Applications of inductive definitions and choice principles to program synthesis

Ulrich Berger and Monika Seisenberger
{u.berger,csmona}@swansea.ac.uk

University of Wales Swansea

Abstract. We describe two methods of extracting constructive content from classical proofs, focusing on theorems involving infinite sequences and nonconstructive choice principles. The first method removes any reference to infinite sequences and transforms the theorem into a system of inductive definitions, the other applies a combination of Gödel's negative- and Friedman's A-translation. Both approaches are explained by means of a case study on Higman's Lemma and its well-known classical proof due to Nash-Williams. We also discuss some proof-theoretic optimizations that were crucial for the formalization and implementation of this work in the interactive proof system Minlog.

1 Introduction

This paper is concerned with the problem of finding constructive content in nonconstructive mathematical theorems. By the constructive content of a theorem we mean

- possibly, a constructive reformulation of the statement of the theorem,
- a constructive proof corresponding to the given nonconstructive proof,
- a program extracted from the proof that computes witnesses of existential statements in the theorem.

In the paper we will

- describe two different methods of transforming a nonconstructive proof into a constructive proof and critically assess them from a constructive point of view,
- sketch how to extract programs from these proofs and discuss the nature of these programs,
- discuss proof-theoretic optimizations which are of independent interest and seem to be necessary in order to make these methods feasible for the implementation of larger case studies in an interactive theorem prover.

The overall aim of the paper is to give insight into possible constructive aspects hidden in nonconstructive theorems and describe - in a nontechnical way - how these aspects can be brought to light and computationally exploited. The results presented here summarizes work done by the authors over the past few

years [Sei03,BO04,BBS02,BSS01,Sei01] largely as part of the MINLOG group at the University of Munich [BBS+98].

We will focus on theorems about infinite sequences as they typically occur in analysis and infinitary combinatorics. Our running example and main case study will be Higman's Lemma [Hig52][1] and its classical proof due to Nash-Williams [NW63] which we will briefly describe now. Higman's Lemma is concerned with *well-quasiorders* (wqos), that is, binary relations $(A, \leq_A)$ on a set A of letters with the property that each infinite sequence $(a_i)_{i\in\omega}$ of letters is 'good', i.e. there are indices $i < j$ such that $a_i \leq_A a_j$. (The simplest examples of wqos are finite sets with equality.) From $(A, \leq_A)$ one can derive a binary relation $(A^*, \leq_{A^*})$ on the set A^* of (finite) words over the alphabet A where $v \leq_{A^*} w$ means that v is *embeddable* into w, i.e. v can be obtained from w by deleting some letters or replacing them with ones which are smaller with respect to $\leq_A$.

Higman's Lemma If $(A, \leq_A)$ is a well-quasiorder, so is $(A^*, \leq_{A^*})$.

Proof (Nash-Williams, 1963). Our assumption is that every infinite sequence of letters is good. We have to prove that every infinite sequence of words is good. That a bad sequence of words is impossible is an immediate consequence of the following two facts:

1. For every bad sequence $(w_n)_{n\in\omega}$ of words there exists a bad sequence $(w'_n)_{n\in\omega}$ which is lexicographically smaller, i.e. there is an index n_0, such that $w'_i = w_i$ for all $i < n_0$ and w'_{n_0} is a proper initial segment of w_{n_0}.
2. If there exists a bad sequence of words, then there also exists a minimal bad sequence of words (with respect to the above lexicographic ordering).

Proof of 1. If $(w_n)_{n\in\omega}$ is bad, then all w_n must be nonempty, i.e. of the form $w_n = v_n * a_n$. As $(A, \leq_A)$ is a wqo, the sequence of letters $(a_n)_{n\in\omega}$ must contain a 'monotone' subsequence $a_{n_0} \leq_A a_{n_1} \leq_A \ldots$ with strictly increasing indices n_i. To obtain such a subsequence, one must apply a nonconstructive choice principle (we omit details). Now, the sequence $w_0, \ldots, w_{n_0-1}, v_{n_0}, v_{n_1}, \ldots$ is lexicographically smaller and also bad, since, if it were good, then so would be $(w_n)_{n\in\omega}$.

Proof of 2. Assume there is a bad sequence of words. We define a minimal bad sequence of words by repeated nonconstructive choices: if words $w_0, \ldots, w_{n-1}$ have already been chosen such that they begin a bad sequence, we choose a shortest word w_n such that $w_0, \ldots, w_{n-1}, w_n$ begins bad sequence as well. Clearly, the resulting sequence $(w_n)_{n\in\omega}$ is bad and lexicographically minimal.

The aspect of Nash-Williams' proof which is most problematic from a constructive point of view is not so much the use of classical logic, but the definition of infinite sequences in a constructively unacceptable way. One way around this problem is to reformulate the statement of the theorem in such a way that any reference to infinite objects is avoided. In section 2 we will do this, following

[1] Higman's lemma is used, for example, in term rewriting theory for termination proofs [CTB94,Tou02].

an approach of Coquand and Fridlender [CF94]. The main idea is to express the property of being a well-quasiorder by an inductive definition. In [CF94] the case $A = \{0, 1\}$ was treated whereas we allow an arbitrary alphabet A with a decidable well-quasiorder.

In section 3 we take another approach: we leave the statement of Higman's Lemma as it is, but apply a combination of Gödel's negative- and Friedman' A-translation [Fri78] to constructivize the proof. This idea goes back to Constable and Murthy [CM91] who formalized Higman's Lemma in a system of second order arithmetic representing infinite sequences by their graphs and proving the necessary choice principles by impredicative comprehension and classical logic [Mur90]. Our formalization differs from theirs in that we work in a finite type system where infinite sequences are available as objects type $\mathbb{N} \to \rho$. This has the advantage that the formalization of Nash-Williams' proof is technically simpler, however we have to include the nonconstructive choice principles as axioms. Applying negative- and A-translation we obtain a proof which is constructive modulo the translated choice principles. These translated principles can be reduced to a relativized version of bar induction, a principle whose constructive status is controversial. Nevertheless it is possible to extract an executable program that computes for any infinite sequence of words indices $i < j$ witnessing that it is good. However, to show the termination of this program one again needs these constructively questionable principles.

Analyzing the programs extracted from the respective proofs we will see that both use recursion along certain wellfounded trees which, however, are given in very different ways. The program extracted from the inductive proof works - as expected - on inductively defined trees, while the program from the A-translated proof recurs on the wellfounded tree given by a total continuous functional of type two.

We will also discuss some proof-theoretic optimizations which were crucial for fully formalizing and implementing this case study. The logical and inductive rules of the proof calculus are modified in order to be able to discard redundant parts of the programs already at extraction time, and the combined Gödel/A-translation is optimized with the effect that extracted programs have much lower types and a simpler control structure.

Proof-theoretically, both methods yield only suboptimal results as Higman's Lemma is in fact provable in ACA_0, a system that is conservative over arithmetic ([SS85,Gir87], see also [Sim88,dJP77,Sch79,MR90,RS93]). The interest in our methods lies in the fact that they both yield proofs and programs that make constructive use of the crucial ideas in Nash-Williams' nonconstructive proof, whereas the proof in [SS85] does not seem to be related to Nash-Williams' proof. The same applies to another inductive proof given by Fridlender [Fri97] which is based on an intuitionistic proof of Veldman (published in [Vel04]). Veldman's proof does not require decidability of $\leq_A$ and uses the idea of Higman's original proof. The proof-theoretic strength of a form of the minimal bad sequence argument that is sufficiently general for Nash-Williams' proof has been analyzed by Marcone [Mar96].

2 Inductive definitions

In this section we describe how Nash-Williams' classical proof of Higman's Lemma given in the introduction may be transformed into a constructive inductive argument. The crucial idea, due to Coquand and Fridlender [CF94], is to reformulate the notion of a well-quasiorder without referring to infinite sequences, but using a (generalized) inductive definition instead. For a finite sequence $as := [a_0, \ldots, a_{n-1}]$ of elements in $(A, \leq_A)$, let $\mathsf{Good}\, as$ be the property that there are $i < j < n$ such that $a_i \leq a_j$ (hence, an infinite sequence is good iff all its finite initial segments are). Define a predicate $\mathsf{Bar}_A \subseteq A^*$ inductively by

$$\frac{\mathsf{Good}\, as}{\mathsf{Bar}\, as} \qquad \frac{\forall a\, \mathsf{Bar}\, as * a}{\mathsf{Bar}\, as}.$$

Classically, $\mathsf{Bar}\, as$ means that every infinite sequence starting with as is good. Hence, classically, $(A, \leq_A)$ is a wqo iff $\mathsf{Bar}\,[\,]$ holds. The 'if' part of this equivalence can be proven easily using the induction principle for Bar

$$\frac{\forall as\, (\mathsf{Good}\, as \to B(as)) \qquad \forall as\, (\forall a\, B(as * a) \to B(as))}{\forall as\, (\mathsf{Bar}\, as \to B(as))}$$

with $B(as) :=$ 'every infinite sequence extending as is good'. The converse direction requires a non-constructive choice principle which will be discussed in detail in the next section.

Inductive formulation of Higman's Lemma $\mathsf{Bar}_A\,[\,] \to \mathsf{Bar}_{A^*}\,[\,]$.

In the following we sketch a constructive proof of this reformulation of Higman's Lemma that makes use of the essential ideas in Nash-Williams' proof. For simplicity we first restrict ourselves to a finite alphabet A (well-quasiordered by equality). Our goal is to prove $\mathsf{Bar}_{A^*}\,[\,]$. In the first part of the classical proof one defines from a bad sequence $(w_n)_{n<\omega}$ a lexicographically smaller bad sequence of the form $w_0, \ldots, w_{n_0-1}, v_{n_0}, v_{n_1}, \ldots$. We mimic this on the level of finite sequences by defining a relation $<_{\mathsf{NW}}$ (NW for Nash-Williams) such that $vs <_{\mathsf{NW}} ws$ holds iff all words in ws are nonempty and vs is obtained from ws be the following process. Take a letter a that occurs as the last letter of some word in ws and scan through ws from left to right, chopping off the last letter of the current word if this letter is a, respectively deleting the current word if it does not end with a but some word ending with a has been encountered before. The contrapositive of the first part of the classical proof, "if there is no bad sequence which is lexicographically smaller than $(w_n)_{n\in\omega}$, then $(w_n)_{n\in\omega}$ is good', corresponds now to

$$\forall ws\, (\, \forall vs(vs <_{\mathsf{NW}} ws \to \mathsf{Bar}_{A^*}\, vs) \to \mathsf{Bar}_{A^*}\, ws\,) \qquad (+)$$

This formula immediately entails $\mathsf{Bar}_{A^*}\,[\,]$, since for $ws = [\,]$ the premise of $(+)$ trivially holds ($[\,]$ has no $<_{\mathsf{NW}}$-predecessors). The main idea for proving $(+)$ is to introduce an inductive predicate Bars on A^{***} and a function $\mathsf{Folder} : A^{**} \to A^{***}$

such that $\mathsf{Bars}\,(\mathsf{Folder}\,ws)$ is equivalent to $\forall vs(vs <_{\mathsf{NW}} ws \to \mathsf{Bar}_{A^*}\, vs)$. The inductive definition of Bars parallels Bar:

$$\frac{\mathsf{Good}\, vs_i}{\mathsf{Bars}\,[vs_0,\ldots,vs_{n-1}]} \qquad \frac{\forall w\, \mathsf{Bars}\,[vs_0,\ldots,vs_i * w,\ldots vs_{n-1}]}{\mathsf{Bars}\,[vs_0,\ldots,vs_{n-1}]}.$$

The statement $\forall ws.\mathsf{Bars}\,(\mathsf{Folder}\,ws) \to \mathsf{Bar}_{A^*}\, ws$, which is equivalent to $(+)$, can be proven by main induction on the number of letters that do not occur as an end letter in ws and side induction on $\mathsf{Bars}\,(\mathsf{Folder}\,ws)$. More precisely, given ws with $\mathsf{Bars}\,(\mathsf{Folder}\,ws)$ and the respective induction hypotheses one shows $\forall w\, \mathsf{Bar}_{A^*}\, ws*w$ by structural induction on w, and then concludes $\mathsf{Bar}_{A^*}\, ws$ with the second introduction for Bar. This second side induction corresponds roughly to the definition of the minimal bad sequence in the second part of the classical proof. To prove the induction step the decidability of equality on A ($\leq_A$ in the general case) is used.

An inductive proof of Higman's Lemma for an arbitrary well-quasiordered alphabet - not only a finite one - is given in [Sei03] (see also [Sei01] for an earlier version). In this case, the induction on the number of letters that do not occur as last letters becomes an induction on the predicate Bar_A. Furthermore, the sequence structure, given by the predicate Folder, is replaced by a structure involving trees.

Formalization and Program Extraction We formalized and implemented the inductive proof of Higman's Lemma for a finite alphabet in the Minlog system. In order to obtain interesting computational content we proved from $\mathsf{Bar}_{A^*}\,[\,]$ the original statement of Higman's lemma, $\forall(w_n)_{n\in\omega}\,\exists i,j.\ i<j\ \wedge\ w_i \leq_{A^*} w_j$, using induction on Bar_{A^*}, and extracted a program from the latter proof. For the special case $A=\{0,1\}$ this may be found in the Minlog repository (www.minlog-system.de). The resulting program contains three nested recursions corresponding to the inductions in the proof. Its size as a Minlog term is surprisingly small – just about 50 lines – which is mainly due to refinements of the logic resulting in an optimized program extraction process. In order to explain these refinements, we need some basic facts about *realizability*, the proof-theoretic method underlying program extraction.

Realizability translates formulas into (possibly higher order) types essentially by removing all dependencies of formulas from objects. For example, a universal formula $\forall x^\rho A(x)$ translates into a function type $\rho\to\sigma$ (where A translates to σ) and an inductive predicate, as for example $\mathsf{Bar}\,as$, translates into a type of well-founded trees. The logical rules are translated into natural operations on these types. For example, $\forall$-introduction and -elimination translate into λ-abstraction and application, and introduction and elimination rules for an inductive definition become constructors and recursion operators for trees. Hence, extracted programs are higher type functional programs with iteration constructs restricted to (terminating) structural recursion.

The first refinement is achieved by distinguishing between two types of universal quantifiers, the usual quantifier $\forall$ and a 'noncomputational' quantifier

$\forall^{\mathsf{nc}}$. The meaning of $\forall^{\mathsf{nc}} x^\rho A(x)$ is roughly '$A(x)$ has been established for all x with a proof whose computational content does not depend on x'. Consequently one doesn't assign to $\forall^{\mathsf{nc}} x^\rho A(x)$ the type $\rho \to \sigma$, but just σ. Technically, the computational independence of a proof from an object variable is handled by a strengthened variable condition. The second refinement concerns the distinction between inductive definitions with and without computational content. For instance it is convenient to formalize the embeddability relation between words by means of an inductive definition,

$$\frac{}{[] \leq^* []} \qquad \frac{v \leq^* w}{v \leq^* w * a} \qquad \frac{v \leq^* w}{v * a \leq^* w * a},$$

even though this relation is decidable and could therefore be represented by a boolean function. Normally, such an inductive definition would introduce an inductive datatype into the program, but, if $\leq^*$ is declared as an inductive predicate without computational content, no extra data type is introduced. Of course, the introduction of inductive predicates without computational content is subject to extra conditions that ensure the soundness of the system. The distinction between logical constructs with and without computational content is similar to the distinction Set/Prop in intuitionistic type theory (see e.g. [PW93]), but seems to be more flexible.

3 Classical dependent choice

Now we show how to extract computational content directly from Nash-Williams' proof. Recall that in this proof one derives a contradiction from the assumption that a given sequence of words, let us call it f, is bad. We use Gödel's negative translation combined with the Friedman's A-translation [Fri78] to transform this classical proof into a constructive proof. For convenience we will in the following call this translation simply A-translation although we mean in fact the combination of Gödel- and A-translation. Hence, for a given existential formula A, the A-translation of a formula B, written B^A, is obtained by double negating all atomic and existential formulas (where $\neg C$ is defined as $C \to \bot$) and replacing $\bot$ (falsity) by the formula A. In the case of Higman's Lemma, A will be the formula expressing that f is good, i.e.

$$A :\equiv \exists i, j.\ i < j \wedge f(i) \leq^* f(j).$$

Under this translation all axioms concerning classical logic become intuitionistically provable, and instances of mathematical principles like induction are translated into (different) instances of the same principle. Most importantly, the (false) assumption that f is bad is translated into an intuitionistically provable formula. Altogether one obtains an intuitionistic proof of the translation of $\bot$, i.e. A, from which a program computing indices i, j with $f(i) \leq^* f(j)$ can be extracted. However this is not quite the full story since we did not say how to deal with the nonconstructive choices that occur in Nash-Williams' proof. The

kind of choices used there are captured by the following scheme of *dependent choice*

$$\mathbf{DC} \quad B([]) \wedge \forall xs(B(xs) \to \exists x\, B(xs * x)) \to \exists g \forall n\, B(\bar{g}n)$$

where $\bar{g}n := [g(0), \ldots, g(n-1)]$. In Nash-Williams' proof this scheme is used, for example, with the predicate $B(xs) :\equiv$ 'xs is a list of words that can be extended to an infinite bad sequence of words and if xs is nonempty then the last word in xs is as short as possible'. The A-translation of **DC** is (logically equivalent to)

$$\mathbf{DC}^A \quad \mathrm{Hyp}_1 \wedge \mathrm{Hyp}_2 \wedge \mathrm{Hyp}_3 \to A$$

where

$$\begin{aligned}
&\mathrm{Hyp}_1 \equiv B([])^A \\
&\mathrm{Hyp}_2 \equiv \forall xs(B(xs)^A \wedge (\exists x\, B(xs * x)^A \to A) \to A) \\
&\mathrm{Hyp}_3 \equiv \exists g \forall n\, B(\bar{g}n)^A \to A
\end{aligned}$$

Unlike induction on natural numbers, **DC** does *not* prove intuitionistically its A-translation. However, $\mathbf{DC}^A$ can be reduced intuitionistically to *relativized bar induction* [Coq91,Ber04] (a.k.a. extended bar induction or bar induction on species) a scheme whose constructive status is at least debatable [Luc73,Tro73]. We do not go into this reduction here, but instead informally explain how to directly interpret $\mathbf{DC}^A$ computationally in terms of realizability.

The idea of the following interpretation of $\mathbf{DC}^A$ is due to Berardi, Bezem and Coquand [BBC98]. In order to realize $\mathbf{DC}^A$ we assume we are given realizers G_1, G_2, G_3 of the hypotheses $\mathrm{Hyp}_1, \mathrm{Hyp}_2, \mathrm{Hyp}_3$ respectively. We have to compute a realizer of A. We use G_3. So, we need to compute some function g and a realizer h of $\forall n\, B(\bar{g}n)^A$. We compute g and h in stages. Suppose we have already computed finite approximations xs and ys to g and h, i.e. $xs = [x_0, \ldots, x_{n-1}]$, $ys = [y_0, \ldots, y_n]$ and y_i realizes $B([x_0, \ldots, x_{i-1}])^A$ for $i \leq n$ (we get started by setting $y_0 := G_1$). We run G_3 with the arguments g and h defined by

$$g(i) = \begin{cases} x_i & \text{if } i < n \\ 0 & \text{if } i \geq n \end{cases} \qquad h(i) = \begin{cases} y_i & \text{if } i \leq n \\ \text{see below} & \text{if } i > n \end{cases}$$

If G_3 happens not to query h at any $i > n$, then we are done. If it does, we compute a realizer $h(i)$ of $B(xs * 0 * \ldots * 0)^A$ (with $i - n$ 0s) using a realizer of $A \to B(xs * 0 * \ldots * 0)^A$ (since B^A is logically equivalent to a formula of the form $C \to A$ such a realizer trivially exists). So, we are back to the task of computing a realizer of A. This time we use the realizer G_2. We apply G_2 to our xs and y_n (we assumed y_n realizes $B(xs)^A$) and some realizer of the formula $\exists x\, B(xs * x)^A \to A$ which we compute as follows: given x and a realizer y of $B(xs * x)^A$ we need to compute a realizer of A. We do this by recurring to our main process, but now with the larger approximations $xs * x$ and $ys * y$. Luckily, each branch of the whole computation will eventually terminate because G_3 may assumed to be G_3 *continuous* and therefore querying its arguments at finitely many values only in order to compute a result.

The latter assumption can be justified by a model of realizability in which all functions are continuous. Such a model can, for example, be constructed from the total elements of the Scott/Ershov hierarchy of partial continuous functionals over the flat domains of partial booleans and partial natural numbers [Sco70,Ers77,Tro73,BO04]. It is important to note that models where all functionals are computable –like, for example, HEO [Tro73], or, more generally, the effective topos [Hyl82]– cannot be used here, since in order for the argument given above to be valid the sequences g and h approximated by the xs and ys have to be (possibly non-computable) free choice sequences (a related phenomenon is the fact that the unit interval of recursive reals is not compact).

The recursive process described above can written more formally as follows. A realizer of A is computed as $\Phi([],[G1])$ where for arguments xs, ys with lengths n and $n+1$ respectively (other arguments are uninteresting) $\Phi(\mathit{xs},\mathit{ys})$ is recursively defined by

$$\Phi(\mathit{xs},\mathit{ys}) = G_1(\tilde{\mathit{xs}}, \lambda i.\begin{cases} y_i & \text{if } i \le n \\ E(G_2(\mathit{xs}, y_n, (\lambda \mathit{xs}, \mathit{ys}.\Phi(\mathit{xs}*x, \mathit{ys}*y)))) & \text{if } i > n \end{cases})$$

where $\tilde{\mathit{xs}} := \lambda i.\mathsf{if}\ i < |xs|\ \mathsf{then}\ x_i\ \mathsf{else}\ 0^\rho$ and E is the (trivial) realizer of $A \to B(\mathit{xs}*0*\ldots*0)^A$. By some simple technical manipulations (like coding xs and ys into one sequence e.t.c.) this can be primitive recursively reduced to the following higher type recursion scheme

$$\mathbf{MBR}\quad \Psi(\mathit{xs}) = Y(\lambda i.\begin{cases} x_i & \text{if } i < |\mathit{xs}| \\ H(\mathit{xs}, \lambda x.\Psi(\mathit{xs}*x)) & \text{if } i \ge |\mathit{xs}| \end{cases})$$

where xs varies over finite sequences of some type ρ and the equation is of some type ν that doesn't contain function types (in the example of Higman's Lemma ν is the type of pairs of integers). In [BO04] the functional Ψ is called *modified bar recursion* and it is shown that Spector's bar recursion [Spe62], which is the scheme

$$\mathbf{SBR}\quad \Psi(\mathit{xs}) = \begin{cases} G(\mathit{xs}) & \text{if } Y(\tilde{\mathit{xs}}) < |\mathit{xs}| \\ H(\mathit{xs}, \lambda x.\Psi(\mathit{xs}*x)) & \text{if } Y(\tilde{\mathit{xs}}) \ge |\mathit{xs}| \end{cases}$$

can be primitive recursively defined from **MBR**, but not vice versa. Berardi, Bezem and Coquand [BBC98] proved the correctness of the above sketched computational interpretation of **DC** using a special realizability interpretation based on infinite terms. Oliva and the first author [BO04] showed that Kreisel's modified realizability [Kre59] together with Plotkin's adequacy theorem [Plo77] can be used instead (thus avoiding infinite terms, the role of which is taken over by the Scott/Ershov model of partial continuous functionals).

In our case study we worked with a realizer of $\mathbf{DC}^A$ which is defined from **MBR** [Sei03]. It would be even more direct (and probably technically simpler) to work with the realizer of the minimal bad sequence sequence argument in the form of an open induction principle [Coq97,Ber04] instead of reducing open induction to (classical) dependent choice.

Proof-theoretic and computational optimizations As explained above the A-translation replaces every atomic formula C by $(C \to A) \to A$ where A is an existential formula. This has the effect that higher types and many case distinctions come up in the extracted program which may lead to complex and inefficient code. In [BBS02] a refined A-translation is introduced that minimizes double negations and hence reduce these negative effects. These refinements are implemented in Minlog and we have tested them in our case study.

Another improvement is specific to Minlog's implementation of normalization by evaluation [BES98]. We introduced the functional Ψ in Minlog as a program constant together with a rewrite rule corresponding to **MBR**. Normalization by evaluation means that in order to normalize a term, it is evaluated as a functional (Scheme) program where **MBR** is interpreted as a recursive higher type procedure that is defined according to the rewrite rule for **MBR**. This normalization procedure is rather fast, however, when running (i.e. normalizing) programs containing **MBR** one observes a certain inefficiency which can be explained by the fact that if (in **MBR**) Y calls its argument at different values $\geq |xs|$ the expression $H(xs, \lambda x.\Psi(xs * x))$ (which does not depend on k) is evaluated repeatedly. An obvious method to avoid this inefficiency is to equip the argument of Y with an internal memory that stores the value of the expression $H(xs, \lambda x.\Psi(xs * x))$ after it has been computed for the first time. We have done this and gained a considerable speed-up.

4 Conclusion

In this paper we discussed two methods of constructivizing classical proofs. We applied them to the classical Nash-Williams proof of Higman's Lemma and extracted two different programs. Both methods and the case study are implemented in Minlog (`www.minlog-system.de`). The main computational principle used in the extracted programs is recursion on wellfounded trees. However, while in the program extracted from the inductive proof the trees are inductively generated as the elements of an inductive data-type, in the program obtained from the A-translated Nash-Williams' proof wellfounded trees are given by continuous functionals of type two. Although both forms of wellfounded recursion are known to be of different strength in general ([Spe62], [Tro73], Appendix by J. Zucker), it is possible that the particular instances used here are in some way equivalent. It was our hope that by analyzing the extracted programs such an equivalence could be revealed. Unfortunately, the program corresponding to the second version is still too complex to permit such an analysis, although it is considerably shorter than the program extracted by Murthy [Mur90]. It also remains unclear how our programs are related to those extracted by Murthy [Mur90] and Herbelin [Her94]. Since Higman's Lemma is provable in a system which is conservative over first-order arithmetic, we know that wellfounded recursion is not needed at all, but Gödel primitive recursion (of type $\mathbb{N} \to \mathbb{N}$ if the alphabet A is finite) would suffice. Such a program could be extracted in principle from an imple-

mentation of e.g. the proof by Schütte and Simpson [SS85], but nobody has so far undertaken such an implementation.

The inductive approach presented in this paper may be carried out in any theorem prover supporting program extraction from inductive definitions (see, for instance, [Berg04] for an implementation of Higman's Lemma in Isabelle and [Fri97] for a formalization of a different proof in Alf). The second approach is more specific to the Minlog system, since it requires an implementation of the refined A-translation which does not seem to be available in other systems. In particular, the optimizations via memoization directly rely on Minlog's normalization procedure.

The program extraction from Nash-Williams' classical proof via A-translation described in section 3 could be extended in a straightforward way to a corresponding classical proof of Kruskal's Theorem, even in its strong form with gap condition [Sim85]. The latter would be interesting because then we could extract a program from a theorem for which no constructive proof is known so far. On the other hand, the inductive method of section 2 seems to be much harder to generalize, since, unlike the A-translated proof, the inductive proof is not obtained by a mechanical translation process, but rather by picking up the essential ideas and transforming them into a constructive argument. At present, it is an open problem to find a corresponding inductive proof for Kruskal's theorem.

Another interesting problem is whether *strong normalization*, i.e. termination of *every* reduction sequence, holds for suitable variants of **MBR** or **SBR** (the versions given above are obviously not strongly normalizing since the 'else' branch of the case analysis can always be rewritten). For example, **MBR** could be reformulated by replacing the conditional expression in the right hand side of its defining equation by a call of an auxiliary constant Ψ' with an extra (boolean) argument in order to force evaluation of the test $k < |s|$ before the subterm $\Phi ygh(s * x)$ may be further reduced.

$$\begin{aligned}\Psi(xs) &= Y(\lambda i.\Psi'(i, xs, i < |xs|))\\ \Psi'(i, xs, \mathsf{T}) &= x_i\\ \Psi'(i, xs, \mathsf{F}) &= H(xs, \lambda x.\Psi(xs * x))\end{aligned}$$

Proving strong normalization for recursion schemes of this kind is the subject of ongoing research.

References

[BBS+98] H. Benl, U. Berger, H. Schwichtenberg, M. Seisenberger, and W. Zuber. Proof theory at work: Program development in the Minlog system. In W. Bibel and P. H. Schmitt, editors, *Automated Deduction – A Basis for Applications II*, 41–71. Kluwer, Dordrecht, 1998.

[BBC98] S. Berardi, M. Bezem, and T. Coquand. On the computational content of the axiom of choice. *The Journal of Symbolic Logic*, 63(2):600–622, 1998.

[Ber04] U. Berger. A Computational Interpretation of Open Induction. *Proc 19th IEEE Symp. Logic in Computer Science*, 2004.

[BO04] U. Berger and P. Oliva. Modified Barrecursion and Classical Dependent Choice. To appear in Lecture Notes in Logic, Springer, 200x.

[BBS02] U. Berger, W. Buchholz, and H. Schwichtenberg. Refined program extraction from classical proofs. *Annals of Pure and Applied Logic*, 114:3–25, 2002.

[BSS01] U. Berger, H. Schwichtenberg, and M. Seisenberger. The Warshall Algorithm and Dickson's Lemma: Two Examples of Realistic Program Extraction. *Journal of Automated Reasoning*, 26:205–221, 2001.

[BES98] U. Berger, M. Eberl, and H. Schwichtenberg. Normalization by evaluation. In B. Möller and J.V. Tucker, editors, *Prospects for Hardware Foundations*, Lecture Notes in Computer Science 1546, 117–137. Springer, 1998.

[Berg04] S. Berghofer. A constructive proof of Higman's Lemma in Isabelle. In S. Berardi and M. Coppo, editors, em Types for Proofs and Programs (TYPES 2003), Lecture Notes in Computer Science 3085, Springer, 2004.

[CTB94] E. A. Cichon and E. Tahhan Bittar. Ordinal recursive bounds for Higman's theorem. *Theoretical Computer Science*, 201:63–84, 1994.

[CM91] R. Constable and C. Murthy. Finding computational content in classical proofs. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, 341–362, Cambridge University Press, 1991.

[Coq91] T. Coquand. Constructive Toplogy and Combinators. In *Constructivity in Computer Science*, Lecture Notes in Computer Science 613, 159–164, 1991.

[Coq97] T. Coquand. A Note on the Open Induction Principle. Chalmers University, 1997.

[CF94] T. Coquand and D. Fridlender. A proof of Higman's lemma by structural induction, 1994. ftp://ftp.cs.chalmers.se/pub/users/coquand/open1.ps.Z.

[Ers77] Y. Ershov. Model C of partial continuous functionals. In R. Gandy and M. Hyland, editors, *Logic Colloquium 1976*, 455-467, North Holland, 1977.

[Fri97] D. Fridlender. *Higman's Lemma in Type Theory.* PhD thesis, Chalmers University of Technology and University of Göteburg, 1997.

[Fri78] H. Friedman. Classically and intuitionistically provably recursive functions. In D. Scott, G. Müller, editors, *Higher Set Theory*, Lecture Notes in Mathematrics 669, 21–28, Springer, 1978.

[Gir87] J.-Y. Girard. *Proof theory and complexity.* Bibliopolis, Naples, 1987.

[Göd58] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts. *Dialectica*, 12:280–287, 1958.

[Her94] H. Herbelin. A program from an A-translated impredicative proof of Higman's Lemma. http://coq.inria.fr/contribs/higman.html, 1994.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society (3)*, 2(7):326–336, 1952.

[Hyl82] M. Hyland. The effective topos. In A.S. Troelstra and D. Van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, 165–216. North Holland, 1982.

[dJP77] D. de Jongh and R. Parikh. Well partial orderings and their order types. *Indagationes Mathematicae*, 39:195–207, 1977.

[Kre59] G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. A. Heyting, *Constructivity in Mathematics*, 101–128. North–Holland, 1959.

[Luc73] H. Luckhardt. Extensional Gödel functional interpretation – a consistency proof of classical analysis. *Lecture Notes in Mathematics*, 306, Springer, 1973.

[Mar96] A. Marcone. On the logical strength of Nash-Williams' theorem on transfinite sequences. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors,

Logic: from Foundations to Applications; European logic colloquium, 327–351, 1996.

[Mur90] C. R. Murthy. *Extracting Constructive Content from Classical Proofs*. PhD thesis, Ithaca, New York, 1990.

[MR90] C. R. Murthy and J. R. Russell. A Constructive proof of Higman's Lemma. In *Proceedings of the Fifth Symposium on Logic in Computer Science*, 257–267, 1990.

[NW63] C. St. J. A. Nash-Williams. On well–quasi–ordering finite trees. *Proc. Cambridge Phil. Soc.*, 59:833–835, 1963.

[PW93] C. Paulin-Mohring and B. Werner. Synthesis of ML programs in the system Coq. *Journal of Symbolic Computation*, 15:607–640, 1993.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 4:223–255, 1977.

[RS93] F. Richman and G. Stolzenberg. Well Quasi–Ordered sets. *Advances in Mathematics*, 97:145–153, 1993.

[Sch79] D. Schmidt. Well–orderings and their maximal order types, 1979. Habilitationsschrift, Mathematisches Institut der Universität Heidelberg.

[Sei01] M. Seisenberger. An Inductive Version of Nash-Williams' Minimal-Bad-Sequence Argument for Higman's Lemma. In P. Callaghan, e.al., *Types for Proofs and Programs*, Lecture Notes in Computer Science 2277, Springer, 2001.

[Sei03] M. Seisenberger. *On the Constructive Content of Proofs*. PhD thesis, University of Munich, 2003.

[SS85] K. Schütte and S. G. Simpson. Ein in der reinen Zahlentheorie unbeweisbarer Satz über endliche Folgen von natürlichen Zahlen. *Archiv für Mathematische Logik und Grundlagenforschung*, 25:75–89, 1985.

[Sco70] D. S. Scott. Outline of a mathematical theory of computation. *4th Annual Princeton Conference on Information Sciences and Systems*, 169–176, 1970.

[Sim85] S. G. Simpson. Nonprovability of certain combinatorial properties of finite trees. In L.A. Harrington, e.al., *Harvey Friedman's Research on the Foundations of Mathematics*, 87–117, North–Holland, 1985.

[Sim88] S. G. Simpson. Ordinal numbers and the Hilbert Basis Theorem. *J. Symbolic Logic* 53, 961–974, 1988.

[Spe62] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathmatics. In F. D. E. Dekker, *Recursive function theory*, 1–27, North–Holland, 1962.

[Tro73] A. S. Troelstra. Metamathematical Investigation of Intuitionistic Arithmetic and Analysis, *Lecture Notes in Mathematics* 344, Springer, 1973.

[Tou02] H. Touzet. A characterisation of multiply recursive functions with Higman's lemma. *Information and Computation*, 178:534–544, 2002.

[Vel04] W. Veldman. An intuitionistic proof of Kruskal's theorem. *Archive for Mathematical Logic*, 43:215–264, 2004.