80-990 Keith Douglas
w/ Wilfried Sieg kdouglas@andrew

Turing's Analysis of Computation (Part 1: Historical Influences)

Introduction

As is well known Turing's epochal 1936 paper (Turing 1936. All page references to Turing hereafter are to this document unless otherwise stated) was one of several papers from that time to provide an analysis of computability. Since this analysis was done in the context of providing a (negative) answer to one specific problem (the Entscheidungsproblem) it is important when understanding the limitations and features of Turing's analysis to keep in mind this intended goal.

The present paper discusses the background to Turing's paper in order to get a grasp on the intellectual climate that surrounded his argument. It is not the goal of the present author to show that these specific works had a direct influence on Turing; we are instead concerned with the general intellectual climate in logic and metamathematics that would have greeted his proposal. Thus we may even examine what Herbrand's work has in common with Turing's despite the fact that Herbrand did not live to see Turing's paper.

In particular, this paper examines the historical context for Turing's problem and shows that many of his contemporaries' views on mathematics and the philosophy thereof are consonant with the features of Turing's paper that made it unique. These, as noted by Sieg (1994) are those grounded in the limitations and abilities of the agent performing the computations in question, but are not limited to such. After Sieg I shall call this agent the "computor."

The predecessors and contemporaries of Turing I will discuss are: Hilbert and Bernays, Ackermann, Herbrand, Gödel, Church, Rosser, Kleene, Löwenheim, and Post. This paper thus has 6 sections. The first of these will discuss Hilbert, Ackermann and Herbrand. As we shall see these are of a kind; from that we can move to Gödel. Gödel stands between what might be called the "Hilbert school" and

the "Princeton School" of Church, Rosser and Kleene, and thus gets his own section. The Princeton school follows in its own section. I shall also briefly discuss Post and Löwenheim, who fall somewhat outside this scheme and get a brief section each.

The Hilbert School and Turing

Given that Hilbert himself formulated the first decision problems, it is not terribly surprising that his own understanding of mathematics should play a substantial role in their resolution. Despite the fact that Turing himself was not literally a member of the Hilbert school, we shall see that the "formalist" character of the Hilbert school is adopted in some details by Turing.

I start by examining key philosophical claims by Hilbert (1925) in his address "On the Infinite." As the title of the address suggests, Hilbert is concerned therein with the nature of the infinite. In particular, he is concerned with how we come to know about it. Hilbert's motivations are thus broadly speaking epistemological in character. His suggestion that we consider the infinite a *façon de parler* is the first feature Turing is to adopt. Turing's computors too are finitistic in character in three ways. One is that they may only make use of a finite number of internal states (i.e., Turing's states of mind: pp. 136); a second is that they may only use (at any given time) a finite amount of scratch memory (tape) (pp. 117). Third, they may only take a finite amount of time and steps to produce their answer. This last feature is returned to again by Hilbert in his discussion of the difference between infinite and unbounded. The tape or paper used by the Turing computor is unbounded in Hilbert's sense.

As is well known, Turing's computors are deterministic. This too is brought up by Hilbert in the context of his discussing rules of inference. He tells us that these should be finitist processes that produce the same result each time they are applied in a given context. This is exactly parallel to Turing's requirement that he analyse only what he calls "automatic" machines (pp. 118). Not only does this restriction make Turing's rules deterministic, but

they can thus be seen also as a rule of inference. This is because they are rules of the form "if X then Y" where the conditional is understood as expressing a rule of replacement or whathaveyou and X and Y are well specified. Later on (pp. 375-6), Hilbert returns to rules of inference. He tells us that these must be reliable. This too is adopted by Turing, whose computor's rules' applicability are to be mechanically performable and free from error.

Curiously, however, there is a bit of difference between Hilbert and Turing here as well. Turing writes (118, emphasis added):

> "For some purposes we might use machines (choice machines or *c*-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in §1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice is made by an external operator. **This would be the case if we were using machines to deal with axiomatic systems.** In this paper I deal only with automatic machines, and therefore often omit the prefix *a*-."

If Hilbert takes himself as elucidating formal systems (in the sense of axiomatic, above) then Turing has disagreed with one key part of his presentation. Hilbert would say that the formal system should not be ambiguous in the way that Turing emphasizes above. At the very least we can say that Hilbert and Turing are not using "axiomatic" and "formal" in the same respects.

Another feature of Hilbert's discussion is its understanding of the goal of mathematics. Hilbert tells us we do not want our mathematical systems to be merely free from contradiction; we also want them to exhibit success. The parallel in Turing here is his emphasis on the **outcome** of **well defined** procedures. The computor reaches a definitive answer if one can be found by it (assuming of course that it has been "programmed" correctly). The computer cannot answer "Maybe" or "Outlook hazy, try again later" to the problem posed to it. (It is not clear to me whether Hilbert would have accepted a procedure that possibly "loops forever" as an acceptable well defined procedure. Such are clearly acceptable to Turing.) Another way of looking at this parallelism is through the

reflection principle. I.e., Hilbert suggests if that we have a proof of φ in some formal system, then φ. This is parallel to Turing showing us that if we have a computational procedure for producing φ, then φ. Hence, a proof of the reflection principle is parallel to Turing's argument for the powers of his machines.

On page 376, Hilbert introduces what would be to some mathematicians a shocking feature of his views. He tells us that it is necessary that in order to do mathematics (and not get caught up in the dangers of the infinite), the mathematician must be able to survey certain **concrete** objects. This feature has several analogous roles in Turing's work. It plays a role in the use of the paper or tape that the computor uses as an *aide memoire*. The symbols on the tape are concrete, discrete and sufficiently different from each other, and thus easily surveyable. The computor's rules are also to be similar: as noted previously Turing tells us that it should be unambiguous which one is to apply at any given stage of a computation (pp. 118 and 136); thus the rules have to be well specified and surveyable too. (It is true that Turing says the rules need not be written down *per se*, but we can assume that "surveyable" can include "surveyable in the mind's eye" or similar internal capacities of the computor.)

Hilbert's methodological and epistemological suggestions also anticipate the Turing (and Gödelian) results that there is no uniform procedure for solving all mathematical problems. Hilbert says quite explicitly (pp. 384) that he expects this negative metamathematical result[1] .

Finally (for our purposes), Hilbert discusses how to define functions in his system. He tells us that we ought to define functions by substitution and recursion. This (understanding the species of recursion in the right way) is how Turing's computors

[1] It strikes me that this speaks against the usual presentation of Gödel as striking a blow against the Hilbert formalist school with his incompleteness theorems, but that is another story for another time.

also "define" or represent functions (pp. 141).

Bernays' 1927 is called an appendix to Hilbert's lecture (above) with good reason. For our purposes it is important to note that it anticipates another important result in computational procedures. If a schema of induction is taken to be similar to a search procedure, Bernays' proof that his process terminates after a certain number of steps is parallel to the estimation of the running time of an algorithm. It is from this sort of consideration we can see that Turing (or Post; see below)'s process for doing something is inefficient. We thus can distinguish between correctness and the further requirement of efficiency.

Ackermann directly builds on Hilbert's ideas as well. We now turn to his contributions.

The parallels between Turing and Ackermann's (1928) work are not nearly as important as those between Turing and Hilbert, and not as critical as the connections between Löwenheim (or Post) and Turing. Nevertheless, there is one feature that is worth noting. This is a remark on the nature of primitive functions. Seemingly innocuous, it has has deep significance[2]. Ackermann tells us (pp. 496 and pp. 501) that certain functions cannot be defined by recursion and that we have to start with certain functions as given, e.g. the successor function. The parallel in Turing are the simple operations the computor can perform (pp. 137). Turing and Ackermann both emphasize these "given" operations to avoid an infinite regress.

It is important to note, however, that Turing's operations need not be understood as functions in the strict sense, in so far as they involve marks on paper and so forth. If we look at their "meaning" (not necessary for Turing's analysis) then they may correspond to Ackermann's primitive functions, because as noted above (in the section on Hilbert) Turing's rules are

[2] Some of which will not be brought up until this work's sequel.

deterministic. Hence they have single results for a given input and state which corresponds nicely (and deliberately) to the single-valued nature of the notion of function in mathematics proper.

It is not of course necessary that Turing's primitive operations correspond to the successor function Ackermann uses as a starting point.

Herbrand is also very much in the Hilbert tradition. His work on proof theory (1931) is evidence of that. That said, it should come as a surprise that much of what is common to Herbrand and Turing has already been discussed earlier. However, section 6A of his 1931 certainly does have an important role to play in the work of Turing's which is to follow it. Here Herbrand discusses the importance of the earlier parts of this work for the decision problem itself! In particular, he resolves the decision problem for certain specific cases. Thus it is pretty clear that Turing's analysis should not contradict Herbrand's. For instance, Turing would know his analysis was wrong if it suggested that no cases of the decision problem were decidable.

Now that we have met the Hilbert school, we must move slowly towards Princeton and the Church school. But standing in the middle is Gödel. We now examine his significant contributions on our theme.

Gödel and Turing

Turing's work is often compared to Gödel's masterpiece (1931). This comparison is with good reason; there are many substantial points of similarity between the two works. Here we focus on those having to do with our theme; there are others having to do with the mathematical results each paper produces.

The first of these is found in Gödel's first sentence. He writes (pp. 596):

> "The development of mathematics towards greater precision has led, as is well known, to the formalization of large tracts of it, so that one can prove any theorem using but a few mechanical rules."

The above passage mentions formalization (and by extension, mechanization). Turing's computor is mechanized in the sense that it is not allowed to be "creative" in any way, nor is it allowed to deviate from the rules (and state space) it is given at the beginning of its operation.

Gödel also emphasizes the finitist character of his objects of concern. He tells us (pp. 597) that the results he has found apply not just to Principia Mathematica (PM) or Zermelo-Fraenkel (ZF) set theory, but indeed to any system that results from those two by adding a finite number of axioms. If one takes the Hilbert formalist line seriously (which is of course not to say that Gödel or even Turing did) then the axioms of PM (or ZF) do have the character of the collection of rules in Turing's system. Along the same lines, Gödel's mechanized proofs are analogous to sequences of computations. This counterpart is all the more striking when one recalls that Gödel was able to code such proofs as sequences of natural numbers; Turing's coding of Turing machines so that they can be parsed by the Universal Turing machine is parallel (pp. 154 ff).

Further, Gödel's paper also discusses in some detail (pp. 602 ff) definitions by recursion and substitution similar that we have already met in Ackermann. We need not repeat ourselves here. However, there is one specific development of Gödel's here that is important. This is the development of what might be called "search functions" in the formalism he develops. These allow one to formalize search problems like looking for the least number that satisfies some expression, applying the characteristic function of a particular set on a putative element, and so forth. The Entscheidungsproblem is such a problem, and so the ability to represent more than "ordinary calculation" (i.e. more than just

addition, multiplication and so forth)[3] in formal systems is thus critical for Turing's work. Gödel even gives a formal characterization of **decidable**. Needless to say, this concept is central to Turing's paper. Again the details of implementation are not important; the showing that it can be done is.

Turing himself also compares his work to that of Gödel. He tells us that if the contrary to the results of the incompleteness theorems it would have settled the Entscheidungsproblem in the positive.

Another paper of Gödel's which is of some importance for understanding Turing's work is his 1936. Here Gödel discusses a family of formal systems $\mathbf{S_i}$. Each of these systems have certain functions which are computable. These are analogous to each of the Turing machines; the S_1 system Gödel describes is analogous to the universal Turing machine because it computes any function computable in any of the S_is. This is also of note because Gödel uses this fact to remark that there is a <u>disanalogy</u> between proofs and programs. As he says (pp. 83):

> "Thus, the concept 'computable' is in a certain sense 'absolute,' while practically other familiar metamathematical concepts (e.g. provable, definable, etc.) depend quite essentially on the system with respect to which they are defined."

Finally, we have some remarks by Gödel after the fact (see Gödel 1964) on his views of Turing's work. He tells us that "mechanical procedure" in his work is shown to be **equivalent** to that of a "Turing machine" (pp. 72). Gödel also tells us that the question of whether there <u>are</u> any non-mechanical procedures is not settled by Turing's analysis.

von Neumann was one of those responsible for bringing Gödel's work

[3] Of course, these search functions (etc.) get <u>reduced</u> to arithmetic ones, but that is not the issue. In fact, that they get reduced such is handy, because that reduces the number of primitive operations needed by Turing's computor, and hence makes the process it performs "more mechanical."

to Princeton. As has already been pointed out by Sieg (1994), von Neumann was convinced that the decision problem could not possibly be solved in the affirmative. Nevertheless, one of the first to continue on from Gödel's insights was Church, to which we now turn.

The Princeton School: Church, Rosser, Kleene's influence on Turing

In his (1935), Church uses one of our key terms "effectively" in an important way. He uses the term as an antidote to triviality. On pp. 89, he writes (emphasis added):

> "There is a class of problems of elementary number theory which can be stated in the form that is required to find an **effectively calculable** function f of n positive integers such that $f(x_1, x_2, ..., x_n) = 2$ is a necessary and sufficient condition for the truth of a certain proposition of elementary number theory involving $x_1, x_2, ..., x_n$ as free variables.
>
> An example of such a problem is the problem to find a means of determining of any given positive integer n whether or not there exist positive integers x, y, z, such that $x^n + y^n = z^n$. For this may be interpreted, required to find an **effectively calculable** function f, such that $f(n)$ is equal to 2 if and only if the there exist positive integers such that $x^n + y^n = z^n$. Clearly the condition that the function be **effectively calculable** is an essential part of the problem, since without it the problem becomes trivial."

Church is suggesting that solutions to these problems must be effective in the sense they must give us an answer to the question in a finite amount of time. While it is true that Turing later will add to this notion to further refine the notion, we have a key expression of part of his understanding here in Church.

Another feature that is important here is the rephrasing the question to be investigated in terms of evaluating a function. As we have noted (in our section on Gödel), this makes it possible to set up precise rules by which questions can be investigated, a detail vital to Turing's work.

However, as has been noted by Sieg (1994), there are limitations in Church's analysis that Turing was to overcome. One of these is

in his footnote 10 (pp. 95). Here he responds to a possible objection from a critic complaining that an algorithm he supplies is not effective. Here he connects "effective" with "constructive." This is interesting, but alas also unfortunate, as he tells the reader that:

> "What the criterion of constructiveness shall be is left to the reader."

It is also interesting that this remark is made in the context of what we would now (in, e.g., Avigad 2002) call "unbounded search." Church describes a procedure that may not terminate. This introduction of partial functions (another way to view this concern) is later critical in Turing when he introduces circle-free and circular machines (pp. 119).

Of course, an even more interesting part of Church's paper is the notion of lambda definability he introduces. This is interesting in connection with Turing's work for two reasons. First, it attempts to give a precise characterization of "effectively calculable", a task Turing himself was to attempt in his own paper. Second, it illustrates (like Gödel's 1931) the great power of coding. In particular, it shows that even integers can be coded; they need not be taken as primitive. (Some of this can also be found somewhat in Hilbert (1926, pp. 377), where he suggests that 1 be a symbol for I, 2 a symbol for II, etc. where the I, II, ... are marks on paper. Second, it raises the question of universality. Church (tries to) shows that there is a lambda expression that calculates any effectively calculable function. A uniform procedure in one sense is thus found. However, Turing was able to go one step further, and "combine" all the uniform procedures into one further uniform procedure, the Universal Turing Machine. Of course, later, it was shown that there is indeed a universal lambda expression: the Y-combinator. But that was not done at this time is critical. Turing's universalism is thus an advance over Church's admirable effort.

Church, in his section 7 (pp. 100-102), makes a further discussion

of effective calculability. There are several key features of this part. First, is that Church is giving a definition of effective calculability herein, rather than "merely" an analysis. He tells us to simply identify "effectively calculable function" with "recursive function" and with "lambda definable" where this notion is as developed earlier in the paper.

Second, Church says that a function is effectively calculable if there is an algorithm that computes its values. Here "algorithm" should be taken as a pretheoretic notion. He motivates this suggestion by showing how one can build up effectively calculable functions out effectively calculable steps. The use of atomic steps will also be crucial for Turing, as we have already noted above in our discussion of Ackermann.

Third, Church introduces the notion of "rules of procedure" which consists of expressions obtainable from axioms given some denumerable list of operations. This is analogous to Turing's list of rules, which also uses a denumerable list of operations and finite applications of these operations.

Church's 1936 paper is also somewhat important; this paper reaches the same result as Turing's though by a different means. This is not terribly important for the intellectual climate concerns we are discussing, except in so far as it supplies the usual "sanity check" on Turing's argument.

Kleene's 1935, like Church's from the following year (see above) also introduces some undecidable problems, as well as recursive definitions of functions and many other common features we know by now. Kleene discusses the fact that determining which systems of equations define recursive functions is not recursively enumerable. This, as is well known now, is connected intimately to Turing's diagnalization argument over Turing machines. As he says on pp. 248:

"The definition of general recursive function offers no constructive process for

> determining when a recursive function is defined. This must be the case, if the definition is to be adequate, since otherwise still more general 'recursive' functions could be obtained by the diagonal process."

Kleene also stresses the difference between primitive and general recursive as well as their relation to functions that are recursive simpliciter. Having these various notions on the table is important for Turing to give precise characterization of the power of his computors.

Rosser was also a key figure in what we have called the "Princeton school"; he was later (1939) to informally expound some of Gödel and Church's results. More importantly for our purposes is his 1936 paper which directly deals with the Entscheidungsproblem.

Here we find several key aspects on our theme. The first of these is his stress on Kleene's distinction between general and primitive recursive functions. The second of these is his agreement with Church that we should **identify** "effective calculability" with one of our exact notions: in Rosser's case it is with the general recursive functions. Rosser also proves a version of the non-decidability of the decision problem. His proof is, as might be surmised, makes essential use of the notions of recursive functions we have mentioned. We thus have another constraint upon Turing here: anything he develops must be translatable into the language of recursive functions in order that his results agree with those of Rosser. A final important contribution on our subject that Rosser makes is the weakening of Gödel's results to only require consistency, rather than ω-consistency. This is important, for as is well known, there is a strong connection between Turing's negative solution to the decision problem and the Gödelian results. This modification of Gödel's results by Rosser also "lower the bar" for Turing. By only requiring consistency, in principle this could lower the requirements for Turing's own proofs.

Now that we have met the two "schools" and their bridge, the final

two sections before our conclusion concern two important but peripheral figures, Löwenheim and Post. We devote a section to each in the following.

Löwenheim and Turing

Löwenheim (1915) introduces the Entscheidungsproblem and provides a solution of it for the special case of the singular predicate calculus with identity. For that reason alone he is an (if not the most) important predecessor of Turing. However, there is very little else in his paper beyond this of direct relevance to our theme. It should, however, go without saying that Turing's results must "reduce" in some way to Löwenheim's in the limit of the singular predicate calculus.

Post and Turing

The connections between Post (1921) and Turing are important. Post gives via his (then new) method of truth tables an example of an effective procedure, and further one that solves a parallel decision problem for the case of propositional logic thereby. The truth table method for solving this decision procedure is quite well known. It simply consists in assigning to each propositional letter in turn each of the two truth values and seeing if the truth function is thereby satisfied. Features of this procedure that are important as far as Turing is concerned are: (a) it is finitistic: each propositional formula is composed of a finite number of elementary propositions, so all combinations of their possible truth values is thereby also finite; (b) it is effective in the sense that it can be done without understanding[4]. In fact, Post uses "+" and "-" as truth values - one need not even know the meaning of what the two values are; (c) it also shows that feasibility of a decision procedure is not important. Since the

[4] Now of course we know this because we can easily program a computer to perform Post's algorithm. At the time, however, it is interesting to note that someone might well have claimed that it wasn't mechanical enough: perhaps because it required recognition of some kind or other. Even syntactic recognition is not as mechanical as one might think, or could be, and that's enough to drive the possible objection.

"running time" of the Post algorithm is exponential in the number of elementary propositions, it takes an awfully long time for even very simple compound propositions to be checked, particularly for a human, as Post obviously has in mind. Note, however, that these features are not actually discussed by Post himself.

Post also stresses the formal character of what he is proving results about (pp. 266):

> "Finally, a word must be said about the viewpoint that is adopted in this paper and the method that is used. We have consistently regarded the system of *Principia* and the generalizations thereof as purely *formal developments*, and so have used whatever instruments of logic or mathematics we found useful for a study of these developments."

Turing's own development is parallel in a certain way. He is able to prove results about general features of the Turing machines without regards to specific details of the functions they compute (pp. 132-134).

Conclusions and Future Directions

From the above sections we can extract several recurring themes from Turing's intellectual *zeitgeist* that played an important role in the development of his epochal paper. Most of these are summarized in the table below (see over, figure 1). Italics denote Turing's terminology. I do not mean to suggest that these notions are only found in the author given, just that they are key points for Turing's purposes in that author's work. I take "Turing's use" as he might have understood it, too, in particular when it comes to effective computation.

Now that we have seen what Turing built upon, a future project is to see how these notions play a role in constraining any model of computation that would claim to be a counterexample to his analysis of computation. (Or, slightly more contentiously, what would count as a counterexample to the Church-Turing thesis which was later to "develop" out of these notions we have sketched.)

Notion	Source	Turing's Use
Decision problems	Hilbert	Resolved one in the negative
Finitary reasoning	Hilbert	Computor is finitary
Deterministic	Hilbert	Computer is *automatic*
Success as goal in math	Hilbert	Well defined results that produce unambiguous results
Concreteness	Hilbert	Computer makes use of marks on paper
Broad notion of recursion	Hilbert	Functions calculable by computor are recursive
Well defined decision procedure	Post	Shows that there is none for a given decision problem
Feasibility of procedures not important	Post	Computor very inefficient but "gets the job done eventually" (if the job can be finished)
Formal development	Post	Computation procedure analyzed without regard to meaning
Primitive operations	Ackermann	Write a symbol; move a square left or right
Mechanization	Gödel	Computor behaves mechanically, is not "creative"
Coding	Gödel	Computation sequences
Previous cases of decision problem	Herbrand	Allows that restricted predicate calculus be decidable
Effective calculability	Church	Analyzed this pretheoretic notion
Unterminating search procedures	Church	*Circular* machines
Lambda calculus	Church	A rival to his own analysis of effectively calculable
Integers need not be primitive	Church	Marks on paper in arbitrary notations
Negative solution to the Entshiedungsproblem	Church	Need to agree with Church or have clever argument to show why he's wrong
Which systems equations define recursive functions is not recursively enumerable	Kleene	Diagonalization over Turing machines
General vs. primitive recursive	Kleene	Strongest possible "effective" computors

Figure 1 - Table of Influences on Turing

Works Cited

Ackermann, Wilhelm. 1928. "Zum Hilbertschen Aufbau der Reellen Zahlen." Translated as "On Hilbert's construction of the real numbers." in van Heijenoort 2000 (1967).

Avigad, Jeremy. 2002. Lecture notes for Carnegie Mellon University course 80-311, Spring 2002 semester.

Bernays, Paul. 1927. "Zusatz zu Hilberts Vortrag über 'Die Grundlagen der Mathematik'. Translated as "Appendix to Hilbert's Lecture: 'The Foundations of Mathematics." in van Heijenoort 2000 (1967).

Church, Alonzo. 1935. "An Unsolvable Problem of Elementary Number Theory". Reprinted in Davis 1964.

Church, Alonzo. 1936. "A Note on the Entscheidungsproblem." Reprinted in Davis 1964.

Davis, Martin. 1964. *The Undecidable*. Hewlett: Raven Press.

Gödel, Kurt. 1931. "Über formal unentscheidbare Sätze der Principa mathematica und verwandter Systeme I" Translated as "On Formally Undecidable Propositions of Principia Mathematica and Related Systems I." in van Heijenoort 2000 (1967).

Gödel, Kurt. 1936. "Über de Länge von Beweisen." Translated as "On the Length of Proofs" in Davis 1964.

Gödel, Kurt. 1964. "Postscriptum". In Davis 1964.

Herbrand, Jacques. 1931. "Recherches sur la théorie de la démonstration." Translated as "Investigations in Proof Theory" in van Heijenoort 2000 (1967).

Hilbert, David. 1926 "Über das Unendliche" Translated as "On the Infinite" in van Heijenoort 2000 (1967).

Kleene, Stephen. 1935. "General Recursive Functions of Natural Numbers." Reprinted in in Davis 1964.

Löwenheim, Leopold. 1915. "Über Moglichkeiten im Relativkalkül" Translated as "On Possibilities in the Calculus of Relatives" in van Heijenoort 2000 (1967).

Post, Emil. 1921. "Introduction to a general theory of elementary propositions." Reprinted in van Heijenoort 2000 (1967).

Rosser, John. 1936. "Extensions of Some Thoerems of Gödel and Church." Reprinted in Davis 1964.

Rosser, John. 1939. "An Informal Exposition of Proofs of Gödel's

Theorem and Church's Theorem." Reprinted in Davis 1964.
Sieg, Wilfried. 1994. "Mechanical Procedures and Mathematical Experience". In A. George, ed. *Mathematics and Mind.* Oxford: Oxford University Press.
Turing, Alan. 1936. "On Computable Numbers, With an Application to the Entscheidungsproblem." Reprinted in in Davis 1964.
van Heijenoort, Jean. 2000 (1967). *From Frege to Gödel: A Source Book in Mathematical Logic 1879-1931.* San Jose: toExcel.