The Impossible Machine Theorems

Turing-Kleene-Rice

What is this title? The Far Future

There were two fundamental mathematical theorems in the twentieth century!
Neither of them is named in the conventional math text books nor in Wikipedia.
The first was discovered by Zermelo but melted into his Well Ordering Theorem.
The second double theorems melted into Rice Theorem and the Rice Shapiro Theorem.
These “melting-in”-s are intentional confusions by Formalism to prevent clear understanding.
I will not elaborate more on this evil tendency because it is part of the bigger question why social intentionality exists at all. And only a very far future will answer this question.

A Historical Introduction

But a historical start is very possible. Since I want to keep it short I give now only a map.
This word I used was not accidental because I believe in a fundamental trio of didactics that has the word Map for overviewed knowledge, while Road and Garden for detailed descriptions.
The crucial point is the distinction of these two. The Roads are the must without which understanding is impossible. These are not even knowledge fragments rather abilities.
Walking, talking, reading, writing, counting, multiplying are the six fundamental Roads.
Only the last was a bit controversial as some believed that the multiplication table does not have to be learned blindly rather should be discovered. They were wrong!
At the same time, Larichev’s word problem collection proved that a Seventh Road exists too.
And indeed, the word problems is the single easy road into mathematics.
Not surprisingly, they threw out this crystal clear method to make math hard again for everybody!
But I promised not to lament about the evils of Formalism so I should start my historical sketch.
This starts with Geometry that was actually for a long time the road into mathematics.
For me it was too because I entered a math High School where my teacher started with this.
In the four years I won many competitions and never solved a word problem.
When I started to tutor to get some money while attending uni, did I only encounter Larichev’s ingenious book and realized that this shorter road exists. It’s not accidental that this algebraic road came in the twentieth century. To say that Geometry is over rated is cruel but true.
Of course Geometry will live forever and I will exactly show now why.
Euclid made geometrical derivations and constructions. These two went hand in hand.
The claim that two points determine a line meant also the action of drawing a line with our ruler through two already obtained points. This duality of proofs and constructions also revealed that certain assumptions about lines are special. Namely, the assumptions about parallelity!
We can draw parallel to a base line through an outside point by two methods.
Finding an other outside point same distanced from the base line and then connecting them.
Or connecting the outside point to any point of the base line, measure the angle and copy it to the connector’s other side.
The assumption behind the first method is that the line we obtained will have all of its points same distanced and so we could say that the parallel line is the set of points having same distances from the base line. Even if we can prove our assumption a certain error lies in the conclusion.
The other method has its problem too. Namely, the free choice of the connector.
The “truth” is that these two methods define a same fix line, “the parallel” with the base line through any outside point. But things become even more interesting because we could define this same fix line also as the single one through the given outside point that doesn’t cross the base line.
There are elaborate provable connections between the three approaches and so Euclid already wondered what could be the minimal assumption that guarantees the identity of the three.
He chose a pretty awkward one but in the eighteenth century abstraction won over naivety and so John Playfair’s form was born claiming that “There is only one non crossing line with a base line through an outside point”. Observe that the claim is only the “only”.
Indeed, for the angle copying method we can “easily” show that it is non crossing as follows:
We should regard the M middle point of the connector and then realize that:
Any P point of the base line mirrored to M will be a Q point of the other line.

If the two lines were crossing in a point then it would have to be its own mirroring.
Which would mean a point having a distance from itself.
The new abstract form of the parallelity axiom did not put to rest the unhappiness about it.
By this time of course everybody accepted that very probably there has to be an axiom because the simpler axioms can not imply such. More importantly, the earlier attempts to find a derivation went by assuming the negative of a form of the parallelity axiom and try to get to a contradiction. This is called indirectness and it is a basic form of mathematical reasoning. Once we get such contradiction then it's always easy to convert the argument to a derivation from the other used conditions. By the beginning of the nineteenth century, the search was for the negative of Playfair's form, that is to get a contradiction from more non crossing lines with a base line.
But as Bolyai put it: From nothing I created a new world. So instead of a contradiction a new Geometry showed itself. His father was classmate of Gauss in uni and so he sent his results to him who had a very rude reply. Yet true. Namely, that he also realized those consequences long ago.
A few decades later the idea of a non Euclidean geometry became accepted because within the normal space so called models were found where the unique parallelity I not true.
What is a model? It is actually the essence of human intelligence.
If we show a kid the rules of chess in the morning and in the afternoon we walk out to a park where one of those giant chess boards are with heavy movable pieces, he will at once be able to play there too. Most importantly, we don't even notice the instant identification of the new model.
Animals are unable to do anything like this. They live in the reality of the particular model.
Well, it's sad to say but Gauss and Bolyai were living in the singular reality of space too.
They were even thinking about whether light should be the line and so we could verify things by examining light rays. This actually later became indeed relevant through Relativity but first math had to realize its own superiority. Strangely, even after making models in real space, they didn't realize something even more fundamental. Indeed, these were very smart constructions where the lines could be replaced by circles or line segments where of course the distances are distorted too.
And yet the coin didn't drop till Beltrami realized that "we have solved what we started from".
Indeed, the contradiction searches were to find a derivability of a parallelity axiom from the simpler axioms. But this is impossible because in the weird models where there is no singular non crossing line to a base line the other simpler axioms are all true. Two "lines" cross in a single point, two points determine a single "line", and so on. Now since these are models, every logical necessity should be true in them and so if there were a derivation of the single non crossing line from the simple axioms then it would be true in them contradicting what we can see in them.
So with Beltrami's awakening, actually New Math was born. Logic, Set Theory was conceived.
But we must return to Euclid because there was an ancient twist that has to be followed again.
In spite of the ugly form of Euclid's original parallelity axiom he realized the most practical form.
It is what we all learned in Elementary School, that the sum of angles in a triangle is 180 degree.
Two anti didactical failings of Formalism are attached to this crucial "knowledge".
The first is how insanely they prove it by drawing a parallel line to its base through the corner on the top. Then we see indeed the two base angles copied next to the third already there.
So indeed, the three angles form a single line that is 180 degree. BUT!
Just few more sentences and they could enlighten the little minds toward something much deeper!
Namely, we should draw not a naively parallel rather copy the two base angles on their own.
So we could then repeat our earlier argument for both being non crossing the base.
And then say that if we assume the singularity of the non crossing line then voila these two are the same lines and so indeed get the single line and 180 degree.
The other missed opportunity about this law is the Physical importance of it. Namely, that a non Euclidian space with light rays as lines can not be detected locally. If the light rays are curved then looking at the edge of a table, it will look straight. Only giant triangles of three stars looked from each with theodolites, measuring the angles and communicating them will reveal the truth.
But back to the Greeks, they used this fact of the 180 total for a very important particular goal.
Namely, to find similar smaller triangles. These proved for some initial triangles that their sides can not have a common unit. Or in other words, that the proportion of their sides can not be a fraction. So distances have lengths that can not be always compared as fractions.

Nowadays it's quite plausible that if we take a random stick and measure it with a tape measure then it won't be a perfect length by our marks. Using a micro meter we could refine the length. But eventually a microscope would still refine this and so on.
So a random distance's length is an infinite decimal. How could the Greeks not see this?
Well they had no decimals. But we still didn't see something that we could!
A crucial failing in Elementary School education is coming again!
We all learn the digit by digit calculations of addition, subtraction, multiplication and division.
But then here at the last we are not told some amazing consequences.
The first is that the used remainders that we continue with the next brought down digits, can only be finite many kind, namely less than the divider. Once we run out of the digits of the number we divide, we bring down always zeroes and so the remainders must return and thus the resulting digits too. So we always get infinite decimals where a period is repeating.
Some better teachers mention this. But then comes the direct consequence:
A fraction changed into an infinite decimal is always periodical.
So distances in general are obviously not fractions!
But can we make examples of infinite decimals that are surely not periodical?
We could say just pick the digits randomly and they shouldn't be such. But can we be concrete?
This far they never go in Elementary Schools and they should!
We are almost at the historical introduction of our subject.
The ridiculously over complicated tricks of the Greeks to show non fractional distances by similar triangles seems an ancient naivety by looking at the infinite decimals as distances.
If we draw attention of course to the periodicalness of the fractions!
But a new naivety happened then in the nineteenth century.
Instead of the fractions as special numbers a new class was regarded.
Those that are roots of equations with using only the basic operations and whole numbers.
These were called as algebraic numbers which is understandable. But those that are maybe not such were called as transcendental. The square root of 2 is obviously algebraic because it is the root of the equation: $x^2 = 2$. But whether π the area of a unit circle is algebraic is doubtful.
So we entered a new naivety of finding transcendental numbers for sure.
This was much harder than the Greeks' similar triangles.
And then came Cantor the discoverer of the concept of sets the biggest abstraction of mankind.
To appreciate how he instantly proved that there are plenty of transcendental numbers and that actually these are not "transcendental" at all, rather the majority of numbers, we should follow his new argument for the fractions. Namely, why these can not be all the infinite decimals.
The proof falls in two parts. First we show that the set of all decimals are not a single sequence.
Then we'll prove that the fractions are a single sequence.
We'll only regard both of these numbers under 1.
So the $0.d_1 d_2 \ldots$ decimals and the $\frac{a}{b}$ fractions with $0 \leq a < b$.
For the non sequencability of all decimals enough to create for any sequence of decimals a decimal not in the sequence. So imagine a sequence of them under each other:

0 . 2 0 1 9 3 0 5 5 1 . . .

0 . 5 8 4 0 8 1 7 5 0 . . .

0 . 0 4 3 2 7 9 8 7 4 . . .

0 . 7 0 6 3 6 0 5 5 1 . . .

.

The diagonal decimal of this sequence is : $D = 0.2833\ldots$ The name says it all!
We used the diagonal digits, the first from the first, the second from the second and so on.
It would be very strange if this decimal were in our list but nothing forbids it.

Now comes the trick! We alter our digits to anything else or to be specific adding 1 to each and for 9 meaning the 0 digit as result. So our anti-diagonal decimal is: $D^{+1} = 0 . 3 9 4 4 . . .$ This now can not be in our list for sure because the first digit is not the first in the first decimal, the second is not the second in the second, and so on!

Now comes the second part, sequencing the fractions:
$\frac{0}{1}, \frac{0}{2}, \frac{1}{2}, \frac{0}{3}, \frac{1}{3}, \frac{2}{3}, \frac{0}{4}, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \frac{0}{5}, \frac{1}{5}, . . .$
Quite confusingly, we repeated the 0 numeratored fractions in every group but even if we just omitted these, we would still have repeated values. We could of course leave out all earlier valued ones and so get a sequence of only the different valued ones.
Both arguments are extremely simple and yet shocking. No wonder Cantor had plenty of enemies.
But we must admit that we left a little error in our argument.
The repetitions in the fractions were not an error just a strange side effect but if we had repetitions among the decimals then we had an actual error because then the values could be sequenced.
This “glitch” is luckily minor because the value of two decimals are smaller or bigger according to their first non identical digits except that the all 9 ending ones are equal to the same placed 0-s plus the previous digit increased by 1. In fact, $0 . 9 9 9 . . = 1$ and so is not even replacable.
But these little errors can be taken care easily.
And now comes the crucial post script.
How all this relates to the mentioned difficulties of showing transcendental numbers.
This is the part that should have made his enemies turn around and say wow!
We’ll show that the algebraic numbers are also just a single sequence.
For the equations themselves we can first realize that they can be all combined and reduced to zero on the right, that is regarded as whole coefficient polynomials.
Since an n-th order such polynomial equation has only n many possible roots, it’s enough to list the possible equations and then insert the possible roots as groups.
So we can add all positive coefficients, all negative ones and the order of the highest exponent.
Having such total then it is again only a finite number of combinations that can come about.
So actually, we can list all of them by this increasing total.

A giant leap would be then to regard all those decimals that are obtainable by some effective rules.
Since the effective rules would have to be given by rule tables and these again be listable by the increasing complexities, we would get that all the effective decimals are a single sequence too.
So the fractions, the algebraic numbers and all other even more elaborate special numbers were at once included and turn out to be just mere sequences of decimals.
This grand idea is sound and yet incorrect because effectivity as such has a fundamental twist.
The immediate problem is that we didn’t specify the word “obtainable”!
The most obvious specifying would be to generate the consecutive digits of a decimal by rules.
But an instant generalization of this would be to regard the natural numbers as inputs, namely as the positions of digits and then by some rules get the required digit as result.
We could argue that this method is not really an effective one since we can not try out all places and so a strict generation of the digits is impossible. But then a new grand idea could come in.
It is called “dove-tailing” and means to process all numbers in increasing order but not for the full process only for a single step. So we start the input 1 by making only one step of our process then do one step on 2. Then comes the trick that we don’t step to 3 rather return to 1 and make one more step on its processing. Then do the same for 2 and only then do we start 3.
The idea is clear, we always revisit all earlier numbers! Eventually, all numbers will be processed up to arbitrary large number of steps and so we’ll get all possible digits of our effective decimal.
The big difference is that this method doesn’t give the digits consecutively in increasing order of the places. The place 1000000 might be fully processed and giving the one millionth digit way before finishing place 1 , that is getting the first digit.
This brings in an even more crucial generalization! Namely, allowing that our digit determination by processing a place number would not yield an answer in finite steps at all!

So then, such processes would not always even determine a decimal.
Strangely, the essence behind this generalization comes out by first making a specialization by regarding the decimals in binary form, that is allowing only the digits 0 and 1.
Do realize at once that such binary sequence is actually a separation of the set of natural numbers.
The places of the 0-s is one set and the places of the 1-s is its complement.
Now comes the preparation of the "essence" by regarding those binaries where either the 0-s or the 1-s will all be determined in finite steps.
And now the "essence" is the surprise that this choice is not symmetrical.
So even though either the places of the 0-s or the 1-s both determine a binary, the finite determination of one of them does not imply a necessary finite determination of the other!
Even more precisely: A binary can have effective 0 places and yet non effective 1 places and vice versa. Or with our previous observation of the binaries as separation of the naturals:
A set of the naturals can be effective, yet its complement be non effective.
And this is indeed, the essence of all that will follow about effectivities.
But that's only my use of the word effective for these most general finitely determinable sets.
Some do not use it at all and some use it for the previously mentioned narrower one where both the 0-s and the 1-s are effectively determinable. I will say for these that they are dually effective, while the opposing nomenclature obviously says for my effective that it is semi effective!
Unfortunately, the even stupider original naming of it was recursively enumerable sets.
Indeed, recursive just referred to a simpler method for getting functions but the range as collection soon revealed itself as having wider possible sets than the domain. This provoked a generalization of the effectivity of functions by allowing only partial functions that are not defined for all input values as a temporary step. Indeed, then among these obviously wider class of partial functions we "accidently" get fully defined ones that could not be obtained earlier. This messy way of getting a supposedly final class of effective functions was obviously not convincing as "the" effectivity!
So Kleene who invented this method never even claimed this. Strangely, this theoretically so messy road became the most fruitful. Church arrived at the same possible collection sets but by a straight out generation method. This lambda calculus is ugly as hell, yet he was the first to claim that an objective effective collection method has been found. Gödel was obviously not happy.
But the Church Thesis as the claim of such objective entity was accepted. Then came Turing.
His machine approach was not as convincing at the start either as now they claim it.
But eventually the new name became Church Turing Thesis and even Gödel admitted that the machine approach is the best. Which of course doesn't mean that it can be regarded as a definition of Effectivity. Such does not exists! Only diverse frameworks that somehow all lead to the same final collections. Observe that in theory, there could have been an approach that contains all other possible frameworks as its particular cases. The philosophical question is why such doesn't exist.
A last morbid detail about the effectivity nomenclature is Soare's success in introducing the word computable as a replacement of the old recursive that as I mentioned Kleene widened.
Soare's admiration for Turing is sweet and we also must admit that Turing himself used the word computable in this manner and in my view illogically. His article's title started as Computable Numbers With Relation To . . . We might think that he meant natural numbers that are collected by his new computational effectivity, but not! He meant real numbers, that is decimals or binaries and used the word computable as the dual effectivity, that is machine determinability of both places. So the crucial half sidedness of effectivity was thus hidden by the word computable just as earlier by the word recursive.
Forgetting these less important though not accidental confusions, we must mention finally a very strange formal coincidence! An opposite of the binarization of the decimals would be to widen the possible ten digits to allow any m many symbols of an alphabet, that is regard infinite texts,
Such infinite texts are still not fully exploited, yet strangely, this idea but restricted to only finite texts became the "best" input concept for Turing's machines. Even more strangely, we must admit that this finite text as input concept is not that crisp as the concept of Turing Tables that actually determine the machines.
On a blog I once found a question relating to this input confusion of the Turing machines.
The smart ass monkeys and parrots of course at once silenced the truly smart enquirer.

My usage of the word Effectivity has two validations. One is in the future when this article will connect to the other about the f Widening Theorem. This will mean that random sets are grasped! But there is an other validation very much in the present! It is didactical.
The simple truth is that Effectivity is one sided! "Usually" if one side of a separation is effective then the other can not be! The full elaboration of this "usually" is actually this article.
But there is an immediate result reflecting this too! Namely, that those effectivities that happen to be dual must be "hidden"! We can not effectively list them. Due to anti-diagonality again!
Indeed, we show now that for any effective list of dually effective separations, there exists one that is not in the list:
Make two containers for numbers! An L left and an R right one.
Look at the first separation in our list and check which side contains 1.
Put a 1 into the opposite of our two containers! Then look at the second separation.
Now check where 2 lies. Put a 2 again in our opposite container.
Continuing this way we get a separation that can not be in our list!
And yet if the list was effective then this new separation is dually effective.

Visions

I already tried to give the bigger picture but kept it in its historical frame.
Now I show how simple and beautiful the even bigger picture is.
As I explained, the real turning point was Beltrami's recognition that the already discovered models had to obey logical necessities. And this happened when Logic was not even born yet! Neither the concept of Sets. But that's how history works, through dialectical necessitations.
This is how concepts ripen. Then they are born formally and a morbid reversal becomes the "straight" way to define the earlier inexact usages. So today we say that sets can define structures and structures can be models of an axiom system. And most importantly, the earlier naïve axioms and derivable theorems also became formalized through an also ancient recognition.
Strangely, this recognition was not made by Euclid, rather Aristotle.
The reason for this absurdity that mathematics reclaimed something that should've been its from the start is still not quite clear. But that Aristotle's formal logic is fishy became clear to even the philosophers. Then Newton warned Physics in a motto to stay away from metaphysics altogether. Hegel smartly replied that "thus" Physics should stay away from thinking. Which of course didn't mean that he thought that Newton didn't think. By the way, Hegel also warned against formal logic and that he didn't see deeper, is understandable by his missing the essence of math as such.
But Gauss the greatest mathematician ever didn't realize either that formal logic possesses the hidden treasure that would be the solution to his very obsession, the precision of derivations.
The magic wand was still missing that could rejuvenate formal logic.
And this magic wand is again a never emphasized or rather intentionally unemphasized point.
It is the use of variables that math gradually and only very reluctantly gained possession of.
This is a mystery but the miracle is itself how the use of variables is an a priori God given ability.
In the twentieth century this became so evident that we do not even see it!
Something again that sharply differentiate us from the animals.
I explained how the rules of chess are at once transferred in a child's mind to all models.
This is a more delicate instantaneous ability that we shouldn't tinker with just let come alive!
Larichev's word problems did exactly this. But how does this relate to Aristotle?
He recognized that in our thinking some trivial steps use the "every" and "exist" assumptions.
Seemingly two very different ones but strangely, forming a pair through the concept of negation.
Indeed, the "not every" means that there has to exist some that is not such and the "not exists" means that every that exists is not such. We could continue this to some deep existential thoughts, but Aristotle instead used it to show the everyday consequences, like if there is white horse then it can not be true that all horses are black, and so on. The problem is that we use these so called quantified deductions so trivially, that drawing attention to them seems useless.
What was not realized by the mathematicians way up to and tragically including Gauss too, was that actually all mathematical claims use these two quantors. But the magic wand was missing!

So the claims that there is white horse or every horse is black should not make a connection between the colors and the horses. Yet our conventional grammatical education supports this because horse is a noun while the color is an adjective. But stepping above a bit we can realize that being a horse is exactly just like being black or white an "adjective" if we regard every possible object.
The every possible object is then a variable x , y , a , b , whatever letter we choose.
So the correct claims should be formed as: there is an x so that x is horse and x is white.
The "and" connective stepped in too ringing the bell that then all the similar "or" , "then" and so on connectors can play their roles.
So now if H(x) is being horse, W(x) is being white, B(x) is being black and we abbreviate the exists some as $\exists$ while the every as $\forall$ then our claims are:

$\exists x$ (H(x) and W(x)) or $\forall x$ (H(x) then B(x)). And our relation by the negation is that:

$\exists x$ (H(x) and W(x)) then not $\forall x$ (H(x) then B(x)).

Introducing abbreviations for the and as $\wedge$ for the then as $\rightarrow$ even for the not as $\neg$:

$\exists x (H(x) \wedge W(x)) \rightarrow \neg \forall x (H(x) \rightarrow B(x))$.

This then should be derivable by the assumption that: $\forall x (W(x) \rightarrow \neg B(x))$. And so:

$\forall x (W(x) \rightarrow \neg B(x)) \rightarrow [\exists x (H(x) \wedge W(x)) \rightarrow \neg \forall x (H(x) \rightarrow B(x))]$ is a

logical necessity! Mathematical Logic just came alive in front of our eyes!
I call this strange quantification of everyday language as Grammatics and decades ago suggested to introduce it as the eighth Road in Elementary Schools, parallel to the word problems.
So the golden road to mathematics is right here right now but it's better to make people hate math and let the strange exceptions get only through who then become good slaves for something.
What this something is I don't know yet unfortunately.
The rules of Logic soon became recognized and with Cantor's set concept mathematics entered its new dawn. But sets confronting Logic as this new participant had to reveal something strange.
Something hidden like the quantors were but much more subtle.
As we saw, from Cantor's anti-diagonality the difference of infinities arose.
The points or decimals are a bigger infinity than the simple sequence of the naturals.
But the same idea can be repeated and so it turns out that the possible subsets of the set of all decimals is an even bigger infinity. The trivial question is whether this anti-diagonal infinity jump hides other infinites in-between. So as simplest case, if there is a set of decimals that is not sequencable but less than all the possible decimals. Cantor believed in this and it became called the Continuum Hypothesis. Paul Cohen proved that it is not decidable from the axioms of sets.
Axioms of sets? When did that happen? Well that was the strange thing that I mentioned above.
The confrontation of sets with Logic. Something that Cantor could not witness.
Zermelo recognized it but its necessitation started way before. In fact, we can link it to the already so emphasized Beltrami awakening that models should obey the rules of Logic.
As the rules of Logic were recognized and sets or structures were properly used as the models, this "should" became a natural consequence. But only few recognized that a new, even deeper relation "should" be true too. Namely, a Logic should not only be correct but "complete", that is being able to derive everything that is true in all the models where the assumptions are true.
So in a negative version, if something is not derivable logically then it is because in some models it is false. So a proof of this completeness of Logic would require to find models.
But when all this awakened, the fundamental tool to create such models were not yet recognized.
It only came through Zermelo's new realization of something very logical about sets themselves.
The building of sets were envisioned as collections. This is what effectivity targets too.
But theoretical set collection is not restricted to collect the members by rules.

So the natural idea is to collect them by some $P(x)$ property.
Unfortunately, if P talks about the membership in general then a similar anti-diagonal situation comes about that now leads to a contradiction. Indeed, let $\{ x ; P(x) \}$ be the set collected by P.
If $P(x) = x \notin x$ that is x not being a member of itself then: $\{ x ; x \notin x \}$ can neither be member or not member of itself. Indeed, assuming that:
$x \in \{ x ; P(x) \} \leftrightarrow P(x)$, that is: $x \in \{ x ; P(x) \} \rightarrow P(x)$ and $x \notin \{ x ; P(x) \} \rightarrow \neg P(x)$, then:
$\{ x ; x \notin x \} \in \{ x ; x \notin x \} \rightarrow \{ x ; x \notin x \} \notin \{ x ; x \notin x \}$ but also:
$\{ x ; x \notin x \} \notin \{ x ; x \notin x \} \rightarrow \{ x ; x \notin x \} \notin \{ x ; x \notin x \}$. So both assumptions imply the other.
This paradox was the main reason that the deeper one waited for Zermelo to be realized.
The solution of this one eventually meant that we can not collect all properties and instead must restrict what collections are allowed that would presumably not lead to contradiction.

Now imagine an S set of points on a line with the single restriction that there is no right most point. That is, for every P point in S there is an other in S that is right to it.
Prove that there is a $P_0, P_1, P_2, \ldots$ sequence of points in S that goes rightward.
We could say: Pick a point P_0, then one right to it P_1, then one right to it again, and so on.
What is the problem here? The "pick a point" or the "and so on" ? The truth is that their togetherness! It describes such rightward going sequence but doesn't really prove the existence!
To feel that a better solution can exist, observe that if we could prove that an infinite C subset of S exists so that for every P in C there are only finite many points in C that are left to P, then we truly proved an existence of our previous sequence. Indeed, then first of all, there has to be a left most point in C, that is one which has no point to its left and this is okay because the "nothing" is finite. But more importantly, by assigning as subscript to every P member of C the number of C elements left to P, we at once get the desired right going sequence in full.
Observe that a sequence is actually a function and so our better, immediate presentation of the desired function went by finding a set. Strangely, the perfect solution that guarantees such set will use again functions. Namely, requiring a new axiom, the Axiom Of Choice about functions:
For any F function having $F(x)$ values that are all non empty sets, there exists a $c(x)$ function, defined on the same sets as F but with $c(x) \in F(x)$. So c is a sample function from F.
The application for our previous problem goes like this.
For every x point in S let $F(x)$ be the set of points in S right to x.
By our condition this F has only non empty sets as $F(x)$ values.
So by the Axiom Of Choice a $c(x)$ exists that for every x in S, $c(x)$ is a point right to x.
Now pick a point from S say P_0.
Let's call a finite set of points a (P_0, c) set if P_0 is a member and for every other P member: $P = c(x)$ with x being left to P and no point between them.
The only one element (P_0, c) set is $\{P_0\}$ the set having only P_0 as member.
The single two element one is $\{P_0, c(P_0)\}$. And so on!
But these three sentences were just a visualization.
What however is a provable fact is that the C combining of all the (P_0, c) sets is exactly as the C we used earlier. So:
C is infinite. C is a subset of S. For every P in C there are only finite many left to it in C.
And most importantly, we can see at once that C exists!
So it is not an empty set and so the previous three properties not just describe some phantom.
Indeed, P_0 is a member of it, because $\{P_0\}$ is a subset of it because it is a (P_0, c) set.
To fully appreciate all this, you must read the twin article, titled:
The f-widening Theorem. But now more about Effectivity.
The big vision of structures as reality and Logic connecting it to descriptions can be changed to Effectivity being regarded as the connection of altering reality by some rules.
The rules operate on a finite part of reality but not on a fix part due to the continual changes made exactly by the alterations. Games are a perfect example.
We have fix rules for allowed steps and the reality is the continually altering situation.

An initial input is for example the dealt cards and from then our choices alter this initial situation.
In chess, the initial input is always the same but the other player's move are still micro inputs.
We can imagine a one person game and even one where the steps are not free rather also predetermined in the rules.
The main duality is the rule table, the inside of the machine and the continually altering structure as the outside. A very different and refined picture to the naïve vision where the whole machine is an inside. Real computers have only finite memory which again helps to miss the point.
The Turing machine uses an infinite memory line and so has a potentially arbitrary large memory.
Only this makes it clear that this set of arbitrary long texts is its outside universe.
The world of infinite texts is a jump even further and as I mentioned, not yet explored.
The inside feeling will come about if we imagine how the applied rules form a sequence without even looking at the changing structure.
We only see incoming memory contents as applicable micro inputs to which we reply by actions.
But these micro inputs were our own makings, except from the initial input.
So looking back to our action past plus the initial input, we will find every micro input.
In other words, this inner determination is self-contained! A bit more complex than the simple rule table but it can happen without the outside structure, "without vision".
If you are frightened by this bleak scenario then you should be indeed!
Because our material universe is a version of such inner darkness.

Texts, Alterations, Transformers

We have a fix alphabet that can include any symbols even the blank or space.
The finite sequences made from these will be called as texts.
The fundamental idea is what we learned in Elementary School already.
We read letter by letter. We write letter by letter. And we even learned to calculate letter by letter.
We meant here the digit by digit calculation methods where the texts are base ten numbers.
Using squared pages, our math teachers showed us how to calculate the addition, subtraction, multiplication and even division of two numbers. The methods are a bit different in Europe and in the US. But the essence is the same. We write the new calculated digits in new squares and so actually there is no alteration. In fact, we were told that if we make a mistake then just cross it over and write the corrected one next to it. So no eraser was allowed.
The new digits were obtained by calculating single digit operations in our head.
So obviously these methods assumed that we already know additions and the times-table.

Can we avoid the squared paper? Seems like a stupid question, yet this is the start of everything.
To begin with, how could we then have the two numbers that we have to operate from.
We can not just write them after each other because then they melt into a number from which we could have no way of telling what they were. The simplest solution is of course a separation symbol say | that we'll use in our new approach too but with a different meaning.
Now that we are walking into a seemingly stupid restriction of not using the plane only a single line, we can walk into an even more stupid sub restriction whether we can achieve a separation without a new symbol. But the amazing answer for both restrictions is that: Yes we can.
For the bigger linear restriction this is much harder to see but for this less important one we can see it at once. Indeed, suppose we know that a number or for that matter, any text actually contains two texts without a separation marker. Can we get the two from one?
The idea could be called as the stuttering method.
We double every symbol in the first text while we do not in the second.
If you are a fast thinking person you might say at once that there is a glitch. Namely, the start of a non repeated symbol should be the first in the second text but what if the second text happens to start with a repeated symbol. Luckily, the solution to avoid this glitch is easy.
Indeed, we can place any dummy non repeated symbol as separator. So this first non repeating symbol must simply be ignored and after that starts the second text as it is without repeats.

As an example, lets combine apple and cider, so form the apple|cider dual text without using |.
The solution: apple|cider=aappppllleeacider. The single a was the actual separator that avoided |.
So if someone knows this trick then he can perfectly get the two texts from the combined.
There is a deeper problem of course, as always. If someone just gets texts then how should he know whether a coded double or a normal single text is coming. Our results will not reply to this.
So we have a buried problem. And buried problems always resurface! Always! I have no clue when this one will and so I will lie and continue as if nothing would have happened.
So let's continue our insane approach of using texts on a single line. Having the two numbers that we must for example add digit by digit is still very far from being solved.
As you remember addition starts from the last digits. We somehow must locate these, add them as single digits, write the result somewhere else, plus remember the overflow 1 if the two were more than ten together. Writing the results somewhere else is of course already a remembering and whether these rememberings must be additional writings in our single line or can be solved by our method internally is still not clear. The simple fact is that all this is impossible!
Our feeling is that we can not avoid some kind of helping place, like an other line.
Unbelievably, this is not the real cause of impossibility. We simply need an eraser!
This is unbelievable because exactly our aim to keep the two numbers intact seemed to be the problem! And indeed, we must do that yet we need to destroy and replace digits.
The modern solution of an eraser is the memory cell. We can rewrite it as many times we want to.
So we must use infinite many memory cells next to each other, in short a memory line.
I still don't want to go into how a machine or rule table could solve the calculations on such line.
Instead, I want to give you the big picture.
The fundamental duality that will emerge is whether an alteration sequence is finishing or not.
Addition, subtraction or multiplication were trivially finishing methods. Once we used up all digits we also finished the result. But division was different because we strived for a decimal as result and when the number being divided finished we brought in zeroes and continued.
Better teachers explained that the possible remainders being only finite many, the result must become repeating too. So then we could stop once we get a full repeating segment there.
This makes it clear that a process doesn't necessarily must finish. But it also suggests something that was the crucial mistake before Turing entered the scene. Namely, that we must give an external condition about the altering texts that should decide a finish. Turing's heuristic step was to make this stop or as he called it the halt command by the machine, that is the rule table itself.
For the addition, subtraction, multiplication this is not so surprising because we see that the data, the given numbers finished by reading an empty space. But at division only the examination of the alteration itself can show that a full repeating segment is already there. So Turing's idea seems very stupid. Yet the really big picture is that the seemingly logical alteration examinations are not only an unnecessary extra baggage on the fix methods but can not really grasp the crucial non finishing processes. And that was the reason why Number Theory did not raise this question.
A typical Number Theory problem is to tell if a number is prime. But this is not a real problem at all in our new vision. In finite time we can try out all smaller numbers as potential dividers, so tell if neither worked and so our number is a prime. A never finishing situation would have to try out all numbers not just the smaller ones. And this is not so absurd at all. I could claim something about the number 100 that requires to try out all numbers. Then of course, it is the existence of such and such number above 100 that counts as the basic question. And then the generalization means that for any number we ask this same question which simply means whether there are infinite many such numbers. So we have plenty of such questions but they all became Number Theoretical problems rather than processes. Are there infinite many twin primes, and so on.
A much more exciting class of Number Theoretical problems claim something even more drastic than an infinity of examples, namely that something is true for all numbers, that is the opposite is never true. But actually even these split in two class. Some like Fermat's Last Theorem claim a surprising hidden correlation, while others like the Goldbach Conjecture claim a statistically logical tendency to become perfect.
From a machine's view point all these become irrelevant and yet very relevant!
A conventional machine only knows what to look for and trying out numbers.

Such conventional machine is stupid because if Number Theory proves that such numbers do not exist then it will try out numbers forever unnecessarily.
Turing Machines are smartly stupid. Indeed, since all conventional machines can be translated into these, they will also run forever for such searches. But they know some machines that must run forever without success that Number Theory doesn't know about.
Number Theory could get insulted by this statement and reply as follows:
I know for a fact, that whatever set of numbers a machine can collect by any decisions, I can define too. In fact, I can derive from my axioms for any particular number if it is inside.
But can you derive also for a particular number if it is not inside? Well, … There you go!
As we see, the crucial point became not the infinity of cases rather the simpler all or nothing.
Can we collect a set of numbers by some rules exactly, so collect those and only those numbers.
We of course omitted the oppositeness of another collection as part of the challenge and this makes the whole problem much wider. But this is a useful widening because it raises the question whether in theory could all number sets be collected by machines. The negative answer comes without knowing anything about machines only that they can be described by some rules.
If we accept a framework of machines or rule systems that use a fix number of basic actions then every such particular machine or rule table is a finite set of those basic actions. So similarly as our texts are finite segments of symbols, here too we have finite sets of basic actions.
The crucial common consequence is that both the texts from an alphabet and the machines made from basic actions are merely a single sequence. I only show it for the texts and the crucial trick is to establish an alphabetical order. A similar basic action order could be for machines.
Windows altered its symbol order from XP in Windows 7 and so the listing of file names are different now then were earlier. Annoying as hell but irrelevant for us now.
The point is that we can use any accepted alphabetical order and combine it with the length increase, so use a simple length-alphabetical ordering to list all texts:
a , b , . . , z , aa , ab , ac , . . , ba , . . . , za , . . , zz , aaa , aab , . . .
Unbelievable, but all texts will appear.
Using the 0 , 1 alphabet, that is regarding all binary sequences: 0 , 1 , 00 , 01 , 10 , 11 , 000 , . .
These correspond to simple natural numbers and give then: 0 , 1 , 0 , 1 , 2 , 3 , 0 , . . .
As we see the correspondence is not unique and so we get infinite many repetitions.
Nevertheless, we get all numbers appear. The members without the later repetitions are simply the naturals 0 , 1 , 2 , 3 , . . where the sequencability feels trivial.
Now let's get back to the point, that similarly, all machines made in a framework can be listed.
Let's list these not horizontally rather vertically under each other as M_1 , M_2 , . .
But also lets put as a vertical heading all naturals. Finally, line by line a h = halt under the n headings in the M_n row if M_n halts from the n input:

	1	2	3	4	5	6	7	.	.	.
M_1			H			H				
M_2		H			H	H				
M_3		H								
.										

Using the $T\downarrow M$ symbolism for M halting from the T input, our table means that:

$3\downarrow M_1$, $6\downarrow M_1$, . .
$2\downarrow M_2$, $5\downarrow M_2$, $6\downarrow M_2$, . .
$2\downarrow M_3$, . .
.

We will use a much better verbal expression for our symbolism $T\downarrow M$, that indeed follows the symbolism, namely we'll say that T halts in M. And here is another symbolism:
$| M | = \{ T ;\ T\downarrow M \}$ the set of all those texts that halt in M and thus is collected by M.
So our rows above as selected sets are: $| M_n |$.
Later we'll use a refined version of $\downarrow$ if we want to tell at what s step number T halts in M. We'll write $T\downarrow sM$. We'll also use $T\uparrow sM$ if for no $s' \leq s$ did $T\downarrow s'M$ happen, that is up to s T did not halt yet in M. Finally, we'll use $T\uparrow M$ for the negative of $T\downarrow M$ which means that T runs forever in M. Observe the following intuitive facts:
While collecting all those T inputs that halt in M, that is $| M |$ is a logical definition of an effective collection, to collect those T that run forever in M seems quite contrary a non effective collection. Of course we have two problems here.
First of all, there is a certain ineffectivity even in the halting collection because we didn't tell how we should try the possible inputs. Obviously the above described sequencing of all texts helps a bit but then just trying them all out in this order is not a truly effective collection because already the first trial can fail and then we never get to try the others. We'll come back to this problem.
Secondly, the non effectivity of inputs that run forever is not a necessity because we might have a method that establishes this without the actual running forever. We'll come back to this too.
Finally, observe that while $T\uparrow M$ has this problem $T\uparrow sM$ is obviously effective for a given s.
This nuance will be the key in our second Impossible Machine Theorem.
Behind all this formalism something vital had not been emphasized!
We saw above that the binary texts that we listed so smartly, contained all natural numbers.
I just said: Nevertheless, we get all numbers appear. That was the point and the repetitions just a minor detail. These minor details that the teachers usually don't even mention are the buried treasure! Proofs proceed somewhere and actually throw sand in our eyes to avoid asking about these minor details. The kids who seem to be good in math, are simply the ones who can get over easily these little lies, while those who don't know what's going on, actually become nauseated by the successive abstractions and lies. But since they think that something is wrong with them, they build a defensive wall. They should say: I don't get it and try to tell what feels confusing. It would always turn out that they saw something important unexplained. Anyway, back to our situation:
The vertical listing of all M_1 , M_2 , . . . machines must hide an even more astonishing repetition than our earlier binary forms. So same lines must be infinite many times repeating!
Indeed, these are just telling the $| M_n |$ collected numbers by halts but we can obviously make machine variants that halt from same inputs, or as we should say, in which the same inputs halt.
The reason that I could have been lying again and keep this incredible repetition galore hidden is because our goal is again some obsession to prove something, rather than show everything.
But this is a yet unavoidable misery of mathematics. So remember my obsession, my goal was to show that some number collections can not be made by machines at all.
To show this, we listed all machine collections above and so now using nothing about machines, we'll show that the list can not contain all number sets. Or to avoid even this indirectness, we can also say that we'll simply show for any list of number sets that there is a set not listed.
So you can understand that avoiding the repetitions of the machine collections was logical because the whole argument will avoid machines anyway. Any list of collections must be incomplete because a new collection can be made. This missing new collection will be effective in some sense yet not in the sense that machines can collect. So this avoiding of machines has this other strange twist that I should have just keep silent about.
But now that I mentioned that our example will not be effective in the machine sense, I could use an other word for it like say it will be concrete. Very concrete, in fact. We'll choose the members of our claimed missing set by writing y = yes under our heading. Here is a possible such choice:

1	2	3	4	5	6	7	.	.	.
y		y	y			y			

These for example would collect the $\{ 1 , 3 , 4 , 7 , . . . \}$ set of naturals.

The lines in our table use H instead of y because we used Turing machines for them but those halts are exactly the yes decisions for collection and so each has a similar corresponding set.
Our y selection of course could easily be same as one of the H selections.
Cantor was the genius who realized first the amazingly simple concrete way to choose the y-s that excludes this. Just like Turing, he saw something childishly simple that others didn't.
In his case, the name of his idea is an instant give-away! It is called the anti-diagonal argument.
We already explained this idea in the previous historical introduction but for decimals.
Now that we used natural number collections by the y yes symbol, this anti diagonality is even simpler. Indeed we just look at the diagonal places of our list and wherever there is a H we will not place our y but if there is no H we will:

1	2	3	4	5	6	7	.	.	.
	y		y						

Now simply observe that this line:
can not collect the same as the one in the first line since 1 was not in that but is in this.
can not collect the same as the one in the second line since 2 was in that but is not in this.
can not collect the same as the one in the third line since 3 was not in that but is in this.
And so on, it can not collect either of the lines collected by our sequenced machines.
A machine oriented post script to this not machine oriented proof is that we used very correctly the y and not an H for our choosing of the non appearing set.
Bot not only because we claimed that we listed all machine collections by the H-s but also due to how we did choose our y-s.
Indeed, we placed them under those inputs that do not halt in the machine of the line.
And to make so many halts exactly at some non halts of all other machines would be absurd.
Or let's say very improbable! And now a different question that also concerns chances but has nothing to do with machines is this: Could randomly chosen y-s be same as a line?
Strangely, after this proof that the list is definitely not complete, we tend to say no.
But now we return to our original challenge for Number Theory, to make an effective collection so that the complementing set, the numbers outside is definitely not effective, that is not machine collectable. And this was not a meaningless challenge. It lead to recognizing the limitations of Number Theory. Of course a challenger should be able to solve the challenge himself easily.
And Turing Machines can.
The concept of halting as collection criteria is the already exposed main tool.
This means that actually we go against our intuitive start from the elementary school calculations.
There the result was the goal. But now, that we regard all possible mechanically possible text alterations, the final altered text at the halt will be completely ignored. But this is not quite true.
We'll regard a special class of machines where the result is everything. Namely, the machines that halt from any input text. This is very logical. For these always halting machines the halting is sure and so the result is the essence. We even call these as text transformers or just transformers.
Using Greek capitals for these, $T\Theta$ is the transformed text that the Θ transformer produces from the T initial text. This can be used as input for an other normal machine M and so for example $T\Theta \downarrow M$ means that the T text transformed by Θ halts in M.
The conventional notation for the halting arrow is not as we introduced it rather $TM\downarrow$.
Observe that using this convention, our transformer usage would be ambiguous.
Indeed, $T\Theta M\downarrow$ could mean that T is the input for a ΘM machine. Luckily, with our notation this becomes a meaningful general law. Indeed, such ΘM new machine becomes meaningful, namely as the one where the Θ transformer is built into M as a starter.
And thus in our notation this can be expressed as: $T\Theta \downarrow M \;\leftrightarrow\; T\downarrow \Theta M$.

The Duplicator and Universal machines

There will be two transformers needed for our crucial theorem that was our title too.
The simpler one is a machine that turns any T text into T|T. So we call it the Δ duplicator.
Of course, it is not simply duplicating T , rather places a stuttered version and a non stuttered separating symbol in front of T. But the short claim is of course: $T\Delta = T|T$.
The previous claim about transformers built into M machines will now mean for our particular duplicator and an M_0 machine that:

1. $T|T \downarrow M_0 \quad \leftrightarrow \quad T\Delta \downarrow M_0 \quad \leftrightarrow \quad T \downarrow \Delta M_0$.

The more important second necessary player in our first result is not a transformer at all, rather a very “halting conscious” machine that Turing called as universal. In fact, this was the essence of his article, to show that such universal machines do exist. These are machines that can imitate any other machine. The meaning of imitation reflects our previous point that we ignore results.
But even the alterations will be ignored and we merely require that such U universal machine should be able to collect the same S inputs as any M machine. Which of course is meaningless because all M machines collect different inputs by their haltings.
So we use the M machine also as a translated second S = <M> text.
By translated, we mean that from the alphabet of the machine actions and buildups we go to the same alphabet as the input texts.
So the universal U machine must have two inputs combined by our earlier explained T|S method, that is T being stuttered and S as a normal following a single non stuttered symbol.
T is an input for any M machine that U gets as the S = <M> second text as program.
And then the imitation as same collection means that:

2. $T|\langle M\rangle \downarrow U \quad \leftrightarrow \quad T \downarrow M$.

So U must first separate the T and <M> texts which means to avoid the stuttering in T|<M>.
Then put aside this true T and start its real business to use <M> as a program.
Namely, as a program to imitate what M would do with T.

Computers are all universal. Even the ancient ones using Basic. But for a primitive system as Turing’s, using only a single memory line and cell by cell alterations it is very surprising.
There are many such “primitive” systems that can imitate in some sense all computers.
But the real magic of Turing’s system is that it visibly can do that!
And the secret of this visibility is the use of “halt” right at the bottom, that is in the rule tables.
A reverse concept of the <M> program is the [T] machine defined by any T text.
This is simply the M machine for which T = <M> if there is such, or a chosen “empty” machine that never halts and thus collects nothing, if T is not a program, that is there is no such M.
Of course, there are real never halting and thus non collecting machines too but this ambiguity is true in general. There are different machines that halt from exactly the same inputs and so collect same text sets. We’ll call them variants, reflecting our previous idea of simulation.
Back to universality, it can be explained with the reverse concept as: $T|S \downarrow U \quad \leftrightarrow \quad T \downarrow [\,S\,]$.

Both the program concept and the machine defined by a text are obviously relative to a U universal machine and so when we use these concepts we always assume a fix U.
Now we can combine our two above results:
Applying 1. in reverse for M_0 as U , T as <M> and finally 2. with T as <M> :

$$\langle M\rangle \downarrow \Delta U \quad \leftrightarrow \quad \langle M\rangle \Delta \downarrow U \quad \leftrightarrow \quad \langle M\rangle|\langle M\rangle \downarrow U \quad \leftrightarrow \quad \langle M\rangle \downarrow M.$$

Nothing special, unless we observe that the input is the same in the two ends.
And so the $\langle M\rangle$ input either halts both in ΔU and M or it doesn’t halt in both.
In a sense, we can say that these two machines “share” the $\langle M\rangle$ input.

To make this "sharing" concept more important, first let's define it in general:
Two M , M' machines share the T input if $T\downarrow M \leftrightarrow T\downarrow M'$.

Now let's call two **T** , **T'** text sets complementing if they are exact opposites, that is one contains exactly the texts that are not in the other. And then we can denote **T'** also as $\neg$**T**.
Similarly, we call two M , M' machines complementing if the texts collected by them, that is $|M|$, $|M'|$ are complementing. But we shouldn't introduce $\neg$ applied to a machine because:
While a text set always has a complementing one, an M machine not necessarily has a complementing machine. But observe the simplest that:
Sharing and being complementing are exact opposites!
Two M , M' share some input exactly if they are not complementing.
And so to show that an M_0 machine can not have a complementing one, it is enough to show that for any M machine, M_0 and M share some input.
This is exactly what we showed above for $M_0 = \Delta U$. It shares with any M the <M> input.
And so ΔU has no complementing machine, that is $|\Delta U| = |M|$ is impossible. And so:
The texts not collected by ΔU , that is $\neg|\Delta U|$ can not be collected by any machine at all!
Or to say it in an even shorter way, $\neg|\Delta U|$ is uncollectable.
Our above two facts can help to prove other input sharings and thus non complements.
The simplest is for U itself. The shared input with an M is $<\Delta M>|<\Delta M>$. Indeed:

$$<\Delta M>|<\Delta M>\downarrow M \quad \leftrightarrow \quad <\Delta M>\Delta\downarrow M \quad \leftrightarrow \quad <\Delta M>\downarrow \Delta M \quad \leftrightarrow \quad <\Delta M>|<\Delta M>\downarrow U.$$

Without guessing this ad hoc shared input, we could have argued indirectly as well:
Suppose that there were an M machine complementing with U. Then we had:

$$T\downarrow \Delta U \leftrightarrow T\Delta\downarrow U \leftrightarrow T|T\downarrow U \leftrightarrow \neg(T|T\downarrow M) \leftrightarrow \neg(T\Delta\downarrow M) \leftrightarrow \neg(T\downarrow \Delta M).$$

And so ΔM were a machine complementing ΔU which we proved not to be exist.

Parameter Theorem

Universality was able to change the S second text into the [S] machine.
Kleene realized that a similar transfer of an S into any M machine is also possible.
S thus becomes a parameter in M so we actually get an S depending machine function.
But we don't have to introduce new effective machine functions because we have the concepts of programs and defined machines already. So we must use the < M > program of M, combine this with S into the new machine's program and then go back and find the actual machine.
The combining is universal, so if we use the programs and defined machines relative to a fixed U universal machine, then there is also a fix Π transformer that can produce the new machine's program from $S|<M>$.

Parameter Theorem:
There is a Π transformer that for all T , S , M : $T|S\downarrow M \quad \leftrightarrow \quad T\downarrow [\,(S|<M>)\Pi\,]$

The $(S|<M>)\Pi$ text can also be regarded as a transform of merely S by an M depending $\Pi(M)$ transformer, so an other form of the claim is that: $T|S\downarrow M \quad \leftrightarrow \quad T\downarrow [\,S\Pi(M)\,]$.

Kleene then also realized that this single most fundamental method implies some very strange consequences about machines. The key expression is redundancy! We saw such redundancy in our effective listing of the naturals by listing the binary texts and I lamented about the possible redundancy of machines too. Kleene showed that such redundancy is unavoidable!

There was a much earlier sign of all this! I mentioned it in the historical introduction but now I repeat it again. Cantor who discovered the mentioned anti-diagonal method, had another fundamental surprise that concerned the fractions or rationals.
They are dense, if we regard them on the number line. So between any two fraction there is another. Indeed, for example their middle value, that is sum divided by 2.
We might think that then the fractions are an infinity more than the simple sequence of the naturals but less than the infinity of all decimals.
The being less assumption is suggested by what I mentioned about the good Elementary School math teachers. As I said, they should mention that at the division process we always get repeating of the result. This division process actually gives the decimal form of any fraction.
There is a consequence then that even the good teachers never tell! Namely, that this periodicalness of the fractions as decimals at once shows that the fractions can not be all decimals.
Actually this hides two solutions. Firstly, to show a decimal that is not periodical for sure.
Such example could be regarding all numbers after each other increasingly as a decimal.
The second deeper solution is that we should just pick our digits randomly and then it seems impossible to get a perfectly periodical one. So though this second is not as exact as the first, it not only raises the question of randomness but suggests that the fractions should be much less than all the decimals. At any rate, all this was just a detour because Cantor by his second surprise showed that the fractions are merely a single sequence. Now I show it again but for all fractions:

$$\frac{1}{1} \quad , \quad \frac{1}{2}, \frac{2}{1} \quad , \quad \frac{1}{3}, \frac{2}{2}, \frac{3}{1} \quad , \quad \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1} \quad , \quad \frac{1}{5}, \ . \ . \ .$$

As unbelievable it is, we indeed have all possible fractions listed.
But our whole point is not this now, rather the unbelievable redundancy hiding it.
Namely, all fractions appear infinite many times!
All this is not accidental as Kleene proved it in his two ground breaking theorems.
It's amazing that not only the effectivity of infinites has to have redundancies but life needs it too.
Indeed, the twenty amino acids have much more possible nucleic acid codings choosing them.
But Kleene was bewildered by his grand discoveries and so missed something even bigger.
Most amazingly, Turing who hit upon the crucial new fundamental solution to grasp effectivity at once as halt being in a rule system, also missed this bigger thing.
He not because he was bewildered by the details rather by the newer and newer basic ideas.
This new bigger thing is our whole subject. But we also arrived at something much more universal that caused these failures and which has to be spelled out!

Conditional machines, Impossible Machine Theorems

So what is this universal thing that I mentioned as the cause of blindness even for geniuses?
It is the fundamental problem not only with Mathematics but all science.
The continuously emerging trap of formalism that sweeps the unsolved basic questions under the carpet. Indeed, a formalism can not deal with all depths and so we must ignore logical questions.
Sadly, this is exactly what blocks most people from getting into science at all.
So this is not just a problem of science but of human understanding as such.
The kids who are seemingly better in math in school are simply the ones who can step into forms easier and suppress their deeper questions. The ones who feel that the thinking in forms is like a brick wall, are not less talented at all but they can not raise their deeper questions and so start to feel inferior. Then the real stupidity of compensation steps in by the typical nonsense that " I always hated math and were better in humanities" and so on. The false abstraction of course steps in thinking as such too. So true philosophical thinking can not avoid math and science.
This bigger didactical void and the false compensational delusions caused by it is the true fall of Mankind. But now we must sweep under the carpet this most important problem too and instead spell out the two basic facts about machines that were not observed by Kleene nor Turing.

They correspond to two M_0M and $M_0M\uparrow$ machines using $T|S$ double text inputs.

$T|S\downarrow M_0M \quad \leftrightarrow \quad T\downarrow M_0$ and $S\downarrow M$.

$T|S\downarrow M_0M\uparrow \quad \leftrightarrow \quad T\downarrow sM_0$ and $S\uparrow sM$. So T halts in M_0 at an s step and in no $s' \leq s$ step does $S\downarrow s'$ happen. Or quite simply: When T hats in M_0 then S didn't halt in M yet.

A T text is never halting if $[T]$, the machine defined by T is never halting, that is:
$[T]$ halts from no input, that is: $|[T]| = |T|$ is the empty set.
So never halting could be also called as non collecting.
Two M, M' machines are variants if they halt exactly from the same inputs, that is collect the same texts: $S\downarrow M \leftrightarrow S\downarrow M'$ that is: $|M| = |M'|$.
Two T, T' texts are variants if the machines defined by them are variants: $|T| = |T'|$.

First Impossible Machine Theorem:
There is no machine in which all never halting texts halt but all variants of a T_0 do not halt.
We'll show that such N would imply complementing machine for all M machines.
That is: If for an N machine, for all E, V texts:
If $[E]$ does not halt from any input then N halts from E. And:
If V is a variant of T_0 then N does not halt from V. Or even more precisely: If

$T\uparrow[E]$ for all $T \quad \rightarrow \quad E\downarrow N$ and
$(T\downarrow[T_0] \leftrightarrow T\downarrow[V])$ for all $T \quad \rightarrow \quad V\uparrow N$.

Then for any M machine there were a complementing one.

Let $[T_0]$ be M_0 and we'll use our first M_0M double input machine.

We'll use the following two features about it :

$S\uparrow M \quad \rightarrow \quad T|S\uparrow M_0M$ for all T. And
$S\downarrow M \quad \rightarrow \quad T\downarrow M_0 \leftrightarrow T|S\downarrow M_0M$ for all T.

Using for these, first the Parameter Theorem then the corresponding N features:

$S\uparrow M \quad \rightarrow \quad T|S\uparrow M_0M$ for all $T \quad \rightarrow \quad T\uparrow[S\Pi(M_0M)])$ for all $T \quad \rightarrow$
$S\Pi(M_0M)\downarrow N \quad \rightarrow \quad S\downarrow \Pi(M_0M)N$.

$S\downarrow M \quad \rightarrow \quad T\downarrow M_0 \leftrightarrow T|S\downarrow M_0M$ for all $T \quad \rightarrow \quad T\downarrow M_0 \leftrightarrow T\downarrow[S\Pi(M_0M)]$ for all $T \quad \rightarrow$
$S\Pi(M_0M)\uparrow N \quad \rightarrow \quad S\uparrow \Pi(M_0M)N$.

And thus the complement machine of M would be: $\Pi(M_0M)N$.
Second Impossible Machine Theorem:
There is no machine that all variants of a T_0 halt in it but no F halts in it with $|F|$ being a finite subset of $|T_0|$.

We'll show again that such N would imply complementing machine for all M.
But now we use $M_0M\uparrow$. Remember that a $T|S$ halts in it if at T halting in M_0 the S is still not halting in M. But more importantly:
$S\uparrow M \rightarrow T|S\downarrow M_0M\uparrow \quad \leftrightarrow \quad T\downarrow M_0$. And
$S\downarrow M \rightarrow T|S\downarrow M_0M\uparrow \quad \leftrightarrow \quad T\downarrow M_0^S$. Where M_0^S denotes the restriction of M_0 to inputs that halt in less steps than $S\downarrow M$ needs. Observing that this is an F and using $M_0 = [T_0]$:

$S{\uparrow}M \;\to\; T|S{\downarrow}M_0M{\uparrow} \;\leftrightarrow\; T{\downarrow}\,M_0 = T{\downarrow}[\,S\Pi(\,M_0M{\uparrow}\,)\,] \;\leftrightarrow\; T{\downarrow}[\,T_0\,]) \;\to$
$|\,S\Pi(\,M_0M{\uparrow}\,)\,| = |\,T_0| \;\to\; S\Pi(\,M_0M{\uparrow}\,){\downarrow}N \;\to\; S{\downarrow}\,\Pi(\,M_0M{\uparrow}\,)\,N.$

$S{\downarrow}M \;\to\; T|S{\downarrow}M_0M{\uparrow} \;\leftrightarrow\; T{\downarrow}M_0^{\,S} = T{\downarrow}[\,S\Pi(\,M_0M{\uparrow}\,)\,] \;\leftrightarrow\; T{\downarrow}\,M_0^{\,S}\,) \;\to$
$|\,S\Pi(\,M_0M{\uparrow}\,)\,| = |\,F\,| \;\to\; S\Pi(\,M_0M{\uparrow}\,){\uparrow}N \;\to\; S{\uparrow}\,\Pi(\,M_0M{\uparrow}\,)\,N.$

And thus the complementing machine of M would be $\Pi(\,M_0M{\uparrow}\,)\,N$.

Observe that if we have a Φ transformer that gives as $S\Phi$ two possible programs.
Namely for an $S{\uparrow}M$ simply T_0 while for an $S{\downarrow}M$ the $T_0^{\,S}$ program of the $[\,T_0\,]^S$ machine, then we again have a complementing machine of M as ΦN:
$S{\uparrow}M \;\to\; |\,S\Phi\,| = |\,T_0| \;\to\; S\Phi{\downarrow}N \;\to\; S{\downarrow}\,\Phi N$. And
$S{\downarrow}M \;\to\; |\,S\Phi\,| = |\,T_0^{\,S}| = |\,F\,| \;\to\; S\Phi{\uparrow}N \;\to\; S{\uparrow}\,\Phi N$.
But we can not simply define Φ this way since nothing guaranties it to be effective!
We can define an f text function of course as: $f(S) = T_0$ if $S{\uparrow}M$ and $f(S) = T_0^{\,S}$ if $S{\downarrow}M$.
But then we can also prove that this f can not be an effective Φ transformer if M has no complementing machine. Indeed, suppose such Φ and let $(\Phi = T_0)$ be the machine defined from Φ that halts only when the result of Φ is T_0. Then
$S{\uparrow}(\Phi = T_0) \;\to\; S\Phi \neq T_0 \;\to\; f(S) = T_0^{\,S} \;\to\; S{\downarrow}M$. And
$S{\downarrow}(\Phi = T_0) \;\to\; S\Phi = T_0 \;\to\; f(S) = T_0 \;\to\; S{\uparrow}M$.
So this $(\Phi = T_0)$ were a complement of M that does not exist and so Φ does not exist either.
So instead of a proof we got a vicious cycle.

And yet the exact proof could seemingly apply a similar direct definition:
$T|S{\downarrow}M_0M{\uparrow} \;\leftrightarrow\; (\,S{\uparrow}M$ and $T{\downarrow}[\,T_0\,]\,)$ or $(\,S{\downarrow}M$ and $T{\downarrow}[\,T_0^{\,S}\,]\,)$.
The big difference is that we used the T input as well!
Indeed, now this definition is simply collecting the same $T|S$ texts as our original definition.

Rice Shapiro Theorem

This theorem gives the two almost perfect necessary conditions for a variant complete **T** text set to be an effective $|\,M\,|$. Or in other sense, for an $|\,M\,|$ collection to be variant complete.
The almost perfect means that there is a reversal with the assumption that some effective set of finite collectors are inside. I will not go into this reversal.

The two conditions correspond to the two impossible machines.
They say that for every $T \in M$:

1. For all S : $|\,S\,| \supseteq |\,T\,| \;\to\; S \in |\,M\,|$ too.
2. There is some F that : $|\,F\,|$ is finite , $|\,F\,| \subseteq |\,T\,|$, $|\,F\,| \in |\,M\,|$ too.

2. is trivial by the Second Impossible Machine Theorem so we only must show 1.

This goes indirectly by first creating an M_T and then showing that if there is an S with:
$|\,S\,| \supseteq |\,T\,|$ and $S \notin |\,M\,|$ then M_T is an impossible machine of the first kind.
For any S text regard the $[\,S\,] \vee [\,T\,]$ machine that halts if $[\,S\,]$ or $[\,T\,]$ halts.
Intuitively, we just let them run parallel and either halts we halt.

If its program $< [S] \vee [T] >$ halts in M then that means to be a variant of a member in $| M |$.
If $| S | \supseteq | T |$ then of course this implies that S must be also a variant of that member.
Also observe that if $| S | \subseteq | T |$ then $< [S] \vee [T] >$ is automatically a variant of T.
And this includes the case if $| S |$ is the empty set that is S is a never halting program.
Thus the $\{ S ; < [S] \vee [T] > \downarrow M \} = | M_T |$ effective collection contains all never halting texts but can not contain any variants of an S that $| S | \supseteq | T |$ and $S \notin | M |$.
And so indeed M_T were an impossible machine if such S would exist.

In the followings I will not use this beautiful combined consequence of the two Impossible Machine Theorems. Instead, go into the heuristic consequence of the first one. Even later when I show some impossible effective variant complete sets, I will use concrete new arguments.

Consequences of the First Impossible Machine Theorem.

The direct consequence is what was discovered by Rice and preceded the Rice Shapiro Theorem.
The heuristic consequences of this consequence were not emphasized at once. Yet in my view, this became the most important insight into the essence of the program concept since Turing.
Lately, this became corrected by more and more text books but history itself was not examined.
This is striking because Kleene about the same time as Rice presented his proof, just finished his Introduction To Metamathematics that became the Bible of New Math.
So the “Bible” missed the most important insight into the essence of programs.

The concept of variant texts was a crucial feature in our proofs but in the first impossible theorem this was directly visible as condition only in one side, the M_0 variants not halting in N.
Of course, the other side, containing all never halting texts also meant that all variants of the never halting texts are there.
More importantly, in both sides, these variant sets were merely subsets among otherwise non restricted text sets. Rice’s Theorem, the direct consequence and then the heuristic consequences even more, rely on the variantness as being actually a whole side in a new sense. Indeed:
The condition of the singular M_0 variants can be hugely generalized by requiring that for any T that does not halt in N we should have the same for all T’ variant of T.
Trivially, yet quite amazingly then the other side is automatically such variant complete set too.
That is, again with any T that halts in N any T’ variant must too.
Indeed, if this T’ were not there then it were in the other side and then T would have to be too.
So quite simply, a variant complete set is actually a variant complete separation of all texts.
Now we might jump to the conclusion that then our simple consequence is that no N machine can have such variant complete separation of the texts as the texts halting and not halting in N.
But this is false for two reasons:
Firstly, because by our generalization we forgot about something that was trivial before.
Namely, that we have an M_0 with all variants in one side and we have never halting texts in the other. And so neither of the two sides is empty. This is crucial because the trivial separation that has all texts as one side and the empty set as the other, has an also trivial dual collection.
Secondly, because now we don’t know which side has the never halting texts.
Thus we can only be sure that both sides can not be machine collected.
So our correct special consequence is:

Rice’s Theorem:
In a non trivial variant complete separation of all texts, both sides can not be collected.
And collected of course means collected by some machines.
Either only one side or neither. So at least one side must be uncollectable.

To prove this, suppose there were two machines collecting the two sides. Let's check which side contains the never halting texts and which of our two machines collects this side.
Then this machine is exactly an impossible N machine because we can just pick any T text in the other side (since it is not empty) and the $[T] = M_0$ machine is the concrete machine in the First Impossible Machine Theorem.
The mentioned heuristic new insight into the program concept is an application of the case when one side is collectable and so we know that the other side must be not.
In fact, we'll find a whole class of M machines that collect the variants automatically too and thus produce collectable sets with uncollectable complements for every choice of M.
The idea for this, is an extra ingenuity beyond Rice's Theorem. And it could be best approached by a question that relates it to a crucial deficiency of Number Theory. Namely, why couldn't Number Theory show such number collections with uncollectable complements?
Now, what does Number Theory do? Tries to define number sets by arithmetical means.
Using numbers as dead numbers. But numbers, just as all texts, are actually programs for a chosen universal machine too. So the big idea is to collect not dead data, rather programs.
Namely, use texts as programs to collect them! And then the variantness falls into our lap.
All we have to do is watch something not in the texts as programs, rather actually watch how they run as collectors. Indeed, variantness is same collecting so then we automatically get that.
But this raises a deep problem that we already mentioned when introduced our halting collection.
In order to actually get a collected text set, we must try out all possible texts as inputs.
We showed how this is a single list and so we can go through them, but feeding them to a machine one by one is useless. What if the first text already does not halt in our machine? We can't wait forever to try out the second. The solution is the so called dove-tailing. Do a little now and then return later to do more. Actually a very good advice for everything. We have the infinite line as infinite memory so we can easily put aside the partial results. So we start with the first text. Do a few steps on it then start the second. Then return to the first do a few step, then on the second too and start the third. Then return again and now do few steps on all three, then start the fourth text.
When a text gets to a halt in our original machine then this dove-tailing machine should not halt just give a signal. Our dove-tailing machine will run forever and actually generate the texts that would have been collected by the original machine. So we turned a collector into a generator.
And now the variant complete collection of programs is easy:
We make this generator version run and watch something that is observable in finite time about the generated, that is already collected inputs.
The most trivial such beginning observable feature is whether a given text occurs.
With using numbers as texts, if for example 100 will be generated.
So we regard all possible T numbers, regard them as programs for our chosen U universal machine, change them into number generators and just wait for 100 to pop up.
This process that we just described is an M machine that collects the T numbers that as programs collect 100. Most importantly, M will collect all variants of such T-s too.
So M collects a variant complete **T** set of texts, that is numbers.
This is trivially not empty and neither is the set of all numbers.
And so the $\neg$**T** set, that is the numbers that as programs do not collect the number 100 are an uncollectable set. No machine can collect them.
A more interesting finitely observable feature is whether a prime number will pop up.
And so we obtained that the numbers that as programs do not generate prime numbers is not generable. But this has nothing to do with primes. For any other property like being a square number, the same result is obtained.
A similarly, finitely observable feature is whether a given number of certain numbers will pop up.
Say, whether a T number as program, will collect at least twenty square numbers.
So the negative is again not effective. But if we regard whether exactly twenty square numbers are generated then this is not a finitely observable feature. Simply because for those generators that only generate twenty or less square numbers, we can not be sure forever.
Similarly, a finitely not observable feature is if a T text as program collects all primes.

So while the numbers that as programs collect primes is an effective number set with certainly not effective complement, the numbers that as programs collect all primes is not such.
But there is more! We can actually use a second dove-tailing and list **T**. For example:
All those T texts that as programs collect 100. This will be a single sequence of texts so that all variants are inside too. The complement set of this generated number sequence is not generable.
Indeed, it is uncollectable as a set and if it were generable it would be collectable too.
This was actually a bit nonsensical argument. Because our generation concept was ab ovo from collection anyway. But generation can come from other methods too. Most importantly, from derivations by some rules. Such derivation rules are then freely usable like in mathematical derivations of the theorems from the axioms.
But observe that we can apply our rules in any fix order one by one and then we get a generation in any derivation system. A hidden assumption was of course that the rules and axioms are finite or sequencable in an effective order, that is generable themselves.

A detour "How It Happened"

This generability of the axioms and rules was not emphasized in Gödel's original proof of his Incompleteness Theorem in spite of him using a numbering of the expressions and derivations too.
The numbering was more emphasized as Number Theory being able to talk about itself. Which is true but covered the simpler facts of effectivities. These were only clearly recognized by Turing.
In fact, his crucial recognition was that even these derivabilities or generabilities are misleading.
The point is collectability by halts! This is the essence of being effective! Of course, not to use generations and derivations would be a lie because we have intuitive plausibilities about these too.
In fact, a side paradox of this generability of the derivabilities is something that was not spelled out before Turing though it related to the false expectations that Gödel destroyed by his Incompleteness Theorem. From this you can guess that the false expectation was a completeness which means that all statements can be decided, that is either the statement or its negative is a derivable theorem. A partial encouragement for this expectation was Presburger's results just a year before Gödel's result, showing that Number Theory with using only addition is complete.
This partial Number Theory could be called the Baby Genius Arithmetic.
Indeed, it is baby like since only uses addition, but it is also genius like because uses the infinity of the inductions axioms. Presburger result was even more promising due to some strange facts known about models. Namely, that in spite of the Number Theory axioms seemingly describe a sequence of the naturals, we can add artificial new non standard naturals as members so that the axioms remain valid. So Presburger's result showed that these non standard models of the naturals are merely phantoms because they do not contain any new truths.
Everybody was crossing their fingers, especially Hilbert that for the full Number Theory the same thing could happen. But Gödel showed that using multiplication makes a crucial difference.
From the previous remarks we would think that he found some non standard models with truths that are not true among the normal naturals and so these became the undecidable statements.
Indeed, they are not derivable since otherwise they would have to be true in the normal naturals too, and their negatives are not derivable either because otherwise they were true in the non standard models. But this didn't happen at all!
Instead, he showed that multiplication allows the already mentioned numberings and thus self asserting statements about derivations that imply by a diagonality the non derivabilities.
An effectivity version would say something entirely different.
But to say that "better truth" we first must explore a hidden paradox about the generation of the derivabilities. Namely, it means that the theorems of mathematics are generable by a machine.
The freedom of choices in a derivation system mixing with our knowledge how hard the new derivations can be, makes this seem impossible. But observe the crucial fact that such machine generation of all theorems is totally unpredictable in its lengths. Idiotic overcomplicated consequences are also generated and the short genuine gems may come only after waiting astronomical times. But this seemingly absurd fact being true also suggest that effectivity "rules".

So now the better effectivity version of what Gödel showed is more meaningful:
We claimed earlier that Number Theory is stupid because it collects dead numbers instead of programs. In spite of this being true, multiplication allows collections that are complex even if not apparently. Then if the axioms are rich enough to follow these collections, the derivabilities become complex too. This complexity means that the case by case derivable explicit, that is logically built up properties can imitate or express all effective collections.
As a consequence, any rule systems that we could define by adding new basic symbols and axioms for them, can be defined explicitly. An astonishing consequence of this is that exponentiation can also be defined explicitly with multiplication.
A more important consequence is that the set of all theorems is a generable set with non generable complement. This then at once implies the incompleteness, that is the existence of undecidable statement pair. In fact a crucial nuance emerges too.
This nuance is that completeness is not identical with all statements or their negations being derivable. Simply because this "or" could mean both!
Then of course we had a contradiction in our axiom system.
With assuming that this can not happen which is also called as the consistency of our axioms, then to defy the "or" we only must defy the "either-or". And this makes a huge difference.
Indeed, it allows the most amazing indirect argument of mathematics ever:
Knowing that the expressibilities of our axiom system implies non generable non-theorems plus assuming its consistency, we can prove indirectly that there must be undecidable statement pair.
An undecidable statement means actually a pair of negated statements that neither is a theorem.
As we said, our argument will be indirect so we must assume that there is no undecidable statement and get a contradiction. But what does it mean not to have undecidable statement?
That for every negated statement pair it is not true that neither is theorem.
Which means that at least one of them is a theorem. This means a normal "or".
But by assuming consistency this "or" can only be an "either-or".
So enough to derive a contradiction from this. Now comes the promised huge difference!
Indeed, the either-or derivability of the negated statement pairs plus the generability of the theorems implies at once a generability of the non-theorems too, contradicting our assumption.
But you may ask, what is this "at once implication"? And this simple step is the most amazing.
Well, if we assume the either-or situation for being a theorem for any negated statement pair then quite simply when we derive a theorem we should just negate it. This step is so simple that if the generation of the theorems was okay then this last step added will make again a valid generation.
This argument is like Cantor's original diagonality, an instant mathematical soul awakener.
It took a year for Gödel to prove his Second Incompleteness Theorem that targets the subtle assumption, consistency itself, as being undecidable in the system.
All this might suggest that the axioms of Number Theory are a mysteriously delicate balance of saying enough but not telling too much to become inconsistent.
A perfection causing the incompleteness. But this is totally false! The real perfection is deeper.
Robinson realized that there is a Wisely Minimal Arithmetic too. This uses also multiplication but avoids the geniality of the infinite many induction axioms. The full system was:

1. $x' \neq 0$
2. $x' = y' \rightarrow x = y$
3. $x + 0 = x$
4. $x + y' = (x + y)'$
5. $x \cdot 0 = 0$
6. $x \cdot y' = x + x \cdot y$

7(∞). $[\, P(0) \text{ and } \forall x\, (P(x) \rightarrow P(x'))\,] \rightarrow \forall z\, P(z)$
$P(0)$ is called the 0 condition and $\forall x\, (P(x) \rightarrow P(x'))$ the step condition.

The appearing unquantified open variables of course all mean universality.
The meanings are pretty obvious. A logical question is why a reversal of axiom 2. is not stated.

Simply because it is part of Logic.
Now omitting the $7(\infty)$. induction axioms could seem like a crazy idea because for example the following four statements need this for their proof:

$x' \neq x$
$x \neq 0 \rightarrow \exists y\,(y' = x)$
$x + y = y + x$
$x \cdot y = y \cdot x$

It would be a very useful exercise to see their derivations as I show them in the Effectivity article.
But now the point is that Robinson realized that in a crucial sense it is enough to take the second as a new axiom 7. in place of $7(\infty)$. An even more useful exercise is what I also show there.
Non standard models for the new seven axioms where the other three statements remain false.
The seven axioms by the way can be even combined into a single Q statement.
You may wonder what could be the mentioned "enough" that Robinson realized because by these models it is obvious that his system, that is Q can not prove $x' \neq x$ and the commutativenesses.
You might guess that this "enough" must be the expressibilities of the effective collections and so getting that the non-theorems are again a non generable set and so by the heuristic argument we can get incompleteness. In fact, you might also think that since these three statements are visibly non derivable from Q thus the crucial consistency assumption is now irrelevant because in an inconsistent system everything is derivable. But this "visible" was only through models.
And so the consistency of Q remains not provable in Q itself. Strange enough but it is not the new extra feature that made this Q the claimed wisest minimality. Instead, think about why Q could be relevant at all for the original system of Arithmetic with putting back $7(\infty)$.
Because it showed a completely new complexity meaning of the theorems beyond that the non-theorems are not generable. And this was the single most important second deepening of the effectivity concept beside Rice's Theorem. The secret is in our face!
Instead of the non-theorems we must regard the anti-theorems!
These are the negatives of theorems. So these are the refutable statements or the intentional non-theorems. We can not even contemplate these to become theorems because it would cause inconsistency! Now the theorems and these anti-theorems form a so called inseparable pair.
Meaning that no dually effective pair of sets can contain them. So such inseparable pairs are unfixably antagonistic. Next realize that this inseprability of the theorems and anti-theorems then means that any consistent extension of Q has automatically not dually effective theorems and non-theorems. So if the extension is not only consistent but generable too then the non-theorems are automatically not generable. And so we get our original incompleteness of Arithmetic too!
But let's not forget to emphasize the conceptual novelty behind inseparability.
Indeed, naively we might think that by adding new axioms to an incomplete system like Q we must somehow achieve a completeness by choosing the right new axioms. Not so! We can not balance the increase of power and consistency to a perfection! If we choose new axioms that avoid inconsistency then we can never get to a perfect complete system.
And this again shows the effectivity supremacy over the linguistic self references.
Inseparability is inherent in two disjoint sets that forbids any effective supersets.
But there is even more! The nuance of saying less above by that the theorems and non-theorems can not be both effective in an extension, is actually very useful. Indeed, for example regarding all true statements in the standard model about addition and multiplication we get a system where the theorems and non-theorems are symmetrical and so if one were generable then the other were too.
And so we obtained that the arithmetical truths are not generable. The falsities neither and of course the hidden assumption was that Q is among the truths. This brings us back to similar situations but caused by not inseparability rather the First Impossible Machine Theorem.

Dually Uncollectable Sets

Indeed, the First Impossible Machine Theorem just tells that one side that contains all the never halting variants must be uncollectable. What about the other side? We were able to make them collectable and this was the heuristic consequence that altered our concept of the program.
It definitely altered mine! Realizing why Number Theory was stupid by using dead numbers that could be actually programs. If we use them as programs and run them and collect those that show some "finitely observable running features" then the complement is uncollectable for sure.
But let's forget the big picture now and instead remember that Rice's Theorem came about by avoiding the asymmetry of having all variants of a single M_0 in one side while all the non collecting variants on the other. So we might think that if instead of avoiding the asymmetry we increase it and so we have these alone as one sides, that is regard the two situations that we separate the never halting machines and everything else or the variants of an M_0 and everything else, then these should be trivial situations. Well, they are not so trivial!
The first case is not dually uncollectable so the programs that collect anything is collectable too.
The trick is to use a universal machine and watch not just all possible inputs but programs too.
So in the generation vision we must dove-tail not just the inputs but all input program pairs.
The other extreme, when we have only the variants of a fix M_0 as one side is finally a case of our title but, we need a trick again to see this. But even before this:
A ridiculously special M_0 situation is most important and has to be handled separately.
Let M_0 be the machine that simply just halts at once, regardless any input.
The variants are obviously the always halting machines, that is the transformers.
We'll use a strange new anti-diagonal argument to show that these are uncollectable.
The strange new feature will be to use the results which is not so surprising about transformers.
We alter every Θ transformer's result at $<\Theta>$, that is alter $<\Theta>\Theta$ to anything else. Thus get an assignment of results at $<\Theta>$ for any Θ and being different from what Θ dictates.
A machine that would collect all $<\Theta>$ programs could be expanded to do this assignment as a transformer and so getting a contradiction with it not being among the transformers.
Historically, these transformers were the original goal of Effectivity as number functions.
But by the previous argument it turned out that the set of all possible effective functions can not be effective. Of course, this was more of a theoretical dilemma because the concrete methods of effective function constructions failed too.
Kleene found a solution to both problems. We simply must regard as effective not these earlier envisioned effective functions, rather the effective partial functions that may have undefined places, that is inputs. The original goal then still can be conceived as effective total functions but then they are indeed not an effective subset among the partial ones.
Now we can start the general case, that is show that the variants of any M_0 are uncollectable.
Observe a strange way of saying our claim: The collectors of a collection are uncollectable.
Indeed, the collection is $|M_0|$ and the variants of M_0 are the collectors of this collection.
The trick is first to regard a wider impossible collection. Namely, not just collect the variants, that is the collectors of $|M_0|$ rather all the sub collectors. So all the M machines that $|M| \subseteq |M_0|$.
We include the empty collections as subsets and so these M machines include the never halting machines and so we know at once that this collection is uncollectable if the other side has some variants. To get such is easy by regarding any fix T text outside $|M_0|$ and regard all machines that collect T for sure. Now we can show indirectly our original goal.
Suppose that we had an M^0 machine that collects those machines that are variants of M_0.
Let's regard any M machine and combine it with M_0 running parallel from the same input.
Either halts, we halt. So we combine the collections.
We feed this $M \vee M_0$ machine to M^0. If this halts, that is $M \vee M_0$ is an M_0 variant, it means that $|M| \subseteq |M_0|$. So we obtained an impossible machine. Thus M^0 can not exist.

Observe a crucial detail in our used subset situation.
$|M_0|$ could not be the set of all texts because then the other side had been empty.
This actually means that M_0 could not be an always halting machine, that is a transformer.
And so our new result does not give a new proof for our earlier result about the transformers.

Finally a situation that requires even stronger tricks. We separate the machines according to whether they collect, that is halt from finite or infinite many inputs.
The two side are obviously a variant complete separation and the finite collections of course include the no collections and so this side is obviously uncollectable.
We show that the other is as well.
We regard the generator versions that are also finite or infinite because the generated texts can not repeat. So imagine them under each other and thus giving an infinite table of texts.
We use of course anti-diagonality again.
Add a symbol to the $T_{1,1}$ first text of the first row to get $T_{1,1}{}^{+}$.
Now go in the second row till the first T_{2,k_1} text comes that is longer than $T_{1,1}{}^{+}$.
Increase this again to get $T_{2,k_1}{}^{+}$. Find the first T_{3,k_2} in the third row longer than $T_{2,k_1}{}^{+}$.
Increase this again, and so on. We get an S sequence of increasing long texts.
The first member in S is already longer than the first member of the first row and so the other ones in S will be longer too. So S can not contain the first member of the first row for sure.
Then it will again not contain for sure the second member of the second row because all members from the second member are longer than that and the earlier first member is of course shorter.
And this remains so for all diagonal members of our table. The n-th can not be in S because the n-th or later members of S are longer while the earliers are all shorter.
What about the complement set, the finite generators? The never halting machines, that collect nothing are among these. So by the Impossible Machine Theorem this side is not effective either.
Observe that the only reason our proof worked was that we could create a collected text set that was definitely not the content of any of the listed sequences. To create a new sequence would have not been a proof because we have no way to tell how a collected content becomes generated.
But then you may ask why did we need generation at all? Because from mere collections we would have had to pick the incremented members randomly and then the constructed fast increasing text sequence would have not been an effective one.
So this direction of the generations as being effective was needed.

Something Even More Heuristic

The heuristic consequence of Rice's Theorem that finite collection features of programs are all effective program sets with non effective complements, has an even more heuristic translation.
Staring at a program, a very good programmer can imagine how it will run.
The finitely recognizable generation features of course can be verified by trial too.
But can he tell some finite feature failing too, that would take infinite time to verify?
Sometimes he can. He might just be lucky or have a hunch. But what if he claims a system.
A sure way to tell from some signs in a program if a finite running feature will fail.
That is impossible! So analyzing a program can not predict something that trials can not settle.
There can be no perfect correlation between some program feature and the failing of a finite running feature. Indeed, such correlation would be a C machine that collects programs and they all fail a finite running feature. But the M machine that collects the programs by their finite running feature is one that has no complement and so my obsession, the sharing of inputs can be used as the equivalent of this no complement. Thus M and C must share some P program too.
So then either P halts in both M and C or it doesn't halt in either.
In the first case P is a counter example for the correlation because C collects P and yet the finite running feature will happen. In the second case P is a counter example because C will not collect it and yet the finite running feature will fail.

Appendix 1.: The Kleene Theorems

It would be unfair not to tell the two amazing theorems that Kleene discovered about the necessary redundancies of an effective system. Especially because the second will also give an easy second proof of Rice's Theorem. He used not Turing Machines rather his partial recursive functions as effectivity framework. But his proofs can be easily transferred to Turing Machines as I will do now. He was able to surpass the narrow diagonalization method that uses doubling.
As we saw, even the non complement feature of U came out only indirectly through ΔU.
The idea with hindsight is very simple. We must mix any M machine into the diagonalization.
So the solution falls into two parts. First we must find an involvement of M and then use diagonalization for that involvement. The first part, his Parameter Theorem is very plausible.
And we used it already in the derivations of both of the Impossible Machine Theorems.
But the two consequences that the diagonalization created are very implausible.

Recursion Theorem:
For every M there is an M^U text that for all S : $S|M^U \downarrow M \leftrightarrow S\downarrow[M^U]$.
The M^U notation makes sense since $[M^U]$ is like a "personal" universal machine for M.

Let M^+ be a machine that imitates the halting of $S|(T|T)\Pi \downarrow M$ but using S|T as input.
In other words: $S|(T|T)\Pi \downarrow M \leftrightarrow S|T\downarrow M^+$.
And then let $M^U = (<M^+>|<M^+>)\Pi$. Then indeed: $S|M^U \downarrow M =$
$S|(<M^+>|<M^+>)\Pi \downarrow M \quad \leftrightarrow \quad S|<M^+>\downarrow M^+ \quad \leftrightarrow \quad S\downarrow[(<M^+>|<M^+>)\Pi] = S\downarrow[M^U]$.

The first $\leftrightarrow$ is true by our definition of M^+ with $T = <M^+>$.
The second $\leftrightarrow$ is true by the Parameter Theorem with $T = <M^+>$ and $M = M^+$.

As we said, $[M^U]$ is like a personal universal machine for M. But continuing our result to involve the real universal machine U : $S|M^U \downarrow M \leftrightarrow S\downarrow[M^U] \leftrightarrow S|M^U \downarrow U$.
So any M shares all the $S|M^U$ texts with U. Unfortunately, this still shows only U to be a machine without complement. Amazingly, Rice's Theorem that is a consequence of this theorem can create an abundance of machines without complements. But didactically it's better to derive the following consequence first as an intermediate theorem:
Fix Point Theorem:
For every Θ text transformer there is an F text that $|[F\Theta]| = |[F]|$.
That is, for all S : $S\downarrow[F\Theta] \leftrightarrow S\downarrow[F]$.

There is an M machine that for all S , T texts: $S\downarrow[T\Theta] \leftrightarrow S|T\Theta \downarrow U \leftrightarrow S|T\downarrow M$.
Then for M^U guaranteed by the Recursion Theorem we have that:
For all S : $S\downarrow|M^U\Theta| \leftrightarrow S|M^U \downarrow M \leftrightarrow S\downarrow|M^U|$. So an F is M^U.
The first $\leftrightarrow$ is true by our choice of M with $T = M^U$.
The second $\leftrightarrow$ is true by the Recursion Theorem.

Rice's Theorem:
Let a variant complete separation of all texts be **A** , **B** with neither being empty.
Then they can not be both effective.

Let $A \in \mathbf{A}$, $B \in \mathbf{B}$ and define a Θ text function as: $T \in \mathbf{A} \rightarrow T\Theta = B$, $T \in \mathbf{B} \rightarrow T\Theta = A$.
If both **A** , **B** were effective then this Θ were a text transformer and then by the Fix Point Theorem for some F text we would have that $[F\Theta] = [F]$.
Thus $F\Theta$ and F are variants and so they should be in the same of either **A** or **B**.
But by the definition of Θ we have that $F \in \mathbf{A} \rightarrow F\Theta = B \in \mathbf{B}$ and $F \in \mathbf{B} \rightarrow F\Theta = A \in \mathbf{A}$.

The Recursion Theorem seemed merely as a first step to Rice's Theorem but its name already suggests something on its own. For this we need a little detour.

Historical View

The Parameter and Recursion Theorems were found by Kleene for partial recursive functions.
He defined these as $y = f(x_1, x_2, . , x_n)$ effectively calculable functions but with allowing undefined y values for some x tuples. Instead of Turing's text alterations that naturally can run forever, Kleene introduced a single "search for first possible value of an equality".
Amazingly, this is enough to bring in the crucial leap from an old class of effective total functions, the so called primitive recursive functions.
The functional obsession stems from these, which were built up by functional substitutions.
In truth, the real cause of success in functional approaches is deeper and yet not quite clear.
Identical functions including identity of the arity and this new partial meaning, that is being defined at the same input tuples, is denoted as $f \cong g$.
While set theoretically we regard functions identical according what they assign, now these letters correspond to how they are built by the allowed rules. So $f \cong g$ means more than just two variables used for same functions. It means two buildups being the same partial functions.
The "programs" are now numerical "codes" so $e = \langle f \rangle$ is the enumeration code of an f buildup and in reverse $f = [e]$ is the buildup determined by e.
The crucially different feature as opposed to machines is not merely using numbers instead of texts but what comes with this, the arity of these functions.
Fixing an input variable in an f buildup partial function we get a partial function of n-1 arity.
This is expectable to be built up but what's more, its code can be calculated from the $\langle f \rangle$ code and the fixed value by a fix effective two input total function. This would of course depend on which variable we fix and what was the arity of f. To simplify the situation we only fix x_1.

Parameter Theorem:
For every n there is an $S^n(,)$ two variable effective total function that gives the codes for all the first variable parametrizations of f, using $\langle f \rangle, x_1$ inputs. That is, as $S^n(\langle f \rangle, x_1)$. Thus $[S^n(\langle f \rangle, x_1)]$ are those buildups and so: $f(x_1, x_2, . . , x_n) \cong [S^n(\langle f \rangle, x_1)](x_2, . . , x_n)$.

Recursion Theorem:
For any $f(x_1, x_2, . . , x_n)$ buildup there is a p number that: $f(p, x_2, . . , x_n) \cong [p](x_2, . . , x_n)$.

Let $S^n = S$, $f(S(x_1, x_1), x_2, . . , x_n) = g(x_1, x_2, . . , x_n)$, $\langle g \rangle = c$, $S(c, c) = p$. Then:
$f(p, x_2, . . , x_n) = f(S(c,c), x_2, . . , x_n) = g(c, x_2, . . , x_n) \cong [S(\langle g \rangle, c)](x_2, . . , x_n) =$
$[S(c, c)](x_2, . . , x_n) = [p](x_2, . . , x_n)$.
$\cong$ is the parameter theorem with $S^n = S$, $f = g$, $x_1 = c$.

Fix Point Theorem:
For any t one variable effective total number function there is a p value that:
$[t(p)](x_2, . . . , x_n) \cong [p](x_2, . . . , x_n)$.

As the notation suggests, the trick is to regard a new x_1 variable in place of p.
Of course, for some x_1 values $[t(x_1)]$ may not be a buildup with n-1 arity.
This then just means being undefined for all $x_2, . . . , x_n$ inputs.
This failure is effective from $t(x_1)$ and thus from x_1 too.

For those x_1 values where the arity of $[\,t(x_1)\,]$ is indeed n-1 we can calculate the values for all defined inputs and this means an effective $f(x_1, x_2, .., x_n)$ calculation and thus a buildup. Thus $[\,t(x_1)\,](x_2, ..., x_n) \cong f(x_1, x_2, .., x_n)$ and so regarding the p value guaranteed by the Recursion Theorem for this f we get:
$[\,t(p)\,](x_2, ..., x_n) \cong f(p, x_2, .., x_n) \cong [\,p\,](x_2, ..., x_n)$.

Now we show that in reverse too, the Fix Point Theorem implies the Recursion Theorem:
Let $f(x_1, x_2, .., x_n) = [\,e\,](x_1, x_2, .., x_n) = [\,S^n(e, x_1)\,](x_2, ..., x_n) = [\,t(x_1)\,](x_2, ..., x_n)$.
And let p be the value guaranteed for this t by the Fix Point Theorem. Then:
$f(p, x_2, .., x_n) = [\,e\,](p, x_2, .., x_n) = [\,S^n(p, x_1)\,](x_2, .., x_n) = [\,t(p)\,](x_2, .., x_n) =$
$[\,p\,](x_2, .., x_n)$.

The "recursion" meaning

A form of the Peano rules used for exponentiation too and starting from 1 as first natural are:

$x + 1 = x'$
$x + y' = (x+y)'$

$x \cdot 1 = x$
$x \cdot y' = x + x \cdot y$

$x^1 = x$
$x^{y'} = x \cdot x^y$

A fully relational representation though first seems more complicated, is actually much better.
Already $x' = y$ is replaced by $x \lhd y$ and then of course we must regard all naturals as names.
The "recursive" rules can be formulated as matrix conditions implying the particular relation:

$$\left.\begin{array}{c}(x \lhd z)\ (y = 1)\\ (x + v = w)\ (v \lhd y)\ (w \lhd z)\end{array}\right] \rightarrow\ x + y = z$$

$$\left.\begin{array}{c}(x = z)\ (y = 1)\\ (x \cdot v = w)\ (v \lhd y)\ (x + w = z)\end{array}\right] \rightarrow\ x \cdot y = z$$

$$\left.\begin{array}{c}(x = z)\ (y = 1)\\ (x^v = w)\ (v \lhd y)\ (x \cdot w = z)\end{array}\right] \rightarrow\ x^y = z$$

The lines mean "or" so the different ways to get the target relation after →, while the members in a line are "and" so show all the conditions necessary to get the target.
Now comes the whole point of all this preparation.
We could generalize exponentiation and then that again and again but amazingly, we can in one slash introduce a general all n leveled $x\{n\}y = z$ hyper operation relation with four variables.
The $n = 1$ case is addition, $n = 2$ is multiplication, $n = 3$ is exponentiation and so on.
This actually gives a better understanding of these initial three basic operations too.

Namely, the fundamental rule is quite simple as: $x\{n\}y = x\ \{n-1\}\ (x\{n\}(y-1))$.

Indeed: $x \cdot y = x + x \cdot (y-1)$ and $x^y = x \cdot x^{(y-1)}$.

Of course, it can not work for addition because there is no lower operation there.

Instead observe that:

The $x\ \{1\}\ 1 = z$ relation is $x \lhd z$ so this as condition with $n = 1$ and $y = 1$ is enough.

The $x\ \{n\}\ 1 = z$ relation with $n \neq 1$ is $x = z$ so these as conditions with $y = 1$ is enough.

Thus the final rule system is:

$$\left.\begin{array}{c} (n = 1)\ (y = 1)\ (x \lhd z) \\ (n \neq 1)\ (y = 1)\ (x = z) \\ (x\ \{n\}\ v = w)\ \ (v \lhd y)\ \ (k \lhd n)\ (x\ \{k\}\ w = z) \end{array}\right] \rightarrow \quad x\{n\}y = z$$

A more readable version is:

$$\left.\begin{array}{c} n = 1\ ,\ y = 1\ ,\ z = x + 1 \\ n > 1\ ,\ y = 1\ ,\ z = x \\ x\ \{n\}\ (y-1) = w\ ,\ x\ \{n-1\}\ w = z \end{array}\right] \rightarrow \quad x\{n\}y = z$$

Or the simplest way:

$$x\{1\}1 = x + 1$$

$$x\{n\}1 = x \qquad \text{if } n > 1$$

$$x\{n\}y = x\ \{n-1\}\ [\,x\ \{n\}\ (y-1)\,] \qquad \text{if } n > 1\ ,\ y > 1$$

With using the $h(x, n, y) = x\{n\}y$ hyper operation function:

$$h(x, n, y) = x + 1 \qquad \text{if } n = 1\ ,\ y = 1$$

$$h(x, n, y) = x \qquad \text{if } n > 1\ ,\ y = 1$$

$$h(x, n, y) \cong h[\,x, n-1, h(x, n, y-1)\,] \qquad \text{if } n > 1\ ,\ y > 1$$

The problematic case is the third as definition because it is not explicit, it uses itself.

Using $\cong$ instead of $=$ only made sure that we don't require a function necessarily defined for all variable values. In the case derivation meaning we regard all possible derivations.

But this equation is merely a final requirement about a target that may not even exist.

Now let the used h on the right side be regarded as a hypothetical $[\,e\,](x, n, y)$ and the defined one as a different $f(e, x, n, y)$. Then we have a perfectly explicit definition by cases as:

$$f(e, x, n, y) = x + 1 \qquad \text{if } n = 1\ ,\ y = 1$$

$$f(e, x, n, y) = x \qquad \text{if } n > 1\ ,\ y = 1$$

$$f(e, x, n, y) \cong [\,e\,](x, n-1, [\,e\,](x, n, y-1)) \qquad \text{if } n > 1\ ,\ y > 1$$

By the Recursion Theorem we have a p value of e that $f(p, x, n, y) \cong [\,p\,](x, n, y)$.

Then of course this $[\,p\,](x, n, y)$ will satisfy exactly the equations for $h(x, n, y)$.

So the Recursion Theorem produces a partial function satisfying any single implicit definition.

And such single implicit equation always has a solution.
This is true but remember we only guaranteed partial solutions.
The above generalized operation happens to be total but our result did not show that at all.
It is not hard by other arguments though.
To see how crucial the partialness is for the general claim of the Recursion Theorem, regard the following "absurd" implicit requirement: g (n) ≅ g (n) + 1. It leads to: f (e , n) ≅ [e] (n) + 1.
And not surprisingly, the guaranteed solution f (p , n) = [p] (n) = [p] (n) + 1 is impossible for any n again. But that's okay!
It just means that our "solution" is undefined for every n. Never halts!
There are plenty of different such empty functions and corresponding p values!

Finally, I still lament a bit about the h (x , n , y) = x{n}y hyper operation function.
First of all, the reason I used it as an example to demonstrate that the Recursion Theorem implies it being a partial recursive function, was that the three basic operations would have been trivial examples. Indeed, they are primitive recursive and so were in the class that Kleene expanded.
Quite oppositely, already before Kleene's expansion, h (2 , n , n) = 2 {n} n = h (n) was proven to be a function that outgrows any primitive recursive f (n).
Observe that: x{n}1 = x + 1 or x depending on whether n = 1 or n > 1. Also:
1{n}y = 1 for all n > 2 because first of all 1{n}1 = 1 since n > 1. And then:
1{n}2 = 1{n – 1} (1 {n} (2 – 1)) = 1 {n – 1} (1{n} 1) = 1 {n – 1} 1 = 1 since n – 1 > 1.
1{n}3 = 1{n – 1} (1 {n} (3 – 1)) = 1 {n – 1} (1{n} 2) = 1 {n – 1} 1 = 1 since n – 1 > 1.
And so on we can step up to any y. Finally, observe the even more surprising fact that:
2{n}2 = 2 {n – 1} [2 {n} 1] = 2 {n – 1} 2 = 2 { n – 2} 2 = . . . = 2 {1} 2 = 4.
To see why in spite of this, 2{2}3 can grow incredibly fast, we'll see in the next section.
But now we should just look at h (x , 4 , y) = x{4}y.
To increase the h value with a fix x from y to y + 1 , we must raise x to the old result.

So since x{4}1 = x thus x{4}2 = x^x. Similarly: x{4}3 is $x^{(x^x)} = x^{x^x}$.
This was not a trivial omission of the brackets rather a simple convention. Indeed, an other much smaller value is $(x^x)^x$ but could just as well be abbreviated the same way omitting the brackets.
So exponentiation is not arbitrary in its order unlike addition and multiplication.
And so, for multiplication we can simply say that it is repeated addition and for exponentiation that it is repeated multiplication but for x{4}y we can't just say that it is repeated exponentiation.
An even simpler but usually not quite emphasized fact is that the exchangeability of order already fails for exponentiation. Indeed, $2^3 = 8$ but $3^2 = 9$ so this feels trivial but similar rules created addition and multiplication and for those the order is irrelevant. So this is actually a mystery!
Observe something else! For the three basic operations z is always at least as big as x or y.
Now with x{n}y = z this remains but the third input n can grow above z because for example, for all n values we get merely x as initial z value at y = 1. This fact, that we can get small z values for big n hides the quite opposite fact how fast z can grow with n.
As we showed, 2{n}2 = 4 for all n and yet we'll show in a second how fast 2{n)3 grows.
But this does not mean a real complexity of x{n}y = z as a relation. It is very "primitive".
Our rules give the impression that the collectable tuples are complex because we have the freedom to choose derived cases and stupid choices can become an infinity of cases that are only a very small subset of all derivable cases. But we can be much smarter too. We can regard an m fixed value and try all numbers up to m as inputs, that is case conditions in our matrix rule system.
At the start we'll only be able to use these numbers in the ◃ , = and ≠ basic relations there.
Then we get some target tuples and we can again regard input values up to m but now use the obtained target quadruples too. Repeating this, we get a stage where no more new targets can be obtained. Simply because we can't get infinite many target tuples by using only numbers up to m.
So we derived all quadruples with using only input values up to m.
Now observe that if we increase m to any M and use inputs up to M then we can not get any new quadruples with having all the four members only up to m.

Indeed, in the target quadruple, the maximal variable value was always at least as big as other values in the conditions. So the trying out of all inputs up to m must have produced such already.
And so we can generate the outside tuples too as follows:
We derive the inside ones in the above systematic manner using m = 1 , 2 , 3 , . . and at every such m value, once we have all the derivable ones, we list all x , n , y , z quadruples with values up to m and delete the ones we derived as inside ones.
So the x{n}y = z quadruples as set has a generable complement so it is dually effective.
Just like all classical Number Theoretical relations.

The leaping sub sequence, Ackermann function

Now I will show why 2{n}3 grows so fast. Compare the followings with the hollow abstractions you will find on Wikipedia about the Ackermann function.
Let z_1 , z_2 , . . . be a sequence of increasing natural numbers with $z_1 > 1$.
By this condition we obviously also have that $z_y > y$ too. Using y as index will make sense.

As an example, let's regard : z_y = 2 , 3 , 5 , 8 , 13 , 21 , . . .

Now let's regard the first element, here 2 and find the this indexed, that is second element, 3.
Then, the this indexed, that is third which is 5, then the fifth which is 13, and so on.

z_y = 2 3 5 8 13 21 . . . . 56 . . . .

This as new sequence, without the first member is the leaping sub sequence of z_y :

leap z_y = 3 5 13 56 . . . . . 757 . . .

Here we can again locate the leaping members:

leap z_y = 3 5 13 56 757

And so again without the first member we get:

leap^2 z_y = 13 757 . . . .

This can be continued and the sequences obviously grow faster and faster.

The weakest possible start is obviously:

y+1= 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 . .

This will be also denoted as $A_0 = A_0(y) = y + 1$. Because:

With A_1 , A_2 , . . . we'll denote the successive leaping sub sequences:

A_1 = 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 . .

A_2 = 5 , 7 , 9 , 11 , 13 , 15 , 17 , 19 , 21 . 23 , 25 , 27 , 29 , 31 , 33 , 35 , 37 , 39 . .

A_3 = 13 , 29 , 61 , 125 , . . 125 is the 61-th member in the previous sequence.

A_4 = 65533 , . . 65533 is the 13-th member in the previous sequence.

As we see, even with our minimal A_0 start we soon get incredibly huge numbers.

Observe that the first member of a sequence was found in the previous as the indexed by the first member there. The later members are again found in the previous but indexed by the previous member in the present sequence. So the leaping construction can be formalized as:

The two leaping formulas: $A_n(1) = A_{n-1}(A_{n-1}(1))$ and $A_n(y) = A_{n-1}(A_n (y-1))$.

And now comes the big surprise: $A_n(y) = h(2, n, y+3) - 3 = 2\{n\}(y+3) - 3$.

By the unique determinacy of the two leaping formulas, it is enough to show that $2\{n\}(y+3) - 3$ obeys those two formulas. Obeying the first would mean:

$2\{n\}(1+3) - 3 = 2\{n-1\}[(2\{n-1\}(1+3) - 3) + 3] - 3$ Leaving out the -3 ends:

$$2\{n\}4 = 2\{n-1\}[2\{n-1\}4]$$

Using the basic operation rule on the left twice and then that $2\{n\}2 = 4$:

$2\{n-1\}[2\{n\}3] = 2\{n-1\}[2\{n-1\}(2\{n\}2)] = 2\{n-1\}[2\{n-1\}4]$ indeed.

This in reverse means a derivation of the rule obedience. The other one can be similarly verified.

Finally, observe that $2\{n\}3 = 2\{n-1\}[2\{n\}2] = 2\{n-1\}4 = A_{n-1}(1) + 3$ and that every sequence member eventually gets to the front and a later $A_{n-1}(1)$ over grows it.
Thus indeed, $2\{n\}3$ grows almost as fast as the $A_n(n)$ diagonal members.
And Ackermann showed that these outgrow every $f(n)$ primitive recursive function.

Appendix 2.: Turing Machines, Busy Beavers

In a sense, this should've been the start but I intentionally wanted to show what basic assumptions were sufficient for the Impossible Machine Theorem, rather than get lost in the details.
A Turing table has n columns numbered by 1 , 2 , . . . , n and m rows corresponding to the m symbols used in an alphabet outside on a memory line. The members in the table are triplets, starting with a symbol then a left or right arrow and then a column number. Except one triplet somewhere, that instead of a column number has a H halt command as third member.
Here is a concrete Turing table:

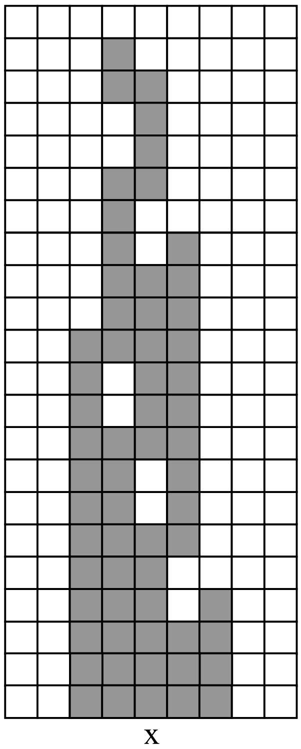
1 → 2	1 ← 2	1 ← 3
1 → H	0 → 3	1 ← 1

As we see, n = 3 and m = 2. This second means that we have only two symbols and since I told that these are the firsts in the triplets, if we look through these first members we see that actually our alphabet here is { 0 , 1 }. But I also said that the rows correspond to the alphabet.
This obviously means that we have to have a fixed alphabetical order because our table doesn't have any markings for the rows. In our case the logical alphabetical order is 0 , 1 so the first row corresponds to 0 and the second to 1. This correspondence tells what a head outside the memory line sees as new symbol at the memory cell where it is standing at a moment.
The first symbol of a triplet is the new symbol that the head must write in place of what he saw.
Then the arrow tells in which direction the head must move one step.
Finally the third member in the triplet tells the table head the new column it must go.
There it can find its row again by the symbol that the memory head will read at the next cell.
That already tells the two motions that happen outside on the memory line and inside the table.
For this to happen the two heads must have some communication and then the table and the memory line becomes a machine. The table head is only moving in this finite table while the memory head is on the infinite memory line. They intricately influence each other's motion.
Of course, our real interest is neither of these motions rather the texts that change on the line.
As I explained earlier, Turing went back to Elementary School digital calculations for inspiration.
The crucial point was to allow erasing that was forbidden in school. Of course the modern solution of erasing is the memory cell. Can be changed as many times we want it.
A huge disadvantage is that this way the history of changes is erased too.
So as a temporary clarifying solution I will use the good old squared paper and instead of rewritings, make under writings for our above example. Of course, we still need two assumptions.
The inner one is that we always start in the first column. The outer one in general is more complex, namely the content of the memory line and the position of the line head at the start.
But now we'll assume an all 0 initial memory content. Then the first few steps will be:

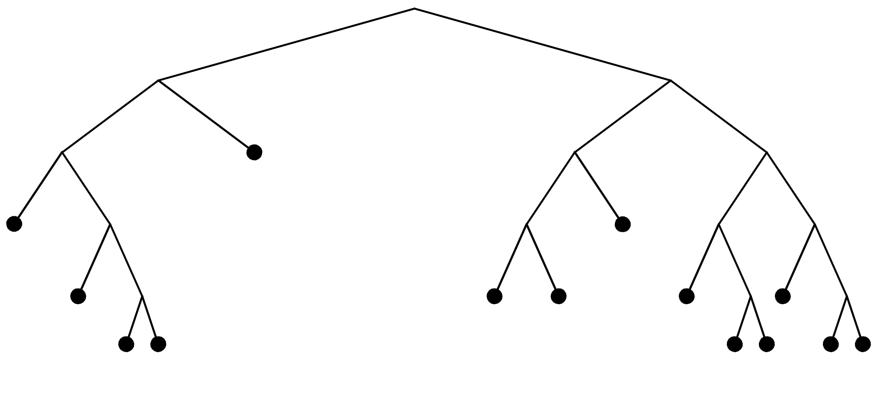
```
. . . . 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . .
                          1
                            1
                          0
                            1
```

The position of the line head at the start is clear by the second line where a 1 is.
As we said, we started in the first column and the read symbol was a 0 ordering the table head to use the first row: 1 → 2. So the line head should rewrite 0 to 1 but now just under write it.
The arrow told to move a step right and check the memory content there.
Finally, the 2 told the table head to move to the second column.
From here everything repeats, so again the read symbol decides the row and so on.
Just observe that if in our previous cycle, had the memory content been a 1 then in the first column we would have had to use the second row 1 → H and thus also write a 1 new symbol and make a right move but then make a halt. So we were this close to a very fast finish.

Will we ever come back to this option? This is the essence of everything!
Yet by looking at the table, in spite of being so small, it is a jungle and we have no clue about this.
The truth is that we'll come back to this option at the 21-th step. And here is the full history till halt with a more visual representation, using white squares as 0-s and grey ones as 1-s:

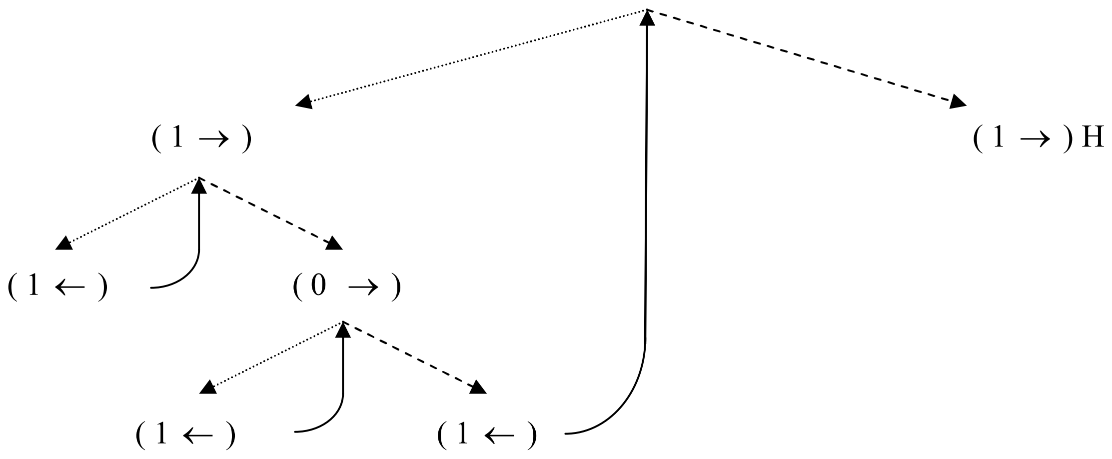


The x marks the head position at the halt and this reveals that without showing it you could not tell this at once. This is so because our figure shows the updated lines, so allowing really only 0 and 1 while the original underwriting start above had the blanks too through which we could see the last contents of the memory cells.
A better visual representation of a table's inner cycles can be done by a so called action tree.
This can be obtained by first regarding a finite graph:



The branchings correspond to the read symbols of the alphabet. In our case this is binary and so we have only left or right branchings so we can choose for example left for 0 and right for 1.
Such finite graph becomes a redirected graph if we redirect the dots, except one.
This suggests already that this will be the halt while the redirected ones the other columns.
But this is not quite so because the merely continuing branchings are also columns that simply go to the next column.

And finally, we must place the memory line actions to the branchings to get an actual action tree.
These are the first two members of the table triplets in parenthesis.
Here is the action tree version of the above table with solid lines as read 0 and dashed as 1.



As you see, we marked no columns at all and actually in this case they are unique anyway.
Indeed, the top is column 1 and this determines the next as 2 which again determines 3.

If we would want to introduce these as special flow charts then it is amazingly simple too.
Remember that in general we have rectangles containing the actions and rhombuses containing the yes or no decisions. Here we make decisions prior any action but they are not yes or no decisions, rather for any possible number value. So we don't need rhombuses at all.
The rectangles are then the actions in the parenthesis.

Returning to our particular machine, it is called the Busy Beaver Step Champion among the $m = 2$ and $n = 3$ machines.
First we should realize something very simple yet quite surprising.
I forgot this fact three times and every time I realized it again I was astonished.
A machine can halt or run forever from a starting input. This is the essence of everything.
But this gives a false impression that for a set of possible inputs the possible runs or halts are two infinite sets. And this is not true if we chose only from finite many possible inputs.
In particular if we only regard inputs shorter than a given k length.
Of course we haven't defined yet what actually an input is. Above the input was the all zero line.
So let's just say that we allow some 1-s too and then we can regard as input the binary segment containing all these allowed 1-s. Of course the length of the input is not the number of 1-s because by placing them further the combinations with the 0-s in-between can become arbitrary big.
But we could regard as the k input length the number of digits between the first and last 1.
The possible tables and thus machines with given m , n are only finite many.
Thus for given k long inputs we can try out all particular tables for all particular inputs.
This is a finite number of runnings. Some will halt some will not. But there can only be finite many that halts and each having a concrete step number to reach the halt!
The maximal of these is $s (k , m , n)$.
The machine or machines that have this many steps to halt are the Step Champions for k , m , n.
Tibor Rado realized that the s maximal step number is not an effectively calculable function.
Even with the simplest $k = 0$, that is all 0 input situations. $s (m , n)$ grows extremely fast.

$s (2 , 2) = 6$, $s (2 , 3) = 21$, $s (2 , 4) = 107$, $s (2 , 5) = 47{,}176{,}870$.

The $s (2 , 3) = 21$ case has only one machine that does this many steps and it was our example.
Actually, Rado was first looking for not the maximal step number rather the maximal written 1-s.
Of course with non halting machines we can easily write infinite many 1-s but he asked what is the maximum among the halting ones.
Observe that knowing s we can easily get that or any other strange features too.
Indeed, we just have to start trying out all the machines up to s many steps and find the champion for that particular feature.

As we saw, the maximal 21 step machine produced five 1-s but a much shorter 13 step machine can produce six. So the largest step number is not necessarily the largest in every respect. By the way, the busy beaver name refers to how these machines run back and forth.

Finally, I'll show a most boring but still very useful representation of the Turing Machines.
The idea here is that we really just want to show the action history as it happened.
Merely as a sequence of the triplets from our table.
So our example should start as:

$(1 \rightarrow 2) (1 \leftarrow 2) (0 \rightarrow 3) (1 \leftarrow 1) \ . \ . \ .$

To make better sense of this, we will write an **I** initial segment of the memory line as start and then so called helping duplets in-between the triplets. These will contain a position value on the memory line and the last memory content there underneath:

(–2 –1 0 +1 + 2 +3) (1 → 2) +1 (1 ← 2) 0 (0 → 3) +1 (1 ← 1) 0 (. . .) . . .
0 0 0 0 0 0 0 1 1 0

The upper value can be established trivially from the previous position and the arrow that comes. But for the lower memory content we must go back to the last same position and use the there applied new writing, which is the first member in the triplet, or use the **I** initial line.
So a general triplet sequence becomes: **I** ( ) d_1 ( ) d_2 ( ) . . .
The table and **I** determine perfectly the triplets and the in-between written helping duplets.
The first triplet's determination is by using column 1 and the symbol under 0 in **I** as row.
From then on, every triplet is determined by the last column and the lower value of d as row.
Observe that this is also an exactification of the input concept and allows the start from anywhere in the input because it can be located as the 0 position.
Most amazingly, we do not need the memory line at all!
Yet such triplet sequence is a D alteration sequence on the memory line!
But it is very easily verifiable whether it is a correct one using only the table and **I**.
The correctness of the continuing triplets is very simple to verify and the only other condition is that **I** should be correct, that is wide enough to give contents for all new positions.
The reason I used the letter D for such alteration sequence is this.
It corresponds to a derivation sequence in an axiom system. In fact, **I** corresponds to the axioms while the Turing Table to the Logical Rules. If this analogy could be continued to have some further meaning, it could lead to the use of Turing Machines directly in axiom systems.
Or there is a wider system that includes both!
That would be a big step towards defining Effectivity.

Appendix 3.: Multiplication Magic

Now I show why multiplication is really such a big jump that allows the coding of expressions. Then we can make self-referring statements that bring about undecidable statement pairs.
This was how Gödel was able to avoid effectivity. The non generability of the non-theorems was not mentioned by him. Simply because diagonality among the effectivities was replaced by these self-referring statements. We were not quite exact because this could have only meant self reference about the truths. But he used self reference about the derivations.
And indeed, he coded not just the expressions but the possible derivations too.
Our little glimpse will not get into the details of that.

Suppose we have a $(r_1, r_2, \ldots, r_n)$ so called tuple of natural numbers!
Can we create a code for it? We instantly think of a single number but a pair of c, d would be just as good. Indeed, for arbitrary long tuples just a double is a perfect result.
The more important requirement is that we should be able to recover our tuple from c, d using multiplication with logical symbols. This luckily allows division or rather dividability and even remainders. And indeed, with these it is possible to define an explicit $F(c, d, i) = r$ expression so that for any $(r_1, r_2, \ldots, r_n)$ tuple there are c, d codes that:
For all i up to n we have that $F(c, d, i) = r$ is derivable for a single r.
This is great because this way we don't need to decode n, rather just go up to n many members in our sequence of the defined r-s.
Now let's see what such expression can do because it is unbelievable.
We can explicitly define exponentiation! Indeed:
$x^y = z$ means that there is a $(r_1, r_2, \ldots, r_y)$ tuple that $r_1 = x$, $r_2 = x^2, \ldots, r_y = x^y = z$.
Here we still have the dots and y as subscript but we can avoid all that by saying instead:

$$\exists c \exists d \{ F(c,d,1) = x \wedge \forall k [k < y \rightarrow F(c, d, k) \cdot x = F(c, d, k+1)] \wedge F(c, d, y) = z \}.$$

We can see that any other repeatedly calculated tuples can be formalized if the calculation is expressible by addition and multiplication. And indeed, by such tuples we can describe anything effective. So not just exponentiation but all generable collections just became explicit.
Then we can show that the cases of these explicit definitions are even derivable in our system.
So the generable sets are not just explicitly definable in our language but expressible in our axiom system.
The theorem we need for the concrete $F(c, d, i) = r$ is called the Chinese Remainder Theorem:

Let $d_1, d_2, \ldots, d_n$ be real, that is non zero naturals and all relative primes to each other!
Then for every $r_1, r_2, \ldots, r_n$ values, each under the corresponding d, that is $r_i < d_i$,
we can find a c so that these r-s are all the remainders of the corresponding d-s in c.
And these r values may include zeroes as indeed, remainders may be such.

The notation reveals that our tuple will be the remainders in c but how do we get the mentioned d_i values from a single d. First of course, let's find our d.
Let $r_1 + r_2 + \ldots + r_n + n = m$ and then $d = 2 \cdot 3 \cdot \ldots \cdot m = m!$
Then $d_i = i\,d + 1$ will satisfy their conditions in the Chinese Remainder Theorem.
The first condition, the r values being under them is trivially true because :
$r_i < m < m! = d < i\,d + 1 = d_i$
The second condition, the relative primness means that:
For every $i \leq n$ the $i\,d + 1$ values are all relative primes to each other. First of all:
They can not have a non 1 divider of d as divider, because such leaves 1 remainder.

Now if two of them $j\,d+1$ and $k\,d+1$ had a common p prime factor then p would divide $(k-j)\,d$ too. But p divides separately and can't divide d since it is a divider of d too. But neither can divide $k-j<n<m$ because $d = m!$, so $k-j$ is a divider of d too.
Thus we can use the theorem and claim that such c exists.
This c is a concrete number, though the theorem doesn't give it explicitly.
For our purpose it is enough that it exists.

Then the explicit expression for $F(c, d, i) = r$ is :

" r is the remainder of $i\,d+1$ in c " $=$ $r < i\,d+1 \wedge \exists\, q<c\, [\, c = q\,(i\,d+1)+r\,]$

This is always defined for all i values up to a point, namely when $i\,d+1$ exceeds c.

Some people define remainders even beyond there, by regarding the q quotient as 0.
Then the expression is giving a full infinite sequence of r-s.

For a proof sketch of the Chinese Remainder Theorem observe these:
Let $P = d_1\, d_2\, ..\, d_n$. The $0, 1, 2, \dots, P-1$ values under P mean P many choices to try as c and as easy to see, they all give different remainder tuples. But the possible under valued tuple combinations are also this many combinatorically as product of choice numbers.
Thus, at least one tried value under P must be a c, giving our tuple.
There is an uglier method to find an $F(c, d, i) = r$ expression without the Chinese Remainder Theorem and it is using a c that is explicit from the tuple.