Turing's Fallacies

Timm Lampert
Humboldt University Berlin

Abstract

This paper reveals two fallacies in Turing's undecidability proof of first-order logic (FOL), namely, (i) an "extensional fallacy": from the fact that a sentence is an instance of a provable FOL formula, it is inferred that a *meaningful* sentence is proven, and (ii) a "fallacy of substitution": from the fact that a sentence is an instance of a provable FOL formula, it is inferred that a *true* sentence is proven. The first fallacy erroneously suggests that Turing's proof of the non-existence of a circle-free machine that decides whether an arbitrary machine is circular proves a significant proposition. The second fallacy suggests that FOL is undecidable.

1 Introduction

In his famous paper from 1936, Turing proved what was later called the "Church-Turing theorem", i.e., the theorem that the property of logical theoremhood within first-order logic (FOL) is undecidable. This is a well-established theorem of mathematical logic. Questioning this theorem might seem illegitimate to those familiar with meta-mathematical proof methods or those who believe that meta-mathematical theorems are irrefutable. Contrary to model-theoretic proofs of the correctness and completeness of FOL-calculi, however, Turing's proof of the undecidability of FOL rests on unreliable semantic principles concerning the instantiation of logical formulas. This paper identifies these problematic semantic principles through a close examination of Turing's original proof. Note that it does not follow from this critique that FOL is decidable. It merely follows that one should not trust meta-mathematical undecidability proofs resting on instantiations of logical formulas.

This kind of critique is not new. According to my understanding of

Wittgenstein, it is consistent with Wittgenstein's remarks on Gödel.[1] Unfortunately, Wittgenstein does not closely examine Turing's famous undecidability proof, although he was in close contact with Turing at Cambridge during the second half of the nineteen-thirties and was one of the first to receive Turing's paper in February 1937.[2] Nor does he closely examine Gödel's proof or any other meta-mathematical undecidability proof. Instead, he complains rather generally that instead of inferring undecidability theorems, one might question the underlying interpretations of meta-mathematical undecidability proofs.[3] However, the intent of this critique is highly controversial.[4] I avoid discussing this controversy in this paper. Instead, I present a systematic critique of Turing's proof that illustrates why referring to interpretations within Turing's undecidability proof is problematic. Thus, I wish to make plausible a certain kind of critique of undecidability proofs that seems to me to represent the relevant systematic value of Wittgenstein's scepticism regarding undecidability proofs.

Turing[5] proves the impossibility of a circle-free machine $\mathcal{D}$ that can decide whether an arbitrary machine is circular. On this basis, he proves the impossibility of a machine $\mathcal{E}$ that can decide whether an arbitrary machine $\mathcal{M}$ will ever print 0. Given these conclusions, he then proves the impossibility of a machine that can decide whether an arbitrary FOL formula is a theorem. I show in section 2 that a circle-free machine $\mathcal{D}$ (and, consequently, $\mathcal{E}$) is undefinable. This work is important for analysing Turing's proof of the undecidability of FOL. I show in section 3 that Turing's undecidability proof rests on a fallacy. Instead of concluding that FOL is undecidable, one may conclude that the configurations of certain machines are undefinable within the language of FOL.

2 Proof of the Impossibility of $\mathcal{D}$

This section discusses Turing's proof of the impossibility of a circle-free machine $\mathcal{D}$ that can decide whether an arbitrary machine $\mathcal{M}$ is circular. According to Turing, a machine is circular iff it never prints more than

[1] Cf. in particular Wittgenstein (1967), part I, appendix I, and part V, §17-19 and Lampert (2017).

[2] Cf. Turing's letter to his mother AMT/K/1/54, Turing Digital Archive (http://www.turingarchive.org/browse.php/K/1/54), and Floyd (2016b), p. 9, footnote 3.

[3] Cf., e.g., Wittgenstein (1967), part I, appendix I, §8, §10, §17.

[4] Cf., e.g., Rodych (1999) and Floyd and Putnam (2000).

[5] Turing (1936).

a finite number of 0s or 1s. Therefore, circle-free machines print endless sequences of 0s and 1s. Turing calls circle-free machines "satisfactory", whereas circular machines are "unsatisfactory" because they become stuck somewhere. They either reach a configuration from which there is no possible move, or if they continue moving, they no longer print 0s or 1s.[6]

Whereas Turing's machines start with empty tapes and should print endless binary sequences, "Turing machines" as the term is used in the modern literature, begin with a finite binary code and should return a finite binary code. According to this definition, machines that do not halt are "unsatisfactory". Therefore, instead of proving the impossibility of a machine that can decide the circularity of machines, it is proven that there can be no machine that solves the halting problem. In the following, however, I will focus on Turing's original conception of machines.

Turing provides two proofs of the impossibility of a machine $\mathcal{D}$. In the following, I primarily discuss Turing's first proof, which relies on Cantor's diagonal argument. The principles of this proof are the same as those of modern proofs of the unsolvability of the halting problem.

2.1 Turing's First Proof

Turing's first proof is very short an refers to an anti-diagonal sequence β of an enumeration of computable sequences:[7]

> In fact, by applying the diagonal process argument correctly, we can show that there cannot be any such general process [i.e., a general process for determining whether a given number is the description number (D.N.) of a circle-free machine]. The simplest and most direct proof of this is by showing that, if this general process exists, then there is a machine which computes β.

As Turing mentions, this proof applies Cantor's diagonal argument, which proves that the set of all infinite binary sequences, i.e., sequences consisting only of digits of 0 and 1, is not countable. Cantor's argument, and certain paradoxes, can be traced back to the interpretation of the following FOL theorem:[8]

$$\neg\exists x\forall y(Fxy \leftrightarrow \neg Fyy) \tag{1}$$

[6] Cf. Turing (1936: 233).

[7] Cf. Turing (1936: 246).

[8] This reduction of the diagonal argument and its application to the analysis of paradoxes has a long tradition, cf. Simmons (1993: 25), footnote 10.

(1) is proven by reducing the corresponding existential claim to absurdity.

Interpreting (or instantiating) Fxy as "x shaves y" yields the barber paradox; interpreting Fxy as "$y \in x$" yields Russell's paradox. In Cantor's proof, the binary anti-diagonal sequence β is defined with respect to the sequences α_k comprising some enumeration of binary sequences, where k is the index identifying each sequence in this enumeration. Let n be the index representing the n-th position in the k-th sequence. Then, the sequences α_k can be represented by sequences of coefficients $a_{k,n}$. Thus, the diagonal sequence β' is defined as $\sum_{n=1}^{\infty} a_{n,n} \cdot 10^{-n}$, and the corresponding anti-diagonal sequence β requires $b_n \neq a_{n,n}$. Therefore, the proof that β is not identical to any sequence α_k in the enumeration corresponds to the following interpretation of (1):

$$\neg\exists k \forall n (b_n = a_{k,n} \leftrightarrow b_n \neq a_{n,n}) \tag{2}$$

Turing distinguishes between incorrect and correct applications of Cantor's diagonal argument to *computable* sequences α. These are defined as sequences that are computable by *circle-free* machines.[9] Using the diagonal argument to prove that the (enumerable) computable sequences are *not enumerable* is an *incorrect* application of the diagonal argument because it does not consider the possibility that the anti-diagonal sequence β of the enumeration of the computable sequences may be definable by a circular machine.[10] Thus, β is simply not among the enumerable computable sequences that are defined by circle-free machines.

In fact, a *correct* application of the diagonal argument presumes the enumerability of the computable sequences. Therefore, instead of proving the non-enumerability of the computable sequences, Cantor's diagonal argument proves that the anti-diagonal sequence β of the enumeration of the computable sequences is *not computable*. It follows that the concept of an anti-diagonal *computable* sequence β is inconsistent because of (2). Consequently, the concept of a *circle-free* β-machine, i.e., one that computes β, is also inconsistent. The proof that such a β-machine does not exist is a proof by contradiction that reduces the corresponding existential claim to absurdity. The proven theorem is a tautology, i.e., an instance of a provable FOL formula.

Turing proves the impossibility of a circle-free machine $\mathcal{D}$ based on this proof through another proof by contradiction. If $\mathcal{D}$ exists, then β must

[9] Cf. (Turing 1936: 233).
[10] Cf. (Turing 1936: 246).

be computable by a circle-free β-machine $\mathcal{H}$ composed of (i) the circle-free machine $\mathcal{D}$ and (ii) a universal machine $\mathcal{U}$ that computes the sequence of a circle-free machine from the standard description (S.D.) of said circle-free machine that $\mathcal{U}$ receives as input, and, finally, (iii) a machine S switching 0 to 1 and vice versa. Turing proved in section 6 of his paper that $\mathcal{U}$ exists and S trivially exists; therefore, it is impossible for $\mathcal{D}$ to exist because, according to the diagonal argument, no circle-free β-machine exists.

Interestingly, Turing makes the following comment on his first proof of the non-existence of $\mathcal{D}$:

> This proof, although perfectly sound, has the disadvantage that it may leave the reader with a feeling that "there must be something wrong".

Turing calls his proof "perfectly sound" while simultaneously conceding, without any further explanation, that some readers may not be convinced by such a proof. Turing's second proof, which I consider in section 2.5, does not rely on Cantor's diagonal argument. Turing seems to believe that scruples regarding his proof concern (correct) applications of Cantor's diagonal argument and, thus, the particular method of proof, not what is proven. In the following, I argue that this is not the case.[11]

2.2 Two Types of Proof by Contradiction

Because interpretations of theorem (1) have given rise to several well-known instances of both theorems and paradoxes, an effort is made to distinguish between "good" and "bad" diagonal arguments.[12] However, before looking for such a discrimination criterion, one should examine what the proofs of the theorems based on interpretations of (1) actually prove.

The reason why a proof by contradiction that rests on the reduction of a single existential claim to absurdity is suspicious becomes clear when this type of proof by contradiction is compared with alternative proofs by contradiction. Turing's proof is not a proof by contradiction that proves the negation of one of *several* assumptions. Such a proof does not prove a trivial

[11] In her reconstruction of Turing's first proof, Floyd (2012: 33) seems to miss that Turing reduces the non-existence of D to a primarily proved non-existence of a circle-free β-machine by applying Cantor's diagonal argument. That is why she believes that Turing's first proof does not provide a convincing argument for the non-existence of $\mathcal{D}$. According to my reconstruction, however, Turing's first proof should be convincing to all those (including Turing) who accept diagonal arguments in terms of interpretations of (1).

[12] Cf., e.g., Simmons (1993) and Sylvan (1992).

tautology but rather negates only one of several consistent assumptions. By contrast, a proof by contradiction resulting in an instance of (1) rests on only *one* assumption that claims the existence of "something" that satisfies an inconsistent propositional function. In this case, the proven theorem is a trivial, meaningless tautology. Assuming the existence of a circle-free β-machine or a circle-free machine $\mathcal{D}$ calls for something that is impossible by definition. Therefore, strictly speaking, it is not proven that a *certain machine* does not exist. Instead, it is proven that an assumption such as that adopted for the proof cannot even be made because it does not refer to any specific entity.

Wittgenstein distinguishes the use of proofs by contradiction in mathematics from the use of similar proofs in physics. The former presume inconsistent definitions, whereas the latter are based on a set of well-defined assumptions that are inconsistent when taken together:[13]

> The difficulty which is felt in connexion with *reductio ad absurdum* in mathematics is this: what goes on in this proof? Something mathematically absurd, and hence unmathematical? How – one would like to ask – can one so much as assume the mathematically absurd at all? That I can assume what is physically false and reduce it *ad absurdum* gives me no difficulty. But how to think the – so to speak – unthinkable?

Wittgenstein's critique of proofs by contradiction in mathematics does not question the logical validity of such proofs, only the meaningfulness of what is proven. This type of criticism explains why such proofs are judged as "perfectly sound" by, e.g., Turing, on the one hand, while remaining suspicious for others, e.g., Wittgenstein, on the other. In contrast to Wittgenstein, I do not wish to exclude proofs by contradiction that are based on several meaningful assumptions from mathematics. Therefore, I merely will call proofs by contradiction that result in tautologies "empty".

2.3 Meaningless Tautologies

Let us call instances of logical theorems "tautologies". Whether tautologies are meaningful or lack sense is a controversial question. Wittgenstein is a prominent advocate of the latter, as Quine is of the former.[14] Similarly,

[13] Wittgenstein (1967: 147).

[14] Cf. Wittgenstein (1994), remark 4.461, Quine (1960: 25). Quine argues against Wittgenstein's doctrine that tautologies and contradictions lack sense by presuming the validity of any sort of proof by contradiction in general and the validity of undecidability proofs in particular. Therefore, he presumes precisely what is in question here.

whether contradictions (instances of contradictory formulas) and inconsistent concepts lack sense remains a subject of much discussion. The questions are whether tautologies or contradictions state anything of substance and whether inconsistent concepts refer to anything of substance.

For those who answer these questions in the affirmative, logical formulas lack sense in general; however, instances of such formulas are meaningful and have truth values regardless of the type of logical formula from which they arise. Different instances may differ in meaning. Therefore, different instances of logical theorems may prove the truth of propositions that differ in meaning.

However, this point of view is deficient, as can be shown by assigning instances of one and the same formula to several structurally equivalent formulas that differ only in their non-logical symbols. Thus, instead of interpreting one and the same formula, say $P \vee Q$, from two different propositions, one of the two propositions may be assigned to $P \vee Q$ and the other to, say, $R \vee S$. In this case, structurally equivalent formulas that are instantiated based on different propositions that are not tautologies or contradictions are not logically equivalent, whereas *all* formulas instantiated by tautologies (or contradictions) are logically equivalent. Thus, in interpreting $P \vee \neg P$ and $Q \vee \neg Q$ from two different propositions (or sentences), the logical equivalence of these two formulas shows that the two propositions are also logically equivalent. Therefore, given that logically equivalent propositions do not differ in meaning, all tautologies have the same meaning. Furthermore, if meaningfulness is measured by the existence of logically independent propositions, then tautologies lack meaning because they follow from any proposition. Therefore, all tautologies have the "same" meaning, namely, none. The same holds for contradictions: they are all logically equivalent and meaningless because every proposition follows from any contradiction. One might still maintain that tautologies are true and contradictions are false; however, no content is labelled as true or false. In this respect, the proof of a tautology proves nothing. From this perspective, tautologies and contradictions both belong to non-extensional contexts, such as meta-linguistic and intensional ones. Unlike in the usual extensional contexts, expressions within tautologies and contradictions do not refer to any entity.

The conception of tautologies and contradictions as meaningful is correlated with the view that contradictory propositional functions are meaningful and refer to the empty set. This view expresses a purely extensional account of logical semantics that does not distinguish between meaningful but unsatisfied concepts and meaningless (unsatisfiable) concepts. Misled

by a material mode of speech, one infers from the fact that expressions refer to certain entities in other contexts that expressions within tautologies and contradictions also refer to entities. However, inconsistent concepts do not specify anything and, therefore, are unable to identify any sort of set, including the empty set.

A material mode of speech that pretends to reference entities misleadingly suggests that different theorems should be correlated with different interpretations of empty proofs by contradiction. However, considering equivalence relations between expressions instead of presuming reference reveals that such proofs do not prove anything specific because all types of tautologies can be replaced with one and the same symbol, $\top$. This symbol no longer depends on the value of any propositional function. It is an illusion that meaningful existential claims about barbers, sets or machines can be reduced to absurdity within empty proofs by contradiction.

2.4 Extensional Fallacy

Those who claim that empty proofs by contradiction prove anything of substance succumb to an "extensional fallacy": misled by a material mode of speech within a negated existential claim, they erroneously infer that *something* does not exist. However, this fallacy can be overcome by considering that the logical form of the expression is a tautology and concluding from this that the expression lacks meaning. This applies to Turing's proof of the non-existence of a circle-free β-machine. In fact, the concept of a circle-free β-machine is simply inconsistent, and the corresponding theorem is an empty tautology that does not prove anything.

Because empty proofs by contradiction do not prove anything, one might ask for alternative interpretations that do justice to the intention to refer to some entity. In fact, one refers to entities that satisfy some meaningful concept as soon as the domain is restricted to a proper domain of definition. In the case of Turing's proof, for example, one might conceive of "x is a machine that computes β" [15] as a partial function that is not defined for the description number (D.N.) of the machine in question. However, according to this understanding, it is not possible to prove the corresponding negative existential claim, and the relevant β-machine is not circle-free. Therefore, such a definition does not define the entity to which Turing intends to refer.

[15] Turing (1936: 246).

2.5 Turing's Second Proof

Although Turing believes his first proof to be "perfectly sound", he has a sense of the scruples one might feel against diagonal arguments based on anti-diagonal sequences. Therefore, he delivers a second proof of the impossibility of a machine $\mathcal{D}$ that he believes should not be met with similar resistance.[16] However, although this proof is not based on (1), it nevertheless proves the same. Consequently, it is also an empty proof proving a tautology. Thus, it cannot escape the extensional fallacy. This fallacy depends not on the specific proof but rather on the interpretation of what is proven.

Because his second proof is based not on the anti-diagonal β but rather on the diagonal β', it does not rely on (2). In contrast to his first proof, Turing's second proof contains a more detailed description of a machine, $\mathcal{H}'$, that is stipulated to compute β'. In particular, Turing describes how to compose $\mathcal{H}'$ from $\mathcal{D}$ and the demonstrable existing universal machine $\mathcal{U}$ such that $\mathcal{H}'$ is circle-free iff $\mathcal{D}$ is circle-free. As Turing explains, $\mathcal{H}'$ and, therefore, $\mathcal{D}$ are meant to be circle-free because they are stipulated to *compute* β'.[17] However, in the case that $\mathcal{H}'$ computes the value of the position of β' that is related to its own S.D., $\mathcal{H}'$ cannot be circle-free because the computation of the value of the function depends on the value of the very same function.[18,19] Therefore, $\mathcal{H}'$ and, consequently, $\mathcal{D}$ are simultaneously defined as circular and circle-free. No machine that is proven not to exist is specified, but an inconsistent concept that is unsatisfiable by definition is provided.

3 Proof of the Impossibility of FOLD

The aim of Turing[20] is to provide a negative solution to Hilbert's decision problem: it is impossible to define a machine (denoted in the following by FOLD, for "FOL Decider") that can decide whether any given FOL formula is an FOL theorem. Turing intends to prove the impossibility of FOLD by showing that the existence of FOLD implies the existence of machines that have already been proven to be impossible, such as $\mathcal{E}$ and $\mathcal{D}$. Based

[16] Cf. Turing (1936: 246).
[17] Cf. Turing (1936: 247).
[18] Cf. Turing (1936: 247).
[19] Floyd (2012), section 2.2.3, and Floyd (2016a), section 4.5, emphasize the peculiarity of this sort of argument that shows that applying a rule in the special case of self-application is empty rather than contradictory. I agree. However, I go further than Floyd in arguing that *what* Turing argues for is empty. In this respect, Turing's second proof suffers from the same problem as his first proof.
[20] Turing (1936).

on the analysis presented in section 2, one might expect a proof showing that the concept of FOLD is inconsistent and, therefore, that the mere assumption of FOLD is meaningless. However, this is not the case. In fact, Turing's impossibility proof for FOLD differs significantly from his proofs of the impossibility of $\mathcal{D}$ in its additional element of a formalization for machines.

3.1 FOLD vs. $\mathcal{D}$

Unlike $\mathcal{D}$, $\mathcal{H}$, $\mathcal{H}'$ or $\mathcal{E}$, FOLD is not defined by its ability to decide some property of machines. Instead, FOLD is defined to address a purely logical problem: to decide whether some logical formula follows from a finite set of logical formulas. Unlike the assumption of $\mathcal{D}$, the mere assumption of FOLD is not inconsistent because the concepts of the provability and decidability of FOL-formulae are well defined. Thus, although one has not yet specified a machine that satisfies the requirements to be called FOLD, the concept of such a machine is consistent.

That the existence of FOLD implies the existence of a machine $\mathcal{E}$ (and, consequently, a machine $\mathcal{D}$) follows only from the presumption that for every machine $\mathcal{M}$, some formalization of $\mathcal{M}$ and its configurations exists such that the task of deciding on logical relations between logical formalizations is *interpretable* as the task of deciding whether $\mathcal{M}$ ever reaches a certain configuration. For example, deciding that the formalization of an arbitrary machine is provable is interpretable as deciding that the formalized machine is circle-free. Given this presumption, the task of deciding on the relevant logical formalization is indeed as inconsistent as the concept of $\mathcal{D}$. This is because it is impossible for decisions on formalizations to be correlated with the behaviours of formalized machines that include FOLD. This is evident, e.g., from a machine TFOLD$\overline{C}$ composed of (i) a translation machine T that translates machines, i.e. the sequences of instructions, and their states into their formalizations, (ii) FOLD, and (iii) a machine $\overline{C}$ that does not print 0 or 1 iff FOLD decides that the provability is positive. When applied to its own D.N., TFOLD$\overline{C}$ is circular iff TFOLD$\overline{C}$ is not circular according to the interpretation of the logical decision of FOLD.

According to Turing, this argument proves the non-existence of FOLD (as T and $\overline{C}$ trivially exist). However, in section 3.5, I argue that this conclusion is based on a fallacy. As an alternative to Turing's conclusion, one might question the presumption that for *every* machine $\mathcal{M}$ (including machines that contain FOLD), there is a correct formalization such that the logical implications of the formalizations are correlated with the sequence

of configurations of $\mathcal{M}$. However, for now, it is sufficient to note that a material mode of speech allows one to speak of "a circle-free machine $\mathcal{D}$" in the same manner as one may speak of "a machine FOLD" but that in doing so, one loses sight of the fact that one refers to a well-defined machine in only one of these cases. Unlike $\mathcal{D}$, FOLD is not non-existent in the sense of being defined inconsistently.

3.2 Turing's Undecidability Proof

Turing's proof of the impossibility of FOLD is based on his proof of the impossibility of a machine $\mathcal{E}$ that can decide whether an arbitrary machine $\mathcal{M}$ will ever print 0. The proof of the impossibility of $\mathcal{E}$, in turn, proves that $\mathcal{E}$ must contain $\mathcal{D}$ in addition to other well-defined machines. Therefore, it is based on the proof of the impossibility of $\mathcal{D}$. I ignore the details of the proof of the non-existence of $\mathcal{E}$ because the principles of this proof are similar to those considered in section 2.

The specific type of configuration (or even state[21]) is irrelevant to undecidability proofs of FOL; the question is whether $\mathcal{M}$ ever reaches that configuration (or state). Modern proofs of the undecidability of FOL are predominantly based on the impossibility of a machine that solves the halting problem instead of the impossibility of a machine that can decide whether a given symbol is ever printed. Turing refers to $\mathcal{E}$ instead of $\mathcal{D}$ because no "circular" or "circle-free" configuration exists. In the following, I generally abstain from these peculiarities of Turing's undecidability proof. My critique concerns all undecidability proofs based on the formalization of machines and their configurations.[22]

An undecidability proof of the type presented by Turing provides an effective procedure for formalizing machines and their configurations and proves that a machine $\mathcal{M}$ reaches a specific configuration c iff the formalization of c logically follows from the formalization of $\mathcal{M}$ and its initial configuration. Therefore, given FOLD, one possesses a machine that can decide whether $\mathcal{M}$ will ever reach a certain configuration c.

[21] States are constituents of configurations. A configuration (in Turing's words, a "complete configuration") consists of the number of the scanned square, the complete sequence of all symbols on the tape, and the state of the machine (in Turing's words, the "m-configuration"), cf. Turing (1936: 232).

[22] In fact, my critique applies to all types of proofs of the undecidability of FOL (or fragments of FOL) because they all essentially involve logical formalization. Church, e.g., proved the undecidability of FOL shortly before Turing by formalizing recursive functions instead of machines. However, I abstain from undecidability proofs that are not based on the concept of a machine to keep the terminology as simple as possible.

Turing formalizes the existence of a configuration containing a printed 0 with the following formula:

$$\exists s \exists t(N(s) \wedge N(t) \wedge R_{S_1}(s,t)) \tag{3}$$

The intended interpretation of $N(x)$ is "x is a natural number", and that of $R_{S_1}(x,y)$ is "in the complete configuration x (of $\mathcal{M}$), the symbol on the square y is [0]"[23]. $\mathcal{M}$ refers to the formalized machine; the configurations and squares of the machine are symbolized by numbers. In the following, I use Δ to refer to some formalization of the existence of a specific configuration (or state) in question. Therefore, Turing's formula (3) is a specific instance (FOL-formula) of the meta-variable Δ. In undecidability proofs based on the halting problem, Δ represents a formula that formalizes the existence of a halting state. For simplicity, I will also use Δ as an *abbreviation* for some formalization of the existence of some specific configuration (or state) and Γ as an abbreviation for some formalization of a machine $\mathcal{M}$ and its initial configuration.

Turing introduces additional intended interpretations, $\Im_i$, to formalize (i) the successor function, (ii) the description of the scanned square, and (iii) the description of the state of a machine. On this basis, he describes a general formalization procedure for constructing a formula Γ that represents the instructions of a given machine $\mathcal{M}$ and its initial configuration. Similar to Turing, I abbreviate the formalization of these instructions as $Des(\mathcal{M})$. In addition, I denote the formalization of the initial configuration by $I(\mathcal{M})$. $Des(\mathcal{M})$ and $I(\mathcal{M})$ are components of Γ. A proof of the undecidability of FOL in the manner used by Turing consists of a proof of the following equivalence:

BC: $\mathcal{M}$ reaches configuration c iff $\Gamma \vdash \Delta$.

The intended interpretation $\Im_i$ of Δ on the right-hand side is identical to the statement on the left-hand side. Therefore, the following equivalence is also intended to hold: $\Im_i(\Delta)$ is true iff $\mathcal{M}$ reaches configuration c. Turing abbreviates his formula of the conditional $\Gamma \rightarrow \Delta$ that corresponds to the right-hand side of BC as $Un(\mathcal{M})$. Therefore, in his case, the following is to be proven: $\mathcal{M}$ will print 0 iff $\vdash Un(\mathcal{M})$.

Turing proves each direction of the equivalence with a separate lemma. Lemma 1 proves the left-to-right direction. In this case, Turing's proof provides a schema for proving $Un(\mathcal{M})$ given that $\mathcal{M}$ prints 0. In contrast

[23] Turing (1936: 259).

to Lemma 1, the proof of Lemma 2, which proves the right-to-left direction, is very short and does not refer to any syntactic considerations. Instead, the proof of Lemma 2 is based on semantics. Because my criticism refers to Lemma 2, I quote the complete proof below:[24,25]

> LEMMA 2. *If $Un(\mathcal{M})$ is provable, then S_1 [i.e. 0] appears on the tape in some complete configuration of $\mathcal{M}$.*
>
> If we substitute any propositional functions for function variables in a provable formula, we obtain a true proposition. In particular, if we substitute the meanings tabulated on pp. 259-260 in $Un(\mathcal{M})$, we obtain a true proposition with the meaning "S_1 appears somewhere on the tape in some complete configuration of $\mathcal{M}$".

In the following, I show that this proof rests on a fallacy.

3.3 Empty vs. Semantic Proofs

Taken literally, Turing's proof of Lemma 2 considers the instantiation of $Un(\mathcal{M})$. This formula is provable by supposition. Therefore, its instantiation is a tautology. Consequently, one might apply the analysis presented in section 2 and object to Turing's claim that no proposition with a certain meaning is proven because a tautology is meaningless.

However, this literal interpretation does not do justice to Turing's proof. $Un(\mathcal{M})$ is of the form $\Gamma \rightarrow \Delta$. The proposition "$S_1$ appears somewhere on the tape in some complete configuration of $\mathcal{M}$" (or "$\mathcal{M}$ reaches configuration c" in general) is an instance of (3) (or Δ in general). Turing's proof of Lemma 2 argues for the truth of this proposition given $\vdash \Gamma \rightarrow \Delta$. Because of the correctness and completeness of FOL, one can also use $\forall \mathfrak{F}(\mathfrak{F}(\Gamma) = F \vee \mathfrak{F}(\Delta) = T)$ instead of $\vdash \Gamma \rightarrow \Delta$. Therefore, the implication from right to left in BC can also be written as follows:

BC': If $\forall \mathfrak{F}(\mathfrak{F}(\Gamma) = F \vee \mathfrak{F}(\Delta) = T)$, then $\mathfrak{F}_i(\Delta) = T$

Turing's proof uses universal quantifier elimination on the left-hand side of BC', replacing $\mathfrak{F}$ with $\mathfrak{F}_i$. In addition, he presumes that $\mathfrak{F}_i(\Gamma) = T$ because $\mathcal{M}$ and its initial configuration are given and $\mathfrak{F}_i(\Gamma)$ is nothing but a description of $\mathcal{M}$ and its initial configuration. Then, $\mathfrak{F}_i(\Delta) = T$ follows through

[24] Turing (1936: 277).

[25] By "the meanings tabulated on pp. 259-260", Turing refers to the intended interpretations of the function variables used in $Un(\mathcal{M})$, such as $N(x)$ and $R_{S_1}(s,t)$; see above, p. 12.

the application of disjunctive syllogism. Contrary to some arbitrary $\Im_i$ of a formula $\Gamma \rightarrow \Delta$ that is assumed to be provable, the intended interpretation, $\Im_i(\Delta)$, is a meaningful proposition. Thus, proving that the intended interpretation of Δ is true, given that $\Gamma \rightarrow \Delta$ is provable, is not meaningless.

However, Turing's proof of Lemma 2 presumes that for any machine $\mathcal{M}$, formulas Γ and Δ exist such that Turing's intended interpretations, $\Im_i(\Gamma)$ and $\Im_i(\Delta)$, are among the (logically possible and, thus, admissible) interpretations $\Im$ of Γ and Δ. My criticism of Turing's proof is related to this presumption.

Contrary to the empty proofs by contradiction of the non-existence of $\mathcal{D}$, Turing's undecidability proof of FOL rests on several assumptions that together imply that some unsolvable problem should be solvable by applying FOLD to decide on the relevant formalizations. In addition to the (auxiliary) assumption of the existence of FOLD, it is assumed that for *any* machine $\mathcal{M}$, there exists a correct formalization $\Gamma \rightarrow \Delta$ that allows one to infer from $\vdash \Gamma \rightarrow \Delta$ that $\mathcal{M}$ reaches a certain configuration c.

Turing justifies the correctness of his formalization by means of a general semantic principle sP, which may be summarized as follows (cf. the first sentence of the proof of LEMMA 2 quoted above):

sP: Any instance of a provable formula is a true proposition.

The problem with sP is its implication that any instance of a provable formula is also an *admissible* interpretation of that formula. As we will see, this is not the case.

In applying sP, Turing refers explicitly to his intended interpretations of function variables (predicates). In addition, he refers implicitly to the ordinary truth functional interpretation of logical constants. Other undecidability proofs than Turing's do not refer explicitly to sP. However, they also rest on a semantic justification of Lemma 2 (or some analogous lemma) that implicitly presumes sP. For example, they presume the contrapositive of sP by arguing that Δ does not follow from Γ if the intended interpretation of Δ is false but the intended interpretation of Γ is true.[26]

sP concerns the interpretation of FOL formulas. Therefore, it goes beyond pure formal logic. Note that sP also reaches outside pure formal semantics because the intended interpretations are provided in terms of specific expressions that instantiate logical formulas or their parts. Formal semantics, however, need not presume more than that formulas are interpreted from truth values and function variables are interpreted from sets. It does

[26] Cf. Boolos et al. (2003: 130).

not depend on the assumption that specific sentences instantiate formulas that indeed refer to truth values, nor the assumption that specific ordinary predicates instantiate function variables that indeed refer to sets. Therefore, sP does not follow from the correctness of FOL that is justified by formal semantics independent of specific instantiations of formulas. Indeed, the correctness of a calculus does not imply that specific sentences instantiating formulas satisfy the principles of formal semantics. Those sentences and their constituents may, e.g., not satisfy the principle of extensionality. Clearly, a proof of correctness does not imply that arbitrary machines and their configurations are capable of being correctly formalized.

As I illustrate in the following section, sP is not a valid principle. Indeed, numerous counter-examples exist. Recently, problems with logical formalizations and the methodological difficulty of justifying such formalizations have been discussed in great detail by logicians with philosophical backgrounds.[27] The problem of finding and justifying *correct* formalizations is only one aspect of adequate logical formalizations. It seems that the widespread praxis of logical formalization conceals the fact that formalizations require justification and may fail. As I illustrate in the following, the instantiation of logical formulas is not sufficient to conclude that inferences between instances also behave according to the laws of logic.

3.4 Counter-examples for sP

In specifying counter-examples for sP, I consider only "effective formalizations". Effective formalizations are defined by purely syntactic considerations. They simply involve replacing grammatical predicates with logical predicates. However, effective formalizations do not imply that the logical constants of the formalizations correspond to paraphrases of the logical constants in the formalized propositions. For example, propositions of the form "Fs are Gs", such as "humans are mortals", may be effectively formalized as $\forall x(Fx \rightarrow Gx)$. However, effective formalizations do not (i) refer to the meanings of predicates, (ii) refer to inferences of propositions, (iii) consider the context dependence of the meanings of expressions, or (iv) logically analyse propositions to reveal their "real logical forms". Turing's formalizations are effective formalizations. Therefore, I consider only problems relating to effective formalizations.

To question sP, one might consider problems that relate to the truth-functional definition of "$\rightarrow$", such as the paradox of implication or the so-

[27] Cf., e.g., Peregrin and Svoboda (2017), Peregrin and Svoboda (2013), Baumgartner and Lampert (2008), Brun (2004), and Sainsbury (2001).

called "ex falso quodlibet" or "verum ex quodlibet". In these cases, formulas are provable, whereas their instances are true only for a certain meaning that is not intended. In particular, when a given formula Γ is contradictory, $\Gamma \rightarrow \Delta$ is trivially logically valid, and $\Im_i(\Delta)$ may still be false. However, I do not consider examples of this sort because the critical point of Turing's proof concerns the interpretation of predicates ("function variables"). Instead, I take for granted the truth-functional interpretations of logical constants and the satisfiability of any formula Γ.

Thus, let us consider several typical problematic instances of valid argument schemata in FOL based on the interpretation of predicates. Sainsbury[28] offers the following example (4)/S2:

$$\forall x(Fx \rightarrow Gx), Fa \vdash Ga \tag{4}$$

(4) is a valid argument schema (and therefore, the corresponding implication is a provable formula). Commonly, propositions of the form "Fs are Gs. a is F. Therefore, a is G" are identified as instances of (4). However, only one of the following instances is valid:

S1: Humans are sensitive to pain. Harry is a human. Therefore, Harry is sensitive to pain.

S2: Humans are evenly distributed over the Earth's surface. Harry is a human. Therefore, Harry is evenly distributed over the earth's surface.

This example shows that one must distinguish admissible and inadmissible instances to avoid invalid instances of a provable formula. Instances are inadmissible iff their effective formalizations permit inferences on the formal side that do not correspond to valid inferences of the formalized propositions. Thus, instances are admissible iff the respective effective formalizations are correct. One cannot presume that instances of formulas behave according to the logical rules that apply to the formulas. Roughly speaking, the "real logical form" of an inadmissible instance is not identical to the logical form of its effective formalization. In the case of S2, e.g., one would usually formalize "... is evenly distributed over the earth's surface" as a second-order predicate. Consequently, in S2, "humans" refers to all humans collectively, not to each individual human. However, we are not concerned with non-effective formalization strategies for immunizing sP against counter-examples. Instead, we are concerned with counter-examples for sP that arise because of effective formalizations.

[28] Sainsbury (2001: 50).

A predicate in ordinary language cannot necessarily be considered an admissible instance of a predicate in FOL. Another well-known example of a predicate that does not behave as a predicate of FOL is the predicate "to exist". Often, it is better formalized by the existential quantifier than as a first-order predicate, e.g., to refute ontological proofs of the existence of God.

Predicates of another sort are related to so-called "opaque contexts", in which certain positions of predicates do not refer to objects in the domain. The following example (5)/S4 is attributable to Montague:[29]

$$\exists x(Fax \wedge Gx) \vdash \exists xGx \tag{5}$$

S3: Hans loves a woman. Therefore, a woman exists.

S4: Hans seeks a unicorn. Therefore, a unicorn exists.

(5) is usually accepted as a correct formalization of propositions such as S3. Therefore, let us effectively formalize propositions of the form "someone Fs a G" as $\exists x(Fax \wedge Gx)$. However, S4 is not valid in a natural reading. Montague uses this as an argument for his intensional logic. Quine argues that "x seeks y" is not an admissible instance of a first-order predicate because the position to the right of "seek" is opaque (non-referential).[30] As in the previous case, we are not concerned with the analysis of this specific example; instead, we are concerned only with the fact of inadmissible instances and, consequently, incorrect effective formalizations.

Another type of counter-example for sP is induced by predicates that can be diagonalised, such as "x is false" or "x is a predicate that does not apply to itself". In such cases, even the most plain logically valid argument schema yields instances that are not clearly valid. Consider, e.g., the following valid argument schemata of propositional logic:

$$\vdash \neg(P \leftrightarrow \neg P) \tag{6}$$

$$\vdash P \vee \neg P \tag{7}$$

As long as one does not diagonalise predicates such as "x is false" by substituting a name for x that refers to the resulting phrase, valid instances of (6), and (7) result. However, in the special case of diagonalisation, one might argue that the resulting propositions are true iff they are not true or that they are not either true or false (and nothing else). From this, one

[29] Montague (1966: 266).
[30] Cf. Quine (1960), §30.

might infer that the respective instances of (6), or (7) are not true. Again, one may provide various types of strategies for analysing and/or avoiding this situation. One might, for example, reject the presumption that instances of x are capable of referring in the diagonal case. Diagonalisation, so to speak, induces opaque positions. Alternatively, one might reject the application of logic in the diagonal case because the resulting propositions do not satisfy the bivalence principle. As another alternative, one might introduce a distinction between meta-language and object language to avoid incorrect effective formalizations. However, for our purposes, it is sufficient to accept the possibility of incorrect effective formalizations. The instantiation of logical formulas is no guarantee that their instances behave as their logical counterparts do with respect to inferential implications.

This enumeration of various types of counter-examples for sP is far from complete. There are no clear-cut criteria that guarantee a correct effective formalization; what seems to work in some cases might fail in syntactically similar cases. Furthermore, no consensus exists regarding how to analyse such cases or how to avoid incorrect formalizations. However, all that is relevant with regard to Turing's proof is that sP is not a valid principle on which a proof can be based. One cannot infer from the provability of a formula that a particular intended interpretation is true. Indeed, it may well be that the intended interpretation is not included among the admissible instances of a provable formula.

3.5 Fallacy of Substitution

I call the fallacy that arises from inferring the truth of an instance from the fact that it is an instance of a provable formula (or a valid argument schema) "the fallacy of substitution". This fallacy rests on the unreliable semantic principle sP. Turing's proof of Lemma 2 is a straightforward example of this fallacy. Contrary to logical fallacies, such as "affirmation of the consequence", fallacies of substitution cannot be detected through logic. Instead, *prima facie*, those fallacies seem to be justified by logic. Only an argument that reaches beyond logic by pointing to the problem of inadmissible interpretations can reveal that a straightforward application of logic is not justified.

This situation poses a general problem facing all types of arguments concerned with instances of a logical formula: how can one justify the admissibility of the instances under consideration? Any justification must evidently extend beyond logic. This is particularly applicable to meta-logical proofs, such as Turing's undecidability proof. Such proofs judge the properties of

pure, formal logic, such as the decidability of FOL-provability, by representing these properties within the language of FOL. Such a representation makes use of intended (or "standard") interpretations. Thus, it must be justified that these interpretations are, in fact, admissible in all cases (including diagonalisation).

Therefore, one might reject the entire concept of judging upon formal properties, such as decidability and provability, by employing intended interpretations.[31] However, in the following section, I explain why Turing's proof provides a reason not only for such general doubt but also for specific doubts regarding the use of sP to formalize the input of machines including FOLD (or some logic machine in general) in the special case of diagonalisation.

3.6 Formalization of the Diagonal Case

Turing's machines are not physical machines but rather sets of instructions. Starting from some initial configuration, they induce a sequence of configurations. Turing translates instructions and configurations into ordinary propositions that are instances of his logical formalizations. I do not criticize this translation. The problem of correct formalizations is not a problem specific to the formalization of natural language. Instead of ordinary propositions, one might use a standardized or artificial language to describe machines and their configurations. I simply identify configurations with their descriptions in terms of sentences that are instances of logical formulas.[32] The problem of correctly formalizing a machine $\mathcal{M}$ is, therefore, identical to the problem of formalizing the description of $\mathcal{M}$'s configurations. The question is whether the logical consequences of the formalization of a machine's instructions and its initial configuration correspond to the machine's sequence of configurations.

Turing's proof presumes that for *any* machine $\mathcal{M}$, some correct formalization exists. According to his proof, FOLD would make it possible to decide whether *any arbitrary* machine $\mathcal{M}$ can ever reach a certain configuration based on the corresponding formalization $\Gamma \rightarrow \Delta$. Proofs of the impossibility of a machine are based on critical diagonal cases of machines that contain the machine that is proposed to decide a given property of

[31] This seems to be the position of Wittgenstein, cf. Wittgenstein (1967), part I, appendix I in relation to Gödel's proof, cf. especially the "notorious paragraph" §8, where Wittgenstein recommends "to give up the interpretation" instead of inferring that Gödel's formula G is undecidable.

[32] Below, I also call these instantiated formulas simply "descriptions" for brevity. This ambiguous use is conventional, cf., e.g., Boolos et al. (2003: 130).

that machine. The proof proves that there is no consistent solution in such cases. When the problem in question is to be solved by applying FOLD, the critical diagonal case is a machine $\mathcal{M}_{\mathcal{D}}$ that contains FOLD evaluating the formalization of $\mathcal{M}_{\mathcal{D}}$. The question is whether in such cases, the existence of a correct formalization is ensured: are the logical consequences of logical formulas correlated with sequences of configurations of $\mathcal{M}_{\mathcal{D}}$?

Turing infers the truth of $\Im_i(\Delta)$ by assuming $\Gamma \vdash \Delta$. His conclusion is valid only when a formalization Γ exists for $\mathcal{M}$ and its initial configuration such that $\Im_i(\Gamma) = T$ and $\Im_i$ is an admissible instance of $\Im(\Gamma)$, cf. the argument based on BC' on p. 13. In the following, I challenge this presumption with respect to the formalization of $\mathcal{M}_{\mathcal{D}}$.

In his formalization of machines $\mathcal{M}$, Turing considers only machines with the initial configuration defined by an empty tape. His formalization of the initial configuration is the *same* for *all* $\mathcal{M}$, namely, $\forall y R_{S_0}(u, y) \wedge I(u, u)$ (u refers to the number that has no predecessor, i.e., 0).[33] According to Turing's intended interpretation, this formula is instantiated by "all squares in the initial configuration (of $\mathcal{M}$) are empty, and the initial square is scanned". The restriction to machines with initially empty tapes is remarkable because Turing also considers machines that start with a certain standard description S.D. or description number D.N. The consideration of such "meta"-machines is essential for considering impossible machines that decide their own properties. Furthermore, Turing's "universal machine", $\mathcal{U}$, is a machine that prints the configurations of a machine with the S.D. from which $\mathcal{U}$ begins. For all these meta-machines, the intended interpretation $\Im_i$ of a formula Γ that contains $\forall y R_{S_0}(u, y) \wedge I(u, u)$ is trivially false. Thus, Turing seems to presume in his formalization of $\mathcal{M}$ that meta-machines occur only within composed machines that start with empty tapes. One might imagine a machine M_D that starts with an empty tape that first generates the D.N. of a machine that starts with an empty tape, then generates Γ from D.N. and Δ, and finally returns $\Gamma \rightarrow \Delta$ (or the respective binary code) to FOLD for evaluation.

Following Post, one alternative is to release the restriction to machines $\mathcal{M}$ that start with empty tapes. Furthermore, any machine must function whether it receives its input from another machine or directly. In particular, one must be able to interpret the relationship between the input and output of a machine in terms of a function that the machine $\mathcal{M}$ computes independently of how $\mathcal{M}$ receives its input. Finally, it does not matter whether we consider the input to FOLD in terms of logical formulas or the binary code.

[33] Cf. Turing (1936: 260).

Therefore, let us assume in the following that FOLD is initialized directly with formalizations of machines $\mathcal{M}$ and their initial configurations.

Let us consider an assumed machine $\mathcal{M}_{\mathcal{D}}$ denoted by FOLD$\overline{C}$, cf. p. 10, that starts from its own formalization Γ. The formalization $Des(\mathcal{M})$ of the instructions of the machine, as one part of Γ, poses no problems. In particular, Turing's formalization procedure ensures that $Des(\mathcal{M})$ is satisfiable. However, the formalization $I(\mathcal{M})$ of the initial state, the other essential part of Γ, is undefinable in this case. Regardless of what formula is intended to describe the initial configuration, some part of this formula must describe exactly this formula (or its binary code). This is impossible because the description of the symbols on a tape must be longer with respect to the filled squares than the symbols described on the tape. In particular, this applies to Turing's means of formalizing the configurations of his machines. It is impossible to state, without more than one symbol filling only one square, what symbol is described and where it is within a certain configuration. Therefore, it is impossible to interpret some part of any formula Γ as describing precisely that same formula Γ. This may be stated in another way: to describe certain configurations, the description must have a certain logical form, and this condition is not satisfied in the diagonal case.[34] As in the case of the counter-examples for sP discussed in section 3.4, the syntax and formal semantics of FOL are simply not appropriate for the formalization of the problem in question. In this respect, there exists no formula Γ such that its intended interpretation, $\Im_i(\Gamma)$, describes the instructions *and* initial configuration of FOLD$\overline{C}$. This may be restated as follows: regardless of what is offered as a formalization Γ of FOLD$\overline{C}$ and its initial configuration, it cannot imply a correct formalization of the initial configuration of the tape in this case.

This reasoning for the non-existence of a correct formalization of FOLD$\overline{C}$ starting from its own formalization does not rely on the fact that the hypothetical assumed machine FOLD does not exist. This is so because the same argument is valid if FOLD is replaced by a well-known and existing theorem prover T_L. Because applying Lemma 2 presumes that $\Gamma \vdash \Delta$, no principal difference from FOLD exists in this case that is decidable by the "semi-decider" T_L. Assuming that $\mathrm{T}_L\overline{C}$ is initialized with its own formalization similarly results in a diagonal case in which the intended interpretation of the formalization implies that the machine decides its own behaviour. The impossibility of a correct formalization in such cases is based not on the non-existence of the relevant machine but rather on the fact that the in-

[34] Cf. Wittgenstein (1994), remarks 3.33 and 3.332.

tended interpretation $\Im_i$ is not among the admissible interpretations of any logical formula. It is not the logic machines that cannot exist but rather the correctness of their formalizations in the diagonal case. To maintain the contrary is similar to inferring that the counter-examples discussed in section 3.4 do not exist because there are no correct effective formalizations of those examples.

Even if one attempts to circumvent this argument by preceding FOLD with another machine whose input is described by Γ, the problem remains that FOLD must be capable of deriving descriptions from Γ that must, in turn, be described. Consider, e.g., a machine CTFOLD$\overline{C}$ such that a copy machine C copies the D.N. of CTFOLD$\overline{C}$ and a translation machine T generates $I(CTFOLD\overline{C})$ and $Des(CTFOLD\overline{C})$ from each binary code and, finally, returns $\Gamma \rightarrow \Delta$ to FOLD. However, deriving descriptions from Γ gives rise to configurations that, in turn, involve descriptions that must be derived. In such a case, it cannot be presumed that the logical consequences of Γ correspond to the sequences of configurations of the machine: unlike in a non-diagonal case, it is impossible to correlate one-to-one derivations of formalizations of configurations from Γ with configurations of CTFOLD$\overline{C}$ starting from its own D.N. This fact affects the existence of a correct formalization and, therefore, the question of whether the intended interpretation is among the possible interpretations of a logical formula. However, there is no reason why the possibility of a purely logical decision concerning the provability of a logical formula should be affected by a circular definition of descriptions of configurations describing descriptions that are, in turn, described. The resulting problems are evidently problems of interpretation and not of algorithmic logic. Diagonalisation rules out an isomorphism between sequences of configurations and the logical relations of assumed formalizations. The result of this is that there is no correct formalization in the diagonal case.

Formalizing logic machines yields problems similar to those that arise in the case of formalizing diagonalisable predicates, cf. above p. 17. The correctness of a formalization is not ensured in the diagonal case. This says nothing about the possibility of diagonal functions, only about the possibility of correct logical formalizations in such cases.

According to this argument, there is no reason to maintain the impossibility of deciding FOL-theoremhood. When a correct formalization of some proposition, P, is presumed, FOLD could also be used to decide whether P is true. However, not all propositions are decidable in this way because not all propositions are definable within the language of FOL. It is, for example, not possible to decide through logical formalization whether *any arbitrarily*

chosen machine $\mathcal{M}$ ever halts.

4 Undecidability vs. Undefinability

According to the analysis presented here, proofs of the impossibility of $\mathcal{D}$ or FOLD prove not undecidability theorems but undefinability theorems. In the case of $\mathcal{D}$, such a machine that is intended to be circle-free is not definable. In the case of FOLD, such a machine is definable and may exist, but FOLD and its configurations are not fully definable within the language of FOL.

When one interprets Turing's proofs as undecidability proofs, one is misled by a material mode of speech that interprets certain instances of logical formulas, without reservation, as statements about machines. Instead, one should first ask whether the relevant instances are consistently interpretable in terms of statements about machines. Any intended interpretation of a logical formula makes use of some other language in place of FOL. The relationship between the language used for interpretation and FOL is not unique. Instances of provable formulas need not be meaningful or true; indeed, they may well (i) lack sense by virtue of being tautological, (ii) be nonsensical in that they lack unambiguous truth values, or (iii) be false. Concluding that any instance of a provable formula is a meaningful and true proposition is a fallacy, be it an extensional fallacy or a fallacy of substitution. Turing's undecidability proof of FOL is based on these two types of fallacies. Therefore, his proof is invalid. There is no compelling reason to give up the search for a decision procedure for FOL. The problems encountered when defining such a procedure are logical in nature and cannot be resolved by considerations that are beyond pure logic.

References

Baumgartner, M. and T. Lampert (2008). "Adequate Formalization". *Synthese* 164: 93-115.

Boolos, G. S., J. P. Burgess and R. C. Jeffrey (2003). *Computability and Logic*, 4th edition. Cambridge, UK: Cambridge University Press.

Brun, G. (2004). *Die richtige Formel. Philosophische Problem der logischen Formalisierung*. Frankfurt A.M.: Ontos.

Floyd, J. (2012). "Wittgenstein's Diagonal Argument: A Variation on Cantor and Turing". In P. Dybjer, S. Lindström, E. Palmgren and G. Sundholm (eds.), *Epistemology versus Ontology*. New York: Springer, pp. 25-44.

Floyd, J. (2016). "Turing on "Common Sense": Cambridge Resonances". In J. Floyd and A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing – Turing 100*. New York: Springer, expected date of publication 2016, pp. 1-52.

Floyd, J. (2016). "Chains of Life: Turing, Lebensform, and the Emergence of Wittgenstein's Later Style". *Nordic Wittgenstein Review* 5(2): 7-89.

Floyd, J. and H. Putnam (2000). "A Note on Wittgenstein's 'Notorious Paragraph' about the Gödel Theorem". *The Journal of Philosophy XCVII* 11: 624-32.

Lampert, T. (2017). "Wittgenstein and Gödel: An Attempt to make 'Wittgenstein's Objection' Reasonable". *Philosophia Mathematica*, in print.

Montague, R. (1966). "The Proper Treatment of Quantification in Ordinary English". In R. H. Thomason (ed.), *Formal Philosophy. Selected Papers*. New Haven, CT: Yale University Press, pp. 247-70.

Peregrin, J. and V. Svoboda (2013). "Criteria of Logical Formalization". *Synthese* 190: 2897-924.

Peregrin, J. and V. Svoboda (2017). *Reflective Equilibrium and the Principles of Logical Analysis. Understanding the Laws of Logic*. New York: Routledge.

Quine, W. V. O. (1948). "On What There Is". *The Review of Metaphysics* 2: 21-38.

Quine, W. V. O. (1960). *Word and Object*. Cambridge, MA: MIT Press.

Rodych, V. (1999). "Wittgenstein's Inversion of Gödel's Theorem". *Erkenntnis* 51: 173-206.

Sainsbury, M. (2001). *Logical Forms*, 2nd edition. Oxford: Blackwell.

Simmons, K. (1993). *Universality and the Liar: An Essay on Truth and the Diagonal Argument*. New York: Cambridge University Press.

Sylvan, R. (1992). "Grim Tales Retold: How to Maintain Ordinary Discourse About–and Despite–Logically Embarrassing Notions and Totalities". *Logique & Analyse* 35: 349-74.

Turing, A. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society* 2(42): 230-65.

Wittgenstein, L. (1967). *Remarks on the Foundations of Mathematics*. Cambridge, MA: M.I.T. Press.

Wittgenstein, L. (1994). *Tractatus Logico-philosophicus*, trans. D. F. Pears and B. F. McGuinness. London: Routledge.