THE UNDERDETERMINATION OF THE CHURCH-TURING THEOREM

TIMM LAMPERT

Humboldt University Berlin, Unter den Linden 6, D-10099 Berlin
e-mail address: lampertt@staff.hu-berlin.de

Abstract. This paper argues that it cannot be proven that all computable functions are expressible in a language based on first-order logic (FOL) without presuming that FOL is undecidable. Since undecidability proofs of FOL presume the expressibility of computable functions within a language based on FOL, undecidability proofs of FOL are analysed as being underdetermined. They, in fact, prove that either FOL is undecidable or not all computable functions are expressible within a language based on FOL. The critique of the expressibility of computable functions within FOL is based on an analysis of the diagonal method and its application in undecidability proofs.

keywords: Church-Turing Theorem, Church-Turing Thesis, semantics and computation, decision problem, diagonalization

1. Introduction

Undecidability proofs of FOL rest on proofs concerning the expressibility of computable functions within a language based on FOL. It is argued that a contradiction follows from expressing a decision function for FOL within a language based on FOL in the case of diagonalization. This contradiction is used to reduce the assumption that FOL is decidable to absurdity. This section argues that this conclusion is not conclusive because the possibility to express a decision function for FOL within a language based on FOL cannot be proven without assuming that FOL is undecidable. Thus, the contradiction may likewise be used to reduce the assumption to absurdity that a decision for FOL is expressible within a language based on FOL. By analysing proofs based on the diagonal method as proofs of undefinability theorems, it is argued that this alternative conclusion is straightforward.

This argument is inspired by Wittgenstein's critique of undecidability proofs (cf. [Wittgenstein (1967)], Part I, Appendix I and Part V, §§18f.). Wittgenstein's critique, however, is very general and related to Gödel's proof of the incompleteness of axiomatic systems of arithmetic (like *Principia Mathematica PM* or *Peano Arithmetic PA*). In contrast, I relate the analysis of undecidability proofs as being underdetermined to Turing's and Church's proof of the undecidability of FOL and show, which step in these proofs is questioned for what reasons. These proofs are based on a contradiction following from expressing an *hypothetical assumed decision function* for decidability of FOL in a language based on FOL. The question is what to infer from *this contradiction*. In contrast, Gödel's proof does not consider expressing provability as a computable function (cf. Def. 46 in

[Gödel (1986)], p. 171, where Gödel defines 'x is a provable formula' by a notion 'of which we cannot assert that it is recursive'). His proof is based on the much weaker assumption of recursively defining 'y is a proof of x' and expressing this recursive function as an open formula that he abbreviates as yBx. Provability (in PM or PA) is merely expressed by binding the free variable y in yBx. Gödel's proof derives a contradiction through diagonalizing $\exists yyBx$ and assuming it or its negation to be provable within an axiomatic system such as PM or PA. This contradiction can also be used to show that these axiomatic systems are incomplete (given their (ω-)consistency). Such an option is not available in the case of undecidability proofs of FOL since FOL can be conceptualized in an axiom-free form. Wittgenstein's sort of critique should not be related to Gödel's proof in the first place but to Turing's and Church's proof. However, I will not relate my critique any further to Wittgenstein's remarks that give rise to many exegetical problems. Instead, this paper explains in standard terms why and how the critique of undecidability proofs as being underdetermined applies to Turing's and Church's proofs.

I introduce the terminology needed for this analysis in sections 2 to 4. Section 2 is related to expressing computable functions by Turing machines or recursive functions. Section 3 discusses the idea of expressing propositional functions by open formulas of a language based on FOL. Section 4 then analyses proofs by contradiction based on the diagonal method as inexpressibility theorems. Based on this analysis, I will analyse Turing's proof (section 5) and Church's proof (section 6) and argue that they can be analysed as inexpressibility proofs of a decision function for FOL within a language based on FOL instead of inexpressibility proofs of a decision function for FOL in a language of computation such as Turing machines or recursive functions.

2. Expressing a Computable Function

Undecidability proofs can be analysed as proofs that show that certain functions cannot be expressed by Turing-computable or recursive functions. This section specifies the notion of expressing a computable function.

The general problem in undecidability proofs is the question of whether certain functions are isomorphic.

Definition 2.1. Two functions are *isomorphic* iff

(1) their domains can be correlated in a one-to-one manner,
(2) their ranges can be correlated in a one-to-one manner, and
(3) all and only correlated values are assigned to correlated arguments.

Definition 2.2. Functions can be *mapped onto each other* iff they are isomorphic.

Definition 2.3. An *interpretation of a function* f is a set of rules for mapping a function g onto f.

Definition 2.4. A function f *expresses* a function g iff f is interpreted such that g is mapped onto f.

Remark 2.5. Typically, proofs that show that a function g cannot be expressed by functions f of a specific type are proofs by contradiction. For the sake of reducing an assumption to absurdity, one may assume that a function f expresses a function g although, in fact, f cannot express g. In that case, there is no interpretation of f that maps g onto f. Although no such interpretation exists, one '*intends* to interpret' f such that it can be mapped onto

g. The proof then shows that such an intention must fail: no set of rules can be specified to realize this intention. Thus, the *intended interpretation* of a function f of a specific type is not '*reliable*'. I will specify this terminology in relation to the interpretation of FOL expressions below. However, I will also apply it to functions, as explained in this remark.

I presume familiarity with the concepts of Turing machines and recursive functions. String functions computed by Turing machines provide a paradigm for computable functions. The question of the computability of a function g in which we are interested can be posed as the question whether string functions f can be interpreted such that g can be mapped onto f.[1] *Turing's thesis* states that any computable function can be expressed by a Turing-computable string function.

Definition 2.6. *Turing's thesis*: A function is computable iff it is expressible by a Turing-computable string function.

Church's thesis specifies an alternative statement based on number-theoretic recursive functions.

Definition 2.7. *Church's thesis*: A function is computable iff it is expressible by a recursive function.

Since one can specify rules for mapping Turing-computable string functions and recursive functions onto each other, these two theses are provably equivalent.

Definitions 2.3 to 2.7 suffer from the vagueness of the *concept of a rule* in Definition 2.3. However, there is no need to enter a discussion of this concept in this paper. By referring to rules, I wish to exclude non-standard mappings such as a mapping of natural numbers to a binary notation for natural numbers such that all and only binary strings ending with '0' are strings of numbers of halting Turing machines. Such a notation would make the halting problem decidable (cf. [Boker & Dershowitz (2008)], p. 206, applying [Shapiro (1982)], p. 17, corollary C2a). Such a non-standard mapping cannot be specified by any rules. It is merely specified by a general description that says nothing about *how* to realize the mapping. Therefore, the halting problem is not computable according to the above definitions, even though there may exist such a non-standard mapping relative to a non-standard notation that is not available through any interpretation in terms of a set of rules.

I also leave it an open question whether the concept of a rule implies the concept of computability, which would render the Church-Turing thesis circular. This is a well-known problem in the literature (cf., in particular, [Rescorla (2007)]) that this paper does not address. The same applies to the general problem of giving a precise definition of the Church-Turing thesis that specifies *in general, precisely and non-circularly* how to express arbitrary functions by Turing-computable string functions or recursive functions. This paper is not concerned with a critique of the Church-Turing thesis. Instead, I presume the standard terminological background for undecidability proofs. I consider the (im)possibility

[1] Cf., e.g., [Weihrauch (2000)], p. 3:

> Ordinary computability theory first introduces computable partial word functions $f\colon \subseteq \Sigma^* \to \Sigma^*$ explicitly, for example, by means of Turing machines. For defining computability on other sets M (rational numbers, finite graphs, etc.) words are used as 'names' or 'codes' of elements of M. While a machine still transfers words to words, the user interprets these words as names of elements from the set M.

of deciding FOL in a standard way under the assumption that no problems arise from the application of the Church-Turing thesis. It is uncontroversial that any program written in an ordinary computer language can be mapped onto a Turing-computable string function by applying rules of a concrete translation procedure that can itself be carried out by a computer program. Thus, the question at stake can be specified in a very concrete way, independently of the notorious problems with giving a precise *general* definition of a computable function: Do undecidability proofs indeed prove that the endeavour to specify a decision procedure for FOL-provability in an ordinary computer language is futile?

3. Expressing a Propositional Function

Computer languages in general and Turing machines or definitions of recursive functions in particular are languages for expressing computable functions. In contrast, FOL is, first and foremost, a language that can be used to express propositional functions by open formulas containing at least one free variable relative to an interpretation $\Im$. Interpretations $\Im$ of FOL expressions obey the *principles of extensional semantics*: a propositional variable is interpreted by a proposition that refers to one and only one truth value (principle of bivalence), a variable is interpreted by a set serving as the domain of quantification, a name symbol is interpreted by one and only one object of the domain, an n-place function symbol is interpreted by a function mapping n-tuples of the domain to values of the domain, an n-place predicate symbol is interpreted by a set of n-tuples from the domain, and logical connectives and quantifiers are interpreted as truth functions mapping atomic propositional functions such that the truth value of a complex proposition depends on nothing but the referential relations of its constituents.

Definition 3.1. An *interpretation* $\Im$ *of an FOL expression* φ assigns extensions (truth values, objects, or sets) to the respective constituents of φ in accordance with the principles of extensional semantics.

Definition 3.2. A *number-theoretical interpretation of an FOL expression* φ interprets φ with the natural numbers as the domain of quantification.

Definition 3.3. The language of arithmetic, L_A, is based on FOL, with '=' as the only predicate and with the addition of the constant '0', the one-place function symbol 'S', and the two-place function symbols '+' and '$\times$'. L_A is interpreted by its standard fixed number-theoretical semantics $\Im_A$.

While interpretations $\Im$ of FOL expressions are not fixed and are not necessarily number-theoretical in nature, the interpretation $\Im_A$ of L_A is fixed and number-theoretical. '0' is interpreted to refer to the number 0; 'S' is interpreted by the successor functions; '+', by addition; '$\times$', by the multiplication of natural numbers; and '=', by equality. I call this the 'standard fixed number-theoretical semantics of L_A'.

Definition 3.4. The *interpretation* $\Im_A$ *of an* L_A *expression* φ assigns extensions (truth values, natural numbers, or sets of n-tuples of natural numbers) to the respective constituents of φ in accordance with the standard fixed number-theoretical semantics of L_A.

L_A is 'a language based on FOL', i.e., translation rules can be defined that eliminate all the vocabulary extending FOL such that logical properties, e.g., satisfiability, are preserved. In the following, I abstain from considering languages other than L_A that are based on FOL.[2]

Definition 3.5. A *propositional function $f(x)$ is expressed by an open L_A- or FOL formula* $\varphi(x)$ relative to an interpretation $\Im$ iff for all x that satisfy $f(x)$, the objects of the domain $\Im(x)$ satisfy $\Im(\varphi)$.

This definition can be trivially extended to n-place propositional functions. Henceforth, I tacitly presume that considerations concerning one-place functions also extend to n-place functions.

Definition 3.6. An *intended interpretation* I of an L_A or FOL expression φ assigns ordinary expressions to the constituents of φ with the intention of providing an interpretation $\Im(\varphi)$.

Example 3.7. The intended interpretations of the vocabulary with fixed interpretations are $I(\neg)$ = 'not', $I(\wedge)$ = 'and', $I(\vee)$ = 'or', $I(\rightarrow)$ = 'if ... then ...', $I(\leftrightarrow)$ = 'if and only if', $I(\forall)$ = 'all', $I(\exists)$ = 'some', $I(0)$ = 'zero', $I(S)$ = 'successor of', $I(+)$ = 'plus', and $I(\times)$ = 'multiplied by'.

Remark 3.8. L_A or FOL formulas can be paraphrased as ordinary propositions through the use of intended interpretations.

Unlike interpretations ($\Im$), intended interpretations (I) do not obey the principles of extensional semantics *by definition*. Rather, intended interpretations can be *unreliable* (inadmissible, not permissible, or impossible).

Definition 3.9. An intended interpretation I of an L_A or FOL expression φ is *reliable* iff it provides an interpretation $\Im(\varphi)$.

This definition does not provide any *criterion* for identifying unreliable intended interpretations. The identification of (un)reliable interpretations is a non-trivial problem.[3] An unreliable intended interpretation may *intend* to provide a reliable interpretation of on open formula in terms of a (well-defined) propositional function. However, this intention may fail to be realized. The mere assignment of well-formed ordinary propositions to well-formed FOL expressions does not guarantee that the intended interpretations, in fact, yield well-defined propositional functions.

FOL is *sound* relative to interpretations $\Im$ but not to intended interpretations I since the latter can be unreliable.[4] One does not conclude that FOL is unsound because interpreting

[2]Definition 3.5 and the terminological distinction between expressing and capturing (cf. below Definition 4.1 are reminiscent of [Smith (2013)], p. 41.

[3]For example, [Peregrin and Svoboda(2017)], p. 70, propose an inferential 'criterion for reliability (REL)' that compares formal and informal inferential relations (cf., e.g., [Brun(2003)] for another extensive discussion of the topic). One might use REL to identify *unreliable* intended interpretations in the case that informal judgements are available; if, e.g., $I(\varphi)$ does not follow informally from $I(\psi)$ (i.e., the inference is not judged to be truth-preserving) but $\psi \vdash \varphi$, then $I(\varphi)$ and $I(\psi)$ cannot both be reliable. Of course, informal judgements can be assumed neither to be reliable nor to be available. This applies, in particular, to the intended interpretations of the hypothetical diagonal cases involved in undecidability proofs (cf. the following sections). However, criteria such as REL show that one cannot simply infer the validity of intended interpretations from the provability of the formulas that are interpreted. Independent evidence is needed. In the case of undecidability proofs, such independent evidence may well reside in the specification of a decision procedure, which would demonstrate the unreliability of the intended interpretations involved.

[4][Kreisel (1967)], pp. 152-154, deliberately distinguishes 'intuitive logical validity' (= logical validity according to an intended interpretation I) from 'truth in all set-theoretic structures' (= truth in all $\Im$),

the FOL proof $\exists x(Fax \wedge Gx) \vdash \exists x Gx$ by 'Jack seeks something that is a unicorn; therefore, something is a unicorn' yields an inference that is not truth-preserving (here making use of the following intended interpretations: $I(x)$ = the set of all beings, $I(F)$ = '... seeks ...', $I(a)$ = 'Jack', and $I(G)$ = '... is a unicorn'). Instead, one concludes that 'seeks' is not a predicate that obeys the principles of extensional semantics since its second position is not 'referential'. It may well be that intended interpretations of provable formulas φ of a language based on FOL are not true; in such a case, the intended interpretations are not reliable.

Example 3.10. $I(P)$ = 'This proposition is not true' is not reliable since it does not refer unambiguously to a truth value, according to the Liar Paradox. One cannot infer from the provable formula $P \leftrightarrow P$ that 'This proposition is not true iff This proposition is not true' is true.

Example 3.11. $I(F(x, y))$ = 'x is a member of y' and $I(x, y)$ = 'the set of all sets' are not reliable since these interpretations do not refer unambiguously to sets (Russell's Paradox). One cannot infer from the provability of $\neg\exists x\forall y(Fyx \leftrightarrow \neg Fyy)$ that the following statement is true: 'Not: Something is a set x of all sets y such that y is a member of x iff y is not a member of y' (more fluently, 'A set composed of all normal sets does not exist').

Remark 3.12. The analysis of paradoxes is controversial. Ramsey's distinction between semantic paradoxes (e.g., the Liar Paradox) and logical or set-theoretical paradoxes (e.g., Russell's Paradox) is prominent (but also controversial). Ramsey claimed that only the former show that certain expressions do not validly refer (or have a well-defined meaning), although one might think that they do, while the latter are veridical paradoxes proving that certain axioms or assumptions that one might think to be true are, in fact, false. According to this analysis, Russell's Paradox falsifies the unrestricted comprehension axiom schema of naive set theory (i.e., the schema $\exists x\forall y(y \in x \leftrightarrow \varphi(y))$. Such an analysis presumes that the intended interpretations of logical formulas are reliable in the case of logical or set-theoretical paradoxes. However, the analysis given in Example 3.11 contradicts this analysis; according to the analysis given in Example 3.11, the analysis of Russell's Paradox as a veridical paradox commits the *fallacy of intended interpretations*, as defined in Definition 3.13. However, Example 3.11 is given only for the sake of illustration. The reasoning put forward in this paper presumes neither any *analysis of paradoxes* nor any *general claim* concerning the possibility of expressing 'self-reference' within the language of logic. It is merely assumed that intended interpretations may violate the semantic principles of FOL (for whatever reason) and, thus, it does not go without saying that one can infer the

stating merely that the former implies the latter without *assuming* that the latter also implies the former. However, the assumption that provability implies 'intuitive logical validity' is assumed by Kreisel without any further argument, and this assumption is not deliberate ([Smith (2013)], p. 356, likewise confuses soundness with 'evident validity'). Kreisel uses this assumption as the basis of his erroneous 'informal rigorous proof' for the equivalence of provability, 'intuitive logical validity' and 'truth in all set-theoretic structures' given the equivalence of provability in a sound and complete calculus and truth in all set-theoretic structures. However, the soundness of FOL calculi is related to extensional semantics ('set-theoretic structures', in Kreisel's words), and there is no way to infer the truth of instances or intended interpretations of provable formulas from the soundness of FOL inference rules. One would instead need to show that the intended interpretations obey extensional semantics. However, there is no rigorous way to do this; discussions of non-extensional, so-called 'opaque' contexts are endless in the philosophy of logic. Typically, they are identified by arguing that instances or intended interpretations of *provable* FOL formulas cannot be judged to be true.

truth of intended interpretations of provable formulas. It is the analysis of logical or set-theoretical paradoxes as veridical paradoxes that risks sanctioning such an inference on the basis of Ramsey's controversial *general claim* concerning his distinction of paradoxes. This paper argues that any proof resting on such an inference should be justified by independent reasoning and that a critical analysis of established proofs must scrutinize whether such a reasoning is, in fact, provided.

Definition 3.13. The *fallacy of intended interpretations* is the inference of the truth of $I(\varphi)$ from the provability of φ.

Committing this fallacy is suggestive since (i) interpreting logical formulas by intended interpretations seems to sanction those interpretations as logical (= obeying the laws of logic and its semantics) and (ii) intended interpretations are intended to refer to set-theoretical structures, and it comes as a paradoxical, counterintuitive surprise when one realizes by some independent reasoning that they do not, in fact, refer to that to which they are intended to refer. Semantic intention or intuition with regard to reference relations is not a criterion for successful reference, however.

For the following, it is sufficient to acknowledge that intended interpretation may not be reliable and that, therefore, one cannot infer the truth of an intended interpretation from the provability of the formula it interprets. Independent evidence is needed to argue for the reliability of intended interpretations. This is especially true when diagonalization is involved (cf. the following section).

4. Inexpressibility Theorems and the Diagonal Method

The diagonal method can be analysed as a method of showing that a certain function cannot be expressed by a certain type of function. This section presents the details of this analysis.

The term 'diagonalization' stems from Cantor's application of the diagonal method by referring to the *diagonal of a matrix*. Cantor defines an *anti*-diagonal by a function g that assigns to each position n a value that differs from that in position n of the n-th entry of the matrix. This method of defining an anti-diagonal is used to prove that no function f defining a sequence of values in the matrix expresses g. To prove this, it is assumed that g would define a function enumerated in the matrix. One can refer to the enumerated entries in a matrix by indices. The diagonal method shows that g cannot be identified with some function f_n defining the values in the n-th row since the value in the n-th position of the sequence defined by f_n would not be well defined in that case. Diagonalization is either the process or the result of taking the index of a function f_n as its own argument. The diagonal method is the application of diagonalization to prove that no function f_n can express a function g defined over the enumeration of functions f since it is impossible to interpret any of the enumerated f_n as g.

A so-called *non-matrix-topological* definition can be formulated by defining the diagonalization of *open formulas* $\varphi(x)$ in a language based on FOL. Let $\ulcorner\gamma\urcorner$ be the numeral of the Gödel number of $\varphi(x)$; then, the process of replacing x with $\ulcorner\gamma\urcorner$—or the result of this process, $\varphi(\ulcorner\gamma\urcorner)$—is the diagonalization of $\varphi(x)$.

Undecidability proofs based on the diagonal method typically involve the application of the *Diagonalization Lemma.* This lemma presumes an axiom system that is at least as strong as the (very weak) Robinson Arithmetic Q of elementary arithmetic. Q can be formulated in L_A. Although it is not proven rigorously in an absolute sense that Q is consistent, Q

is assumed to be sound relative to $\Im_A$ (and, consequently, to also be consistent). Though one may question this assumption from a finitistic point of view, I take the soundness and consistency of Q for granted. In particular, Q is strong enough to *capture* diagonalization in terms of a recursive function *diag*. While the notation for expressing a function refers to the interpretation of a language (e.g., $\Im_A$ in the case of L_A), the notion of capturing refers to provability within a theory (cf. [Smith (2013)], p. 119):

Definition 4.1. A theory T *captures* the one-place numerical function f by means of the open wff $\varphi(x, y)$ iff, for any m, n,

(i) if $f(m) = n$, then $T \vdash \varphi(\overline{m}, \overline{n})$, and
(ii) $T \vdash \exists! y \varphi(\overline{m}, y)$.

Here, (ii) satisfies the uniqueness condition for functions. Note that (i) and (ii) imply that if $f(m) \neq n$, then $T \vdash \neg\varphi(\overline{m}, \overline{n})$.

Diagonalization in terms of a recursive function *diag* is independent of any intended interpretation of the result of diagonalization. It is based on the purely syntactic action of replacing a variable in a formula with a certain Gödel numeral, which, in turn, is defined through purely syntactic means. I do not question that diagonalization in this sense is definable in terms of a recursive function *diag*, nor that this recursive function can be expressed by a function *Diag* within L_A and captured by any theory T that is at least as strong as Q. Referring to *Diag* in the diagonalized function $\varphi(x)$ makes it possible to derive the *Diagonalization Lemma*:

Definition 4.2. If T is at least as strong as Q and $\varphi(x)$ is an open formula with one free variable x defined in a language at least as rich as L_A, then there exists a closed formula γ in the language of T such that $T \vdash \gamma \leftrightarrow \phi(\ulcorner\gamma\urcorner)$.

I take this lemma (and, consequently, its proof) for granted. This lemma concerns the provability of a formula, not the validity of its intended interpretations.

Many paradoxes, such as the Liar Paradox, Russell's Paradox and Richards' Paradox, are based on the diagonalization of *open sentences of either ordinary language or a semi-formal language*. One can diagonalize 'x is not true' by replacing 'x' in 'x is not true' with an expression (such as 'this proposition') that is intended to refer to the resulting expression. Similarly, 'x is a set that is not a member of itself' is diagonalized by replacing 'x' with an expression that is intended to refer to a set specified by 'x is a set that is not a member of itself'.

Here, I will define diagonalization for *expressions* of the form $f(x)$, such as $f_n(x)$, open formulas $\varphi(x)$ or open sentences such as 'x is not true'. To do so, I use the general term 'code' for any sort of sign or denotation of such an expression, be it an index, a Gödel numeral, a deictic expression or some other sort of sign. I do not presume that such a code indeed refers to the expression to which it is assigned (although it is intended to refer to it). For example, one can generate the Gödel numeral $\ulcorner\gamma\urcorner$ for a formula φ, and this numeral can occur in a formula ψ; whether $\ulcorner\gamma\urcorner$ refers to a number, to φ, or to both according to a reliable interpretation of ψ is not presumed in the process of generating $\ulcorner\gamma\urcorner$. Furthermore, I refer to the *result* of the process of replacing x in $f(x)$ with the code for $f(x)$ as the 'diagonalization'.

Definition 4.3. Let $\ulcorner\gamma\urcorner$ be the code for $f(x)$; then, $f(\ulcorner\gamma\urcorner)$ is the *diagonalization* of $f(x)$.

This definition is rather general; it does not imply that $\ulcorner\gamma\urcorner$ is the code for $f(\ulcorner\gamma\urcorner)$. According to the *Diagonalization Lemma*, for example, it is only by diagonalizing an open formula $\varphi(x)$ containing $Diag$ that a resulting formula $\varphi(\ulcorner\gamma\urcorner)$ is obtained that contains an expression $Diag(\ulcorner\gamma\urcorner)$ that is provably equivalent to the Gödel number of $\varphi(\ulcorner\gamma\urcorner)$.

The diagonal method is based on diagonalization. In contrast to diagonalization, the diagonal method possesses the two features (i) and (ii) identified in Definition 4.4 *by definition* (cf. [Jacquette (2004)], p. 70):

Definition 4.4. The *diagonal method* is a method of deriving proofs by contradiction by making use of diagonalization, hence implying (i) some sort of inversion, denial, negation or imposition of the complement of a diagonalization and (ii) some sort of self-application (or intended self-reference) in the diagonalization resulting from (i).

Definition 4.5. A *diagonal argument* is an argument based on the diagonal method.

Definition 4.6. A *diagonal case* is a diagonalization used via the diagonal method in a proof by contradiction.

Definition 4.4 is admittedly vague. Any definition of the diagonal method depends on the extent of the proofs one intends to cover. For the following discussion of Turing's and Church's proofs, a general and rather vague characterization of the diagonal method that is consistent with its standard use suffices. Neither a discussion of the diagonal method in general nor its general validity is at stake here; only the application of this method in proofs of the undecidability of FOL is of interest.

One may question Definition 4.4 not only because it is vague but also because its reference to 'proofs by contradiction' is either too narrow or inadequate. The former critique can be justified by, e.g., the distinction between good and bad diagonal arguments (cf. [Simmons (1993)], p. 29): Bad diagonal arguments do not reduce to absurdity the assumption of an entity that is not well defined, whereas good diagonal arguments either do not assume any entity that is not well defined or reduce to absurdity the assumption that some entity is well defined (which is a standard analysis to which I refer). The latter critique of the classification of diagonal arguments as proofs by contradiction may be justified by noting that proofs by contradiction are a specific class of logical proofs that presume well-defined entities (cf., e.g., RFM IV, §28, as well as the intuitionistic project of a negationless mathematics as formulated by [Griss (1946)] and [Nakano (2019)] for relating Wittgenstein's and Griss' analyses of proofs by contradiction in mathematics). I do not question these critiques. I simply follow the traditional understanding of the application of the diagonal method in undecidability proofs such as Turing's and Church's. As the analyses in sections 5 and 6 will show, it is standard to analyse these proofs as proofs by contradiction that reduce to absurdity the assumption that it is possible to define a decision function for FOL-provability by applying the diagonal method as characterized by the features specified in Definition 4.4. I discuss neither the relation of these proofs to other proofs using the diagonal method nor the question of whether it is adequate to formalize diagonal arguments within a language based on FOL.

It is, however, important to note that diagonalization is defined for *expressions*, not functions. The intention of this distinction is to explicate the application of the diagonal method to prove inexpressibility (or so-called 'undefinability') theorems, which is crucial to the following discussion. One cannot assume that intended interpretations of an expression of the form $f(x)$ express a function if those intended interpretations are intended to apply to diagonal cases. Even if the intended interpretation of expressions of the form $f(x)$ is a function, one cannot assume that this intended interpretation is also reliable in diagonal cases. The intention to interpret a function f_n as, for example, g (defining the *anti*-diagonal over an enumeration of functions $f_1, f_2, \ldots$) fails due to its diagonalization: it does not make sense to speak of a resulting diagonalized function since the result does not satisfy the requirement that a function must assign one and only one value to each of its arguments. Therefore, expressions of the form $f(x)$ do not necessarily express a function; only their reliable interpretations do. The function g is well defined only as a function that is not part of the enumeration over which it is defined. However, there is no interpretation that maps g onto a function f_n due to diagonalization. The diagonal method proves the impossibility of the intention to express a function in a certain way. Similarly, 'x is not true' and 'x is not a member of itself' are not well-defined propositional functions if their diagonalizations are intended to be interpreted as self-referential propositions.

For the discussion of the Church-Turing theorem, the extent to which one believes in the possibility of reliable interpretations of diagonalized expressions is irrelevant. No general claim about the reliability of intended interpretations involving 'self-reference' is assumed. It is sufficient to acknowledge that the diagonal method is used to establish inexpressibility theorems in terms of proofs by contradiction. Before turning to the discussion of Turing's and Church's proofs, I will illustrate the application of the diagonal method to prove inexpressibility theorems based on (i) the impossibility of expressing the halting function by means of Turing machines and (ii) the impossibility of expressing arithmetic truth within L_A.

The impossibility of solving the halting function is an example of an inexpressibility theorem concerning a language for expressing computable functions. It can be proven that the halting function $h(x)$ (which assigns a value of '1' to codes x for Turing machines that halt when started with x and a value of '2' to codes x for Turing machines that do not halt when started with x) *is not expressible by any Turing-computable function*. To show this, the assumption that some Turing machine H can be interpreted as computing the function $h(x)$ is reduced to absurdity by considering a diagonal case. One way to consider such a diagonal case is by considering a composition of a copy machine C, the assumed halting machine H and a dithering machine D that halts iff it is not started with '1' (cf. [Boolos et al. (2003)], pp. 39f.). The dithering machine causes the result of H to be '*inverted*' (feature (i) of Definition 4.4). The diagonal case arises from supposing that CHD is started with its own number (feature (ii) of Definition 4.4). H cannot be interpreted as intended in this diagonal case: if it assigns a value of '1', D makes it such that CHD does not halt, and if it does not assign a value of '1', D makes it such that CHD halts. What is proven is the impossibility of an interpretation: $h(x)$ is well defined only so long as it is not assumed to be computable since otherwise it would be capable of being diagonalized, which is incompatible with its definition as a function. We cannot specify rules for a consistent interpretation of $h(x)$ as a Turing-computable function.

The proof of Tarski's theorem is an example of an inexpressibility theorem concerning a language for expressing propositional functions. It proves that the supposed propositional function 'x is the Gödel number of an L_A formula that is true according to $\Im_A$' ($=$ True$_{L_A}(x)$) is *not expressible by an open formula in* L_A (for the following, cf., e.g., [Smith (2007)], section 21.5). The assumption that True$_{L_A}(x)$ is expressible in L_A by an open formula $T_A(x)$ yields a contradiction in the case of the diagonalization of its *negation* $\neg T_A(x)$ (cf. feature (i) in Definition 4.4). Let the Gödel numeral of $\neg T_A(x)$ be $\ulcorner L \urcorner$; then, by the *Diagonalization Lemma*, $L \leftrightarrow \neg T_A(\ulcorner L \urcorner)$ is provable in Q. If Q is sound (as assumed; see p. 8 above), then $L \leftrightarrow \neg T_A(\ulcorner L \urcorner)$ is true according to $\Im_A$. However, if $T_A(x)$ expresses

$\mathrm{True}_{L_A}(x)$, then $T_A(\ulcorner L \urcorner) \leftrightarrow L$ is also true according to L_A (cf. feature (ii) in Definition 4.4), which yields a contradiction. Given the *Diagonalization Lemma* and the soundness of Q, this proof reduces to absurdity the assumption that $\mathrm{True}_{L_A}(x)$ is expressible by $T_A(x)$ within L_A. The diagonal method shows that any intended interpretation of an open L_A formula $\varphi(x)$ expressing $\mathrm{True}_{L_a}(A)$ is unreliable. Hence, 'truth in L_A' is a well-defined propositional function only so long as it is not the intended interpretation of an open L_A formula.

These proofs show how the diagonal method is used to prove that certain functions are not expressible either within a language for expressing computable functions (= a language of computation based on Turing machines or recursive functions) or within a language for expressing propositional functions (= a logical language based on FOL). Undecidability proofs for FOL combine both types of languages and consider a characteristic function for FOL-provability. By the diagonal method, it is proven that the assumption that this characteristic function is expressible in both the language of computation *and* the language of logic yields a contradiction. *Prima facie*, this contradiction can be used only to reduce the *conjunction* to absurdity. However, it is instead used to reduce the first conjunct, the assumption of the decidability of FOL, to absurdity. The reason for reducing the first conjunct to absurdity is the proof of a lemma (Turing) or a theorem (Church) that, roughly speaking, states that *any* computable function is expressible within a language based on FOL. This *general* claim justifies the *specification* to the assumed decision function $Prov(x)$ for FOL-provability and its expressibility in the form of a propositional function $P(x)$: if FOL were decidable, $Prov(x)$ would be expressible as a propositional function $P(x)$ even in the case of the diagonalization of the propositional function $\neg P(x)$. The truth of this claim must be assumed to reduce the first (instead of the second) conjunction of the above conjunct to absurdity. In the following sections, we will take a closer look at the proofs of the corresponding lemma (in Turing's proof) and theorem (in Church's proof) and ask whether these proofs indeed justify this specification.

5. Turing's Proof

Two sorts of undecidability proofs can be distinguished: *primary* undecidability proofs, which are not based on any other undecidability proof, and *secondary* undecidability proofs, which are based on other undecidability proofs. The undecidability of the halting problem is a primary undecidability proof: it directly refers to a diagonal case to show that it is impossible to interpret any Turing-computable string function as the halting function. Turing's undecidability proof of FOL, in contrast, is a secondary undecidability proof. His undecidability proof rests on the relation of some undecidable function concerning Turing machines (i.e., the halting function) to a logical function of interest (i.e., a decision function for FOL-provability).

To relate properties of Turing machines to properties of formulas, one must interpret those formulas. To describe the instructions and configurations of Turing machines M, an 'intended interpretation' for the non-logical vocabulary in logical formulas is established. For example, [Turing (1936)], pp. 259f., presents the following intended interpretations for the atomic open formulas ('function variables', in his terminology) that he uses:

$R_{S_l}(x, y)$ is to be interpreted as 'in the complete configuration x (of M), the symbol on the square y is S'.

$I(x, y)$ is to be interpreted as 'in the complete configuration x, the square y

is scanned'.

$K_{q_m}(x)$ is to be interpreted as 'in the complete configuration x, the m-configuration is q_m'.

$F(x, y)$ is to be interpreted as 'y is the immediate successor of x'.

By 'complete configuration', Turing refers to what now is called the 'state' of a Turing machine M, while 'm-configuration' is Turing's term for 'configuration'. The details of Turing's intended interpretation, however, are not relevant to the following critique. What is important is that Turing's proof, as well as modern versions of it (cf., e.g., [Boolos et al. (2003)], p. 127), is based on intended (or 'standard') interpretations using ordinary language with an intended meaning. These interpretations are intended to map properties of Turing machines to properties of logical formulas.

As Turing originally defined them, Turing machines start with an empty tape, and 'bad', circular machines are those that become stuck, while good, 'non-circular machines' never halt. In contrast, Turing machines as they have been defined since Post and Davis start with a non-empty tape, and the 'good' ones are those that halt. These differences also play no essential role in the following discussion. Thus, I will neglect to distinguish between these different concepts of Turing machines.

All versions of Turing's proof relate decision problems regarding the behaviour of Turing machines to the decision problem regarding properties of FOL formulas. It does not matter which logical properties (such as FOL-provability, satisfiability or logical validity) are considered since they are all interdefinable. This is why I discuss simply 'the decidability of FOL', meaning that some logical property of FOL is decidable. Likewise, 'the decision problem' is short for 'the problem of whether some logical property is decidable'.

According to decision problems regarding the behaviour of Turing machines, any function concerning some non-trivial property of Turing machines can be chosen since they are all undecidable according to Rice's Theorem. Turing, for example, refers to the problem of deciding whether a Turing machine ever prints the symbol '0'. Modern versions of Turing's proof intend to map the halting function onto an assumed decision function for FOL formulas. Given such a mapping, the undecidability of FOL follows from the undecidability of the corresponding decision problem for Turing machines.

To relate an undecidable problem concerning the behaviour of Turing machines to the decision problem, a *translation procedure* is defined to translate the codes and configurations of Turing machines into logical formulas (including some background theory). Such a procedure is a purely syntactic exercise and is itself a computable string function. In the following, I refer simply to 'translations of Turing machines M', including the translations concerning their configurations (e.g., the statement that M reaches a halting state at some time), and following Turing, I abbreviate the corresponding formulas by $Un(M)$. Given M and its initial configuration, $Un(M)$ can be computed. The question is whether a one-to-one correlation can be established between the logical properties of $Un(M)$ and the properties of M due to the specified translation procedure.

Proving this one-to-one correlation is the crucial element of (all versions of) Turing's proof. This is done by proving a *lemma* that purports to prove that $Un(M)$ is provable (alternatively: true in all interpretations) iff M has the property in question, e.g., ever printing '0' or halting, (= *Turing's Lemma*). The direction from right to left is proven by relating each step of the behaviour of a Turing machine to an inferential, truth-preserving step in a proof of $Un(M)$ (= *Lemma 1*). I will relate my critique only to the direction from left to right (= *Lemma 2*). The proof of this direction is very short and 'easy'

([Boolos et al. (2003)], p. 130). It is based on an intended interpretation of logical formulas describing the configurations of the translated Turing machine. I quote the complete proof of [Turing (1936)], p. 262:

> LEMMA 2. If $Un(M)$ is provable, then S_1 [i.e., the symbol '0', T.L.] appears on the tape in some complete configuration of 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 $Un(M)$, we obtain a true proposition with the meaning 'S_1 appears somewhere on the tape in some complete configuration of M'.

Function variables is Turing's term for open formulas, and *propositional functions* is his term for instances (or intended interpretations) of open FOL formulas. The fact that he uses the term 'propositional functions' shows that he does not question that any instance or intended interpretation is, in fact, reliable. If one could assume that any instance or intended interpretation of an open FOL formula is indeed a (well-defined) propositional *function*, then he would be quite correct that any such instance or intended interpretation of a provable formula would be true. However, this cannot be presumed, as shown by the discussion in section 3 and the discussion of Tarski's theorem. As his second sentences makes evident, Turing, in fact, refers to intended interpretations of predicates used in $Un(M)$ (quoted above on p. 11), and he stipulates in the first sentence of his proof that from the provability of a formula, the truth of its intended interpretation follows. Hence, his 'proof' is a nice example of the fallacy of intended interpretations (cf. Definition 3.13).

Turing's intended interpretation of $Un(M)$ seems to be unproblematic as long as applied to normal (non-diagonal) cases. One might argue that, unlike in the case of paradoxes, the question of reliability does not arise because Turing machines and their description are well determined. However, this view poses a risk of succumbing to the fallacy of intended interpretation. The question is whether the intended interpretations of $Un(M)$ are also reliable in the diagonal cases that are relevant to the primary undecidability proofs on which Turing's proof is based. It was argued in section 3 that the unreliability of intended interpretations may come as a surprise, contradicting semantic intuitions, and it was argued in section 4 that the diagonal method is used to demonstrate that intended interpretations do not, in fact, provide the interpretations $\Im$ that they are intended to provide. No effective translation procedures for clear-cut concepts such as Turing machines nor the intuitive evidence of the correctness of the translation in normal cases suffices to demonstrate the correctness of the translation procedure in diagonal cases. Unreliable intended interpretations may well result from completely determined descriptions. Consider, for example, a book consisting of well-formed sentences that are all interpreted as propositions capable of being false or true. The sentences in such a book can be enumerated. Let the book contain the sentence 'Sentence number 27 is not true', and suppose that this sentence itself is the 27th sentence of the book. The sentences in the book, their enumeration, sentences of the form 'Sentence number n is not true' and the intended interpretation of provable formulas of the form $P \leftrightarrow P$ by means of sentences of the form 'Sentence number n is not true' are all well determined, yet they still produce the Liar Paradox. More importantly, one cannot conclude from the unproblematic referential relations of expressions established for typical contexts that they still refer in the same way in atypical contexts such as intensional, metalanguage or tautological contexts. The question is whether diagonal contexts with intended self-referential interpretations are, in fact, extensional contexts or whether they also should be classified as atypical, non-extensional contexts that do not permit inference from the provability of formulas to the truth of their intended interpretations. Neither Turing nor any modern version of his proof addresses the question of the reliability of the intended interpretations in the relevant diagonal cases. For example, [Boolos et al. (2003)], p. 130, presume without further argument that 'truth in all interpretations' ($\Im$) also applies to their 'standard interpretation' (I), without considering the application of their standard interpretation to diagonal cases.

However, if one considers FOL to be decidable, then diagonalization inevitably comes into play as soon as a decision on logical properties is correlated with a decision on the properties of Turing machines. According to the *Turing thesis*, the decidability of FOL implies the existence of a Turing machine that computes a function that expresses a decision function for FOL formulas. Let this Turing machine M be denoted by 'FOL' (in italics). A primary undecidability proof, e.g., a proof of the undecidability of the halting problem, directly applies to the hypothetical machine FOL if *Turing's Lemma*, applied to the halting problem, is assumed.

Let T be a translation machine that generates $Un(M)$ (or its code) from two sequences of strokes m and n: the first is a code for the input of the machine M, and the second codes the instructions for M. According to *Turing's Lemma*, the composition $TFOL$ of T and FOL would behave as a halting machine H since the decision on the provability of $Un(M)$ coincides with the decision on the halting of M with code n when started with m. Applying the diagonal method, as in the case of the proof of the unsolvability of the halting problem, we consider the composition $CTFOLD$ of a copy machine C, a translation machine T, the decision machine FOL and the dithering machine D. This yields a contradiction in the diagonal case: the decision of FOL cannot be interpreted as a decision about the property of the halting of $CTFOLD$ when started with its own number.

This contradiction can be interpreted in two ways. Under the assumption of *Turing's Lemma*, one can reduce to absurdity the assumption that FOL exists (since C, T, and D obviously exist). However, one may likewise reduce the *lemma* to absurdity and argue that the diagonal case of $CTFOLD$ when started with its own number shows that the intended interpretation is not reliable in this special case. In the first interpretation, one questions the expressibility of a decision function for FOL-provability by means of an assumed Turing machine FOL. In the second interpretation, one questions the expressibility of the halting function for Turing machines M by a propositional function $Un(M)$ in the language of FOL. In this interpretation, one would 'withdraw the intended interpretation' of the decision on $Un(M)$ in a diagonal case such as $Un(CTFOLD)$ when started with its own number (cf. RFM, p. 51e, §10). This is a straightforward conclusion since the proof of the *lemma* is based on a fallacy. Thus, what is absurd according to this second interpretation is not the assumption that FOL is decidable but instead the assumption that a decision on $Un(M)$ could be used to decide on the halting function (or some other uncomputable function).

Therefore, instead of rejecting the decidability of FOL by espousing the validity of the mapping of functions concerning the behaviour of Turing machines onto an assumed decision function for FOL by means of intended interpretations, one may reject the possibility of expressing the behaviour of Turing machines by FOL expressions when diagonal cases are involved. This rejection simply applies the proof of the impossibility of expressing the halting function by a Turing-computable function to an assumed decision function for FOL-provability: it is not the computability of the functions onto which the halting function is intended to be mapped that is called into question but rather the claim that such a mapping is even possible in diagonal cases.

An undecidability proof of FOL must be based on a proof of a *lemma* that is independent of FOL's decidability. Versions of Turing's proof do not satisfy this condition since they are based on intended interpretations of $Un(M)$ and these intended interpretations cannot obey the semantics of FOL in the case that FOL is decidable. Therefore, Turing's proof shows only that either FOL is undecidable or properties of Turing machines cannot be expressed by properties of $Un(M)$. Thus, the proof is underdetermined.

The translation procedure for generating formulas $Un(M)$ applies to Turing machines *in general*. Turing's proof assumes that this translation procedure is *correct in any case*, meaning that the resulting intended interpretation of $Un(M)$ does, in fact, express the behaviour of the described Turing machine in any arbitrary case. This *general claim* justifies the argument in the proof of *Lemma 2*. No special reasoning is provided that this general claim, in fact, specifically applies to diagonal cases involving a machine FOL. However, diagonal cases are designed to reduce correlations between functions to absurdity. Therefore, special reasoning is needed to argue that a *lemma* that correlates a function concerning the behaviour of Turing machines to a function deciding on a logical property of $Un(M)$ holds not merely in normal cases but also in diagonal cases involving FOL.

In such cases, a hypothetical decision function for FOL is assumed to be expressed within FOL. Rejecting the intended interpretation of $Un(M)$ in diagonal cases implies rejecting that FOL-provability is expressible within the language of FOL. This can be seen from the diagonal case of $CTFOLD$ started with its own number. In this case, FOL, which is hypothesized to compute the characteristic function for FOL-provability, is started with $Un(CTFOLD)$, which implies the translation of FOL. This diagonal case makes it impossible to simultaneously trust both what the supposed machine FOL, in fact, decides and what it decides according to the intended interpretation of $Un(CTFOLD)$. If FOL decides that $Un(CTFOLD)$ is provable, it, in fact, assigns the value '1' to $Un(CTFOLD)$; however, since D halts only if not started with '1', it must assign the value '2' (or its code in stroke notation) to $Un(CTFOLD)$ according to the intended interpretation of $Un(CTFOLD)$, which states that $CTFOLD$ halts when started with its own number. Thus, reducing the correctness of the intended interpretation to absurdity (instead of reducing the assumption of the existence of FOL to absurdity) implies rejecting the claim that it is possible to express a decision function for FOL-provability within FOL. A decision function for FOL-provability cannot be expressed within FOL since in the diagonal case, the intended interpretation of such a formula implies a statement on the provability of the very formula that is reduced to absurdity by the diagonal method since it necessarily contradicts the result of the supposed decision machine in the diagonal case.

Turing's proof is only indirectly concerned with the expression of FOL-provability within FOL since it is a secondary undecidability proof that is related to some undecidable property of Turing machines. By contrast, the following discussion of Church's proof will be directly concerned with the question of expressing provability within a language based on FOL.

6. Church's Proof

While Turing's proof is based on the concept of Turing machines, Church's proof is based on recursive functions. Turing's proof rests on the undecidability of a decision problem for Turing machines that is proven to be unsolvable independent of the technique for expressing properties of Turing machines by logical formulas. Church's proof, in contrast, is not similarly based on the undecidability of a decision problem for recursive functions that can be proven independent of the technique for expressing recursive functions by logical formulas.

Nevertheless, the key feature of both proofs is that the undecidability proof of FOL rests on a technique for expressing computable functions within a language based on FOL. While Turing's proof rests on a translation procedure for expressing the behaviour of Turing machines through logical formulas, Church's proof rests on expressing recursive functions by L_A formulas (which are reducible to FOL formulas). I will argue that the proof of the 'theorem' that every recursive function is expressible within L_A suffers from a similar deficiency as the proof of *Turing's Lemma.*

For simplicity, I refer to modern versions of Church's proof, in particular, to [Smith (2013)].

Church's undecidability proof of FOL rests on a proof that the property of provability in Robinson Arithmetic (Q) is undecidable (= undecidability of Q). Church argues that if FOL is decidable, then Q is decidable. The proof of this implication is based on (i) the translation of L_A formulas into FOL formulas while preserving satisfiability and (ii) the deduction theorem applied to the axioms of Q. The translation rules that eliminate the function symbols (including '0') and identity are defined by [Boolos et al. (2003)], chapter 19.4. These authors prove the correctness of the translation procedure by proving their propositions 19.12 and 19.13 (cf. [Boolos et al. (2003)], pp. 256 and 257). The proofs of these propositions are plain and unproblematic.

One might classify Church's undecidability proof as a secondary proof since it is based on the undecidability of provability in Q. However, this analogy to the classification of Turing's proof as a secondary proof is misleading with regard to the critique of Turing's proof in the previous section. While the reduction of the decision problem to a decision problem for Turing machines is problematic, the reduction of the undecidability of FOL to the undecidability of Q is, in contrast, rather trivial. It does not concern the relation of a language based on FOL to a language of computable functions (in Church's case, recursive functions) but rather concerns the relation of a language based on FOL (in Church's case, L_A) to FOL. If FOL is decidable, then indeed, Q is also decidable. Instead it is the expressibility of recursive functions in L_A, on which the undecidability proof of Q is based, that involves a problem similar to that of Turing's proof. The undecidability proof of Q directly concerns the possibility of expressing an assumed recursive function for Q-provability in L_A and capturing it in Q. Since L_A is reducible to FOL and since provability in Q is reducible to provability in FOL, this proof therefore concerns the possibility of expressing FOL-provability within FOL.

Similar to Turing's proof, which is based on a procedure for translating Turing machines and their configurations into FOL formulas, Church's proof is based on a procedure for translating (canonically) defined recursive functions into L_A formulas. The resulting open L_A formulas are of the rather simple Σ_1 type (i.e., a prenex normal form with only unbounded existential quantifiers). Recursive properties can be defined by means of their characteristic functions. Thus, given a recursive property $f(x)$, one can generate a Σ_1 expression that *expresses* and *captures* the recursive property, given that the translation procedure indeed achieves the purpose for which it is designed.

I call $\neg P(\ulcorner\gamma\urcorner)$ the diagonal case of $P(x)$, where $P(x)$ denotes the supposed Σ_1-type propositional function that expresses and captures a presumed recursive property $Prov(x)$

defined by a characteristic recursive function that is isomorphic to a decision function for Q-provability (implying a decision function for FOL-provability).

According to the *Church thesis*, the assumption that FOL (and, consequently, Q) is decidable implies that a decision function for FOL (and, consequently Q) can be mapped onto a recursive function. As in Turing's proof, the assumption that FOL is decidable is unproblematic as long as it is not related to any diagonal case. In Church's proof, diagonalization comes into the play only when (i) expressing recursive functions within the language of L_A and (ii) considering diagonalization within L_A.

Given the *Diagonalization Lemma* and the consistency of Q (cf. Definition 4.2 and p. 8), the diagonal case immediately shows that $Prov(x)$ cannot exist *if* $Prov(x)$ *is to be expressed and, moreover, captured by* $P(x)$. This conclusion follows trivially from the following three assumptions (cf. [Smith (2013)], p. 183):

Assumption 1: $Q \vdash \gamma \leftrightarrow \neg P(\ulcorner\gamma\urcorner)$.
Assumption 2: If $Q \vdash \gamma$, i.e., $Prov(\ulcorner\gamma\urcorner)$, then $Q \vdash P(\ulcorner\gamma\urcorner)$.
Assumption 3: If $Q \nvdash \gamma$, i.e., not-$Prov(\ulcorner\gamma\urcorner)$, then $Q \vdash \neg P(\ulcorner\gamma\urcorner)$.

Assumption 1 applies the *Diagonalization Lemma* to $\neg P(x)$, Assumptions 2 and 3 follow from the assumption that $P(x)$ captures the presumed recursive property $Prov(x)$. Assumptions 1 to 3 together imply a contradiction. Thus, if Q is consistent and the *Diagonalization Lemma* holds, then $P(x)$ cannot capture a presumed recursive property $Prov(x)$. A similar argument can be presented on the basis that Q is sound and $P(x)$ expresses $Prov(x)$.

The derived contradiction either can reduce to absurdity the assumption that FOL-provability and, consequently, Q-provability is recursively definable or can reduce to absurdity the claim that the presumed recursive property $Prov(x)$ is expressed and captured by the open formula $P(x)$ resulting from the translation procedure. In the latter case, Church's proof is interpreted not as an undecidability proof but as analogous to the proofs of undefinability theorems, such as the proof of Tarski's theorem. Church, however, advocates for the first option on the basis of the general theorems that (i) any recursive property is expressible within L_A (the *expressing theorem*) and (ii) any recursive property can be captured within Q (the *capturing theorem*). The latter theorem is based on the former. I focus on the former since it is this theorem and its proof that can be questioned in a manner similar to the critique of *Turing's Lemma*.

The procedure for translating the canonical definitions of recursive functions into L_A formulas is itself computable. The crucial question is whether this translation procedure is *correct* in the sense that it indeed results in formulas that *express* the translated recursive functions and properties. The proof of the *expressing theorem* does not prove this rigorously. Instead, it consists of specifying a translation procedure and simply appealing to semantic evidence to argue for its correctness. Smith's proof nicely illustrates this.

The proof of the *expressing theorem* for recursive functions proves the theorem for the initial functions, for the composition of recursive functions, for primitive recursion and for minimization (cf. [Smith (2013)], chapter 15 and p. 297). The question of the criterion for the correctness of the translation procedure already arises in the simplest case of the translations of the initial functions. Smith's proof of the *expressing theorem* starts with proving the theorem for these functions. I quote the complete proof for the successor and zero functions from [Smith (2013)], p. 110:

Proof for (1) There are three easy cases to consider:

i. The successor function $Sx = y$ is expressed by the open wff $\mathtt{Sx} = \mathtt{y}$.

ii. The zero function $Z(x) = 0$ is expressed by the wff $\mathtt{Z(x, y)} =_{\mathtt{def}} (\mathtt{x} = \mathtt{x} \wedge \mathtt{y} = \mathtt{0})$.

In contrast to the tricky case of expressing recursion, the translation procedure is 'trivial' ([Smith (2007)], p. 110) in the case of the initial functions. In this case, its correctness seems to be plain and unquestionable. However, it is interesting to note that the proof, in fact, consists of prescribing *how* the functions are to be translated without explicitly stating *why* the translation is *correct*. The answer to this question must relate the recursively defined functions to the *arithmetic* interpretation $\Im_A$ of the propositional functions in L_A. $\Im_A$ assigns truth values to propositional functions for given instances of their arguments. The translation of a recursively defined function $f(x) = y$ into a propositional function $\phi(x, y)$ is correct iff for all m and n, $\phi(\overline{m}, \overline{n})$ is true according to $\Im_A$ iff $f(m) = n$ (cf. [Smith (2013)], p. 43). How can one know that this is the case?

To answer this question, one must answer the following questions: (i) How can one gain knowledge of the truth values of $\phi(x, y)$ for specific values m and n? (ii) How can one know that the truth values of $\Im_A(\phi(\overline{m}, \overline{n}))$ correspond to the computation of $f(m) = n$? (iii) How can one know the answers for *all* of the infinite number of possible values?

With regard to (i), one cannot refer to provability within Q to justify the truth value of $\Im_A(\phi(\overline{m}, \overline{n}))$ since Q is justified by its soundness and the *capturing theorem* is based on the *expressing theorem*: the syntax is measured by the presumed semantics, not the other way round. So, one must simply accept the truth values of L_A formulas according to $\Im_A$ as primitives (as emphasized by [Smith (2013)], section 5.3). Furthermore, we do not know the truth values of the arithmetical interpretation $\Im_A$ of L_A formulas in non-trivial cases. Thus, we cannot justify or control the translation procedure in such cases by independently comparing truth values.

The same argument applies regarding the second question. We might think to run a computation and compare the result of what the computation *does* with the truth value of what the corresponding L_A proposition *says* according to $\Im_A$. However, let us ignore any specific problems with this method since it cannot be effective in any case because of (iii): it is obvious that we cannot justify any general answer on the basis of specific values.

What we, in fact, do in justifying the correctness of the translation procedure is to relate *paraphrases* or *intended interpretations* of the relation between x and y in the recursively defined function to *paraphrases* or *intended interpretations* of the L_A expression $\phi(x, y)$. In doing so, one relates intended interpretations of the recursively defined functions in terms of *number-theoretic (or arithmetic) extensional functions* to the extensional understanding of the arithmetic interpretation $\Im_A$ of the corresponding propositional L_A functions.[5] In the 'Proof for (1)' quoted above, for example, Smith interprets $Sx = y$ as the successor function and $Z(x) = 0$ as the zero function: this is a number-theoretic interpretation since the values of the variables are numbers and not, e.g., symbols, and it is extensional since it does not matter how the values of the functions are assigned to their arguments. Such an interpretation is, indeed, best suited to be compared with the interpretation $\Im_A$ of the corresponding propositional functions. In the case of $Sx = y$ and $\mathtt{Sx} = \mathtt{y}$, one must judge that 'y is assigned to the successor to x' is true iff 'y is equal to the successor of x' is true. In the case of the zero function, one must compare '0 is assigned to x' and 'x is identical to itself and y equals 0'. The translation is correct iff the two paraphrases are coextensive. Note that the paraphrases depend on the syntax of the paraphrased expressions and differ

[5]Cf. the 'remarks about extensionality' by [Smith (2007)], pp. 29f., pp. 34f. and 35ff., and [Smith (2013)], chapter 14.4.

Recursive Def.	L_A **Formalization**
$0! = 1$ $(Sx)! = x! \times Sx$	$\exists_c \exists_d (\exists_u (c = u \times S(d \times S(0)) + S(0) \wedge$ $\exists_m\, m + u = c \wedge \exists_m\, m + S(0) = d \times S(0)) \wedge$ $\exists_u (c = u \times S(d \times S(x)) + y \wedge \exists_m\, m + u = c \wedge$ $\exists_m\, m + y = d \times S(x)) \wedge$ $\forall_z (\exists_m\, m + z = x \Rightarrow (z \neq x \Rightarrow \exists_v \exists_w (w = v \times S(z) \wedge$ $\exists_u (c = u \times S(d \times S(z)) + v \wedge \exists_m\, m + v = d \times S(z) \wedge$ $\exists_m\, m + u = c) \wedge \exists_u (c = u \times S(d \times S(S(z))) + w \wedge$ $\exists_m\, m + w = d \times S(S(z)) \wedge \exists_m\, m + u = c)))))$

Table 1: Translation of the Factorial Function into a Propositional L_A Function

to a greater extent as the expressions to be paraphrased become more complex. Consider, for example, the still rather 'easy' case of the translation of the factorial function into a Σ_1 formula, as presented in Table 1. The paraphrases of the expressions already differ significantly since the syntax of each paraphrased expression is completely different. The similarity of the paraphrases in the case of the successor function is an exception in this respect.

The assertion that the paraphrases of the initial functions are coextensive is, of course, rather trivial and beyond doubt when one takes the underlying semantics for granted. However, the implications of the method of evaluating the correctness of the translation are important: the 'criteria' for correctness are judgements concerning the coextensionality of intended paraphrases or interpretations. This method is not only the implicit method used in the proof of the initial functions; the same method is applied in the cases of composition, recursion and minimization, both by Smith and in alternative proofs of the *expressing theorem.* These proofs basically consist of defining *how* the recursive functions are translated; if the correctness of the translation procedure is addressed at all, then nothing more is done than to appeal to evidence based on one's semantic intuitions. Smith, for example, ends his translation of the factorial function into the formula given in Table 1 with the following emphatic statement: 'For this *evidently* expresses the factorial function' (my emphasis). Likewise, he ultimately 'proves' that the described procedure works in any arbitrary case by presenting a general translation schema φ and asserting that 'it is then *evident* that φ will serve to express the p.r. defined function f' ([Smith (2013)], p. 118, my emphasis). In the case of minimization, he again simply defines *how* minimization is translated into Σ_1 formulas and simply avows that this '*evidently* expresses' ([Smith (2007)], p. 277, my emphasis)[6] the function defined by minimization. Smith's proof is in no way less explicit than others, nor does it differ in method from other proofs of the *expressing theorem.* On the contrary, his proof is admirably clear and understandable.

My point of critique concerns not the complexity of the translation procedure but its application to hypothetical diagonal cases. I have no doubt that recursive functions such as the factorial function can be expressed and captured by the translation procedure. However, the question is whether it can be justified that such a procedure is *correct in any arbitrary case*, including the hypothetical case of a recursive definition of Q- or FOL-provability. Such a case inevitably results in a contradiction due to the diagonal context. Any generalization

[6]Cf. [Smith (2013)], p. 297: 'Intuitively, `F` expresses f; i.e. $f(m) = n$ iff $F(\overline{m}, \overline{n})$ is true (think about it!).'

of the evidence gained from rather trivial, non-diagonal cases to arbitrary cases presumes a similarity between those cases. However, diagonal cases giving rise to contradictions are not similar to non-diagonal cases, which alone are addressed when appealing to semantic intuitions. This is why any sort of purported inductive proof concerns only the generality of the specified translation procedure and not its correctness.

According to the defined translation procedure, a presumed recursive property $Prov(x)$ can be related to a propositional function $P(x)$. The proof of the *expressing theorem* neither proves nor provides any evidence that $P(x)$ would indeed express $Prov(x)$ given that $Prov(x)$ exists. In this hypothetical case, the method of judging upon the equivalence of intended interpretations necessarily fails. By hypothesis, any intended interpretation of $Prov(\ulcorner\gamma\urcorner)$ must (i) be coextensive with 'the formula with number γ (i.e., $\neg P(\ulcorner\gamma\urcorner)$) is provable' and (ii) be expressed by $P(\ulcorner\gamma\urcorner)$. The interpretation $\Im_A$ of $P(\ulcorner\gamma\urcorner)$, however, cannot be coextensive with 'the formula with number γ (i.e., $\neg P(\ulcorner\gamma\urcorner)$) is provable' if Q is sound.

Of course, one can argue that this contradiction reduces the hypothetical assumption that Q-provability is recursively definable by $Prov(x)$ to absurdity. However, this argument is not conclusive because for it to be conclusive, it would be necessary to show that $Prov(x)$ would be expressible by $P(x)$ if $Prov(x)$ were to exist. In short, undecidability is shown only by independently proving expressibility; otherwise, one may likewise reduce to absurdity the assumption that L_A is capable of expressing Q-provability or, more fundamentally, that FOL is capable of expressing FOL-provability.

Proofs of the *expressing theorem* do not explicitly address the question of how to prove the theorem for hypothetical diagonal cases. If this question is addressed, it becomes clear that any undecidability proof begs the question. Undecidability proofs, in fact, demonstrate that a diagonal case rules out the possibility of expressing and capturing the property in question. How can one justify that the *expressing theorem* holds for *all* recursively defined properties without presuming that any property that is demonstrably not expressible in L_A is not recursive? Circularity can be avoided only if it is conceded that the undecidability proof is underdetermined; under the assumption that Q is consistent, the diagonal case shows either that Q- and, consequently, FOL-provability cannot be defined recursively or that FOL-provability is not expressible within L_A and, consequently, FOL due to the diagonal case. Semantic evidence cannot settle the matter since our semantic intuitions are derived from normal cases that we can oversee and control.

Diagonal cases as considered in paradoxes such as the Liar Paradox, Russell's Paradox or Richards' Paradox show that our semantic intuitions are not reliable. Undecidability proofs resting on the technique for expressing diagonal cases in the language of logic do not prove that it is indeed possible in diagonal cases to express what one intends to say by means of the language of logic. The use of the diagonal method to prove theorems such as the unsolvability of the halting problem or Tarski's theorem and paradoxes such as the Liar Paradox or Richards' Paradox cast doubt on the use of the diagonal method to establish the undecidability of FOL. Interpreting Turing's and Church's proofs as undefinability proofs for FOL-provability within FOL is a straightforward alternative.

7. Conclusion

In addition to Turing's and Church's canonical proofs, many other versions of undecidability proofs of FOL (or fragments thereof) exist at present (cf. [Boerger et al.(2001)]). However, all of these undecidability proofs rest on the expression of undecidable problems within FOL and ultimately invoke diagonal cases. Any proof that it is possible to express undecidable problems within FOL is based on a claim of the general validity of translation procedures that encounter semantic problems in diagonal cases. Rigorous proofs concerning the possibility of defining a decision procedure for FOL should not be affected by such semantic problems. Instead of making a hopeless attempt to prove strong general claims concerning the expressibility of supposed functions within a language based on FOL, it would be more fruitful to attempt to solve the decision problem. Such an attempt concerns the possibility of algorithmizing the application of rules of a correct and complete calculus such that not only theoremhood but also non-theoremhood can be decided within a finite number of steps. This effort to seek a positive solution to the decision problem is not affected by problems arising from attempting to express the limits of logic within the language of logic. Whatever may result from this work, it can only increase our power to decide first-order logic formulas.

Acknowledgement

I am grateful to Anderson Nakano for the critical discussion of this paper and its previous versions.

References

[Boolos et al. (2003)] Boolos, G.S., Burgess, J.P., Jeffrey, R.C.: *Computability and Logic*, second edition, Cambridge: Cambridge University Press.

[Boerger et al.(2001)] Börger, E., Grädel, E. & Gurevich, Y.: *The Classical Decision Problem*, Springer.

[Boker & Dershowitz (2008)] Boker, U. and Dershowitz, N.: ‘The Church-Turing Thesis over Arbitrary Domains’, in: Avron, Dershowitz, Rabinovich (eds.), *Pillars of Computer Science*, Springer, Berlin, 199-229.

[Brun(2003)] Brun, G. (2003). *Die richtige Formel. Philosophische Probleme der logischen Formalisierung.* Frankfurt: Ontos.

[Dreben (1979)] Dreben, B., Goldfarb, W.D.: *The Decision Problem. Solvable Classes of Quantificational Formulas*, Addison-Wesley, London.

[Gödel (1986)] Gödel, K. 1986: ‘On formally undecidable propositions of *Principia Mathematica* and related systems I’, *Collected Works Volume I Publications 1929-1936*, Oxford UP, Oxford, 144-195.

[Griss (1946)] Griss, G.F.C.: ‘Negationless intuitionistic mathematics I’, *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings of the section of sciences 49*, 1127-1133.

[Jacquette (2004)] Jaquette, D.: ‘Diagonalization in Logic and Mathematics’, in Gabbay & Guenthner (eds): *Handbook of Philosophical Logic 11*, 2nd edition, Springer, Dordrecth, 55-148.

[Kreisel (1967)] Kreisel, G.: ‘Informal rigour and completeness proofs’, in Lakatos (ed.): *Problems in the Philosophy of Mathematics*, Amsterdam, North-Holland, 138-171.

[Nakano (2019)] Nakano, A.: ‘Anti-realism and anti-revisionism in Wittgenstein’s philosophy of mathematics: a contribution to an exegetical issue’, unpublished.

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

[Rescorla (2007)] Rescorla, M.: ’Church’s thesis and the conceptual analysis of computability’, *Notre Dame Journal of Formal Logic* 48(2), 253-280.

[Shapiro (1982)] Shapiro, S.: ‘Acceptable notation’, *Notre Dame Journal of Formal Logic*, 23(1), 14-20.

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

[Smith (2007)] Smith, P.: *An Introduction to Gödel’s Theorems*, first edition, Cambridge University Press, Cambridge.

[Smith (2013)] Smith, P.: *An Introduction to Gödel’s Theorems*, second edition, Cambridge University Press, Cambridge.

[Turing (1936)] Turing, A.: ‘On Computable Numbers, with an Application to the Entscheidungsproblem’, in *Proceedings of the London Mathematical Society* 2 (42), 230-265.

[Weihrauch (2000)] Weihrauch, K.: ‘Computable Analysis’, Springer, Berlin, Heidelberg.

[Wittgenstein (1967)] Wittgenstein, L.: *Remarks on the Foundations of Mathematics* (RFM), Massachusetts: M.I.T. Press.