The greatest common divisor: a case study for program extraction from classical proofs

U. Berger H. Schwichtenberg

May 29, 1996

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. We apply H. Friedman's A–translation [3] followed by a modified realizability interpretation to extract a program from this proof. However, to obtain a reasonable program it is essential to use a refinement of the A–translation introduced in Berger/Schwichtenberg [1, 2]. This refinement makes it possible that not all atoms in the proof are A–translated, but only those with a "critical" relation symbol. In our example only the divisibility relation $\cdot|\cdot$ will be critical.

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 $\mathrm{abs}(k_1a_1 - k_2a_2)$ 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.

$$\forall a_1, a_2(0 < a_2 \to \exists k_1, k_2(\mathrm{abs}(k_1a_1 - k_2a_2)|a_1 \wedge \mathrm{abs}(k_1a_1 - k_2a_2)|a_2 \wedge 0 < \mathrm{abs}(k_1a_1 - k_2a_2))).$$

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 $c = \mathrm{abs}(k_1a_1 - k_2a_2)$ with $k_1, k_2 \in \mathbf{N}$. 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 least element principle and <-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 least element principle. The least element principle w.r.t. a measure μ says

$$\exists\vec{k}\, M(\vec{k}) \to \exists\vec{k}\,(M(\vec{k}) \wedge \forall\vec{\ell}[\mu(\vec{\ell}) < \mu(\vec{k}) \to M(\vec{\ell}) \to \bot])$$

(in our example $M(k_1, k_2) \equiv 0 < \mathrm{abs}(k_1 a_1 - k_2 a_2)$ and $\mu(k_1, k_2) \equiv \mathrm{abs}(k_1 a_1 - k_2 a_2)$). In order to reduce this to the induction scheme we use the fact that the formula above is classically equivalent to

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

which nothing but the principle of *<–induction* for the complement of M, $N(\vec{k}) := M(\vec{k}) \to \bot$. We can write this as

$$\mathrm{Prog}(N) \to \forall\vec{k}\, N(\vec{k}),$$

where

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

In the formal treatment of our example it will be more convenient to use the least element principle in the form of <–induction.

To prove <–induction we assume that N is progressive,

$$w_1 \colon \mathrm{Prog}(N),$$

and prove $\forall\vec{k}\, N(\vec{k})$. This is achieved by proving $\forall n B(n)$, where

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

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

Base. $B(0)$ follows easily from the lemma

$$v_1 \colon \forall m(m < 0 \to \bot)$$

and Efq: $\bot \to N(\vec{k})$. Efq is not needed if (as in our example) N is a negation.

Step. Let n be given and assume $w_2 \colon B(n)$. To show $B(n+1)$ let $\vec{k}$ be given and assume $w_3 \colon \mu(\vec{k}) < n + 1$. We will derive $N(\vec{k})$ by using the progressiveness of N, w_1, at $\vec{k}$. Hence we have to prove

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

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

Detailed proof of the theorem.

Now we repeat the proof of the theorem in some more detail using $<$–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

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

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

The principle of $<$-induction is used with

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

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 $\ell_2 := q_1 k_2$. 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 $\perp$ 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

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

The refined A–translation.

The proof of the principle of $<$–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. We will now briefly recall the term extraction process described in [1, 2] in general, and will see that we don't need to worry about these omissions.

Let $\forall \vec{x}_1 C_1, \ldots, \forall \vec{x}_\ell C_\ell$ be Π–formulas (i.e. C_i quantifier free) and $A_1, \ldots, A_m$ quantifier free formulas (in our example $C_1 \equiv 0 < a_2$ ($\vec{x}_1$ is empty), $A_1 \equiv \mathrm{abs}(a_1k_1 - a_2k_2)|a_1$, $A_2 \equiv \mathrm{abs}(a_1k_1 - a_2k_2)|a_2$, $A_3 \equiv 0 < \mathrm{abs}(a_1k_1 - a_2k_2)$). Assume we have a classical proof of

$$\forall \vec{a}(\forall \vec{x}_1 C_1 \to \ldots \to \forall \vec{x}_\ell C_\ell \to \exists \vec{k}(A_1 \wedge, \ldots, \wedge A_m)),$$

i.e., a deduction

$$d[u\colon \forall \vec{k}(A_1 \to \ldots \to A_m \to \perp), v_1\colon \forall \vec{x}_1 C_1, \ldots, v_\ell\colon \forall \vec{x}_\ell C_\ell,]\colon \perp.$$

To keep the derivation short we allow auxiliary lemmata $v_{\ell+1}\colon \forall \vec{x}_{\ell+1} C_{\ell+1}, \ldots, v_n\colon \forall \vec{x}_n C_n$ asserting true Π–formulas. So, in fact, we have

$$d[u, v_1, \ldots, v_n]\colon \perp.$$

We sketch the main steps leading from this derivation to an intuitionistic proof of

$$A := \exists^* \vec{k}(A_1 \wedge \ldots \wedge A_m)$$

($\exists^*$ is the constructive existential quantifier) and hence to terms computing a witness $\vec{k}$.

1. Let $L := \{A_1 \to \ldots \to A_m \to \perp, C_1, \ldots, C_n\}$. Determine inductively the L–*critical* relation symbols as follows: If $(\vec{D}_1 \to P_1) \to \ldots \to (\vec{D}_m \to P_m) \to R(\vec{t})$ is a positive subformula of a formula in L, and for some i, $P_i \equiv \perp$ or $P_i \equiv Q(\vec{s})$ where Q is L–critical, then R is L–critical too.

2. For each formula F let its *A–translation* F^A be the formula obtained from F by replacing $\bot$ by A and each subformula $R(\vec{s})$, where R is L–critical, by $(R(\vec{s}) \to A) \to A$. Find derivations

$$d_u\colon \forall \vec{k}(A_1 \to \ldots \to A_m \to \bot)^A, \quad \text{and} \quad d_{v_i}[v_i\colon \forall \vec{x}_i C_i]\colon \forall \vec{x}_i C_i^A$$

following the recipe given in [2].

3. Replace in $d[u, v_1, \ldots, v_n]\colon \bot$ each formula by its A–translation. We obtain a derivation

$$d_A[u_A, v_{1A}, \ldots, v_{nA}]\colon A,$$

where $u_A\colon \forall \vec{k}(A_1^A \to \ldots \to A_m^A \to A)$ and $v_{iA}\colon \forall \vec{x}_i C_i^A$ (induction axioms are replaced by new induction axioms for the A–translated formulas). Furthermore replace in the derivation above the free assumptions by the derivations constructed in step 2. We get the translated derivation

$$d^{\mathrm{tr}}[v_1, \ldots, v_n] := d_A[d_u, d_{v_1}, \ldots, d_{v_n}]\colon A.$$

4. Apply Kreisel's modified realizability interpretation [4] to extract a finite list of terms

$$\vec{r} := (d^{\mathrm{tr}})^{\mathrm{ets}}$$

such that $A_1[\vec{r}/\vec{k}] \wedge \ldots \wedge A_m[\vec{r}/\vec{k}]$ is provable from $v_1, \ldots, v_6$.

Comments.

– Term extraction commutes with the logical rules, e.g. $(d_1 d_2)^{\mathrm{ets}} = d_1^{\mathrm{ets}} d_2^{\mathrm{ets}}$, and substitution, i.e.

$$(d^{\mathrm{tr}})^{\mathrm{ets}} \equiv (d_A[d_u, d_{v_1}, \ldots, d_{v_n}])^{\mathrm{ets}} \equiv d_A^{\mathrm{ets}}[d_u^{\mathrm{ets}}, d_{v_1}^{\mathrm{ets}}, \ldots, d_{v_n}^{\mathrm{ets}}].$$

By the latter we may first extract terms from the derivations $d_A, d_{v_1}, \ldots, d_{v_n}$ and also from the proof of $<$–induction separately, and then substitute these terms into the terms extracted from $d_A[u_A, v_{1A}, \ldots, v_{nA}]\colon A$.

– Assume that we have fixed some system of lemmata $\forall \vec{x}_1 C_1, \ldots, \forall \vec{x}_n C_n$ and computed the L-critical relation symbols according to step 1. Then it's clear that we may use any other true $\to\forall$–formula D as a further lemma, provided D does neither contain $\bot$ nor any L–critical relation symbol. The simple reason is, that in this case $D^A \equiv D$. In the sequel we will call such formulas D *harmless*.

Computing the L–critical relation symbols.

Now we come back to our example. Let us repeat the main lemmata used in the proofs of the principle of $<$–induction and the theorem.

$$
\begin{aligned}
&v_0\colon 0 < a_2,\\
&v_1\colon \forall m(m < 0 \to \bot),\\
&v_2\colon \forall a, q, c, r(a = qc + r \to (0 < r \to \bot) \to c|a),\\
&v_3\colon \forall a, c(0 < c \to a = q(a, c)c + r(a, c)),\\
&v_4\colon \forall a, c(0 < c \to r(a, c) < c),\\
&v_5\colon \forall a_1, a_2, k_1, k_2, q, r(a_1 = q \cdot \mu(k_1, k_2) + r \to r = \mu(f(a_1, a_2, k_1, k_2, q), qk_2)),\\
&v_6\colon \forall a_1, a_2, k_1, k_2, q, r(a_2 = q \cdot \mu(k_1, k_2) + r \to r = \mu(qk_1, f(a_2, a_1, k_2, k_1, q))).
\end{aligned}
$$

The only critical relation symbol w.r.t. $v_0, \ldots, v_6$ is $\cdot|\cdot$. Since the parts of our proofs which were left implicit concerned neither $\cdot|\cdot$ nor $\bot$, they may be viewed as applications of harmless lemmata (in the sense of the second comment) and hence won't cause problems.

Formal derivations.

We now spell out the derivation term $d[u, v_0, v_1, \ldots, v_6]\colon \bot$ formalizing the proof of the theorem. We use the following abbreviations.

$$
\begin{aligned}
\mu(\vec{k}) &:= \mathrm{abs}(k_1a_1 - k_2a_2),\\
q_i(\vec{k}) &:= q(a_i, \mu(\vec{k})),\\
r_i(\vec{k}) &:= r(a_i, \mu(\vec{k})),\\
\vec{\ell}_1(\vec{k}) &:= f(a_1, a_2, k_1, k_2, q_1(\vec{k})), q_1(\vec{k})k_2,\\
\vec{\ell}_2(\vec{k}) &:= q_2(\vec{k})k_1, f(a_2, a_1, k_2, k_1, q_2(\vec{k})),\\
N(\vec{k}) &:= 0 < \mu(\vec{k}) \to \bot,\\
\mathrm{Prog} &:= \forall \vec{k}(\forall \vec{\ell}[\mu(\vec{\ell}) < \mu(\vec{k}) \to N(\vec{\ell})] \to N(\vec{k})),\\
B(n) &:= \forall \vec{k}(\mu(\vec{k}) < n \to N(\vec{k})).
\end{aligned}
$$

Recall that, using the abbreviations above, u denotes the assumption

$$u\colon \forall \vec{k}(\mu(\vec{k})|a_1 \to \mu(\vec{k})|a_2 \to N(\vec{k})).$$

The derivations below are given in a natural deduction calculus and are written as typed λ–terms according to the well–known Curry–Howard correspondence. By e we will denote (different) subderivations of d which derive a harmless formula from harmless assumptions.

There is no need to make them explicit since they will disappear in the term extraction process. The derivation of the theorem is given by

$$d \equiv d_{<\text{-ind}}^{\text{Prog} \to \forall \vec{k} N(\vec{k})} d_{\text{prog}}^{\text{Prog}} 01(e^{0<\mu(0,1)}[v_0]),$$

where

$$\begin{aligned}
d_{<\text{-ind}} &\equiv \lambda w_1^{\text{Prog}} \lambda \vec{k}. \text{Ind}_{n,B(n)} d_{\text{base}} d_{\text{step}} (\mu(\vec{k})+1) \vec{k} e^{\mu(\vec{k})<\mu(\vec{k})+1}, \\
d_{\text{base}} &\equiv \lambda \vec{k}, w_0^{\mu(\vec{k})<0}, \tilde{w}_0{}^{0<\mu(\vec{k})}. v_1 \mu(\vec{k}) w_0, \\
d_{\text{step}} &\equiv \lambda n, w_2^{B(n)}, \vec{k}, w_3^{\mu(\vec{k})<n+1}. w_1 \vec{k} (\lambda \vec{\ell}, w_4^{\mu(\vec{\ell})<\mu(\vec{k})}. w_2 \vec{\ell} (e^{\mu(\vec{\ell})<n}[w_4, w_3])), \\
\\
d_{\text{prog}} &\equiv \lambda \vec{k}, u_1^{\forall \vec{\ell}(\mu(\vec{\ell})<\mu(\vec{k}) \to N(\vec{\ell}))}, u_2^{0<\mu(\vec{k})}. u \vec{k} d_{\text{div}_1}^{\mu(\vec{k})|a_1} d_{\text{div}_2}^{\mu(\vec{k})|a_2} u_2, \\
d_{\text{div}_i} &\equiv v_2 a_i q_i(\vec{k}) \mu(\vec{k}) r_i(\vec{k}) (e^{a_i = q_i(\vec{k})\mu(\vec{k}) + r_i(\vec{k})}[v_3, u_2]) d_{\not<_i}^{0<r_i(\vec{k}) \to \bot}, \\
d_{\not<_i} &\equiv \lambda u_{3,i}^{0<r_i(\vec{k})}. u_1 \vec{\ell}_i(\vec{k}) (e^{\mu(\vec{\ell}_i(\vec{k}))<\mu(\vec{k})}[v_5, u_2, v_4]) (e^{0<\mu(\vec{\ell}_i(\vec{k}))}[u_{3,i}, v_5]).
\end{aligned}$$

A–translation and term extraction.

Preparation. We let

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

where $\vec{k} \equiv k_1, k_2$. In the modified realizability interpretation every formula F is mapped to a finite list $\tau(F)$ of finite types over nat, such that if d derives F then d^{ets} has type $\tau(F)$. For instance $\tau(A) = \vec{\text{nat}} := (\text{nat}, \text{nat})$. Using some obvious abbreviations to denote finite sequences of types — for instance $\text{nat} \to \vec{\text{nat}} \to \vec{\text{nat}}$ abbreviates (ρ, ρ), where $\rho \equiv \text{nat} \to \text{nat} \to \text{nat} \to \text{nat}$ — we compute the types of some further formulas.

$$\begin{aligned}
&\tau(N(\vec{k})^A) = \tau(\bot^A) = \tau(A) = \vec{\text{nat}}, \\
&\tau(\text{Prog}^A) = \vec{\text{nat}} \to (\vec{\text{nat}} \to \vec{\text{nat}}) \to \vec{\text{nat}}, \\
&\tau(B(n)^A) = \tau(a|c^A) = \vec{\text{nat}} \to \vec{\text{nat}}.
\end{aligned}$$

If F neither contains $\bot$ nor $\cdot|\cdot$ then $\tau(F^A) = \tau(F) = ()$ and hence the sequence of extracted terms of a derivation of F is empty too. Furthermore note that $(\text{Ind}_{n,B(n)^A})^{\text{ets}} \equiv \vec{R}$ where $\vec{R} \equiv (R_1, R_2)$ are simultaneous primitive recursion operators of type

$$\vec{R} \colon (\vec{\text{nat}} \to \vec{\text{nat}}) \to (\text{nat} \to (\vec{\text{nat}} \to \vec{\text{nat}}) \to (\vec{\text{nat}} \to \vec{\text{nat}})) \to \text{nat} \to (\vec{\text{nat}} \to \vec{\text{nat}})$$

with

$$\begin{aligned}
&R_i \vec{y} \vec{f} 0 = y_i, \\
&R_i \vec{y} \vec{f} (z+1) = f_i z (R_1 \vec{y} \vec{f} z)(R_2 \vec{y} \vec{f} z).
\end{aligned}$$

Now we are prepared to compute the extracted terms. To make the relation between the derivations and their extracted terms as clear as possible we denote the finite list of object– (or function–) variables corresponding to the assumption w_i by $\vec{w}_i$ etc. According to step 3 and step 4 and the comment the extracted terms are given by

$$(d^{\mathrm{tr}})^{\mathrm{ets}} \equiv (d^{\mathrm{tr}}_{<-\mathrm{ind}})^{\mathrm{ets}}(d^{\mathrm{tr}}_{\mathrm{prog}})^{\mathrm{ets}}01$$

where

$$\begin{array}{rcl}
(d^{\mathrm{tr}}_{<-\mathrm{ind}})^{\mathrm{ets}} & \equiv & \lambda \vec{w}_1^{\vec{\mathrm{nat}}\to(\vec{\mathrm{nat}}\to\vec{\mathrm{nat}})\to\vec{\mathrm{nat}}}\lambda\vec{k}.\vec{R}(d^{\mathrm{tr}}_{\mathrm{base}})^{\mathrm{ets}}(d^{\mathrm{tr}}_{\mathrm{step}})^{\mathrm{ets}}(\mu(\vec{k})+1)\vec{k}, \\
(d^{\mathrm{tr}}_{\mathrm{base}})^{\mathrm{ets}} & \equiv & \lambda\vec{k}.d^{\mathrm{ets}}_{v_1}\mu(\vec{k}), \\
(d^{\mathrm{tr}}_{\mathrm{step}})^{\mathrm{ets}} & \equiv & \lambda n, \vec{w}_2^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}, \vec{k}.\vec{w}_1\vec{k}(\lambda\vec{\ell}.\vec{w}_2\vec{\ell}), \\
\\
(d^{\mathrm{tr}}_{\mathrm{prog}})^{\mathrm{ets}} & \equiv & \lambda\vec{k}, \vec{u}_1^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}.d^{\mathrm{ets}}_u\vec{k}(d^{\mathrm{tr}}_{\mathrm{div}_1})^{\mathrm{ets}}(d^{\mathrm{tr}}_{\mathrm{div}_2})^{\mathrm{ets}}, \\
(d^{\mathrm{tr}}_{\mathrm{div}_i})^{\mathrm{ets}} & \equiv & d^{\mathrm{ets}}_{v_2}a_iq_i(\vec{k})\mu(\vec{k})r_i(\vec{k})(\vec{u}_1\vec{\ell}_i(\vec{k})).
\end{array}$$

It remains to compute d^{ets}_u, $d^{\mathrm{ets}}_{v_1}$ and $d^{\mathrm{ets}}_{v_2}$.

$$d_u\colon \forall\vec{k}(((\mu(\vec{k})|a_1\to A)\to A)\to((\mu(\vec{k})|a_2\to A)\to A)\to 0<\mu(\vec{k})\to A),$$

$$d_u \equiv \lambda\vec{k}, u_4^{((\mu(\vec{k})|a_1\to A)\to A)}, u_5^{((\mu(\vec{k})|a_2\to A)\to A)}, u_6^{0<\mu(\vec{k})}.u_5(\lambda u_8^{\mu(\vec{k})|a_2}.u_4(\lambda u_7^{\mu(\vec{k})|a_1}.\exists^{*+}_A\vec{k}u_7u_8u_6)).$$

Here we used the $\exists^*$–introduction axiom

$$\exists^{*+}_A\colon\forall\vec{k}(\mu(\vec{k})|a_1\to\mu(\vec{k})|a_2\to 0<\mu(\vec{k})\to A)$$

with $(\exists^{*+}_A)^{\mathrm{ets}}\equiv\lambda\vec{k}.\vec{k}$. We obtain

$$d^{\mathrm{ets}}_u \equiv \lambda\vec{k}, \vec{u}_4^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}, \vec{u}_5^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}.\vec{u}_5(\vec{u}_4\vec{k}).$$

The computation of $d^{\mathrm{ets}}_{v_1}$ is easy:

$$\begin{array}{l}
d_{v_1}\colon\forall m(m<0\to A), \\
d_{v_1}\equiv\lambda m, u_9^{m<0}.\mathrm{Efq}_A(v_1mu_9), \\
d^{\mathrm{ets}}_{v_1}\equiv\lambda m\vec{0}
\end{array}$$

(instead of $\vec{0}$ any terms of type $\vec{\mathrm{nat}}$ would do). The control structure of the extracted program is introduced by $d^{\mathrm{ets}}_{v_2}$:

$$\begin{array}{l}
d_{v_2}\colon\forall a,q,c,r(a=qc+r\to(0<r\to A)\to(c|a\to A)\to A), \\
d_{v_2}\equiv\lambda a,q,c,r,u_{10}^{a=qc+r},u_{11}^{0<r\to A},u_{12}^{c|a\to A}.g_{0<r}u_{11}(\lambda u_{13}^{0<r\to\bot}.u_{12}(v_2aqcru_{10}u_{13})),
\end{array}$$

where

$$g_{0<r} \colon (0<r \to A) \to ((0<r \to \bot) \to A) \to A$$

is a derivation with extracted terms

$$g_{0<r}^{\mathrm{ets}} \equiv \lambda \vec{x}, \vec{y}.\mathbf{if} \quad 0<r \quad \mathbf{then} \quad \vec{x}^{\vec{\mathrm{nat}}} \quad \mathbf{else} \quad \vec{y}^{\vec{\mathrm{nat}}} \quad \mathbf{fi}.$$

Hence

$$d_{v_2}^{\mathrm{ets}} \equiv \lambda a, q, c, r, \vec{u}_{11}^{\vec{\mathrm{nat}}}, \vec{u}_{12}^{\vec{\mathrm{nat}}}.\mathbf{if} \quad 0<r \quad \mathbf{then} \quad \vec{u}_{11} \quad \mathbf{else} \quad \vec{u}_{12} \quad \mathbf{fi}.$$

The final program.

If we plug d_u^{ets}, $d_{v_1}^{\mathrm{ets}}$ and $d_{v_2}^{\mathrm{ets}}$ into the program pieces we obtain

$$(d^{\mathrm{tr}})^{\mathrm{ets}} \equiv (d^{\mathrm{tr}}_{<-\mathrm{ind}})^{\mathrm{ets}} (d^{\mathrm{tr}}_{\mathrm{prog}})^{\mathrm{ets}} 01$$

where

$$\begin{aligned}
(d^{\mathrm{tr}}_{<-\mathrm{ind}})^{\mathrm{ets}} &=_{\alpha\beta\eta} \quad \lambda w_1^{\vec{\mathrm{nat}}\to(\vec{\mathrm{nat}}\to\vec{\mathrm{nat}})\to\vec{\mathrm{nat}}} \lambda \vec{k}'.\vec{R}(\lambda \vec{k}.\vec{0})(\lambda n, \vec{w}_2^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}, \vec{k}.\vec{w}_1\vec{k}\vec{w}_2)(\mu(\vec{k}')+1)\vec{k}',\\
(d^{\mathrm{tr}}_{\mathrm{prog}})^{\mathrm{ets}} &=_{\beta} \quad \lambda \vec{k}, \vec{u}_1^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}.(d^{\mathrm{tr}}_{\mathrm{div}_2})^{\mathrm{ets}}((d^{\mathrm{tr}}_{\mathrm{div}_1})^{\mathrm{ets}}\vec{k}),\\
(d^{\mathrm{tr}}_{\mathrm{div}_i})^{\mathrm{ets}} &=_{\beta} \quad \lambda \vec{u}_{12}^{\vec{\mathrm{nat}}}.\mathbf{if} \quad 0<\vec{r}_i(\vec{k}) \quad \mathbf{then} \quad \vec{u}_1\vec{\ell}_i(\vec{k}) \quad \mathbf{else} \quad \vec{u}_{12} \quad \mathbf{fi}
\end{aligned}$$

and hence

$$\begin{aligned}
(d^{\mathrm{tr}}_{\mathrm{prog}})^{\mathrm{ets}} =_{\beta} \lambda \vec{k}, \vec{u}_1^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}. \quad & \mathbf{if} \quad 0<r_2(\vec{k}) \quad \mathbf{then} \quad \vec{u}_1\vec{\ell}_2(\vec{k}) \quad \mathbf{else}\\
& \mathbf{if} \quad 0<r_1(\vec{k}) \quad \mathbf{then} \quad \vec{u}_1\vec{\ell}_1(\vec{k}) \quad \mathbf{else}\\
& \vec{k} \quad \mathbf{fifi}.
\end{aligned}$$

Therefore we get, using the fact that $\mu(0,1)=a_2$,

$$\begin{aligned}
(d^{\mathrm{tr}})^{\mathrm{ets}} =_{\beta} \quad & \vec{R}(\lambda \vec{k}.\vec{0})\\
& (\lambda n, \vec{w}_2^{\vec{\mathrm{nat}}\to\vec{\mathrm{nat}}}, \vec{k}. \quad \mathbf{if} \quad 0<r_2(\vec{k}) \quad \mathbf{then} \quad \vec{w}_2\vec{\ell}_2(\vec{k}) \quad \mathbf{else}\\
& \qquad\qquad\qquad\qquad \mathbf{if} \quad 0<r_1(\vec{k}) \quad \mathbf{then} \quad \vec{w}_2\vec{\ell}_1(\vec{k}) \quad \mathbf{else}\\
& \qquad\qquad\qquad\qquad \vec{k} \quad \mathbf{fifi})\\
& (a_2+1)01
\end{aligned}$$

To make this algorithm more readable we may write it in the form $(d^{\mathrm{tr}})^{\mathrm{ets}} = \vec{h}(a_2+1,0,1)$, where

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

As an example let us try out the extracted algorithm to compute coefficients k_1, k_2 such that $\gcd(66,27) = |k_1 \cdot 66 - k_2 \cdot 27|$.

$$
\begin{array}{ll}
\vec{h}(28,0,1) & \mu(0,1) = 27 \\
 & 0 < r_1 = 12 \quad q_1 = 2 \\
 & \underbrace{q_1 k_1}_{0} \pm 1, \quad \underbrace{q_1 k_2}_{2} \qquad -1, \text{ if } k_2 a_2 < \underbrace{k_1 a_1}_{0} \quad \text{No} \\
 & 1, \quad 2 \\
\vec{h}(27,1,2) & \mu(1,2) = 12 \\
 & 0 < r_1 = 6 \quad q_1 = 5 \\
 & \underbrace{q_1 k_1}_{5 \cdot 1} \pm 1, \quad \underbrace{q_1 k_2}_{5 \cdot 2} \qquad -1, \text{ if } \underbrace{k_2 a_2}_{2 \cdot 27} < \underbrace{k_1 a_1}_{1 \cdot 66} \quad \text{Yes} \\
 & 4, \quad 10 \\
\vec{h}(26,4,10) & \mu(4,10) = |4 \cdot 66 - 10 \cdot 27| = |264 - 270| = 6 \\
 & 6|66 \\
 & 0 < r_2 = 3 \quad q_2 = 4 \\
 & \underbrace{q_2 k_1}_{4 \cdot 4}, \quad \underbrace{q_2 k_2}_{4 \cdot 10} \pm 1 \qquad -1, \text{ if } \underbrace{k_1 a_1}_{4 \cdot 66 = 264} < \underbrace{k_2 a_2}_{10 \cdot 27 = 270} \quad \text{Yes} \\
 & 16, \quad 39 \\
\vec{h}(25,16,39) & \mu(16,39) = |16 \cdot 66 - 39 \cdot 27| = |1056 - 1053| = 3 \\
 & 3|66 \\
 & 3|27 \\
 & \text{Result: } 16, 39
\end{array}
$$

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.

- 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 least element principle (or, equivalently, $<$–induction) and the relation symbol $<$ is not critical. Because then in the extracted terms of $<$–induction, the step $(d^{\mathrm{tr}}_{\mathrm{step}})^{\mathrm{ets}} \equiv \lambda n, \vec{w}_2^{\,\vec{\mathrm{nat}} \to \vec{\mathrm{nat}}}, \vec{k}.\vec{w}_1 \vec{k}(\lambda \vec{\ell}.\vec{w}_2 \vec{\ell})$ has in its kernel no free occurrence of n.

– If one removes n according to the previous remark it becomes clear that our gcd algorithm is similar to Euklid's. The only difference lies in the fact that we have kept a_1, a_2 fixed in our proof whereas Euklid 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).

– There is an interesting phenomenon which may occur if we extract a program from a classical proof which uses the least element principle. Consider as a simple example the wellfoundedness of $<$,

$$\forall g^{\mathrm{nat} \to \mathrm{nat}} \exists k (g(k+1) < g(k) \to \bot).$$

If one formalizes the classical proof "choose k such that $g(k)$ is minimal" and extracts a program one might expect that it computes a k such that $g(k)$ is minimal. But this is impossible! In fact the program computes the least k such that $g(k+1) < g(k) \to \bot$ instead. This discrepancy between the classical proof and the extracted program didn't show up in our gcd example since there was only one $c = \mu(k) > 0$ such that c divides a_1 and a_2, whereas in the example above there may be different $c = g(k)$ such that $g(k+1) < g(k) \to \bot$.

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)))
```

Here 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 nat one might as well pick the constant 0 (as we did in the text).

Acknowledgements. We would like to thank Monika Seisenberger and Felix Joachimski for implementing the refined A–translation and doing the gcd example in the MINLOG system.

References

[1] Ulrich Berger and Helmut Schwichtenberg. Program development by proof transformation. In H. Schwichtenberg, editor, *Proof and Computation*, volume 139 of *Series F: Computer and Systems Sciences*, pages 1–45. NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 20 – August 1, 1993, Springer Verlag, Berlin, Heidelberg, New York, 1995.

[2] Ulrich Berger and Helmut Schwichtenberg. Program extraction from classical proofs. In D. Leivant, editor, *Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, IN, USA, October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 77–97. Springer Verlag, Berlin, Heidelberg, New York, 1995.

[3] 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.

[4] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North–Holland, Amsterdam, 1959.