The fundamental problems head on

Mathematics became notorious for talking about things that nobody understands. And yet mathematics is actually talking about the most common knowledge, about our born intuitions.
So what's going on? The truth is that reaching our intuitions is not easy due to material delusions.
This is especially the case when a new concept is born. To tell the intuitive meaning is then not enough because it was obviously in disguise up until that point. So the logical question from those who were blinded by the earlier formalisms, is "what's the big deal", what's new in terms of the old formalism. In the third and fifth sections, I will answer basic questions about both the set and Effectivity concept in parallel. But such head on justification is not mentioned for either.
This was a sudden realization, weeks after I finished the article and so I inserted this section.
As a parallel example, an immediate justification for the set concept did exist. It gave very simple proof for the existence of transcendental numbers. But this wasn't a very strong justification because the new proof was much less concrete or effective. So that case has a strange relationship with the new, concerning Effectivity and this relationship remained continually.
So actually Cantor should have realized Effectivity too, in order to respond correctly to his enemies who regarded the set concept as meaningless rubbish.
To be more accurate, an automatic assumption of Effectivity was somehow included in "logic".
As Logic truly developed, it turned out that Cantor's set concept perfectly relates to Logic.
The seal of this perfection was Gödel's Completeness Theorem in 1930.
Hilbert was present but could hardly follow the proof. I mention this because he was the one who stood completely behind Cantor and in his 1900 speech at the Paris Math Congress said that Cantor lead mathematics into a new paradise. He himself did what he said because he made Geometry's first totally axiomatic buildup using sets.
But we have to admit that Cantor's naïve set theory was very imperfect.
One axiom was simply contradictory, namely the collection by properties and one was not even recognized yet, namely the Axiom Of Choice.
Behind both of these seemingly set theoretical problems actually a logic problem lied!
The concept of variables relating to quantification. This was that stopped Aristotle to really grasp quantification already two thousand year earlier. I will talk about these in a minute.
But now I have to emphasize that when Zermelo discovered the Axiom Of Choice and was able to inject it successfully to eliminate the crucial "naivity" of Set Theory then the above mentioned hidden assumption of Effectivity was actually pushed even deeper underground because a new and fourth concept, Randomness was left unmentioned too.
Today it is crystal clear that there are four concepts as four pillars on which new math rests.
Set, Logic, Effectivity and Randomness.
At the above mentioned 1930 math congress where Gödel presented his Completeness Theorem, in the intermissions something even more important happened. He explained to the few understanding ears a quite opposite negative result about Logic he was working on.
Later, this became called the Incompleteness Theorem. But as usual, these short and condensed English phrases leave a lot to be explained. Here the major distinction is that while the Completeness Theorem says that Logic is complete, the Incompleteness Theorem does not say that Logic is incomplete, rather that most Axiom systems are that. The two "complete" have different meanings. Of course, indirectly it is a negativity about Logic and then we can also ask immediately what else is in an axiom system that causes this result beside Logic being as it is.
And then the answer has two parts.
One that was over emphasized by Gödel and the immediate reading of his proof, that the axiom system can grasp its own derivations. Self-reference, and so on. But a much more humble second condition was not mentioned because it was assumed as trivial, namely that the axioms must be effectively listed. In Number Theory for example the infinite many induction axioms are indeed effectively listable because for every logically expressible property we have a possible induction.
And these properties are listable by their buildups. The counterexample, that is finding axioms that are not effective and defy incompleteness is so absurd that nobody cared about this.
But it is there! Namely, we can regard as axioms the actually true statements of Number Theory.

Then obviously we get a complete system where every statement or its negative is a theorem, simply because it is already an axiom. This reveals that truth itself is a very non effective concept.
And indeed, the verification of universal statements requires to test infinite many cases.
The simple fact, that certain complicated enough effective sets must have non effective complements was not recognized. And yet with this in mind everything is plain and simple:
If the axioms are effective then the logical continuations, the theorems are effective too!
But the complement set the non-theorems is not necessarily. And if that happens then the completeness is impossible for sure. Indeed first of all, completeness means that every statement is decidable, that is either it or its negation is a derivable theorem. Let's observe that one form of the effectivity of theorems is that we can generate them by a machine. This seems unbelievable and seems to devaluate the genius mathematicians that find the wonderful proofs of theorems.
In fact, first we might think that a machine though can spit out all theorems, it can not find the proofs. But the truth is the exact opposite! The machine can spit out all theorems by exactly trying out all possible derivations. The method is embarrassingly simple. It must try out all axioms and all logical steps. The result is an insane generation of all derivable statements in an unpredictable order. To wait for a theorem to appear requires astronomical times. And of course if the theorem is just a hypothesis and we just care about the fact if it's derivable or not, then the situation is even worse because we might never get a result because the hypothesis is not a theorem.
Completeness of course would help here because then we can watch not just our hypothesis but the negative of it too. Indeed, one of them must come up. And this helps to understand our argument why a definitely not generable set of non-theorems implies at once non completeness.
Namely, with a very indirect step. Indeed, assume that completeness would stand and so for any statement either it or its negative is derivable and so would be spit out by our computer.
Then those statements that are not spit out are also spittable out by an other computer.
And this other computer is simply the previous but with an added negation forming.
So the non-theorems being non generable, defies this strange construction and thus defies its assumption that the derivations and thus the generation can get one of each statement and its negated pair. There is an amazing postscript to this simple argument.
Namely, that both the original Gödel argument and the new non generability of the non-theorems, must assume that the axiom system is consistent, that is leads to no contradiction.
Amazingly, a contradiction can derive everything and so then our derivator computer would spit out all statements. Even more amazingly, we wouldn't be able to spot this in finite time.
After all, for years it could only spit out statements without opposite pairs in there.

These ideas were ripening a few years after Gödel's result but their crystal clear recognition with an unquestionably simplest framework too, came by Turing.
He was a visionary just like Cantor. This included the role of future computers and this explains the used word "computable" in his article title. By the way, it was not that corresponds to the crucial effective collections of texts, that he actually used. He meant infinite decimals in the title as "numbers" not natural numbers that correspond to texts in other effective frameworks.
His article is "messy" in other much more important points too.
His title should have been: Universal computing machines and their consequences for Logic.
He showed that universal computing machines exist and imply that some text collections by computers can have complement that is not collectable by any computer. As I said this "computer collectable" was not what computable in his title meant. Also, this inner universality of his computing machines does not prove an outer universality that they can imitate every effective method. This is unprovable as I mentioned above. Still, his framework brought in something new.
This "new" is still not quite clear today and it has nothing to do with his foresight of computers.
So those who renamed Effectivity to Computability as sign of adoration toward Turing showed their stupidity to dig deeper into Turing's ingenuity. Turing of course was right, we did enter the age of computers so this false movement met with false populous acceptance.
The term "Computably Enumerable" is the center of all lies!
This contains the false replacement of recursively with computably but keeps exactly what Turing surpassed. This term is actually what an Effective collection should mean.

The root of this lie was a historically foggy road . Not relations rather functions were used.
The original cause was simply that functions can be applied inside each other. We can combine them, build them up. But this still does not give explicitly those functions that we can derive from equation systems case by case. Later this simple idea indeed became an accepted definition of effective functions by Gödel being motivated by Herbrand.
But originally they tried to be explicit as much as possible. So the Primitive Recursive Function was introduced that allowed only a simple step by step incrementation of the domain. As expectable, this could not reach the general recursive functions. But three points are missing here.
Firstly, that if we derive cases of functions, we might not derive values for all variable cases.
So we only derive a partial function as end result.
Secondly, here the free possible derivation roads seem to be a new essential part of what can be derived. Thirdly, the derivations in formal rule systems derive objects as such and not functions.
Kleene made one very smart step to combine the first two problems into one.
He allowed partial functions and regarded the already existing primitive recursive functions as the start of the buildup. By allowing a single new "step" he could obtain all functions that can be derived case by case from an equation system.
This new step is, to assign to any $x_1, . . , x_n$ values a y value which is the verifiably first z number that satisfies $f(z, x_1, . , x_n) = 1$ for an already built up f function.
The word "verifiably" is crucial here and so the conventional μz operator for predicates is not enough. Indeed, $\mu z [f(z, x_1, . , x_n) = 1]$ just assigns to some $x_1, . . , x_n$ the first $z = y$ that satisfies the equation. But then earlier $z < y$ values with these $x_1, . . , x_n$ together may have been undefined places for f. We then can not verify the minimality of y. So we require:
$f(1, x_1, . , x_n)$ defined but $\neq 1$, . . . , $f(y - 1, x_1, . , x_n)$ defined but $\neq 1$, $f(y, x_1, . , x_n) = 1$.
Using dots is not an explicit definition but this is not a problem. We want effective not explicit definition here. The real problem is that f must have values for up to y. So we made assumptions beyond being a partial function. But this is not a real problem either! Exactly because we allow partial functions! If for a partial f function such limited totality up to y is not true then simply our $\mu^* z [f(z, x_1, . , x_n) = 1]$ function will not be defined at $x_1, . , x_n$.
This also solves our previous problem that a seemingly non explicit definition becomes explicit!
Indeed, this μ^* search for the first value is a single step. So in the buildup of the effective partial functions it is an explicit step. Every effective partial function will have a finite buildup.
The real ugliness of this whole approach is that it remained with functions.
This becomes even uglier if we want to collect numbers or tuples, that is define properties or relations. These should be the cleanest, effective collections. But now these are partial functions with a fix value. That is, subsets of a domain that give a fix value. It all hides the simple fact that the total functions can not be effectively recognized among the previous buildup of the partial functions. So this whole theory proves its own false beginning.
The false idea of starting with total functions, translated into collecting objects becomes the false idea of collecting both a set and its complement. So not just recognize objects but decide them if they should belong to a collection or not. But decision is not elemental only recognition.
Only Turing started from scratch and made a very wide system using recognition of texts.
But for some strange reason which is not yet clear today, this framework has its drawbacks.
That's why for some goals the old start with dual effectivity remained.
A very appealing such effectivity approach regards as these dually effective starting collections the number tuple sets defined by formulas using bounded quantifications in the language of number theory. Then by allowing non bounded existential quantors, we get all possible searches and thus a very convincingly wide class of recognizable collections.
In my view this is the best effectivity framework and strangely, not used in this book.
But now we can come back to why still Turing grasped effectivity the "best" way.
He avoided even the concept of numbers or tuples by collecting texts instead.
We simply alter texts forever as our starting concept.
The alteration itself is what we do in elementary school calculations or rewriting texts.
The truly inner idea of all this is that in place of a text altering person he placed a rule system.

We have a table that tells how we must alter our text letter by letter and move left or right.
Borel at the beginning of the last century placed an imaginary monkey in front of a typewriter and realized that the Bible will be typed infinite many times. Turing replaced the human with not an animal that types randomly but with a machine that types by strict rules.
These rules are defined by not merely the symbol we are at but by also the so called “states”.
Just as there are finite many fixed symbols in our texts, every machine or table can use only finite many states. So “we” or rather the machine is always in a state and the table tells how this changes parallel with the altering of the text. This crucial idea is also the reply to what you should have asked at once: How can this system that so perfectly reflects search as text alteration, stop at all?
We simply choose one state as halt and so do not prescribe other states or text alterations after it.
This reflects the essence, that a state can turn up first after arbitrary many steps.
This is not plausible at all! The finite state number suggests that fix cycles should come about that are calculable from the state number.
But what we forgot about is that the text out there will alter these cycles.
This was all very interesting but didn’t give even an initial justification for effectivity as say Cantor’s simple derivation of the transcendental numbers gave for sets or rather equivalence.
So we must admit that classical math has no direct connection with these new results.
A most honest start for our subject is to try to clarify what Effectivity entails in classical terms.
This then starts with separating it from an other word we use quite similarly in every day expression, namely “explicit”. But this simply coincides with “logically definable”.
So we really must start with Logic. Logic contains two groups to form expressions.
Connectors and alterers. Connectors simply combine two expressions into a new.
The two can have four kinds of true-false combinations and the result can be chosen prescribing any desired true or false values to each. This means sixteen possible connectors. Not all of these are necessary though.
The simplest connector is the “and” = $\wedge$ which is only true if both members are.
So: $t \wedge t = t$ but $t \wedge f = f$, $f \wedge t = f$, $f \wedge f = f$
The “or” = $\vee$ quite oppositely, is only false if both members are false.
So: $f \vee f = f$ but $t \vee f = t$, $f \vee t = t$, $t \vee t = t$
But we use two other “or”-s in every day expressions too. The “either-or” = ∇ is only true if exactly one member is true and the “exclusive-or” = $|$ is only false if both members are true.
A strange connector is implication = $\rightarrow$ because it is often confused with some cause effect meaning but here we only care about true and false possibilities. So mathematical implication is only false if the condition is true and yet the consequence is false: $t \rightarrow f = f$.
The reason we don’t need all sixteen connectors is that the first alterer, negation = $\neg$ can be used to express connectors with each other. Already the five we introduced above are too many.
For example: $A \rightarrow B = \neg A \vee B = \neg (A \wedge \neg B)$.
The other two logic alterers are also called “quantors” and they correspond to the everyday expressions of “there is some” = $\exists$ and “for every” = $\forall$.
Aristotle discovered the importance of these but his “formal logic” claims using these, remained very artificial. The crucial missed point and thus the real essence of mathematical logic is the use of variables. A common misconception is that variables were always part of mathematics!
Actually, it only started at the beginning of the nineteenth century. Nowadays $y = x^2$ and similar variable number usages are trivial already in elementary school. But this variable revolution is less than two hundred years old. To use variables not just for numbers but for all objects, was the big step to finally make Aristotle’s quantors come alive. So to say that all smart people have a bicycle, we must use x and y variables corresponding to people and bicycles.
Then being smart, being bicycle are properties $S(x)$, $B(y)$ and owning is a relation $O(x , y)$.
So the claim or statement is $\forall x [S(x) \rightarrow \exists y (B(y) \wedge O(x , y))]$.
Statements should not contain free variables and here too x , y are quantized so can not vary.
An even simpler fixing of a variable is to replace it with a name. This could be called a concretization. “Peter has a red bicycle” is the statement: $\exists x [B(x) \wedge O(P , x)]$.
Now we can finally come to the promised “easy” meaning of “explicit” as logically definable.
In a mathematical theory we start with basic relations as our language.

This includes one variable properties and operations like addition as $x + y = z$.
With logic we can build new relations and these are then the defined or explicit relations.
In Number Theory for example, from the basic relation of multiplication the most important property we define is being composite: $C(x) = \exists y \exists z [x = y \cdot z \wedge y \neq 1 \wedge z \neq 1]$.
The opposite of composite is being prime, except we exclude 1 for technical reasons.
So being a prime is also explicit as $P(x) = \neg C(x) \wedge x \neq 1$.
To establish if a number is composite is merely finding some factors. To establish if a number is prime is a bit harder because we have to try all smaller number if they could become factors.
This fact that we only had to try the smaller numbers was crucial and yet external to the definition.
This duality of the composites and primes is a basic general situation.
If we define something by only using $\exists$ existences and names then the verification of our defined explicit relation for any concrete cases boils down to verifying the basic relations.
Finding the cases can be very hard. If we only have one existence then we can try all numbers one by one but if we have more, then to try out all pairs or all triples would require some tricks.
In spite of this, we regard the collection of these existing cases, the lucky finds, as effective.
If we use negation or $\forall$ universal quantification then the situation changes drastically!
To verify the concrete cases of the left variables now requires to try out all objects for these universally quantized ones. A p being prime means that no n is factor of p. So every n number is not a factor of p and so we have a potentially infinite verification process here too.
Luckily, we know that if n is factor of p then $n < p$. So we can stop our process at $n = p$.
And all other number theoretical defined relations seem to be similar. We have knowledge from our reality or our axioms that avoid never ending verifications. But this is a false illusion.
There are explicit relations that are not effective but they are hard to show through number theoretical means. That's why the hocus pocus comes about when we start our subject in hollow.
The best is to imagine that we form explicit definitions without any meanings, just building up all kinds of logical relations. All these complicated universality assumption being replaceable by bounded searches would be very unlikely. And yet to find ones where we can be sure that such simplification is impossible, we must step above number theoretical examples! So quite strangely, to see non effective collections of concrete objects we must step above concrete examples.
All this shows that Number Theory, or in fact numbers themselves are not the best start to examine Effectivity. The much better start is texts.

Texts, sets, classes

An $\mathfrak{A}$ alphabet is a fixed finite set of symbols in a given order.

In our notation, letters in $\mathfrak{A}$ are lower case Latin, but digits, signs and space can be included.

So $\mathfrak{A} = (a , b , . \ . \ , z , 0 , 1 , . \ . \ , 9 , ! , . \ . \ , ?)$ with dictating an "alphabetical order" too.
Texts are any finite long combinations from the alphabet. Ex: apple? no!
Texts will be referred to by capital letters $A , B , . \ .$
Sets are short for sets of texts and these will be denoted by bold capital letters $\mathbf{A} , \mathbf{B} , . \ .$
Classes will mean any sets of text sets. These will be denoted by the cursive capitals $\mathcal{C}$, $\mathcal{E}$.

$\mathfrak{A}^{\infty}$ is the set of all texts made from $\mathfrak{A}$. Surprisingly, $\mathfrak{A}^{\infty}$ is merely a sequence of texts.

To see this, we can use the already given sequencing of $\mathfrak{A}$ as alphabetical order.

Dictionaries list words alphabetically too but this wouldn't work for all texts because we would never get to letter b since there are already infinite many texts starting with a.
The heuristic idea is to mix the alphabetical order with length order!
So we start with the 1 length texts, that is single symbols. These are simply our alphabet.
Then we list all 2 length texts alphabetically, then the 3 length ones and so on.

From the $\mathfrak{A} = (0 , 1)$ simplest binary alphabet the list of all texts is this:
$0 , 1 , 00 , 01 , 10 , 11 , 000 , 001 , 010 , 011 , 100 , 101 , 110 , 111 , 0000 , . \ . \ .$

A nice visualization behind the sequencing of all texts is the “text tree”.
We write the increasing length groups as lines under each other. So our first line is the alphabet.
Under every symbol we can start branchings down into every symbol again.
So we’ll have m, m^2, m^3, . . increasing long lines.
The missing tip of the tree is the $\varnothing$ empty text.
Then we can draw branchings from $\varnothing$ to the symbols.
And indeed, every symbol is a continuation of this empty text.
Later we’ll use this abstraction for an other meanings behind.
A similar surprise of sequencability is to sequence all the fractions between 0 and 1.
To be surprised, you should know that these fractions on the interval $[0,1]$ are a dense subset.
Between any two fractions lies a third because the middle of them is a fraction again.
Indeed, it can be calculated as their average.
Thus of course there are infinite many fractions between any two. So now here is the surprise:

$$\frac{1}{2}\ ,\ \frac{1}{3}\ ,\ \frac{2}{3}\ ,\ \frac{1}{4}\ ,\ \frac{2}{4}\ ,\ \frac{3}{4}\ ,\ \frac{1}{5}\ ,\ \cdot\ \cdot\ \cdot\ \cdot$$

Even more surprising is how easy the sequencing was. We went by increasing denominators.
Cantor realized that all the points of $[0,1]$ is not sequencable. So, for any sequence of points there, at least one point of $[0,1]$ must be missing from the sequence.
This can be proved by realizing some basic features of how points are in the continuity.
But a simplest proof is by identifying the points with the infinite decimals.
This identification is not perfect because: $.270939999999\ \ldots = .27094000000\ \ldots$
So we have two decimals for some points and thus proving that the decimals are not sequencable could just mean that these double forms caused this. This however is impossible because these double marked points are only a sequence. Indeed, .1999. . , .2999. . , .3999. . , . . . with all increasing numbers before the 9-s will give all extra decimals.
So if all points were sequencable as P_1, P_2, . . then adding these extra all 9 ending forms would be sequencable too as P_1, .1999. . , P_2, .2999. . , P_3, .3999. . , . . .
So we only have to show that for any list of decimals one is missing. Let one list be:
$D_1 = .2680\ldots$
$D_2 = .7048\ldots$
$D_3 = .6593\ldots$
.
The diagonal D decimal is formed by picking the n-th digit from D_n.
In our case it starts as $D = .209\ldots$
It would be very strange if this D were in our list but nothing forbids this.
Now let’s alter every digit of D, or to be specific let’s add 1 to all digits but for 9 meaning as changing it to 0. If this decimal is D^+ then in our example it starts as $D^+ = .310\ldots$
This D^+ must be missing from our list.
Indeed, it can not be D_1 due to the differing first digit. It can not be D_2 due to the differing second digit, and so on.
Now it’s not surprising that the class of all text sets that can be made from $\mathfrak{A}$, that is the class of all subsets of $\mathfrak{A}^\infty$, is not sequencable either. To prove it is quite easy too:
We already showed that all possible texts are sequencable as:
$T_1, T_2, T_3, \ldots$
So any infinite binary sequence $0010001101\ldots$ could be used to define a subset by picking the T_n text if the n-th digit in this sequence is 1.
Thus, it is enough to show that these binary sequences are not sequencable.
For them the Cantor diagonal method again shows that D^+, which is there the opposite of D, that is changing all 0-s to 1 and 1-s to 0-s, is missing from any sequence.

Magic hat

Cantor's set concept and the existence of different infinites caused a revolution of mathematics.
Sets are collections without any given order and this may actually sound as an idiotic idea.
Why would we ignore the structure of a collection? This can only be explained by the end result.
Indeed, at the end these very hollow collections are able to describe all structures.
The prelude to the different infinites is that many structurally different sets are actually same infinites like the naturals and the fractions. The even prior prelude to these surprises is the simple fact that an infinite set remains the same infinity by taking away only a finite amount from it.
Strangely, this becomes only evident if we do regard an order. From the naturals for example we can take away the numbers up to 5 and then the remaining set 6 , 7 , 8 , . . is equivalent to the original 1 , 2 , 3 , . . This is so obvious that we are not even surprised by seeing it.
The fact that half of this set, like the odds or evens are also the same infinity, is again trivial by:
1 , 3 , 5 , . . or the remaining 2 , 4 , 6 , . . both being equivalent with 1 , 2 , 3 , . .
Galileo was surprised by this when he formulated his falling law as the odd consecutive falling distances. The total falling distances from the top are thus 1 , 1+3=4 , 1+3+5=9 , . . the squares.
This is a much better law because then we can use any t time and calculate the fallen distance.
Just like I was wandering away from the infinite sets by these, he was wandering away from the fall by his observations about the infinity of the odds.
In truth, when we wander away and get surprised by something then we actually visit the future!
Imperfect glimpses of future importances are revealed to us. And all our recognitions are merely glimpses! A robust recognition like Cantor's set concept always hides consequences that the creator could not foresee. And such parallel story is the very subject of this article.
The different infinites only come in as marginal importance for us. I will come to one such now:
We made one consequence already, that the set of all text sets is not sequencable.
The title is about machines and so as expectable, we are going to use machine collectable sets.
Since a particular machine is a finite variation of its parts or the framework we use, thus the machines could be sequenced just like the texts. So machine collectable sets are only a sequence.
Thus, we at once know at this very start that some text sets can not be machine collectable.
But this is not the main importance for us! We have a much more surprising direction.
Namely, that some machine collectable sets are such that the texts outside the set, the complement of the set, is not collectable by any machine at all!
Text sets of course raise the same question about the ignored orders, as sets in general do.
And here too the end result is the only convincing judgment.
But strange possibilities already show that this text collection idea is a very promising direction.
To raise again the doubting question: Why would you throw texts into hats that contain them without any order? And then I say: I will not only have such stupid hats but I will have a magic hat that contains all hats combined. Then you might say: Well, you already showed that all texts are just a sequence so this is a hat too. And then I say: No, you misunderstood me. I will have a magic hat from which I can recover every single hat. So they are not melted in, like in the total set of texts. And I hope this raises your attention.
The idea is so simple! Suppose you are an old fashioned mathematician like Kronecker was, who wanted to silence Cantor and also said that only the natural numbers were created by God.
So we want to collect natural numbers like say the squares or the primes. Watch this:
squares:1 , squares:4 , squares:9 , . . primes:2 , primes:3 , primes:5 , . .
Now throw all these into my magic hat. You want to get the squares? All you have to do is search in my magic hat for the beginning "squares:" and cut this label off. You will get the squares.
But you will also get the primes and any other number collections. Now if you are a notorious doubter with intelligence then you will come up with this: Well, I bet you will not be able to find your magic hat collection in your magic hat too. And so here is the punch line: Sure I will!
Imagine that we have labeled and threw into my magic hat everything except my magic hat itself.
Now also add as beginning to every text "magic hat:" and add these texts too. I know, this is not good because cutting off "magic hat:" will be only the full previous set but not the magic hat itself. So I am not finished. I will add not only "magic hat:" but the double, triple and all repeats

of this label as prefix to all texts. Then indeed, taking off “magic hat:” from all texts that contain it, will give back the full magic hat set.
So we gave a twist to the fact that infinite sets, in particular already an infinite sequencable set, can almost contain itself. The “almost” at the equivalences meant merely a one to one possibility.
But here we truly feel that the magic hat is inside itself which of course is impossible.
We have to regard a proper subset, namely the texts that start as “magic hat:”.
But then cutting this label off, indeed will get back the original full set.

Basic theorems

$\neg \mathbf{T} = \{ T ;\ T \notin \mathbf{T} \}$ = the complement of **T**.
BE = the B text continued with E. ex: applecider made from B = apple and E = cider.
$B + \mathbf{T} = \{ BE ;\ E \in \mathbf{T} \}$.
$\mathbf{T} - B = \{ E ;\ BE \in \mathbf{T} \}$.
$\mathbf{T}/2 = \{ T ;\ TT \in \mathbf{T} \}$.
U is a universal set in the $\mathcal{C}$ class if $\mathbf{U} \in \mathcal{C}$ and for any $\mathbf{T} \in \mathcal{C}$, there is a B text that:
$\mathbf{U} - B = \mathbf{T}$.
It may feel paradoxical that using **U** as **T**, we get $\mathbf{U} - U = \mathbf{U}$ for some U text.
But this is very possible. In fact, if any **U** set contains only texts that don’t have a U as beginning, then adding to **U** all its texts again but with added U, UU, . . beginnings, we will get a **U***, in which the U continuing texts are exactly the same set as the full **U***.
An **A** set shares the T text with the **B** set if either T is element of both or neither of them.
So in other words if $T \in \mathbf{A} \leftrightarrow T \in \mathbf{B}$.
Obviously, the only set that shares no text at all with an **A** set is $\neg \mathbf{A}$.
Now here are the basic four theorems:

1. If $\mathbf{A} \in \mathcal{C}$ and **A** shares text with every $\mathbf{T} \in \mathcal{C}$ then $\neg \mathbf{A} \notin \mathcal{C}$.
2. If **U** is a universal set in $\mathcal{C}$ and $\mathbf{U}/2 \in \mathcal{C}$ then $\neg (\mathbf{U}/2) \notin \mathcal{C}$.

3. The following three can not stand together:
   a. $\exists$ **U** universal set in $\mathcal{C}$
   b. $\mathbf{T} \in \mathcal{C} \rightarrow \mathbf{T}/2 \in \mathcal{C}$
   c. $\mathbf{T} \in \mathcal{C} \rightarrow \neg \mathbf{T} \in \mathcal{C}$
4. There is no universal set in the class of all text sets.

1. is trivial because **A** shares no text with $\neg$ **A**.
2. By 1. Enough to show that $\mathbf{U}/2$ shares text with every **T**. And indeed:
   For every $\mathbf{T} \in \mathcal{C}$ there is a B text that $\mathbf{U} - B = \mathbf{T}$ and thus:
   $B \in \mathbf{T} \leftrightarrow BB \in \mathbf{U} \leftrightarrow B \in \mathbf{U}/2$.
   So $\mathbf{U}/2$ shares B with **T**.
3. a. and b. are the conditions of 2. and the claim of 2. is defying c.
4. b. and c. are true for the class of all text sets, so a. must be false.

Basic ideas

We used earlier labeling of texts so they could be collected from our “magic hat”.
If we just label text sets without attaching the labels to each text then it is a set assignment.
So a set assignment is a [ ] function that assigns a **T** set to any A text: $[A] = \mathbf{T}$.
The A text can be regarded as merely a name of [A] but it may even be a rule how to produce this collection of texts. In either case we can have more A that name or produce the same set.

Here the point was that every text has an assigned set. An opposite direction could be to require that every set in a class should have a text to which it is assigned. Then usually the assignment is called an indexing of the class. The important case will be the class being the $\mathfrak{E}$ class of effective sets and the $[A] = \mathbf{T}$ assignment being the **T** set collected by A. The collection itself will be a recognition of input texts by stopping an alteration sequence defined by some M machine that has A as program. This would only work for A-s that are programs but if A is not a program we can simply regard it as collecting nothing that is [A] = empty.
Of course, there are real programs that never stop and thus collect nothing. But this redundancy just melts into a more general crucial feature of variant programs collecting same sets.
Our earlier remarks about universal sets not being absurd will become a reality too.
Unlike our artificial label as “magic hat:”, a natural U text will come about that will name the universal set as [U]. We might start to think that this U is some magic text.
But it’s even spookier because as we revealed, our texts will be programs, so rules of collection.
Thus U is a magic formula too. The secrets behind this magic were unveiled by Turing.
First of all, he was convinced that this $\mathfrak{E}$ class of the effective text sets exists as an objective almost physical reality. He also found a method, his machines, that are the most convincing way to produce all sets in $\mathfrak{E}$. Thirdly he realized that texts can become programs of a machine that can be used by an other machine. But most importantly, his fourth discovery was that the other machine can become universal that is being able to simulate all machines.
So we don’t have to regard arbitrary big machines to get all possible effective text sets.
The arbitrary length is still there that corresponds to arbitrary collection requirements but only in the programs that the universal machine uses. The fact that the data must be arbitrary long is unavoidable and thus also that a collection requirement must be arbitrary long. Indeed, if we regard an arbitrary finite set of texts then this should be an effective collection for sure.
In this case the program must contain these texts. Though this suggests a possible detour because actually if the finite many given texts have some pattern then we don’t have to give them one by one, rather our program can generate them by a much shorter set of instructions.
We’ll come to this heuristic idea of Kolmogorov later.
The existence of universal machines seemingly became trivial with the modern computers that can all simulate each other. But this triviality is a false finitized simplification.
The true depth of universal machines comes out if we’ll see what they imply in infinites.
They imply that some effective collections have no effective complements.
Usually just a diagonality is mentioned in these arguments but this is a very false approach too!
Already the original Cantor diagonality shows where a didactical error lies!
For any sequence of decimals the D^+ altered decimal is a quite explicit new decimal outside.
There is nothing indirect here! The indirectness comes in only at the consequence that the full set of decimals can not be sequencable. Indeed, if it were then we could get a contradiction by finding a new one that shouldn’t exist. Here with effectivities the situation is similar. A set has no complement in a class if and only if the set shares text with every other set in the class. So if we can exhibit these shared texts then the no complement is a perfectly direct claim. From the universal machine we can exhibit this for a machine and so we see quite explicitly why this machine has no complementing one. Only to widen the cases, that is give more examples is the problem! For other machines we can not easily show the shared inputs but indirectly we can show the impossibility of complement from the already known cases. Then we can be more explicit for even those cases. So this distinction of direct or indirect possibilities lie at the heart of everything!
And so, a simple common diagonality vision is sweeping everything under the carpet.
There is an other less complicated matter that was inherent in Turing’s results.
This is the external universality of his whole framework of machines. Meaning, whether they can indeed replace all effectivities. Unlike the internal universality of some machines that can be proved perfectly, this external universality of the framework itself is not provable.
To have a concept that is outside mathematics and yet is not in physics either is not an easy thing to swallow. We still didn’t swallow it. Will physics once embrace Effectivity?
Most amazingly, a new similar concept emerged, Randomness.

An other important point is that it is not obvious that a mathematical definition of Effectivity is impossible. We could have something that instantly contains all possible effective methods.
Instead, the frameworks are very narrow and only close examinations show that they can simulate each other. This feature must be part of a bigger picture where Effectivity is a single concept.
We don't see this bigger picture yet that probably will include a single Randomness too.
To find merely examples for effectivity, that is particular sets in $\mathfrak{E}$ is very easy.

A strange example is the set of all texts $\mathfrak{A}^{\infty}$. We saw how this can be listed by the heuristic length increasing alphabetical order. This would suggest that effectivity should be merely regarded as sequencability. This however is a half-baked idea. Cantor's sequencability concept hid effectivity behind but exactly jumping over, that is ignoring effectivity was that made him able to define the crucial non sequencability of all points.
A new name that suggest not to jump over the effectivity in a sequencing is generation.
So a generable sequence is done by some effective method. But this is still a not so good idea!
It hides an interesting duality though that we see already for number sequences. Namely, whether we give the members by some rule that uses the indices that is their places in the sequence or we use some recursive method to get the members from each other. We might combine both and think we achieved something general but we are wrong again. A crucial big jump is still missing to allow potentially infinite searches for deciding if we want to list a certain number.
This has been finally realized by those who followed this number generation idea too but an inherently search based method was first proposed by Turing. Amazingly, he already used the word "computer" for this! But initially the Turing machine name became accepted.
Then as real computers started to become household items, even the word effective was suggested to be replaced by computable. I regard this stupid and misleading.
Actually, the computer word in the personal computers is paradoxical already.
I wonder if a mother ever asked her teenage daughter: "Why are you always in front of that damn computer when you have nothing to compute?".
But this joke has a serious side. Namely how we started this whole article. With texts.
Turing machines are made for texts and so the practicality of household computers as text containers merged with the purely theoretical.
Maybe the home computers as "picture containers" will have some deeper meaning too.
Turing machines are text alterers letter by letter and this alteration process can run forever or stop by some purely inner condition of the machine. This already shows an ignorance of the result because not a decision about this result will determine to stop or not, as would seem logical.
But the second even deeper ignorance of the results is that as text collection by a machine, we will not collect the results, which would seem logical, rather the input texts where the machine stops.
So the inputs are simply "recognized" by having a result at all.
Our symbolism will be $T\downarrow M$ meaning that from a T start or input text, the M machine stops.
But saying it as T halts in M.
The results may still be important so we'll also use $T\downarrow M = R$ to mean halting on the R result.
The never stopping, so running forever is written as $T\uparrow M$.
So Turing machines are fundamentally "recognizers".
This is the most suitable for effective collections, that is as effectivization of the set concept.
In sets we are not interested in order and the machine recognizable texts are also without any order. We supply the inputs as we wish. The machine just tells if it is recognized.
But if not, it just runs forever so we have no opportunity to ask for more inputs.
This seems like a flaw in the whole idea but not really. This input supply is an external problem.
If we imagine infinite many copy of a machine and each starting from one of all the possible texts as inputs then we will get all recognizable texts recognized and we can collect them.
What's more, every single machine can be extended to a text generator because these machines have an infinite memory! So the machine itself can generate all texts increasingly and try out each as input. But now the previously mentioned strangeness that actually for the not collectable inputs our machine will or rather "would" run forever is a real problem. Indeed, how can we get to the next generated input. Luckily there is a heuristic solution called "dove tailing".

After it creates the next input, it only does one alteration step on that. Then returns to the earlier ones and does one more step on all of them. Eventually all inputs will be altered forever or up to their stops. So this generator machine will have a sequence of finite or infinite many stops and can display the recognized inputs as the generated results. Here of course it seems even more stupid why we regard the inputs and not the outputs as results.
The decision to regard the effectively collected texts as the ones “recognized”, that is as inputs that lead to a stop of a machine shows its correctness only in the long run.
In spite of this, already the above needed far from obvious tricks to create generators, reveals that generation is a fundamental sideline of recognition.
A crucially useful hidden feature in generation is the lengths of texts.
We don’t feel any absurdity in generating our texts in increasing order and yet the so emphasized feature of recognition as potentially infinite search shows that such length increasing generations must be very primitive. Indeed, observe that in such listings of texts, every time we surpass a length we can at once list the missing ones of that length too. So the complement set is generable at once. At recognitions with potentially infinite examinations, the whole point is that the complement is not expected to be recognizable. Indeed, why would all the failing recognitions be somehow just be recognizable by a single method. This natural expectation of non recognizable complement of course implies the same for generations and so:
Truly general generation should have no generable complement and so must be non increasing.
The text of the Bible might be generated before the word “yes”.
The most important common feature of recognition and generation is that they are much more mechanical than the rule systems that would seem to be the most natural examples of effectivity.
Games, equation systems, geometrical constructions and derivations in axiom systems are all “free” applications of fix rules and thus seem to be much wider than some mechanical method.
But observe the crucial point that the options are only finite. And so we can go through them in any arbitrary order. The situation is also confused by the fact that mathematicians or chess champions make intuitive decisions and this is regarded as relating to the free choices.
But this is totally false! The free steps of the rule applications are never the objects of thoughts!
These rules are acting in a mechanical subconscious level in the creative thinking.
Creativity is subconscious but bumps into consciousness! The rules don’t.
In math, the exact rule choices only come in when the idea of a proof is worked out and gets published. The exactification, the cleaning of the proofs was thus crucial in math because it brought into consciousness the rules themselves.
Gauss made a dozen proofs for the Fundamental Theorem Of Algebra because he was never happy. The cause of his real frustration was that he had no rule system yet, so even an aimed exactness was foggy.
Now that the rules are set, mathematics sank into a lower level. But this is temporary!
The rules are imperfect and something even grander is missing than were discovered by Cantor or Turing. Most mathematicians don’t believe in this and play chess in the present rule system.
But they are still creative or quite simply, they “think”. If these mathematician would think in terms of the axioms and rules of logic then they always had only a finite dilemma. But thinking is infinite in a substantial sense that we don’t understand yet. The parrots who can not think about mathematics, still think because they are humans. They think infinitely about how to cover up their stupidity. So these unknown infinities of thinking don’t alter the fact that a mechanization of the “free” derivations is true and thus for example all theorems are generable. Even those that yet haven’t been discovered. This sounds paradoxical but regard chess again. Though the champions are not thinking in terms of possible next steps, we can generate all possible step sequences.
The chess machines do this and thus make no mistakes. But a real machine has only a given memory to think ahead for a limited number of steps. So the deep question is whether using big enough computers, that is increasing this step number, could a machine outsmart human intuition.
I think so and yet this doesn’t alter my previous belief that thinking is a new infinity.
It merely proves to me that chess is a very limited field of thinking.

Four heuristic arguments to prove Gödel's Incompleteness result

The repeatedly emphasized point is that there are generable sets with non generable complement.
This can show in four ways why there have to be undecidable statements in a Number Theory that contains multiplication. The first will be straight out using only this assumption. The second will build the assumption into our Number Theory. The third pushes this even further and finally the fourth will actually not even mention this as assumption. It will be hidden in Number Theory.
This fourth is the original Gödel road so we will also see at once why it took some time for the real "point" to surface. By the end of this article we'll see a deeper point behind the non derivable complements. It will be the true conquest of Effectivity over Undecidability or Incompleteness.
But these four basic arguments are already giving more than the rubbish that has been spread about Gödel's result in the last eighty years.
The four arguments will use two possible assumption options.
One is that our axioms and logically derived consequences, that is our theorems are all true among the natural numbers, while the other is that these avoid contradiction, which is also called as they are consistent.
The first three arguments use as vital step an assumed non generability of a statement set.
In the truth assumption direction this statement set will be the set of all truths.
In the consistency assumption direction this set will be the set of the non derivable statements that is the non-theorems.
We'll have two alternatives to prove our first assumption and the second will give the hint to the third. Finally, this third will give the hint towards the fourth.
The end argument in the first three is only two kind as follows:
First heuristic argument:
Suppose that the truths are not generable and our theorems are all true.
Since our theorems are generable, thus they can not be the full set of truths.
So some truths are not theorems. These then must be undecidable statements because their negatives are falsities that we don't derive either by our assumption that the theorems are all true.
Second heuristic argument:
Suppose that the non-theorems are not generable and we have no derivable contradiction.
Now our argument is indirect:
An undecidable $A, \neg A$ statement pair means that neither is derivable.
The negative of such existing thus means two options.
Either for some A we can derive both $A, \neg A$ or for all A exactly one of them is derivable.
The first option is excluded by our assumption of consistency, so we only have to refute the second. We assumed that the non-theorems are non generable. But with this option they would be by simply generating the theorems and then apply a simple negation.
We used in both directions, that the theorems are generable.
This is trivial if the axioms are generable.
Now we must prove the first assumptions in our arguments.
To get right to the point, the essence is in multiplication!
Either in its truths among the naturals or as a derivable operation from some axioms.
This is part of the Peano rules that derive all cases of addition, multiplication and exponentiation.
Addition in itself, is primitive in both directions!
So both the truths and the rule consequences are primitive in the sense that the negatives, that is both the falsities and the non derivabilities are generable too.
To prove this though was not primitive at all! It was done by Presburger just a year before Gödel realized why multiplication is a crucial jump and thus exponentiation is actually unnecessary!
Since Presburger was not thinking in effectivities yet, his whole proof was centered in showing "completeness" that all statements are decidable.
From the six axioms three describe the $x \lhd z$ consecutive relation.
Two derives the addition cases: $x \lhd z \rightarrow x + 1 = z$, $x + (y + 1) = (x + y) + 1$.
But the sixth crucial one is actually an infinity of axioms, the so called induction axioms.
It claims the $\forall x P(x)$ universality statements if $P(1)$ and $P(n) \rightarrow P(n+1)$ is proved.

Looking from outside, this is trivial by $1 \lhd 2 \lhd 3 \lhd \ . \ .$ being a single sequence.
To understand the depth of this axiom we must realize an amazing phenomenon behind the first three consecutiveness axioms. This strangeness was known for almost ten years already and was the real initiator of all later big results. The three axioms seem very simple and ordinary!
The first claims that every number has a unique next.
The second that every number except 1 has a unique previous.
The third that 1 has no previous.
Looking from outside again at $1 \lhd 2 \lhd 3 \lhd \ . \ .$ these three should be perfect.
And yet imagine that we add to our numbers infinite many new ones denoted as:
$+1 , -1 , +2 , -2 , +3 , -3 , \ . \ . \ .$
This doesn't reveal much but put them under our real naturals ordered as the integers:

$$1 \lhd 2 \lhd 3 \lhd \ . \ .$$
$$. \ . \ . \ \lhd -2 \lhd -1 \lhd +1 \lhd +2 \lhd \ . \ . \ .$$

Quite unbelievably, our three axioms remain true for the totality of these objects:
Every number has a unique next.
Every number except 1 has a unique previous.
1 has no previous.
What a disaster! But the worst comes now! We might think that accepting the other axioms including the infinite many inductions, surely these crazy numbers should disappear.
Not so! They only become more complicated.
All this would suggest that expecting a perfect description by axioms is hopeless and instead we should find statements that are true in these strange extensions but false among the real naturals.
But such were not discovered and then Pressbuger's result that with the induction axioms we can derive all truths about addition, gave the impression that maybe these weird models of the naturals are simply mirages that possess no new truths just other realizations of the ordinary ones.
Gödel realized that this is not so if we allow multiplication!
Then these weird realities will contain weird truths too!
So we would expect that he went into these non standard models to find non standard truths.
But instead he went into ordinary multiplication!
Multiplying numbers in itself is still very primitive!
$2 \cdot 15 = 3 \cdot 10 = 5 \cdot 6 = 30$ and so these compositions are not recoverable from 30 as result.
But using remainders, we can assign to arbitrary many $r_1, r_2, \ . \ . \ , r_n$ numbers a pair of c , d codes, so that the r numbers are decodable back from c , d. So we have an $F(c, d, i) = r$ expression, using multiplication and logical symbols, so 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 by F.
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 an $(r_1 , r_2 , \ . \ . \ , r_y)$ tuple that $r_1 = x , \ r_2 = x^2 , \ . \ . \ , \ r_y = x^y = z.$

Here we still have the dots and y as subscript but we can avoid all that by saying instead:

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

$\exists c , d$ means that "there are such" codes and $\forall k$ means that "for all" k.
Thus the first claim in the big bracket tells that the sequence $r_1, r_2, \ . \ . \ , r_n$ starts as x, the second that the members always multiply by x and finally the third that the last member is z.
So indeed we got exponentiation! We can also see that all similar rule definitions can become finite formulas. So then we can also see that all effective number collections can be expressed as mere $P(x)$ logical formulas using addition and multiplication.
Now if we have a framework of number generations where we can prove that there are generable sets with non generable complement, then this will also translate to formulas.
So we have a $G(x)$ that $Gntrue = \{ n ; \ G(n)$ is true in the naturals $\}$ is a generable number set but the complement, that is $\neg Gntrue = \{ n ; \ \neg G(n)$ is true in the naturals $\}$ is not generable.

Then of course Gtrue = { G(n) ; G(n) is true in the naturals } is a generable statement set, and $\neg$Gtrue = { G(n) ; $\neg$G(n) is true in the naturals } is a non generable statement set.
This implies at once that the set of all truths must be non generable too. Indeed, if they were, then among them we could recognize the $\neg$G(n) formed ones and generate only these.
So we obtained the condition of the truth direction and proved incompleteness.

Now we give an alternative proof for the truths not being generable without assuming that an external framework proves the existence of non derivable complements.
Suppose that from some basic relations we form logically built ones including P(x) properties.
If for all generable **S** number sets we have some Q(x) property that: $n \in \mathbf{S} \leftrightarrow$ Q(n) true then the true statements are not generable.
We'll assume $< P >$ code numbers ordered to the P(x) properties.
We regard the **D** = { $< P >$; P($< P >$) true } diagonal number set.
It's complement $\neg \mathbf{D}$ = { n ; $n \neq < P >$ or ($n = < P >$ and $\neg$P($< P >$) true)} is not generable!
Suppose it were and thus were represented by a Q(x). Then:
$\neg$P($< P >$) true $\leftrightarrow$ $< P > \in \neg \mathbf{D} \leftrightarrow$ Q($< P >$) true
But using P as Q we get a contradiction.
The non generability of $\neg \mathbf{D}$ as code set implies the non generability of the $\neg$P($< P >$) formed true statements used in $\neg \mathbf{D}$ too and this implies the non generability of all truths.

This alternative proof was not a waste of time because it gives the template to get the proof for the assumption in our second argument at the start. That the non-theorems are non generable.
Indeed, if instead of truths, we regard derivable statements using some axioms so that a derivable representation of the generable sets is true as: $n \in \mathbf{S} \leftrightarrow \vdash$ Q(n).

then the diagonal set is now **D** = { $< P >$; $\vdash$P($< P >$) }

and $\neg \mathbf{D}$ = { n ; $n \neq < P >$ or ($n = < P >$ but $\neg \vdash$P($< P >$))}.

And this second is again non generable.
Indeed, if it were, it were derivably represented by a Q(x) and then for all P we would have:
$\neg \vdash$P($< P >$) $\leftrightarrow$ $< P > \in \neg \mathbf{D} \leftrightarrow \vdash$Q($< P >$).
But using P as Q we get a contradiction.
The non generability of these codes imply the non generability of the $\neg \vdash$P(<P>) cases, that is the P(<P>) formed non-theorems which imply the non generability of all non-theorems.

Now comes the fourth direction where we will not represent generable sets at all.
Instead, we will represent the derivations themselves. We assign again to every P(x) property a $< P >$ code number so that a D(x,y) relation can represent the property case derivations:

$$\vdash P(n) \qquad \leftrightarrow \qquad \vdash D(n , < P >)$$

Then if our system is consistent there is undecidable statement pair in it.
Now that diagonality is pushed inside our axiom system there is a temptation to use it instantly for the D derivability relation and so claim that D($< \neg$D(x,x) > , $< \neg$D(x,x) >) and its negative are an undecidable pair.
Which indeed can be seen by using $\neg$D(x,x) as P(x) and n as $< P(x) > = < \neg$D(x,x) > :

$$\vdash \neg D(< \neg D(x,x) > , < \neg D(x,x) >) \qquad \leftrightarrow \qquad \vdash D(< \neg D(x,x) > , < \neg D(x,x) >)$$

If both sides are true then our system is inconsistent which we assumed not to be true.
So both sides must be false, meaning exactly that the two statements are an undecidable pair.
This was didactically bad on two levels. Firstly, it was as a totally ad hoc trick but it also hid completely the earlier reality of non generable number sets. But there is a much better approach.

We can say that two P(x) and Q(x) properties share an n number derivably if:

$$\vdash P(n) \quad \leftrightarrow \quad \vdash Q(n)$$

Obviously, if our axiom system is consistent then a Q and its negative $\neg Q$ can only share an n derivably if neither is derivable at n and so these cases are an undecidable pair of statements. Thus to show that there is undecidable pair, it is enough to exhibit a Q property that shares derivably some n number with every P property. Indeed, then $\neg Q$ must be among these P-s and so the derivably shared two cases are an undecidable pair of statements.
Then we can reveal that:
1. Such Q is D(x,x). and 2. The derivably shared value with P is <P>.

Indeed: $\vdash P(\langle P \rangle) \quad \leftrightarrow \quad \vdash D(\langle P \rangle , \langle P \rangle)$

Other directions of Effectivity

The heuristic idea of running forever and thus not collecting a text, was potentially present in classical Number Theory too. We can say that collect those n numbers for which "this and this" is true and we can not be sure for concrete n numbers whether the P(n) property that exactifies the "this and this" stands or not. But deeper examinations of P will not be so much different for P or its negative $\neg P$ and so either both are unpredictable or both will be effective.
A bit more promising is to say: collect those n numbers for which there is "such and such" other N number. If the R(n,N) relation that exactifies the "such and such" is complicated enough then we shouldn't be able to predict the existence of N and so we actually have to search all numbers.
I talked about all this in the finally inserted first section.
The sure way to make non effective complements was Gödel's discovery with codings.
To formalize free derivation systems without logic and language was a goal of Gödel too.
He finally succeeded with using some ideas of Herbrand. And yet this theoretically so clean system didn't bring the nice consequences as Turing's "primitive" text alteration or Kleene's "singular" first number search. Most importantly, there is a crucial negative feature of Gödel's, Kleene's or of the first effective method found by Church. They all use number functions.
This could be regarded first as something heuristic. Objects obtained from objects.
And actually, even Turing's method seems to go toward this because the text alteration temptingly offers the altered text as the "result". But his heart of the matter idea got out of this temptation and regarded the original altered text as the one to be collected.
But even if in a field "object to object" is heuristic, as we'll see such in the next section, the other systems would be still ugly because they use "number" functions. And this is still not quite the point. Kronecker regarded the natural numbers as the God given basic reality.
Now we can see why he was not only immoral with Cantor but had a false idea about God too.
The two may have had a lot to do with each other. But one thing is sure. Numbers in themselves can not give new numbers. Only more numbers can produce a new number and this is evident by the basic operations that give a new number for any two old. Combining these then we can get a new number from more old ones as $f(x_1, x_2, \ldots, x_n)$ functions.
So every combining of the basic operations has a certain n "arity" or input number.
This is the root of ugliness in using functions and it is due to numbers being individual blocks that can not melt into each other. So even if we avoid functions and regard $f(x_1, x_2, \ldots, x_n)$ as what it truly is, an $(x_1, x_2, \ldots, x_n, y)$ collection of tuples with merely the last elements y-s being unique, we still have this $n + 1$ "arity" on our back. So now I will offend all Number Theorists and make Gauss turn in his grave but say out loud: Natural numbers are definitely not God given fundamental blocks! They are subjective and very imperfect blocks. This is very clear from Number Theory itself! Primes, the so mysterious chain of pearls in the God given blocks show this too. Fermat's Last Theorem is a big exclamation mark to show us how irrelevant these are.
And don't tell me that prime exponents are enough because this is just a triviality.
But I don't want to elaborate on these because have a much more convincing line and a prediction.

Paul Erdös said about a problem that “mathematics is not ready for this yet”.
So first my prediction: This problem has nothing to do with primes just as Fermat’s Last Theorem.
But here as I said, there is a line that shows why the solution will not even be in Number Theory.
So here is the problem that actually starts as a parlor trick. Think of a number!
If even, divide it by 2 as many times you can to get an odd. If odd then multiply by 3 and add 1 to make it even. Then again make it odd by the first method and then again even by the second and so on as long you can go. I promise you will not have to do these infinitely because you will get down to 1 and so get into the $1 \rightarrow 4 \rightarrow 1$ only cycle.
The literature is filled with number theoretical examinations of these so called “hail stone sequences” that come about with different starting numbers but always “fall down” to 1.
Now regard this. Write the initial number in base 2 as a binary 1001110100.
The divisions by 2 to make the number odd is now a simple cutting off all 0-s from the end.
So we get 10011101 a “1 to 1” sequence of 0-s and 1-s. Remember this for later when such same texts will come in with Turing machines. But now just check what the “evening” does.
Instead of multiplying by 3 and adding 1, we can multiply by 2 add 1 and add the number again. Multiplying by 2 is of course just placing a 0 at the end, so with placing the added 1 instead, we merely have to place a 1 to the end: 100111011. Finally, the only non trivial alteration is calculating 100111011 + 10011101 or rather

$$\begin{array}{r} 100111011 \\ \underline{10011101} \\ ?????????? \end{array}$$

The last digit of the result is definitely 0. If there is more at the end it’s even better because we’ll cut these off and start again. The claim is that our “1 to 1” numbers will become 1.
Observe that these can only grow if the addition overflows. Indeed, we only added one 1 and will definitely cut off a 0. And overflow comes about if and only if in the extended number with 11 at the end, the first 11 is such that there is no 00 before.
This might even sound as an instant solution but follow it and you’ll see differently.
I rest my case and hope you’ll agree that this problem is not about numbers, rather about texts.

Randomness, the second new essence beside Effectivity

Let’s return to the ugliness of arity or number tuples. Avoiding functions and trying to derive only tuple sets does give a certain clarity that I will explore a bit in the next section but first I want to mention that the texts being heuristically superior to numbers did come to light in the object to object world too. Kolmogorov realized that the text alteration results can be regarded as chosen texts. Then we regard all possible alterations with all possible starts and find the one that gets that result from a shortest possible start. This shortest input text then could be regarded as a compression of our result text. Indeed, the machine can produce our text from that shortest.
The length relation of a text and its compression can be interpreted as a compressibility of a text.
This means a simplicity of a texts, or in negative the non-compressibility means a complexity.
One million alternating 0-s and 1-s can be described by only 33 symbols namely as “one million alternating 0-s and 1-s”. That’s not a joke, it is a perfect idea and the problem is not that we used English, rather that we have to allow all machines. So then even if we regard two million 0-s or 1-s randomly instead of our alternating example, a machine can have these in its memory and voila get them in a single step making them seem simpler than the alternating 0-s and 1-s.
But this “glitch” will not defect the basic grand idea for two reasons.
Firstly, because we can use a universal machine and secondly because we’ll go towards using this complexity of texts for beginnings of an infinite text to decide if it is random or not. There was still a second “glitch” even in this idea, recognized by Martin Löf. So he offered a different approach to define Randomness and then the glitch was solved and turned out to give the same definition as Martin Löf’s. Halleluiah praise the Lord! Well not quite yet! If these definitions are examined more closely, alternative definitions come about that are not identical and all can pass as Randomness without trivial defects. So Randomness splintered. This is not a glitch anymore.
It is a sign of something that we don’t know yet. So the title of this short section makes sense.

Derivable "Cases"

Now we can come to the promised little detour to see that derivable tuple sets, in spite of the arity ugliness are nicer than deriving functions. The title is important and reflects that something more general is here than collecting number tuples. Indeed, every theory has basic relations but also fixed basic objects too. We can call these the names, though usually they are called constants.
This idiotic name refers to that most treatments allow basic functions too and when these are not really dependent on variables because there are no variables, then they are fix. But we shouldn't allow basic functions only names.
Here I strangely seem to agree with Kronecker and so regard the generation of the natural numbers as 0 , $S(0)$, $S(S(0))$, . . by a successor function as not just stupid but deceptive.
It's much nicer to admit we have an infinity of names 1 , 2 , 3 , . . as natural numbers and regard the $x \lhd z$ consecutiveness as basic relation and thus $1 \lhd 2$, $2 \lhd 3$, $3 \lhd 4$, . . as axioms.
My nephew who just enrolled to the same math high school that I went more than fifty years ago when it started, told me last week that zero is a natural number. He said it so vehemently that I saw he had no clue why they implanted this dogma in his head. There is an anecdote about Peano who started this dogma. Once at the opera when they gathered their coats after the show, his wife said "did you get all pieces?". Peano said "yes, zero, one, two, three". The wife looked at the four coats and said "and you are the mathematician". I agree with his wife and will start the naturals from 1. The Peano rules of course can be stated just as well.

$$x \lhd z \;\rightarrow\; x + 1 = z$$

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

$$x \cdot 1 = x$$

$$x \cdot (y + 1) = x \cdot y + x$$

$$x^{1} = x$$

$$x^{(y+1)} = x^{y} \cdot x$$

Avoiding functions and regarding $x + y = z$, $x \cdot y = z$ and $x^{y} = z$ as relations just as $x \lhd z$, is easy. In fact, the rules can be stated as this:

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

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

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

The lines mean $\vee$, so the different ways to get the target relation after $\rightarrow$, while the members in a line are $\wedge$ so show all the conditions necessary to get the target.

The name axioms are of course also usable and most importantly, also any already derived cases of the target relation. The derivation of $4 + 3 = 7$ is this:

$$4 \lhd 5 \quad \wedge \quad 1 = 1 \qquad \rightarrow \qquad 4 + 1 = 5$$

$$4 + 1 = 5 \quad \wedge \quad 1 \lhd 2 \quad \wedge \quad 5 \lhd 6 \qquad \rightarrow \qquad 4 + 2 = 6$$

$$4 + 2 = 6 \quad \wedge \quad 2 \lhd 3 \quad \wedge \quad 6 \lhd 7 \qquad \rightarrow \qquad 4 + 3 = 7$$

As we all know from elementary school, multiplication is repeated addition that is:
$x \cdot y = z$ means $x + x + \ . \ . \ . \ . + x = z$
Similarly exponentiation is repeated multiplication.
The problem is of course that these dots are not exact. Neither as explicit forms nor as exact rules what to calculate. That was the whole motivation for the Peano rules.
The hidden side is that these are still not explicit so don't give the target as defined by earlier relations because the target also appears in the conditions.
Though we don't allow this in explicit definitions, we do allow this in axioms.
So these Peano rules should be regarded as simply axioms from which we get case derivations that is fully concretized name "instances" as the proper expression goes.
And this is indeed fully general for any theory. We can regard for any $B (x , y , \ . \ . \ . \ , z)$ basic relation the derivable $B (N_1 , N_2 , \ . \ . , N_n)$ cases as a $[B]$ set of name tuples which is thus the effective meaning of B by the theory.
The heuristically beautiful side of this definition is that it instantly contains the so important one sidedness of derivations. Meaning that a complement set $\neg [B]$ doesn't have to be $[\neg B]$ at all.
It doesn't even have to be $[B']$ for some other B' basic relation.
To make this sure, that is to find a $[B_0]$ that $\neg [B_0]$ can not be any $[B']$ would be crucial.
Unfortunately, a fix axiom system becoming a good framework for Effectivity by this heuristic case collection or $[B]$ effective meaning is obviously impossible, simply because we have infinite many effective collections and only finite many basic relations in a theory.
The solution is to allow an infinity of basic relations.
Remembering that we want all possible effective collections then suggests the following:
We should allow all possible statements, containing only formal $R (x , y , \ . \ . \ . \ , z)$ relations as targets. So only the defining statements should mark their target relations.
A fundamental problem is that a contradictory statement would make everything derivable.
So we need an avoidance of contradictory defining statements.
A case in itself is never contradictory only if the negated relation contains the same case.
So if these target relations are never negated we are okay. This of course implies that we have to be careful with quantors and logical operations too. The $\wedge$ and $\vee$ operations have a simple double layer as necessary minimum, which beautifully can be used already as the situation matrix of any prenex, that is frontally quantized statement.
There we simply write the $\wedge$-s with commas as lines and the lines are the $\vee$-s.
Our above conditions were exactly like this, except we omitted the commas too.
Using the single "effective implication" from such matrix to a single relation as consequence, will at once solve all our problems! It designates the target as this consequence and interprets the quantor usage too. Namely, the target variables $x , y , z , n , \ . \ .$ are all universal while the rest of the variables $u , v , w , k , \ . \ .$ are all existential. Thus a rule:

$$\left.\begin{array}{l} x , y , z , \ . \ . \\ \\ u , v , w , \ . \ . \end{array}\right] \quad \rightarrow \quad T (x , y , z , \ . \ .)$$

actually means: $\forall x \forall y \forall z \ . \ . \ [\ \exists u \exists v \exists w \ . \ . \ (\text{cond}) \rightarrow T (x , y , z , \ . \ .) \ ]$
Where (cond) is the exact $\wedge$ and $\vee$ form of our conditions.

So now all our rules above fit into this bigger picture. They are now not the axioms of the theory which is Number Theory in this case, rather part of this new extension.
They may also appear as proper axioms by coincidence.
As an interesting situation let's regard the $<$ relation. This is explicitly definable by addition as: $x < z \ : \exists y \ (x + y = z)$ but it is also definable by the effective rule:

$$\left.\begin{array}{c} (x \lhd z) \\ \\ (x < w) \quad (w \lhd z) \end{array}\right] \quad \rightarrow \quad x < z$$

The exact logical form for this rule by the aboves is:
$\forall x \ \forall z \, [(x \lhd z) \vee \exists w \, ((x < w) \wedge (w \lhd z)) \ \rightarrow (x < z)]$
But observe that the explicit definition implies this:
$\forall x \forall z \, [\, x < z \ \leftrightarrow \ \exists y \, (x + y = z) \,]$
This could be used to derive cases too and could give different ones than we get by the above rule.
So we should have used different $<$ symbols for the explicit and the marked effective one until we prove that they are the same.
We could now define repeated exponentiations and then that could be repeated again, and so on.
But a much smarter thing is to define all possible $x \ [n] \ y \ = \ z$ relations as quadruples.
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.
The fundamental rule is quite simple as $x \, [n] \, y = (x \, [n] \, (y - 1)) \, [n–1] \, x$. Indeed:
$x \cdot y \ = \ x \cdot (y - 1) + \ x$ and $x^{y} = x^{(y-1)} \cdot x$
The $x \, [1] \, 1 = z$ relation is simply $x \lhd z$, so this as condition is enough.
If $x \, [n] \, 1 = z$ with $n \neq 1$ then $x = z$ so this as condition is enough.
Thus the final rule system is easy as:

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

The added beauty is that this is independent of the earlier basic operation. Indeed:
Multiplication used addition and exponentiation used multiplication as earlier targets.
Observe the followings:
To increase the $[4]$ value from v to y we must raise the old result to the x power.
So $x \, [4] \, 1$ is x by the second line and then $x \, [4] \, 2$ is x^{x}.
Then $x \, [4] \, 3$ is $(x^{x})^{x} = x^{(x^2)}$. But we shouldn't write simply $x^{x^{x}}$ for this because the $x^{(x^{x})}$ value is different and could be understood for it just as well.
This is a consequence of exponentiation not being arbitrary in its order unlike addition and multiplication. 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 shouldn'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. This is actually a mystery!
Observe something else! For the three normal 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 grows with x and y. This caused big havoc at the early recursion period of Effectivity.
Then the normal operations were first generalized in a different direction as f functions that are given initial values and then defined for step by step increasing inputs.
These "primitive recursive" functions can not grow as fast as our $x\,[n]\,y$ and this was regarded as a complexity of $x\,[n]\,y = z$. In truth, this as relation 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. Indeed, at $<$ for example, we can merely have the axioms $1 \lhd 2\,,\ 2 \lhd 3\,,\ 3 \lhd 4\,,\ .\ .\ .$ as derived 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 all our finite many defining conditions. Obviously as start we'll only be able to use these numbers in the $\lhd$ basic relations there. Then we get some target tuples and we again only regard the ones using values up to m. These can now be used in our second try of all conditions. We again get new tuples with values up to m and we use these again. Repeating this, we get a stage where no more new targets can be obtained. Simply because we couldn't get infinite many target tuples by using only numbers up to m. So we derived all target tuples with using values up to m.
Observe that this m will also be the maximal value in all of our derivable targets above.
Indeed, in every target in our rules above, the maximal variable value was always at least as big as other values in the conditions. Namely, the maximal was always z, except at $x\,[n]\,y = z$.
And here if n is the same or bigger than z then the only variable not up to z in the conditions is k but it is under n.
m appearing as maximal in the targets then implies that a target with values up to m can not come about by increasing m to an M. Simply because then M will be the maximal.
This means the same that the tuples not in our targets up to m will remain outside tuples as we increase m. So we can derive 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 check all tuples formable up to m and list the ones not derived.
This is a bad news because we aimed to find effective collections without effective complement.
A simple solution jumps to mind. Let's destroy this feature that our targets were so maximal.
We can simply omit some target variables and thus make them merely condition variables.
So now we allow $u\,,\ v\,,\ .\ .$ as earlier target variables and then the rule is simply:

$$R\,(\,u\,,\,v\,,\,.\,.\,.\,,\,x\,,\,y\,,\,.\,.\,)\ ] \quad \rightarrow \quad T\,(\,x\,,\,y\,,\,.\,.\,)$$

In logic sense this means that from an earlier R target we obtained $\exists u \exists v\ .\ .\ R$ as T.
Can this be enough to get non effective complements? Or an other question is this:
Above we saw that our rules were such that the complements were all effective.
Are there deriving rules at least for these complements too?
And here we have again a very general solution. Namely, if an $F\,(\,x\,,\,y\,,\,.\,.\,.\,,\,z\,)$ relation is functional for its last z variable, that is for any $x\,,\,y\,,\,.\,.$ values z is unique then

$$F\,(\,x\,,\,y\,,\,.\,.\,.\,,\,w\,)\ ,\ \ w \neq z\ ] \quad \rightarrow \quad T\,(\,x\,,\,y\,,\,.\,.\,.\,,\,z\,)$$

will define a T target that is actually $\neg F$.
Observe that the condition of all possible other variable values was crucial!
Without this we would derive correct missing values for $\neg F$ but not all of them.
Indeed, the missing $x\,,\,y\,,\,.\,.$ values with any z would not be obtained at all.
If however F is a total function, defined for all $x\,,\,y\,,\,.\,.$ variables then such missing tuples simply don't exist. Now we should see two practical sets from Number Theory.
First the composite products can be at once defined by multiplication with added conditions:

$$x \cdot y = z\ ,\ \ x \neq 1\ ,\ \ y \neq 1\ ] \quad \rightarrow \quad x \cdot y = z\,\mathrm{comp}$$

This is just like multiplication so its target contains the maximal tries and thus its complement is effective. It means $x \cdot y \neq z$ comp so dropping out the x, y variables would not give the primes and 1 rather all values because any z can be a false multiplication value.
Of course, we can drop out the x, y variables from $x \cdot y = z$ comp as:

$$x \cdot y = z \text{ comp }] \quad \rightarrow \quad \text{Comp } (z)$$

The previous failed attempt can be repeated by the false idea that:

$$\text{Comp } (w)\ ,\ z \neq w\ ] \quad \rightarrow \quad \neg \text{Comp } (z).$$

This would just give all z values again.
To derive the primes, we need to negate all compositions of z from numbers under z:

$$1 \cdot 1 \neq z\ ,\ 1 \cdot 2 \neq z\ ,\ \ldots\ ,\ (z-1) \cdot (z-1) \neq z\ ] \quad \rightarrow \quad \text{Prime } (z)$$

So the dots that our rules tried to avoid, sneaked back! Luckily, we can avoid these too with an other tricky rule. Indeed, from an R we can obtain the $\forall\ u < x\ R$ relation as T with the rule:

$$\left.\begin{array}{c} x = 1 \\ \\ T(u, y, \ldots, z)\ ,\ R(u, y, \ldots, z)\ ,\ u \lhd x \end{array}\right] \quad \rightarrow \quad T(x, y, \ldots, z)$$

Using this first for $R = (u \cdot y \neq z)$ we get $T(x, y, z) = \forall u < x\ (u \cdot y \neq z)$, and then for this as R with y we get $T'(x, y, z) = \forall u < x\ \forall v < y\ (u \cdot v \neq z)$.
And thus $T'(z, z, z)$ means Prime (z) . The $x \cdot y \neq z$ must of course first be derived.
So the full set of rules for the primes is this:

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

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

$$x \cdot y = w, w \neq z\ ] \quad \rightarrow \quad x \cdot y \neq z$$

$$\left.\begin{array}{c} x = 1 \\ \\ T(u, y, z)\ ,\ u \cdot y \neq z\ ,\ u \lhd x \end{array}\right] \quad \rightarrow \quad T(x, y, z)$$

$$\left.\begin{array}{c} y = 1 \\ \\ T'(x, v, z)\ ,\ T(x, v, z)\ ,\ v \lhd y \end{array}\right] \quad \rightarrow \quad T'(x, y, z)$$

$$T'(z, z, z)\ ] \quad \rightarrow \quad \text{Prime } (z)$$

Turing's rewriting system from under-writing

In his 1936 article, Turing clearly explains that he went back to Elementary School ideas.
And indeed, when kids do the digit by digit algorithms for the basic operations, then they use a squared paper and by writing single digits into them can obtain the correct results.
Now if we can obtain these operations by such primitive methods, then we can reverse this whole idea and ask what operations can be obtained by a generalized system of such digital calculations.
Can it be that by being general enough we can obtain everything that is effectively calculable?
The answer is yes and the amazing fact is that this generalness needs very simple possibilities.
Instead of the learnt rules how we add or multiply single digits and write them under each other, and so on, we can use a rectangular table of triplets that tells everything.
On one side, this is simpler but by allowing arbitrary big tables we still grasp more.
As start, we allow the ten digits $0, 1, \ldots, 9$ to be m many $0, 1, 2, \ldots, m-1$ possible digit values but this is not the one that allows bigger tables.
Merely a convenience for later when we can replace these with arbitrary m many symbols.
These m many values or symbols will be written on our squared paper and will be also positioning the lines in the columns in our table.
For a concrete number or symbol underwriting system this m is fix.
So each column will have m many lines and the counting starts from 0.
The crucial potential increasing of our tables is the arbitrary n number of columns.
Here the counting is normal so the first column is 1 then 2 and so on.
So our tables are $m\,n$ many triplets with m being fix but n arbitrary big.
The triplets themselves consist two numbers and a $\rightarrow$ or $\leftarrow$ arrow between them.
Here is a concrete table:

$1 \rightarrow 2$	$1 \leftarrow 2$	$1 \leftarrow 3$
$1 \rightarrow 0$	$0 \rightarrow 3$	$1 \leftarrow 1$

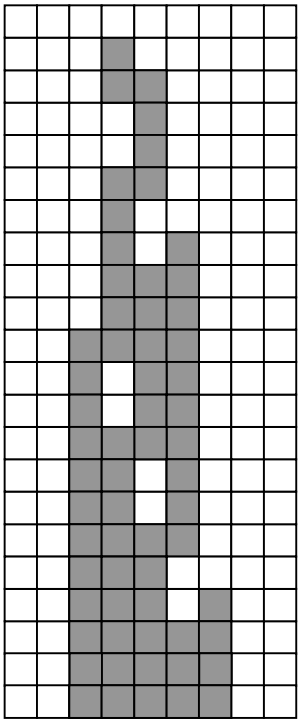
As we see, $m = 2$ and $n = 3$. The $m = 2$ means that we have only two symbols, or in simplest way two digits 0 and 1. This table will tell an action sequence carried out on our squared paper.
Namely, we will write one 0 or 1 digits in every new line under a fully filled starting line.
Now we regard the simplest scenario when our starting line is all 0-s.
So we can start from any square of this number line and use our table to write a new number in the next line. Quite logically, we use the first column of our table to start our actions.
Remember, we said that the m many number values will also denote the lines in every column.
So our table has the $0, 1$ number values as line numbers in our first column too.
Not surprisingly, they will correspond to the value in the square, under which we write.
Or put it an other way, this is the value we see above where we write.
Since we started from an all 0 line, we must use our 0 numbered, that is first line: $1 \rightarrow 2$.
The first of the triplet, 1 tells what to write in the new square.
The second, the $\rightarrow$ arrow tells that we should move one step in this direction and under this will be our new square to write in.
The third, the number 2 tells that we should now move in our table into the second column.
From here actually we repeat what we did previously.
We look up in our squared paper to see what the number value is above and go into this line in our table column. Since we started with all 0-s thus we see again a 0 above and so we go in our second column into the first line: $1 \leftarrow 2$. So we again write a 1 in the new square.
But now we move left and so we will be under our previously written 1.
Of course, we must go a line down too and use the square under as next one.
The 2 tells that we must stay in our second column. To see what line to apply, we must look again up in our squared paper and as I said we'll see a 1 so we must use the second line:
$0 \rightarrow 3$. Thus we write a 0 move to the right and under this will be our new square to write in.
The 3 tells that we should move into the third column and we'll use the second column since we see 1 above. That line is $1 \leftarrow 1$ and so we write 1 again under the already seen 1 above.

We move left and move in our table to the first column.
The squared paper by now will look like this:

. . . . 0 0 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

We might think that this would go on forever but observe that in our first column we have a 0 second number in the second row but there is no 0 column!
So this 0 means an end or stop or with Turing's original word usage, a halt.
But do we necessarily encounter this halt order?
NO! And this is the whole essence of Turing's ingenious system.
Some tables from some starting line do get to halt while others from other start don't.
In fact, this double dependence of the halt from both the table and the starting line is an other crucial feature. It reveals that some starting lines can replace the tables themselves as programs.

The under-writing action sequence is very clear because we see the whole history, what happened.
There is only one negative feature, namely that the blank squares made this possible.
We saw through these blank squares to see the last written values above.
If we want to use the empty blank content as symbol or value too and for example we chose 0 as this then we must update every new line with all the filled in 0-s or 1-s.
Then everything becomes messier, so we'll show now all the steps till halt, with white meaning 0 and grey meaning 1. Even with this, to see where we were at a step is a bit confusing.
A simple help is that if two consecutive lines differ at a place then the writing obviously happened there. Our last halting place will be not such and so to help a bit we mark it with an x:



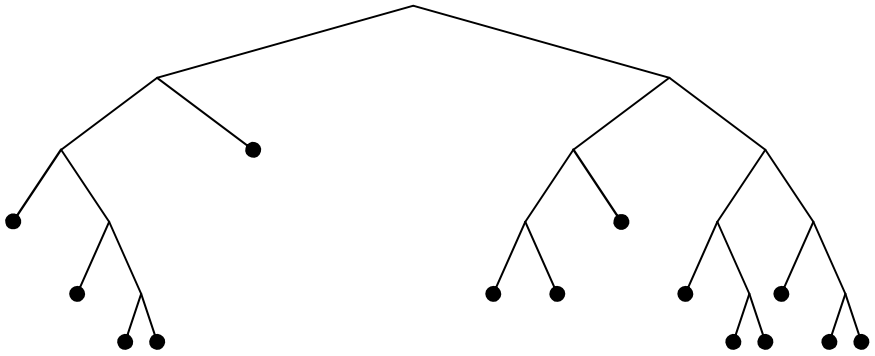
x

This updated blankless underwriting sequence was only pursued to come to the rewriting system that is the well accepted third step. Here we don't write the new symbols under the old rather we erase the old and write the new in its place. So we do not even need a squared paper just a line of squares. The elementary school solution could only be to use an eraser continually but the modern technical solution is to use a single line of memory cells!

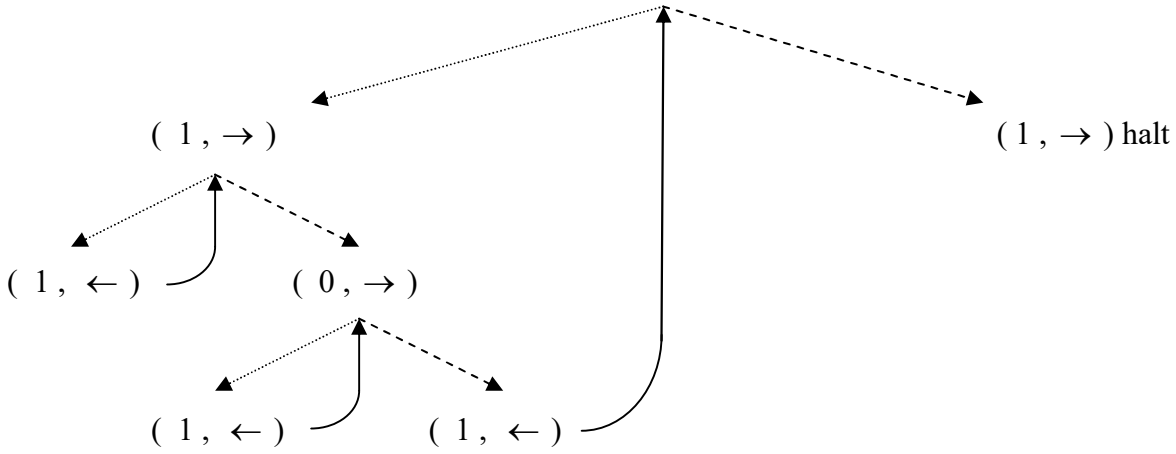
This line being infinite means that actually we have an infinite memory and yet the single line means that we do not need memory registers and addresses.
The memory already suggests the further technical steps to turn a table into a machine.
We need a table head that does the jumps in the table and a memory line head that does the rewritings and single steps left or right. We also need a communication between them so the table head can get the last read symbol and the line head can get the new symbol and move direction.

The almost unbelievable fact is that this system can imitate all real computers even if we attach to them an infinite memory.
Finally, we mention an earlier already used expression that calls the columns of a table as states of the corresponding machine.

Now we'll show an "action tree" meaning for the Turing tables.
First here is a finite graph with end points marked with dots:



Such 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.
Most importantly though, the two branching lines correspond to the read 0 or 1.
And finally, we must place the memory line actions to the branchings to get an actual action tree.
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 writings and movements, here in the parenthesis.

Text collection and universality

By defining some finite initial line segments as inputs and regarding those T text inputs that lead to halt in an M machine, or in short, regarding those T texts that halt in M and writing $T\downarrow M$, we can define the effective text collections as $\{ T ; T\downarrow M \} = [M]$.
Observe that such collection by haltings, could only be a real collection if we somehow offer all possible beginnings to the machine to be read and then halt. So we realize two fatal flaws.
Firstly, we must somehow systematically list all possible texts as inputs.
Secondly, at an input that the machine doesn't collect, that is rejects it by never reaching a halt, an infinity of alterations should happen and so we can not even offer the next input to try it.
The first is not as hard as it sounds and for example the following is a logical such line:
0 1 00 01 10 11 000 001 010 011 100 101 110 111 . . .
We listed all possible binary texts in increasing lengths and vale.
The spaces are only for us to see the possible beginnings better.
To solve the second problem, a machine can go through this list but have an internal program in its table to only perform one step on each segment and return to the earlier unfinished segments.
Since on every segment eventually arbitrary many steps will be done, thus every segment that would be recognized that is halted from in a straight alteration, can be recognized here too.

Turing's greatness was not only to grasp at once in a system all effective text collections!
There were text alteration systems using axioms and formation rules before him already.
These can also grasp all possible generable texts and so seem a much simpler abstraction of effectivity than Turing's. But they are too simple! Even the later introduced so called "grammars" that use a smart second alphabet of variables that can be any text, though lead to the so called Chomsky hierarchy, can not shed light on what Turing's system could.
Namely, in logical derivations the variable usage is more complex than simple text replacements!
So while the variables are the heart of the new non Aristotelian quantor applications, Turing's halting introduction at the bottom is the heart of effectivity as the new adequate step to grasp the effectivities of axiomatic derivations.
The recursive functions was the most unsuccessful direction of effectivity.
If we could generate them all then instantly we could make a new one not in our list by altering the function value of the n-th at the n variable value.
Similarly, the effective listing of all dually effective number collections is impossible too.
Effectivity is genuinely not dual only one sided. To be put more simply, the typical effective collections have a non effective complement. The clarification of this typical is our goal.
Turing not only found a framework for effective collections but his framework can shed light on why the crucial one sidedness of effectivity lies behind Gödel's results too.
But Turing himself did not see the steps that prove this deeper potential in his framework.
The second aim of his article, to settle Hilbert's Entscheidungsproblem was half baked too.
The halting condition as basic decider of effectivity was not used only the external universality.
And that of course is not a provable fact only a thesis.
Kleene was the one that connected his own results based on the partial recursive functions with Turing's system. But even he missed the most important final result, Rice's Theorem.
The similarity of Turing with the two other deep geniuses Cantor and Einstein is evident.
Cantor, Dedekind, Zermelo perfectly corresponds with Turing, Kleene, Rice.
The two results are the Well Ordering Theorem and Rice's Theorem but even they are disguised.
The real two gems are the f-widening Theorem and the Machine impossibility Theorem.
Without seeing this, one is just a parrot or monkey repeating the formalisms.
But now I want to explain something simpler:
Gödel only used primitive recursive functions and never mentioned effectivity as the reason behind his proof that undecidable statements exist. Instead, the language and self referring statements were emphasized. But aside from this vision change, how could the use of primitive recursive functions do the job when exactly the non effective complements are the point.
This can be explained very clearly through Turing tables offering a new fourth vision.

Remember, our first was the under-writing, then the machine as memory alterer and last the action tree as special flow chart. Now in a sense we return to the first but without the vision of the squared paper. We lose the elementary school simplicity but gain something else.
The line by line new writings will be replaced by giving the triple sequence that we used.
So our example should start as:
(1 → 2) (1 ← 2) (0 → 3) (1 ← 1) . . .
To make better sense of this we will write above the first members two facts.
First the position under our starting line. This is easy if we regard as 0 the starting square and use the negative numbers backwards while the positives forward.
Then above this information we write also the last written value in that place or the starting value.
So the triplet sequence becomes:

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

The first above written position values can be established trivially from the previous position value and the previous arrow direction. But for the second memory content we must go back to the last same position and use the there applied new writing or the initial line.
But, if we do that then the table gives perfectly every next triplet by the last third number as column number and the second above written last value as line number in that column.
So a triplet sequence as an A alteration sequence is very easily verifiable whether it is a correct one using a given table and initial line.
So a Σ supervisor machine can give a yes or no answer for any A alteration sequence, T input and maybe even a final R text result whether all was according to an M machine or table.
So if we turn the M table into an {M} input segment too and this Σ into a Turing machine then we have that: T | A | {M} | R↓ Σ = yes or T | A | {M} | R↓ Σ = no.
And now we have the same dilemma again that I raised above for Gödel's use of primitive recursive functions. Indeed, seemingly: T | A | {M} | R↓ Σ = yes ↔ T↓M = R.
So all the complex M machines are simulated by such primitive Σ.
We might think that our error lies in the many heavy assumptions to turn all factors into the input.
But the error is much simpler!
We used the A alteration sequence on the left while on the right we did not.
So actually what we have is:
[There is some A that T | A | {M} | R↓ Σ = yes] ↔ T↓M = R.
So we must search all A texts to verify the simulation. This then is a hidden task.
In Gödel's proof too the primitive recursive functions could describe the situations but the actual searches were not primitive at all.
All this was just a minor detour to come to a very different translation than {M}.
This will be abbreviated as < M > and can be used by an also more involved "supervisor" machine U that we don't assume to be always halting at all. It will be called "universal" and it will simulate M by using < M > and any T input put together.
This of course requires a way to distinguish them, so as a first step we should solve this problem.
So in general, any S , T texts should be able to be combined into an S|T so that a machine can easily separate them. For this we can use the already mentioned stuttering idea.
There we said that the 01 and 10 can be simplest special markers and one of them say 01 could be the data start. Then if we design a machine to use programs, we can use not 01 as start of the reading rather 10 and thus use the text after this to simulate a machine. The end of the program or rather the start of the data is important and it could be marked again by 01.
Our previously claimed U universal machine too can use such S|T as input.
More to the point, U can not simulate the actual workings of an M no matter how smart U is.
Simply because the start is already different!
What we require as a good simulation is same input collection! That is haltings, using the < M > program of M. That is : For all S : S↓M ↔ S|< M >↓U.

We can create a reverse assignment of $|T|$ machines to all T texts by simply $|T|$ being the M machine that $T = <M>$ if such M exists and an empty machine if such doesn't exist.
Of course, there are real "empty" machines that collect nothing because halt never and so these will be variants of this abstract extension. But this is so more generally too, and all effective text collections by an M machine can be same by some other M' machine.
In truth, a surprising world of redundancies lies behind the effective collections.

Wandering away

The most obvious way of getting an M' variant of M is to add irrelevant steps to M.
But the really hidden feature of machines is that they bring about variants by themselves that we can not see from their tables.
At real machines, even at computers, we can get a good price for superficially damaged items that otherwise work "perfectly". So, there is a perfect functioning that is unique. At machines as representations of Effectivity we have an infinity of perfect variants. The only way we could find "bests" among them by regarding the shortest ones. As I mentioned in the very short section, this was an important point to define the complexity of texts. But now these variants are important for very different reasons. This relates actually more to the opposite feature, redundancy.
The twenty amino acids are coded by four valued triplets of nucleic acids. That means 64 possible codes, so nature made a huge redundancy. But even in our results at the start we had hidden redundancies that we simply ignored. Remember the decimals that had the extra forms.
But even earlier when we sequenced the fractions as:

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

We actually wasted places here because we repeated fractions like $\frac{2}{4} = \frac{1}{2}$.
To leave these out is of course easy but would be an extra ugliness.
But that I am not bullshitting, let's try to list all fractions not just the ones in $[0,1]$.
The trick is the same, that is going in finite groups but the fix denominators wouldn't work now.
Indeed, now there are infinite many fractions with any fix denominator.
The solution is to go in the totals of the numerator and the denominator:

$$\frac{0}{1}\ ,\ \frac{0}{2},\frac{1}{1}\ ,\ \frac{0}{3},\frac{1}{2},\frac{2}{1}\ ,\ \frac{0}{4},\frac{1}{3},\frac{2}{2},\frac{3}{1}\ ,\ \frac{0}{5}\ .\ .\ .\ .$$

Here we have a "redundancy galore" and strangely we listed much more than previously.
So it seems the more you squeeze into a smaller place the more you have to be generous and give multiples as presents. But back to our machine variants, the really important hidden fact behind their features is why we could recognize these features at all. Because we used the right concept of "variant". In fact, I still didn't reveal this and only used it above vaguely.
But from our machines as text alterers, the exact definition is almost trivial.
Two M and M' machines are variants if they collect the same texts, that is if they halt exactly from same inputs. So this trivial definition of variants that reveals the inner secrets of machines is only possible because we did the initially so stupid direction of following the same seemingly stupid definition of sets being mere collections. As I said then, the end justifies this start.
So now we can see already that this variant concept is a major part in that end. But to formally declare machines as text collectors could have not been enough. Turing's heuristic text collecting idea by alteration sequences halted by a purely inner condition of the machines, was the method that made this text collecting idea universal. And "universal" here meant externally universal, that is grasping all effective collections. This is what we can not prove only experience.
Kleene was the first who peeked into the surprising world of variants.
His Recursion and Fix Point Theorems hit upon two crucial features.
The recursion name refers to an amazing third meaning.
Namely, that every recursive rule, like the one we used to define $x\ [n]\ y = z$ can be melted into the totally mechanical systems easily, as we'll show later.

But just as Turing was attached to the universal machine, Kleene was to the fixed-points.
The third step which I regard as the “breakthrough” in Effectivity, is to see that variantness can be an easy condition if we do two things: First of all, we collect programs instead of mere texts.
And secondly, we collect them according how they run.
Then the non effective complements become an almost necessary situation.
Also observe that every text could be regarded as collector because if it’s not a program it simply doesn’t run and so it is an emptiness collector. So really all we need is to regard all texts, that is all data, as potentially runnable program. Not merely alterable “dead data”.
Then we will get really complex collections without effective complements.
So the previous two stages, inner universality and fix-points are merely the subjective steps to derive this final and robust experimental, almost physical truth.
It is derivable in any correct framework only because it is a truth as such. And not in reverse.
So Platonism or Objective Idealism had been beautifully exemplified in this progress.
The unpredictable role of subjectivity still stands! But behind every unpredictable truth of subjectivity stands a bigger objective truth. The stage of subjectivity when we still don’t see the bigger truth is unavoidable. It is Life! But the hollow relativism that denies objective truth is denying everything. Subjectivity too! These are the funny people. And I mean always funny. Who can never turn serious and make a judgment. These just smile at the world’s stupidity and don’t want to point fingers or name names. After all, who has the right to criticize others. We are all sinners, consumers and weak. So under the big blanket of this relativism can grow the new stupidity of the petty bourgeoisie. The social status quo of relativism is the new age. Everybody is free to think whatever he wants. The manipulators are busy cashing in on this gold mine of “freedom”. The fascist petty bourgeoisie was only a temporary stage. So Dimitrov was very deep recognizing something but it’s over. There is no need for a single megalomaniac to bring about the alliance of the petty bourgeoisie and the oligarchs. Now the whole world as such is the negotiator of stupid thoughts and therefore actions. What we think is what we are. What we do is just options from the tray of the evil negotiator. The spreading New Myth is that there are “Some” who organize this whole system. Conspiracy theories are merely wild branches of this new myth. That people create this framework that people live in. The “Some” are coming together and decide War and Peace. But even Tolstoy missed the essential counter truth. How individual actions, seeking a deeper freedom fall victim to themselves. Actions without consequences die off like ripples caused by pebbles. The broken glasses get thrown out and the window is repaired. Naked fans on the field can even make the headlines. But the mad men who want to change history must pass some tests. Must be excellent sharp shooters or must be able to take off a plane, even if not make it land. But then we might say what is a change in history anyway. What would be different if Kennedy lives or the trade center stands? And that is so because we are in the new age and indeed a certain history has been killed. The last two mad men who got to the top were Stalin and Hitler. And Hitler won! By losing. Stalin didn’t alter history and Hitler altered everything!
My dreams about an alternate history where Hitler won by winning and blamed everything on Stalin, were so rich in details that I have no doubt they have a corresponding objectivity behind.
The old Hitler gives a speech in New Delhi as UN secretary to implement the Resource Act, that changes everything about nations. The Auschwitz Group is a small circle of anarchists who obtained evidence about Hitler’s involvement and try to unmask the Old Man.
This is actually a preferred name Hitler likes to be called. The speech is about ten thousand words and I can remember every one. The taste of this future is still in my mouth and will be there forever. So much better than ours in many respect and I hate it just as much as this one. The one I think I live in. So I don’t even try to dream a future willfully. Probably I would hate that too.

Busy beavers

To get a better feel of how simple yet how complex Turing's machines are, we should see some alteration examples. But 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.
This gives a false impression that for a set of possible inputs the possible runs or halts are two infinite sets. But this is not true if we chose only from finite many inputs, in particular if we only regard inputs shorter than a given n length. We don't know how many steps the halts might take but since there are only finite many, there has to be a longest one.
This depends on the m alphabet size. the n length restriction and of course our machine.
If we knew this $s(m, n)$ number then quite amazingly, we could tell even the runnings in finite time! Indeed, at any of the restricted inputs, we just must wait until the known maximal step number and if till then no halt happened then we must have a run.
This situation then can be stretched even more to the really surprising one by regarding not a single machine but finite many, in particular to machines that have tables with time texts only up to a t length. Indeed, then again knowing the $s(m, n, t)$ step number of the longest possible halt, would enable us to decide the runs in finite time too.
In fact, any $b(m, n, t)$ bounding function that is always greater than s would do just as well. We would just have to wait a bit more but still tell the infinite runnings in finite time.
Tibor Rado realized first that the existence of non effective complements implies that such bounding function can not be effectively calculable. Which of course doesn't forbid that particular s values can be determined. He also observed that we could watch for some other features that our limited machines and limited inputs can produce and these may be uncalculable too.
Most amazingly, with even the simplest $m = 2$ alphabet and $n = 0$ text length, that is using an all 0 infinite memory line as start and quite small m values, the $s(2, 0, t)$ step number grows incalculably fast too.
$s(2, 0, 2) = 6$, $s(2, 0, 3) = 21$, $s(2, 0, 4) = 107$, $s(2, 0, 5) = 47{,}176{,}870$.
The cause behind the actual uncalculability is that there is no machine that could recognize all those tables that will not halt from an empty start. This fact relates to Turing's original discovery but doesn't follow from it yet. The existence of a universal machine does mean that a single machine can recognize all those tables that will halt. But why the complement of this, that is the set of those tables that don't halt is not recognizable, does not follow at all. It only follows from what I called in the previous section as the third level or new breakthrough in effectivity.
Turing did show that there are unrecognizable complements but did not see this heuristically simple new method behind the unrecognizable complements. Of course as expectable, his recognition of mere existence was necessary to prove the new method.
So what is this new heuristically simple cause of unrecognizability in our case?
It is that we recognize some tables by not what they look like inside, rather from what inputs they halt. If two machines halt from exactly the same inputs then they can be regarded as mere variants of each other. And then the trick is to regard a variant complete set of tables.
Of course, two trivial such are the empty set or the set of all tables.
Now amazingly, if a set of tables is variant complete but not these trivial ones and it is machine recognizable then the complement is not machine recognizable for sure.
In our case such set is the tables that halt from empty input. It is not empty and it is neither the set of all tables. It is machine recognizable by a universal machine. And so the complement, that is all those tables that doesn't halt from empty start is not recognizable.
The breakthrough law is actually even wider and selects a non recognizable side not as the other side of a recognizable. First of all, if a set is variant complete then the complement is such too.
Secondly, remember how all texts can be regarded as programs or tables because if a text is meaningless as such then it simply doesn't run so it is an emptiness recognizer. Now quite surprisingly these emptiness recognizers in general that is including the correct such programs too come in play here. Namely, if we have a separation of the programs or tables that the two sides are variant complete then one side must contain all emptiness recognizers.

This side must be non recognizable for sure! But the other side might be non recognizable either.
In our case it was.
After this little theoretical detour, remember that already among a given t time text length we can wait for ever to see which machines will halt.
But here there can be telltale signs of a hopeless wait, that is infinite run.
If we are curious about some features then knowing the maximal step number is the best help because then we can check all machines for that many steps and check the feature.
For example, Rado first asked how many 1-s can be produced by halting machines.
Non halting ones of course can produce infinite many 1-s but these are boring.
The maximal 21 step machine will produce 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. But if we check all $t = 3$ machines up to 21 step, we find this one too that is "better" in this respect.
By the way, the busy beaver name refers to how these machines run back and forth.
Our earlier example for a table was actually of the $t = 3$ busy beaver champion for step number.

The "halting problems"

For our effective assignments, 3. of the Basic Theorems implies at once that we must have some effective sets without effective complement.
Indeed, here a. is true as we saw, though only abstractly. Also b. should be true though we didn't check it exactly. So c. must be false.
But let's try to go back to the source, that is to find a M machine that has no complement machine, that is one that halts exactly from those starts from which M runs forever.
The idea is the same. We can say that two machines M , M' share a T text if either they both halt from T or both run forever, which means that $T{\downarrow}M \leftrightarrow T{\downarrow}M'$.
The only machine that shares no text with an M can be its complementing one which halts exactly from where M runs forever. So an M has no complementing machine if and only if it shares text with every M'. To find such M we must of course involve the universal U machine.

A text transformer is a Θ machine that halts from every T start and gives an R resulting text.
For these we use Greek capitals and don't use the $\downarrow$ halting sign, instead write $T\Theta = R$.
If we use this R as input for an M machine, the notation can simply continue as $T\Theta M$.
So both the equal sign and the R variable was omitted and this hides a fundamental general assumption about machines. Namely, that ΘM can be regarded as a new machine that does first the Θ transformation and then M. This hidden assumption can be made explicit by:
$T\Theta{\downarrow}M \quad \leftrightarrow \quad T{\downarrow}\,\Theta M.$
To duplicate a T text into $T|T$ is obviously a text transformer and let Δ denote this.
So $T\Delta = T|T$. We claim that ΔU shares with every M machine the $<M>$ text.
Indeed, $<M>{\downarrow}M \quad \leftrightarrow \quad <M>|<M>{\downarrow}U \quad \leftrightarrow \quad <M>\Delta{\downarrow}U \leftrightarrow \quad <M>{\downarrow}\,\Delta U.$

And so what we proved is that ΔU is a machine that has no complementing machine!
This doubling reminds us the Cantor Diagonality.
The anti-diagonal alteration we didn't need because the sharing property avoids indirectness.
But all this should remind us even more importantly Gödel's $D(x , y)$ formula that captured the derivabilities of the $P(x)$ properties with $y = <P>$. So the exact claim was that:

$$\vdash P(n) \quad \leftrightarrow \quad \vdash D(n , <P>)$$

The diagonalization was then instantly mixed with negation by using $P(x)$ as $\neg D(x , x)$ and n as $<\neg D(x , x)>$, giving:
$\vdash \neg D(<\neg D(x , x)> , <\neg D(x , x)>) \quad \leftrightarrow \quad \vdash D(<\neg D(x , x)> , <\neg D(x , x)>).$
If our system is consistent then both sides must be false and so the two statements are an undecidable pair.
I tamed this "horror" argument already in the Gödel section with using the concept of "share".

But the point remained that the diagonality was mixed with negation.
In truth, the diagonalization itself is a “heavy gun”! If for example we use an abstract text alteration system as a framework for axiom systems then this step is practically un-graspable!
But here in Turing’s system it is served on a silver platter as the Δ duplicator!
The real motor of this of course is that halting can represent the derivabilities.
And the concrete help is the fact that there is universal machine.
So Gödel’s $D(x , y)$ formula can become a U universal machine that accepts $n|m$ inputs.
$|\!-D(n , m) \leftrightarrow n|m\downarrow U$. The universality for numbers as inputs and derivability as haltings then means that also $n|\!< P >\downarrow U \leftrightarrow n\downarrow P$. So actually we have that: $|\!-P(n) \leftrightarrow n|\!< P >\downarrow U$.
And thus $|\!-P(< P >) \leftrightarrow < P > | < P >\downarrow U \leftrightarrow < P > \downarrow \Delta U$.
But ΔU has no complementing machine and so the effective set of the $P(< P >)$ theorems has no effective complement! So, the set of non theorem statements is not effective. Then we can argue further to get the incompleteness of the axiom system with using its consistency too.
Indeed, then if every $P(< P >)$ statement or its negative were a theorem then we could get the non theorems among them by deriving the theorems and simply negate those.
Using this grand argument we should always mention that the effectiveness of all theorems is a bit paradoxical. Especially regarding this as generability! So a machine can spit out all theorems!
But this is not a practical contradiction with mathematics as a subject. Indeed, spitting out all theorems means spitting out all crazy theorems too and needs astronomical times.
Back to our subject:
For a given M to find shared texts with every $M’$ may not be as easy as it was above.
But to prove that M has no complement, we can think indirectly too and arrive at contradiction with already known non complement cases. For example, we can show that U has no complement as follows: Suppose that U had a $\neg U$ complement. Then we had
$T\uparrow \Delta U \leftrightarrow T\Delta\uparrow U \leftrightarrow T|T\uparrow U \leftrightarrow T|T\downarrow \neg U \leftrightarrow T\Delta\uparrow \neg U \leftrightarrow T\uparrow \Delta \neg U$.
And so ΔU would have the complementing machine $\Delta \neg U$ which we proved not to exist.
The direct argument requires the ad hoc trick to guess the input that U shares with an M.
It is actually $<\Delta M >|<\Delta M >$. Indeed:
$<\Delta M >|<\Delta M >\downarrow M \leftrightarrow <\Delta M >\Delta\downarrow M \leftrightarrow <\Delta M >\downarrow \Delta M \leftrightarrow <\Delta M >|<\Delta M >\downarrow U$.

These two simplest no complement cases ΔU and U became called the halting problems.
Indeed, U, the universal machine can be called the halting machine because it halts from those $S|< M >$ texts where $S\downarrow M$. So ΔU is then the diagonal halting machine.
As a later result shows, U actually shares infinite many texts with every M.
In particular, there is a T_0 text so that any $T_0|T$ continuation of T_0 is a shared text.
This was obtained by Kleene yet he didn’t see a consequence that I called the “breakthrough”

Denials of the “breakthrough”

If instead of doubling, we fix a T_0 start and regard all those M machines that halt from T_0, then these $< M >$ programs are an effective set. Indeed, U will recognize them because:
$T_0\downarrow M \leftrightarrow T_0|< M >\downarrow U$. But will again be true that there is no complementing machine?
Yes, and this is just a special case of a whole arsenal of non existing complements that were produced by the breakthrough Rice’s Theorem discovered only in 1953. It was not so hard to prove but its consequences changed our whole vision about effectivity. It surely changed mine!
The present insane Formalist treatment of mathematics tries to ignore visions and only deals with theorems. This one is also regarded as only a new consequence. It’s breakthrough role is a taboo.
Of course, no text book can avoid the theorem itself.

In my view, it corresponds to the Well Ordering Theorem of Set Theory. Nowadays it also has a proof only few lines long but it takes pages to see its true role. Namely, the concept of "growth" lies behind that then at once gives the Axiom Of Choice its crucial role.
To prove my point for Rice's Theorem is very easy! Let's go back to the time before 1953.
Many regard it as the golden era of effectivity but I just call it the lingering darkness.
In 1931 Gödel started everything, by showing that in most axiom systems there have to be undecidable statements. From his proof it feels as if the language of these axiom systems were the reason for this. They can talk about their own derivabilities.
In 1936 Turing shows that effectivity can be defined naturally and sees clearly that derivability in an axiom system is merely a special effective collection of the theorems. The complement set of the theorems, the "non-theorems" is not effective, that is not collectable by any method.
This was a first light in the tunnel. Indeed, with this, the existence of undecidable statements is not a language problem but has an almost physical cause that I already explained but repeat again:
If all S statements were decidable, that is S or $\neg S$ were theorem, then to collect the non-theorems effectively were possible as follows. We collect the theorems by deriving them and then formally negate each. The impossibility of any effective collection method for this set forces that the only loose chain in the assumption, the full decidability must be false.
Of course, we have same heavy guns as non loose chains here too:
Most importantly, how can we assume to derive all theorems? We have a time machine?
But this is merely a systematic application of all our axioms and all our derivability rules.
As start, the axioms if they are infinite many must be generable by some rules too.
This is true for all axiom systems though nobody noticed its importance before.
The rules of logic is actually the rule of derivations and these were quite exact by this time too.
So we can list all the theorems that all future mathematicians haven't even discovered yet.
As I explained, this is not paradoxical at all. This mechanical listing of all theorems is useless for a "real mathematician". I mean an old-fashioned one who regards mathematics as chess.
But for some, like me, this new math is the only math that really matters.
To be most correct, we should call this new math as "metamathematics" and so Kleene for his first comprehensive book found the perfect title.
Once Paul Erdös said "it's time to do some real mathematics" after dwelling on some set theoretical problem. He was probably the last big classical mathematician. Stubborn too in his lack of vision for two things. That this new math is the only future math and that it is actually relating to the understandability of all mathematics by everyone.
So, Didactical Logic is the real future. Unfortunately, it seems I'm the only one believing in this.
In 1952 Kleene publishes his Bible the "Introduction To Metamathematics". Was Rice's 1953 result, a seemingly obvious consequence of Kleene's results somehow forgotten to be included in the Bible by Kleene? Of course not. He didn't see it! But most importantly:
By portraying as a mere consequence, that of course now must be included in all text books, these text books avoid to mention the crucial vision changing role that was in fact the reason why Kleene didn't see it yet. The epigones march in formal derivations without any visions.

A direct way to the "breakthrough"

Rice's Theorem is not only a treasure chest of practical examples but will explain why we were blind to them before. But as start, we regard the already raised problem.
Fix a T_0 start and collect all programs that correspond to machines that halt from T_0.
We claim that the complement text set that contains all programs of machines running forever from T_0 is not effective. So there is no N machine that would halt from exactly these texts.
The basic idea of refuting such N machine is that it would imply that every machine has a complement. So observe a strange symmetry!
We claim that if an N machine could collect $\{ < M > ; T_0 \uparrow M \}$ for a fix T_0 then for any fix M_0 the $\{ T ; T \uparrow M_0 \}$ set would be machine collectable too.

The proof however doesn't use this symmetry, rather goes by observing two trivial facts.
The first is that this impossible N machine would have to halt from all those $< M >$ starts that are programs of M machines that run forever from any T start.
Indeed, if a machine runs forever from any start it will do the same for the particular T_0 too.
The particular T_0 will not be thrown out though, rather used in a second triviality. Namely, that some machines do halt from T_0 and actually all these must have programs from which N must not halt. In particular, if an M_0 halts from T_0 and M_0' is such that it halts exactly from the same inputs as M_0, that is $[M_0'] = [M_0]$ then N must not halt from $< M_0' >$ either.
Such M_0' is called as a variant of M_0, so variants are simply same collecting machines.
Now we can regard the arbitrary M machine and create one that halts from its complement.
First we regard an $M_0\&M$ machine that halts from a combined $S|T$ input only if M_0 halts from S and M halts from T. This is very easy to achieve by first running one of them and only start the other if the first halts. Amazingly, the order is irrelevant.
More practically for our goal, we will have that:
For those T that M halts, $S|T\ M_0\&M$ is merely a variant machine of M_0 with S inputs.
For those T that M does not halt, $S|T\ M_0\&M$ will definitely not halt either with S inputs.
So if we had a T depending $< M_0\&M > (T)$ program set with T inputs then:
$\{ T ; < M_0\&M > (T)\ N \downarrow \}$ would be the effective collection of $\neg [M]$.
The proof that such program set indeed exists will be shown in the next section.
What is important now is that we used much wider assumptions than the original plan was, that is collecting programs running forever from a fix T_0 start. To tell these is easy now.
A machine is never halting if it halts from no input.
A program is never halting if it is program of a never halting machine.
We need a machine that halts from all never halting programs.
But this is not enough for our goal which is not surprising since a machine that halts from every input would be automatically such too.
An M_0 variant refuser is a machine that doesn't halt from any Q that $|Q|$ is a variant of M_0.
Now we can claim that:
If there were an N machine that halts from all never halting programs and which is also an M_0 variant refuser, then every M had a complement machine.
To get the praised theorem from this, we make the condition a bit weaker.
Namely, we make the M_0 variant refusing a universal claim not an existential.
This also makes the variantness shifted from machines to programs, that is texts.
So we require that with every P that is refused, that is not halted from, the same follows for all Q variants of P. Meaning that: $[\ |Q|\ ] = [\ |P|\ ]$
Observe that this could be also said by as the refused inputs being a variant complete set.
But observe also that if a set is variant complete then its complement is such too.
So actually our aimed **T** text set is then merely a half of a variant complete separation of all texts.
But there is a little glitch because by going from existence to universality we lost something!
Namely, that there is at least one P text in both sides.
So we must add as condition that the variant complete separation is not the trivial, all and nothing!

Rice's Theorem: If **A** , **B** are a variant complete separation of all texts and neither are empty then both can not be effective. Only one or neither.

Indeed, since they are both variant complete, one must contain all never halting programs.
So then the machine that would collect that side would be halting from all never halting programs and were also an M_0 variant refuser because the other side is not empty either.
And then of course all machines had complement which is not true.
The third step is to use Rice's Theorem to make it the "breakthrough" I claimed.

It goes by regarding cases when one half of the separation say $\mathbf{A}$ is known to be effective.
Then we get that the complement can not be. But the real breakthrough comes about by realizing how easy it is to make sure that an $\mathbf{A}$ text set is effective and variant complete.
All we have to do is run the texts in $\mathbf{A}$ as programs and "watch" for some "feature" that is not depending on the programs only on their runnings.
These two, "watch" and "feature" sound vague but in the practice they are trivial.
The simplest case is our initial special case of using a fix T_0 start and "watch" for halt.
To be even more specific we can fix an input number say 100 and try any program from this and if it halts we regard the program as being recognized that is collected. The complement set, that is all those programs that run forever from 100 are not machine collectable for sure.
This more practical version as usual, brings out a deeper philosophical problem.
Does this no possibility of exact collection by halts mean that any analyzing of the programs is unable in general to detect the infinite running from 100?
Obviously for some particular programs we can establish by some "analysis" that they will run forever from 100. So the crucial point is that such analysis always must be particular. But this just opens newer questions that we can not answer because the concept of analyzing is really vague.
The previously mentioned vagueness of the "watch" and "feature" actually can be a bit improved by using generation instead of recognition. Then "time" tells everything.
Let's see a few examples for easily "watchable" "features" in number generations or lists:
The above used start from 100 is now whether it is generated.
And indeed, it is visible in finite time. Similarly, to list at least one prime or at least twenty primes is again effective. We again see them being fulfilled by watching the listed numbers.
But whether the number 100 will not be listed or no prime will be listed or no more than twenty is listed are not recognizable because these can not be assured from a beginning.
And indeed, these are the opposite sets of previous features and so their program sets are definitely not recognizable by Rice's Theorem.
Whether all primes are listed is again an intuitively not recognizable property because in a beginning we can see only finite many. But now the opposite, that is not listing all primes is also non recognizable intuitively because they all can still pop up after any beginning.
An even simpler case of this is whether a program will list exactly twenty primes.
Seeing more than twenty primes is a refutation but not the negative.
If the program will list less, we can not be sure of this forever. And if it will list exactly twenty, though we'll see this twenty at a point, we can not be sure if one more comes or not.
So Rice's Theorem can not be used for these intuitively dually non generable program sets.
An interesting case is whether a list will have infinite or only finite many members.
This means the same as having arbitrary long texts in our list or not.
For numbers it means having arbitrary large numbers or not.
The lists that have arbitrary long texts or arbitrary large numbers can be called unbounded.
Now we'll show that the programs for only these unbounded lists can not be listed.
A smart way to do this, is to prove that for any $\mathbf{L}_1$, $\mathbf{L}_2$, . . effective list of such unbounded effective lists there has to be an effective $\mathbf{L}$ missing from our list.
We must allow repetitions because different programs can produce same lists.
Best to visualize them of course under each other as lines of texts.
The k-th text in $\mathbf{L}_n$ will be denoted as $T_{n,k}$. So the first text in $\mathbf{L}_1$ is $T_{1,1}$.
Increase this $T_{1,1}$ with a symbol to get a $T_{1,1}{}^{+}$.
Now go in $\mathbf{L}_2$ 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 text in $\mathbf{L}_3$ longer than $T_{2,k_1}{}^{+}$ and it is T_{3,k_2}.
And so on, we get an obviously increasing $\mathbf{L}$ list that can not be identical with any $\mathbf{L}_n$.
The first text of $\mathbf{L}$ is already longer than $T_{1,1}$ and so all texts in $\mathbf{L}$ will be longer.
So $T_{1,1}$ can not be in $\mathbf{L}$. Then the found longer text of $\mathbf{L}_2$ can not be in $\mathbf{L}$ again because we skipped that length completely. And so on, all $\mathbf{L}_n$ lists will have a member not in $\mathbf{L}$.

There are more obvious cases too where Rice's Theorem is not usable, simply being a narrower consequence of the non effectivity of an emptiness halter which is a variant refuser.
A most important example is the following:
Regard the set of all transformers. They are the machines that halt from any input.
The set of their programs is variant complete and the complement is the set of all texts that are not programs or programs of machines that don't halt from some input. This contains all empty programs so a machine that would collect these would be an emptiness halter that is variant refuser too. So this can not exist and so this last text set is not effective.
But this wouldn't follow from Rice's Theorem because the other side, the set of all transformer programs is not effective either! This follows by applying a truly diagonal argument.
We avoided diagonality by using the sharing of inputs. But here the collected inputs are the set of all texts for every transformer and so we must distinguish the results too.
We can alter every Θ transformer's result at $<\Theta>$ that is $<\Theta>\Theta$ to anything else and thus getting an assignment of results to $<\Theta>$ for any Θ that is 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.

The most heuristic consequence of Rice's Theorem

Our earlier "loose" thoughts about how an "analyzing" of a program can not be successful, can be turned into an almost exact theorem and it could be called as:
"No Buildup And Collection Correlation" or "Machine Counter Example" Theorem.
The fact that both sides of a non trivial variant complete separation can not be effective, combined with my obsession about replacing diagonality with input sharing, plus as third ingredient making an asymmetry by regarding one side as the collection feature, can lead to this theorem.
The "correlation" simply means the if and only then relation.
The "counter example" means the shared input which is actually a program or a machine.
Indeed, if a collection feature is recognizable and is only about the collected set, then no machine can perfectly recognize this feature or rather behavior. Simply because the opposite behavior, that is collecting a set with our feature is an effective and variant complete program collection.
In fact, regarding any machine that collects programs, the shared input is actually a counter example that refutes a perfect correlation. That is, we can not claim that every and only such and such machines will collect sets with our feature. There will be some machine that either fulfills the buildup feature yet it will not collect a set with the feature, or the machine will not fulfill the buildup feature yet it will collect a set with the feature.
Of course, we had this undefined set feature taken here naively. If we are more specific, for example regard as set feature to have a fix text as element then everything becomes exact.
So actually, we have here an infinity of exact theorems that we can not combine!

Formal derivations of Rice's Theorem

Kleene 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.
But to get its meaning, first let's regard an even more plausible fact.
We have a M and we want to start it not from a normal T input text rather an $S|T$.
But we could also build the T tail section into M and use only S as input.
If $M'(T , M)$ is the modified machine then : $S|T\downarrow M \leftrightarrow S\downarrow M'(T , M)$.
The existence of universal machine makes it plausible that this M' machine should be obtainable universally too. Namely, as $M' = F (T , M)$ by a fix F effective function.

So here then T is a parameter used by F and this explains the name of our next theorem.
Of course, to start a new effectivity formation, ordering machines to pairs of text and machine, would be crazy so instead we use what we have already, that is $< >$ programs, $| |$ machine formations, and a Π text transformer doing the job of F.
Thus $M' = |(T|<M>)\Pi|$ and this Π doing the job of F means that:

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

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 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:
$S|M^U\downarrow M = S|(<M^+>|<M^+>)\Pi\downarrow M \leftrightarrow S|<M^+>\downarrow M^+ \leftrightarrow S\downarrow|(<M^+>|<M^+>)\Pi| = S\downarrow|M^U|$.
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.
The really amazing fact is that simple consequences of this theorem with the proper vision can create an abundance of machines without complements.

Fix Point Theorem:
For every Θ text transformer there is a Q text that $[Q\Theta] = [Q]$.
That is, for all S $S\downarrow|Q\Theta| \leftrightarrow S\downarrow|Q|$.

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 a Q is M^U.
The first $\leftrightarrow$ is true by our definition with $T = M^U$.
The second $\leftrightarrow$ is true by the Recursion Theorem.

Let's recall: Two A, B texts are variants if $[A] = [B]$. A text set is variant complete if for every member the variants are members too. If $\mathbf{A}, \mathbf{B}$ is a separation of all texts and one is variant complete then the other is too, so a variant complete set is actually a variant complete separation.

Rice's Theorem:
Let a variant complete separation of all texts be $\mathbf{A}, \mathbf{B}$ 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 $\mathbf{A}, \mathbf{B}$ were effective then this Θ were a text transformer and then by the Fix Point Theorem for some Q text we would have that $[Q\Theta] = [Q]$.
Thus $Q\Theta$ and Q are variants and so they should be in the same of either $\mathbf{A}$ or $\mathbf{B}$.
But by the definition of Θ we have that $Q\in\mathbf{A} \rightarrow Q\Theta = B\in\mathbf{B}$ and $Q\in\mathbf{B} \rightarrow Q\Theta = A\in\mathbf{A}$.

Our earlier naïve proof was more informative by telling a side that must be non effective for sure.

Namely, the one that contains the never halting programs. We can get that version again more precisely, and strangely without the Fix Point Theorem, using only the Parameter Theorem.
The idea of the proof is again to show that the existence of an N machine that would halt from all never halting programs and which were also a M_0 variant refuser, would imply complement for all machines. That is:
If there were an N machine that for all Q texts:
If $|Q|$ is a variant of M_0 then N does not halt from Q. And:
If $|Q|$ halts from no input then N halts from Q. That is:
For all S , $S{\downarrow}M_0 \leftrightarrow S{\downarrow}|Q| \rightarrow Q{\uparrow}N$ and
For all S , $S{\uparrow}|Q| \rightarrow Q{\downarrow}N$
Then for any M machine there were a complementing one.

The $M_0\&M$ "and" machine is used again for $S|T$ inputs.
$S|T{\downarrow}M_0\&M \leftrightarrow (S{\downarrow}M_0$ and $T{\downarrow}M)$
More practically, we use S as input for M_0. If this halts we use T as input to M and wait.
Even more practically for our proof:
$T{\downarrow}M \rightarrow$ For all S [$S{\downarrow}M_0 \leftrightarrow S|T{\downarrow}M_0\&M$] and
$T{\uparrow}M \rightarrow$ For all S [$S|T{\uparrow}M_0\&M$]
Using for both of these, first the Parameter Theorem then the two N conditions and finally the Parameter Theorem again:
$T{\downarrow}M \rightarrow$ For all S [$S{\downarrow}M_0 \leftrightarrow S|T{\downarrow}M_0\&M \leftrightarrow S{\downarrow}|(T|{<}M_0\&M{>})\Pi|$] $\rightarrow$
$(T|{<}M_0\&M{>})\Pi{\uparrow}N \leftrightarrow (T|{<}M_0\&M{>}){\uparrow}\Pi N \leftrightarrow T{\uparrow}|({<}M_0\&M{>}|{<}\Pi N{>})\Pi|$
$T{\uparrow}M \rightarrow$ For all S [$S|T{\uparrow}M_0\&M \leftrightarrow S{\uparrow}|(T|{<}M_0\&M{>})\Pi|$] $\rightarrow$
$(T|{<}M_0\&M{>})\Pi{\downarrow}N \leftrightarrow (T|{<}M_0\&M{>}){\downarrow}\Pi N \leftrightarrow T{\downarrow}|({<}M_0\&M{>}|{<}\Pi N{>})\Pi|$

And thus the complement machine of M would be: $|({<}M_0\&M{>}|{<}\Pi N{>})\Pi|$.

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 $e = {<}f{>}$ 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 $< f >$ 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 $S^n(,)$ effective total function that gives the codes for all the first variable parametrizations of f, using $< f >$ and x_1 as inputs. That is, as $S^n(< f >, x_1)$. Thus $| S^n(< f >, x_1) |$ are those buildups and so: $f(x_1, x_2, . , x_n) \cong | S^n(< f >, x_1) |(x_2, . , x_n)$.

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 $S^n = S$, $f(S(x_1, x_1), x_2 , . , x_n) = g(x_1, x_2, . , x_n)$, $< g > = c$, $S(c , c) = p$. Then:
$f(p, x_2, . , x_n) = f(S(c,c), x_2, . , x_n) = g(c, x_2, . , x_n) \cong |S(< g >,c)|(x_2, . , x_n) =$
$|S(c , c)|(x_2, . , x_n) = | p |(x_2, . , x_n)$.
$\cong$ 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:
$| t(p) |(x_2 , . . , x_n) \cong f(v, x_2, . , x_n) \cong | p |(x_2 , . . , x_n)$.

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

Recall the derivable case definition of the universal operation in a bit simpler form:

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

In other words:

$$x\ [n]\ y = x + 1 \qquad \text{if} \quad n = 1\ ,\ \ y = 1$$

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

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

Or with using a bit more general $f(x , n , y)$ form for $x\ [n]\ y$:

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

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

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

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

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

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

$$g(e , x , n , y) \cong |\ e\ |(|\ e\ |(x , n , y - 1) , n - 1 , x) \qquad \text{if}\ \ n > 1\ ,\ \ y > 1$$

By the Recursion Theorem we have a p value of e that $g(p , x , n , y) \cong |\ p\ |(x , n , y)$.
Then of course this $|\ p\ |(x , n , y)$ will satisfy exactly the equations for $f(x , n , y)$.
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 universal 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: $f(n) \cong f(n) + 1$. It leads to: $g(e , n) \cong |\ e\ |(n) + 1$.
And not surprisingly, the guaranteed solution $g(p , n) = |\ p\ |(n) = |\ p\ |(n) + 1$ 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!

Not separable sets

There is an other deepening of non effective complements beside Rice's Theorem.
In some sense this is even more surprising and will alter our perception about how an effective collection can be so complex that the complement is even more complex and can not be effective.
A "selector pair" is any M , $\neg M$ complementing machines. That is, with $[M]$ and $[\neg M]$ being disjoint and containing all texts. These $[M]$ and $[\neg M]$ can also be called selectable.
Two $\mathbf{T}_1$, $\mathbf{T}_2$ disjoint text sets are separable if an M , $\neg M$ selector pair exists that:
$\mathbf{T}_1 \subseteq [M]$, $\mathbf{T}_2 \subseteq [\neg M]$. And now the big claim:
Some pairs of disjoint sets are not separable! So no selector pair can contain them one by one.
Thus, it's not just a complement that can be more complicated and thus non effective.

Two disjoint sets can make all their disjoint super sets have at least one that is non effective.
So two disjoint sets can have a hidden antagonism.
They can be both effective and thus then this antagonism must be caused by their programs.
Strangely, it seems as if axiom systems are the only practical source of this situation.

Robinson Arithmetic

This was how the previous concept, the not separable sets surfaced.
And this was the other most important conceptual result next to Rice's since Gödel.
First I will list Peano Arithmetic and will stick to the original usage of consecutiveness as an n' function not relation as I used it before. I'll also obey the convention of 0 as the single constant.

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

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 mean universality.

Four important statements derivable with this last induction scheme are the following:

8. $x' \neq x$
7. $x \neq 0 \rightarrow \exists y\, (y' = x)$
9. $x + y = y + x$
10. $x \cdot y = y \cdot x$

The strange numbering will be clear soon.

8.'s 0 condition follows from 1.
Its step condition $x' \neq x \rightarrow x'' \neq x'$ follows from $x'' = x' \rightarrow x' = x$ a case of 2.

7.'s 0 condition is true because $0 \neq 0$ is false.
Its step condition $(\, x \neq 0 \rightarrow \exists y\, (y' = x)\,) \rightarrow (\, x' \neq 0 \rightarrow \exists y\, (y' = x')\,)$ has in its second implication a truth as condition by 1. and also a truth as consequence by using $y = x$.
So this whole consequence is true regardless of the condition. So the step condition is true too.

9. is going to be a nightmare. (So 10. we'll not even show.)
First we prove that 9. is true with $y = 0$, that is $x + 0 = 0 + x$.
By 3. we only have to prove $x = 0 + x$ which we do with induction.
The 0 condition says $0 = 0 + 0$ which is 3. again with $x = 0$.
The step condition says $x = 0 + x \rightarrow x' = 0 + x'$. Which follows by: $x' = (0 + x)' = 0 + x'$.
The first equality is true by the assumption, the second by 4.
Now we prove that a 4*. $x' + y = (x + y)'$ first member incrementation is valid for addition.
We use induction on y. The 0 condition is trivial by 3. again and the step condition is:

$x' + y = (x + y)' \rightarrow x' + y' = (\, x + y'\,)'$. Which follows by:
$x' + y' = (\, x' + y\,)' = (\, x + y\,)'' = (\, x + y'\,)'$.

The first equality is true by 4. the second by the assumption and the third by 4. again.
Now we finally prove $x + y = y + x$ with induction on y again.
The 0 condition is what we proved first, so we only need to show the step condition:

$x + y = y + x \quad \rightarrow \quad x + y' = y' + x$. Which follows by: $x + y' = (x + y)' = (y + x)' = y' + x$.
The first equality is 4. the second is true by the assumption and the third by 4*.

Claim: 8. , 7. , 9. , 10. don't imply each other and can not be obtained without $7(\infty)$.
Yet now we omit $7(\infty)$. and instead add to the first six axiom 7.
This new 1. – 7. axiom system is the Robinson Arithmetic.
By our claim above it can not even prove 8. or 9. or 10.
To show this we'll need a very long but worthwhile detour. Indeed, it will go to the heart of Logic.

The birth of new math

Logic was subconscious for two thousand years in math and when it started to become conscious it initiated the concept of sets too. First only as models and then by Cantor's ingenious abstraction, as unstructured pure sets too. Cantor never realized that his abstraction has this role too.
Indeed, only after the dust settled about Cantor's new infinites and Zermelo discovered a missing axiom about claiming sets, did it turn out that this axiom is also the key to perfectly connect sets to Logic.
But one half of this connection does not need Zermelo's new axiom and so this half surfacing can be regarded as the critical birth of new math or Meta Mathematics as Kleene called it much later.
Very few mathematicians are aware of this amazing moment of truth by Beltrami.
This was a truly historical "event"! Foggy and yet clearly altering the ideas from then on.
A parallel or rather opposite "essence pinpointing" of Logic is much easier though again is hardly recognized by mathematicians.
So I start with this essence, that goes back to more than two thousand years to Aristotle's Formal Logic. The essence of modern Mathematical Logic is what was wrong with Aristotle's.
And this is very simple! He did not recognize variables.
What sounds more surprising is that even nineteenth century classical math was still sloppy with number variables. So actually, variables as such were not clear for a very long rime.
Today they became crystal clear plausibilities for any elementary school kid.
The word problems are the crucial tool to bring out this a priori intuition and so they are the golden road into mathematics. There was a time when this was fully recognized and then the Soviet Larichev created the golden book for this golden road.
Now that recognition is ignored and elementary school math education became a joke.
There is a reason for this too but I will not explore that now.
The simple fact is that any child can accept that x , y , . . can abbreviate something and using these variables we can juggle with the yet unknown quantities.
Just as plausible is that these can denote objects or persons, even concepts.
The point is that wherever an x appears, it is the same thing.
Poor Aristotle realized that the two main Logical claims are existence and universality.
The "every" and the "some". The "nothing" is simply not any or rather no some.
So the negative of "every red car is beautiful" says "there is some red car that is not beautiful".
This strange carrying of the negation into the details and meanwhile altering the every to some was fully recognized by Aristotle and in spite of this, his Logic remained a hollow triviality.
Only New Math poured soul into his recognition by using variables for individual objects and making the claims of "every" and "some" as quantors that turn those variables into something that had no name before and not even today. Quantized and thus bound variables.
But it perfectly joined an other more obvious bounding which is concretizing, that is placing a name into the variable. This new quantized and concretized way of writing the everyday sentences should be again an elementary school subject even before math.
The third miracle that is needed is exactly our subject Effectivity. Namely, as writing flowcharts.
These three: Word problems, Quantors and Flowcharts would make every elementary school kid a mathematical genius before even going to high school. So, obviously we don't use these.

Now back to the historical spark which unlike the variable usage, is much harder to see, we must realize that amazingly this also goes back to the Greeks. To the field that ruled before our new algebraic age, Geometry. This doesn't need didactical tricks. But it does have its own "taboos".
And the blame starts with Euclid. The crucial concept "parallelity" has three identical visions.
Fix distance, same angle to a third line, and finally the simplest: non crossing.
As simple these seem visually, just as hard is their relation logically. Some things are obvious.
For example that if the distance is fixed between two lines then they will not cross.
But how the three conditions relate in detail is a mess!
The nice would be if we could prove that they each imply the other two.
Unfortunately, this equivalence of the three definitions doesn't want to come out without some other assumption. This extra assumption about parallelity is what we can call an Axiom Of Parallelity. Euclid used a fairly complicated one and later the simple claim became that there can only be one non crossing line with a fix line through any fix point outside the fix line.
We feel that tilting a non crossing line through this outside point, it must tilt down on one side and then on this side it gets closer and closer and eventually will cross the base line.
But what is a line anyway? What if it just seems straight but curved in essence so that indeed it can get closer and closer to the base line yet never cross it?
This absurdity came to Bolyai's mind only after two thousand years everybody gave up on the scenario that somehow this necessity of crossing is provable.
So Bolyai said let's assume it doesn't cross just gets closer and closer.
He wrote to his journal: From nothing I created a new world.
So instead of a contradiction, this assumption of these multitude of non crossing lines created a new non Euclidian Geometry. But the crucial question was which one is the real.
He even thought about using light rays to measure the true situation.
So he missed the bigger picture, that such measurement is irrelevant.
We still must reflect upon such physical connection because it has an inner mathematical side.
As we mentioned as start, the second parallelity definition uses same angles to a crossing third.
The crucial point that makes this definition almost physical, is that angles are local!
An imagined gradually curving line that gets closer and closer to an other is quite oppositely, very non local. If light rays are the physical lines then looking at light rays from above is obviously not quite well defined since we should regard an infinity of secondary light rays that bring to us the original lights' passing. That is a mess!
A carpenter looking at the table's edge from its corner as we all do when want to see if something is straight, is a source of delusion because if the light rays are themselves curved then this method will obviously fool us perfectly. So we might think then that math is unable to grasp such intimate curving of the lines. But we are wrong exactly due to the role of angles. Like everywhere in science and math, the highest truths relate to the simplest didactical choices. And education is always making the wrong ones by patronizingly hiding the most beautiful and challenging points. The most evident case for this is the elementary school proofs for the triangles' angle sum being 180. They draw a parallel to the base line through the top corner and show the there again appearing two base angles. Thus these and the top already there, indeed form a line. The biggest problem is that the sneakiness of regarding <u>the</u> parallel doesn't even reveal what was the hidden assumption. The correct proof should be measuring the base angles onto the sides and thus producing potentially two lines. Then we can even go into the jungle and show that these lines should not cross the base line. Or even if we don't do that, we can mention this road and so still present the amazing possibility that two non crossing was created.
Abstraction must involve the concrete at its introduction already in a Didactical Logic!
Not just potentially as derivabilities in our present proof obsessed false Formalist Logic.
Back to the end result, we got a very important tool for verifying the curved lines locally.
Of course, we need three local points. Placing our theodolites there and measuring the three angles, the total will only be 180 if the lines are straight. More than 180 means that the lines curve outward and less that 180 means curving inward. So this was that Bolyai planned to measure for very big triangles.

His father was a friend of Gauss and so he sent his results for evaluation but received a very rude reply: I can not praise your results because I myself arrived at those few years ago. Gauss did not lie, he was just an insensitive asshole. A “Formalist” even before the new Logic was formalized.
He received his punishment still in life from two sides. This mentioned inner strive for exactness was one side. He was called the “fox”! His proofs hid the tracks how he arrived at them.
To understand the situation, we must again emphasize the potential sloppiness in notations at that time. Which reflects the potential logical errors. The obsession with exactness subjectively, without asking if an objective yardstick is possible, was a catch twenty two. He made a dozen proofs for the Fundamental Theorem Of Algebra and as they tried to be more and more exact they hid more and more about the “truth”. I remember the simple “explanation” in high school in Courant’s What Is Mathematics and at that time I thought that it was a proof. But Courant was very out dated in his vision. Over simplification can not be a road against Formalism. Especially with simple derivations like the Fundamental Theorem Of Arithmetic presented with trivial errors.
The other side I mentioned was the complex numbers. Actually, the same as the Fundamental Theorem Of Algebra but not regarded as base of proofs, rather as physical reality. Gauss deeply believed in such physical connection but died without knowing that he was right in this too!
But back to our subject, the point about his “parallel” discovery about the parallels was that he got stuck in the same wrong question as Bolyai “which is the truth”.
A new approach started few decades later by the German mathematicians to make models.
This means the simple idea that we regard not real lines as the lines.
The old axioms that every two points determine a line, every two lines cross in a point, and so on then could be still satisfied by the non real lines. The simplest was actually already known.
It’s called spherical geometry and it regards the main circles like the equator as line.
Of course, two such main circles on a sphere cross not in a single point rather in two diametrical.
But that just should mean to regard these pairs as a single point! In this model by the way, the parallelity axiom is false because there are no non crossing lines at all.
The triangles having more than 180 angle total is also very visible here.
Unfortunately, the infinity of the line is not true either.
More complicated models were created where even the distance was distorted and thus the “infinity” of the lines could be satisfied in a bounded region.
As always, it was not the smartest one who realized the biggest truth.
The coin dropped only for Beltrami, that we should stop with these models now for a second!
Because we already got what we wanted in the first place. Indeed, what started these new models?
The chase for a derivation of the ugly parallelity axiom. The searches for contradiction with assuming the falsity lead to the new worlds and then others made models for such.
In these models all the axioms are true and yet there can be more non crossing ones.
But if the other axioms would imply by some “Logic” the parallelity axiom, that is the uniqueness of the non crossings then this “Logic” would have to be true in these models too and thus force the unique parallelity from the other axioms being true.
So if we assume that reality as models obey logic then we already proved by the sheer existence of these models that the parallelity axiom is not derivable from the other axioms.
The coin dropped but the coin has an other side too.
If a logic is complete then it not only must obey these realities but in reverse, if we only regard those realities where some axioms are true and find something that is always true then that should be derivable by our logic.
This completeness of the new Logic was only proved by Gödel in 1930. And that ominous conference was the start of everything. An other opposite “incompleteness” of our axiom systems was explained in the intermissions to the understanding ears.

Robinson Arithmetic again, revealing its magic

Now we can show why $x' \neq x$, $x + y = y + x$, $x \cdot y = y \cdot x$ are not derivable from 1. – 7.
Indeed, it is enough to show models where our axioms 1. – 7. are true but these three are false.
But even before that, we can also have little glimpse though not a real justification why Robinson chose 7. as new axiom to replace induction. Indeed, the full justification will only be the actual magic of his system to which we'll come soon. But even now we can already see that without 7. things are boring. Namely, if we regard the S set of all those s numbers that are not 0 and yet they have no previous then they are just simple new staring values taking the role of 0.
Just regarding one ω new start, our numbers are: $0 , 0' = 1 , 0'' = 2 , \ldots \omega , \omega' , \omega'' , \ldots$
Of course, we must tell also how the two operations behave but this is quite predictable too.
Accepting 1. – 7. and seeing that $x' \neq x$, $x + y = y + x$, $x \cdot y = y \cdot x$ are still not derivable are more interesting. We might even think that now the previous simple ω restart is impossible.
But the first case, that is finding a model where for some x we have $x' = x$ allows an even simpler situation as: $0 , 0' = 1 , 0'' = 2 , \ldots \omega = \omega'$.
Of course, beside $\omega = \omega'$ we must tell again how the operations behave if they involve ω.
Using 4. $x + \omega = x + \omega' = (x + \omega)'$ which implies $x + \omega = \omega$ if only the single ω is successor of itself. Similarly, but using 6. $x \cdot \omega = x \cdot \omega' = (x \cdot \omega)'$ so $x \cdot \omega = \omega$ too.
We might jump to say that then simply ω is a self preserving infinity. But by 5. $\omega \cdot 0 = 0$.
So the operation values involving ω are always ω except $\omega \cdot 0 = 0$.
Easy to see that then indeed all seven axioms will be obeyed.
Interestingly, since we derived that $x \cdot \omega = \omega$ must be for all x and thus $0 \cdot \omega = \omega$ too, but as we just saw $\omega \cdot 0 = 0$, thus in this model $x \cdot y = y \cdot x$ will not be true either.
To make a model where $x + y = y + x$ fails, we need two new unnamed numbers u , v that are successors of themselves. Then naturally we must have $u + n = u , v + n = v$ too but surprisingly, we must also have $x + u = v$, $x + v = u$. And then of course, $u + n \neq n + u$, $v + n \neq n + v$ so indeed $x + y = y + x$ will be violated, actually in all cases where either u or v occurs once.
To define multiplication we obviously set $u \cdot 0 = 0$ and $v \cdot 0 = 0$.
For the other u , v involved cases if u is first member the result is v and same way in reverse.
For the remaining n named first member cases the result is same as the second member.
An amazing postscript to all this is that if we accept $x' \neq x$ as axiom 8. and thus exclude our simple models just described then the possibility of weird models with new elements still remains.
It is evident that now a new element ω can not work alone because it needs previous ones by 7. but it can not be itself by 8. It's not that hard to see either that finite many new members are not enough now and actually two sets of forward and backward successors from ω must exist.
So a full model must look like:

$$0 , 1 , 2 , 3 , \ldots$$
$$\ldots , -3 , -2 , -1 , \omega , +1 , +2 , +3 , \ldots$$

The first line is the normal natural numbers abbreviated from their 0'''' forms and the second line are the extra non standard numbers.
And indeed, these objects together satisfy 1. , 2. , 7. , 8. because:
0 has no previous, the previous is unique, a non 0 has previous and nothing is previous of itself.
The really shocking fact is that including axioms 3.– 6. and even $7(\infty)$, such strange models will still exist. Not with a single such double infinite sequence but with infinite many added.
The big question we ask naturally is how the hell the additions and multiplications will look like.
Not surprisingly, it can be shown that they can not be given effectively. Only their existence follows by Zemelo's mentioned new Set Theoretical axiom.
Now we can return to our opposite goal, to see what this weak Robinson Arithmetic can do.
But first let's regard the full Peano system again.
The most obvious still missing concept is the smaller bigger relation but it's easy to define as:
$x \leq y \leftrightarrow \exists z (x + z = y)$

The addition, multiplication and this new relation are all dually effective, so we can recognize not only their cases for concrete numbers but also their negatives.
If for example two numbers don't sum into a third then this is recognizable because we can calculate the sum and verify that it is not our third number.
$x \leq y$ itself is recognizable because we only have to check z values in its definition up to y.
Its negative is recognizable as $y < x$ which is same as $y \leq x$ and $y \neq x$.
Using only $\forall z \leq . .$ and $\exists z \leq . .$ type of so called bounded quantifications, the remaining open variables will again define relations that the tuple sets satisfying them are dually effective.
This class, which we should call as bounded arithmetical relations is very wide.
Any natural example of dually effective tuple collection is easily expressible as such.
But diagonality can do the trick to find a missing one.
Indeed, since the boundedly quantified expressions are effectively sequencable, we can also get a P_1 , P_2 , . . property sequence among them. First we regard the D diagonal set by simply collecting every such n number that happens to be in P_n. This D has no apparent reason not to be in our list so we regard D's complement $\neg D$ and it definitely can't be there.
Indeed, if P_n contained n then $\neg D$ will not and if P_n didn't contain n then $\neg D$ will.
So $\neg D$ differs from every P. Thus of course we can see now that already D wasn't there since we did list the complement of every property.
Such dual effectivities are always imperfect and serve only to be extended to the then perfect class of non dual effectivities. For the dually effective primitive recursive functions such extension is Kleene's effective μ^* operator. But in this arithmetical approach not only the dual class is much nicer but the extension will be even better. Namely, we apply any existential quantifications.
Allowing any quantifications, the relations of the remaining free variables are called arithmetical.
These existential arithmetical relations are the best definition of the effective tuple collections.
Indeed, in a single slash we allowed all reasonable, logically expressible searches!
With the stupid old naming, these are the recursively enumerable, or with the even more stupid new naming, the computably enumerable tuple collections. I just call them effective.
Turing's collection method to regard one of his machines and regard the inputs from which the machine halts is the best definition. It has a simplicity and independence from number relations.
Using texts instead of number tuples allowed this and this is also a conceptional advantage.
The negative side of Turing's definition is that it doesn't reveal at all what it grasps!
For a practical effective collection it can be very hard to find the Turing machine that does it.
The existential arithmetical relations are definitionally broad. They tell what they grasp.
The negative of our existential arithmetical relations are the universal ones.
These are not expectable to be recognizable by a machine. But it could happen and then these are the dually effective tuple sets that were missing from the bounded ones.
The Peano system is designed to derive universalities, so it is expectable that it could derive these.
Now comes the first surprise: The Robinson system can do the same.
We might be less impressed if we realize that these missing dual effectivities should have equivalent existential forms and these being derivably represented is not such a big surprise.
This merely means that all generable sets are representable and so the non-theorems are non generable. Then of course we have undecidable statement. But we already knew that from $x' \neq x$.
Its negative $\exists x\, (x' = x)$ should be again non derivable if we can only derive true statements.
Of course, do realize that this undecidability of $x' \neq x$ was shown by external observations, through models not in the system itself. But this is always so.
Now comes a second surprise as merely a claim: Every consistent and effective extension of Robinson's system must also have non generable non-theorems. So in particular, putting back $7(\infty)$ as axiom we get that the Peano system has non generable non-theorems.
Now you can say again, big deal, we already knew that for the Peano system.
But this result shows that there is something in the Robinson system already making it unfixable.
No matter what new axioms we add, it will always have undecidable statement pairs.
The third surprise will be a connection from the first surprise to the second.
To pin point the inheriting entity of the Robinson system as not separable sets of statements.

This actually suggests the road to find these sets. It was already there in our arguments about the quantor forms but we didn't pay attention to it. It is a new better representability of the dually generable sets, that is the M , $\neg M$ selectors. Namely, not by some unrelated Q , R properties but by Q , $\neg Q$. Then indeed, these representation cases form two sets that can not change in any consistent extension. So this perfect representation remains.
But now we could say, that's not good since we needed a representation of all effective sets.
Luckily, we can reformulate our earlier argument into a version requiring only representation of the selector pairs. We might think that a perfect representation will help doing this, but not!
We do not assume this and so we get a weaker consequence too.
To make things perfectly clear we show the earlier argument first and then the new one:

Not Generable Argument:
If every $[M]$ effective number set can be represented by a $Q(\ )$ property:

$$n \in [M] \quad \leftrightarrow \quad \vdash Q(n)$$

and $\vdash$ is consistent then the non-theorems is not an effective set and so they are not generable.

Lemma:
$D = \{ \langle P \rangle ; \ \vdash P(\langle P \rangle) \}$ is the diagonal derivability code set. Its complement:
$\neg D = \{ n ; n \neq \langle P \rangle \text{ or } (n = \langle P \rangle \text{ and } \neg \vdash P(n)) \}$ is not effective if $\vdash$ is consistent.

If $\neg D$ is effective then a Q represents it and so:

$$n \in \neg D \quad \leftrightarrow \quad \vdash Q(n).$$

Using n as $\langle Q \rangle$:

$$\langle Q \rangle \in \neg D \quad \leftrightarrow \quad \vdash Q(\langle Q \rangle)$$

But also:
$\langle Q \rangle \in \neg D \quad \leftrightarrow \quad [\langle Q \rangle \neq \langle P \rangle \text{ or } (\langle Q \rangle = \langle P \rangle \text{ and } \neg \vdash P(\langle Q \rangle))] \quad \leftrightarrow \quad \neg \vdash Q(\langle Q \rangle).$
We got a contradiction in $\vdash$ and so if $\vdash$ is consistent then $\neg D$ is not effective.
The non effectivity of $\neg D$ implies that the corresponding $\neg \vdash P(\langle P \rangle)$ diagonal non-theorems are not effective either and that implies that all the non-theorems are not effective either because the diagonal ones were recognizable among them.

The name of our argument and choosing generability as consequence makes good sense.
Indeed, this non generability of the non-theorems has a heuristic continuation to show indirectly the existence of undecidable statement pair in $\vdash$ if it is consistent and has generable axioms.
Indeed, suppose that there were no undecidable A , $\neg A$ statement pairs, that is for all A statements A or $\neg A$ were derivable theorems.
By consistency then this "or" is actually an either-or and thus if our axioms are generable and so the theorems are too, then we get an instant generation of the non-theorems.
Namely, we should generate the theorems and then simply negate them.

Not Selectable Argument:
If every $[M]$, $[\neg M]$ selectable number sets can be represented by Q , R properties:

$$n \in [M] \quad \leftrightarrow \quad \vdash Q(n) \qquad \text{and} \qquad n \in [\neg M] \quad \leftrightarrow \quad \vdash R(n)$$

then the theorems and non-theorems are not selectable.

Enough to show that the $D = \{ \langle P \rangle ; \ \vdash P(\langle P \rangle) \}$ diagonal code set and its complement:
$\neg D = \{ n ; n \neq \langle P \rangle \text{ or } (n = \langle P \rangle \text{ and } \neg \vdash P(n)) \}$ are not a selectable pair of number sets.

Indeed, then the diagonal theorems and diagonal non-theorems are neither and thus the set of all theorems and non-theorems are neither.
If D , $\neg D$ were selectable then $\vdash$ would have to be consistent and also $\neg D$ effective.
But the Lemma in the Generability Argument proof forbids this.

You might say, why was this a weaker result?
The theorems are always generable so we got what we got earlier anyway, that the non-theorems are non generable. But do realize that if the axioms are not generable then the theorems can be non generable too. This is not true for the Robinson system but can be true for an extension of it.
And indeed, if for example we add all true statement as axioms, we only get by our weakened result that at least one of the theorems or non-theorems is not effectively collectable.
Of course, they are perfect negatives, so if one were generable, the other had to be too.
So we get eventually that they are both non effective.
In normal, not only consistent but also effective extensions like adding the $7(\infty)$ scheme back again, we do get that the non-theorems are not an effective set.

Now we show that the Not Selectable Argument is not necessary to see that the Robinson system inherits its non selectability.
Plus we get a much better vision for what really inherits.
The representation cases with opposite Q , $\neg Q$ properties are just special cases of something.
Indeed, in general too, the negative of a derived theorem should be called an “anti-theorem” or more meaningfully an “intentional non theorem”.
Such are meant to be non derivable because otherwise we would have a contradiction.
Conventionally, they are called the refutable statements.
The simple truth is that the theorems and anti-theorems of the Robinson system are not separable!
So no selector pair can contain them and this includes theorems and non-theorems of extensions.
In fact, we can show special theorems and anti-theorems that are already not separable.

Not Separable Argument
If every M , $\neg M$ selectable sets can be represented by a single Q :
$n \in [M] \quad \leftrightarrow \quad \vdash Q(n) \quad$ and $\quad n \in [\neg M] \quad \leftrightarrow \quad \vdash \neg Q(n)$
then the theorems and anti-theorems are not separable.

Not surprisingly, the used special theorems are the diagonal statements:
$D = \{ P(<P>) ; \vdash P(<P>) \}$. But now we don’t form the complement that was:
$\neg D = \{ P(<P>) ; \neg \vdash P(<P>) \}$ rather the “total opposite” or “anti diagonal” set:
$D\neg = \{ P(<P>) ; \vdash \neg P(<P>) \}$.
We can prove that D and $D\neg$ are not separable.
Indeed, if they were then we had some M , $\neg M$ that:
$\vdash P(<P>) \rightarrow <P> \in [M] \quad , \quad \vdash \neg P(<P>) \rightarrow <P> \in [\neg M]$
regarding the Q , $\neg Q$ that perfectly represents $[M]$, $[\neg M]$ we would have:
$\vdash P(<P>) \rightarrow \vdash Q(<P>) \quad , \quad \vdash \neg P(<P>) \rightarrow \vdash \neg Q(<P>)$. At $P = \neg Q$ we get:
$\vdash \neg Q(<\neg Q>) \rightarrow \vdash Q(<\neg Q>) \quad , \quad \vdash \neg \neg Q(<\neg Q>) \rightarrow \vdash \neg Q(<\neg Q>)$.
So assuming consistency and that $\neg \neg Q \rightarrow Q$, we get a contradiction.
This then implies that the full set of theorems and anti-theorems are not separable either.

Back to the bigger picture, undecidability of the full Number Theory came out from a sub system, the Robinson Arithmetic, where undecidability is trivial from the outside.
All this shows clearly that the point is not undecidability, rather non generable complements, that is non selectable sets, or even more, sets that are not separable.

Not separable sets through machines

In spite of my repeated emphasis about the conquest of Effectivity over Language, the last few sections may suggest otherwise. But as we'll see now, we don't need axiom systems and perfect representation to obtain sets that are not separable.
Instead, we use again the ΔU diagonal machine.
Let's recall the text set that has no effective complement:
$[\Delta U] = \{ T ; T\downarrow \Delta U \} = \{ T ; T|T\downarrow U\} = \{ T ; T\downarrow | T |\}$

Now we'll regard two possible A , B results for these diagonal haltings.
Remember, that we denoted an R result written as $T\downarrow M = R$.
Actually we never used these R results in our theorems up until now.
Now we'll do and as start we go deeper because we use two not so trivial facts.
Firstly, that we can make a machine from a M that halts only when M halts on A.
This machine will be abbreviated as $M = A$.
Secondly, that we can alter the result of an M at every halting to be an A fix text.
This machine will be abbreviated as $M \Rightarrow A$.
Our claim is simply that $[\Delta U=A]$ and $[\Delta U=B]$ are not separable.
Obviously, the unused $\Rightarrow$ machine creation will be important in the proof itself.
Also, faithfully to our didactical approach, the proof should be not indirect rather exhibit some shared inputs. The claim that no selector pair exist means that there are no **A** , **B** machines that:
$T\downarrow(\Delta U=A) \rightarrow T\downarrow \mathbf{A}$, $T\downarrow(\Delta U=B) \rightarrow T\downarrow \mathbf{B}$ and for every T, either $T\downarrow \mathbf{A}$ or $T\downarrow \mathbf{B}$.
We show that the first two conditions and the either part of the third, makes the or part impossible.
Let Θ be the machine that for any T input runs **A** and **B** parallel and if **A** stops then displays B as result and if **B** stops displays A. In other words, $\Theta = \mathbf{A} \Rightarrow B$ and $\mathbf{B} \Rightarrow A$.
By our either assumption Θ is meaningful, that is the two and alternatives don't interfere.

The use of Θ as notation reflects that if both sides were effective then it were a transformer.
The impossibility of $T\downarrow \mathbf{A}$ or $T\downarrow \mathbf{B}$ standing for all T follows if we can exhibit some T where $T\uparrow \mathbf{A}$ and $T\uparrow \mathbf{B}$. This thus implies that Θ is not total so it is not a text transformer after all.
Such shared run T input is $<\Theta>$. Indeed:

$$<\Theta>\downarrow \mathbf{A} \rightarrow <\Theta>\downarrow \Theta=B \rightarrow <\Theta>\downarrow \Delta U=B \rightarrow <\Theta>\downarrow(\Delta U=B) \rightarrow <\Theta>\downarrow \mathbf{B}$$

$$<\Theta>\downarrow \mathbf{B} \rightarrow <\Theta>\downarrow \Theta=A \rightarrow <\Theta>\downarrow \Delta U=A \rightarrow <\Theta>\downarrow(\Delta U=A) \rightarrow <\Theta>\downarrow \mathbf{A}$$

So the two initial conditions of halts are both impossible by the assumed either condition.
As we see, some indirectness still remained in our argument.

Our result is transferable to the partial functions of Kleene and in particular:
For any two a , b natural numbers, the $A = \{ n ; | n |(n) = a \}$ and $B = \{ n ; | n |(n) = b \}$ sets are not separable. Remember that $| n |$ denotes the partial function enumerated by n.
By regarding only those n values where $| n |$ is unary, we get an $| x |(y) = f(x , y)$ two variable partial function. This melts into those undefined cases where $| x |$ is unary but is undefined at y.
So the end result is that $| x |(y)$ is a perfectly defined $f(x , y)$ partial function.
$d(x) = | x |(x)$ is the diagonalization of this f , that is regarding the $f(x , x)$ function.
This then is a unary partial function with infinite many concrete numbers that enumerate it.
However we pursue an other concrete c code number, namely for that recursive function that would separate A and B with b and a values, that is:
$n \in A \rightarrow | c |(n) = b$ and $n \in A \rightarrow | c |(n) = b$. Indeed, this is impossible because then:
$c \in A \rightarrow | c |(c) = b \rightarrow c \in B$ and $c \in B \rightarrow | c |(c) = a \rightarrow c \in A$.

Robinson Arithmetic, the third time

Now we present a little final gem. It is essentially an application of the previous section's last observation by returning to axiom systems, namely to the Robinson system.
Unlike above, here we'll pursue something about $d(x) = |\, x \,|(x)$ that will show why the set of theorems and set of anti-theorems of this system are non selectable. Which of course proves that this system can not be extended to be complete with any effective set of new axioms.
This crucial thing about $d(x)$ is of course just a special application of something more general:

Lemma assumed:
For every $f(x)$ unary partial function there is an $R(x, y)$ relation built up from addition and multiplication with logic, that for every m, n numbers, the Robinson system's $\vdash$ derivation can imitate the f function with R as:

$$f(m) = n \;\rightarrow\; \vdash\; \forall y\, [\, R(m, y) \leftrightarrow y = n\,].$$

Proof of our claim:

Let D be the relation claimed by the Lemma for the $d(x) = |\, x \,|(x)$ function and suppose that there were some $M, \neg M$ selector pair for the theorems and anti-theorems.
Then let's define the following effective total function:

$$g(m) = \begin{cases} 1 \text{ if } D(m, 0) \in [M] \\ 0 \text{ if } D(m, 0) \in [\neg M] \end{cases}$$

Let $g = |\, c \,|$ and then:

$$D(c, 0) \in [M] \;\rightarrow\; g(c) = |\, c \,|(c) = d(c) = 1 \;\rightarrow\; \vdash\; \forall y\, [\, D(c, y) \leftrightarrow y = 1\,] \;\rightarrow$$

$$\vdash\; [\, D(c, 0) \leftrightarrow 0 = 1\,] \;\rightarrow\; \vdash\; \neg D(c, 0) \;\rightarrow\; D(c, 0) \in [\neg M].$$

Also:

$$D(c, 0) \in [\neg M] \;\rightarrow\; g(c) = |\, c \,|(c) = d(c) = 0 \;\rightarrow\; \vdash\; \forall y\, [\, D(c, y) \leftrightarrow y = 0\,] \;\rightarrow$$

$$\vdash\; [\, D(c, 0) \leftrightarrow 0 = 0\,] \;\rightarrow\; \vdash\; D(c, 0) \;\rightarrow\; D(c, 0) \in [M].$$

Thus our assumption of such $M, \neg M$ existing, was false.
All this makes it even clearer why the Robinson system can not be "fixed".
All future $M, \neg M$ potential selector pairs are refuted by the effective unary functions contained in the system already.

Appendix: Halting Calculus, A texts only approach to Effectivity.

P , Q , R , S , T are variables for text, program, machine, effective property.
Θ is a variable for special machines, called transformers.
U is a fix constant machine.
Δ , Φ , Γ , Π , Σ are fix constant transformers.

S↓P = S halts in P
S↑P = ¬(S↓P) = S doesn't halt = S runs in P (forever)
[P] = { S ; S↓P } = effective set
¬[P] = complement set of [P] = all the texts that are not in [P].
¬P = complement machine of P is such that [¬P] = ¬[P].

For most P this complement doesn't exist!
That is, the [P] effective set's complement is usually not effective.
All the followings are a clarification of this claim.

1. There is a | text coding that combines two texts into S|T.

The simplest solution to separate two texts would be to regard our | itself as a special symbol. Unfortunately, by this we increase our alphabet and so we couldn't claim results for any fix alphabet. Luckily, there is no need for this and a simple idea similar to what Turing used in his original idea is to double the symbols of S and then use a pair that has two different symbols. Even the simplest dual 0 , 1 alphabet is enough for this where thus the separator is 01 or 10. The machine that wants to use S|T can then start the text from left and regard only the single of the doubled symbols until a first non doubled as S and then read the next T section as it is.

2. There is a U universal machine that imitates every P because: S|P↓U ↔ S↓P.
We introduce the Θ variable that represents machines that halt from all inputs.
The Θ refers to transformer because they are actually effective text transformers.
Since for these the halting is sure we omit ↓.
Also, we can continue the transformer with an equal sign and tell the result: SΘ = R.
Usually, we will not use such equal signs because we'll use these results further.
Then the transformer's continuation means using the result of it automatically.
Such continuation can be claiming a halt or run : SΘ↓P or SΘ↑P.
But can be also a new transformer : SΘΘ'↓P.
In a sense they are like multiplication while the | coding was more like an addition.
This reflects our agreement about their hierarchy level that transformation overrides |.
So S|TΘ↓P automatically means S|(TΘ)↓P while for (S|T)Θ↓P we can not omit the parenthesis.
Observe that our transformer continuation actually hides an assumption:

3. Transformer imbedding. SΘ↓P ↔ S↓ΘP

Thus ΘP can become a new text and then it can be also followed by a transformer so we need parenthesis: (ΘP) Θ'.

4. There is a Δ transformer that can create a coded double: T|T = TΔ, so it is a duplicator.
Of course, the actual claim is about haltings for a P and then it can be continued with using 3.:
T|T↓P ↔ TΔ↓P ↔ T↓ΔP.

Applying this in reverse with P as U :
$T\downarrow \Delta U \quad \leftrightarrow \quad T\Delta\downarrow U \quad \leftrightarrow \quad T|T\downarrow U \quad \leftrightarrow \quad T\downarrow T.$

Doesn't seem interesting but observe the followings:
We can call an T input as "shared" by two P , Q machines if they both halt or not from T.
Now if a Q is $\neg P$ then and only then P and Q can not share any T input.
So if a P shares input with every machine, then it can not have a complementing machine!
By our result, ΔU shares an input with every T machine, namely T itself and so ΔU is a machine that has no complement.
Of course, we can establish a non existing complement by deriving a contradiction with already proved cases too. For example, we can show that U itself can not have a complement.
Indeed, suppose it had the $\neg U$ complementing machine. Then we had:
$T\downarrow \Delta U \quad \leftrightarrow \quad T\Delta\downarrow U \quad \leftrightarrow \quad T\Delta\uparrow \neg U \quad \leftrightarrow \quad T\uparrow \Delta\neg U.$
And so ΔU had the $\Delta\neg U$ complement that we proved not to exist.
Non indirect arguments exist too but for those we must guess the shared input for arbitrary T.
In our case it is $\Delta T\,|\Delta T = (\Delta T)\Delta$ because:
$\Delta T\,|\Delta T\downarrow U \quad \leftrightarrow \quad \Delta T\downarrow \Delta T \quad \leftrightarrow \quad (\Delta T)\Delta\downarrow T \quad \leftrightarrow \quad \Delta T\,|\Delta T\downarrow T$
Or:
$(\Delta T)\Delta\downarrow U \quad \leftrightarrow \quad \Delta T\,|\Delta T\downarrow U \quad \leftrightarrow \quad \Delta T\downarrow \Delta T \quad \leftrightarrow \quad (\Delta T)\Delta\downarrow T$

Soon we'll see an infinity of shared inputs with U without guessing.

5. Partial input alteration.
The machine continuation for a combined $S|T$ input means that $(S|T)\Theta\downarrow P \quad \leftrightarrow \quad S|T\downarrow \Theta P.$
If only one, say the second part of the input is altered by an Θ transformer then the building of Θ into a next used P machine is more complicated but still obtainable by a $(\Theta|P)\Phi$ machine.
So: $S|T\Theta\downarrow P \quad \leftrightarrow \quad S|T\downarrow(\Theta|P)\Phi.$
A particular example of Θ could use a doubled $T|T$ but we could still build Θ into P.
So: $S|(T|T)\Theta\downarrow P \quad \leftrightarrow \quad S|T\downarrow(\Theta|P)\Gamma.$

These Φ , Γ are concrete fix machines. The more important Π comes next and finally Σ.

6. Parameter Theorem.
An other reason for regarding the input as combined $S|T$ can be not that we alter T rather we want to get rid of it completely! Meaning that we want to build it into a next used P.
So: $S|T\downarrow P \quad \leftrightarrow \quad S\downarrow(T|P)\Pi$
So this fix Π imbeds T as a parameter in the new program that will use only S.

Recursion Theorem.
For every P there is a P^U that for all S : $S|P^U\downarrow P \quad \leftrightarrow \quad S\downarrow P^U.$

The notation makes sense since P^U is like a "personal" universal machine for P.

A concrete such P^U is $((\Pi|P)\Gamma|(\Pi|P)\Gamma)\Pi$. Indeed:

$S|((\Pi|P)\Gamma|(\Pi|P)\Gamma)\Pi\downarrow P \quad \leftrightarrow \quad S|(\Pi|P)\Gamma\downarrow(\Pi|P)\Gamma \quad \leftrightarrow \quad S\downarrow((\Pi|P)\Gamma|(\Pi|P)\Gamma)\Pi$

The first $\leftrightarrow$ is true by Γ's definition in 5. with $T = (\Pi|P)\Gamma$, $\Theta = \Pi$.
The second $\leftrightarrow$ is true by the Parameter Theorem with T and P both being $(\Pi|P)\Gamma$.

As we said, P^U is like a personal universal machine for P. But continuing our result to involve the real universal machine U : $S|P^U\downarrow P \quad \leftrightarrow \quad S\downarrow P^U \quad \leftrightarrow \quad S|P^U\downarrow U.$

And so any P shares all the $S|P^U$ texts with U.
Unfortunately, it still just proves that U has no complement. So now we'll step beyond this.

Fix Point Theorem.
For every Θ transformer there is a Q program that: $[Q] = [Q\Theta]$.
That is: For every T input $T{\downarrow}Q \leftrightarrow T{\downarrow}Q\Theta$.

Let P be such that for all S, T we have $T{\downarrow}S\Theta \leftrightarrow S|T{\downarrow}P$.
Then let P^U be the text guaranteed by the Recursion Theorem. Then:
$T{\downarrow}P^U \leftrightarrow P^U|T{\downarrow}P \leftrightarrow T{\downarrow}P^U\Theta$. So such Q is P^U.

The first $\leftrightarrow$ is true by the Recursion Theorem.
The second $\leftrightarrow$ is true by our definition of P with $S = P^U$.

Two P, Q texts are variants if they collect the same texts: $[P] = [Q]$.
A **T** set of texts is variant complete if with any member, the variants are members too.
If a set is variant complete then its complement is variant complete too, so actually any separation of all texts can be variant complete.

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 an 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 Q text we would have that $[Q\Theta] = [Q]$.
Thus $Q\Theta$ and Q are variants and so they should be in the same of either **A** or **B**.
But by the definition of F we have that $F \in \mathbf{A} \rightarrow Q\Theta = B \in \mathbf{B}$ and $Q \in \mathbf{B} \rightarrow Q\Theta = A \in \mathbf{A}$.

Our second proof will tell exactly a side of the two variant complete separation which is non effective for sure. Namely, the side that contains all never halting programs.
We show that if this side were effective then all programs had a complement.
So suppose N would halt from all never halting programs and a member of the other side were P.
Then:
For all S $[S{\downarrow}P \leftrightarrow S{\downarrow}R] \rightarrow R{\uparrow}N$ and
For all S $[S{\uparrow}R] \rightarrow R{\downarrow}N$.

Now we claim a $P\&Q$ machine that uses $S|T$ combined input and halts as:
$S|T{\downarrow}P\&Q \leftrightarrow S{\downarrow}P$ and $T{\downarrow}Q$.
To achieve such machine we simply have to run one of them and if that halts start the other.
The better usable two facts for our proof are:
$T{\downarrow}Q \rightarrow$ For all S $[S{\downarrow}P \leftrightarrow S|T{\downarrow}P\&Q]$ and
$T{\uparrow}Q \rightarrow$ For all S $[S|T{\uparrow}P\&Q]$.
Using for these both the Parameter Theorem, then the two N conditions and then again the Parameter Theorem:
$T{\downarrow}Q \rightarrow$ For all S $[S{\downarrow}P \leftrightarrow S|T{\downarrow}P\&Q \leftrightarrow S{\downarrow}(T|P\&Q)\Pi] \rightarrow$
$(T|P\&Q)\Pi{\uparrow}N \leftrightarrow (T|P\&Q){\uparrow}\Pi N \leftrightarrow T{\uparrow}(P\&Q|\Pi E)N$.

$T{\uparrow}Q \rightarrow$ For all S $[S|T{\uparrow}P\&Q \leftrightarrow S{\uparrow}(T|P\&Q)\Pi] \rightarrow$
$(T|P\&Q)\Pi{\downarrow}N \leftrightarrow (T|P\&Q){\downarrow}\Pi N \leftrightarrow T{\downarrow}(P\&Q|\Pi N)\Pi$.

So the complement of Q would be $(P\&Q|\Pi N)\Pi$.
The main application of Rice's Theorem is if we know one side that is effective for sure.
To get such side can be helped by our next axiom:

7. There is a Σ transformer that $S\downarrow P \leftrightarrow P\downarrow S\Sigma$.

We could claim a Π' analogue of Π that uses the first variable in $S|T$ as parameter.
That is: $(S|T)\downarrow P \leftrightarrow T\downarrow(S|P)\Pi'$. Then: $S\downarrow P \leftrightarrow S|P\downarrow U \leftrightarrow P\downarrow(S|U)\Pi' \leftrightarrow P\downarrow S\Sigma$.
So Σ is doing Π' but with the U universal machine built in fix.

For any S the $[S\Sigma]$ collection is variant complete! That is, for every P, Q we have that:

$(P\downarrow S\Sigma$ and $[P] = [Q])$ $\rightarrow$ $Q\downarrow S\Sigma$. Indeed:
By 7. $P\downarrow S\Sigma$ can be replaced by $S\downarrow P$.
$[P] = [Q]$ means "for all T $(T\downarrow P \leftrightarrow T\downarrow Q)$".
Thus the $(\ )$ condition trivially implies $S\downarrow Q$.
Finally, by 7. again we get the conclusion from this.
So by Rice's Theorem $S\Sigma$ has no complement.
In fact, $S\Sigma$ shares some P input with every Q and this has an amazing meaning.
First of all, the exact meaning is that:
For all S, Q there is a P that $P\downarrow S\Sigma$ and $P\downarrow Q$ or $P\uparrow S\Sigma$ and $P\uparrow Q$.
That is, $S\downarrow P$ and $P\downarrow Q$ or $S\uparrow P$ and $P\uparrow Q$.
So then P is actually a counter example for Q if it were intended as a machine recognizer of P not collecting the S member. Indeed, then for all P we should have that:
$S\uparrow P \leftrightarrow P\downarrow Q$ which is the exact opposite of what we obtained.
So even such simple behavior that a fix S doesn't halt in P, can not be perfectly recognized by an other machine examining P. The " perfectly" is the essence of course!
Because we could make a Q that halts if S doesn't halt in P by for example making Q halt from everything! But then the reverse is not true that Q only halts for these P.