Efficient Program Extraction in Elementary Number Theory using the Proof Assistant Minlog

Franziskus Wiesnet

TU Wien
This research was funded by the Austrian Science Fund (FWF) **10.55776/ESP576**.

TYPES 2025 – June 12, 2025

Computational Formulas

General Examples

Computational Formulas

General Examples

The elementary **computationally relevant** formulas *in our context* are of the form

$$A \vee B \qquad \text{and} \qquad \exists_x A.$$

- $A \to B$ is computationally relevant iff B is.
- $A \wedge B$ is computationally relevant iff at least one of the conjunction parts is computationally relevant.
- $\forall_x A$ is computationally relevant iff A is.

FWF Österreichischer Wissenschaftsfonds

Computational Formulas

General Examples

The elementary **computationally relevant** formulas *in our context* are of the form

$$A \vee B \qquad \text{and} \qquad \exists_x A.$$

- $A \to B$ is computationally relevant iff B is.
- $A \wedge B$ is computationally relevant iff at least one of the conjunction parts is computationally relevant.
- $\forall_x A$ is computationally relevant iff A is.

Note:
Equalities and boolean terms are non-computational.
Examples are $\neg A$ for any formula A;
$A \vee^b B$, $A \wedge^b B$ for boolean terms A and B, as well as
$n = gcd(n, m)$, $\exists_i^{< \mathsf{S}\, m} i \cdot n = m$, ...

Formal Program Extraction from Proofs

Compact Overview

Formal Program Extraction from Proofs

Compact Overview

Let a computationally relevant formula A be given.

FWF Österreichischer Wissenschaftsfonds

Formal Program Extraction from Proofs

Compact Overview

Let a computationally relevant formula A be given.

From A one can calculate the **type** $\tau(A)$ and the **realiser predicate** $A^{\mathbf{r}}$ of A.

Formal Program Extraction from Proofs

Compact Overview

Let a computationally relevant formula A be given.

From A one can calculate the **type** $\tau(A)$ and the **realiser predicate** $A^{\mathbf{r}}$ of A.

From a formal proof M of A one can construct the **extracted term** $\mathrm{et}(M)$ of type $\tau(A)$.

Formal Program Extraction from Proofs

Compact Overview

Let a computationally relevant formula A be given.

From A one can calculate the **type** $\tau(A)$ and the **realiser predicate** $A^{\mathbf{r}}$ of A.

From a formal proof M of A one can construct the **extracted term** $\mathrm{et}(M)$ of type $\tau(A)$.

If A and M are **r**-free, the **soundness theorem** states

$$A^{\mathbf{r}}\,\mathrm{et}(M).$$

Formal Program Extraction from Proofs

Compact Overview

Let a computationally relevant formula A be given.

From A one can calculate the **type** $\tau(A)$ and the **realiser predicate** $A^{\mathbf{r}}$ of A.

From a formal proof M of A one can construct the **extracted term** $\mathrm{et}(M)$ of type $\tau(A)$.

If A and M are **r**-free, the **soundness theorem** states

$$A^{\mathbf{r}}\,\mathrm{et}(M).$$

The construction of $\mathrm{et}(M)$ is implemented in Minlog.

FWF Österreichischer Wissenschaftsfonds

Minlog

Proof Assistant

Minlog

Proof Assistant

- Developed in the 1990s by the Logic Group at Ludwig Maximilian University of Munich, led by **Helmut Schwichtenberg**.
- Implemented in the programming language **Scheme**.
- Based on the **Theory of Computational Functionals**, which builds on partial continuous functionals and information systems.
- Uses tactic scripts that are **closely aligned with traditional textbook-style proofs**.
- Enables **formal program extraction** from proofs, with output in **Haskell**.
- Especially well-suited for **constructive analysis**.

Minlog
Proof Assistant

- Developed in the 1990s by the Logic Group at Ludwig Maximilian University of Munich, led by **Helmut Schwichtenberg**.
- Implemented in the programming language **Scheme**.
- Based on the **Theory of Computational Functionals**, which builds on partial continuous functionals and information systems.
- Uses tactic scripts that are **closely aligned with traditional textbook-style proofs**.
- Enables **formal program extraction** from proofs, with output in **Haskell**.
- Especially well-suited for **constructive analysis**.

Novelty: Number theory in Minlog with program extraction.

Numeric types

Nat vs. Pos

Numeric types

Nat vs. Pos

The **natural numbers** $\mathbb{N}$ are given by

$$0 : \mathbb{N}, \quad \mathsf{S} : \mathbb{N} \to \mathbb{N}.$$

FWF Österreichischer Wissenschaftsfonds

Numeric types

Nat vs. Pos

The **natural numbers** $\mathbb{N}$ are given by

$$0 : \mathbb{N}, \quad \mathsf{S} : \mathbb{N} \to \mathbb{N}.$$

The **positive binary numbers** $\mathbb{P}$ are given by

$$1 : \mathbb{P}, \quad \mathsf{S}_0 : \mathbb{P} \to \mathbb{P}, \quad \mathsf{S}_1 : \mathbb{P} \to \mathbb{P}.$$

Numeric types

Nat vs. Pos

The **natural numbers** $\mathbb{N}$ are given by

$$0 : \mathbb{N}, \quad \mathsf{S} : \mathbb{N} \to \mathbb{N}.$$

The **positive binary numbers** $\mathbb{P}$ are given by

$$1 : \mathbb{P}, \quad \mathsf{S}_0 : \mathbb{P} \to \mathbb{P}, \quad \mathsf{S}_1 : \mathbb{P} \to \mathbb{P}.$$

There are canonical functions

$$\mathsf{PosToNat} : \mathbb{P} \to \mathbb{N} \quad \text{and} \quad \mathsf{NatToPos} : \mathbb{N} \to \mathbb{P},$$

with $\mathsf{NatToPos}(0) := 1$.

FWF Österreichischer Wissenschaftsfonds

GCD algorithm

Euclidean algorithm

GCD algorithm

Euclidean algorithm

Definition

The greatest common divisor on the natural numbers is defined by the following rules:

$$
\begin{aligned}
\mathrm{gcd}(0, n) &:= n \\
\mathrm{gcd}(m, 0) &:= m \\
\mathrm{gcd}(\mathrm{S}\, m, \mathrm{S}\, n) &:= \begin{cases} \mathrm{gcd}(\mathrm{S}\, m, n - m) & \text{if } m < n \\ \mathrm{gcd}(m - n, \mathrm{S}\, n) & \text{otherwise} \end{cases}
\end{aligned}
$$

FWF Österreichischer Wissenschaftsfonds

GCD algorithm

Stein's algorithm

GCD algorithm

Stein's algorithm

Definition

The greatest common divisor on the positive binary numbers is defined by the following rules:

$$
\begin{aligned}
\mathrm{gcd}(1, p) &:= 1 \\
\mathrm{gcd}(\mathrm{S}_0\, p, 1) &:= 1 \\
\mathrm{gcd}(\mathrm{S}_0\, p, \mathrm{S}_0\, q) &:= \mathrm{S}_0(\mathrm{gcd}(p, q)) \\
\mathrm{gcd}(\mathrm{S}_0\, p, \mathrm{S}_1\, q) &:= \mathrm{gcd}(p, \mathrm{S}_1\, q) \\
\mathrm{gcd}(\mathrm{S}_1\, p, 1) &:= 1 \\
\mathrm{gcd}(\mathrm{S}_1\, p, \mathrm{S}_0\, q) &:= \mathrm{gcd}(\mathrm{S}_1\, p, q) \\
\mathrm{gcd}(\mathrm{S}_1\, p, \mathrm{S}_1\, q) &:= \begin{cases} \mathrm{gcd}(\mathrm{S}_1\, p, q - p) & \text{if } p < q \\ \mathrm{gcd}(p - q, \mathrm{S}_1\, q) & \text{if } p > q \\ \mathrm{S}_1\, p & \text{otherwise.} \end{cases}
\end{aligned}
$$

Versions of Bézout's Identity

FWF Österreichischer Wissenschaftsfonds

Versions of Bézout's Identity

Theorem (Bézout's identity)

For two integers $a, b \in \mathbb{Z}$ *there are* $u, v \in \mathbb{Z}$ *with* $\gcd(a, b) = ua + vb$.

FWF Österreichischer Wissenschaftsfonds

Versions of Bézout's Identity

Theorem (Bézout's identity)

For two integers $a, b \in \mathbb{Z}$ there are $u, v \in \mathbb{Z}$ with $\mathrm{gcd}(a, b) = ua + vb$*.*

Theorem (Bézout's identity on natural numbers)

$$\forall_{n,m} \exists_{l_0} \exists_{l_1}.\ \mathrm{gcd}(n, m) + l_0 \cdot n = l_1 \cdot m \vee \mathrm{gcd}(n, m) + l_0 \cdot m = l_1 \cdot n$$

Versions of Bézout's Identity

Theorem (Bézout's identity)

For two integers $a, b \in \mathbb{Z}$ *there are* $u, v \in \mathbb{Z}$ *with* $\gcd(a, b) = ua + vb$.

Theorem (Bézout's identity on natural numbers)

$$\forall_{n,m} \exists_{l_0} \exists_{l_1}.\ \gcd(n, m) + l_0 \cdot n = l_1 \cdot m \lor \gcd(n, m) + l_0 \cdot m = l_1 \cdot n$$

Theorem (Positive binary version of Bézout's identity)

$$\begin{aligned} \forall_{p_0, p_1}. \quad & \exists_q\ q \cdot p_0 = p_1 \\ \lor \quad & \exists_q\ q \cdot p_1 = p_0 \\ \lor \quad & \exists_{q_0} \exists_{q_1}\ \gcd(p_0, p_1) + q_0 \cdot p_0 = q_1 \cdot p_1 \\ \lor \quad & \exists_{q_0} \exists_{q_1}\ \gcd(p_0, p_1) + q_1 \cdot p_1 = q_0 \cdot p_0 \end{aligned}$$

FWF Österreichischer Wissenschaftsfonds

The Fundamental Theorem of Arithmetic

Existence of the Prime Factorisation

FWF Österreichischer Wissenschaftsfonds

The Fundamental Theorem of Arithmetic

Existence of the Prime Factorisation

Theorem

$$\forall_p \exists_{ps} \exists_m .\ \mathbf{P}_m(ps) \wedge \prod_{i<m} ps(i) = p$$

The Fundamental Theorem of Arithmetic

Existence of the Prime Factorisation

Theorem

$$\forall_p \exists_{ps} \exists_m .\ \mathbf{P}_m(ps) \land \prod_{i<m} ps(i) = p$$

Notation.

$ps : \mathbb{N} \to \mathbb{P}$

$\prod_{i<0} ps(i) := 1, \quad \prod_{i<n+1} ps(i) := \left(\prod_{i<n} ps(i)\right) \cdot ps(n)$

FWF Österreichischer Wissenschaftsfonds

The Fundamental Theorem of Arithmetic

Uniqueness of the Prime Factorisation

FWF Österreichischer Wissenschaftsfonds

The Fundamental Theorem of Arithmetic

Uniqueness of the Prime Factorisation

Theorem

$$\forall_{n,m,ps,qs}. \quad \mathbf{P}_n(ps) \to \mathbf{P}_m(qs) \to \prod_{i<n} ps(i) = \prod_{i<m} qs(i) \to$$
$$n = m \;\wedge\; \exists_f \exists_g \left(\mathsf{Pms}_n(f,g) \wedge \forall_{i<n} ps(f\ i) = qs(i)\right).$$

FWF Österreichischer Wissenschaftsfonds

The Fundamental Theorem of Arithmetic

Uniqueness of the Prime Factorisation

Theorem

$$\forall_{n,m,ps,qs}.\quad \mathbf{P}_n(ps) \to \mathbf{P}_m(qs) \to \prod_{i<n} ps(i) = \prod_{i<m} qs(i) \to$$
$$n = m \ \wedge\ \exists_f \exists_g \left(\mathsf{Pms}_n(f,g) \wedge \forall_{i<n} ps(f\ i) = qs(i)\right).$$

Notation.
$f, g : \mathbb{N} \to \mathbb{N}$,
$\mathsf{Pms}_n(f,g) :\Leftrightarrow \qquad f \circ g = g \circ f = \mathrm{id} \quad \wedge \quad \forall_{i \geq n} f(i) = i$

Fermat Factorisation

Motivation

Fermat Factorisation

Motivation

For natural numbers $a > b$, we have

$$a^2 - b^2 = (a + b)(a - b).$$

Fermat Factorisation

Motivation

For natural numbers $a > b$, we have

$$a^2 - b^2 = (a + b)(a - b).$$

Idea: Let n be given, search for m such that $m^2 - n$ is a square, say $m^2 - n = l^2$, then $n = (m + l)(m - l)$. If $m - l > 1$, this yields a non-trivial factorisation of n.

Fermat Factorisation

Motivation

For natural numbers $a > b$, we have

$$a^2 - b^2 = (a + b)(a - b).$$

Idea: Let n be given, search for m such that $m^2 - n$ is a square, say $m^2 - n = l^2$, then $n = (m + l)(m - l)$. If $m - l > 1$, this yields a non-trivial factorisation of n.

Question: Is this search always successful for composite numbers? And is this search bounded?

Fermat Factorisation

Motivation

For natural numbers $a > b$, we have

$$a^2 - b^2 = (a + b)(a - b).$$

Idea: Let n be given, search for m such that $m^2 - n$ is a square, say $m^2 - n = l^2$, then $n = (m + l)(m - l)$. If $m - l > 1$, this yields a non-trivial factorisation of n.

Question: Is this search always successful for composite numbers? And is this search bounded?

Answer: Yes, for odd numbers.

Fermat Factorisation

Fermat Factorisation

Lemma

Let $p = 2q + 1 > 1$ *be an odd number, that is not a perfect square. If* p *is a composite number, then* $p = p_1^2 - p_0^2$ *with* $p_0 < p_1 < q$.

Fermat Factorisation

Lemma

Let $p = 2q + 1 > 1$ be an odd number, that is not a perfect square. If p is a composite number, then $p = p_1^2 - p_0^2$ with $p_0 < p_1 < q$.

Proof.

Let $p = q_0 \cdot q_1$ with $q_0 < q_1$. As p is odd, also q_0, q_1 must be odd, hence

$$q_0 = 2r_0 + 1 \quad \text{and} \quad q_1 = 2r_1 + 1.$$

We define

$$p_0 := \frac{q_1 - q_0}{2} = r_1 - r_0 > 0, \qquad p_1 := \frac{q_1 + q_0}{2} = r_1 + r_0 + 1 > 0.$$

Then

$$p_1^2 - p_0^2 = q_0 \cdot q_1 = p.$$

Clearly, $p_0 < p_1$. Furthermore, $q_0, q_1 > 2$ and $q_0, q_1 < q$, therefore $p_1 < q$. $\square$

Fermat Factorisation

Theorem

Let $p > 1$ be a natural number. Then p is either prime or there are $q_0, q_1 > 1$ with $p = q_0 \cdot q_1$.

Fermat Factorisation

Theorem

Let $p > 1$ be a natural number. Then p is either prime or there are $q_0, q_1 > 1$ with $p = q_0 \cdot q_1$.

Proof.

Without loss of generality, we may assume that $p = 2q + 1$ is odd, not a perfect square, and $p \notin \{3, 5\}$.
From $p > 5$ we get $\lfloor \sqrt{p} \rfloor < q$ (!), and therefore define

$$l := \mu_{\lfloor \sqrt{p} \rfloor \leq i < q}(\mathrm{IsSq}(i^2 - p)).$$

If $l = q$ there is not $i < q$ with $\mathrm{IsSq}(i^2 - p)$ and therefore p is prime by the lemma above. If $l < q$, we have $l^2 - p = r^2$ for $r = \lfloor \sqrt{l^2 - p} \rfloor$. Therefore $p = l^2 - r^2 = (l + r)(l - r)$. Furthermore $l - r \neq 1$ (!), which completes the proof. $\square$

Thank you!

Questions are welcome.