Philosophy of Logic – Reexamining the Formalized Notion of Truth

Because formal systems of symbolic logic inherently express and represent the deductive inference model formal proofs to theorem consequences can be understood to represent sound deductive inference to deductive conclusions without any need for other representations.

I am approaching these things from the frame of reference of the Tarski Undefinability Proof. I created Minimal Type Theory as a universal Tarski metalanguage eliminating the need to switch back and forth and mix and match between a meta-language and a separate object language.

The other great thing about MTT is that it has its own provability operator thus no need to go through the cumbersome process of artificially contriving a provability predicate in a language that is woefully inadequate for this task. Expressions in a language with its own provability operator can be directly examined within the deductive inference model.

First we lay the foundation of expressing semantic truth directly within a formal system. All of semantic truth has its ultimate ground of being in expressions of language that have been defined to be true.

https://en.wikipedia.org/wiki/Theory_(mathematical_logic)
The construction of a theory begins by specifying a definite non-empty conceptual class E the elements of which are called statements. These initial statements are often called the primitive elements or elementary statements of the theory, to distinguish them from other statements which may be derived from them.

A theory T is a conceptual class consisting of certain of these elementary statements. The elementary statements which belong to T are called the elementary theorems of T and said to be true. In this way, a theory is a way of designating a subset of E which consists entirely of true statements. (Haskell Curry, Foundations of Mathematical Logic, 2010).

From this basis we can infer that every formal proof of theorems in such a (Haskell Curry) formal system would exactly correspond to deriving the conclusion of sound deductive inference. R. B. Braithwaite explains this in depth below:

KURT GÖDEL
Translated by B. MELTZER
Introduction by R. B. BRAITHWAITE
2 INTRODUCTION

In order to show that in a deductive system every theorem follows from the axioms according to the rules of inference it is necessary to consider the formulae which are used to express the axioms and theorems of the system, and to represent the rules of inference by rules Gödel calls them “mechanical” rules, p. 37) according to which from one or more formulae another formula may be obtained by a manipulation of symbols. Such a representation of a deductive system will consist of a sequence of formulae (a calculus) in which the initial formulae express the axioms of the deductive system and each of the other formulae, which express the theorems, are obtained from the initial formulae by a chain of symbolic manipulations. **The chain of symbolic manipulations in the calculus corresponds to and represents the chain of deductions in the deductive system.**

But this correspondence between calculus and deductive system may be viewed in reverse, and by looking at it the other way round Hilbert originated metamathematics. Here a calculus is constructed, independently of any interpretation.

From the above we can see that the formal proof to theorem consequences expressed in symbolic logic represents and expresses sound deductive inference to deductive conclusions. One way to look as this might be that formal proof to theorem consequences corresponds to and expresses the sound deductive inference model.

Since the conclusions of sound deductive inference are understood to be true we can formulate this universal truth predicate:

$\forall F \in \text{Formal\_Systems}\ \forall x\ \text{WFF}(F)\ \ (\text{True}(F, x) \leftrightarrow (F \vdash x))$

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. This provides the way to rewrite his proof using a single concise language.

```
%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
     ;
```