Refuting Incompleteness and Undefinability Version(14)

Within the (Haskell Curry) notion of a formal system we complete Tarski's formal correctness: ∀x True(x) ↔ ⊢ x and use this finally formalized notion of Truth to refute his own Undefinability Theorem (based on the Liar Paradox), the Liar Paradox, and the (Panu Raatikainen) essence of the conclusion of the 1931 Incompleteness Theorem.

When we understand that analytical truth is nothing more than the conclusion of sound deductive inference, and we assume the (Haskell Curry) notion of a formal system and its axioms:

> The construction of a theory begins by specifying a definite non-empty conceptual class E the elements of which are called statements. ... The elementary statements which belong to T are called the elementary theorems of T and said to be true.
> (Haskell Curry, Foundations of Mathematical Logic, 2010)

We can specify and analyze analytical Truth directly within the formal system with no need for model theory. Within (Haskell Curry) the ultimate basis for truth is simply expressions of language having their Boolean property assigned the semantic value of true, AKA axioms specified in the language.

Since we already know that valid deductive inference is truth preserving, then we can also understand that the theorems of any (Haskell Curry) formal system must also be true. By assuming these (Haskell Curry) notions we can complete the RHS of Tarski's formal correctness: ∀x True(x) ↔ φ(x) with this expression: ∀x True(x) ↔ ⊢ x.

Now that we have defined a universal Truth predicate, we can finally evaluate the formalized Liar Paradox much more effectively. "This sentence is not true." is formalized as: LP ↔ ~ ⊢ LP.
When the Liar Paradox asserts that it is not a theorem this assertion can only be true when it is a theorem, thus proving that the Liar Paradox is self-contradictory.

Since we have unified the formal proof of theorems with sound deductive inference we can see that a self-contradictory expressions of language must be treated as a deduction based on contradictory premises, thus unsound and therefore not true.

Since Tarski's Undefinability proof:

(1) x ∉ Pr ↔ p	// p ↔ ~Provable(x)
(2) x ∈ Tr ↔ p	// p ↔ True(x)
(3) x ∉ Pr ↔ x ∈ Tr	// ~Provable(x) ↔ True(x)
(4) either x ∉ Tr or ~x ∉ Tr	// ~True(x) ∨ ~True(~x)
(5) if x ∈ Pr, then x ∈ Tr	// Provable(x) → True(x)
(6) if ~x ∈ Pr, then ~x ∈ Tr	// Provable(~x) → True(~x)
(7) x ∈ Tr	// True(x)
(8) x ∉ Pr	// ~Provable(x)
(8a) ~x ∉ Tr	// ~True(~x)
(9) ~x ∉ Pr	// ~Provable(~x)

Is anchored in the Liar Paradox:

THEOREM I. (α) In whatever way the symbol 'Tr', denoting a class of expressions, is defined in the metatheory, it will be possible to derive from it the negation of one of the sentences which were described in the condition (α) of the convention T;

It would then be possible to reconstruct the antinomy of the liar in the metalanguage, by forming in the language itself a sentence x such that the sentence of the metalanguage which is correlated with x asserts that x is not a true sentence.

And the Liar Paradox has been shown to be ill-formed, Tarski's proof fails at step(3).

https://plato.stanford.edu/entries/goedel-incompleteness/
The first incompleteness theorem states that in any consistent formal system F within which a certain amount of arithmetic can be carried out, there are **statements of the language of F** which can neither be proved nor disproved in F. (Panu Raatikainen Fall 2018)

Simplifying and formalizing above essence of the conclusion of the 1931 Incompleteness Theorem:
$\exists F \in \text{Formal\_Systems}\ (\exists G \in \text{Language}(F)\ (G \leftrightarrow \sim(F \vdash G)))$
There exists an F element of Formal_Systems
There exists a G element of the Language of F
such that G is materially equivalent to a statement of its own unprovability in F.

When the above expression is shown to be unsatisfiable this proves that no such G that Kurt Gödel posits actually exists, thus refuting the essence of the conclusion of his 1931 Incompleteness Theorem.

We can see that $G \leftrightarrow \sim(F \vdash G)$ expresses the exact same thing as the formalized Liar Paradox, except in this case the results are constrained to the formal system of F.

(a) If G was provable in F this would contradict its assertion that G is not provable in F.
∴ G is not provable in F.

(b) If ~G was provable in F this would contradict its assertion that G is not not provable (thus provable) in F. **∴ ~G is not provable in F.**

https://en.wikipedia.org/wiki/Sentence_(mathematical_logic)
A sentence can be viewed as expressing a proposition, something that must be true or false.

Within the context of our Truth predicate True(F, G) ↔ Theorem(F, G) we determine that ~(True(F, G) ∧ True (F, ~G)) ↔ Logic_Sentence(F, G), thus G is not a **statement of the language of F** and the most elegant essence of the conclusion 1931 Incompleteness Theorem that exists in the world: (Raatikainen 2018) is refuted.

Minimal Type Theory (formal specification)

Minimal Type Theory is essentially higher order logic (HOL) with types as an augmentation to first order logic (FOL) syntax. MTT is intended to be used as a universal Tarski meta-language eliminating messy mixing and matching between his object language and metalanguage. There is no need to force-fit meta-language variables directly into the object language or otherwise move back and forth between two languages. We simply have one language that can express anything.

```
%left  IDENTIFIER            //  Letter+ (Letter | Digit)*  // Letter includes UTF-8
%left  SUBSET_OF             //  ⊆
%left  ELEMENT_OF            //  ∈
%left  FOR_ALL               //  ∀
%left  THERE_EXISTS          //  ∃
%left  IMPLIES               //  →
%left  PROVES                //  ⊢
%left  IFF                   //  ↔
%left  AND                   //  ∧
%left  OR                    //  ∨
%left  NOT                   //  ~
%left  ASSIGN_ALIAS          //  :=  LHS is assigned as an alias name for the RHS (macro substitution)
%%

sentence
      : atomic_sentence
      | '~' sentence %prec NOT
      | '(' sentence ')'
      | sentence   IMPLIES      sentence
      | sentence   IFF          sentence
      | sentence   AND          sentence
      | sentence   OR           sentence
      | quantifier IDENTIFIER   sentence
      | quantifier IDENTIFIER   type_of IDENTIFIER sentence   // Enhancement to FOL
      | sentence   PROVES       sentence                      // Enhancement to FOL
      | IDENTIFIER ASSIGN_ALIAS sentence                      // Enhancement to FOL
      ;

atomic_sentence
      : IDENTIFIER '(' term_list ')' // ATOMIC PREDICATE
      | IDENTIFIER                   // SENTENTIAL VARIABLE   // Enhancement to FOL
      ;

term
      : IDENTIFIER '(' term_list ')' // FUNCTION
      | IDENTIFIER                   // CONSTANT or VARIABLE
      ;

term_list
      : term_list ',' term
      | term
      ;

type_of
     : ELEMENT_OF                                            // Enhancement to FOL
     | SUBSET_OF                                             // Enhancement to FOL
     ;

quantifier
     : THERE_EXISTS
     | FOR_ALL
     ;
```