's 60th birthday.

[3] Berger, U., W. Buchholz and H. Schwichtenberg, *Refined program extraction from classical proofs*, Annals of Pure and Applied Logic **114** (2002), pp. 3–25.

[4] Berger, U., M. Eberl and H. Schwichtenberg, *Normalization by evaluation*, in: B. Möller and J. Tucker, editors, *Prospects for Hardware Foundations*, LNCS **1546**, Springer Verlag, 1998 pp. 117–137.

[5] Berger, U., M. Eberl and H. Schwichtenberg, *Term rewriting for normalization by evaluation*, Information and Computation **183** (2003), pp. 19–42, International Workshop on Implicit Computational Complexity (ICC'99).

[6] Berger, U., H. Schwichtenberg and M. Seisenberger, *The Warshall algorithm and Dickson's lemma: Two examples of realistic program extraction*, Journal of Automated Reasoning **26** (2001), pp. 205–221.

[7] Coquand, T. and M. Hofmann, *A new method for establishing conservativity of classical systems over their intuitionistic version*, Mathematical Structures in Computer Science **9** (1999), pp. 323–333.

[8] Ferreira, F. and P. Oliva, *Bounded functional interpretation*, Annals of Pure and Applied Logic **135** (2005), pp. 73–112.

[9] Girard, J.-Y., P. Taylor and Y. Lafont, "Proofs and Types," Cambridge University Press, 1989.

[10] Gödel, K., *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes*, Dialectica **12** (1958), pp. 280–287.

[11] Hernest, M.-D., *The `MinLog` proof-system for Dialectica program-extraction*, Free software, with full code source and documentation @ http://www.brics.dk/~danher/MinLogForDialectica.

[12] Hernest, M.-D., *A comparison between two techniques of program extraction from classical proofs*, in: M. Baaz, J. Makovsky and A. Voronkov, editors, *LPAR 2002: Short Contributions and CSL 2003: Extended Posters*, Kurt Gödel Society's Collegium Logicum **VIII** (2004), pp. 99–102.

[13] Hernest, M.-D., *Light Functional Interpretation*, Lecture Notes in Computer Science **3634** (2005), pp. 477 – 492, Computer Science Logic: 19th International Workshop, CSL 2005.

[14] Hernest, M.-D., "Feasible programs from (non-constructive) proofs by the light (monotone) Dialectica interpretation," PhD Thesis, École Polytechnique `and` University of Munich (LMU) (2006), In preparation, draft available @ http://www.brics.dk/~danher/teza/.

[15] Kohlenbach, U., *Proof Interpretations and the Computational Content of Proofs*, *Lecture Course*, latest version in the author's web page.

[16] Kohlenbach, U., *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 (1996), pp. 225–260.

[17] Kohlenbach, U., *Some logical metatheorems with applications in functional analysis*, Transactions of the American Mathematical Society **357** (2005), pp. 89–128.

[18] Kohlenbach, U. and P. Oliva, *Proof mining: a systematic way of analysing proofs in Mathematics*, Proceedings of the Steklov Institute of Mathematics **242** (2003), pp. 136–164.

[19] Kreisel, G., *Interpretation of analysis by means of constructive functionals of finite types*, in: A. Heyting, editor, *Constructivity in Mathematics*, North-Holland Publishing Company, 1959, pp. 101–128.

[20] Leuştean, L., *A quadratic rate of asymptotic regularity for CAT(0)-spaces*, Journal of Mathem. Analysis and Applications (2006), To appear, downloadable from Elsevier's "Science Direct", Articles in Press, Corrected Proof.

[21] Cadence Research Systems, *Chez Scheme*, http://www.scheme.com (2006).

[22] Oliva, P., *Understanding and using Spector's bar recursive interpretation of classical analysis*, in: *Proceedings of CiE'2006*, LNCS **3988** (2006), pp. 423–434, Available in the author's Web page @ http://www.dcs.qmul.ac.uk/~pbo/.

[23] Oliva, P., *Unifying functional interpretations*, Notre Dame Journal of Formal Logic (2006), to appear, downloadable from the author's Web page.

[24] Schwichtenberg, H., *Minimal logic for computable functions*, Lecture course on program-extraction from (classical) proofs. Available in the author's web page or in the Minlog distribution [25].

[25] Schwichtenberg, H. and Others, *Proof- and program-extraction system* Minlog, Free code and documentation at http://www.minlog-system.de.

[26] Troelstra, A., editor, "Metamathematical investigation of intuitionistic Arithmetic and Analysis", Lecture Notes in Mathematics **344**, Springer-Verlag, Berlin - Heidelberg - New York, 1973.