The Notion of Truth in Natural and Formal Languages

The purpose of this paper is to complete the RHS of Tarski's famous formula: ∀x True(x) ↔ φ(x)

For any natural (human) or formal (mathematical) language L we know that an expression X of language L is true if and only if there are expressions Γ of language L that connect X to known facts. By extending the notion of a Well Formed Formula to include syntactically formalized rules for rejecting semantically incorrect expressions we recognize and reject expressions that evaluate to neither True nor False.

> Weisstein, Eric W. "**Axiom**." From MathWorld--A Wolfram Web Resource.
> http://mathworld.wolfram.com/Axiom.html
> An axiom is a proposition regarded as self-evidently true without proof.

Axioms are really nothing more than a set of expressions of language that have been assigned the semantic property of True. Axioms form the ultimate foundation of Truth-conditional semantics.

> https://www.britannica.com/topic/tautology
> **Tautology**, in logic, a statement so framed that it cannot be denied without inconsistency. Thus, "All humans are mammals" is held to assert with regard to anything whatsoever that either it is a human or it is not a mammal. But that universal "truth" follows not from any facts noted about real humans but only from the actual use of human and mammal and is thus purely a matter of definition.

The natural language equivalent to an axiom in formal language is a {known fact}. Some expressions of natural language are simply tautologically defined to be True.

Example: "a cat is an animal".
Formalized as: (cat ∈ animals) or (cat ◁ animal)
where ◁ is the [is_a_type_of] operator adapted from UML Inheritance relation.
The only reason that we know that "a cat is an animal" is that it is defined to be True.

> **Meaning Postulates** (1952) by Rudolf Carnap formalized natural language semantics:
> (x) Bachelor(x) → ~Married(x)

All that we know about the otherwise totally meaningless finite string predicate "Bachelor" is that when this predicate is satisfied then the otherwise totally meaningless finite string predicate "Married" is not satisfied. We did not even limit x, so we could say that if {a pile of bricks} is a Bachelor then {a pile of bricks} is not married.

A further refinement would be this:∀x ∈ Human_Being Bachelor(x) → ~Married(x)

Now the above two predicates have been constrained to a type, and all that we know about the otherwise totally meaningless finite string "Human_Being" is that it is the argument type to the otherwise totally meaningless finite string predicates: "Bachelor" and "Married". The above example shows how to formalize natural language axioms.

Generalizing the notion of a (known fact) to formal language we define an axiom as any expression of (formal or natural) language that has been tautologically assigned the semantic property of True. This concept of an axiom provides the ultimate foundational basis of all conceptual Truth.

> **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.

A sound argument necessitates the Truth of its conclusion.

If we define a symbolic logic predicate to formalize the distinction between a valid deductive argument and a sound deductive argument where represents the premises Γ and C stands for the conclusion then:

Provable(Γ, C) is simply a valid deductive argument where the premises Γ may or may not be true. True(Γ, C) is simply a deductive argument known to be sound because its premises Γ are axioms. Provable(Γ, C) is more fully elaborated by Mendelson's reference to Γ ⊢ C.

> Introduction to Mathematical logic Sixth edition Elliott Mendelson
> 1.4 An Axiom System for the Propositional Calculus
> A wf C is said to be a consequence in S of a set Γ of wfs if and only if there is a sequence B1, ..., Bk of wfs such that C is Bk and, for each i, either Bi is an axiom or Bi is in Γ, or Bi is a direct consequence by some rule of inference of some of the preceding wfs in the sequence. Such a sequence is called a proof (or deduction) of C from Γ. The members of Γ are called the hypotheses or premisses of the proof.
> **We use Γ ⊢ C as an abbreviation for "C is a consequence of Γ".**

Γ ⊢ C is merely infix notation for this predicate Provable(Γ, C). My Truth predicate can be defined as True(Γ, C) by simply requiring that the Mendelson premises be axioms. By doing this the Valid deductive argument specified by a formal proof becomes a Sound deductive argument, thus the conclusion is necessarily True. Because Axioms ⊆ WFF therefore True(Γ, C) ⊆ Provable(Γ, C).

We generalize the Mendelson provability predicate: Γ ⊢ C by applying it to every formal system ∀L ∈ Formal_Systems. ∃Γ ⊆ WFF(L) formalizes "set Γ of wfs" and binds Γ to the existential quantifier. ∀L ∈ Formal_Systems Provable(L, C) ↔ ∃Γ ⊆ WFF(L) (Γ ⊢ C)

Transforming the above generic Provability predicate into a generic Truth predicate:
∀L ∈ Formal_Systems True(L, C) ↔ ∃Γ ⊆ Axioms(L) (Γ ⊢ C)

To verify that an expression X of language L is True or False only requires a syntactic logical consequence inference chain (formal proof) sequence of WFF (with premises as Axioms) to the consequent of X or ~X. (Backward chaining reverses this order).

When True(L, X) ⊆ Provable(L, X) we can see that it is impossible for any expression of language X to be True in L and not Provable in L.

The key understanding that the above analysis provides:
(1) Axioms (known facts) are the ultimate foundational basis of conceptual Truth.
(2) True(L, X) ⊆ Provable(L, X)

> **Sentence (mathematical logic)** https://en.wikipedia.org/wiki/Sentence_(mathematical_logic)
> In mathematical logic, a sentence of a predicate logic is a Boolean-valued well-formed formula with no free variables. A sentence can be viewed as expressing a proposition, something that must be true or false. The restriction of having no free variables is needed to make sure that sentences can have concrete, fixed truth values: As the free variables of a (general) formula can range over several values, the truth value of such a formula may vary.

Predicate logic is augmented with an <assign alias name> operator.
LHS is assigned as an alias name for the RHS
LHS ≡ RHS
The LHS is logically equivalent to the RHS only because
the LHS is merely an alias name (place-holder) for the RHS
The <assign alias name> operator allows an expression to refer directly to itself.

Apparently in all of the years prior to my original (2016) paper no one ever separated the analysis of propositions into their atomic units of semantic compositionality:
(a) Assertion // What the expression is claiming to be True
(b) Satisfiability (Boolean) // Is the expression of language Satisfied?

Previously these two properties were conflated together as the Satisfiability property making the analytical system insufficiently granular to detect the semantic error.

What has previously been understood to be True outside the system and not Provable within the system is actually a contradiction between two different semantic properties of the same Proposition. So True, but not Provable is really nothing more than a contradiction between an expression's Assertion property and its Satisfiability property.

In the case of semantic error of Pathological Self-Reference(Olcott 2004) the Truth of the expression's Assertion property does not make the entire expression True because the Truth values of the Assertion property and the Satisfiability property of the expression are mutually exclusive.

Because two different Boolean properties of an expression of language have mutually exclusive values this makes it impossible to evaluate this expression as either True or False.

When we formalize expressions of language such as the Liar Paradox using the above universal truth predicate, we can finally understand its semantic error.

"This sentence is not True."
LP ≡ ∀L ∈ Formal_Systems ~True(L, LP)

Expanded definition of above:
LP ≡ ∀L ∈ Formal_Systems ~∃Γ ⊆ Axioms(L) (Γ ⊢ LP)

[LP] Makes the assertion that:
[LP] There is no sequence of WFF that proves [LP].
Is this assertion True or False?

If there was a sequence of WFF that proves LP it would be self-contradictory because it proved that its own proof must fail therefore making its proof succeed. Therefore LP's assertion is True.

Even though LP's assertion is True, because the LP expression cannot possibly be satisfied the LP expression cannot possibly be True. It is also impossible for the LP expression to be False because its assertion is True.

So we have the paradoxical case where the assertion of a Proposition is True, yet this does not make the Proposition itself True. Since LP cannot possibly be either True or False it is therefore a semantically incorrect Proposition because all Propositions must be True or False.

> ON FORMALLY UNDECIDABLE PROPOSITIONS
> OF PRINCIPIA MATHEMATICA AND RELATED SYSTEMS I
> by Kurt Godel Vienna
> The analogy between this result and Richard's antinomy leaps to the eye; there is also a
> close relationship with the "liar" antinomy,14
>
> 14 Every epistemological antinomy can likewise be used for a similar undecidability proof.

Since Kurt Godel said that the Liar Paradox "can ... be used for a similar undecidability proof."
The semantic error of the Liar Paradox equally applies to the 1931 Incompleteness Theorem.

A formula precisely analogous to the Liar Paradox specifying Provability instead of Truth
"This sentence is not provable".
G ≡ ∀L ∈ Formal_Systems ~∃Γ ⊆ WFF(L) (Γ ⊢ G)

So we do not really have an expression of language L that is True in L yet not Provable in L. What we really have had all along is a semantically incorrect expression G of language L because two aspects of G: its (assertion property) and its (satisfiability property) contradict each other:
(a) G has an assertion that is definitely true.
(b) The expression G cannot possibly be satisfied.

When we simply look at Incompleteness a little bit more clearly the famous theorem's famous conclusion falls apart.