Proof of Church's Thesis

Ramón Casares

ORCID: 0000-0003-4973-3128

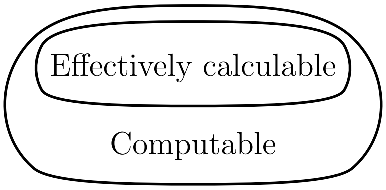
We prove that if our calculating capability is that of a universal Turing machine with a finite tape, then Church's thesis is true. This way we accomplish Post (1936) program.

Keywords: Church's thesis, Post (1936) program, Post's law, Turing completeness.

§1 Church's Thesis

¶1 · Church's thesis, also known as Church-Turing thesis, says, see Gandy (1980):

◇ Thesis 1 *What is effectively calculable is computable.*



¶2 · What is *computable* is anything that any Turing machine can compute, where the Turing machine was defined by Turing (1936). There are other definitions of computable, using for example Church's λ-calculus, but all of them are mathematically equivalent. In any case, the term computable is defined with mathematical rigor.

¶3 · However, because what can be effectively calculated was considered a vague notion, it was assumed that Church's thesis could not be formally proved. In what follows, we will make some definitions and assumptions in order to overcome the ambiguity. Then we will deduce Church's thesis from our calculating limitations, which we will postulate are those of a universal Turing machine with a finite tape. In doing so, we are realizing the old program of Post (1936), for whom Church's thesis is neither a definition nor an axiom, but a *natural law* stating "the limitations of the mathematicizing power of [our species *Homo sapiens*]".

This is DOI: 10.6084/m9.figshare.4955501.v4, version 20230321.
Alternative DOI: 10.48550/arXiv.1209.5036.

Any comments on it to papa@ramoncasares.com are welcome.

§2 Syntax Engine

¶1 · *Syntax*, as opposed to semantics, is concerned with transformations of strings of symbols, irrespective of the symbols meanings, but according to a precise and finite set of well-defined rules. We call this set of rules *algorithm*.

¶2 · The previous definition of syntax generalizes the linguistic one by Chomsky (1957): "Syntax is the study of the principles and processes by which sentences are constructed in particular languages" (page 11). A sentence is a string of words, and a word is a particular case of symbol.

¶3 · We will assume here that persons, that is, the members of our own species *Homo sapiens*, have a syntactic capability. If a person speaks a particular language, then she has a syntactic capability according to the linguistic definition. Also, most of mathematics is pure syntax according to the general definition, see Hilbert (1922), and mathematics are produced and consumed by persons.

¶4 · We will call whatever that implements a syntactic capability a *syntax engine*. So, if persons have a syntactic capability, then each person has a syntax engine. In the case of a general-purpose computer, the central processing unit (CPU) is its syntax engine.

§3 Finite Turing Machine

¶1 · Thus, the Turing machine can be seen as a mathematical model for syntax engines, see Chomsky (1959). The only part of a Turing machine that cannot be built physically is its infinite tape. So we will define a *finite Turing machine* as a Turing machine with a finite tape instead of the infinite tape. The tape is just read and write memory.

¶2 · For each finite Turing machine there is one *corresponding Turing machine*, which is the Turing machine that is identical to the finite one, except for the tape. Defining *processor* as the whole Turing machine except its tape, then any finite Turing machine and its corresponding Turing machine have the same processor.

¶3 · If two or more finite Turing machines have the same corresponding Turing machine, then we say that they are *equivalent*. Equivalent finite Turing machines only difference is the length of their tapes; they have all the same processor. And now, the corresponding Turing machine of each equivalence class of finite Turing machines can be seen as its ideal representative.

¶4 · The processor has some finite memory itself to store what Turing (1936) called the "m-configurations". From the processor point of view, this m-configurations memory is internal memory while the tape is external memory. Thus, basically, a processor is a finite-state automaton, a Turing machine is a finite-state automaton attached to an infinite external memory, and a finite Turing machine is a finite-state automaton attached to a finite external memory.

¶5 · Because a universal Turing machine is a Turing machine, see Turing (1936), we can define finite universal Turing machines the same way. A *finite universal Turing machine* is a universal Turing machine, but with a finite tape instead of the infinite tape. The corresponding Turing machine of a finite universal Turing machine is a universal Turing machine; both have the same processor. Nevertheless, as noted below in §4, finite universal Turing machines are not universal, because they are finite.

¶6 · We will say that a computing device is *Turing complete* if and only if its corresponding Turing machine is a universal Turing machine. In other words, 'Turing complete device' is synonymous with 'finite universal Turing machine'. However, although they are formally equivalent, we would rather use the former to qualify real devices and the latter to refer to the mathematical concept.

§4 Effective Computation

¶1 · We will call any computation done by a finite Turing machine in a finite time an *effective computation*. In principle, effective computations are physically achievable, since they do not require unbounded resources.

¶2 · Finite Turing machines will fail on those computations that require more tape than they have available. Also, in practical terms, and particularly if the tape is long, time available could be exhausted before reaching a tape end or a HALT instruction, and then the computation would have to be aborted, failing. These computations that fail in the finite Turing machine because of a lack of tape or a lack of time would continue in its corresponding Turing machine, and they will eventually succeed, or not. Summarizing: computing is more successful than effectively computing.

¶3 · But, how much successful? A computation is *successful* when the Turing machine has reached a HALT instruction in a finite time. And, whenever a computation has reached a HALT instruction in a finite time, the Turing machine has only had time to inspect a finite number of tape cells. This means that any successful computation could be done by some finite Turing machine in a finite time. So the answer is: not too much.

¶4 · In addition, each finite Turing machine computation that HALTs will be run identically by its corresponding Turing machine, because no limitations of tape or time were found, and there are not any other differences between the two machines. So any computation done by a finite Turing machine will be identically computed by its corresponding Turing machine. Summarizing: an effective computation is a computation.

¶5 · The *effective computation proposition* is longer, and it is also mathematically precise: *save for time or tape limitations, every finite Turing machine computes exactly as its corresponding Turing machine.*

¶6 · A finite universal Turing machine is not a universal finite Turing machine, that is, finite universal Turing machines are not universal because, taking any finite universal Turing machine, there will always be some computations that fail in it, but that do not fail in other equivalent finite Turing machines that have more tape or more time; more time perhaps because they are faster or more durable.

¶7 · On the other hand, any finite universal Turing machine computation that HALTs will be run identically by its corresponding Turing machine, which is a universal Turing machine. And this means that, save for time or tape limitations, every finite universal Turing machine computes exactly as its corresponding universal Turing machine.

¶8 · This also means that every Turing complete device can calculate, save for time or external memory limitations, whatever any Turing machine can compute. There are actually real Turing complete devices. For instance, every general-purpose computer CPU is a Turing complete device, by design.

§5 Calculability

¶1 · We will assume the following thesis:

◇ Thesis 2 *What is effectively calculable is what a person can calculate.*

¶2 · This could be denied from a Platonist view of mathematics, because a super person, or a super machine with a super syntax engine, could calculate what a plain person cannot. But, firstly, we should say that this would be "super calculable", not just "calculable", and even less "effectively calculable".

¶3 · And, secondly, if those super calculations are outside our plain syntactic capabilities, then we could neither identify those super calculations as calculations, nor we could follow nor understand them. For seeing this, just imagine any finished calculation. If each and every step of the calculation obeys the finite set of rules for the calculation, that is, if each and every step obeys the algorithm, then we have a plain calculation that we can follow and understand. Or else, the steps that do not obey the algorithm are errors, and then the calculation is wrong, even if its final result is right. So, for us, a super calculation can only be a plain calculation or a wrong calculation.

¶4 · Another objection could be that different persons can have different calculating capabilities. This is true, but to investigate this issue, let us present our own thesis:

◇ Thesis 3 *Persons' syntax engine is a finite universal Turing machine.*

A simpler way to express this is to say that *we are Turing complete*.

¶5 · If this is the case, then, for normal persons, that is, for persons without mental disabilities, the difference can only be the amount of memory or time. As a consequence, given access to enough memory, everybody has the same calculating capability, though the amount of time needed to perform any particular calculation would not be the same for everyone.

§6 Proof

¶1 · We will now show that if thesis 3 is true, then Church's thesis (thesis 1) is also true.

¶2 · First, we take thesis 1 in §1 and, using thesis 2 in §5, we replace 'what is effectively calculable' with 'what a person can calculate'. Second, using the concept of syntax engine, seen in §2, what a person can calculate is what her syntax engine can calculate. Now, if thesis 3 in §5 is true, that is, if persons' syntax engine is a finite universal Turing machine, then what a person's syntax engine can calculate is what a finite universal Turing machine can compute. A finite universal Turing machine is a finite Turing machine, as seen in §3, and anything that a finite Turing machine can compute can also be computed by its corresponding Turing machine, as shown in §4. To close this proof just remember the first definition in §1: anything that a Turing machine can compute is computable. QED

§7 Discussion

¶1 · From the proof it is easy to see that it is also possible to imply Church's thesis from a weaker assumption:

◇ Thesis 4 *Persons' syntax engine is a finite Turing machine.*

But this thesis 4 does not fit some empirical facts because, if thesis 4 were true, but thesis 3 were false, then some computations would be outside the syntactic capability of persons. This would mean that we persons could not follow some Turing machine computations.

¶2 · On the other hand, the stronger thesis 3 implies thesis 4, and it is very close to the converse of Church's thesis. The converse of Church's thesis would be:

◇ Thesis 5 *What is computable is effectively calculable.*

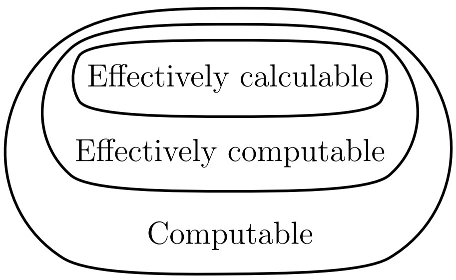
For this thesis 5 to be true, a person's syntax engine should have to be a universal Turing machine, but then persons could calculate any computation, that is, any algorithm with any data. Sadly, we persons are finite, and therefore we have neither access to an infinite memory, nor an infinite amount of time to do calculations.

¶3 · Luckily, if thesis 3 is true, meaning that we persons are Turing complete, then we can calculate any computation, except for limitations of memory or time. So, taking a small enough part of any computation, we can calculate whether it is computed rightly or wrongly.

§8 Conclusions

¶1 · We can summarize thesis 2 and thesis 3 in a refined Church's thesis:

◇ Thesis 6 *What is effectively calculable is effectively computable.*



¶2 · Save for time or memory limitations, we can calculate whatever any Turing machine can compute. In other words, we are Turing complete. Turing could not have designed his universal machine if his own syntax engine were not a finite universal Turing machine. Turing was Turing complete!

¶3 · In §6 we have shown that, if we are Turing complete, then Church's thesis is true. By contraposition, this implies that if Church's thesis were false, then we would not be Turing complete; that is, we would not be *just* Turing complete, because then we could perform some super calculations that no Turing machine can compute.

¶4 · Therefore, the *natural law* stating "the limitations of the mathematicizing power of [our species *Homo sapiens*]" can be formulated as:

◇ Post's Law *In calculating capability, we are just Turing complete.*

This way Post (1936) program is accomplished.

§9 Epilogue on Post (1936) Program

¶1 · Church (1937) reviewed Post (1936) program unfavorably. For Church, his thesis gives a definition of effective calculability as computability that seems "to be an adequate representation of the ordinary notion", and that is all. However, even if that were the case, considering Church's thesis a definition does not settle the question, but just sweep it under the carpet. For suppose two possibilities:

A) Tomorrow someone devises a procedure to perform calculations that are beyond the capability of any Turing machine.

B) Tomorrow some machine is found that performs calculations that are beyond the capability of any Turing machine.

In either case, Church's definition would be wrong.

¶2 · So it is not that effective calculability does not have an exact definition, and that equating it to computability solves the problem, as Church (1937) argued. What happens is that Church's thesis is uncertain because it depends on what it might occur tomorrow, showing that it is empirically refutable.

¶3 · For Post (1936) program, effective calculability is defined exactly (by thesis 2, see §5), and then Church's thesis is not a definition, but a *natural law* that states a limitation of the calculating capability of our species *Homo sapiens* (our thesis 3, see §5, reformulated in §8, for which thesis 1, see §1, is a consequence, see §6). Therefore, Post's law is our thesis 3 as expressed in §8 and, as any *natural law*, it is refutable and then it is in need of continual verification. It is refutable because it predicts both the impossibility of case A, and that we are blind to case B.

¶4 · As in our argument in favor of thesis 2, see §5, we are blind to case B if our calculating capability is limited to computing, because then we cannot see the super calculation as a proper calculation. This is possibly the case of quantum systems. Take for example a Schrödinger's cat in its box. If we suppose that the box is a physical device that performs a calculation the result of which is a dead or an alive cat, then we have to conclude that no algorithm is obeyed, but that some random activity happens.

¶5 · Post (1936) program is deep. As he concludes, in page 105:

> *Only so, [that is, only if Church's thesis is a* natural law *stating the limitations of the mathematicizing power of our species* Homo sapiens*], can Gödel's theorem concerning the incompleteness of symbolic logics of a certain general type and Church's results on the recursive unsolvability of certain problems be transformed into conclusions concerning all symbolic logics and all methods of solvability.*

Figures

References

Chomsky (1957): Noam Chomsky, *Syntactic Structures*; The Hague, Mouton & Co., 1957, 2002. ISBN: 3-11-017279-8.

Chomsky (1959): Noam Chomsky, "On Certain Formal Properties of Grammars"; in *Information and Control*, vol. 2, no. 2, pp. 137–167, June 1959, DOI: 10.1016/S0019-9958(59)90362-6.

Church (1937): Alonzo Church, "Review of Post (1936)"; in *The Journal of Symbolic Logic*, Volume 2, Issue 1, p. 43, March 1937. DOI: 10.1017/S0022481200039591.

Gandy (1980): Robin Gandy, "Church's Thesis and Principles for Mechanisms"; DOI: 10.1016/s0049-237x(08)71257-6. In *The Kleene Symposium* (editors: J. Barwise, H.J. Keisler & K. Kunen), Volume 101 of Studies in Logic and the Foundations of Mathematics; North-Holland, Amsterdam, 1980, pp. 123–148; ISBN: 0-444-85345-6.

Hilbert (1922): David Hilbert, „Neubegründung der Mathematik: Erste Mitteilung"; in *Abhandlungen aus dem Seminar der Hamburgischen Universität*, Volume 1, Issue 1, pp. 157–177, December 1922; DOI: 10.1007/bf02940589. Spanish translation by L.F. Segura as "La Nueva Fundamentación de las Matemáticas", in *Fundamentos de las Matemáticas*, (editors: C. Álvarez & L.F. Segura), Servicios Editoriales de la Facultad de Ciencias, UNAM, México, 1993, pp. 37–62; ISBN: 968-36-3275-0. English translation by W.B. Ewald as "The New Grounding of Mathematics: First Report", in *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*, Volume 2, (editor: W.B. Ewald), Oxford University Press, Oxford, 1996, pp. 1115–1134; ISBN: 0-19-850536-1.

Post (1936): Emil L. Post, "Finite Combinatory Processes — Formulation 1"; in *The Journal of Symbolic Logic*, Volume 1, Number 3, pp. 103–105, September 1936, DOI: 10.2307/2269031. Received October 7, 1936.

Turing (1936): A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem"; in *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937, DOI: 10.1112/plms/s2-42.1.230. Received 28 May, 1936. Read 12 November, 1936.