MATHEMATICAL LOGIC FOR COMPUTER SCIENCE (A.A. 20/21)

HANDOUT N. 16

Abstract. Gödel's First Incompleteness Theorem. Tarski's Theorem on the undefinability of arithmetical truth. Gödel's Second Incompleteness Theorem. Löb's Theorem. Unprovability of Consistency.

1. Syntactic properties are decidable

When dealing with decision problems about membership in sets of sentences we are supposing that computer programs can deal with different data types, such as strings, arrays, etc., which can be used to properly encode syntactic objects like sentences. This is natural from a contemporary point of view. Moreover, if we wish to only have natural numbers as data type, we can invoke the possibility of coding numerically syntactic objects such as terms, formulas, sentences, sequences of sentences, etc.

Gödel used this idea in the "pre-computer Era" in order to reason about algorithmic decision problems about syntactical objects, in particular about formal proofs. The process of coding syntactical objects as numbers was called the "arithmetization of syntax".

Nowadays, we can assume to have fixed a suitable coding function that codes objects such as symbols, strings of symbols and sequences of strings of symbols as natural numbers. We will denote this function by *code*. When we deal with decision problems about sentences or proofs we are dealing in fact with decision problems about their codes under this fixed coding. In this context, when we refer to a set S of sentences we refer to the set of codes of the sentences in S.

The fundamental observation in what follows is that many syntactic properties are algorithmically decidable. For example: given n, decide whether n is the code of a variable of the language is a machine-decidable task. Analogously: given n, decide whether n is the code of an axiom of MA; given n, decide whether n is the code of a proof in MA, etc. are all machine-decidable tasks. For the latter concept, it is important to stress that deciding whether a string is a formal proof from axioms in MA is computable because the axioms of MA are computable (they can be recognized by a machine). The other conditions in a definition of a formal proof are obviously machine-decidable. This observation generalizes: if a theory T is decidable then the notion "this string σ is a formal proof with premisses in T" is also decidable.

Nowadays this is easy to believe and unproblematic. At Gödel's time this had to be proved carefully by exhibiting computable definitions (programs) for each of these decision problems. A theory T such that the property $A(n) = n$ is the code of an axiom of T is computable is called a *recursively axiomatized theory.* All theories with decidable axioms are of this kind.

When below we talk about the expressibility of a set of sentences in a theory we mean the expressibility of the set of numerical codes associated with the sentences of the set in question. For example, when we consider the expressibility of $Theor(T) = \{E\ T \vdash E\}$ we mean the expressibility of the set $\{code(E)\ T \vdash E\}$.

2. Arithmetical Truth is not Arithmetical

We extend the notion of representability/expressibility in a theory from functions to relations. Let $R \subseteq \mathbf{N} \times \mathbf{N}$ be a binary relation on natural numbers. We say that a formula $F(x, y)$ *represents* or *expresses* in T the relation R if the following holds:

For every $a, b \in \mathbf{N}$

$$R(a,b) \Rightarrow T \vdash F(\bar{a},\bar{b}),$$
$$\neg R(a,b) \Rightarrow T \vdash \neg F(\bar{a},\bar{b}).$$

If R is a computable (i.e., decidable) relation (i.e., its characteristic function is computable), the existence of a formula representing R in **MA** follows from the Representability Theorem for **MA**.

Corollary 2.1. *All computable relations are representable in any theory $T \supseteq$ **MA**.*

Proof. Exercise (hint: a relation is expressible in a theory if and only if its characteristic function is representable in that theory. $\square$

We can ask whether there is a formula $F(x)$ such that it represents/expresses in T the theorems of T, i.e. such that: if $T \vdash E$ then $T \vdash F(\overline{code(E)})$ and if if $T \nvdash E$ then $T \vdash \neg F(\overline{code(E)})$. We will show that this is impossible if $T \supseteq$ **MA** and is consistent. Note that T is not necessarily a formal theory or a decidable theory here!

The interest of the above question comes from the following natural question: is there a formula $F(x)$ in the language of arithmetic such that $F(x)$ defines in the standard model $\mathcal{N}$ the set of sentences that are true in the standard model $\mathcal{N}$? I.e. is there a formula $F(x)$ such that

$$\mathcal{N} \models E \Leftrightarrow \mathcal{N} \models F(\overline{code(E)})?$$

That is we are asking whether the property of being a first-order sentence (in the language of arithmetic) true in the natural numbers can be defined in the natural numbers by a first-order formula in the language of arithmetic. This question is related to the previous one if once considers the theory T consisting of all true sentences in the standard model $\mathbf{N}$. The being expressible in T is equivalent to being definable in $\mathbf{N}$.

We already know that truth in $\mathbf{N}$ cannot be done by a Σ_1 formula, but in principle it could be possible if one uses an arbitrarily complex sentence (with as many alternations of $\forall$ and $\exists$ quantifiers as you want). We show below that this is impossible.

Let D be the following function (called the Diagonal Function):

If u is the code of a formula $A(x)$ with x as the only free variable then $D(u)$ is the code of $A(\overline{u})$.

It is not hard to show that D is computable.

Theorem 2.2. *Let T be a first-order theory. If D is representable in T and T is consistent then the set of theorems of T is not expressible in T.*

Proof. Suppose D is representable in T. Let $F_D(x, y)$ be a formula that represents it. Then for every $k, j \in \mathbf{N}$

$$\text{If } D(k) = j \text{ then } T \vdash F_D(\overline{k}, \overline{j})$$

and, for every $k \in \mathbf{N}$,

$$T \vdash \forall y \forall z((F_D(\overline{k}, y) \wedge F_D(\overline{k}, z)) \to y = z).$$

Suppose by way of contradiction that the theorems of T are expressible in T. Let $F_T(y)$ be the formula expressing the theorems of T in T. Then for each formula H,

$$T \vdash H \Rightarrow T \vdash F_T(\overline{code(H)})$$

and

$$T \nvdash H \Rightarrow T \vdash \neg F_T(\overline{code(H)}),$$

where we denote by $code(H)$ the code of the formula H. Consider the following formula $A(x)$.

$$\forall y(F_D(x, y) \to \neg F_T(y)).$$

Let p be the code of this formula. Consider $A(\overline{p})$:

$$\forall y(F_D(\overline{p}, y) \to \neg F_T(y)).$$

Let q be the code of $A(\overline{p})$. By definition of D we have $D(p) = q$. Therefore

$$T \vdash F_D(\overline{p}, \overline{q}).$$

We now want to prove that $T \vdash \neg F_T(\overline{q})$. Reason by cases.

(Case 1) If $T \nvdash A(\overline{p})$, then $A(p)$ is not a theorem of T and therefore $code(A(\overline{p}) = q \notin Theor(T)$. Therefore, by the definition of expressibility,

$$T \vdash \neg F_T(\overline{q}).$$

(Case 2) If $T \vdash A(\overline{p})$, then also

$$T \vdash F_D(\overline{p}, \overline{q}) \to \neg F_T(\overline{q}).$$

therefore

$$T \vdash \neg F_T(\overline{q}).$$

Thus we proved that $T \vdash \neg F_T(\overline{q})$.

$T \vdash F_D(\overline{p}, \overline{q})$ and from the provable functionality of the formula F_D it follows that

$$T \vdash F_D(\overline{p}, y) \to y = \overline{q}.$$

We also have the following (given that $T \vdash F_T(\overline{q})$)

$$T \vdash y = \overline{q} \to \neg F_T(y).$$

Therefore

$$T \vdash F_D(\overline{p}, y) \to \neg F_T(y).$$

By generalization we then have

$$T \vdash \forall y(F_D(\overline{p}, y) \to \neg F_T(y)),$$

i.e.

$$T \vdash A(\overline{p}).$$

therefore $T \vdash F_T(\overline{code(A(\overline{p}))})$, i.e.

$$T \vdash F_T(\overline{q}).$$

therefore T is inconsistent.

$\square$

Note that the fact that all computable relations/functions are representable in any $T \supseteq \mathsf{MA}$ gives us a tool for proving results about algorithmic non-decidability, *without assuming* the existence of undecidable problems. In fact, if a relation is not representable in a theory extending **MA** then it is not computable. We immediately get the following corollary.

Corollary 2.3. *If T is consistent and all computable functions are represented in T then the theorems of T are not expressible in T. In particular they are not a computable set.*

Corollary 2.4. *If* **MA** *is consistent then each consistent extension of* **MA** *has an undecidable set of theorems.*

Proof. If T is a consistent extension of **MA**, then every computable function is representable in T. $\square$

We now show how the above results imply a negative answer to the question of whether the set of sentences true in the standard model can be expressed by a formula. The following Theorem is due to Alfred Tarski, who proved it soon after listening to Gödel presenting his Incompleteness Theorems.

Theorem 2.5 (Tarski's Undefinability Theorem). *The set of sentences in the language of arithmetic that are true in* **N** *is not definable in* **N** *by a formula in the language of arithmetic.*

Proof. Consider the extension T of **MA** obtained by adding all the sentences true in the standard model **N**. T is consistent (since $\mathbf{N} \models T$). Then the set of theorems of T is not expressible in T. But the set of theorems of T is the set of true sentences in **N**. Furthermore, a relation is expressible in T if and only if it is the interpretation of a formula in **N**. Therefore the arithmetic truths are not definable by a first-order formula in the language of arithmetic (such formulas are called arithmetical formulas). $\square$

From the algorithmic point of view, Tarski's Theorem implies that the problem of deciding the truth of a first-order sentence in the structure $(\mathbf{N}, 0, 1, +, \times)$ in the language of arithmetic has a high degree of unsolvability (is *very* non-computable).

3. Gödel's Second Incompleteness Theorem: the Problem of Consistency

One of the major goals of Hilbert's Program for the Foundations of Mathematics was to establish the consistency of formal theories. Once an axiomatic formal (c.e.) theory is formulated we wish to establish its consistency, i.e., the fact that the theory does not prove a contradiction. This means that there exists no proof of a contradictory formula, e.g., $A \wedge \neg A$, or $0 = 1$. Since proofs are finite formal objects this is an assertion about the non-existence of a finite formal object of a certain kind. The hope of Hilbert was that this non-existence could be proved using *elementary combinatorial reasoning* on the formal theory treated as a symbolic, object. In this way one would in some sense ensure a secure foundation for the theory. The scheme would be the following: prove the consistency of T using reasoning weaker than the one formalized in T, e.g. using reasoning that can be formalized in a theory T' that is weaker than T.

Gödel's Second Theorem shows that the above program is doomed to failure in a precise sense. Since consistency of a theory T is a syntactical property, using the technique of arithmetization one can show that it can be expressed by a formula in the language of arithmetic. Gödel's Second Theorem shows that if T is a consistent extension of **MA** this formula is not provable in T. I.e. one shows that consistency of T cannot be proved by a proof in T. A fortiori it cannot be proved by a theory T' that is weaker than T.

We observed that in any decidable theory T the relation P: "π is a proof of b in T" is decidable. In any T extending **MA** this relation is representable. Let $Proof_T(x, y)$ denote the formula that represents the relation P in T.

The idea of the proof of Gödel's Second Theorem is as follows: let Con_T be the following sentence

$$\neg \exists y Proof(\overline{code(0 = 1)}, y),$$

where $Proof(x, y)$ is the formula expressing the computable relation: $P(a, b)$ iff b is the code of a proof in T of the formula with code a. Then Con_T expresses the consistency of T. Notice that $\mathsf{MA} \vdash \neg(0 = 1)$ so also $\mathsf{MA} \vdash \exists y Proof(\overline{code(\neg(0 = 1))}, y)$. Consider the sentence

$$Con_T \to G$$

where G is Gödel's sentence for T. Then the above sentence says that if T is consistent then G is not provable in T (since G says that G is not provable in T). Thus, intuitively the above sentence is a formal version of Gödel's First Incompleteness Theorem (in particular of the implication: if T is consistent then T does not prove G).

The whole point of the proof of Gödel's Second Incompleteness Theorem is then to show that the above sentence is provable in T:

$$T \vdash Con_T \to G.$$

For, then, we have that $T \nvdash Con_T$, since by Gödel's First Incompleteness Theorem we know that $T \nvdash G$, provided T is consistent. To establish the above provability one needs to analyze the argument we used to prove Gödel's First Incompleteness Theorem and show that it can be formalized in T.

We denote by $Provable(x)$ the formula $\exists y Proof(x, y)$.

The proof of Gödel's Second Incompleteness Theorem starts by showing that the following *derivability conditions* hold, for all sentences E, F.

(DC1) If $T \vdash E$ then $T \vdash Provable(\overline{code(E)}))$.
(DC2) $T \vdash Provable(\overline{code(E \to F)}) \to (\overline{Provable(\overline{code(E)}) \to Provable(\overline{code(F)})}$.
(DC3) $T \vdash Provable(\overline{code(E)}) \to Provable(\overline{code(Provable(\overline{code(E)}))})$.

The above conditions in words say the following: the first say that if T proves E then T proves that T proves E. The second says that if T proves that T proves an implication $E \to F$ then T proves the implication: if T proves E then T proves F. The third says that if T proves that T proves E then T proves that T proves T proves E.

We will not give a proof of the above derivability conditions. For a detailed proof we refer to W. Rautenberg, *A Concise Introduction to Mathematical Logic*, Section 7.1. We only observe that they hold in classical theories such as Peano Arithmetic (which is essentially **MA** plus the induction scheme for all formulas F

$$F(0) \wedge \forall x(F(x) \to F(x + 1)) \to \forall x F(x).$$

For readability we write $\Box E$ for $Provable(\overline{code(E)})$, i.e. $\exists y Proof(\overline{code(E)}, y)$. Then the derivability conditions can be written as follows:

(DC1) If $T \vdash E$ then $T \vdash \Box E$.
(DC2) $T \vdash \Box(E \to F) \to (\Box E \to \Box F)$.
(DC3) $T \vdash \Box E \to \Box\Box E$.

Con_T is then $\neg\Box(0 = 1)$.

Remark We observe that, if the derivability conditions hold, then Con_T is independent of the choice of the contradiction $(0 = 1)$. This follows from the fact that the following rule is derivable from the above conditions:

$$\text{If } T, E \vdash F \text{ then } T, \Box E \vdash \Box F.$$

Then

$$T \vdash E \leftrightarrow F \Rightarrow T \vdash \Box E \leftrightarrow \Box F.$$

So, one can define Con_T using any contradictory sentence C instead of $(0 = 1)$ as long as $T \vdash C \leftrightarrow (0 = 1)$.

Note that $DC1$ holds in **MA** since it is a consequence of the Representability Theorem: If $T \vdash E$ then $T \vdash Proof(\overline{code(E)}, \overline{d})$ for d the code of a proof of E in T and therefore $T \vdash Provable(\overline{code(E)})$, i.e. $T \vdash \Box E$.

On the other hand, the converse of $DC1$ does not necessarily holds, but it holds in ω-consistent theories extending **MA**.

Theorem 3.1 (Fix-Point Theorem)**.** *Let $T \supseteq$ **MA**. For any formula $F(x)$ with x as the only free variable there exists a sentence E such that*

$$T \vdash E \leftrightarrow F(\overline{code(E)}).$$

E is called a fix-point of $F(x)$.

Proof. Let $F_D(x, y)$ be a formula that represents the diagonal function D in T. Consider the formula

$$\forall y(F_D(x, y) \to F(y)).$$

Let p be the code of the above formula. Consider the following sentence E:

$$\forall y(F_D(\overline{p}, y) \to F(y)).$$

Let q be the code of this sentence. By definition of D we have that $D(p) = q$. By the Representability Theorem, $T \vdash F_D(\overline{p}, \overline{q})$.

We now prove $T \vdash E \to F(\overline{q})$.

Assume E, i.e. $\forall y(F_D(\overline{p}, y) \to F(y))$. Then $F_D(\overline{p}, \overline{q}) \to F(\overline{q})$. Since $T \vdash F_D(\overline{p}, \overline{q})$ we have $F(\overline{q})$ by Modus Ponens. By the Deduction Theorem the proof is finished.

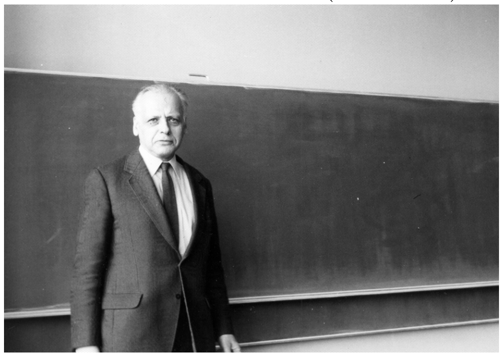
We now prove $T \vdash F(\overline{q}) \to E$.

Assume $F(\overline{q})$ and $F_D(\overline{p}, y)$. Since $F_D(x, y)$ represents a function we have that T proves injectivity of $D(x, y)$, thus T proves that there exists a unique y such that $F_D(\overline{p}, y)$. Since we already know that T proves $F_D(\overline{p}, \overline{q})$ we have $y = \overline{q}$. Therefore from $F(\overline{q})$ we deduce $F(y)$. We have thus established $T, F(\overline{q}), F_D(\overline{p}, y) \vdash F(y)$. Thus by the Deduction Theorem $T, F(\overline{q}) \vdash F_D(\overline{p}, y) \to F(y)$ and thus $T, F(\overline{q}) \vdash \forall y(F_D(\overline{p}, y) \to F(y))$. Again by the Deduction Theorem the proof is finished. $\Box$

Note that by applying the above theorem to the formula $\neg Provable(x)$ one gets a sentence E satisfying $T \vdash G \leftrightarrow \neg Provable(\overline{code(G)})$ that is essentially Gödel's unprovable sentence. Applying the theorem to the formula $Provable(x)$ one gets a sentence H satisfying $T \vdash H \leftrightarrow Provable(\overline{code(H)})$ that asserts its own provability. If H true in the standard model? Is it provable or disprovable in T?

We show that H is provable in T and true in the standard model.

Figure 1. Martin Löb (1921-2006)



Theorem 3.2 (Löb's Theorem). *Let T be an extension of* **MA** *for which the Derivability Conditions hold. If $T \vdash \Box E \to E$ then $T \vdash E$.*

Proof. Consider the formula $Provable(x) \to E$. Apply to it the Fix-point Theorem to get L such that

$$T \vdash L \leftrightarrow (Provable(\overline{code(L)}) \to E),$$

that is

$$T \vdash L \leftrightarrow (\Box L \to E).$$

We obtain a proof of E in T by reasoning as follows. From $T \vdash L \to (\Box L \to E)$ and (DC1) we get $T \vdash \Box(L \to (\Box L \to E))$ and by (DC2) and Modus Ponens we get $T \vdash \Box L \to \Box(\Box L \to E))$. Again by (DC2) and logic we get $T \vdash \Box L \to \Box\Box L \to \Box E$. From (DC3) we have $T \vdash \Box L \to \Box\Box L$ and therefore $T \vdash \Box L \to \Box E$. By choice of E we know $T \vdash \Box E \to E$ and thus $T \vdash \Box L \to E$. Since we already know $T \vdash L \leftrightarrow (\Box L \to E)$ we infer $T \vdash L$. By (DC1) we have $T \vdash \Box L$. Therefore $T \vdash E$. $\Box$

Theorem 3.3 (Gödel's Second Incompleteness Theorem). *Let T be a formal extension of* **MA**. *If T is consistent then $T \nvdash Con_T$.*

Proof. $T \vdash 0 \neq 1$. Since T is consistent $T \nvdash (0 = 1)$. Therefore $T \nvdash \Box(0 = 1) \to (0 = 1)$. Therefore $T \nvdash \neg\Box(0 = 1)$. $\Box$

4. Extra: Gödel's original approach

We first describe a coding function - *code*( ) - similar to the one used by Gödel. The function assigns to syntactic objects such as variables, terms, formulas, proofs, a numerical code. This is roughly done as follows.

First define *code*( ) for each symbol of the language (variables, constants, relation and function symbols, parentheses, quantifiers, boolean connectives). For example, if we are dealing with the language of arithmetic we assign a number to each variable v_i, to the constants 0 and 1, to the function symbols $+$ and $\times$ and to the relation symbol $<$, as well as to all logical symbols (,), $=$, $\exists$, $\forall$, $\neg$, etc. of the formal language. It is useful to assign odd positive integers as code to these symbols.

Next we want to be able to assign a numerical code to formal expressions of the language, e.g., to formulas such as $(\exists v_3)((v_3 + 0) < v_6)$. These are just finite ordered sequences of symbols: $u_0 u_1 \ldots u_t$ and we code them as follows, where $p_0, p_1, p_2, \ldots$ are the primes in increasing order:

$$code(u_0 u_1 \ldots u_t) = 2^{code(u_0)} 3^{code(u_1)} \ldots p_t^{code(u_t)}.$$

Next we want to code proofs, which are just finite sequences of formulas. So we want to code sequences of expressions: $e_0 e_1 \ldots e_t$, where each e_i is a finite sequence of symbols. We do this as follows:

$$code(e_0 e_1 \ldots e_t) = 2^{code(e_0)} 3^{code(e_1)} \ldots p_t^{code(e_t)}.$$

The *code* function is injective, and moreover codes of symbols can be distinguished from codes of expressions and codes of expressions from codes of sequences of expressions (Why?).

We next describe Gödel's original proof of his First Incompleteness Theorem.

The goal is to show that any consistent decidable theory of arithmetic that is at least as strong as Minimal Arithmetic is incomplete. Let's fix one such theory T.

Consider the following relation $R(a,b) \subseteq \mathbf{N} \times \mathbf{N}$:

$R(a,b)$ iff a is the code of a formula with a single free variable and b is the code of a proof in T of the formula obtained by substituting the numeral $\overline{a}$ in the formula coded by a to all free occurrences of the unique free variable of that formula.

E.g., if $a = code(\exists v_4(v_7 + 1 = v_4 \times v_4))$ and b is a code of a proof in T of the sentence $\exists v_4(\overline{a} + 1 = v_4 \times v_4)$ then $(a,b) \in R$.

It is not hard to convince oneself that R is a computable relation (in fact it is primitive recursive). Let $F(x,y)$ be a formula that *represents* in T the relation R. The existence of such a formula follows from the Representability Theorem for **MA**. By definition this means that the following implications hold:

For every $a, b \in \mathbf{N}$

$$R(a,b) \Rightarrow T \vdash F(\overline{a}, \overline{b}),$$
$$\neg R(a,b) \Rightarrow T \vdash \neg F(\overline{a}, \overline{b}).$$

We consider the following formula.

$$(\forall y)\neg F(x,y).$$

Let m be a code of this formula. We consider the following sentence G.

$$(\forall y)\neg F(\overline{m}, y).$$

Theorem 4.1 (Gödel's First Incompleteness Theorem, Part I)**.** *Let T be a decidable theory extending* **MA***: If T is consistent then G is unprovable in T.*

Proof. Let T be consistent and suppose that T proves G. Let d be the code of such a proof. Then $R(m,d)$. Then $T \vdash F(\overline{m}, \overline{d})$. Therefore $T \vdash \exists y F(\overline{m}, y)$. Therefore T is inconsistent. □

Let's try to prove that $T \nvdash \neg G$, so as to show that T is incomplete. Suppose by way of contradiction that $T \vdash \neg G$. I.e., $T \vdash \exists y F(\overline{m}, y)$. By consistency of T, T does not prove G. Then, for each number n, n is not the code of a proof of G in T, i.e., for every n, $R(m,n)$ does not hold. Therefore for every n, $T \vdash \neg F(\overline{m}, \overline{n})$. Intuitively a theory T such that

$$T \vdash \exists y F(\overline{m}, y),$$

and

$$\text{For all } n \in \mathbf{N} (T \vdash \neg F(\overline{m}, \overline{n})$$

is not desirable, but we cannot exclude that this happens for a consistent theory T.

To solve (or, rather, elude) the problem Gödel introduced the following notion of ω-consistency, which essentially rules out this undesired behaviour.

Definition 4.2 (ω-consistency)**.** A theory T is ω-consistent if there is no formula $A(x)$ such that

$$\text{For each } n \in \mathbf{N} \;\; T \vdash A(n),$$

and

$$T \vdash \neg(\forall x) A(x).$$

It is easy to show that an ω-consistent theory is also consistent (Exercise). Under the assumption of ω-consistency we can conclude the above argument.

Theorem 4.3 (Gödel's First Incompleteness Theorem, Part II). *If T is ω-consistent then $\neg G$ is unprovable in T.*

Proof. Let T be ω-consistent and suppose that T proves $\neg G$, i.e., $\exists y F(\overline{m}, y)$. By consistency of T, T does not prove G, as shown above. Then, for each number n, n is not the code of a proof of G in T, i.e., for every n, $R(m, n)$ does not hold. Therefore for every n, $T \vdash \neg F(\overline{m}, \overline{n})$. But then T is ω-inconsistent. $\square$

One can improve the above results by showing that the assumption of consistency is sufficient to show that T does neither prove nor disprove G. This result is due to Rosser.

Let $H(x, y)$ be a formula that represents in T the following relation $S(a, b) \subseteq \mathbf{N} \times \mathbf{N}$:

$S(a, b)$ iff a is the code of a formula with a single free variable and b is the code of a proof in T of the formula obtained by substituting the numeral $\overline{a}$ to each free occurrence of the variable of the formula coded a in the negation of the formula coded by a.

Consider the following formula.

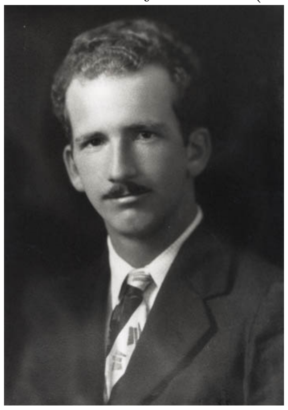
$$(\forall y)(F(x, y) \to (\exists z)(z \leq y \land H(x, z))).$$

Let m be the code of this formula. We consider the following sentence E.

$$(\forall y)(F(\overline{m}, y) \to (\exists z)(z \leq y \land H(\overline{m}, z))).$$

The following Theorem is due to Rosse.

Figure 2. J. Barkley Rosser (1907-1989)



Theorem 4.4 (Rosser's Theorem). *If T is a formal consistent theory extending* **MA** *then T is incomplete.*

Proof. Exercise. $\square$