

Essential Incompleteness of Arithmetic Verified by Coq

Russell O'Connor

Radboud University Nijmegen

University of California at Berkeley

DIAMANT

November 18, 2005

Essential Incompleteness of Arithmetic Verified by Coq

1. Proofs

- 1. Informal

- 2. Formal

2. The Incompleteness Theorem

- 1. The statement

- 2. Coding of data as numbers

- 3. Representation of primitive recursive functions

3. Results and Conclusions

- 1. Statistics

Mathematical Proofs

- Lemma: For a and b in \mathbb{N} , if $ab = 0$ then $a = 0$ or $b = 0$.
- Proof by induction on a .
 - If $a = 0$ then we are done.
 - If $a = a' + 1$ then we have $(a' + 1)b = 0$.
 - So $a'b + b = 0$ (by definition of multiplication).
 - So $a'b = 0$ and $b = 0$, and we are done (by a lemma about addition).

Formal Language

- Formal Language for Statements
 - Logical Symbols
 - $\forall, \exists, \Rightarrow, \wedge, \vee, \neg, x_0, \dots$
 - Non-logical Symbols
 - $0, S, +, \times.$
- Example
 - $\forall x_0. \forall x_1. x_0 \times x_1 = 0 \Rightarrow x_0 = 0 \vee x_1 = 0$

Formal Proofs

- Deduction Rules

- Logical Axiom Schemas

$$\frac{}{A \vdash A} \quad \frac{}{\vdash x = x} \quad \frac{}{\vdash (\neg A \Rightarrow \neg B) \Rightarrow (B \Rightarrow A)} \quad \dots$$

- Logical Rule Schemas

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Delta \vdash A}{\Gamma \cup \Delta \vdash B} \quad \frac{\Gamma \vdash A}{\Gamma \vdash \forall x A} \quad x \notin FV(\Gamma)$$

- Non-logical Axioms (Schemas)

- $\forall x_0, \forall x_1, x_0 \times S x_1 = x_0 \times x_1 + x_0$
 - $\forall x_0, \dots, \forall x_n, A[x_i/0] \Rightarrow (\forall x_i, A \Rightarrow A[x_i/S x_i]) \Rightarrow \forall x_i, A$
 - ...

Proof Theory

- Reasoning can be made formal.
- Formal Proofs form a mathematical structure.
- The study of this structure is **proof theory**.
- Material Includes:
 - Cut Elimination
 - Ordinal Analysis
 - Incompleteness Theorems
 - ...

Incompleteness & The Halting Problem

- A computer program can, **in principle**, check deductions.
- Computers programs can be represented in the language of arithmetic.
 - Includes recursive functions
 - A program halting is a mathematical statement
- If all true statements are provable the you can solve the halting problem.

Computer Verification

- Reasoning can be made formal.
- A computer program implement a formal proof system and check deductions.
- Coq is such a system.
 - Coq uses an intensional constructive type theory.
- One can do proof theory in a computer system.

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending **NN** such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

A finite axiomatization of:
 $0, S, +, \times, <$
without induction

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.



This is provability in the internal system.
Proofs here are classical.

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

Inductive data types for:

- Terms
- First order formulas
- Proofs

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

Recursive functions for:

- Free variables
- Substitution

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a **sentence** G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.



A formula with no free variables

Gödel-Rosser Incompleteness Theorem

For any decidable **axiom system** T extending
NN such that T can express its own axioms,
there exists a sentence G such that if either
 $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

Formula \Rightarrow Prop
(think Formula \Rightarrow type)

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

For every formula φ , $T \vdash \varphi$

Gödel-Rosser Incompleteness Theorem

For any **decidable** axiom system T extending NN such that T can express its own axioms, there exists a sentence G such that if either $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

- For every formula φ , $\varphi \in T$ or $\varphi \notin T$
- Classically this is trivially true
- This is not the recursive requirement.

Gödel-Rosser Incompleteness Theorem

For any decidable axiom system T extending
NN such that T can express its own axioms,
there exists a sentence G such that if either
 $T \vdash G$ or $T \vdash \neg G$ then T is inconsistent.

This replaces the requirement that T is recursive
This is weaker than T being recursive

Gödel-Rosser

Incompleteness Theorem

Incompleteness

: forall T : System,

Included Formula LNN T ->

RepresentsInSelf T ->

DecidableSet Formula T ->

exists f : Formula,

Sentence f /\

(SysPrf T f \/ SysPrf T (notH f))

-> Inconsistent LNN T)

Gödel-Rosser Incompleteness Theorem

Rosser's Incompleteness

```
: forall T : System,  
  Included (fol.Formula LNN) NN T ->  
  forall (repT : Formula) (v0 : nat),  
  (forall v : nat, In v (freeVarFormula LNN repT) -> v = v0) ->  
  (forall f : Formula,  
    mem (fol.Formula LNN) T f ->  
    SysPrf T (substituteFormula LNN repT v0 (natToTerm (codeFormula f)))) ->  
  (forall f : Formula,  
    ~ mem (fol.Formula LNN) T f ->  
    SysPrf T  
      (notH (substituteFormula LNN repT v0 (natToTerm (codeFormula f)))) ->  
  (forall x : Formula,  
    mem (fol.Formula LNN) T x \/ ~ mem (fol.Formula LNN) T x) ->  
  exists f : Formula,  
    (forall v : nat, ~ In v (freeVarFormula LNN f)) /\  
    (SysPrf T f \/ SysPrf T (notH f) -> Inconsistent LNN T)
```

T Can Express Its Own Axioms

T can express a set S if there exists a formula with one free variable $\varphi(x)$ such that:

- If $n \in S$ then $T \vdash \varphi(n)$
- If $n \notin S$ then $T \vdash \neg\varphi(n)$

T Can Express Its Own Axioms

T can express its own axioms if there exists a formula with one free variable $\varphi(x)$ such that:

- If $\psi \in T$ then $T \vdash \varphi(\ulcorner \psi \urcorner)$
- If $\psi \notin T$ then $T \vdash \neg \varphi(\ulcorner \psi \urcorner)$

$\ulcorner \psi \urcorner$ is ψ coded by a number written as a closed term.

Coding

- Extensive use of Cantor Pairing Function

$$cPair(a, b) = a + \sum_{i=1}^{a+b} i$$

- Formulas are coded by giving each symbol a number and pairing it with its recursively coded arguments.
- No prime number decomposition theorem is needed.

Primitive Recursive Functions

- Inductive data type for primitive recursive expressions
- Evaluation function for primitive recursive expressions
- Every primitive recursive function is representable in NN
 - Requires Chinese remainder theorem and Gödel's beta function

Primitive Recursive Functions

- Inductive definition of primitive recursive functions
- Expressions for primitive recursive functions
- Evaluation of primitive recursive functions
- Every primitive recursive function is **representable in NN**
 - Requires Chinese remainder theorem and Gödel's beta function

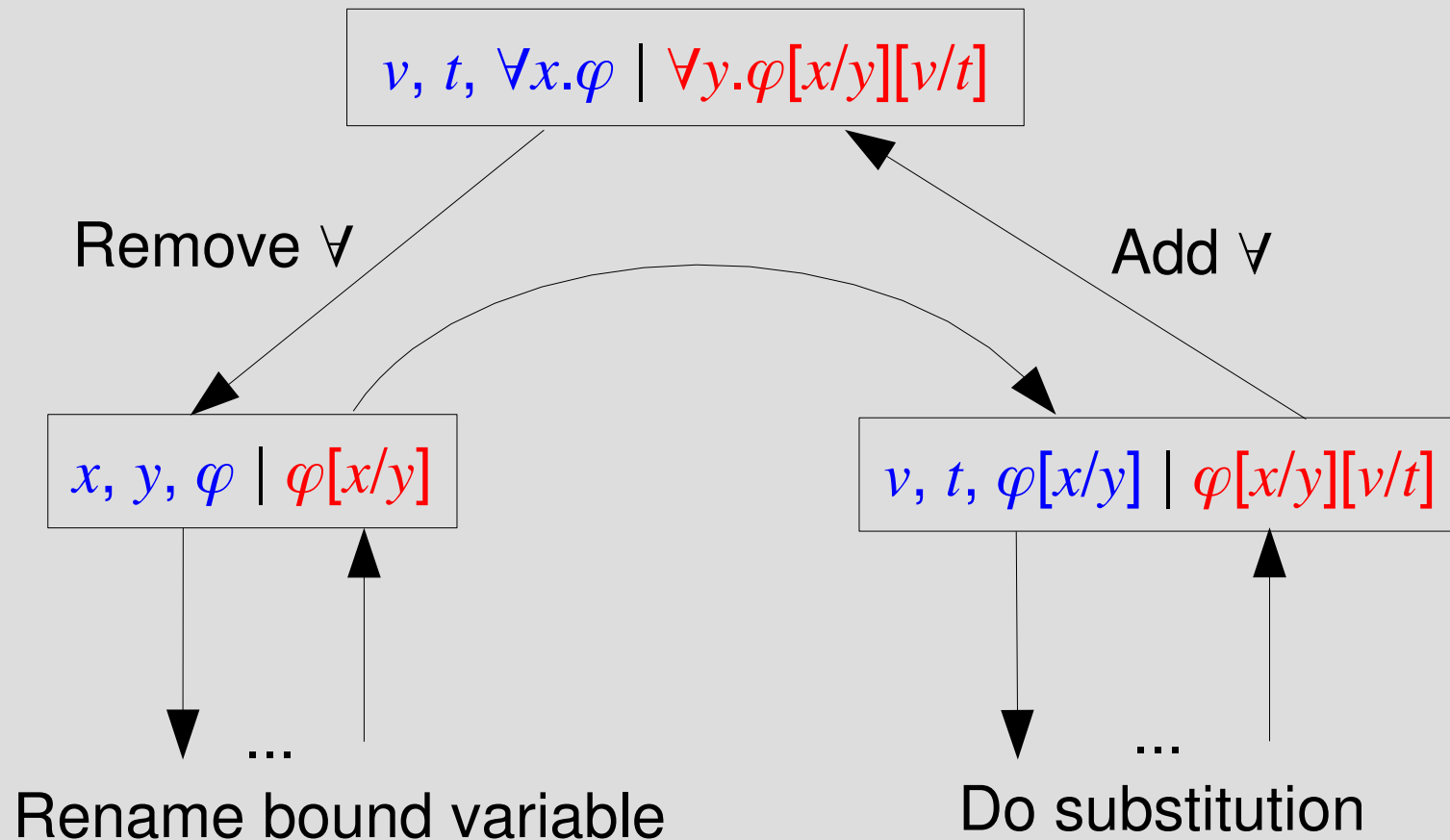
An n -ary function, f , is representable by φ in T if

$$T \vdash \varphi(x_0, \ulcorner a_1 \urcorner, \dots, \ulcorner a_n \urcorner) \Leftrightarrow x_0 = \ulcorner f(a_1, \dots, a_n) \urcorner$$

Substitution Is Primitive Recursive

- Recursion on depth does not work.
 - Consider $(\forall x.\varphi)[v/t]$. It becomes $\forall y.\varphi[x/y][v/t]$.
- Course-of-values recursion does not work.
 - Recursive call's input may have a larger code than the original input.
- Instead consider the trace of the computation.
 - A primitive recursive function can verify a trace.
 - Do a bounded search for the correct trace.

Substitution Is Primitive Recursive



Corollaries

- Gödel Sentence with ω -consistency
- If PA is consistent then PA is essentially incomplete.

Corollaries

- Gödel Sentence with ω -consistency
- If **PA is consistent** then PA is essentially incomplete.

Proved that PA is consistent

Corollaries

- Gödel Sentence with ω -consistency
- PA is essentially incomplete.

Corollaries

- Gödel Sentence with ω -consistency
- PA is incomplete.

Machine Verified Proofs

- 1986 – Shankar – Boyer-Moore
 - Incompleteness for any finite extension of Z2 (hereditarily finite set theory)
- 2003 – O’Connor – Coq
 - Incompleteness for any “nice” extension of NN
- 2004 – Harrison – HOL-Light
 - Incompleteness for any Σ_1 -complete system with Σ_1 -definable axioms

The Good

- No changes in Coq were made.
- Dependent types
 - Ensures that all formulas are well-formed.
 - Some modules are abstract over the language.
- A determined novice Coq user can prove difficult theorems.

The Bad

- Dependent types are hard to work with.
- Program Extraction will not produce the Gödel sentence.
 - Neither possible nor practical
- Used a lot of cut and paste
 - This is a problem in other programming languages as well.

The Future

- Gödel's second incompleteness theorem
 - Formalize the Hilbert-Bernays-Löb derivability conditions:
 1. If $PA \vdash \varphi$ then $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \urcorner)$
 2. $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \urcorner)$
 3. $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \Rightarrow \psi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \psi \urcorner)$

The Future

- Gödel's second incompleteness theorem
 - Formalize the Hilbert-Bernays-Löb derivability conditions:
 1. If $PA \vdash \varphi$ then $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \urcorner)$
 2. $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \urcorner)$
 3. $PA \vdash \text{Pr}_{PA}(\ulcorner \varphi \Rightarrow \psi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \varphi \urcorner) \Rightarrow \text{Pr}_{PA}(\ulcorner \psi \urcorner)$

The 2nd condition is the hardest to prove

Statistics

- Took me approximately 16 months
 - Shankar took about 18 months
- Proof Size
 - 46 files
 - 7 036 lines of specifications
 - 37 906 lines of proof
 - 1 267 747 total characters.
- Information Content
 - 146 008 bytes (`gzip -9`)