

A comparison between two techniques of program extraction from classical proofs

Mircea-Dan Hernest *

Mathematisches Institut der Universität München

The techniques for program extraction from (classical) proofs can be viewed as either based on cut elimination or on proof interpretations. In contrast to the hyper-exponential worst-case complexity of cut elimination, proof interpretations are practically feasible due to their modularity [4]. The normalization of terms (programs) extracted by proof interpretations would nevertheless equate the worst-case complexity of the two aforementioned classes of techniques. The separation of program extraction from cut elimination (normalization) by means of proof interpretations appears to be more useful in practice since the normalization of extracted programs is actually not needed in important applications [5].

Amidst proof interpretations, Kreisel's modified realizability and Gödel's functional interpretation (FI for short) have a prominent place. The two differ only with respect to the treatment of logical implication. In contrast to modified realizability, Gödel's technique takes counterexamples into account and therefore directly extends to classical (arithmetical) proofs via some negative translation. The extension goes further to (classical) analytical proofs [6]. Remarkable new theorems in Numerical Analysis have recently been obtained by means of (Kohlenbach's monotone variant of) functional interpretation [7]. Unlike FI, modified realizability requires an intermediate so-called Friedman-Dragalin "A-translation" after a negative translation and the combination of the three interpretations is only known to handle classical proofs of Π_2^0 -formulas. Various refinements of the latter combined proof interpretation were produced in recent years [1, 2]. One purpose of such refinements was the simplification of the translation in terms of complexity of the extracted programs. On the other hand the extension of the class of proofs at input was attempted.

The Berger-Buchholz-Schwichtenberg refined interpretation (BBS for short) [1] succeeds in both directions. It can directly handle classical proofs of formulas in a class which extends Π_2^0 . The type complexity of the extracted programs is significantly reduced in comparison with the unrefined combination of the three aforementioned translations. BBS was implemented in the proof system MINLOG and has been successfully used to extract programs from classical proofs. However, BBS cannot directly handle all proofs of Π_2^0 -formulas in the semi-classical

* danher@mathematik.uni-muenchen.de, Mathematisches Institut der Universität München, Theresienstr. 39, D-80333 München, Germany; funded by the "Deutsche Forschungsgemeinschaft" via the "Graduiertenkolleg Logik in der Informatik" (GKLI)

arithmetical system Z of Berger–Buchholz–Schwichtenberg [1]. System Z can be obtained from the minimal arithmetical system Z_0 by adding the ex-falso-quodlibet schema $\perp \rightarrow F$. BBS fails to directly interpret $\perp \rightarrow F$ in the sense that Theorem 3.3 of [1] would no longer hold if Z_0 were replaced by Z as input system. BBS can directly extract programs from *minimal logic* proofs of formulas in a class which extends Π_2^0 . Although the *strong* (intuitionistic) \exists may be allowed in parts of the proof at input, the existential quantifiers to be realized are *weak* (classical, $\neg\forall\neg$, usually denoted \exists^{cl} , see [8]). The situation can be repaired by using a preprocessing *reduced* negative translation [8]. BBS can even handle all PA^ω proofs of a class of formulas which extends Π_2^0 by means of so-called *definite* and *goal* formulas, after a negative translation preprocessing [9].

Gödel’s functional interpretation directly interprets all Z -proofs. System Z may be viewed as the Natural Deduction formulation of some restriction of PA^ω to the language without the strong \exists . It can also be viewed as a superset of HA^ω without the strong \exists . The design of system Z and of functional interpretation in Natural Deduction style [3] trigger the elimination of a preprocessing negative translation in the interpretation of this subsystem of PA^ω . The treatment of full PA^ω under FI nevertheless requires a negative translation preprocessing, just like BBS. The (rather big) difference is that FI handles¹ all proofs in (suitable versions of) PA^ω via this preprocessing.

In conclusion, all proofs which BBS handles directly, FI handles directly as well. On the other hand, there exist proofs which FI handles directly but BBS handles only via a negative translation. An example of such is the proof of the statement that if $h, s : \mathbb{N} \mapsto \mathbb{N}$ are two functions over the set of natural numbers and s has only strictly positive values then $h \circ s \circ h$ cannot be the identity (this case-study suggested by Berger got to be called *hsh*). More formally,

$$\forall s, h \left[\overbrace{(\forall m s(m) \neq 0)}^A \longrightarrow \exists n h(s(h(n))) \neq n \right] \quad (1)$$

We implemented a FI-extraction module in the proof system MINLOG. We compared the machine performance of the two program-extraction algorithms on the *hsh* example. The solution found by machine via both BBS and FI is unexpectedly clever. It relies on the fact that the conjunction of the following three clauses cannot hold in the presence of the implicative assumption of (1), which we denoted by A :

$$hs \underbrace{h(shh0)} = shh0 \quad (2)$$

$$hsh(h0) = h0 \quad (3)$$

¹ In the sense that realizing terms can be produced for the negative translation of the conclusion of the (arbitrary) PA^ω proof at input. This is the basis for the full modularity feature of FI-based techniques which contrasts with the more limited modularity of the techniques based on modified realizability (including the BBS technique). Here *modularity* means the ability of making unrestricted use of Lemmas in the proof at input, see also [4]. The techniques based on modified realizability can only use those Lemmas for which a realizer can be manually (external) if not machine (automated, internal) provided [9].

$$hsh(0) = 0 \tag{4}$$

More exactly, the following hold:

$$(2) \wedge (3) \implies hsh0 = shh0 \tag{5}$$

$$(4) \wedge (5) \implies shh0 = 0 \tag{6}$$

$$(6) \wedge A \implies \perp \tag{7}$$

We conclude that $(2) \wedge (3) \wedge (4) \wedge A \implies \perp$, hence $A \implies \neg(2) \vee \neg(3) \vee \neg(4)$.

In contrast to BBS, FI directly provides the additional term $h(h(0))$ which realizes the $\forall m$ of (1). This gives more information on the reasoning process employed for the extraction of the realizer for $\exists n$. Only the instance $m := h(h(0))$ is needed in the verifying proof. Notice that $s(h(h(0))) = 0$ of (6) is just a counterexample to A .

Since the FI algorithm is already known to have cubic time complexity (see [4] for a detailed proof of this), it would be interesting to investigate the computational complexity of BBS in order to have a quantitative comparison as well between the two extraction techniques.

References

1. U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.
2. T. Coquand and M. Hofmann. A new method for establishing conservativity of classical systems over their intuitionistic version. *Math. Structures Comput. Sci.*, 9(4):323–333, 1999.
3. M.-D. Hernest. A Natural Deduction formulation of functional interpretations. Draft, <http://www.mathematik.uni-muenchen.de/~danher>.
4. M.-D. Hernest and U. Kohlenbach. A complexity analysis of functional interpretations. Technical report BRICS RS-03-12, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark, February 2003.
5. U. Kohlenbach. Proof Interpretations and the Computational Content of Proofs. Lecture Course, <http://www.brics.dk/~kohlenb/newcourse.ps>.
6. U. Kohlenbach. Analysing proofs in analysis. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors, *Logic: from Foundations to Applications. European Logic Colloquium, Keele, 1993*, pages 225–260. Oxford University Press, 1996.
7. U. Kohlenbach and P. Oliva. Proof mining: a systematic way of analysing proofs in mathematics. *Proc. Steklov Inst. Math.*, 242:136–164, 2003.
8. H. Schwichtenberg. Minimal logic for computable functions. Lecture Course, <http://www.mathematik.uni-muenchen.de/~minlog/minlog/mlcf.ps>.
9. M. Seisenberger. *On the Constructive Content of Proofs*. PhD thesis, University of Munich, Faculty of Mathematics, 2003.