Autoformalization of Gödel Second Incompleteness Theorem

Thierry Coquand

Göteborg, 26/05/2026

Why Gödel Second Incompleteness Theorem?

Good test for autoformalization

Non trivial example: only one formalised proof, by L. Paulson in 2013, using hereditary finite sets instead of a direct arithmetic presentation and using a non standard formulation (nominal logic)

Always wanted to understand the proof in all details

If you ask a LLM about the details, you may not get correct answers

Gödel himself never wrote the part II where he was supposed to write these details:
part II of my work exists only in a realm of ideas. (1932)

Sources and Studies in the History of Mathematics
and Physical Sciences

Jan von Plato

Can Mathematics Be Proved Consistent?

Gödel's Shorthand Notes & Lectures on Incompleteness

What version?

H.E. Rose *On the consistency and undecidability of recursive arithmetic*, 1961

J. R. Guard *Lecture Notes on Recursive Arithmetic*, 1963

A. Church *Basic recursive arithmetic*, 1957

Based on the approach of Skolem, 1923, primitive recursive arithmetic, with quantifier free statements (implicitly universally quantified)

What version?

We follow Church's basic recursive arithmetic formulation of Guard. non-logical symbols are a constant O, a unary symbol s, and finitely many additional function symbols/combinators

$$z, id, snd, C, R$$

together with the binary identity $=$. Variables are $x_0, x_1, \ldots$.

What version?

$$\neg(s(O) = O)$$

$$z(t) = O$$

$$id(t) = t$$

$$snd(a, b) = b$$

$$C(g, f_1, f_2)(t) = g(f_1(t), f_2(t))$$

$$R(f, g_1, g_2)(x_0, O) = f(x_0)$$

$$R(f, g_1, g_2)(x_0, s(x_1)) = g_1\big(g_2(x_0, x_1), R(f, g_1, g_2)(x_0, x_1)\big)$$

What version?

$$x_0 = x_1 \to (x_0 = x_2 \to x_1 = x_2)$$
$$x_0 = x_1 \to f(x_0) = f(x_1)$$
$$x_0 = x_1 \to g(x_0, x_2) = g(x_1, x_2)$$
$$x_0 = x_1 \to g(x_2, x_0) = g(x_2, x_1)$$

What version?

$$A \to (B \to A)$$

$$\big(A \to (B \to C)\big) \to \big((A \to B) \to (A \to C)\big)$$

$$(\neg A \to \neg B) \to (B \to A)$$

Inference rules?

$$\frac{A \to B \qquad A}{B} \qquad \frac{A}{A[t/x_0]} \qquad \frac{A[0/x_0] \qquad A \to A[s(x_0)/x_0]}{A}$$

We write $Deriv(A)$ if we can derive A from these axioms and inference rules

Nice system for Gödel's theorem, but it would be difficult/challenging to do it "by hand"

The proof of Gödel I

Coding of the system in itself.

First coding of pairing and projection, done in Church's paper.

Then coding of course-of-value recursion; standard but does not follow usual textbooks/Gödel or Guard (prime numbers). Use hints from chatgpt.

This experiment is not "blind" autoformalization; we interact with the systems, with hints from another LLM or suggestions to read other relevant papers

Write then code for axioms and inference rules, and a function $thmT$ that enumerates the code of derived formulae of T, so $thmT(O),\ thmT(s(O)), \ldots$ are exactly the code of derived formulae.

The proof of Gödel I

In particular this involves the coding of the modus-ponens rule

We need to build a mp function (using the primitive functions and combinators) so that

$$mp(code(A \to B), code(A)) = code(B)$$

However, Guard does not say what happens in general!

It is crucial that $thmT(n)$ is a code of a *valid* formula, so we should add

$$mp(x, y) = code(O = O)$$

if x, y is not of the right form $code(A \to B), code(A)$

This was not found by Claude; even this however is *not enough* for Gödel II.

The proof of Gödel I

We need also to build a code of the substitution function in order to encode the two other derivation rules

The specification in Guard's paper is $sbt(\langle n, code(t)\rangle, code(A)) = code(A[t/x_n])$

There also, it is *not enough* for Gödel II

The proof of Gödel I

Once this is done, we build by diagonalisation a formula G such that G is equivalent to $thmT(x) \neq code(G)$

Intuitively, G is equivalent to the fact that G is not provable

It is then direct to show that $Deriv(G)$ implies $Deriv(O = s(O))$. This is Gödel I.

The proof of Gödel II

We have seen that we can prove $Deriv(G)$ implies $Deriv(O = s(O))$

The proof of Gödel II consists in *internalizing* this: we build a function g such that $Deriv(thmT(x) = code(G) \rightarrow thmT(g(x)) = code(O = s(O)))$

For this, we have a *key* point, that we learnt while interacting with Claude (and suggested by Claude): the specification of mp and sbt are *not* sufficient

Before presenting this key point, I will present something which was useful for doing this proof, and where Claude impressed me. It is about the role de the Deduction Theorem.

How to do proofs in Hilbert-style system

I always thought to use a logical system based on *natural deduction* was better than working with a Hilbert-style system, with modus ponens and axioms

$$A \to (B \to A)$$
$$\big(A \to (B \to C)\big) \to \big((A \to B) \to (A \to C)\big)$$
$$(\neg A \to \neg B) \to (B \to A)$$

How to do proofs in Hilbert style system

This is because the first thing we prove as a *meta theorem* is the Deduction Theorem

$A_1, \ldots, A_n, A \vdash B$ iff $A_1, \ldots, A_n \vdash A \to B$

E.g. using this, it is easy to prove things like $(A \to (B \to C)) \to (B \to (A \to C))$

but this is a *meta theorem*, not something that will work here, where you have to do the proof at the internal level

How to do proofs in Hilbert style system

I only gave to Claude the following two slides

Automath vs Metamath

Two very different approaches

Automath: deduction theorem built in

Metamath: Hilbert-style system

One can "simulate" the Deduction Theorem (Mario Carneiro) by proving systematically $\varphi \to \psi$, instead of proving simply ψ, where φ is a "varying" formula

Automath vs Metamath

One replaces the sequence $\psi_1, \ \ldots \ , \psi_n$ by $\varphi \to \psi_1, \ \ldots \ , \varphi \to \psi_n$

E.g. if ψ_k follows from ψ_i and $\psi_j = \psi_i \to \psi_k$ then one shows that $\varphi \to \psi_k$ follows from $\varphi \to \psi_i$ and $\varphi \to \psi_j = \varphi \to (\psi_i \to \psi_k)$

Since Frege had the axiom $(a \to b \to c) \to (a \to b) \to a \to c$ he probably understood this in 1879, and this understanding was somehow lost in Russell's and then Hilbert's presentation

The proof of Gödel II

Claude understood the method, and applied it successfully to do the internal reasoning!

```
flipImp : (a0 a1 a2 : Formula) ->
          Deriv (imp (imp a0 (imp a1 a2)) (imp a1 (imp a0 a2)))
flipImp a0 a1 a2 =
  let Y : Formula
      Y = imp (imp a0 a1) (imp a0 a2)

      dYZ : Deriv (imp Y (imp a1 (imp a0 a2)))
      dYZ = bCombTwo {Y} {a1} {imp a0 a1} {imp a0 a2}
               (axK Y a1)
               (liftP Y (axK a1 a0))
  in impTrans (axS a0 a1 a2) dYZ
```

The proof of Gödel II

In the proof of Gödel I, we need to show facts such as, e.g. we can prove for all *numerals* n

$$Deriv(id(s^n(O)) = s^n(O)) \qquad Deriv(z(s^n(O)) = O)$$

This kind of statement, proved by induction on n, is *not* a formal statement; it mixes the meta-level and the object level.

In order to represent this in T we need to represent this kind of "mixed" statements

We have an internal function $num(x)$, which plays the role of $n \mapsto s^n(O)$

This function is introduced in Guard, but its role (mixing two levels) is left implicit, and never discussed explicitly in Guard's paper

The proof of Gödel II

The proof of Gödel II was an *interactive* process

We started purely autonomously with Rose's paper

H.E. Rose *On the consistency and undecidability of recursive arithmetic*, 1961

But this paper has a wrong Lemma

Furthermore, as we saw, there were several non trivial insights, only implicit in the papers

Claude stated that he was not able to prove the Lemma, but he could find an alternative path; however when we looked at the statement of what was claimed to be Gödel II, it was a complex statement having little to do with Gödel II

Summary

- Autoformalisation is not merely translation into a proof assistant
- Even classical proofs of Gödel II contain gaps, ambiguities, and unstated invariants
- Interaction with LLMs was useful, but required continuous mathematical guidance (which is actually nice)
- Current systems are assistants for mathematical exploration, not autonomous mathematicians
- For this kind of mathematics (dealing with combinatorial reasoning about syntax) the formal system we used (type theory) was able to faithfully represent the informal proofs. For examples in algebra, differential geometry, functional analysis, . . . more work is needed on the design of the formal system