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 perspicuous.

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 $\lambda$-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
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 \( \iota \) for the natural numbers and \( o \) for the boolean objects \( \text{true} \) and \( \text{false} \).

\[
\mathcal{V} = \{ \iota, o \} \cup \{ \rho \times \sigma : \rho, \sigma \in \mathcal{V} \} \cup \{ \rho \rightarrow \sigma : \rho, \sigma \in \mathcal{V} \}.
\]

The constants are

\[
\begin{align*}
\text{true}^o & \mid \text{false}^o \mid 0^\iota \mid \mathcal{R}_{o,\iota} \mid \mathcal{R}_{o,\iota}.
\end{align*}
\]

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

Terms are

\[
\begin{align*}
x^\rho & \mid c^\rho (c^\rho \text{ a constant}) \mid \langle M, N \rangle \mid \pi_0(M) \mid \pi_1(M) \mid \lambda x^\rho M \mid M N
\end{align*}
\]

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 \( SK \).

\[
\begin{align*}
\mathcal{R}_{o,\iota} MN \text{ true} & \mapsto_{\mathcal{R}} M \\
\mathcal{R}_{o,\iota} MN \text{ false} & \mapsto_{\mathcal{R}} M \\
\mathcal{R}_{o,\iota} MN0 & \mapsto_{\mathcal{R}} M \\
\mathcal{R}_{o,\iota} MN(K + 1) & \mapsto_{\mathcal{R}} NK(\mathcal{R}_{o,\iota} MNK)
\end{align*}
\]

We will prove that for this system of terms every term strongly normalizes. Since the normal form is uniquely determined, the relation \( _{\beta \mathcal{R}} M =_{\beta \mathcal{R}} N \) 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 $M \to^\beta M'$ 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. $\iff$. 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. $\iff$. 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.

If all terms $\tilde{M}$ are sn, then so is $x\tilde{M}$.

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

If $Mx$ is sn, then so is $M$.

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

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

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

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

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

$M$ is sc if and only if $MN$ is sc for all sc $\tilde{N}$.

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

Lemma 2.1 a. Every sc term $M^\rho$ is sn.

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


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

If $M \to^\beta M'$ 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 $M \to^\beta M'$ and $M$ is sc, then so is $M'$. 
Proof. We use (3). Let \( \bar{N} \) be a list of sc terms such that \( MN \) is of ground type. Then \( M\bar{N} \) is sc by (3), hence also sn. Furthermore we have \( M\bar{N} \rightarrow_{\beta} M'\bar{N} \). By definition of strong normalizability \( M'\bar{N} \) is sn. Thus by (3) \( M' \) is sc.

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

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

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

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

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

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

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 \( \bar{L} \) be sc such that \( (\lambda x M)N\bar{L} \) is of ground type. We must show that \( (\lambda x M)N\bar{L} \) is sc. Since \( M[x := N] \) is sc by assumption, \( M[x := N]\bar{L} \) is sc too and hence also sn. Since by lemma 2.1a \( N \) is also sn, by lemma 2.3 it follows that \( (\lambda x M)N\bar{L} \) is sn and hence also sc.

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

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

From lemma 2.5 and lemma 2.1 we directly get

Theorem 2.6 \( \rightarrow_{\beta} \) 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 $R$ does not destroy termination.

We first change the definition of $sc$ for the ground type $\iota$. 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 $M \rightarrow_\beta M'$ are $sc$, and in addition
- if $M = SM_0$, then $M_0$ is $sc$.

**Theorem 2.8** $\rightarrow_\beta R$ is terminating.

**Proof.** We extend the argument above. Clearly it suffices to show that the constants $R_{\iota,\phi}$ and $R_{\iota,\beta}$ are $sc$. We restrict ourselves to $R_{\iota,\psi}$; for $R_{\iota,\phi}$ the argument is similar (and simpler). So let $M, N, K$ be $sc$. We must show that $RMNK$ is $sc$.

The proof is by induction on $sn$ for $M, N$ and on $sc$ for $K$. So let $\bar{L}$ be $sc$. We must show that $RMNK\bar{L}$ is $sc$. Since this term is not of the form $SM_0$, it suffices to consider all reduces $Q$ of $RMNK\bar{L}$ and to show that every such reduce $Q$ is $sc$.

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

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

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

2.2 Intuitionistic Arithmetic for Functionals

The system we consider is essentially Heyting’s intuitionistic arithmetic in finite types $HA^\omega$ 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 tuple of types $(\rho_1, \ldots, \rho_k)$. So atomic formulas are of the form $P(r_1^{\rho_1}, \ldots, r_k^{\rho_k})$. 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 $\top$ of arity ($o$). The intended interpretation of $\top$ is the set $\{true\}$. Hence “$\top(t)$” means “$t = true$”. Every decidable relation (i.e., every relation given by a term $t$ of type $o$), can then be written in the form $\top(t)$. If confusion is unlikely, we will write $t$ instead of $\top(t)$. So formulas are

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

To accommodate the specific ground types $\iota$ and $o$, we need an induction scheme for both types, and in addition a $\forall$-truth axiom.

- $Ind_{\iota,A}$: $A[p := true] \rightarrow A[p := false] \rightarrow \forall p A$
- $Ind_{o,A}$: $A[n := 0] \rightarrow (\forall n.A \rightarrow A[n := n + 1]) \rightarrow \forall n A$
- $T$: $\top(true)$

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

$$\exists^+_{x,A}: \forall x.A \rightarrow \exists^+ x.A$$
$$\exists^-_{x,A,B}: \exists^+ x^\rho A \rightarrow (\forall x^\rho.A \rightarrow B) \rightarrow B \quad (x \notin FV(B))$$

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

$$u^B \mid \text{assumptions} \mid \text{axioms} \mid \langle M^A, N^B \rangle^{AB} \mid \pi(M^{A\land B})^A \mid \lambda u^A M^B \rightarrow^{AB} \langle M^{A\rightarrow B}, N^A \rangle^B \mid \lambda x^\rho M^A \rightarrow^{x^\rho A} \langle M^{\forall x^\rho A}, t^\rho \rangle^{A[x^\rho := t^\rho]}$$

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 $\top$ is the only predicate symbol, and replace $\bot$ by $\top(false)$. Then we have

$$\neg A := A \rightarrow \top(false)$$
$$\exists x A := \neg \forall x \neg A,$$

and stability $\neg \neg A \rightarrow A$ is provable for all $\exists^*$-free formulas $A$, using boolean induction for the case $\top(t)$. 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 $\top(t)$ we can easily construct a boolean term $t_B:o$ such that

$$\top(t_B) \leftrightarrow B$$

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 $\ast$). $\tau(A)$ is intended to be the type of the program to be extracted from a proof of $A$. In case $\tau(A) = \ast$ proofs of $A$ have no computational content; such formulas $A$ are called Harrop formulas.

\[
\tau(P(s)) := \ast
\]
\[
\tau(\exists^* x^\rho.A) := \begin{cases} \rho \times \tau(A) & \text{if } \tau(A) = \ast \\ \ast & \text{otherwise} \end{cases}
\]
\[
\tau(\forall x^\rho.A) := \begin{cases} \ast & \text{if } \tau(A) = \ast \\ \rho \to \tau(A) & \text{otherwise} \end{cases}
\]
\[
\tau(A_0 \land A_1) := \begin{cases} \tau(A_1) & \text{if } \tau(A_1) = \ast \\ \tau(A_0) \times \tau(A_1) & \text{otherwise} \end{cases}
\]
\[
\tau(A \to B) := \begin{cases} \ast & \text{if } \tau(B) = \ast \\ \tau(A) \to \tau(B) & \text{otherwise} \end{cases}
\]

We now define, for a given derivation $M$ of a formula $A$ with $\tau(A) \not= \ast$, its extracted program $\llbracket M \rrbracket$ of type $\tau(A)$.

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

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

\[
\llbracket \text{Ind}_{\rho,A} \rrbracket := \mathcal{R}_{\ast,\rho} : \rho \to (\iota \to \rho \to \rho) \to \iota \to \rho,
\]

where $\rho = \tau(A)$. Similarly boolean induction is realized by $\mathcal{R}_{0,\rho}$.

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

\[
\llbracket \exists^+ x^\rho.A \rrbracket := \begin{cases} \lambda x^\rho x & \text{if } \tau(A) = \ast \\ \lambda x^\rho \lambda y^{\tau(A)}(x, y) & \text{otherwise} \end{cases}
\]
\[
\llbracket \exists^* x^\rho.A \rrbracket := \begin{cases} \lambda x^\rho f^{\rho \to \tau(B)} f x & \text{if } \tau(A) = \ast \\ \lambda x^{\rho \times \tau(A)} f^{\rho \to \tau(A) \to \tau(B)}, f^{\pi_0(\pi_1)} \pi_1(z) & \text{otherwise} \end{cases}
\]

For derivations $M^A$ where $\tau(A) = \ast$ (i.e. $A$ is a Harrop-formula) we define $\llbracket M \rrbracket := \varepsilon$ ($\varepsilon$ 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 $r \text{ mr } A$, 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 τ(A) = *.

\[ ε \text{ mr } P(\tilde{s}) = P(\tilde{s}) \]
\[ r \text{ mr } (\exists x.A) = \begin{cases} ε \text{ mr } A[x := r] & \text{if } τ(A) = * \\ π_1(r) \text{ mr } A[x := π_0(r)] & \text{otherwise} \end{cases} \]
\[ r \text{ mr } (\forall x.A) = \begin{cases} \forall x.ε \text{ mr } A & \text{if } τ(A) = * \\ ε \text{ mr } A → r \text{ mr } B & \text{otherwise} \end{cases} \]
\[ r \text{ mr } (A → B) = \begin{cases} ε \text{ mr } A → ε \text{ mr } B & \text{if } τ(A) ≠ * = τ(B) \\ ε \text{ mr } A_0 ∧ r \text{ mr } A_1 & \text{if } τ(A_1) = * \end{cases} \]
\[ r \text{ mr } (A_0 ∧ A_1) = \begin{cases} π_0(r) \text{ mr } A_0 ∧ π_1(r) \text{ mr } A_1 & \text{otherwise} \end{cases} \]

Note that for \( A \ \exists^*-\text{free} \) we have \( τ(A) = * \) and \( ε \text{ mr } A = A \). For the formulation of the soundness theorem below it will be useful to let \( x_u := ε \) if \( uA \) 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 \( μ(M) \) of \( \llbracket M \rrbracket \text{ mr } B \) from \( \{ x_u \text{ mr } C \mid uC ∈ FA(M) \} \).

**Proof.** Induction on \( M \). Case \( u.A \). Then we have \( \bar{u} : x_u \text{ mr } A \). Let \( μ(u) := \bar{u} \).

Case \( \lambda uA.M^B \). We must find a derivation \( μ(\lambda uM) \) of \n
\[ \llbracket \lambda uM \rrbracket \text{ mr } (A → B) \]

Subcase \( τ(A) = * \). Then we have \( \llbracket \lambda uM \rrbracket = \llbracket M \rrbracket \), hence
\[ \llbracket \lambda uM \rrbracket \text{ mr } (A → B) = ε \text{ mr } A → \llbracket M \rrbracket \text{ mr } B. \]

By induction hypothesis we can define \( μ(\lambda uM) := λ\bar{u}μ(M) \) with \( \bar{u} : ε \text{ mr } A \). Subcase \( τ(A) ≠ * = τ(B) \). Then we have \( \llbracket \lambda uM \rrbracket = ε \) and
\[ \llbracket \lambda uM \rrbracket \text{ mr } (A → B) = ∀x.x \text{ mr } A → ε \text{ mr } B. \]

By induction hypothesis define \( μ(\lambda uM) := λx_uλ\bar{u}μ(M) \) with \( \bar{u} : x_u \text{ mr } A \). Subcase \( τ(A) ≠ * ≠ τ(B) \). Then we have
\[ \llbracket \lambda uM \rrbracket \text{ mr } (A → B) = ∀x.x \text{ mr } A → \llbracket \lambda uM \rrbracket x \text{ mr } B \]

Because of \( \llbracket \lambda uM \rrbracket = λx_u[\llbracket M \rrbracket ] \) and since we identify terms with the same β-normal form, by induction hypothesis we can define \( μ(\lambda uM) := λx_uλ\bar{u}μ(M) \).

Case \( M^A → B \). We must find a derivation \( μ(MN) \) of \( \llbracket MN \rrbracket \text{ mr } B \). Subcase \( τ(A) = * \). Then we have \( \llbracket MN \rrbracket = \llbracket M \rrbracket \). By induction hypothesis we have derivations \( μ(M) \) of
\[ \llbracket M \rrbracket \text{ mr } (A → B) = ε \text{ mr } A → \llbracket M \rrbracket \text{ mr } B \]

and \( μ(N) \) of \( ε \text{ mr } A \); hence we can define \( μ(MN) := μ(M)μ(N) \). Subcase \( τ(A) ≠ * = τ(B) \). Then we have \( \llbracket MN \rrbracket = ε \). By induction hypothesis we have derivations \( μ(M) \) of
\[ \llbracket M \rrbracket \text{ mr } (A → B) = ∀x.x \text{ mr } A → ε \text{ mr } B \]

and \( μ(N) \) of \( \llbracket N \rrbracket \text{ mr } A \); hence we can define \( μ(MN) := μ(M)\llbracket N \rrbracket μ(N) \). Subcase \( τ(A) ≠ * ≠ τ(B) \). Then we have \( \llbracket MN \rrbracket = \llbracket M \rrbracket \llbracket N \rrbracket \). By induction hypothesis we have derivations \( μ(M) \) of
\[ \llbracket M \rrbracket \text{ mr } (A → B) = ∀x.x \text{ mr } A → \llbracket M \rrbracket x \text{ mr } B \]
and $\mu(N)$ of $[N] \text{mr } A$; hence we can define $\mu(MN) := \mu(M)[N] \mu(N)$.

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

$$\langle M_0, M_1 \rangle \text{mr } (A_0 \land A_1).$$

\textbf{Subcase} $\tau(A_0) = * = \tau(A_1)$. Then we have $\langle M_0, M_1 \rangle = \varepsilon$, hence

$$\langle M_0, M_1 \rangle \text{mr } (A_0 \land A_1) = \varepsilon \text{ mr } A_0 \land \varepsilon \text{ mr } A_1 \text{ and by induction hypothesis we can define } \mu(\langle M_0, M_1 \rangle) := \langle \mu(M_0), \mu(M_1) \rangle.$$

\textbf{Subcase} $\tau(A_0) = * \neq \tau(A_1)$. Then we have $\langle M_0, M_1 \rangle = [M_1]$, hence

$$\langle M_0, M_1 \rangle \text{ mr } (A_0 \land A_1) = \varepsilon \text{ mr } A_0 \land \varepsilon \text{ mr } A_1 \text{ and by induction hypothesis we can define } \mu(\langle M_0, M_1 \rangle) := \langle \mu(M_0), \mu(M_1) \rangle.$$

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

$$\pi_0(M) \text{ mr } A_0.$$

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

$$[M] \text{ mr } (A_0 \land A_1) = [M] \text{ mr } A_0 \land \varepsilon \text{ mr } A_1,$$

hence we can define $\mu(\pi_0(M)) := \pi_0(\mu(M))$. \textbf{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

$$[M] \text{ mr } (A_0 \land A_1) = \varepsilon \text{ mr } A_0 \land [M] \text{ mr } A_1.$$

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

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

$$\lambda z [M] \text{ mr } A.$$

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 $u: B$ free in $M^A$, hence also does not occur free in the free assumption $\pi: x_u \text{ mr } B$. \textbf{Subcase} $\tau(A) \neq *$. Then we have

$$\lambda z [M] \text{ mr } \forall z A = \forall z, (\lambda z [M]) z \text{ mr } A.$$

Since we identify terms with the same $\beta$-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 $M^\forall A.t$. We must find a derivation $\mu(Mt)$ of $[[Mt]]$ $mr A[x := t]$. By definition we have $[[Mt]] = [Mt]$. Subcase $\tau(A) = \ast$. By induction hypothesis we have a derivation of 

$$[[M]] mr \forall z A = \forall z.\varepsilon mr A$$

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

$$[[M]] mr \forall z A = \forall z.[[M]]z mr A,$$

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

Case $\exists x.\forall x.A \rightarrow \exists x^\circ A$. We must find a derivation $\mu(\exists x^+_A)$ of 

$$[[\exists x^+_A]] mr \forall x.A \rightarrow \exists x^\circ A.$$

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

$$\forall x.A \rightarrow \exists x^\circ A = \forall \vec{x}x.A \rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow x mr \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr A.$$

Hence we can define $\mu(\exists x^+_A) = \lambda\vec{x}\lambda x x \lambda y(x, y)$ and we obtain 

$$\lambda\vec{x}\lambda x x \lambda y(x, y) mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr A.$$

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

$$\varepsilon mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \varepsilon mr (\forall \vec{x}x.\rightarrow \exists x^\circ A) = \varepsilon mr (\forall \vec{x}x.\rightarrow \varepsilon mr A \rightarrow x mr B) = \varepsilon mr (\forall \vec{x}x.\rightarrow \varepsilon mr A \rightarrow \varepsilon mr B) = \forall \vec{x}x.\varepsilon mr A \rightarrow \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr B.$$

Hence we can define $\mu(\exists x^+_A) = \lambda\vec{x}\lambda x x \lambda y \lambda y(\lambda u u).$ Subcase $\tau(A) \neq \ast = \tau(B)$. Then we have 

$$[[\exists x^+_A,B]] = \varepsilon$$

and we obtain 

$$\varepsilon mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr A.$$

Hence we can define $\mu(\exists x^+_A) = \lambda\vec{x}\lambda x x \lambda y \lambda y(\lambda u u).$ Subcase $\tau(A) \neq \ast = \tau(B)$. Then we have 

$$[[\exists x^+_A,B]] = \varepsilon$$

and we obtain 

$$\varepsilon mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr B.$$

Hence we can define $\mu(\exists x^+_A) = \lambda\vec{x}\lambda x x \lambda y \lambda y \lambda y(\lambda u u).$ Subcase $\tau(A) \neq \ast = \tau(B)$. Then we have 

$$[[\exists x^+_A,B]] = \varepsilon$$

and we obtain 

$$\varepsilon mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr B.$$

Hence we can define $\mu(\exists x^+_A) = \lambda\vec{x}\lambda x x \lambda y \lambda y \lambda y(\lambda u u).$ Subcase $\tau(A) \neq \ast = \tau(B)$. Then we have 

$$[[\exists x^+_A,B]] = \varepsilon$$

and we obtain 

$$\varepsilon mr \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\rightarrow \exists x^\circ A = \forall \vec{x}x.\varepsilon mr A \rightarrow \varepsilon mr B.$$
Hence we can define $\mu(\exists^*_x A, B) = \lambda x \lambda u \lambda v. v xu$. 

Subcase $\tau(A) \neq \tau(B)$. Then we have $\exists^*_x A, B = \lambda x \lambda z (z \pi_0(x) \pi_1(x))$ and we obtain

$$\begin{align*}
&\lambda x \lambda z (z \pi_0(x) \pi_1(x)) \text{ mr } \forall x [\exists^* x A \rightarrow (\forall x A \rightarrow B) \rightarrow B] \\
&= \forall x \forall z (z \pi_0(x) \pi_1(x)) \text{ mr } ((\forall x \rightarrow B) \rightarrow B) \\
&= \forall x \forall z \forall y (x \text{ mr } A) + z \pi_0(x) \pi_1(x) \text{ mr } B.
\end{align*}$$

Hence we can define $\mu(\exists^*_x A, B) = \lambda x \lambda u \lambda v. v \pi_0(x) \pi_1(x) u$. 

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

$$t \text{ mr } \forall x \exists^* y B = \forall x B[ y := tx].$$

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

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

Here $\Gamma$ 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 Define and Goal Formulas

For simplicity we only treat formulas in the $\rightarrow \forall$ 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 $B \rightarrow C$ is relevant, and

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 $\bot$).

$$
\begin{align*}
G := & \ P | D \rightarrow G \quad \text{provided } D \text{ irrelevant } \Rightarrow D \text{ quantifier-free} \\
& | \forall x G \quad \text{provided } G \text{ irrelevant} \\
D := & \ P | G \rightarrow D \quad \text{provided } D \text{ irrelevant } \Rightarrow G \text{ irrelevant} \\
& | \forall x D
\end{align*}
$$
Lemma 3.1 For definite formulas $D$, relevant goal formulas $G$ and an arbitrary formula $A$ we have derivations

\[
N^A_D : D \rightarrow D^A \\
H^A_G : G^A \rightarrow \neg G \rightarrow A
\]

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 $D_1, \ldots, D_i$ be relevant and $D_{i+1}, \ldots, D_n$ irrelevant definite formulas, and $G_1, \ldots, G_k$ be relevant quantifier-free and $G_{k+1}, \ldots, G_m$ irrelevant goal formulas. Let $A := \exists y \check{G}$ and

\[
t := \lambda y, \check{x}. \text{if } \neg G_1 \text{ then } [H^A_{G_1}](x_1) \text{ else } \ldots \text{ if } \neg G_k \text{ then } [H^A_{G_k}](x_k) \text{ else } y.
\]

Assume that we have a derivation

\[
M : \check{D} \rightarrow (\forall y, \check{G} \rightarrow \bot) \rightarrow \bot.
\]

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

\[
\check{D} \rightarrow [M^A] [N^A_D] \ldots [N^A_{D_1}] t \text{ mr } \exists y \check{G}.
\]

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 $f : \mathbb{I} \rightarrow \mathbb{I}$ 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 $f$.

\[
\forall n \neg n < f0 \rightarrow \forall n n < f(gn) \rightarrow (\forall m. \neg n < fm \rightarrow n < f(m + 1) \rightarrow \bot) \rightarrow \bot
\]

Clearly

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

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

To construct a derivation, assume

\[
v_1 : \forall n \neg n < f0, \\
v_2 : \forall n n < f(gn), \\
u : \forall m. \neg n < fm \rightarrow n < f(m + 1) \rightarrow \bot.
\]

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

\[
\begin{align*}
\text{Ind} & \quad \neg n < f0 & & \forall m. \neg n < fm & & \rightarrow \neg n < f(m + 1) \\
\neg n < f(m) & & gn & & n < f(gn) & & \bot
\end{align*}
\]
with derivation term

\[ M := \lambda v_1 \lambda v_2 \lambda u. \text{Ind}_{m, -n < f m} n(v_1 n)u(g n)(v_2 n), \]

Now let \( A := \exists \mathcal{m}. -n < f m \land n < f(m + 1), \) hence

\[
M^A = \lambda v_1^{\forall n < f 0 \rightarrow A} \lambda v_2^{\forall n < f g n} \lambda u^{\forall m, (n < f m) \rightarrow n < f(m + 1) \rightarrow A}. \text{Ind}_{m, n < f m \rightarrow A} n(v_1 n)u(g n)(v_2 n)
\]

and therefore

\[ \llbracket M^A \rrbracket = \beta \lambda x_1^{\rightarrow \mathcal{m}} \lambda x_2^{\rightarrow \mathcal{m}}. \mathcal{R}_i(x_1 n)x_u(g n) \]

From the proof of theorem 3.2 we generally know \( D \rightarrow \llbracket N_D^A \rrbracket \text{ mr } D^A \) for every relevant definite formula \( D \). In our case, with \( D = \forall n. -n < f 0 \), we can derive directly

\[(\forall n. n < f 0 \rightarrow \bot) \rightarrow (\lambda n 0) \text{ mr } \forall n. n < f 0 \rightarrow A,\]

since we can use ex-falso. So we may assume \( \llbracket N_D^A \rrbracket = \lambda n 0 \). Also, from the proof of theorem 3.2 we generally know \( x \text{ mr } G^A \rightarrow \neg G \rightarrow \llbracket H_G^A \rrbracket(x) \text{ mr } A \) for every relevant goal formula \( G \). In our case, with \( G = -n < f m \), we can derive directly

\[(x \text{ mr } n < f m \rightarrow A) \rightarrow \neg\neg n < f m \rightarrow x \text{ mr } A.\]

So we may assume \( \llbracket H_G^A \rrbracket = \lambda x x \). Now let

\[
t : = \lambda m \lambda x. \text{ if } -G_1 \text{ then } \llbracket H_G^A \rrbracket(x) \text{ else } m = \lambda m \lambda x. \text{ if } n < f m \text{ then } x \text{ else } m.
\]

Then the extracted term according to theorem 3.2 is

\[
\llbracket M^A \rrbracket[N_D^A]t = \llbracket \lambda x_1^{\rightarrow \mathcal{m}} \lambda x_2^{\rightarrow \mathcal{m}}. \mathcal{R}_i(x_1 n)x_u(g n) \rrbracket(\lambda n 0) t = \mathcal{R}_i 0 t(g n) = \mathcal{R}_i 0(\lambda n \lambda x. \text{ if } n < f m \text{ then } x \text{ else } m)(g n).
\]

Informally, the algorithm can be written in the form \( h(g n) \) where \( h : \mathcal{I} \rightarrow \mathcal{I} \) is such that

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

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 perspicuous.

Let \( a, b, c, i, j, k, \ell, m, n, q, r \) denote natural numbers. Our language is determined by the constants \( 0, 1, +, \times \), function symbols for the quotient and the remainder denoted
by $q(a,c)$ and $r(a,c)$, a $4$-ary function denoted by $\text{abs}(k_1a_1 - k_2a_2)$ 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) $v_1, \ldots, v_6$ of them; these will be formulated as we need them.

**Theorem 3.3** For natural numbers $a_1, a_2$ with $0 < a_2$
\[
\exists k_1, k_2. \text{abs}(k_1a_1 - k_2a_2)\vert a_1 \land \text{abs}(k_1a_1 - k_2a_2)\vert a_2 \land 0 < \text{abs}(k_1a_1 - k_2a_2).
\]

*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 $c = \text{abs}(k_1a_1 - k_2a_2)$ with $k_1, k_2 \in \mathbb{N}$. It is a common divisor of $a_1$ and $a_2$ since otherwise the remainder $r(a_1, c)$ 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 $c$. \hfill \Box

**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 $\mu$ says
\[
\exists \bar{k}.R(\bar{k}) \rightarrow \exists \bar{k}.R(\bar{k}) \land \forall \bar{\ell}.\mu(\bar{\ell}) < \mu(\bar{k}) \rightarrow R(\bar{\ell}) \rightarrow \bot
\]
(in our example $R(k_1, k_2) \equiv 0 < \text{abs}(k_1a_1 - k_2a_2)$ and $\mu(k_1, k_2) \equiv \text{abs}(k_1a_1 - k_2a_2)$). In order to reduce this to the induction scheme we use the fact that the formula above is classically equivalent to
\[
(\forall \bar{\ell}.R(\bar{k}) \rightarrow [\forall \bar{\ell}.\mu(\bar{\ell}) < \mu(\bar{k}) \rightarrow R(\bar{\ell}) \rightarrow \bot] \rightarrow \bot) \rightarrow \forall \bar{k}.R(\bar{k}) \rightarrow \bot,
\]
i.e. the principle of *course-of-values induction* for $R(\bar{k}) \rightarrow \bot$ w.r.t. the measure $\mu(\bar{k})$. We can write this as
\[
\text{Prog} \rightarrow \forall \bar{k}.R(\bar{k}) \rightarrow \bot,
\]
where
\[
\text{Prog} := \forall \bar{k}.[\forall \bar{\ell}.\mu(\bar{\ell}) < \mu(\bar{k}) \rightarrow R(\bar{\ell}) \rightarrow \bot] \rightarrow R(\bar{k}) \rightarrow \bot.
\]
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 $w_1: \text{Prog}$ and prove $\forall \bar{k}.R(\bar{k}) \rightarrow \bot$. This is achieved by proving $\forall nB$, where
\[
B := \forall \bar{k}.\mu(\bar{k}) < n \rightarrow R(\bar{k}) \rightarrow \bot,
\]
and using $B$ with $n := \mu(\bar{k}) + 1$. We prove $\forall nB$ by (zero-successor) induction.

**Base.** $B[n := 0]$ follows easily from the lemma
\[
v_1: \forall m. m < 0 \rightarrow \bot.
\]

**Step.** Let $n$ be given and assume $w_2: B$. To show $B[n := n + 1]$ let $\bar{k}$ be given and assume $w_3: \mu(\bar{k}) < n + 1$. We will derive $R(\bar{k}) \rightarrow \bot$ by using $w_1: \text{Prog}$ at $\bar{k}$. Hence we have to prove
\[
\forall \bar{\ell}.\mu(\bar{\ell}) < \mu(\bar{k}) \rightarrow R(\bar{\ell}) \rightarrow \bot.
\]
So, let $\bar{\ell}$ be given and assume further $w_4: \mu(\bar{l}) < \mu(\bar{k})$. From $w_4$ and $w_3: \mu(\bar{k}) < n + 1$ we infer $\mu(\bar{l}) < n$ (using an arithmetical lemma). Hence, by induction hypothesis $w_2: B$ at $\bar{l}$ we get $R(\bar{l}) \rightarrow \bot$. 

**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 $v_0: 0 < a_2$ and

$$u: \forall k_1, k_2. \text{abs}(k_1 a_1 - k_2 a_2) | a_1 \rightarrow \text{abs}(k_1 a_1 - k_2 a_2) | a_2 \rightarrow 0 < \text{abs}(k_1 a_1 - k_2 a_2) \rightarrow \bot.$$  

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

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

$$f: \forall k_1, k_2 \rightarrow (0 < k_1 a_1 - k_2 a_2 \rightarrow \bot \rightarrow \bot).$$

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

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

$$N(k_1, k_2) := 0 < \text{abs}(k_1 a_1 - k_2 a_2) \rightarrow \bot \text{ and } \mu(k_1, k_2) := \text{abs}(k_1 a_1 - k_2 a_2).$$

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

$$u_1: \forall \ell_1, \ell_2, \mu(\ell_1, \ell_2) < \mu(k_1, k_2) \rightarrow N(\ell_1, \ell_2).$$

We have to prove $N(k_1, k_2)$. So, assume $u_2: 0 < \mu(k_1, k_2)$. 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, $\mu(k_1, k_2) | a_1$ 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 $\ell_1, \ell_2$ 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

$$v_2: \forall a, q, c, r. a = q c + r \rightarrow (0 < r \rightarrow \bot) \rightarrow c | a$$

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

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

of the instantiated lemma $v_2$. Here we need the lemmata

$$v_3: \forall a, c. 0 < c \rightarrow a = q(a, c)c + r(a, c),$$

$$v_4: \forall a, c. 0 < c \rightarrow r(a, c) < c$$

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

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

$f$ satisfies the lemma

$$v_5: \forall a_1, a_2, k_1, k_2, q, r. a_1 = q \mu(k_1, k_2) + r \rightarrow r = \mu(f(a_1, a_2, k_1, k_2, q), q k_2).$$

Hence we let $\ell_1 := f(a_1, a_2, k_1, k_2, q_1)$ and $\ell_2 := q_1 k_2$. 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 $\ell_1, \ell_2$ (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

$$v_6: \forall a_1, a_2, k_1, k_2, q, r. a_2 = q \mu(k_1, k_2) + r \rightarrow r = \mu(q k_1, f(a_2, a_1, k_1, k_2, q)).$$
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 exception of

\[ v_2 : \forall a, q, c, r. a = qc + r \rightarrow (0 < r \rightarrow \bot) \rightarrow c | a, \]

which is not definite (we have concluded something positive, i.e. \( c | a \), from something negative, i.e. \( 0 < r \rightarrow \bot \)). 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

\[ \forall a, q, c, r. a = qc + r \rightarrow (0 < r \rightarrow \bot) \rightarrow (c | a \rightarrow \bot) \rightarrow \bot. \]

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.

\[
M := \lambda v_0^{0 < a_2} \\
\quad \quad \quad \lambda v_1^{1, m < 0 \rightarrow \bot} \\
\quad \quad \quad \lambda v_2^{1, n, m < 0 \rightarrow \bot} \quad \text{Prog}
\]

where

\[
M_{\text{cvind}} = \lambda w_{\text{cvind}} \lambda \tilde{k}. \text{Ind}_{n, B}(k, M_{\text{base}}, M_{\text{step}}(\mu(\tilde{k}) + 1)\tilde{k}L^{\mu(\tilde{k})}_n < \mu(\tilde{k}) + 1),
\]

\[
M_{\text{base}} = \lambda \tilde{k} \lambda w_{0}^{0} \lambda \bar{u}_0^{1, \xi < \mu(\tilde{k})}. \text{v} \lambda \mu(\tilde{k}) w_0,
\]

\[
M_{\text{step}} = \lambda n \lambda w_{B}^{1, \xi < \mu(\tilde{k})}. \lambda \tilde{k} \lambda w_{3}^{\mu(\tilde{k})}. L^{\mu(\tilde{k})}_n < \mu(\tilde{k}), w_1(\tilde{k})(\lambda \tilde{\mu} u_4^{\mu(\tilde{k})} < \mu(\tilde{k})). w_2(\tilde{k})(L^{\mu(\tilde{k})}_n < w_4, w_3)),
\]

\[
M_{\text{prog}} = \lambda \tilde{k} \lambda v_0^{\forall k < \mu(\tilde{k})}. 0 < \mu(\tilde{k}) \rightarrow \bot \lambda v_2^{0 < \mu(\tilde{k})}.
\]

Here we have used the abbreviations

\[
\text{Prog} = \forall \tilde{k}. (\forall \tilde{\mu}(\tilde{k}) < \mu(\tilde{k}) \rightarrow 0 < \mu(\tilde{k}) \rightarrow \bot) \rightarrow 0 < \mu(\tilde{k}) \rightarrow \bot
\]

\[
B = \forall \tilde{k}. (\forall \mu(\tilde{k}) < n \rightarrow 0 < \mu(\tilde{k}) \rightarrow \bot
\]
Term extraction

Let

\[ A := \exists \bar{k}. \mu(\bar{k})[a_1 \land \mu(\bar{k})[a_2 \land 0 < \mu(\bar{k})], \]

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

\[ \tau(\text{Prog}^A) = \bar{i} \rightarrow (\bar{i} \rightarrow \bar{i}) \rightarrow \bar{i}, \]
\[ \tau(B^A) = \bar{i} \rightarrow \bar{i}. \]

Furthermore recall \([\text{Ind}_{n,r}^A] = \mathcal{R}_{\bar{i}, \bar{i} \rightarrow \bar{i}} \) where

\[ \mathcal{R}_{\bar{i}, \bar{i} \rightarrow \bar{i}} : (\bar{i} \rightarrow \bar{i}) \rightarrow (\iota \rightarrow (\bar{i} \rightarrow \bar{i}) \rightarrow (\bar{i} \rightarrow \bar{i})) \rightarrow \iota \rightarrow (\bar{i} \rightarrow \bar{i}) \]

Note that \( \mathcal{R}_{\bar{i}, \bar{i} \rightarrow \bar{i}} \) may be viewed as simultaneous primitive recursion operators \( \overline{\mathcal{R}} = (\mathcal{R}_1, \mathcal{R}_2) \) with

\[ \mathcal{R}_{\bar{i}, \bar{i}} f^0 = y_i, \]
\[ \mathcal{R}_{\bar{i}, \bar{i}} f^1 = \mathcal{R}_1 \bar{i} \bar{f}^1(\mathcal{R}_2 \bar{i} \bar{f}^1). \]

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 \( A \)).

\[ [M^A] = \lambda x_{v_1}^{\bar{i} \rightarrow \bar{i}} \lambda x_{v_2}^{\bar{i} \rightarrow \bar{i}} \lambda x_u^{\bar{i} \rightarrow \bar{i}}. [M^A_{\text{vind}}][M^A_{\text{base}}][M^A_{\text{step}}][M^A_{\text{div}}] [\lambda(\bar{k}, \bar{k})] \]

where

\[ [M^A_{\text{vind}}] = \lambda x_{v_1}^{\bar{i} \rightarrow \bar{i}} \lambda x_{v_2}^{\bar{i} \rightarrow \bar{i}} \lambda x_u^{\bar{i} \rightarrow \bar{i}}. [M^A_{\text{base}}][M^A_{\text{step}}][M^A_{\text{div}}](\mu(\bar{k}) + 1)\bar{k} \]
\[ [M^A_{\text{base}}] = \lambda \bar{k}. x_{v_1}^{\bar{i} \rightarrow \bar{i}} \lambda \bar{k}. [M^A_{\text{step}}][M^A_{\text{div}}] \]
\[ [M^A_{\text{step}}] = \lambda n \lambda x_{v_1}^{\bar{i} \rightarrow \bar{i}} \lambda \bar{k}. x_u^{\bar{i}}(\lambda \bar{k}. x_{v_2}^{\bar{i}} \bar{k}), \]
\[ [M^A_{\text{div}}] = \lambda \bar{k}. \lambda x_{v_1}^{\bar{i} \rightarrow \bar{i}}. [M^A_{\text{div2}}](x_u^{\bar{i}}) \]
\[ [M^A_{\text{div2}}] = x_{v_2} \lambda q.(\bar{k}). [M^A_{\text{div2}}] \]
\[ [M^A_k] = x_u \bar{f}(\bar{k}) \]

From the proof of theorem 3.2 we generally know \( D \rightarrow [N_D^A] \) \( \text{mr} \) \( D^A \) for every relevant definite formula \( D \). In our case for \( D_1 = \forall m.m < 0 \rightarrow \bot \) we can derive directly

\( (\forall m.m < 0 \rightarrow \bot) \rightarrow (\lambda n 0) \) \( \text{mr} \) \( \forall m.m < 0 \rightarrow A \),

since we can use ex-falso. So we may assume \( [N_D^A] = \lambda n 0 \). Similarly for

\[ D_2 = \forall a, q, c, r.a = qc + r \rightarrow (0 < r \rightarrow \bot) \rightarrow (c|a \rightarrow \bot) \rightarrow \bot \]
\[ D_2^A = \forall a, q, c, r.a = qc + r \rightarrow (0 < r \rightarrow A) \rightarrow (c|a \rightarrow A) \rightarrow A. \]

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

From this proof we can derive directly

\[ D_2 \rightarrow (\lambda a q \lambda c r \lambda a \lambda y. \text{if } 0 < r \text{ then } \bar{x} \text{ else } \bar{y}). \text{mr } D_2^A. \]

So we may assume \( [N_{D_2}^A] = \lambda a q \lambda c r \lambda a \lambda y. \text{if } 0 < r \text{ then } \bar{x} \text{ else } \bar{y}. \)
Moreover, there is no relevant goal formula, hence \( t = \lambda \vec{y} \vec{y} \). Then the extracted term according to theorem 3.2 is

\[
\llbracket M^A \rrbracket \llbracket N_{D_1}^A \rrbracket \llbracket N_{D_2}^A \rrbracket t \beta \llbracket M_{\text{cvind}}^A \rrbracket' \llbracket M_{\text{prog}}^A \rrbracket' 01
\]

where ‘ indicates substitution of \( \llbracket N_{D_1}^A \rrbracket \), \( \llbracket N_{D_2}^A \rrbracket \), \( t \) for \( x_{v_1}, x_{v_2}, x_u \), so

\[
\llbracket M_{\text{cvind}}^A \rrbracket' = \beta \lambda x_{v_1}. (\lambda n \lambda x_{v_2} \lambda \vec{k}. x_{v_1} \vec{k} x_{v_2} t (\mu(\vec{k}') + 1)) \vec{k}',
\]

\[
\llbracket M_{\text{prog}}^A \rrbracket' = \beta (\lambda \vec{k} \lambda x_{v_1}. t (\llbracket M_{\text{div}}^A \rrbracket' (\llbracket M_{\text{div}}^A \rrbracket' \vec{k})),
\]

\[
\llbracket M_{\text{div}}^A \rrbracket' = \beta (\lambda x \lambda y, \text{if } 0 < r_i(\vec{k}) \text{ then } x \text{ else } y)(x_{v_1} \vec{f}_i(\vec{k})).
\]

and hence

\[
\llbracket M_{\text{prog}}^A \rrbracket' = \beta \lambda \vec{k} \lambda x_{v_1}. \text{ if } 0 < r_2(\vec{k}) \text{ then } x_{v_1} \vec{f}_2(\vec{k}) \text{ else }
\]

\[
( \text{ if } 0 < r_1(\vec{k}) \text{ then } x_{u_1} \vec{f}_1(\vec{k}) \text{ else } \vec{k})
\]

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

\[
\llbracket M^A \rrbracket \llbracket N_{D_1}^A \rrbracket \llbracket N_{D_2}^A \rrbracket t = \beta R(\lambda \vec{k}, \vec{0})
\]

\[
(\lambda n \lambda x_{v_2}^\tau(\lambda \vec{k}'. (\text{if } 0 < r_2(\vec{k}) \text{ then } x_{v_2} \vec{f}_2(\vec{k}) \text{ else }
\]

\[
( \text{ if } 0 < r_1(\vec{k}) \text{ then } x_{u_2} \vec{f}_1(\vec{k}) \text{ else } \vec{k})
\]

\[
(a_2 + 1) 01
\]

To make this algorithm more readable we may write \( \llbracket M^A \rrbracket \llbracket N_{D_1}^A \rrbracket \llbracket N_{D_2}^A \rrbracket t = h(a_2+1, 0, 1) \), where

\[
h(0, \vec{k}) := 0,
\]

\[
h(n + 1, \vec{k}) := \text{ if } 0 < r_2(\vec{k}) \text{ then } h(n, \vec{f}_2(\vec{k})) \text{ else }
\]

\[
( \text{ if } 0 < r_1(\vec{k}) \text{ then } h(n, \vec{f}_1(\vec{k})) \text{ else } \vec{k})
\]
Example

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

$h(28,0,1)$ \hspace{1em} $\mu(0, 1) = 27$
\hspace{2em} $0 < r_1 = 12$ \hspace{1em} $q_1 = 2$
\hspace{2em} $q_1k_1 \pm 1$, \hspace{1em} $q_1k_2 \pm 1$ \hspace{1em} if $k_2a_2 < k_1a_1$
\hspace{2em} No $1$, \hspace{1em} $2$

$h(27,1,2)$ \hspace{1em} $\mu(1, 2) = 12$
\hspace{2em} $0 < r_1 = 6$ \hspace{1em} $q_1 = 5$
\hspace{2em} $q_1k_1 \pm 1$, \hspace{1em} $q_1k_2 \pm 1$ \hspace{1em} if $k_2a_2 < k_1a_1$
\hspace{2em} Yes $4$, \hspace{1em} $10$

$h(26,4,10)$ \hspace{1em} $\mu(4, 10) = |4 \cdot 66 - 10 \cdot 27| = |264 - 270| = 6$
\hspace{2em} $6 | 66$
\hspace{2em} $0 < r_2 = 3$ \hspace{1em} $q_2 = 4$
\hspace{2em} $q_2k_1$, \hspace{1em} $q_2k_2 \pm 1$ \hspace{1em} if $k_1a_1 < k_2a_2$
\hspace{2em} Yes $4 | 4 - 10$
\hspace{2em} $4 \cdot 66 = 264$ \hspace{1em} $10 \cdot 27 = 270$
\hspace{2em} $16$, \hspace{1em} $39$

$h(25,16,39)$ \hspace{1em} $\mu(16, 39) = |16 \cdot 66 - 39 \cdot 27| = |1056 - 1053| = 3$
\hspace{2em} $3 \cdot 66$
\hspace{2em} $3 \cdot 27$

Result: $16, 39$

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 $[M_{\text{step}}^A] = \lambda n . x_1 \rightarrow x_1 . k \cdot x_{w1} \cdot \tilde{k}(\lambda \tilde{f}.x_{w2} \tilde{f})$ 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.

$$\text{(lambda (a1)}$$
$$\text{(lambda (a2)}$$
$$\text{(((((nat-rec-at ')}(arrow nat (arrow nat (star nat nat))))))}$$
We have manually introduced \( r_1, r_2, 111, 112, 121, 122 \) for somewhat lengthy terms corresponding to our abbreviations \( r_i, \ell_i \). 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 \( \bot \rightarrow \exists k A(k) \) of ex-falso-quo-dlibet. In our case, when the existential quantifier is of type \( \iota \) one might as well pick the constant 0 (as we did in the text).

References


