Refined Program Extraction from Classical Proofs: Some Case Studies

Helmut Schwichtenberg*

1 Introduction

It is well known that it is undecidable in general whether a given program meets its specification. In contrast, it can be checked easily by a machine whether a formal proof is correct, and from a constructive proof one can automatically extract a corresponding program, which by its very construction is correct as well. This – at least in principle – opens a way to produce correct software, e.g. for safety-critical applications. Moreover, programs obtained from proofs are "commented" in a rather extreme sense. Therefore it is easy to maintain them, and also to adapt them to particular situations.

We will concentrate on the question of classical versus constructive proofs. It is well known that any classical proof of a specification of the form $\forall x \exists y B$ with B quantifier-free can be transformed into a constructive proof of the same formula (for particularly simple proofs, cf. Friedman [5] or Leivant [7]). However, when it comes to extraction of a program from a proof obtained in this way, one easily ends up with a mess. Therefore, some refinements of the standard transformation are necessary.

In the present paper we make use of a refined method of extracting programs from classical proofs; it is developed in detail in [1]. We demonstrate its use with two rather detailed case studies. Specifically, we extract programs from classical proofs of the existence of

- integer square roots, and
- integer coefficients to linearly combine the greatest common divisor of two numbers from these numbers,

The latter example has already been treated in [2]; here we show how it can be dealt with in the present form of the theory. It turns out that the algorithm does not change; however, the method of how to extract a program from this classical proof becomes more perspicious.

Other interesting examples of program extraction from classical proofs have been studied by Murthy [8], the group around Coquand (see e.g. [3]) in a type theoretic context and by Kohlenbach [6] using a Dialectica-interpretation.

The paper is organized as follows. In section 2 the general background is developed. We start in 2.1 with a short exposition of Gödel's system T. Tait's proof of termination for the simply typed λ-calculus is presented in detail, and is then extended to T. In 2.2 we fix our version of intuitionistic arithmetic for functionals, and recall how classical arithmetic can be seen as a subsystem. Subsection 2.3 develops the basic machinery of modified realizability, and contains a detailed proof of the soundness theorem. The

*Mathematisches Institut der Universität München, Theresienstraße 39, D-80333 München, Germany. Phone +49 89 2394 4413, Fax +49 89 280 5248, `schwicht@rz.mathematik.uni-muenchen.de`

main part of the paper is section 3. There definite and goal formulas are defined, and the main theorem on program extraction from classical proofs (Theorem 3.2) is stated; for the proof we have to refer to [1]. Subsections 3.2 and 3.3 then develop the two case studies mentioned above.

Acknowledgements The proof of termination for Gödel's system T uses ideas from Ulrich Berger, Holger Benl, Ralph Matthes and Thorsten Altenkirch. Monika Seisenberger and Felix Joachimski have contributed a lot to the Minlog system, particularly to the implementation of the translation of classical proofs into constructive ones.

2 General Background

2.1 Gödel's System T

We introduce Gödel's system T of primitive recursive functionals of finite type, formulated as a simply typed lambda calculus with higher type recursion constants.

Types are as for the simply typed lambda calculus, but with concrete ground types ι for the natural numbers and o for the boolean objects *true* and *false*.

$$\iota \mid o \mid \rho \times \sigma \mid \rho \to \sigma.$$

The *constants* are

$$\mathit{true}^o \mid \mathit{false}^o \mid 0^\iota \mid \mathrm{S}^{\iota\to\iota} \mid \mathcal{R}_{o,\rho} \mid \mathcal{R}_{\iota,\rho}.$$

$\mathcal{R}_{\iota,\rho}$ is the *primitive recursion operator* of type $\rho \to (\iota \to \rho \to \rho) \to \iota \to \rho$. $\mathcal{R}_{o,\rho}$ is of type $\rho \to \rho \to o \to \rho$ and represents *boolean induction*, i.e. definition by cases. Instead of $\mathcal{R}_{o,\rho}MNK$ we will often write **if** K **then** M **else** N.

Terms are

$$x^\rho \mid c^\rho \ (c^\rho \text{ a constant}) \mid \langle M, N\rangle \mid \pi_0(M) \mid \pi_1(M) \mid \lambda x^\rho M \mid MN$$

with the usual typing rules. The *conversions* are those for the simply typed lambda calculus, plus some new ones for the recursion operators. We write $K+1$ for $\mathrm{S}K$.

$$\begin{array}{rcl} \mathcal{R}_{o,\rho}MN\,\mathit{true} & \mapsto_{\mathcal{R}} & M \\ \mathcal{R}_{o,\rho}MN\,\mathit{false} & \mapsto_{\mathcal{R}} & N \\ \mathcal{R}_{\iota,\rho}MN0 & \mapsto_{\mathcal{R}} & M \\ \mathcal{R}_{\iota,\rho}MN(K+1) & \mapsto_{\mathcal{R}} & NK(\mathcal{R}_{\iota,\rho}MNK) \end{array}$$

We will prove that for this system of terms every term strongly normalizes. Since the normal form is uniquely determined, the relation $M =_{\beta\mathcal{R}} N$ is decidable (by normalizing M and N). By identifying $=_{\beta\mathcal{R}}$-equal terms (i.e. treating equations on the meta level) we can greatly simplify many formal derivations.

Tait's Proof of Termination for the Simply Typed Lambda Calculus

We first give a well-known proof of *termination* of the simply typed lambda calculus, using method due to W.W. Tait. This proof will later be extended to Gödel's system T.

Tait's proof rests on defining so-called *strong computability predicates*. We present the proof here in a form which avoids intuitive arguments concerning reduction sequences and therefore is suitable for formalization in a theory of inductive definitions.

We begin with a definition of *strongly normalizable terms*, by a strictly positive induction.

If all M' such that $M \longrightarrow_\beta M'$ are strongly normalizable (sn), then so is M.

Obviously M is sn if and only if every reduction sequence starting with M terminates after a finite number of steps. This can be seen as follows. $\Longrightarrow$. Induction on the definition of "strongly normalizable". Consider a reduction sequence starting with M and therein the first reduct M'. The IH for M' yields the assertion. $\Longleftarrow$. By induction on the length of the longest reduction sequence (König's lemma).

We note a number of the properties of the notion "strongly normalizable", to be used below.

$$\text{If all terms } \vec{M} \text{ are sn, then so is } x\vec{M}. \tag{1}$$

Proof. Induction on the definition of sn for $\vec{M}$. Let $x\vec{M} \longrightarrow_\beta N$ be given. It suffices to show that N is sn. From $x\vec{M} \longrightarrow_\beta N$ it follows that $N = x\vec{M}'$, where $\vec{M}'$ arises by substitution of M_i by M_i' with $M_i \longrightarrow_\beta M_i'$. It is to be proved that $x\vec{M}'$ is sn. This follows from the IH for $\vec{M}'$. □

$$\text{If } Mx \text{ is sn, then so is } M. \tag{2}$$

Proof. Induction on the definition of sn for Mx. Let $M \longrightarrow_\beta M'$ be given. It suffices to show that M' is sn. From $M \longrightarrow_\beta M'$ we get $Mx \longrightarrow_\beta M'x$. The IH for $M'x$ then yields that M' is sn. □

We now define when a term M^ρ is strongly computable (sc), by induction on the type ρ.

- M^ι is sc if M^ι is sn.
- $M^{\rho\to\sigma}$ is sc if for every sc N^ρ also $(MN)^\sigma$ is sc.

A term M is called *strongly computable under substitution* if for any sc terms $\vec{N}$ also $M[\vec{x} := \vec{N}]$ is sc.

We note a property of the notion "strongly computable" which will be used below.

$$M \text{ is sc if and only if } M\vec{N} \text{ is sc for all sc } \vec{N}. \tag{3}$$

Proof. Induction on the length of $\vec{N}$. □

Lemma 2.1 *a. Every sc term M^ρ is sn.*

b. If $\vec{M}$ are sn, then $(x\vec{M})^\rho$ is sc.

Proof. By simultaneous induction on the type ρ. *Case* ι. a. By definition. b. By (1).

Case $\rho \to \sigma$ a. Let $M^{\rho\to\sigma}$ be sc. By IHb (with $\vec{M}$ empty) and the definition of strong computability $(Mx)^\sigma$ is sc. By IHa Mx then is sn. By (2) M is sn too. b. Consider $(x\vec{M})^{\rho\to\sigma}$ with $\vec{M}$ sn. Let N^ρ be sc. We have to show that $(x\vec{M}N)^\sigma$ is sc. By IHa N is sn, hence $x\vec{M}N$ is sc by IHb. □

If $M \longrightarrow_\beta M'$ and M is sn, then by the definition of strong normalizability also M' is sn. We now show that the corresponding assertion is also valid for strong computability.

Lemma 2.2 *If $M \longrightarrow_\beta M'$ and M is sc, then so is M'.*

Proof. We use (3). Let $\vec{N}$ be a list of sc terms such that $M\vec{N}$ is of ground type. Then $M\vec{N}$ is sc by (3), hence also sn. Furthermore we have $M\vec{N} \longrightarrow_\beta M'\vec{N}$. By definition of strong normalizablility $M'\vec{N}$ is sn. Thus by (3) M' is sc. $\square$

Lemma 2.3 *Let N be sn. If $M[x := N]\vec{L}$ is sn and of a ground type, then so is $(\lambda x\, M)N\vec{L}$.*

Proof. By induction on the strong normalizability for N and $M[x := N]\vec{L}$. So we consider all the reducts K of $(\lambda x\, M)N\vec{L}$. Clearly it suffices to show that every such K is sn.

Case $K = M[x := N]\vec{L}$, i.e. we have an outer β-conversion. Hence K is sn by assumption.

Case $K = (\lambda x\, M')N\vec{L}$ with $M \longrightarrow_\beta M'$. Then we have $M[x := N]\vec{L} \longrightarrow_\beta M'[x := N]\vec{L}$. By definition of strong normalizability $M'[x := N]\vec{L}$ is sn. Hence by IH K is sn.

Case $K = (\lambda x\, M)N'\vec{L}$ with $N \longrightarrow_\beta N'$. Then we have $M[x := N]\vec{L} \longrightarrow_\beta^* M[x := N']\vec{L}$. By definition of strong normalizability $M[x := N']\vec{L}$ and N' are sn. Hence by IH K is sn.

Case $K = (\lambda x\, M)N\vec{L}'$ with $L_i \longrightarrow_\beta L_i'$ for i and $L_j = L_j'$ for $j \neq i$. Then we have $M[x := N]\vec{L} \longrightarrow_\beta M[x := N]\vec{L}'$. By definition of strong normalizability $M[x := N]\vec{L}'$ is sn. Hence by IH K is sn. $\square$

Corollary 2.4 *If $M[x := N]$ is sc for all sc N, then also $\lambda x\, M$ is sc.*

Proof. Let $M[x := N]$ be sc for all sc N. We have to show that $\lambda x\, M$ is sc. So let N and $\vec{L}$ be sc such that $(\lambda x\, M)N\vec{L}$ is of ground type. We must show that $(\lambda x\, M)N\vec{L}$ is sc. Since $M[x := N]$ is sc by assumption, $M[x := N]\vec{L}$ is sc too and hence also sn. Since by lemma 2.1a N is also sn, by lemma 2.3 it follows that $(\lambda x\, M)N\vec{L}$ is sn and hence also sc. $\square$

Lemma 2.5 *Every term M is sc under substitution.*

Proof. By induction on M. *Case* x. The claim follows from lemma 2.1b or by assumption.

Case MN. $\vec{K}$ be sc. We have to show that $M[\vec{x} := \vec{K}]N[\vec{x} := \vec{K}]$ are sc. This clearly holds, since by IH $M[\vec{x} := \vec{K}]$ as well as $N[\vec{x} := \vec{K}]$ are sc.

Case $\lambda x\, M$. $\vec{K}$ be sc. We have to show that $\lambda x\, (M[\vec{x} := \vec{K}])$ is sc. We now apply corollary 2.4. Let N be sc. By IH for M also $M[x, \vec{x} := N, \vec{K}]$ is sc. Hence by corollary 2.4 the claim follows. $\square$

From lemma 2.5 and lemma 2.1 we directly get

Theorem 2.6 $\longrightarrow_\beta$ *is terminating, i.e. every term M is sn.*

Extension of the Termination Proof to Gödel's System T

This proof can easily be extended to terms with pairing $\langle M_0, M_1\rangle$ and projections $\pi_0(M)$, $\pi_1(M)$. We now show that addition of the conversion rules above for $\mathcal{R}$ does not destroy termination.

We first change the definition of sc for the ground type ι. It is not defined just to be sn (as before), but now is defined inductively, as follows.

Definition 2.7 *M is sc, if*

- *all M' such that $M \longrightarrow_\beta M'$ are sc, and in addition*
- *if $M = \mathrm{S}M_0$, then M_0 is sc.*

Theorem 2.8 *$\longrightarrow_{\beta\mathcal{R}}$ is terminating.*

Proof. We extend the argument above. Clearly it suffices to show that the constants $\mathcal{R}_{\iota,\rho}$ and $\mathcal{R}_{o,\rho}$ are sc. We restrict ourselves to $\mathcal{R}_{\iota,\rho}$; for $\mathcal{R}_{o,\rho}$ the argument is similar (and simpler). So let M, N, K be sc. We must show that $\mathcal{R}MNK$ is sc.

The proof is by induction on sn for M, N and on sc for K. So let $\vec{L}$ be sc. We must show that $\mathcal{R}MNK\vec{L}$ is sc. Since this term is not of the form $\mathrm{S}M_0$, it suffices to consider all reducts Q of $\mathcal{R}MNK\vec{L}$ and to show that every such reduct Q is sc.

Case $K = K_0 + 1$, $Q = NK_0(\mathcal{R}MNK_0)\vec{L}$. We first show that $\mathcal{R}MNK_0$ is sc. So let $\vec{L}'$ be sc. Since $K = K_0 + 1$ is sc, by definition also K_0 is. So by induction hypothesis $\mathcal{R}MNK_0$ is sc, and we obtain that Q is sc.

If the reduction takes place within a subterm $M, N, K, \vec{L}$ the claim follows by induction hypothesis (here we need that strong computability is preserved under conversion steps; cf. lemma 2.2). Since the case $K = 0$, $Q = M\vec{L}$ is trivial this proves the claim. $\square$

Now we can conclude via Newman's Lemma [9] that the normal form is uniquely determined. For simplicity we identify terms with the same $\beta\mathcal{R}$-normal form. Hence every closed term of type ι is identified with a term of the form $S(S(S\ldots(S0)\ldots))$ and every closed term of type o is identified with either *true* or *false*. Such terms are denoted by $\underline{n}$ and called *numerals* (even if they are of type o).

2.2 Intuitionistic Arithmetic for Functionals

The system we consider is essentially Heyting's intuitionistic arithmetic in finite types HA^ω as described e.g. in [4]. It is based on Gödel's system T and just adds the corresponding logical and arithmetical apparatus to it. Classical arithmetic is obtained by restriction to formulas without the constructive existential quantifier $\exists^*$ but using the classical definition $\exists = \neg\forall\neg$ instead. Equations are treated on the meta level by identifying terms with the same normal form.

We now write $r, s, t, \ldots$ for (object) terms, and reserve $M, N, K, \ldots$ for derivation terms.

Arithmetic: Predicate symbols and atomic formulas

Every predicate symbol P now has a fixed arity, which is a tupel of types $(\rho_1, \dots, \rho_k)$. So atomic formulas are of the form $P(r_1^{\rho_1}, \dots, r_k^{\rho_k})$. The choice of the predicate symbols depends on the particular problem under consideration. In most cases there will be the predicate symbols $\bot$ (falsity) of arity () and *atom* of arity (o). The intended interpretation of *atom* is the set $\{\mathit{true}\}$. Hence "$\mathit{atom}(t)$" means "$t = \mathit{true}$". Every decidable relation (i.e., every relation given by a term t of type o), can then be written in the form $\mathit{atom}(t)$. If confusion is unlikely, we will write t instead of $\mathit{atom}(t)$. So formulas are

$$P(r_1^{\rho_1}, \dots, r_k^{\rho_k}) \mid A \land B \mid A \to B \mid \forall x^\rho A \mid \exists^* x^\rho A.$$

To accomodate the specific ground types ι and o, we need an induction scheme for both types, and in addition a "truth axiom" .

$$\begin{aligned}
\mathit{Ind}_{p,A}:&\quad A[p := \mathit{true}] \to A[p := \mathit{false}] \to \forall p A\\
\mathit{Ind}_{n,A}:&\quad A[n := 0] \to (\forall n. A \to A[n := n+1]) \to \forall n A\\
T:&\quad \mathit{atom}(\mathit{true})
\end{aligned}$$

We also need $\exists^*$-axioms:

$$\begin{aligned}
\exists^{*+}_{x,A}:&\quad \forall x. A \to \exists^* x A\\
\exists^{*-}_{x,A,B}:&\quad \exists^* x^\rho A \to (\forall x^\rho. A \to B) \to B \qquad (x \notin FV(B))
\end{aligned}$$

Derivations are within minimal logic. They are written in natural deduction style, i.e. as typed λ-terms via the well-known Curry-Howard correspondence.

$$\begin{array}{l}
u^B \quad \text{(assumptions)} \mid \text{axioms} \mid\\
\langle M^A, N^B\rangle^{A\land B} \mid \pi_i(M^{A_0 \land A_1})^{A_i} \mid\\
(\lambda u^A M^B)^{A \to B} \mid (M^{A\to B} N^A)^B \mid\\
(\lambda x^\rho M^A)^{\forall x^\rho A} \mid (M^{\forall x^\rho A} t^\rho)^{A[x^\rho := t^\rho]}
\end{array}$$

where in the $\forall$-introduction $\lambda x M^A$, x must not be free in any B with $u^B \in FA(M)$.

Classical logic

Assume that *atom* is the only predicate symbol, and replace $\bot$ by $\mathit{atom}(\mathit{false})$. Then we have

$$\begin{aligned}
\neg A &:= A \to \mathit{atom}(\mathit{false})\\
\exists x A &:= \neg \forall x \neg A,
\end{aligned}$$

and stability $\neg\neg A \to A$ is provable for all $\exists^*$-free formulas A, using boolean induction for the case $\mathit{atom}(t)$. Hence classical arithmetic (in all finite types) is a subsystem of our present system based on minimal logic.

So for $\exists^*$-free formulas (no restriction from a classical point of view) we have full classical logic.

Note also that for quantifier-free formulas B built from prime formulas of the form $\mathit{atom}(t)$ we can easily construct a boolean term $t_B : o$ such that

$$\mathit{atom}(t_B) \leftrightarrow B$$

is derivable. Therefore we can use boolean induction to derive *case distinction* for quantifier-free formulas.

2.3 Program Extraction from Constructive Proofs

We now come back to the full system, and assign to every formula A an object $\tau(A)$ (a type or the symbol $*$). $\tau(A)$ is intended to be the type of the program to be extracted from a proof of A. In case $\tau(A) = *$ proofs of A have no computational content; such formulas A are called *Harrop formulas.*

$$\begin{aligned}
\tau(P(\vec{s})) &:= * \\
\tau(\exists^* x^\rho A) &:= \begin{cases} \rho & \text{if } \tau(A) = * \\ \rho \times \tau(A) & \text{otherwise} \end{cases} \\
\tau(\forall x^\rho A) &:= \begin{cases} * & \text{if } \tau(A) = * \\ \rho \to \tau(A) & \text{otherwise} \end{cases} \\
\tau(A_0 \wedge A_1) &:= \begin{cases} \tau(A_i) & \text{if } \tau(A_{1-i}) = * \\ \tau(A_0) \times \tau(A_1) & \text{otherwise} \end{cases} \\
\tau(A \to B) &:= \begin{cases} \tau(B) & \text{if } \tau(A) = * \\ * & \text{if } \tau(B) = * \\ \tau(A) \to \tau(B) & \text{otherwise} \end{cases}
\end{aligned}$$

We now define, for a given derivation M of a formula A with $\tau(A) \neq *$, its *extracted program* $[\![M]\!]$ of type $\tau(A)$.

$$\begin{aligned}
[\![u^A]\!] &:= x_u^{\tau(A)} \quad (x_u^{\tau(A)} \text{ uniquely associated with } u^A) \\
[\![\lambda u^A M]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A) = * \\ \lambda x_u^{\tau(A)} [\![M]\!] & \text{otherwise} \end{cases} \\
[\![M^{A \to B} N]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A) = * \\ [\![M]\!][\![N]\!] & \text{otherwise} \end{cases} \\
[\![\langle M_0^{A_0}, M_1^{A_1} \rangle]\!] &:= \begin{cases} [\![M_i]\!] & \text{if } \tau(A_{1-i}) = * \\ \langle [\![M_0]\!], [\![M_1]\!] \rangle & \text{otherwise} \end{cases} \\
[\![\pi_i(M^{A_0 \wedge A_1})]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A_{1-i}) = * \\ \pi_i [\![M]\!] & \text{otherwise} \end{cases} \\
[\![\lambda x^\rho M]\!] &:= \lambda x^\rho [\![M]\!] \\
[\![Mt]\!] &:= [\![M]\!] t
\end{aligned}$$

We also need extracted programs for induction and $\exists^*$-axioms. For the induction scheme $\mathit{Ind}_{n,A}$: $A[n := 0] \to (\forall n. A \to A[n := n+1]) \to \forall n A$ let

$$[\![\mathit{Ind}_{n,A}]\!] := \mathcal{R}_{\iota,\rho} \colon \rho \to (\iota \to \rho \to \rho) \to \iota \to \rho,$$

where $\rho = \tau(A)$. Similarly boolean induction is realized by $\mathcal{R}_{o,\rho}$.

For the $\exists^*$-axioms we set

$$\begin{aligned}
[\![\exists^{*+}_{x^\rho, A}]\!] &:= \begin{cases} \lambda x^\rho x & \text{if } \tau(A) = * \\ \lambda x^\rho \lambda y^{\tau(A)} \langle x, y \rangle & \text{otherwise} \end{cases} \\
[\![\exists^{*-}_{x^\rho, A, B}]\!] &:= \begin{cases} \lambda x^\rho \lambda f^{\rho \to \tau(B)} f x & \text{if } \tau(A) = * \\ \lambda z^{\rho \times \tau(A)} \lambda f^{\rho \to \tau(A) \to \tau(B)}. f \pi_0(z) \pi_1(z) & \text{otherwise} \end{cases}
\end{aligned}$$

For derivations M^A where $\tau(A) = *$ (i.e. A is a Harrop-formula) we define $[\![M]\!] := \varepsilon$ (ε some new symbol). This applies in particular if A is $\exists^*$-free.

Finally we define the notion of *modified realizability.* More precisely, we define formulas r **mr** A, where A is a formula and r is either a term of type $\tau(A)$ if the latter is a type, or the symbol ε if $\tau(A) = *$.

$$\begin{array}{rcl}
\varepsilon\ \mathbf{mr}\ P(\vec{s}) & = & P(\vec{s}) \\
r\ \mathbf{mr}\ (\exists^* x A) & = & \begin{cases} \varepsilon\ \mathbf{mr}\ A[x := r] & \text{if } \tau(A) = * \\ \pi_1(r)\ \mathbf{mr}\ A[x := \pi_0(r)] & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (\forall x A) & = & \begin{cases} \forall x.\varepsilon\ \mathbf{mr}\ A & \text{if } \tau(A) = * \\ \forall x.rx\ \mathbf{mr}\ A & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (A \to B) & = & \begin{cases} \varepsilon\ \mathbf{mr}\ A\ \to\ r\ \mathbf{mr}\ B & \text{if } \tau(A) = * \\ \forall x.x\ \mathbf{mr}\ A\ \to\ \varepsilon\ \mathbf{mr}\ B & \text{if } \tau(A) \neq * = \tau(B) \\ \forall x.x\ \mathbf{mr}\ A\ \to\ rx\ \mathbf{mr}\ B & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (A_0 \wedge A_1) & = & \begin{cases} \varepsilon\ \mathbf{mr}\ A_0\ \wedge\ r\ \mathbf{mr}\ A_1 & \text{if } \tau(A_0) = * \\ r\ \mathbf{mr}\ A_0\ \wedge\ \varepsilon\ \mathbf{mr}\ A_1 & \text{if } \tau(A_1) = * \\ \pi_0(r)\ \mathbf{mr}\ A_0\ \wedge\ \pi_1(r)\ \mathbf{mr}\ A_1 & \text{otherwise} \end{cases}
\end{array}$$

Note that for A $\exists^*$-free we have $\tau(A) = *$ and $\varepsilon\ \mathbf{mr}\ A = A$. For the formulation of the soundness theorem below it will be useful to let $x_u := \varepsilon$ if u^A is an assumption variable with a Harrop-formula A.

Theorem 2.9 (Soundness) *If M is a derivation of a formula B, then there is a derivation $\mu(M)$ of $[\![M]\!]\ \mathbf{mr}\ B$ from $\{\, x_u\ \mathbf{mr}\ C \mid u^C \in FA(M) \,\}$.*

Proof. Induction on M. *Case $u\colon A$.* Then we have $\bar{u}\colon x_u\ \mathbf{mr}\ A$. Let $\mu(u) := \bar{u}$.
Case $\lambda u^A M^B$. We must find a derivation $\mu(\lambda u M)$ of

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B).$$

Subcase $\tau(A) = *$. Then we have $[\![\lambda u M]\!] = [\![M]\!]$, hence

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \varepsilon\ \mathbf{mr}\ A\ \to\ [\![M]\!]\ \mathbf{mr}\ B.$$

By induction hypothesis we can define $\mu(\lambda u M) := \lambda \bar{u} \mu(M)$ with $\bar{u}\colon \varepsilon\ \mathbf{mr}\ A$. *Subcase $\tau(A) \neq * = \tau(B)$.* Then we have $[\![\lambda u M]\!] = \varepsilon$ and

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \forall x.x\ \mathbf{mr}\ A\ \to\ \varepsilon\ \mathbf{mr}\ B.$$

By induction hypothesis define $\mu(\lambda u M) := \lambda x_u \lambda \bar{u} \mu(M)$ with $\bar{u}\colon x_u\ \mathbf{mr}\ A$. *Subcase $\tau(A) \neq * \neq \tau(B)$.* Then we have

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \forall x.x\ \mathbf{mr}\ A\ \to\ [\![\lambda u M]\!] x\ \mathbf{mr}\ B$$

Because of $[\![\lambda u M]\!] = \lambda x_u [\![M]\!]$ and since we identify terms with the same β-normal form, by induction hypothesis we can define $\mu(\lambda u M) := \lambda x_u \lambda \bar{u} \mu(M)$.

Case $M^{A \to B} N^A$. We must find a derivation $\mu(MN)$ of $[\![MN]\!]\ \mathbf{mr}\ B$. *Subcase $\tau(A) = *$.* Then we have $[\![MN]\!] = [\![M]\!]$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \varepsilon\ \mathbf{mr}\ A\ \to\ [\![M]\!]\ \mathbf{mr}\ B$$

and $\mu(N)$ of $\varepsilon\ \mathbf{mr}\ A$; hence we can define $\mu(MN) := \mu(M)\mu(N)$. *Subcase $\tau(A) \neq * = \tau(B)$.* Then we have $[\![MN]\!] = \varepsilon$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \forall x.x\ \mathbf{mr}\ A\ \to\ \varepsilon\ \mathbf{mr}\ B$$

and $\mu(N)$ of $[\![N]\!]\ \mathbf{mr}\ A$; hence we can define $\mu(MN) := \mu(M)[\![N]\!]\mu(N)$. *Subcase $\tau(A) \neq * \neq \tau(B)$.* Then we have $[\![MN]\!] = [\![M]\!][\![N]\!]$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B)\ =\ \forall x.x\ \mathbf{mr}\ A\ \to\ [\![M]\!] x\ \mathbf{mr}\ B$$

and $\mu(N)$ of $[\![N]\!]$ **mr** A; hence we can define $\mu(MN) := \mu(M)[\![N]\!]\mu(N)$.

Case $\langle M_0^{A_0}, M_1^{A_1}\rangle$. We must find a derivation $\mu(\langle M_0, M_1\rangle)$ of

$$[\![\langle M_0, M_1\rangle]\!] \mathbf{ mr } (A_0 \wedge A_1).$$

Subcase $\tau(A_0) = * = \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = \varepsilon$, hence

$$[\![\langle M_0, M_1\rangle]\!] \mathbf{ mr } (A_0 \wedge A_1) \quad = \quad \varepsilon \mathbf{ mr } A_0 \ \wedge \ \varepsilon \mathbf{ mr } A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$. *Subcase* $\tau(A_0) = * \neq \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = [\![M_1]\!]$, hence

$$[\![\langle M_0, M_1\rangle]\!] \mathbf{ mr } (A_0 \wedge A_1) \quad = \quad \varepsilon \mathbf{ mr } A_0 \ \wedge \ [\![M_1]\!] \mathbf{ mr } A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$. *Subcase* $\tau(A_0) \neq * = \tau(A_1)$. Similar. *Subcase* $\tau(A_0) \neq * \neq \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = \langle [\![M_0]\!], [\![M_1]\!]\rangle$, hence

$$[\![\langle M_0, M_1\rangle]\!] \mathbf{ mr } (A_0 \wedge A_1) \quad = \quad [\![M_0]\!] \mathbf{ mr } A_0 \ \wedge \ [\![M_1]\!] \mathbf{ mr } A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_i), \mu(M_{1-i})\rangle$.

Case $\pi_0(M^{A_0 \wedge A_1})$. We must find a derivation $\mu(\pi_0(M))$ of

$$[\![\pi_0(M)]\!] \mathbf{ mr } A_0.$$

Subcase $\tau(A_1) = *$. Then we have $[\![\pi_0(M)]\!] = [\![M]\!]$. by induction hypothesis we have a derivation $\mu(M)$ of

$$[\![M]\!] \mathbf{ mr } (A_0 \wedge A_1) \quad = \quad [\![M]\!] \mathbf{ mr } A_0 \ \wedge \ \varepsilon \mathbf{ mr } A_1.$$

hence we can define $\mu(\pi_0(M)) := \pi_0(\mu(M))$. *Subcase* $\tau(A_0) = * \neq \tau(A_1)$. Then we have $[\![\pi_0(M)]\!] = \varepsilon$. by induction hypothesis we have a derivation $\mu(M)$ of

$$[\![M]\!] \mathbf{ mr } (A_0 \wedge A_1) \quad = \quad \varepsilon \mathbf{ mr } A_0 \ \wedge \ [\![M]\!] \mathbf{ mr } A_1.$$

hence we can define $\mu(\pi_0(M)) := \pi_0(\mu(M))$. *Subcase* $\tau(A_0) \neq * \neq \tau(A_1)$. Similar; we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$.

Case $\pi_1(M^{A_0 \wedge A_1})$. Similar.

Case $\lambda z M^A$. We must find a derivation $\mu(\lambda z M)$ of $[\![\lambda z M]\!]$ **mr** $\forall z A$. By definition $[\![\lambda z M]\!] = \lambda z [\![M]\!]$. *Subcase* $\tau(A) = *$. Then we have

$$\lambda z [\![M]\!] \mathbf{ mr } \forall z A \quad = \quad \forall z . \varepsilon \mathbf{ mr } A$$

and by induction hypothesis we can define $\mu(\lambda z M) := \lambda z \mu(M)$. The variable condition is satisfied, since $\lambda z M^A$ is a derivation term, and hence z does not occur free in any assumption variable $u\colon B$ free in M^A, hence also does not occur free in the free assumption $\bar{u}\colon x_u$ **mr** B. *Subcase* $\tau(A) \neq *$. Then we have

$$\lambda z [\![M]\!] \mathbf{ mr } \forall z A \quad = \quad \forall z . (\lambda z [\![M]\!]) z \mathbf{ mr } A.$$

Since we identify terms with the same β-normal form, by induction hypothesis we again can define $\mu(\lambda z M) := \lambda z \mu(M)$. As before one can see that the variable condition is satisfied.

Case $M^{\forall z A} t$. We must find a derivation $\mu(Mt)$ of $[\![Mt]\!]\ \mathbf{mr}\ A[x := t]$. By definition we have $[\![Mt]\!] = [\![M]\!]t$. *Subcase* $\tau(A) = *$. By induction hypothesis we have a derivation of

$$[\![M]\!]\ \mathbf{mr}\ \forall z A \quad = \quad \forall z.\varepsilon\ \mathbf{mr}\ A$$

hence we can define $\mu(Mt) := \mu(M)t$. *Subcase* $\tau(A) \neq *$. By induction hypothesis we have a derivation of

$$[\![M]\!]\ \mathbf{mr}\ \forall z A \quad = \quad \forall z.[\![M]\!]z\ \mathbf{mr}\ A,$$

hence we again can define $\mu(Mt) := \mu(M)t$.

Case $\exists^{*+}_{x,A}\colon \forall \vec{x} \forall x.A \to \exists^* x^\rho A$. We must find a derivation $\mu(\exists^{*+}_{x,A})$ of

$$[\![\exists^{*+}_{x,A}]\!]\ \mathbf{mr}\ \forall \vec{x} \forall x.A \to \exists^* x^\rho A.$$

Subcase $\tau(A) = *$. Then we have $\exists^{*+}_{x,A} = \lambda\vec{x}\lambda x x$ and we obtain

$$\begin{aligned}(\lambda\vec{x}\lambda x x)\ \mathbf{mr}\ \forall\vec{x}\forall x.A \to \exists^* xA &= \forall\vec{x}\forall x.x\ \mathbf{mr}\ (A \to \exists^* xA)\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to x\ \mathbf{mr}\ \exists^* xA\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ A.\end{aligned}$$

Hence we can define $\mu(\exists^{*+}_{x,A}) = \lambda\vec{x}\lambda x\lambda u u$. *Subcase* $\tau(A) \neq *$. Then we have $\exists^{*+}_{x,A} = \lambda\vec{x}\lambda x\lambda y\langle x, y\rangle$ and we obtain

$$\begin{aligned}&\lambda\vec{x}\lambda x\lambda y\langle x,y\rangle\ \mathbf{mr}\ \forall\vec{x}\forall x.A \to \exists^* xA\\ &= \forall\vec{x}\forall x\forall y.y\ \mathbf{mr}\ A \to \langle x,y\rangle\ \mathbf{mr}\ \exists^* xA\\ &= \forall\vec{x}\forall x\forall y.y\ \mathbf{mr}\ A \to y\ \mathbf{mr}\ A\end{aligned}$$

Hence we can define $\mu(\exists^{*+}_{x,A}) = \lambda\vec{x}\lambda x\lambda y\lambda u u$.

Case $\exists^{*-}_{x,A,B}\colon \forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B]$ with $x \notin FV(B)$. We must find a derivation $\mu(\exists^{*-}_{x,A,B})$ of $\forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B]$. *Subcase* $\tau(A) = * = \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \varepsilon$ and we obtain

$$\begin{aligned}&\varepsilon\ \mathbf{mr}\ \forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B]\\ &= \forall\vec{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \varepsilon\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ (\forall x.A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall x\, \varepsilon\ \mathbf{mr}\ (A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to (\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ B) \to \varepsilon\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\vec{x}\lambda x\lambda u\lambda v.vxu$. *Subcase* $\tau(A) \neq * = \tau(B)$. Then we again have $[\![\exists^{*-}_{x,A,B}]\!] = \varepsilon$ and we obtain

$$\begin{aligned}&\varepsilon\ \mathbf{mr}\ \forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B]\\ &= \forall\vec{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \varepsilon\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\vec{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \varepsilon\ \mathbf{mr}\ (\forall x.A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \forall x\, \varepsilon\ \mathbf{mr}\ (A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \forall x\forall y(y\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ B) \to \varepsilon\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\vec{x}\lambda x\lambda u\lambda v.v\pi_0(x)\pi_1(x)u$. *Subcase* $\tau(A) = * \neq \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \lambda\vec{x}\lambda x\lambda z(zx)$ and we obtain

$$\begin{aligned}&\lambda\vec{x}\lambda x\lambda z(zx)\ \mathbf{mr}\ \forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B]\\ &= \forall\vec{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \lambda z(zx)\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.z\ \mathbf{mr}\ (\forall x.A \to B) \to zx\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.\forall x\ zx\ \mathbf{mr}\ (A \to B) \to zx\ \mathbf{mr}\ B\\ &= \forall\vec{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.(\forall x.\varepsilon\ \mathbf{mr}\ A \to zx\ \mathbf{mr}\ B) \to zx\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\vec{x}\lambda x\lambda u\lambda v.vxu$. *Subcase* $\tau(A) \neq * \neq \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \lambda\vec{x}\lambda x\lambda z(z\pi_0(x)\pi_1(x))$ and we obtain

$$\begin{aligned}
&\lambda\vec{x}\lambda x\lambda z(z\pi_0(x)\pi_1(x))\ \mathbf{mr}\ \forall\vec{x}[\exists^* xA \to (\forall x.A \to B) \to B] \\
&= \forall\vec{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \lambda z(z\pi_0(x)\pi_1(x))\ \mathbf{mr}\ ((\forall x.A \to B) \to B) \\
&= \forall\vec{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \\
&\qquad \forall z.z\ \mathbf{mr}\ (\forall x.A \to B) \to z\pi_0(x)\pi_1(x)\ \mathbf{mr}\ B \\
&= \forall\vec{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \\
&\qquad \forall z.\forall x\forall y(y\ \mathbf{mr}\ A \to zxy\ \mathbf{mr}\ B) \to z\pi_0(x)\pi_1(x)\ \mathbf{mr}\ B.
\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\vec{x}\lambda x\lambda u\lambda z\lambda v.v\pi_0(x)\pi_1(x)u$. □

If B is $\exists^*$-free, then $\varepsilon\ \mathbf{mr}\ B = B$. Hence for $\forall x^\rho\exists^* y^\sigma B$ with $\exists^*$-free B we have $\tau(\forall x\exists y^* B) = \rho \to \sigma$ and

$$t\ \mathbf{mr}\ \forall x\exists^* yB = \forall xB[y := tx].$$

Then as a corollary to the soundness theorem we obtain the extraction theorem

Theorem 2.10 *From a derivation* $M{:}\forall x^\rho\exists^* y^\sigma B$ *with* B $\exists^*$*-free from* $\exists^*$*-free assumptions* Γ *one can extract a closed term* $[\![M]\!]^{\rho\to\sigma}$ *such that the formula* $\forall xB[y := [\![M]\!]x]$ *is provable from* Γ.

Here Γ should be viewed as lemmata, i.e. true formulas (proved separately, to keep M short). The theorem says that the extracted program is independent of how this shortcut is achieved.

3 Computational Content of Classical Proofs

3.1 Definite and Goal Formulas

For simplicity we only treat formulas in the $\to\forall$ fragment; this is not an essential restriction, since conjunctions can be easily removed.

A formula is *relevant* if it "ends" with $\bot$. More precisely, relevant formulas are defined inductively by the clauses

- $\bot$ is relevant,
- if C is relevant and B is an arbitrary formula, then $B \to C$ is relevant, and
- if C is relevant, then $\forall xC$ is relevant.

A formula which is not relevant is called *irrelevant*.

We define *goal formulas* G and *definite formulas* D inductively. These notions are related to similar ones common under the same name in the context of extensions of logic programming. Let P range over atoms (including $\bot$).

$$\begin{array}{lllll}
G & := & P & \mid D \to G & \text{provided } D \text{ irrelevant} \Rightarrow D \text{ quantifier-free} \\
 & & & \mid \forall xG & \text{provided } G \text{ irrelevant} \\
D & := & P & \mid G \to D & \text{provided } D \text{ irrelevant} \Rightarrow G \text{ irrelevant} \\
 & & & \mid \forall xD &
\end{array}$$

Lemma 3.1 *For definite formulas D, relevant goal formulas G and an arbitrary formula A we have derivations*

$$\begin{aligned} N_D^A &: D \to D^A \\ H_G^A &: G^A \to \neg G \to A \end{aligned}$$

The proof is by induction on definite and goal formulas; it is necessary to prove several additional claims simultaneously, to get the induction through. For details we refer to [1].

Theorem 3.2 *Let $D_1, \dots, D_l$ be relevant and $D_{l+1}, \dots, D_n$ irrelevant definite formulas, and $G_1, \dots, G_k$ be relevant quantifier-free and $G_{k+1}, \dots, G_m$ irrelevant goal formulas. Let $A := \exists^* y \vec{G}$ and*

$$t := \lambda y, \vec{x}.\mathbf{if}\ \neg G_1\ \mathbf{then}\ [\![H_{G_1}^A]\!](x_1)\ \mathbf{else}\ \dots\ \mathbf{if}\ \neg G_k\ \mathbf{then}\ [\![H_{G_k}^A]\!](x_k)\ \mathbf{else}\ y.$$

Assume that we have a derivation

$$M \colon \vec{D} \to (\forall y.\vec{G} \to \bot) \to \bot.$$

Let M^A denote the result of substituting $\bot$ by A everywhere in M. Then we can derive

$$\vec{D} \to [\![M^A]\!][\![N_{D_1}^A]\!] \dots [\![N_{D_l}^A]\!] t\ \mathbf{mr}\ \exists^* y \vec{G}.$$

The proof uses the soundness theorem 2.9 and lemma 3.1. For details we again have to refer to [1].

3.2 Example: Integer Square Roots

Let $f \colon \iota \to \iota$ be unbounded with $f0 = 0$; think of f as the square function $f(n) = n^2$. We want to prove that integer square roots always exist, or more precisely a general form of this proposition, using only the two properties of f above. We specify our problem as follows. Here f, g, n are parameters; g is used as an explicit witness of the unboundedness of f.

$$\forall n\, \neg n {<} f0 \to \forall n\, n {<} f(gn) \to (\forall m. \neg n {<} fm \to n {<} f(m+1) \to \bot) \to \bot$$

Clearly

$\forall n\, \neg n {<} f0$	is a relevant definite formula
$\forall n\, n {<} f(gn)$	is an irrelevant definite formula
$\neg n {<} fm$	is a relevant goal formula
$n {<} f(m+1)$	is an irrelevant goal formula,

hence theorem 3.2 can be applied without inserting any double negations.

To construct a derivation, assume

$$\begin{aligned} v_1 &: \forall n\, \neg n {<} f0, \\ v_2 &: \forall n\, n {<} f(gn), \\ u &: \forall m. \neg n {<} fm \to n {<} f(m+1) \to \bot. \end{aligned}$$

Our goal is $\bot$. From v_1 and u we inductively get $\forall m\, \neg n {<} fm$. For $m := gn$ this yields a contradiction to v_2. The derivation is

$$\dfrac{\dfrac{\dfrac{\mathit{Ind} \qquad \neg n {<} f0 \qquad \forall m. \neg n {<} fm \to \neg n {<} f(m+1)}{\forall m \neg n {<} fm} \qquad gn}{\neg n {<} f(gn)} \qquad n {<} f(gn)}{\bot}$$

with derivation term

$$M := \lambda v_1 \lambda v_2 \lambda u. Ind_{m,\neg n<fm} n(v_1 n)u(gn)(v_2 n),$$

Now let $A := \exists^* m.\neg n<fm \wedge n<f(m+1)$, hence

$$\begin{aligned} M^A = \ & \lambda v_1^{\forall n.n<f0 \to A} \lambda v_2^{\forall n\, n<f(gn)} \lambda u^{\forall m.(n<fm \to A) \to n<f(m+1) \to A}. \\ & Ind_{m,n<fm \to A} n(v_1 n)u(gn)(v_2 n) \end{aligned}$$

and therefore

$$[\![M^A]\!] \ =_\beta \lambda x_{v_1}^{\iota\to\iota} \lambda x_u^{\iota\to\iota\to\iota}.\mathcal{R}_\iota(x_{v_1} n)x_u(gn)$$

From the proof of theorem 3.2 we generally know $D \to [\![N_D^A]\!]$ **mr** D^A for every relevant definite formula D. In our case, with $D = \forall n.\neg n<f0$, we can derive directly

$$(\forall n.n<f0 \to \bot) \to (\lambda n0) \ \mathbf{mr} \ \forall n.n<f0 \to A,$$

since we can use ex-falso. So we may assume $[\![N_D^A]\!] = \lambda n0$. Also, from the proof of theorem 3.2 we generally know x **mr** $G^A \to \neg G \to [\![H_G^A]\!](x)$ **mr** A for every relevant goal formula G. In our case, with $G = \neg n<fm$, we can derive directly

$$(x \ \mathbf{mr} \ n<fm \to A) \to \neg\neg n<fm \to x \ \mathbf{mr} \ A.$$

So we may assume $[\![H_G^A]\!] = \lambda xx$. Now let

$$\begin{aligned} t: \ &= \lambda m \lambda x.\mathbf{if} \ \neg G_1 \ \mathbf{then} \ [\![H_{G_1}^A]\!](x) \ \mathbf{else} \ m \\ &= \lambda m \lambda x.\mathbf{if} \ n<fm \ \mathbf{then} \ x \ \mathbf{else} \ m. \end{aligned}$$

Then the extracted term according to theorem 3.2 is

$$\begin{aligned} [\![M^A]\!][\![N_D^A]\!]t \ &= \Big(\lambda x_{v_1}^{\iota\to\iota} \lambda x_u^{\iota\to\iota\to\iota}.\mathcal{R}_\iota(x_{v_1} n)x_u(gn)\Big)(\lambda n0)t \\ &=_\beta \mathcal{R}_\iota 0t(gn) \\ &= \mathcal{R}_\iota 0(\lambda n \lambda x \mathbf{if} \ n<fm \ \mathbf{then} \ x \ \mathbf{else} \ m)(gn). \end{aligned}$$

Informally, the algorithm can be written in the form $h(gn)$ where $h: \iota \to \iota$ is such that

$$\begin{aligned} & h(0) = 0, \\ & h(m+1) = \begin{cases} h(m) & \text{if } n<f(m) \\ m & \text{else} \end{cases} \end{aligned}$$

3.3 Example: The Greatest Common Divisor

Yiannis Moschovakis suggested the following example of a classical existence proof with a quantifier-free kernel which does not obviously contain an algorithm: the gcd of two natural numbers a_1 and a_2 is a linear combination of the two. Here we treat that example as a case study for program extraction from classical proofs. In the original treatment in [2] we identified one of the predicate symbols as critical (the divisibility relation $\cdot|\cdot$). Here we show how this example can be treated in the present form of the theory, with definite and goal formulas. It turns out that the algorithm does not change; however, the method of how to extract a program from this classical proof becomes more perspicious.

Let $a, b, c, i, j, k, \ell, m, n, q, r$ denote natural numbers. Our language is determined by the constants $0, 1, +, *$, function symbols for the quotient and the remainder denoted by $q(a,c)$ and $r(a,c)$, a 4-ary function denoted by $abs(k_1a_1 - k_2a_2)$ whose intended meaning is clear from the notation and an auxiliary 5-ary function f which will be defined later. We will express the intended meaning of these function symbols by stating some properties (lemmata) $v_1, \dots, v_6$ of them; these will be formulated as we need them.

Theorem 3.3 *For natural numbers a_1, a_2 with $0 < a_2$*

$$\exists k_1, k_2 . abs(k_1a_1 - k_2a_2) | a_1 \wedge abs(k_1a_1 - k_2a_2) | a_2 \wedge 0 < abs(k_1a_1 - k_2a_2).$$

Proof. Let a_1, a_2 be given and assume $0 < a_2$. The ideal (a_1, a_2) generated from a_1, a_2 has a least positive element c, since $0 < a_2$. This element has a representation $c = abs(k_1a_1 - k_2a_2)$ with $k_1, k_2 \in \mathbb{N}$. It is a common divisor of a_1 and a_2 since otherwise the remainder $r(a_i, c)$ would be a smaller positive element of the ideal.

The number $c \in (a_1, a_2)$ dividing a_1 and a_2 is the greatest common divisor since any common divisor of a_1 and a_2 must also be a divisor of c. □

The minimum principle and course-of-values induction.

In order to formally write out the proof above we need to make explicit the instance of the induction scheme used implicitly in the minimum principle. The minimum principle w.r.t. a measure μ says

$$\exists \vec{k} R(\vec{k}) \to \exists \vec{k} . R(\vec{k}) \wedge \forall \vec{\ell} . \mu(\vec{\ell}) < \mu(\vec{k}) \to R(\vec{\ell}) \to \bot$$

(in our example $R(k_1, k_2) \equiv 0 < abs(k_1a_1 - k_2a_2)$ and $\mu(k_1, k_2) \equiv abs(k_1a_1 - k_2a_2)$). In order to reduce this to the induction scheme we use the fact that the formula above is classically equivalent to

$$(\forall \vec{k} . R(\vec{k}) \to [\forall \vec{\ell} . \mu(\vec{\ell}) < \mu(\vec{k}) \to R(\vec{\ell}) \to \bot] \to \bot) \to \forall \vec{k} . R(\vec{k}) \to \bot,$$

i.e. the principle of *course-of-values induction* for $R(\vec{k}) \to \bot$ w.r.t. the measure $\mu(\vec{k})$. We can write this as

$$Prog \to \forall \vec{k} . R(\vec{k}) \to \bot,$$

where

$$Prog := \forall \vec{k} . [\forall \vec{\ell} . \mu(\vec{\ell}) < \mu(\vec{k}) \to R(\vec{\ell}) \to \bot] \to R(\vec{k}) \to \bot.$$

In the formal treatment of our example it will be more convenient to use the minimum principle in the form of course-of-values induction.

To prove course-of-values induction assume $w_1 : Prog$ and prove $\forall \vec{k} . R(\vec{k}) \to \bot$. This is achieved by proving $\forall n B$, where

$$B := \forall \vec{k} . \mu(\vec{k}) < n \to R(\vec{k}) \to \bot,$$

and using B with $n := \mu(\vec{k}) + 1$. We prove $\forall n B$ by (zero-successor) induction.

Base. $B[n := 0]$ follows easily from the lemma

$$v_1 : \forall m . m < 0 \to \bot.$$

Step. Let n be given and assume $w_2 : B$. To show $B[n := n+1]$ let $\vec{k}$ be given and assume $w_3 : \mu(\vec{k}) < n + 1$. We will derive $R(\vec{k}) \to \bot$ by using $w_1 : Prog$ at $\vec{k}$. Hence we have to prove

$$\forall \vec{\ell} . \mu(\vec{\ell}) < \mu(\vec{k}) \to R(\vec{\ell}) \to \bot.$$

So, let $\vec{\ell}$ be given and assume further $w_4 : \mu(\vec{\ell}) < \mu(\vec{k})$. From w_4 and $w_3 : \mu(\vec{k}) < n + 1$ we infer $\mu(\vec{\ell}) < n$ (using an arithmetical lemma). Hence, by induction hypothesis $w_2 : B$ at $\vec{\ell}$ we get $R(\vec{\ell}) \to \bot$.

Detailed proof of the theorem

Now we repeat the proof of the theorem in some more detail using course-of-values induction. As always in classical logic, we may view the proof as an indirect one, deriving a contradiction from the assumption that the claim is false. So let a_1, a_2 be given and assume $v_0\colon 0 < a_2$ and

$$\begin{aligned} &u\colon \forall k_1, k_2. abs(k_1a_1 - k_2a_2)|a_1 \to abs(k_1a_1 - k_2a_2)|a_2 \to \\ &\quad 0 < abs(k_1a_1 - k_2a_2) \to \bot. \end{aligned}$$

We have to prove $\bot$ which will be achieved by proving $\forall k_1, k_2. 0 < abs(k_1a_1 - k_2a_2) \to \bot$ by course-of-values induction and then specializing this formula to $k_1, k_2 = 0, 1$ and using the assumption $v_0\colon 0 < a_2$ $(= abs(0a_1 - 1a_2))$.

The principle of course-of-values induction is used with

$$N(k_1, k_2) := 0 < abs(k_1a_1 - k_2a_2) \to \bot \quad \text{and} \quad \mu(k_1, k_2) := abs(k_1a_1 - k_2a_2).$$

We have to show that N is progressive. To this end let k_1, k_2 be given and assume

$$u_1\colon \forall \ell_1, \ell_2. \mu(\ell_1, \ell_2) < \mu(k_1, k_2) \to N(\ell_1, \ell_2).$$

We have to prove $N(k_1, k_2)$. So, assume $u_2\colon 0 < \mu(k_1, k_2)$. We have to show $\bot$. This will be achieved by using the (false) assumption u at k_1, k_2. We have to prove $\mu(k_1, k_2)|a_1$ and $\mu(k_1, k_2)|a_2$. Informally, one would argue "if, say, $\mu(k_1, k_2)\not| a_1$ then the remainder $r_1 := r(a_1, \mu(k_1, k_2))$ is positive and less than $\mu(k_1, k_2)$. Furthermore we can find ℓ_1, ℓ_2 such that $r_1 = \mu(\ell_1, \ell_2)$. Altogether this contradicts the assumption u_1". More formally, to prove $\mu(k_1, k_2)|a_1$ we use the lemma

$$v_2\colon \forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to c|a$$

at a_1, $q_1 := q(a_1, \mu(k_1, k_2))$ (the quotient), $\mu(k_1, k_2)$ and r_1. We have to prove the premises

$$a_1 = q_1\mu(k_1, k_2) + r_1 \quad \text{and} \quad 0 < r_1 \to \bot$$

of the instantiated lemma v_2. Here we need the lemmata

$$\begin{aligned} &v_3\colon \forall a, c. 0 < c \to a = q(a, c)c + r(a, c), \\ &v_4\colon \forall a, c. 0 < c \to r(a, c) < c \end{aligned}$$

specifying the functions quotient and remainder. Now the first premise follows immediately from lemma v_3 and $u_2\colon 0 < \mu(k_1, k_2)$. To prove the second premise, $0 < r_1 \to \bot$, we assume $u_3\colon 0 < r_1$ and show $\bot$. First we compute ℓ_1, ℓ_2 such that $r_1 = \mu(\ell_1, \ell_2)$. This is done by some auxiliary function f, defined by

$$f(a_1, a_2, k_1, k_2, q) := \begin{cases} qk_1 - 1, & \text{if } k_2a_2 < k_1a_1 \text{ and } 0 < q; \\ qk_1 + 1, & \text{otherwise.} \end{cases}$$

f satisfies the lemma

$$v_5\colon \forall a_1, a_2, k_1, k_2, q, r. a_1 = q \cdot \mu(k_1, k_2) + r \to r = \mu(f(a_1, a_2, k_1, k_2, q), qk_2).$$

Hence we let $\ell_1 := f(a_1, a_2, k_1, k_2, q_1)$ and $\ell_2 := q_1k_2$. Now we have $\mu(\ell_1, \ell_2) = r_1 < \mu(k_1, k_2)$ by v_5, u_2 and v_4, as well as $0 < r_1 = \mu(\ell_1, \ell_2)$ by u_3 and v_5. Therefore, we get $\bot$ by u_1 at ℓ_1, ℓ_2 (using some equality lemmata). This completes the proof of $\mu(k_1, k_2)|a_1$. $\mu(k_1, k_2)|a_2$ is proved similarly using the lemma

$$v_6\colon \forall a_1, a_2, k_1, k_2, q, r. a_2 = q \cdot \mu(k_1, k_2) + r \to r = \mu(qk_1, f(a_2, a_1, k_2, k_1, q)).$$

Turning the lemmata and the claim into definite and goal formulas

In order to make our general theory applicable to the present example we have to make sure that the lemmata assert definite formulas and the claim is given by goal formulas. This is almost the case already, with the sole exeption of

$$v_2\colon \forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to c|a,$$

which is not definite (we have concluded something positive, i.e. $c|a$, from something negative, i.e. $0 < r \to \bot$). We now make it definite by inserting a double negation in front of the final conclusion $c|a$, i.e. we replace the formula of v_2 by

$$\forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to (c|a \to \bot) \to \bot.$$

Clearly this will not affect our proof in any essential way; however, it will make a difference when it comes to extract the program.

Formal proof

The proof of the principle of course-of-values induction and the proof of the theorem were given in such a detail that it is now easy to formalize them completely. Only some arguments concerning $<$ and $=$ were left implicit, but since these will be irrelevant formulas we don't need to worry about these omissions.

$$\begin{aligned}
M := \;& \lambda v_0^{0<a_2} \\
& \lambda v_1^{\forall m. m<0\to\bot} \\
& \lambda v_2^{\forall a,q,c,r.a=qc+r\to(0<r\to\bot)\to(c|a\to\bot)\to\bot} \\
& \lambda v_3^{\forall a,c.0<c\to a=q(a,c)c+r(a,c)} \\
& \lambda v_4^{\forall a,c.0<c\to r(a,c)<c} \\
& \lambda v_5^{\forall a_1,a_2,k_1,k_2,q,r.a_1=q\cdot\mu(k_1,k_2)+r\to r=\mu(f(a_1,a_2,k_1,k_2,q),qk_2)} \\
& \lambda v_6^{\forall a_1,a_2,k_1,k_2,q,r.a_2=q\cdot\mu(k_1,k_2)+r\to r=\mu(qk_1,f(a_2,a_1,k_2,k_1,q))} \\
& \lambda u^{\forall\vec{k}.\mu(\vec{k})|a_1\to\mu(\vec{k})|a_2\to 0<\mu(\vec{k})\to\bot}. \\
& M_{\mathrm{cvind}}^{Prog\to\forall\vec{k}.0<\mu(\vec{k})\to\bot} M_{\mathrm{prog}} 0 1 (L^{0<\mu(0,1)}[v_0])
\end{aligned}$$

where

$$\begin{aligned}
M_{\mathrm{cvind}} &= \lambda w_1^{Prog} \lambda\vec{k}. Ind_{n,B} M_{\mathrm{base}} M_{\mathrm{step}} (\mu(\vec{k}) + 1)\vec{k} L^{\mu(\vec{k})<\mu(\vec{k})+1}, \\
M_{\mathrm{base}} &= \lambda\vec{k}\lambda w_0^{\mu(\vec{k})<0} \lambda \tilde{w}_0{}^{0<\mu(\vec{k})}. v_1 \mu(\vec{k}) w_0, \\
M_{\mathrm{step}} &= \lambda n \lambda w_2^B \lambda\vec{k}\lambda w_3^{\mu(\vec{k})<n+1}. w_1 \vec{k} (\lambda\vec{\ell}\lambda w_4^{\mu(\vec{\ell})<\mu(\vec{k})}. w_2 \vec{\ell} (L^{\mu(\vec{\ell})<n}[w_4, w_3])),
\end{aligned}$$

$$\begin{aligned}
M_{\mathrm{prog}} &= \lambda\vec{k}\lambda u_1^{\forall\vec{\ell}.\mu(\vec{\ell})<\mu(\vec{k})\to 0<\mu(\vec{\ell})\to\bot} \lambda u_2^{0<\mu(\vec{k})}. \\
&\qquad M_{\mathrm{div}_2}^{(\mu(\vec{k})|a_2\to\bot)\to\bot} \lambda u_3^{\mu(\vec{k})|a_2}. M_{\mathrm{div}_1}^{(\mu(\vec{k})|a_1\to\bot)\to\bot} \lambda u_4^{\mu(\vec{k})|a_1}. u\vec{k} u_4 u_3 u_2, \\
M_{\mathrm{div}_i} &= v_2 a_i q_i(\vec{k}) \mu(\vec{k}) r_i(\vec{k}) (L^{a_i=q_i(\vec{k})\mu(\vec{k})+r_i(\vec{k})}[v_3, u_2]) M_{\not<_i}^{0<r_i(\vec{k})\to\bot}, \\
M_{\not<_i} &= \lambda u_{3,i}^{0<r_i(\vec{k})}. u_1 \vec{\ell}_i(\vec{k}) \\
&\qquad (L^{\mu(\vec{\ell}_i(\vec{k}))<\mu(\vec{k})}[v_{4+i}, v_3, u_2, v_4]) (L^{0<\mu(\vec{\ell}_i(\vec{k}))}[u_{3,i}, v_{4+i}, v_3, u_2]).
\end{aligned}$$

Here we have used the abbreviations

$$\begin{aligned}
Prog &= \forall\vec{k}.[\forall\vec{\ell}.\mu(\vec{\ell}) < \mu(\vec{k}) \to 0 < \mu(\vec{\ell}) \to \bot] \to 0 < \mu(\vec{k}) \to \bot \\
B &= \forall\vec{k}.\mu(\vec{k}) < n \to 0 < \mu(\vec{k}) \to \bot
\end{aligned}$$

Term extraction

Let

$$A := \exists^* \vec{k}.\mu(\vec{k})|a_1 \wedge \mu(\vec{k})|a_2 \wedge 0 < \mu(\vec{k}),$$

so $\tau(A) = \vec{\iota} := \iota \times \iota$. Let $C^A := C[\bot := A]$, hence

$$\begin{aligned} &\tau(\mathit{Prog}^A) = \vec{\iota} \to (\vec{\iota} \to \vec{\iota}) \to \vec{\iota}, \\ &\tau(B^A) = \vec{\iota} \to \vec{\iota}. \end{aligned}$$

Furthermore recall $[\![\mathit{Ind}_{n,B^A}]\!] = \mathcal{R}_{\iota,\vec{\iota}\to\vec{\iota}}$ where

$$\mathcal{R}_{\iota,\vec{\iota}\to\vec{\iota}}\colon (\vec{\iota} \to \vec{\iota}) \to (\iota \to (\vec{\iota} \to \vec{\iota}) \to (\vec{\iota} \to \vec{\iota})) \to \iota \to (\vec{\iota} \to \vec{\iota})$$

Note that $\mathcal{R}_{\iota,\vec{\iota}\to\vec{\iota}}$ may be viewed as simultaneous primitive recursion operators $\vec{\mathcal{R}} = (\mathcal{R}_1, \mathcal{R}_2)$ with

$$\begin{aligned} &\mathcal{R}_i \vec{y} \vec{f} 0 = y_i, \\ &\mathcal{R}_i \vec{y} \vec{f}(z+1) = f_i z (\mathcal{R}_1 \vec{y} \vec{f} z)(\mathcal{R}_2 \vec{y} \vec{f} z). \end{aligned}$$

Now we are prepared to compute the extracted term. Let M^A denote the result of replacing every formula C in the derivation M by C^A (note that M^A generally does *not* derive A).

$$[\![M^A]\!] = \lambda x_{v_1}^{\iota\to\vec{\iota}} \lambda x_{v_2}^{\iota^4\to\vec{\iota}\to\vec{\iota}\to\vec{\iota}} \lambda x_u^{\vec{\iota}\to\vec{\iota}}.[\![M^A_{\text{cvind}}]\!][\![M^A_{\text{prog}}]\!]01$$

where

$$\begin{aligned} [\![M^A_{\text{cvind}}]\!] &= \lambda x_{w_1}^{\vec{\iota}\to(\vec{\iota}\to\vec{\iota})\to\vec{\iota}} \lambda \vec{k}.\mathcal{R}[\![M^A_{\text{base}}]\!][\![M^A_{\text{step}}]\!](\mu(\vec{k})+1)\vec{k} \\ [\![M^A_{\text{base}}]\!] &= \lambda \vec{k}.x_{v_1}\mu(\vec{k}) \\ [\![M^A_{\text{step}}]\!] &= \lambda n \lambda x_{w_2}^{\vec{\iota}\to\vec{\iota}} \lambda \vec{k}.x_{w_1}\vec{k}(\lambda \vec{\ell}.x_{w_2}\vec{\ell}), \\ \\ [\![M^A_{\text{prog}}]\!] &= \lambda \vec{k} \lambda x_{u_1}^{\vec{\iota}\to\vec{\iota}}.[\![M^A_{\text{div}_2}]\!]([\![M^A_{\text{div}_1}]\!](x_u\vec{k})) \\ [\![M^A_{\text{div}_i}]\!] &= x_{v_2} a_i q_i(\vec{k})\mu(\vec{k}) r_i(\vec{k})[\![M^A_{\not< i}]\!] \\ [\![M^A_{\not< i}]\!] &= x_{u_1}\vec{\ell}_i(\vec{k}). \end{aligned}$$

From the proof of theorem 3.2 we generally know $D \to [\![N^A_D]\!]$ **mr** D^A for every relevant definite formula D. In our case for $D_1 = \forall m.m < 0 \to \bot$ we can derive directly

$$(\forall m.m < 0 \to \bot) \to (\lambda n 0)\ \mathbf{mr}\ \forall m.m < 0 \to A,$$

since we can use ex-falso. So we may assume $[\![N^A_{D_1}]\!] = \lambda n 0$. Similarly for

$$\begin{aligned} D_2 &= \forall a, q, c, r.a = qc + r \to (0 < r \to \bot) \to (c|a \to \bot) \to \bot \qquad \text{we have} \\ D_2^A &= \forall a, q, c, r.a = qc + r \to (0 < r \to A) \to (c|a \to A) \to A. \end{aligned}$$

Now an informal proof of $D_2 \to D_2^A$ runs as follows. Assume the hypotheses. We prove A by cases on $0 < r$. In case $0 < r$ the hypothesis $0 < r \to A$ yields A. In case $0 < r \to \bot$ we obtain $(c|a \to \bot) \to \bot$ and hence $c|a$. Now the hypothesis $c|a \to A$ yields A.

From this proof we can derive directly

$$D_2 \to (\lambda a \lambda q \lambda c \lambda r \lambda \vec{x} \lambda \vec{y}.\mathbf{if}\ 0 < r\ \mathbf{then}\ \vec{x}\ \mathbf{else}\ \vec{y})\ \mathbf{mr}\ D_2^A.$$

So we may assume $[\![N^A_{D_2}]\!] = \lambda a \lambda q \lambda c \lambda r \lambda \vec{x} \lambda \vec{y}.\mathbf{if}\ 0 < r\ \mathbf{then}\ \vec{x}\ \mathbf{else}\ \vec{y}$.

Moreover, there is no relevant goal formula, hence $t = \lambda\vec{y}\vec{y}$. Then the extracted term according to theorem 3.2 is

$$[\![M^A]\!][\![N^A_{D_1}]\!][\![N^A_{D_2}]\!]t =_\beta [\![M^A_{\mathrm{cvind}}]\!]'[\![M^A_{\mathrm{prog}}]\!]'01$$

where $'$ indicates substitution of $[\![N^A_{D_1}]\!]$, $[\![N^A_{D_2}]\!]$, t for x_{v_1}, x_{v_2}, x_u, so

$$\begin{aligned}
[\![M^A_{\mathrm{cvind}}]\!]' &=_{\beta\eta} \lambda x_{w_1}^{\vec{\iota}\to(\vec{\iota}\to\vec{\iota})\to\vec{\iota}}\lambda\vec{k}'.\mathcal{R}(\lambda\vec{k}.\vec{0})(\lambda n\lambda x_{w_2}\lambda\vec{k}.x_{w_1}\vec{k}x_{w_2})(\mu(\vec{k}')+1)\vec{k}',\\
[\![M^A_{\mathrm{prog}}]\!]' &=_\beta \lambda\vec{k}\lambda x_{u_1}^{\vec{\iota}\to\vec{\iota}}.[\![M^A_{\mathrm{div}_2}]\!]'([\![M^A_{\mathrm{div}_1}]\!]'\vec{k}),\\
[\![M^A_{\mathrm{div}_i}]\!]' &=_\beta (\lambda\vec{x}\lambda\vec{y}.\textbf{if } 0 < \vec{r}_i(\vec{k}) \textbf{ then } \vec{x} \textbf{ else } \vec{y})(x_{u_1}\vec{\ell}_i(\vec{k}))
\end{aligned}$$

and hence

$$\begin{aligned}
[\![M^A_{\mathrm{prog}}]\!]' =_\beta \lambda\vec{k}\lambda x_{u_1}.\ & \textbf{if } 0 < r_2(\vec{k}) \textbf{ then } x_{u_1}\vec{\ell}_2(\vec{k}) \textbf{ else}\\
(\ & \textbf{if } 0 < r_1(\vec{k}) \textbf{ then } x_{u_1}\vec{\ell}_1(\vec{k}) \textbf{ else } \vec{k})
\end{aligned}$$

Therefore we obtain as extracted algorithm, using the fact that $\mu(0,1) = a_2$,

$$\begin{aligned}
[\![M^A]\!][\![N^A_{D_1}]\!][\![N^A_{D_2}]\!]t =_\beta\ & \mathcal{R}(\lambda\vec{k}.\vec{0})\\
& (\lambda n\lambda x_{w_2}^{\vec{\iota}\to\vec{\iota}}\lambda\vec{k}.\textbf{if } 0 < r_2(\vec{k}) \textbf{ then } x_{w_2}\vec{\ell}_2(\vec{k}) \textbf{ else}\\
& \qquad\qquad \textbf{if } 0 < r_1(\vec{k}) \textbf{ then } x_{w_2}\vec{\ell}_1(\vec{k}) \textbf{ else } \vec{k})\\
& (a_2+1)01
\end{aligned}$$

To make this algorithm more readable we may write $[\![M^A]\!][\![N^A_{D_1}]\!][\![N^A_{D_2}]\!]t = h(a_2+1,0,1)$, where

$$\begin{aligned}
h(0,\vec{k}) &:= \vec{0},\\
h(n+1,\vec{k}) &:= \textbf{if } 0 < r_2(\vec{k}) \textbf{ then } h(n,\vec{\ell}_2(\vec{k})) \textbf{ else}\\
&\quad\ (\textbf{if } 0 < r_1(\vec{k}) \textbf{ then } h(n,\vec{\ell}_1(\vec{k})) \textbf{ else } \vec{k})
\end{aligned}$$

Example

Let us use the extracted algorithm to compute coefficients k_1, k_2 such that $\gcd(66, 27) = |k_1 \cdot 66 - k_2 \cdot 27|$.

$$
\begin{array}{ll}
h(28,0,1) & \mu(0,1) = 27 \\
 & 0 < r_1 = 12 \quad q_1 = 2 \\
 & \underbrace{q_1k_1}_{0} \pm 1, \quad \underbrace{q_1k_2}_{2} \qquad -1, \text{ if } k_2a_2 < \underbrace{k_1a_1}_{0} \quad \text{No} \\
 & 1, \quad 2 \\
h(27,1,2) & \mu(1,2) = 12 \\
 & 0 < r_1 = 6 \quad q_1 = 5 \\
 & \underbrace{q_1k_1}_{5\cdot 1} \pm 1, \quad \underbrace{q_1k_2}_{5\cdot 2} \qquad -1, \text{ if } \underbrace{k_2a_2}_{2\cdot 27} < \underbrace{k_1a_1}_{1\cdot 66} \quad \text{Yes} \\
 & 4, \quad 10 \\
h(26,4,10) & \mu(4,10) = |4 \cdot 66 - 10 \cdot 27| = |264 - 270| = 6 \\
 & 6 | 66 \\
 & 0 < r_2 = 3 \quad q_2 = 4 \\
 & \underbrace{q_2k_1}_{4\cdot 4}, \quad \underbrace{q_2k_2}_{4\cdot 10} \pm 1 \qquad -1, \text{ if } \underbrace{k_1a_1}_{4\cdot 66 = 264} < \underbrace{k_2a_2}_{10\cdot 27 = 270} \quad \text{Yes} \\
 & 16, \quad 39 \\
h(25,16,39) & \mu(16,39) = |16 \cdot 66 - 39 \cdot 27| = |1056 - 1053| = 3 \\
 & 3 | 66 \\
 & 3 | 27 \\
 & \text{Result: } 16, 39
\end{array}
$$

Note that, although $3 = |16 \cdot 66 - 39 \cdot 27|$ is the least positive element of the ideal $(66, 27)$, the coefficients 16, 39 are not minimal. The minimal coefficients are 2, 5.

Remarks

1. As one sees from this example the recursion parameter n is not really used in the computation but just serves as a counter or more precisely as an upper bound for the number of steps until both remainders are zero. This will always happen if the induction principle is used only in the form of the minimum principle (or, equivalently, course-of-values induction). Because then in the extracted terms of course-of-values induction, the step $[\![M^A_{\text{step}}]\!] = \lambda n \lambda x^{\vec\iota \to \vec\iota}_{w_2} \lambda \vec{k}. x_{w_1} \vec{k}(\lambda \vec{\ell}. x_{w_2} \vec{\ell})$ has in its kernel no free occurrence of n.

2. If one removes n according to the previous remark it becomes clear that our gcd algorithm is similar to Euclid's. The only difference lies in the fact that we have kept a_1, a_2 fixed in our proof whereas Euclid changes a_1 to a_2 and a_2 to $r(a_1, a_2)$ provided $r(a_1, a_2) > 0$ (using the fact that this doesn't change the ideal).

Implementation

The gcd example has been implemented in the interactive proof system MINLOG. We show the term which was extracted automatically from a derivation of the theorem.

```
(lambda (a1)
 (lambda (a2)
  ((((((nat-rec-at '(arrow nat (arrow nat (star nat nat)))
      (lambda (k1) (lambda (k2) (cons n000 n000))))
     (lambda (n)
      (lambda (w)
       (lambda (k1)
        (lambda (k2)
         ((((if-at '(star nat nat))
            ((<-strict-nat 0) r2))
           ((w l21) l22))
           ((((if-at '(star nat nat))
             ((<-strict-nat 0) r1))
             ((w l11) l12))
            (cons k1 k2))))))))
    ((plus-nat a2) 1))
   0)
  1)))
```

We have manually introduced `r1, r2, l11, l12, l21, l22` for somewhat lengthy terms corresponding to our abbreviations r_i, $\vec{\ell}_i$. The unbound variable `n000` appearing in the base case is a dummy variable used by the system when it is asked to produce a realizing term for the instance $\bot \to \exists k A(k)$ of ex-falso-quodlibet. In our case, when the existential quantifier is of type ι one might as well pick the constant 0 (as we did in the text).

References

[1] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. In preparation.

[2] Ulrich Berger and Helmut Schwichtenberg. The greatest common divisor: a case study for program extraction from classical proofs. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs. International Workshop TYPES '95, Torino, Italy, June 1995. Selected Papers*, volume 1158 of *Lecture Notes in Computer Science*, pages 36–46. Springer Verlag, Berlin, Heidelberg, New York, 1996.

[3] Thierry Coquand and Hendrik Persson. Gröbner Bases in Type Theory. In T. Altenkirch, W. Naraschewski, and B. Reus, editors, *Types for Proofs and Programs*, volume 1657 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Heidelberg, New York, 1999.

[4] Anne S. Troelstra (editor). *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1973.

[5] Harvey Friedman. Classically and intuitionistically provably recursive functions. In D.S. Scott and G.H. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978.

[6] Ulrich Kohlenbach. Analyzing 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] Daniel Leivant. Syntactic translations and provably recursive functions. *The Journal of Symbolic Logic*, 50(3):682–688, September 1985.

[8] Chetan Murthy. Extracting constructive content from classical proofs. Technical Report 90–1151, Dep.of Comp.Science, Cornell Univ., Ithaca, New York, 1990. PhD thesis.

[9] M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.