Proof that Wittgenstein is correct about Gödel

The conventional notion of a formal system is adapted to conform to the sound deductive inference model operating on finite strings. Finite strings stipulated to have the semantic property of Boolean true provide the sound deductive premises. Truth preserving finite string transformation rules provide valid the deductive inference. Conclusions of sound arguments are derived from truth preserving finite string transformations applied to true premises.

Analytical_Knowledge defined as follows: The set of knowledge that can be expressed using language and verified as true entirely on the basis of stipulated relations between expressions of language.

Axioms, rules-of-inference, syntax, and truth conditional semantics are all fully integrated together into the single formalism of finite string transformation rules.

Validity and Soundness https://www.iep.utm.edu/val-snd/

A deductive argument is said to be *valid* if and only if it takes a form that makes it impossible for the premises to be true and the conclusion nevertheless to be false. Otherwise, a deductive argument is said to be *invalid*.

A deductive argument is *sound* if and only if it is both valid, and all of its premises are *actually true*. Otherwise, a deductive argument is *unsound*.

Sound_Deductive_Formalism conforms to the sound deductive inference model:
(a) **True Premises** are finite strings stipulated to have the semantic property of Boolean true or are derived from truth preserving operations on such strings.

(b) **Valid Deduction** is the application of truth preserving finite string transformations to True Premises or finite strings derived from truth preserving operations on such strings.

(c) **Conclusions** of a sound argument are any final or intermediate finite strings derived from truth preserving finite string transformations applied to True Premises or finite strings derived from truth preserving operations on such strings.

To provide a simple intuitive grasp of the **Sound Deductive Formalism** (SDF) we define a very simple formal system named **Simple_Arithmetic**.

Simple_Arithmetic evaluates the infinite set of finite strings representing this relationship: Natural_Number “+” Natural_Number “=” Natural_Number
defined by this AWK regular expression: /[0-9]+[\+][0-9]+[=][0-9]+/
to determine whether or not a formal proof exists that derives the semantic propery of Boolean true for the finite string. (see appendix).

$\forall F \in$ Sound_Deductive_Formalism $\forall X \in$ WFF(F) (True(F, X)) $\leftrightarrow$ Provable(F, X))

When it is understood that every element of the set of analytical knowledge is either a semantic tautology (defined to be true) or deduced from semantic tautologies then we see that these semantic tautologies and deductive rules-of-inference can be expressed as relations between finite strings.

This unifies sound deduction with formal proofs to theorem consequenes, thus making every element of the set of analytical knowledge provable.

AK = Analytical_Knowledge(as defined above)
$\forall x \in AK ((AK \vdash x) \leftrightarrow True(AK, x))$

No analytical expression of language is ever actually true unless there are a connected set of ideas that make it true. What-so-ever connected set of ideas that make an expression of language true can always be expressed as a connected set of relations betweeen finite strings. This connected set of relations betweeen finite strings is the formal proof of the orginal expression of language.

Wittgenstein definitions of True() and False()
'True in Russell's system' means, as was said: proved in Russell's system; and 'false in Russell's system' means: the opposite has been proved in Russell's system. (Wittgenstein 1983,118-119) Formalized by Olcott as:

LHS := RHS means LHS is defined as the RHS
$\forall x (True(RS, x) := (RS \vdash x))$ **// x is a theorem of RS**
$\forall x (False(RS, x) := (RS \vdash \neg x))$ **// ¬x is a theorem of RS**

Wittgenstein's minimal essence of the 1931 Incompleteness Theorem sentence
"I have constructed a proposition (I will use 'P' to designate it) in Russell's symbolism, and by means of certain definitions and transformations it can be so interpreted that it says 'P is not provable in Russell's system'. (Wittgenstein 1983,118-119)
Formalized by Olcott: $P \leftrightarrow (RS \nvdash P)$

When we sum up the results of Gödel's 1931 Incompleteness Theorem by formalizing Wittgenstein's verbal specification such that this formalization meets Gödel's own sufficiency requirement: "Every epistemological antinomy can likewise be used for a similar undecidability proof." then we can see that Gödel's famous logic sentence is only unprovable in PA because it is untrue in PA because it specifies the logical equivalence to self contradiction in PA.

Since the Wittgenstein-Olcott axiom schema define True(RS, x) as Provable(RS, x) then ¬Provable(RS, x) would be defined as ¬True(RS, x). This means that the Wittgenstein-Olcott minimal essence of the 1931 Incompleteness Theorem <IS> The Liar Paradox.

The Formalized Liar Paradox says that P is materially equivalent to Not True.
The truth table shows that this is self-contradictory.

P	↔	¬True(P)	P	↔	RS⊬P
T	F	F	T	F	F
F	F	T	F	F	T

The truth table of minimal essence of the 1931 Incompleteness theorem is identical to the truth table of the Liar Paradox because the third columns of these truth tables are stipulated by the Wittgenstein-Olcott axiom schema to mean exactly the same thing.

The failure of logical equivalence shows that both P and ¬P are contradicted (false) (in the above formula) thus meeting the [epistemological antinomy] sufficiency condition that Gödel stipulated for proof equivalence: “14 Every epistemological antinomy can likewise be used for a similar undecidability proof.” (Gödel 1931:40)

The fact that self-contradictory sentences specified in the language of a formal system cannot be proven in that formal system does not make the formal system itself incomplete or inconsistent as long as unprovable (from axioms) is construed as untrue.

At the most abstract level of analysis:
Conceptual Truth is ONLY semantic relations between concepts that can always be expressed as[1] syntactic relations between finite strings[2] thereby logically entailing that truth cannot possibly ever diverge from provability.

[1] Forming an isomorphism between semantic and syntactic relations:
∀x (True(x) ≅ Provable(x))

[2] Such as words, word phrases or predicate logic expressions.

Examples:
"one" [is a] "Integer"
"cats" [are] "Animals"
"cats" [have] "legs"
"2 + 3" [equals] "5"
"A ∧ B" "↔" "B ∧ A"

To make the above abstraction more concrete we focus on the single relation between concepts of [sound deduction] from the sound deductive inference model. Sound deduction begins with stipulated truth, applies a sequence of truth preserving operations, thus necessarily ends up with truth.

Truth ONLY comes from:
(1) Stipulated truth (the definitions of the meaning of words)
(2) Applying a sequence of truth preserving operations to stipulated truth.

Truth ALWAYS comes from:
(1) Stipulated truth (the definitions of the meaning of words)
(2) Applying a sequence of truth preserving operations to stipulated truth.

When we construe a formal systems axioms to essentially be stipulated truth then this same formal systems theorems would also be true because they were derived by applying truth preserving operations to its axioms. Since this is the way that Truth really works we have proven that true can never diverge from provability.

The bottom line of all this is that the only reason that G is not provable in PA is that G is not true in PA, because as Wittgenstein states true requires provable.

'True in Russell's system' means, as was said: proved in Russell's system; and 'false in Russell's system' means: the opposite has been proved in Russell's system. (Wittgenstein 1983:118)

Furthermore as Curry states True in Tarski's metatheory does not carry over to his theory as Tarski claims.

The terminology which has just been used implies that the elementary statements are not such that their truth and falsity are known to us without reference to {T}. (Curry 1977:45)

Gödel indicates the exact same inescapable contradiction that has been elaborated above. The difference is that he concludes that some truths are unprovable rather than concluding that unprovable entails untrue. (Gödel1931:39-41).

Godel, Kurt 1931. On Formally Undecidable Propositions of Principia Mathematica And Related Systems I, page 39-41. Footnote 14.

Wittenstein, Ludwig 1983. Remarks on the Foundations of Mathematics (Appendix III), 118-119. Cambridge, Massachusetts and London, England: The MIT Press (**quoted in full below**).

Tarski, Alfred 1983. “The concept of truth in formalized languages” in Logic Semantics, Metamathematics. Indianapolis: Hacket Publishing Company, 275-276.

Curry, Haskell 1977. Foundations of Mathematical Logic. New York: Dover Publications, 45

Appendix

Curry, Haskell 1977. Foundations of Mathematical Logic. New York: Dover Publications, 45
We begin by postulating a certain non void, definite class {E} of statements, which we call elementary statements...

The statements of {E} are called elementary statements to distinguish them from other statements which we may form from them or about them in the U language...

Then the elementary statements which belong to {T} we shall call the elementary theorems of {T}; we also say that these elementary statements are true for {T}. Thus, given {T}, an elementary theorem is an elementary statement which is true. A theory is thus a way of picking out from the statements of {E} a certain subclass of true statements…

The terminology which has just been used implies that the elementary statements are not such that their truth and falsity are known to us without reference to {T}.

Wittgenstein, Ludwig 1983. Remarks on the Foundations of Mathematics (Appendix III), 118-119. Cambridge, Massachusetts and London, England: The MIT Press

8. I imagine someone asking my advice; he says: "I have constructed a proposition (I will use 'P' to designate it) in Russell's symbolism, and by means of certain definitions and transformations it can be so interpreted that it says: 'P is not provable in Russell's system'. Must I not say that this proposition on the one hand is true, and on the other hand is unprovable? For suppose it were false; then it is true that it is provable. And that surely cannot be! And if it is proved, then it is proved that it is not provable. Thus it can only be true, but unprovable. "

Just as we ask: " 'provable' in what system?", so we must also ask: " 'true' in what system?" 'True in Russell's system' means, as was said: proved in Russell's system; and 'false in Russell's system' means: the opposite has been proved in Russell's system.-Now what does your "suppose it is false" mean? *In the Russell sense* it means 'suppose the opposite is proved in Russell's system'; *if that is your assumption*, you will now presumably give up the interpretation that it is unprovable.

And by 'this interpretation' I understand the translation into this English sentence.-If you assume that the proposition is provable in Russell's system, that means it' is true *in the Russell sense*, and the interpretation "P is not provable" again has to be given up. If you assume that the proposition is true in the Russell sense, *the same* thing follows. Further: if the proposition is supposed to be false in some other than the Russell sense, then it does not contradict this for it to be proved in Russell's system. (What is called "losing" in chess may constitute winning in another game.)

Tarski, Alfred 1983. “The concept of truth in formalized languages” in Logic Semantics, Metamathematics. Indianapolis: Hacket Publishing Company, 275-276.

According to Thesis A we can construct, on the basis of the enriched metatheory, a correct definition of truth concerning all the sentences of the theory studied.
…
The formulas (8) and (9) together express the fact that *x* is an undecidable sentence; moreover from (7) it follows that *x* is a true sentence.

By establishing the truth of the sentence *x* we have *eo ipso* -by reason of (2)-also proved *x* itself in the metatheory. Since, moreover, the metatheory can be interpreted in the theory enriched by variables of higher order (cf. p. 184) and since in this interpretation the sentence *x*, which contains no specific term of the metatheory, is its own correlate, the proof of the sentence *x* given in the metatheory can automatically be carried over into the theory itself: the sentence *x* which is undecidable in the original theory becomes a decidable sentence in the enriched theory.

Godel, Kurt 1931. On Formally Undecidable Propositions of Principia Mathematica And Related Systems I, page 40-41.

We now obtain an undecidable proposition of the system PM, i.e. a proposition A, for which neither A nor not-A are provable, in the following manner:
...

$$S = R(q)$$

holds for some determinate natural number q. We now show that the proposition [R(q); q]13 is undecidable in PM. For: supposing the proposition [R(q); q] were provable, it would also be correct; but that means, as has been said, that q would belong to K, i.e. according to (1), Bew [R(q); q] would hold good, in contradiction to our initial assumption. If, on the contrary, the negation of [R(q); q] were provable, then n ε K , i.e. Bew [R(q); q] would hold good. [R(q); q] would thus be provable at the same time as its negation, which again is impossible.

The analogy between this result and Richard’s antinomy leaps to the eye; there is also a close relationship with the “liar” antinomy,14 since the undecidable proposition [R(q); q] states precisely that q belongs to K, i.e. according to (1), that [R(q); q] is not provable. We are therefore confronted with a proposition which asserts its own unprovability.15

14 Every epistemological antinomy can likewise be used for a similar undecidability proof.

15 In spite of appearances, there is nothing circular about such a proposition, since it begins by asserting the unprovability of a wholly determinate formula (namely the q-th in the alphabetical arrangement with a definite substitution), and only subsequently (and in some way by accident) does it emerge that this formula is precisely that by which the proposition was itself expressed.

```
/*****************************************************************************
This code snippet demonstrates [truth conditional semantics] for the subset
of analytic knowledge involving "=" relational expressions of the arithmetic
operation of "+" applied to finite strings OF ASCII digits of arbitray length
representing natural numbers.

  Truth-conditional semantics is an approach to semantics of natural
  language that sees meaning (or at least the meaning of assertions)
  as being the same as, or reducible to, their truth conditions.
  https://en.wikipedia.org/wiki/Truth-conditional_semantics

The finite string transformation rules specified by this source-code provide
the means to formally prove whether a finite string of the language of the
Simple_Arithmetic formal system has the semantic property of Boolean true.

This AWK regular expression: specifies the entire language of the
Simple_Arithmetic Sound Deductive Formalist formal system:
/[0-9]+[\+][0-9]+[=<>][0-9]+/
*****************************************************************************/


char AddWithCarry(char D1, char D2, char& Carry)
{
  char SUM   = ADD_Digit[D1][D2];
  if (Carry == '1' && SUM == '9')
  {
    SUM   = '0';
    Carry = '1';
  }
  else if (Carry == '1' && SUM < '9')
  {
    SUM   = ADD_Digit[SUM][Carry];
    Carry = ADD_Carry[D1][D2];
  }
  else // Carry == '0'
    Carry = ADD_Carry[D1][D2];
  return SUM;
}


std::string Add(std::string& OP1, std::string& OP2)
{
std::string SUM;
  char Carry = '0';
  for (int N = OP1.length() - 1; N >= 0; N--)
    SUM += AddWithCarry(OP1[N], OP2[N], Carry);
  if (Carry == '1')
    SUM += '1';
  std::reverse(SUM.begin(), SUM.end());
  return SUM;
}


//
// (Proven && True) || (Unproven && Untrue)
//
bool ProveInput(std::string& OP1,std::string& OP2,
                std::string& SUM, char Relational_OP)
{
  std::string RESULT;
  RESULT = Add(OP1, OP2);
  return (RESULT == SUM);
}
```