Contents

1. Peano Rules

The natural numbers can be imagined as singular objects, that are ordered by the consecutiveness relationship $\lhd$. So in short: $1 \lhd 2 \lhd 3 \lhd \ldots$
The actual decimal forms are immaterial here. The $\lhd$ consecutiveness tells everything and even the decimal forms could be obtained from it.
Addition is a relationship between three numbers. So $x + y = z$ is merely a special form of $R(x, y, z)$. To define addition, we have to collect all those x, y, z triplets for which $R(x, y, z)$ stands. Or R itself can be imagined as the set of these triplets.
Peano realized first formally that the $\lhd$ consecutiveness determines addition by two simple rules. The first tells that addition of 1 is merely the consecutiveness itself:
$(x \lhd z) \rightarrow (x + 1 = z)$
The second rule says that if $x + \bar{y} = \bar{z}$ stands for three numbers, then the consecutive of $\bar{y}$ used instead of $\bar{y}$ and the consecutive of $\bar{z}$ instead of $\bar{z}$, will give a new triplet:
$(x + \bar{y} = \bar{z}\ ,\ \bar{y} \lhd y\ ,\ \bar{z} \lhd z) \rightarrow (x + y = z)$
We'll keep this notation of barred variables denoting values, from which we imply cases that use only the basic letters of $R(x, y, z, \ldots)$
These two rules allow to derive all cases of additions from the consecutivenesses.
For example, lets derive the $4 + 3 = 7$.
By rule 1, $(4 \lhd 5) \rightarrow (4 + 1 = 5)$
By rule 2, $(4 + 1 = 5\ ,\ 1 \lhd 2\ ,\ 5 \lhd 6) \rightarrow (4 + 2 = 6)$
By rule 2 again, $(4 + 2 = 6\ ,\ 2 \lhd 3\ ,\ 6 \lhd 7) \rightarrow (4 + 3 = 7)$
Observe, that in spite of the triviality of the whole process, we had to make smart choices in order to achieve our end result. But we also feel, that possibly there is a way to go blindly and derive all derivable cases in some order. So even a machine could generate these. We'll investigate this later.
Multiplication is derived through addition.
Here, the first rule is that multiplying by 1, keeps the value, that is $x \bullet 1 = x$.
The second rule says that increasing the multiplier by 1, will add x to an earlier $\bar{z}$:
$(x \bullet \bar{y} = \bar{z}\ ,\ \bar{y} \lhd y\ ,\ \bar{z} + x = z) \rightarrow (x \bullet y = z)$
Now the derivation of a case like $4 \bullet 3 = 12$ is even more complex, because we have to go through cases of additions. Yet, in finite steps, using only the four rules, the two of addition and two of multiplication, we can get $4 \bullet 3 = 12$ purely from $\lhd$ cases.
Finally, exponentiation is done through multiplication.
$x^1 = x$ again is similar as at multiplication and then here, the consecutive exponent means multiplication: $(x^{\bar{y}} = \bar{z}\ ,\ \bar{y} \lhd y\ ,\ \bar{z} \bullet x = z) \rightarrow (x^y = z)$
In spite of the crystal clarity of this method, there are a lot of nagging questions too.
It's quite plausible that all cases of the operations are obtainable through finite many steps from cases of $\lhd$. But we have to be innovative. A mentioned totally mechanical or blind method of deriving all cases, seems to be complicated. An opposite need for more effective realization, would find the problem, not in the ad-hocness, how we use the rules, rather in the objects themselves. We have infinite many individual objects. How can that be realized physically? The most obvious is to represent every number with repetition of a single symbol, like a stroke: $|\ |\ | = 3$.
Finally, a third direction of questions, is not about the effectivity of our rules, rather how they determine the actual properties of the operations. Indeed, there is a mystery here. The rules all used the increase of the second member $\bar{y}$ with 1, to get a change

in the third member $\bar{z}$. This lead to a set of derivable triplets where the first two members determine the third. So we obtained operations or functions that give a unique z value for all values of x and y. This is not obvious at all! To emphasize this not obviousness, we can observe other details of the obtained operations or functions. In spite of always using only the second member to step the values, the addition and multiplication are symmetrical or exchangeable in x and y :
$4 + 3 = 7$ and $3 + 4 = 7$ both true and also $4 \bullet 3 = 12$ and $3 \bullet 4 = 12$ both true.
Even more surprisingly exponentiation is not such: $4^3 = 64$ but $3^4 = 81$.
So our derived cases show a lot of strange features, that are not apparent from the rules, yet must follow from them, through the cases. Obtaining operations, that is unique third members, is the first such feature. Of course in elementary school, we introduce these operations as operations, that is calculating a z value from x and y. Also, we visualize them later as $(x + y)$, $(x \bullet y) = (x y)$, (x^y).
This is convenient for algebra.
Indeed we can combine them easily and learn rules like $(x + y)^2 = x^2 + 2 x y + y^2$
Even the basic properties like $x + y = y + x$, $x \bullet y = y \bullet x$, $x^y \neq y^x$
are easier to express this way but the truth of these are "mystically obvious".
The operational notation can be pushed even deeper to replace the $\lhd$ consecutiveness relation with a "next" operation y' or successor function Sy .
The Peano rules then become for addition: $x+1 = Sx$ and $x + Sy = S(x+y)$
Multiplication becomes: $x \bullet 1 = x$ and $x \bullet Sy = (x \bullet y) + x$
Exponentiation becomes: $x^1 = x$ and $x^{Sy} = x^y \bullet x$
The relation derivations are much cleaner and of course not all relations are functions. In fact the simplest derivable relation is such proper relation the $<$ smallerness:
$(x \lhd y) \rightarrow (x < y)$
$(x < \bar{y} , \bar{y} \lhd y) \rightarrow (x < y)$
A more "flexible' system giving the same relation is:
$(x \lhd y) \rightarrow (x < y)$
$(x < \bar{y} , \bar{y} < y) \rightarrow (x < y)$
But we can get it from addition directly too as:
$(x + \bar{y} = y) \rightarrow (x < y)$

2. Completeness and Incompleteness

With the development of Logic, the Peano rules became statements and thus, axioms. The $\rightarrow$ in the rules is then merely the implication. The $1 \lhd 2 \lhd 3 \lhd \ . \ . \ .$ are basic statements for the names. Or with the successor function we only need one name 1 .
The cases of the operations are also statements that can be derived from the axioms by the rules of Logic or logical inferences. But now, with adding some other axioms to these newly represented Peano axioms, we can get a system, that seemingly describes the natural numbers completely. In other words, we expect that Logic can derive all truth about them, that follow through the cases.
The case statements or concrete statements differ from other statements in that they don't contain variables. The variables are all replaced by some names of naturals, or if we use the successor function, then only the single name 1 and repeated use of S appears. These two ways are exactly the same: $4 + 3 = 7$ or SSS1 + SS1 = SSSSSS1.
I prefer the first, which needs the infinite many name axioms but is still cleaner.

Statements that do contain variables are also claiming something about the variables that are not depending on the actual variables as used letters rather on how these letters appear relative to each other. They each usually appear more times and all these places mean the same hypothetical objects. The use of variables and these variable induced statements were unconscious for two thousand years. The root of this mystery is, how human languages could develop without the use of variables. On top of this mystery lies a historical one, that the two crucial forms of statement formations using variables, did become conscious without the variables themselves.
This is attributed to Aristotle, but probably he wasn't the original inventor, rather the collector of the so called syllogism rules, that involve the "every" and "there is".
So the Greeks were conscious of that these two are a strange negative of each other.
Not directly, rather through the negatives of the claims that they operate.
For example, to say that there is no broken bridge in Africa means the same as that all bridges in Africa are unbroken. The validity of this is not learnt, as smart ass developmental psychologist would say. Oh no! This is one of those apriori claims of Kant or less spelled out idealism assumptions of the Greek philosophers already.
The most important apriori human principle, the isomorphism principle will be mentioned soon too. Of course this is not the place to defend idealism. In a sense we have to skip to deeper mysteries, beyond the existence of apriories. Ontological or historical mysteries refer to ones that show how false beliefs or blind spots could keep basic abstractions hidden. This clearly reveals my view that abstractions are not made by humans, they don't evolve. They are merely discovered through getting rid of the blinding obstacles. A simple war is going on! The material universe we live in does evolve and the basic function of this evolution is the hiding of the non material universe that controls matter. Thus the evolution of hiding is also the evolution of revealing. The biggest deception is to regard this evolution as ruled by the universe we live in, or in short, ruled by matter. This shows that it is not the word evolution that is faulty, rather what we may allow under it. The real anatomy of evolution needs an understanding of time that we don't possess yet. Material "development" does spread a certain version of abstractions. But these are distorted false abstractions. And this is the crucial present battle in the war, the fight against Formalism and Metaphysics. These two are merely names for two evils that emerged in this specific time. They are good names if we dissociate them from earlier meanings. Both formalizing and metaphysical reasoning are natural and universal thought processes. When society reflects them back as limitations, then they become what they are today. So it's true that Euclid was a first clear cut formalizer when he purified his parallelity axiom and avoided the explanation of the three forms of parallelity, but we can't call him a Formalist. Neither can we call Hilbert who purified the variable usage and thus avoided the details of concretizations by names. In fact Hilbert was an anti Formalist. He said that if someone understands something truly, then he can explain that to the first person on the street. By today, we live in a world where it is completely accepted, that abstract truth is complicated, unapproachable by the first person on the street. The Wikipedia doesn't spread understanding rather dry facts to even seal the belief that everything is complicated. As a contraposition of this professional Formalism, an actionless metaphysics rules the humanities and cheap triviality of common sense rules the media. It is a War and I wasn't using this word metaphorically. Soon we all have to choose sides! So now we better return to math, to convince you to choose.
Aristotle was a relative Formalist in comparison to his master Plato. But among the dozens of total mistakes he collected, the syllogisms were a strange almost prophetic emptiness. In the dark ages Aristotle was misused and his formal logic was part of the betrayal of philosophy. So we can't be too surprised that latter the reemergence of

neither philosophy nor science wanted to touch him with a ten foot pole. If I had a time machine my first trip would be to face Newton or Gauss and just in an hour I could show them what they didn't see right in front of their eyes. That's a real mission, not taking back some gadget and dazzle them. Indeed, the Aristotelian formal logic was almost as common part of education as the ten commandments.

The new added spark that vitalized the old boring trivialities of Aristotle, was the use of variables. Above for example when we spoke of bridges in Africa, the "every" and "there is" didn't bring much exciting insight. Because we were wrong! We didn't really say what we meant. The "every" and "there is" are not talking about Africa, broken or even being a bridge. They refer to actual object that are not mentioned. So these should be variables, $x , y , z , \dots$ Then being bridge, broken, or in Africa are properties. $A(x)$ can mean being in Africa, $B(x)$ being bridge and $C(x)$ being broken.

A broken bridge in Africa simply means: $A(x)$ and $B(x)$ and $C(x)$

The existence is referring to x , so the claim is $\exists x\, [A(x)$ and $B(x)$ and $C(x)]$.

The negative of this, is: $\neg \exists x\, [A(x)$ and $B(x)$ and $C(x)]$

The old rule of negating an existence by claiming "every" and then negating, means that this is the same as: $\forall x\ \neg\ [A(x)$ and $B(x)$ and $C(x)]$

We can go further inside with the negation, by realizing that the three claims together being false, means at least one being false. But "at least one" is actually "or", so:

$\neg \exists x\, [A(x)$ and $B(x)$ and $C(x)] = \forall x\ \neg\ [A(x)$ and $B(x)$ and $C(x)] =$

$\forall x\ [\neg A(x)$ or $\neg B(x)$ or $\neg C(x)]$

Which means that for every object in the universe it is true that it is not in Africa or it is not a bridge or it is not broken. But this is not what we above claimed as trivial!

Indeed our intuitions went even further in the apriori abstractions blindly! But now we can reveal them. First we can "re-do" the "or-ing" for the first $A(x)$ or $B(x)$ as:

$\forall x\ [\neg (A(x)$ and $B(x))$ or $\neg C(x)]$.

Now comes an even more illogical change, to introduce implication.

$A \rightarrow C$ or A implies C , in math is not a cause and effect relation. A is merely an assumption which must "lead" to the C consequence. But this "lead" is not in time, it merely means that the truth of A necessitates the truth of C. Most importantly if A is false then the implication claims nothing and thus must be regarded as true. So in effect the implication means that the assumption is false or the consequence is true : $\neg A$ or C. Using this with $A = (A(x)$ and $\neg B(x))$ and $C = \neg C(x)$

$\forall x\ [\neg (A(x)$ and $\neg B(x))$ or $\neg C(x)] = \forall x\, [(A(x)$ and $B(x)) \rightarrow \neg C(x)]$.

This is exactly the claim in English as "every bridge in Africa is unbroken".

That's how "sneaky" everyday language is, smuggling in even implication.

Implication is even sneakier when it hides the non spelled out "every" and "there is", so called quantifications of the variables. For example, the Peano Rules hide a crucial oppositeness in the used variables! Indeed if we use some letters that appear both in the assumption and consequence, then it obviously stands for every number. If a letter doesn't appear in the assumption but appears in the consequence, then again it should be true that is usable for all cases. But, if a variable disappears, that is appears in the assumption but not in the consequence, then it merely means some lucky finds, that is existences, that imply the consequence cases. That's why we only used barred letters for such disappearing variables, to emphasize that they are the existences.

Disappearing variables in the implications were also the fundamental method in Hilbert's axiomatization of the quantifications, the final victory of variables.

So now we can return to the idea of adding some axioms to the Peano axioms.

The goal is that the whole system should imply all truths about the relations that we define case by case by the Peano axioms.

At a first glance this means that the axioms determine the $1 \lhd 2 \lhd 3 \lhd \dots$ names.

In other words, only these as a set should satisfy the totality of the axioms.
To show that something is problematic about this expectation, lets try just to describe the $\lhd$ consecutiveness in itself, without Peano axioms.
The infinite many statements: $1 \lhd 2$, $2 \lhd 3$, $3 \lhd 4$, . . . seem to imply that every object has a next, and each, except 1, has a previous.
Formally, these claims are the statements:
$\forall x\ \exists y\ (x \lhd y)$ = For ever x number there is a y so that $x \lhd y$.
$\forall x\ [x \neq 1 \leftrightarrow \exists y\ (y \lhd x)]$ = Every number not 1, has a previous but 1 doesn't.
But these two statements cannot be derived from the concrete cases.
Indeed, Logic only allows $\exists$ inferences from concrete cases and these are $\forall$.
Okay, so we can then accept these statements as additional axioms.
Will then these, with the infinite concrete ones, guarantee that only 1 , 2 , 3 , . . . can satisfy all of our axioms? Unfortunately not! Indeed, we can have alternative consecutives or previous elements, that are not from our natural numbers. Of course to assume for example an alternative $\underline{4}$ number, that also follows 3 , seems to imply that we need infinite many alternative numbers because of our first axiom. So the full totality of objects would be:

$$\begin{array}{ccccccccccccccccccccc}
1 & \lhd & 2 & \lhd & 3 & \lhd & 4 & \lhd & 5 & \lhd & 6 & \lhd & 7 & \lhd & 8 & \lhd & 9 & \lhd & 10 & \lhd & \ldots \\
 & & & & \Delta & & & & & & & & & & & & & & & & \\
 & & & & \underline{4} & \lhd & \underline{5} & \lhd & \underline{6} & \lhd & \underline{7} & \lhd & \underline{8} & \lhd & \underline{9} & \lhd & \underline{10} & \lhd & \ldots & &
\end{array}$$

But we don't have to have infinite many new numbers, because the alternative numbers can join back to the normal:

$$\begin{array}{ccccccccccccccccccccc}
1 & \lhd & 2 & \lhd & 3 & \lhd & 4 & \lhd & 5 & \lhd & 6 & \lhd & 7 & \lhd & 8 & \lhd & 9 & \lhd & 10 & \lhd & \ldots \\
 & & & & \Delta & & & & & & & & \nabla & & & & & & & & \\
 & & & & \underline{4} & \lhd & \underline{5} & \lhd & \underline{6} & \lhd & \underline{7} & \lhd & \underline{8} & & & & & & & &
\end{array}$$

To avoid such alternative naturals is easy, by sharpening our previous two claims from the existence of next and previous numbers to unique existence.
This means simply adding as new axioms the uniqueness for both:
$\forall x\ \forall y\ \forall y'\ [(\ x \lhd y\ ,\ x \lhd y'\ ) \rightarrow (\ y = y'\ )]$ and
$\forall x\ \forall y\ \forall y'\ [(\ y \lhd x\ ,\ y' \lhd x\ ) \rightarrow (\ y = y'\ )]$
Now we might feel convinced that we succeeded, by visualizing that 1 is the unique object without previous, and so step by step, 2 , 3 , . . . are unique too and can't have alternative previous either. In fact, we might say that the uniqueness is sufficient for the first axiom, because a potential alternative previous would have again one, then again, and so on, so would have to lead to an alternative 1, which is impossible, because 1 is unique, by the second axiom already.
This was a faulty argument though, because the alternative backwards elements can go on forever, without an alternative 1:

$$\begin{array}{ccccccccccccccccccccccccc}
 & & & & & & 1 & \lhd & 2 & \lhd & 3 & \lhd & 4 & \lhd & 5 & \lhd & 6 & \lhd & 7 & \lhd & 8 & \lhd & 9 & \lhd & 10 \ \lhd \ldots \\
 & & & & & & & & & & & & & & & & & & \nabla & & & & & & \\
\ldots & \lhd & \underline{-2} & \lhd & \underline{-1} & \lhd & \underline{0} & \lhd & \underline{1} & \lhd & \underline{2} & \lhd & \underline{3} & \lhd & \underline{4} & \lhd & \underline{5} & \lhd & \underline{6} & & & & & &
\end{array}$$

I had to use here the 0 and negatives too for alternative elements.
So we need the fourth axiom and then such infinite back branching is impossible. But now comes the real surprise. The backwards infiniteness is a deeper devil than we

thought. Namely, we can add a double infinite sequence of objects to our names, without relating to our names at all and the axioms still remain valid:

$$1 \lhd 2 \lhd 3 \lhd 4 \lhd 5 \lhd 6 \lhd 7 \lhd 8 \lhd 9 \lhd 10 \lhd \ldots$$

$$\ldots \lhd \underline{-3} \lhd \underline{-2} \lhd \underline{-1} \lhd \underline{0} \lhd \underline{1} \lhd \underline{2} \lhd \underline{3} \lhd \underline{4} \lhd \underline{5} \lhd \ldots$$

We could even add more similar alternative and unrelating sets too.
This example clearly shows that axioms about objects cannot completely describe what we see about the totality of the objects as a set. Simply because the statements are “inside” facts, and cannot grasp the real totality looked from the outside.
These sets of objects that satisfy some axioms are called models. They are the crucial connectors of Set Theory to Logic. This is so, because of the following simple fact:
If the axioms are true in a model, and Logic can derive a statement from the axioms, then that statement is also true in the model. In short, the logical inferences are all valid, in all models. The reverse of this claim would be that those statements that follow in models, are also logically derivable. But here, we must be more specific. Being true in one model, with the axioms is not enough, exactly because the weird models. But, if in <u>all</u> the models that obey some axioms, we have a statement that is always true, then this statement must follow logically from the axioms too. This is the best we can expect from Logic and so, this being true would mean the completeness of the logical inferences, or in short, the completeness of our new Logic. The theorem that this is true, thus had been called The Completeness Theorem, and Gödel proved it and presented it in 1930 at a congress, where Hilbert, the creator of the logical inferences, was present too. The proof is for the negative form, that is for the fact, that if a statement is not logical consequence of some axioms, then it doesn’t have to be true in all models. Gödel’s constructive proof actually made a model where the axioms are true, but the statement is false. So this proof showed a step more, how weird models can exist and are creatable, even wishfully. Hilbert wasn’t quite following the strange proof and less so, the whispers in the intermissions, about Gödel’s new, even more groundbreaking results concerning axiom systems.
Observe, that in the Completeness Theorem, Gödel created weird models with potentially alternative truths. Indeed, through his method in proving that the Logic Hilbert finalized is complete, he was the devil’s advocate because he assumed that something is not derivable from some axioms. This includes obviously the opposites of the derivable theorems but also allows statements where neither it or its opposite is theorem. For such undecidable statements the model creation would give two alternative models. Alternative, not just being visually different from the outside, but having different truth inside. But this stronger possibility was not pursued. He didn’t turn from being the devil’s advocate into becoming the devil and prove that for all reasonable axiom systems such truly alternative models can come about.
Hilbert knew about the weird models in general, but believed that perfect axiom systems are still possible in the original sense, that is deriving all statements, that are true in our names as model. Then, having weird models, that is extra weird elements beside, the real natural numbers, wouldn’t mean alternative truths. They would be merely strange representations of the universal truths. Gödel’s proof of the completeness theorem, left this possibility open, though made it seem very unlikely. Obviously he tried to pursue the logical road, that is construct alternative models with alternative truths, but this road is too difficult. Only decades latter was successfully carried out. So Gödel’s new method for proving undecidable statements, that is the

incompleteness of axiom systems was totally different. Most importantly it wasn't clear yet, even to himself, what the real cause of incompleteness was.
Observe that our human intelligence is based on the Isomorphism Principle. We are thinking in models. A child who can play chess, can play at once on all different boards and pieces, regardless of the size of the board or the particular features of the pieces. The different chess sets are models of a same system of rules, and we grasp this apriori. The weird models show that in math, the isomorphism is not sustainable. So then, as a second level, could be the "isomorphism" of truths. Meaning, that exactly chosen axioms can describe everything, at least logically, that is, decide all statements. As I said, Gödel already in 1930 knew that this is not true, in fact, addition and multiplication of the naturals cannot be axiomatized perfectly.

3. Gödel Numbering

A year earlier, it was proved by Presburger, that addition alone can be axiomatized perfectly. So what makes the multiplication such a big step? Well, we know that primes and all the interesting facts of the naturals need multiplication, but the crucial leap is the following fact: There are two $\alpha(c)$ and $\beta(c, i)$ "decoder" functions, expressible from multiplication, so that :
For every $(x_1 \ldots x_n)$ tuple of numbers, $\alpha(c)$ gives the n number of elements in the tuple and $\beta(c, i)$ gives the i–th element x_i. Obviously, the c code must be different for different tuples and we could start with finding a coder. But the real big claim is that there are decoders too. Indeed, this way, to claim certain tuples, that is sequences, we can simply say that there is a c so that $\alpha(c)$ and $\beta(c, i)$ obey certain rules. Exponentiation for example is merely a sequence that starts with x and has members always multiplied by x. Clearly with y members, we get as last member x^y. Thus $x^y = z$ simply means that there is a c code that :
$\alpha(c) = y$, $\beta(c, 1) = x$, $i < y \rightarrow \beta(c, i{+}1) = x \bullet \beta(c, i)$, $\beta(c, y) = z$.
So, exponentiation is an explicitly expressible relation from multiplication. We don't need the Peano rules for it and most importantly we don't have to introduce a new basic relation symbol for it. The same goes for other new relations that could be introduced by rule systems. They are already all in the arithmetic of multiplication as explicit formulas.
But the most important use of the tuple codings was the famous Gödel numbering of the formulas of arithmetic itself. This allows the talking about numbers to be in fact, a talk about the explicit relations, properties and statements of arithmetic itself. This self reference in the end, allows to say undecidable statements. Just as when we say, "I'm lying now", is undecidable, because if it's true then it's false, and vice versa, so the theory of numbers will have similar undecidable statements.
Gödel deeply believed that the language is the main reason of this imperfection or incompleteness of the axiom systems. The fact that not only Logic creates the derivations by finite rules, but that the axioms as a set has to be ruled too, was not emphasized, because it naturally melted into the Gödel numberings.
Gödel used an infinite set of axioms besides the name axioms: $1 \lhd 2 \lhd 3 \lhd \ldots$
Later, Robinson showed that the four axioms of $\lhd$, that we used to separate the real naturals from the weird elements, plus the four Peano rules of addition and multiplication added to the name axioms, is a system that just as well allows the tuple codings and Gödel numberings. I already mentioned that the "mystery" of the Peano rules is that they determine the relations but don't imply logically all the features that come about through the cases. I even mentioned as a simple case, the exchangeability

of the addition and multiplication order. A fact that every elementary school kid knows and yet it cant be derived in the Robinson arithmetic. So this system is obviously incomplete. But then what's the point of using it for Gödel numbering, when the whole aim of this numbering was to prove incompleteness?
The point is exactly the shift from language towards effectivity as the cause of incompleteness through an argument we soon explain. But first lets observe two obvious things! Firstly, a finite set of axioms is always effective and the case axioms are the basic effective set. Secondly, using non effectively collected axioms there is complete axiom system for arithmetics. Indeed, lets simply accept as axioms all the statements that are true in the $1 \lhd 2 \lhd 3 \lhd \ldots$ model, that is without looking at the weird possible extra elements. The problem with this seemingly perfect collection, is that we cannot effectively determine for every statement if it belongs or not, that is if it is true or not among the naturals. Namely if a statement contains $\forall$, we would have to check infinite many cases, to establish the truth of it.
The reason that this obvious line, from regarding such non effectively collected axioms as an unacceptable "solution" to Hilbert's program was not recognized as a source of effectivity itself, is quite simple. Namely a false oversimplification of effective with directly finite was accepted. The use of the successor function instead of the consecutiveness relation was a big help in this delusion too. The induction axioms that Gödel used, claim that if a formula is true for 1 and inherits from any n to $n+1$, then it is true for all numbers. This also includes potential variables in the formula. In a sense this is the universal version of the primitive 1 increment alterations of Peano. Since the possible formulas can effectively be formed, this system is effective too. In fact its effectivity was trivial and so hidden. The Robinson system avoids the induction axioms and with successor function, it doesn't need the name axioms either, so it is totally finite. Thus, it hides the necessary effectivity of the system even more. And yet strangely, as I said, the real goal of the Robinson system is exactly the demonstration of effectivity over language.
But lets return to 1931 when Gödel published his incompleteness result.
It took a few years to see that it's Effectivity that lurks behind Gödel's proof.
Gödel's belief in the cause as languages was correct, but that deeper cause is still not cleared, and his proof was not really using the deeper reasons. But it didn't use an argument either that would have lead directly to effectivity. Most amazingly, as the vision of Effectivity grew, the actual proofs did not change dramatically. The Gödel numbering remained the main method and the self referring contradictory statements as the actual punch line. Why? Well, this is the real mystery that still remains. But the better vision of Effectivity is undeniable, so lets see this fundamental simple picture:

4. The Effectivity Vision

Logic is merely a special system of rules to derive objects namely statements from the axioms. If the axioms are effectively derivable too, not with Logic rather by some formal rule, then this plus Logic, will derive an effective set, which is the derived theorems. Forget about meanings, even logic and negatives, and merely ask, what should be the set of the non derived statements like? In an even more visual way, forget about objects and regard them as places or positions in an infinite game space.
Using rules, the visited places or points are the generated or ruled or effective set.
But what should be the non visited places like? They are simply the leftover spots, not reached by rules, and so they don't have to be ruled at all. The leftover of a ruled set can and usually is much more complex than any ruled one. It is usually unrulable.

Only very special rules make the leftover be ruled too. If the rules are complex enough, the leftover is even more complex, not ruled at all. The self reference of arithmetics thus merely proves that the rules that generate the arithmetical theorems are complex enough, so that the non theorems are not derivable by any rule system.
But if the axiom system were complete, then a simple method for generating all non theorems would be the following: Generate all the theorems and then take the negative of each. By completeness, we would get every non theorem. Indeed, if an A is a non theorem, then $\neg A$ would have to be a theorem and so, A would be obtained by negating $\neg A$. So the formal negation in languages hides the deeper negation in sets as complements and thus, the effectivity argument too. The mentioned Robinson arithmetic now has a reason. It is obviously not complete. But not obviously something more fundamental is true about it. Namely the non theorems of it, that is the statements that are not derivable in it, can not be derived by any other method either. This then implies also the incompleteness by the heuristic argument we used above. But also a more detailed, second level of effectivity argument shows that this non generability of the non theorems inherits to extensions of this axiom system. Thus also to arithmetic with induction axioms. So the incompleteness of this "full" arithmetic, follows through the inheritance of the non generability of the non theorems. This way it's even clearer that effectivity is the direct source of incompleteness not language. But as I said, there is a dark side to this crystal clear effectivity vision, namely that the actual proofs go back to the self reference. At the second level, that is for the arguments about extensions, this is even more puzzling.
A better visualization of all the aboves, can be through the following abbreviations:
Let S be a system, with effectively given axioms. The theorems derivable by Logic is $[\,S\,]$. In the U universe of all possible statements, the complement of the $[\,S\,]$ set is $U - [\,S\,] = \neg\,[\,S\,]$, these are the non theorems.
Lets call the negative of a theorem an intentional non theorem and together the set of all intentional non theorems should be abbreviated as $[\,S\,]\neg$. Indeed this indicates that we collected $[\,S\,]$ and then negated all elements. Since $[\,S\,]$ is an effective set, this set of intentional non theorems is also effective. If S is an inconsistent system then everything is theorem and so also everything is intentional non theorem too. This sounds stupid but it doesn't matter because we aim for consistent systems. There, the intentional non theorems are all non theorems. That is $\neg\,[\,S\,] \supseteq [\,S\,]\neg$.
Completeness means that the only non theorems are the intentional ones, that is
$\neg\,[\,S\,] = [\,S\,]\neg$. Thus, completeness trivially means that the non theorems are effective. While non effective non theorems means that the non theorems can't be merely the intentional non theorems and so completeness is false. The first level of the effectivity incompleteness argument is showing that the non theorems are not effective. The second level takes a closer look at the intentional non theorems. So we have two effective sets $[\,S\,]$ and $[\,S\,]\neg$. They are disjoint. So, by other word, we could say they are separated. But this word, we will use for something much stronger.
Namely, by a "separator" we mean a machine or method that can separate all elements of a U universe. This in fact is merely two effective collectors L and R, that are complementary and are "put in one box". Indeed, if we know that the two collectors are exactly complementary, then we can simply collect both simultaneously and whichever comes up with an object we simply claim it to be left or right accordingly which method collected it. The assumed complementarity will guarantee that no object is collected in both and that all objects will be separated. The two collections can be called separable by themselves, because the other half is determined already. So separable set simply means an effective set with effective complement. But this can be generalized to two sets S and T, that are effective and

disjoint but together give not the full U universe. A direct separation of the full universe would be then into the three sets S , T and U – S – T. But less is enough. Namely, it's enough if we have two separate conditions of non collection, one for S and one for T. If these have common members, then they can be simply omitted from one of them. On the other hand the S and T sets can be included as non collection for the other. So we need again L and R complementary effective sets that contain the given S and T. In short, two S , T sets are separable, if they are effective, disjoint and contained in L , R complementary effective, that is separable sets. The second level of the effectivity incompleteness argument now is simply the claim that the theorems and intentional non theorems of the Robinson system are inseparable sets. Indeed a consistent extension of this system keeps these sets as part of the wider theorems and intentional non theorems. Then these must be inseparable too, otherwise the Robinson subsets were too. And this inseparability of the theorems and intentional non theorems means that the non theorems are not effective, otherwise the theorems and these would be a separator. So the first level applies at once. This inheritance of the inseparability also means that we can start even with only a partial demonstration of the inseparability for some special theorems and special intentional non theorems of the Robinson system.
Usually the word separable is only used in the more complex version for two sets, and the full separation that is a separable S and $\neg$ S complements are called decidable. But decidable is also used for a single statement meaning that it or its negative is theorem. From this it's also logical to call an $F(x_1 \;.\;.\;.\; x_n)$ formula decidable if for every name replacements of it the obtained statement is decidable.
The mentioned expressibility of rule systems as explicit formulas, hints that in the Robison system, every effective relation can be represented by F formulas. "Represented" meaning that the cases are derivable. But the crucial extra fact is, that effective relations with effective complements can be represented by an F formula so that the complement is represented by $\neg$ F. Thus, these cases are special theorems and intentional non theorems. Using these instead of all, plus with the dual lingo of decidability, then we can tell exactly, why the Robinson system's consistent and effective extensions must all be incomplete. Simply, because in the Robinson system: Every decidable relation is representable by a decidable formula.

5. Recursive Functions

Lets return to the Peano rules. A lot of hidden treasure was left unexplored.
As a simplest exploration lets try to continue the operations beyond exponentiation.
I will use the clear relational method and so we define an R (x , y , z) abbreviated as $x \uparrow y = z$. The meaning of course should be a repetition of exponentiation, just as addition was repetition of the consecutiveness, that is counting and multiplication a repetition of addition and exponentiation a repetition of multiplication.

So $x \uparrow y = z$ simply means: $\left((x^x)^{\cdot^{\cdot}} \right)^x$. The Peano style rules then are obvious:

At y = 1 the z value is the same as x and if $\bar{y}$ is increased by 1, then the $\bar{z}$ value is exponentiated to x : So, $x \uparrow 1 = x$ and

$$(x \uparrow \bar{y} = \bar{z} \;,\; \bar{y} \lhd y \;,\; \bar{z}^{\,x} = z) \rightarrow (x \uparrow y = z)$$

Now it's obvious that we can continue to form such operations or rather relations.

We might even think that this shows how an effective framework outgrows itself because the infinity of these operations is one single effective system, yet not created by the framework. But surprisingly we are wrong! The Peano method can grasp the totality of the infinite many operations. All we need is a new parameter or variable for the level of operation. So we regard an $R\ (x, n, y, z)$ relation that could be also abbreviated as $x\,[n]\,y = z$.
For $n = 1$ it is the addition, for $n = 2$ it is the multiplication, for $n = 3$ it is the exponentiation, for $n = 4$, the just introduced $\uparrow$. But it goes beyond these too.
The first rule is that for $n = 1$ and $y = 1$ the consecutiveness implies our relation.
The second rule is that for $n \neq 1$ and $y = 1$ the z value is always x.
The third rule tells the incrementing but now, not just using earlier $\bar{y}$ but $\bar{n}$ too:

$(x \lhd z) \rightarrow (x\ [1]\ 1 = z)$

$(m \lhd n) \rightarrow (x\ [n]\ 1 = x)$

$(x\ [n]\ \bar{y} = \bar{z}\ ,\ \bar{y} \lhd y\ ,\ \bar{n} \lhd n\ ,\ \bar{z}\ [\bar{n}]\ x = z) \rightarrow (x\ [n]\ y = z)$

This double incrementing was examined way before the connection of effectivity to Logic was considered. It was even proved that the z function obtained this way grows much faster than any single or primitively incremented function could.
All this gave even more push towards regarding functions instead of relations.
The obvious advantage of functions is of course that we can easily combine them by putting into an x_i another g function: $y = f(x_1 \ldots g(\ldots) \ldots x_n)$.
This feature was extended into allowing primitive Peano style definitions, that is value changes for every singular 1 increment. Indeed this can be regarded as a more elaborate way of combinings. Namely we give the function $f(x_1 \ldots x_i \ldots x_n)$ at the $x_i = 1$ value as an other g function of the other variables and then also we give $f(x_1 \ldots Sx_i \ldots x_n)$ as a function combined from f and other functions.
Lets start from the S successor function and two trivial type of functions:
$f(x_1 \ldots x_i \ldots x_n) = x_i$ and $f(x_1 \ldots x_i \ldots x_n) = \text{fix } n$
Using the two rules of combining and primitive Peano inductions, the obtainable functions are called primitive recursive. It was known that these can not grasp the double simultaneous inductions, like the universal operation we defined above. Yet, instead of returning to the wider multiple inductions allowed by Peano style systems, a new third principle was added to the combining and single induction. Why?
Formalism always buries the unsuccessful roads even if they are the most logical subjectively. Thus of course Formalism not only avoids the road to understanding but also blocks deeper questions that resurface latter anyway.
Here the mystery is that allowing multiple intermingled inductions, though seems like a huge generalization, is still not grasping itself. This was never proved or even asked why. The road was simply abandoned. To stress even more how fundamental this forgotten road was, lets observe that Logic swallowed the Peano style rule systems. So we could reformulate the universal Peano road as regarding any set of axioms for any basic relation symbols and then regard as effective relations simply the derivable name cases of the basic relations. A beautiful simple definition that directly relates to Logic. So what's wrong with it? Well exactly the fact that it uses Logic and thus hides something that is already hidden in the logical derivations.
But just above, we explained a different road that used Logic to replace effectivity. Namely instead of using blank basic relations with arbitrary axioms to define the

particular effective relations, we accepted addition and multiplication only, with the minimal that is Robinson axioms and then regard the explicit formulas as the effective. The derivable cases then mean the effective collections. The assumption that this very restricted method could grasp all effective collections is wild, so it can't be regarded as a reasonable definition of effectivity. And yet the mentioned decidable representations show that this "weak" system is even better than merely being able to replace all effective collections. It is doing that with the extra bonus, that the complementing effective relations are represented by also "complementing", that is negated formulas. Of course to even claim these, means that we must have other, wider definition of effectivity. But what is the purpose of this, strictly for results concerning axiom systems? Why should we define something outside, then prove that it can be represented by the system and then use that outside concept for getting result about the system? The concrete form of this external usage is actually a second representation. Namely that the derivations of the system are effective in the wider sense too. So what we really have is an externalization of the Gödel numberings. A fully internal way is simply relate the derivations to particular derivabilities about numbers. This is a complicated and seemingly aimless process with a "happy ending". By dragging the numberings out to the field of Effectivity, it all makes sense. Plus the proofs become easier too. This strange externalization has two additional weirdnesses. The first is that there is no absolute definition of Effectivity, so we have many possible externalizations, which would suggest very different proofs. But the second weirdness is what I already said, that there is a certain dullness in these effectivity arguments, all ending in the same self references of particular statements only.
Of course there is a position that regards Effectivity as the main concern in itself. Then investigating the relation to axiom systems is merely an application. With this view we have to understand what makes the effective methods all the same. We can't prove that they are all the same since we only have examples for it. Some seem very narrow and yet can achieve all known others, while some seem very wide and yet miss something. So we should see behind these illusions and pinpoint the real essentials. It's clear that Logic as a derivation system is rich enough, from the facts I mentioned about the Robinson system. And yet the abstract derivation systems are not rich enough directly. The present false wisdom is that the third principle of the recursive functions grasped this hidden richness of Logic.
First of all, this richness is not hidden at all, because we know that our logic is indirect. So we don't simply derive theorems as a chain, going from obtained ones to new ones. Rather we take a statement, assume it's false and then examine all consequences of this and if it leads to contradiction we go back and claim the statement to be derived too. The third principle of recursive functions grasps this "derivation by examination" feature in an elementary form. Still, what I said about the resurfacing of the buried concepts is true. So accepting the third principle will leave the question whether we really grasped the universal rule systems. This then should be faced, but again as usual, only the original formalizer faces these questions in some roundabout way and then the epigones and "polishers" of the field will do the proper burial. Here the original formalizer is Kleene and his Recursion Theorem is exactly this roundabout confrontation of the abandoned rule systems.
So what is this third principle, the "magic bullet" of effectivity?
It is best approached from relations:
For an $R(x_1 \ . \ . \ . \ x_n)$ we could denote as $\mu(x_i)R$ a new relation, that still keeps its x_i variable, in fact it's is true at x_i if R is true too but R is not true for any smaller numbers than x_i. So: $\mu(x_i)R =$

$R(x_1 \ldots x_i \ldots x_n)$, $\neg R(x_1 \ldots 1 \ldots x_n)$, . . . , $\neg R(x_1 \ldots x_i - 1 \ldots x_n)$
Either there isn't such x_i value at given other variable values because there is no true x_i value at all, or if there is then there can only be one minimal. This condition of existence, that is $\exists\ x_i\ R$, is effective because it merely requires a lucky find and then can be verified if R was effective. So we might jump to the conclusion that the uniqueness of $\mu(x_i)\ R$ gives an easy way to define functions from relations. We might even use $\mu\ x_i R$ for this new function which is defined at only those other non x_i variable values, that guarantee existing x_i value, and picks up as value, the minimal x_i. So this $\mu\ x_i R$ destroys the x_i variable, besides turning the relation into function. But not a real function only a "partial", defined at those other variable values where there are existing x_i values to make R true.
This new "partial function" is then a generalization of the concept of relation. Instead of yes or no, for being in a relation, a partial $f(x_1 \ldots x_m)$ has possible values or nothing for each tuple of variable values. The real domain of course, where value exists, is thus not all tuples only a subset and it splinters into different tuple sets by the different values of f.
The reverse, that is to make relations from functions of course is easy by claiming equations or merely that a function has a certain value. So then the primitive recursive functions can be used to claim such relations and then as the new third operation we can form partial functions. Then among these partial functions we could again make equations, then use μ again form new relations and so on! We can do all that, but we won't get effective partial functions or relations! The reason is simple. To verify the minimality of an x_i value in a relation, we have to verify all the smaller values to be in the negative of the relation. But the negative of course doesn't have to be effective!
The only compromise we can do is that every time we form relations from functions, we check if the functions are defined under the minimal variable values, that we want to apply. At a first application of $\mu\ x_i R$ from primitive recursively formed R, we are okay, because the values under the minimal x_i can be all verified to fail R. This is so because the primitive recursive functions are still total real functions, not partial.
This suggests an other idea of using the $\mu\ x_i R$ operations for only relations with all appearing functions being total. Unfortunately already the first usage from primitive recursive functions leads to unpredictably partial functions. The other restriction of checking the smaller values for each function is at least an effectively verifiable one.
The effective total functions are hidden among the partials!
The total effective functions cannot be derived by fix rules either!
This sounds contradictory! After all they are derived among the partial functions, by the three rules, compositions, single inductions and μ applications. What we really mean is that they can not be derived by themselves, which is the same as not being recognizable by a single method among the partials.
Total functions correspond to effective sets with effective complements. Indeed the truth can be regarded as a fixed value say 1 and then all other values mean false that is not belonging to the collection. But these can be still recognized by the other value.
The non effective complement problem thus corresponds to partial functions that can not be totalized. This includes that the domain of such truly partial function has a complement that is not domain of any other.
Now the obvious question of any sane person would be the following:
What the hell is going on? Why did we go from collecting tuple sets that is relations, to first functions and then to these partial functions, which are relations in disguise?

Is there an objective meaning behind all this? Yes, there is a bigger picture, why functions in general seem to take over relations. I already mentioned the most elementary form of this as the usage of the successor function instead of the consecutiveness relation. This continues in Logic, and in Set Theory becomes even weirder. Functions are merely relations with some $x_1 , . . . , x_m$ variables having all possible values, while the remaining $y_1 , . . . , y_n$ variables having unique values for every picked x combination tuple. So these are very special relations. How can they take over all others? This is due to verifiability, a human intervention into the envisioned general state of affairs. Since this clash of subjectivity versus objectivity is not understood yet, we are faced with the artificial narrowing of our visions to get results. So the use of functions is a purely economical step. An end justifying the means kind of practice. It becomes a full blown lie only if this itself is denied.
Formalism does exactly this! The judgment of someone being a Formalist or not is a much harder problem. It involves the whole private life not just the professional stand. The four smartest people in history, Newton, Gauss, Einstein, Gödel represent a very similar stand, total Formalism in life and negative in science. In short, they didn't say what they should have! To our very topic of course belongs the fact that Gödel was involved with the aimed crystallization of Effectivity. Obviously, this started after the reactions to his Incompleteness Theorem. So in a sense, he gave up the earlier preoccupation with language. The "functionalism" for him was forced through the simple idea that we used above, namely functions can form equations, that is relations. But he regarded this on its own. How far can equations about functions determine the functions? This is exactly like derivation systems with the functional twist. This twist is useful because we can substitute function values into variables. But he was not blinded by this. He knew that it is merely a road that might be wide but still artificial.
Kleene proved that the Gödel definable functions can all be obtained by μ alone.
The method was a Gödel numbering of the effective partial functions. Of course a Gödel numbering is only useful with proper decodability. The crucial fact is that for every m variable number we need separate decoders, but these will be primitive recursive. A decoding actually means imitation. The first decoder should be a D relation, imitating the domain of any effective partial function $f(x_1 . . x_m)$, that is:
$f(x_1 . . x_m)$ defined $\leftrightarrow$ $D(i, x_1 . . x_m)$
So this D would be for every m number a fix $m + 1$ variable relation, so that using its first variable with some i number value and regarding all other variable values where D is true, that is tuples being elements of D as a set, would give the domain of an f partial function. This defines i to be a code or index for f. Other codes might give the same f but the important claim is that we can obtain all partial functions with some i values placed into D.
Since the effective relations can be regarded also as merely domains of partial functions, thus these D relations already Gödel number all effective relations. Why didn't just Kleene do that and forget about partial functions? It is revealed in a sec.
First lets realize that this D couldn't be primitive. Indeed then it would be decidable that is having effective complement and we expect domains that are not such.
D must be extended to a wider relation that might be primitive and in which we have to search the other variables to select the correct first $m + 1$ ones. So observe that such widening means not only more variables that is more columns in a tuple list, but also more lines, maybe even seemingly useless non coding combinations.
The simplest scenario would be to have just one additional variable, that is column added to D and the search being merely for existence. Then we wouldn't have wrong non coding tuples but we would have the good ones with different last variable values.

That's exactly what Kleene's T predicate achieves! It is primitive, has $m + 2$ variables and the existence in its last variable, that is the simple omission of this last column gives the D. Formally:
$f(x_1 \,..\, x_m)$ defined $\leftrightarrow\ \exists z\ T(i, x_1 \,..\, x_m, z)$
So the mentioned repetition of codes, which was merely a possibility, is now a more apparent repetition of even one code and tuple with different z values.
It feels paradoxical that by merely allowing these repetitions, the more complex D becomes a primitive T. This will become natural after we see the meaning of z.
Right now I want to show why the whole functional approach had to be used.
First of all, the last column not only turns the complex D into primitive T but it allows the calculation of the f function values. Namely from the minimal z values by a primitive U. So formally:
$f(x_1 \,..\, x_m) = U(\mu z\, T(c, x_1 \,..\, x_m, z))$
This at once implies that to obtain the partial functions can be limited to one usage of the μ operation. Also, the letter U standing for universal tries to emphasize that a single method can imitate all others. This is a false vision! The T is more the universal method by selecting the domain. But both T and U are dependant on the variable number and this is the crucial point. These decoders are splintering the effective relations into separate sets by their variable numbers. We didn't even emphasized but it's true that different indices can be used for the different variable numbers. But these different indices must relate because wider or narrower tuples relate too.
To relate the indices from $x_1 \,..\, x_m$ m-tuples to wider $x_1 \,..\, x_m, y_1 \,..\, y_n$ m+n tuples is the famous s-m-n theorem of Kleene. It is the basis of his other already mentioned result, the Recursion Theorem that hides the actual solution to Gödel's approach, the functional equation systems. A slightly modified version of it, the Fix Point Theorem on the other hand simply expresses how repetitive the indexings must be. Kleene's 1952 book was the first full exposition of the new math. Effectivity was the bulk of the subject but not ripe yet for an explanation without questioning the foundations. Kleene just tied up the loose ends and was blinded by his own results. The book became a "bible" with a continuously distorted even blinder application. Just like with the real bible, the new interpretations never dared to confront the "book", rather twisted it to support the new messages. Against the false prophetic interpretations of the bible lies the simple and recognizable fact that Jesus was not trying to form a church by scripture and was not envisioning a far future.
Kleene' incomplete vision of Effectivity is irrefutably proved by a simple result he missed, the Rice Theorem. When we explain it, it will be obvious that it is the most important result of mathematics next to the Wellordering Theorem. Both of these are important because their proofs are seemingly simple yet were hidden due to the blind spots that I already talked about. And coincidentally these are theorems that uncover the widest meanings. But both of these can be narrowed down to Formalist emptiness.
The above mentioned T and U decoding, the s-m-n theorem, the Recursion and Fix Point theorems are just one step away from Rice Theorem. You can't outsmart a better vision, a deeper understanding. And these are also the only worthwhile things to spread. But today we live in a regurgitating age. Understanding is not the condition neither the goal of communication.
But lets be concrete! The concrete lie is the involvement of functions in Effectivity.
Effectivity is a property of certain sets made of a universe that has some structure.
The simplest structure is the consecutiveness of the naturals and the simplest sets made from the naturals are subsets or sets of tuples. The restriction of same long tuples in a collection is not natural from view of Effectivity. To collect tuples with varying lengths is a road I started in "Towards a New Foundation of Effectivity".

The machine approach of Turing was revolutionary, because instead of singular objects it uses objects with finite inner structures. This replaces the given relation among them. The words in a dictionary have inner structures as being built up from the letters and their relations to each other are determined by the inner structure already. Tuples can be interpreted as words with space between them, that is as expressions. But the space can be looked as merely an other letter. So the whole tuple problem is avoided too.
Thus the machine approach doesn't need functions and doesn't even suggest them!!!
The machine as object alterator that is expression alterator, continually changes the text. In Turing's method letter by letter. The crucial idea of a finish or "halt" indeed can define a final transformed text, but this is not an equivalent of being a function. It can simulate a function, if we interpret the initial text as a tuple and the final as the function value. You can even follow this through to prove the equivalence with partial functions. If you do that and then use the results of Kleene from partial functions, for the machines, then this shows that not only the partial functions but the machines too are imperfect concepts. There is nothing wrong with even continuing this practice. But to claim that this is the new clear theory of Effectivity is a lie.
Then to drag in big names like Gödel to justify this nonsense, is worse than lie.
Finally to appeal to the computer generation as a recipient and validator of this new math, is a betrayal of both math and people.
The historical details reveal that what we have today as the theory of computability is a compromised, imperfect conceptual framework. Gödel was going in one direction, Church in an other and Turing blasted into the scene with something revolutionary.
When he first conceived his machine he had infinite decimals in mind. This infinite data line later came back with a new meaning. But first his machines had to go through a fundamental transformation. They had to comply with Gödel and Church. Not through them but by Turing himself. The applicability became a two faced or split personality magician. Ugly details and grand simplicities.
The fundamental T predicate of Kleene is the best example. More exactly the added crucial z variable in it. So lets see a few variant interpretations for it:
The widest concept of machines as data alterators means that they keep on working for ever. The stop or end or halt is an artificial pulling the plug. The flexibility of alterations already requires to make decisions according to the machine states and the altered data. These decisions affect the states and the altering itself. So the halt is actually a "meta operation", merely a first occurrence of a chosen combination of the data and machine. The fact that it can be reduced to a chosen state only, gave an even more false extra status. So with this view, we can hypothetically let the machine run always forever and the magical z variable is merely the step number in a machine alteration sequence. The meaning of T itself is simply the termination or halt. This itself can be still interpreted more generally as the condition of halt or the first such condition. In the first case $T(i, x_1 \,.\,.\, x_m, z)$ will be true at all those z values where the machine coded as i, with input $x_1 \,.\,.\, x_m$ is exactly at termination condition at the z-th step. In the second meaning T is only true at one z at most. Of course this second variation of T is obtainable from the first by $\mu(z)T$. Remember, $\mu(\,)$ keeps the variable and the relation and merely claims it to be the first as well. Using the function forming $\mu z\, T$ is much sneakier. If there is only one z value at most then it merely acts as the picker of this value and for the more general meaning it picks the proper one. If the i-th machine never reaches termination condition starting from $x_1 \,.\,.\, x_m$, then of course $\mu(z)T$ is not true at any z and $\mu z\, T$ has no function value. But T itself is not only effective but decidable that is its

negative is effective too. This heuristically follows from the fact that we can try out the i-th machine with $x_1 \;.\;.\; x_m$ starting data and let it "run" exactly z steps. Then it must be recognizable whether it is in termination condition or not. The first part, that is "running" the i-th machine was imagined as done by us who gave the i code and so seemed obvious too. Giving codes to machines is not difficult, after all they must be characterizable by finite data themselves. Finding the code for an object or finding the object for a code, is visualized as mere associating. But T being effective then means that itself has to be a machine or actually two, one for its truth and one for its falseness. The first problem with our hasty assumptions starts with the following realization: If a machine wants to do the trying out of an other machine up to a given step number z , then the only way this can be possible for all z if we can imitate the machine as such. The step number helped in claiming a definite decision but it doesn't help in a simulation at all. So then our T being a universal machine throws us back to doubts about even of its existence let alone its decidability or some kind of primitiveness. Instead of facing this whole uncertainty, we can jump ahead again heuristically as follows: Just as we indexed the machines, we could index all possible finite data alteration sequences. It's a bit tricky because even the initial start can be infinite many, then the two step alterations more then the three step again and so on. But as we see soon, infinites can be stretched easily so the naturals are enough to code all these beginning alterations. Now if z means these, instead of the steps only, then the claim about T is much less. Indeed, it doesn't have to "run" the i-th machine, merely verify that it would do what is included in z already. The termination itself is also easier because the last data in z is also included. Of course instead of the difficulty of simulation, now the double coding makes T hard. One for the machines and one for the finite alterations.
All these arguments were merely meant to make you dizzy and wonder whether the intuitive notions of Effectivity and machines are reliable or not.
The shift from sets and tuples to data altering machines is plausible and simple at one moment but shaky and unreliable at the next.

6. Tuple Lists

Now we again return to the Peano rules, but we won't deviate toward functions.
Remember, that the advantage of functions is that they can be placed into each other.
But in the end we saw that the widening of tuple sets, still remained vital in Kleene's T relation. But widening can be used right from the start! This not only can replace the substitution feature of functions but also the reliance on earlier derived relations.
Multiplication for example needs addition. But instead, we can combine these two into a six variable $R(x, y, z, u, v, w)$ relation. We can also become more uniform and use implications for the initializations too. This means a bit of over complication, using equalities. But this way, all the rules can have the same R consequence, so actually all the implications can be omitted. Thus only the rule assumptions must be given as a matrix. Also, R is abbreviated as $(\;.\;.\;.\;.\;)$:

$y = 1 \quad , \quad x \triangleleft z \quad , \quad v = 1 \quad , \quad u = w$

$(x , \bar{y} , \bar{z} , u , v , w) \quad , \quad \bar{y} \triangleleft y \quad , \quad \bar{z} \triangleleft z \quad ,$

$(x , y , z , u , \bar{v} , \bar{w}) \quad , \quad \bar{v} \triangleleft v \quad , \quad (\bar{w} , u , w , u , \bar{v} , \bar{w})$

The first rule initializes both addition and multiplication.

The second defines addition, keeping the multiplication part unchanged.
The third defines multiplication by using the first addition part too.
The first rule implies the case: (1 , 1 , 2 , 1 , 1 , 1) with $1 \lhd 2$ and u = w = 1.
This rule also implies (1 , 1 , 2 , 2 , 1 , 2) with u = w = 2.
Using still only the numbers 1 and 2 , the third rule implies (1 , 1 , 2 , 1 , 2 , 2) .
Namely, using the first derived (1 , 1 , 2 , 1 , 1 , 1) case in both assumptions.
That is, 1 , 1 , 2 used for both x , y , z and $\bar{w}$, u , w. Plus $1 \lhd 2$ as $\bar{v} \lhd v$.
These three are the only true cases of R, with numbers 1 and 2.
To use the number 3 means to use the first rule to obtain : (2 , 1 , 3 , 1 , 1 , 1) or
(2 , 1 , 3 , 1 , 1 , 1) or (2 , 1 , 3 , 2 , 1 , 2) or (2 , 1 , 3 , 3 , 1 , 3).
Now the second rule allows all these, with changed 1 , 2 start too.
Then the new other cases allowed by the third rule can again be found for sure.
This is so because the barred disappearing variables are always smaller values than the main ones. So we can simply try all combinations of all variables up to 3, and check if they obey all the three claims in the third rule. Those that do, will give tuples for the main variables, those that don’t will not.
This trial and error method can be applied to all the rules, instead of being smart, and can be continued to using bigger and bigger values.
As a consequence of this method, we can obtain the negative of R too, all the tuples that are not obtainable by the rules. Thus R is obviously a decidable relation.
To show something sweet, lets widen R with one more variable, c.
It stands for ”composite” and it thus means simply w when u and v are both not 1.
But I will include the extreme case of both being 1 too. Indeed the opposite of the composite numbers are the primes but 1 is usually excluded, to make the prime factorization unique. So normally 1 is neither composite nor prime, but we have to be strict here, so we regard it as composite. This makes the initialization also simpler, starting from 1 instead of 4 . So the first rule becomes:
$y = 1 \ , \ x \lhd z \ , \ v = 1 \ , \ u = w \ , \ c = 1$
Remember that now our relation is (x , y , z , u , v , w , c).
The second and third rules remain the same with extended c.
Plus we need a fourth rule to obtain the real composites by:
$(x , y , z , u , v , w , \bar{c}) \ , \ \bar{u} \lhd u \ , \ \bar{v} \lhd v \ , \ c = z$
So the new composite c has nothing to do with the old $\bar{c}$. But observe too, that our vital fact about the secondary variables remains true because $\bar{c} \leq c$.
The last seventh element of our tuples will be the same as the previous w only at composite values. But the c values will also have all the composite factors for every w, because the composite w-s will increase without the application of the fourth rule.
So the composites will be listed with lots of repetitions in the seventh column.
Obviously, already the c = 1 value will appear with all possible w values.
But the last column will contain only composite numbers including of course 1.
So the missing ones are exactly the primes. Thus, using our four rules in the negative, that is for checking the tuples that avoid them, we generate the prime numbers too.
The universal x [n] y = z operation can also be easily defined by our matrix method:

$n = 1 \ , \ y = 1 \ , \ x \lhd z$

$n \neq 1 \ , \ y = 1 \ , \ x = z$

$(x \ , \ n \ , \ \bar{y} \ , \ \bar{z} \ ) \ , \ \bar{y} \lhd y \ , \ \bar{n} \lhd n \ , \ (\bar{z} \ , \ \bar{n} \ , \ x \ , \ z)$

The derivable (x , n , y , z) tuples again can be generated increasingly and thus all non satisfying tuples can be generated at once too.
Under 2, that is using only 1, there is no possible true line so (1 , 1 , 1 , 1) is not generated. Using 2, the first line gives:
(1 , 1 , 1 , 2) only, while the second gives: (1 , 2 , 1 , 1) or (2 , 2 , 1 , 2) .
Putting these three into the two brackets of the third line in every possible ways, will evaluate the variables in them. When these two are not contradictory and the two other conditions in the line are true too, then we obtain acceptable variable values for x , n , y , z and thus a tuple (x , n , y , z) or x [n] y = z.
In the opposite cases we obtain no tuple or we get the tuples in the negative.
If the barred variables are not all under some non barred in their values, then this method becomes faulty. Indeed, then we can miss some main variable values that only come about with higher values used in some barred variables. The matrix still defines the tuples but then we have to wait arbitrary long to get even small valued tuples.
This general usage of the matrix goes back to the choice applications of rule systems.
It is still more unified now as repeated applications of matrix lines.
Every application means putting some earlier obtained tuples into the brackets of a line. This defines an evaluation of some of the variables. If they are consistent with each other and also with the other conditions of the line with some valuations of the rest of the variables, then we get a new tuple as the main variable values in their predetermined order. We need lucky earlier tuples and lucky values for the undefined variables too. Both include lucky finds for the main variables that we collect eventually in the tuples and for the barred variables, that are ignored later. If we know that the barred variables are all under some main ones then we can list our tuples increasingly and the complement missing tuples can be listed too. Obviously if a barred, merely existential but not collected variable can not be guaranteed to be under a main, then we can simply use a new main variable for it, that is widen our tuples. Then we get a decidable, increasing wider list, in which the earlier non decidable is merely the subset with ignoring the last column. Of course, we might have more such non lower existence conditions that have to be promoted to main. Still, all complex non decidable, merely derivable tuple lists are subsets of wider decidable ones.
These, decidable or increasingly listable tuple collections are wider than the primitive recursive ones, since they contain the multiple incrementing rules and for example the universal operations x [n] y = z which are non primitive recursive.
The real problem with this "nice" matrix verification method is that with decidable usage, the complement list is not defined positively by an other matrix. There is no negative of matrixes. The matrix itself has a perfect negative since the lines mean "or" scenarios and inside each line, that is scenario, the commas mean "and" conditions, but the barred low conditions can not be rescued, as far as I know. It's a petty, because a similar matrix notation of "and-s" and "or-s" can replace all successively quantized situations. Such situation matrix contains the basic relations or their negatives in the lines as scenarios, instead of the brackets, equalities and $\triangleleft$ in our "effective" matrixes. And this situation matrix picture is the true nature of proof theory instead of the implicative quantification rules of Hilbert.
A more successful approach of effectivity is what we already mentioned, to accept addition and multiplication as basic relations, and then use the language of Logic to build the rest of the effective relations.
This itself can have two roads and two goals in each.
The first road is to regard the truths among the naturals, the second the derivabilities from a system, usually the Robinson one.

The two goals are either all effectively collectable relations, that is tuple sets, or only the decidable, that is increasingly generable collections.
The Logical formulas can't be all regarded as effective in neither of the roads and neither of the goals because of the occurring $\forall$ quantors. To check the truth of this requires to check infinite many cases.
Using existences only, that is $\exists y_1 \ldots \exists y_n F(x_1, \ldots, x_m, y_1, \ldots, y_n)$ formulas is okay. In fact it corresponds to our matrix, method with the x-s being the main collected variables while the y-s as the barred ones. But are these enough to grasp all effective tuple collections? No! The μ operator is still not grasped!
First it seems that now, we can find the true general form μ .
Indeed it's a $\forall_{x_i}$ quantor with x_i being a bound and x_i itself to be replaced with a new z variable. But we don't claim all z values, only under the x_i variable.
This is effectively checkable for a concrete x_i value.
Then $\mu(x_i) A(x_1 \ldots x_i \ldots x_m) = A$ and $\forall_{x_i} z \neg A(x_i = z)$.
The big problem here is that $\neg$ leads out of the effectivity! Indeed, it turns other $\exists$ into new non bounded $\forall$. So to be successful we have to restrict all quantors, that is use only $\forall_{x_i}$ and $\exists_{x_i}$. But this way, we can only grasp relations that are not only effective but have effective complements because the negative of a bounded formula is the same kind. So, can we grasp at least all of these? No, again!
While the bounding of the universalities was a direct necessity, this consequential bounding of the existences is restricting the decidable effectivities.
Remember that among the partial recursive functions the total ones were hidden, meaning also that among the partial domains, the exactly split domains are hidden.
Here, we can't tell among the formulas that define effective relations, that is collect effective tuple sets, exactly which will have a collection that has a complement also collectable by a formula. If the bounded formulas were all the decidable then it would be such exact recognizability of the decidability.
The bounded quantifications are merely a subclass of decidable collections. In fact they correspond to the primitive recursive functions. Not surprisingly then, just as the primitive recursive functions are enough to obtain all partial ones with a single usage of μ , here too, the effective relations can all be obtained with a single usage of $\exists y$ and all other quantifications bounded.
Turning from the truth road towards the derivability, that is the Robinson system, we seem to have a contradiction with what I claimed earlier about all decidable relations being representable by decidable formulas in that system. But it's not a contradiction because we can't tell exactly the formulas that will represent the decidable relations.
In short we still can't recognize all the decidable relations.
So a bigger picture is emerging:
Most effective collections have non effective complement.
The rare ones that have, are the decidable collections.
These are not recognizable among the effectives.
So, while the effective collections are generable by their collection methods,
there can't be generable formal methods for all decidable collections either.
On the other hand, there are lot of methods that collect only decidable collections.
Thus, these narrower methods can't produce all decidable collections.
Yet from such narrower decidable collections, we can easily obtain all effective ones, with some crucial new operation that goes beyond decidability.

7. Self Reference Arguments

We start with the simplest version, without effectivity and also without the details of a Gödel numbering. Thus it is a conditional claim with fairly strong conditions:

T Let in an S axiom system the natural numbers be names and $\vdash$ mean derivability!
If all $P(z)$ properties of S can be indexed as $P_1 , P_2 , \ldots$ so that:
A fix $F(y, z)$ formula can imitate the derivabilities of P-s from their indexes.
That is, for every m, n naturals : $\vdash P_m(n) \leftrightarrow \vdash F(m, n)$
Then there is a k number which used in $F(z, z)$ will give an undecidable statement.

P Let the index of $\neg F(z, z)$ be k, that is $\neg F(z, z) = P_k(z)$.
By our assumption this is imitated by $F(k, z)$ too! So for every n natural:
$\vdash \neg F(n, n) \leftrightarrow \vdash P_k(n) \leftrightarrow \vdash F(k, n)$. At $n = k$ we have:
$\vdash \neg F(k, k) \leftrightarrow \vdash P_k(k) \leftrightarrow \vdash F(k, k)$ This is not a contradiction yet!
If both sides are true, then it is a contradiction in S. So if S is consistent, then both sides have to be false, which exactly means the undecidability of $F(k, k)$.

R As I mentioned, the Robinson system inherits its undecidability, due to the expressibility of all decidable relations in it by decidable formulas.
Now, we explore this stronger undecidability in abstract, without referring to relations, rather directly requiring that other systems' decidable properties are expressible.
But first we need the fairly obvious fact that decidable properties remain in extensions:

D $P(z)$ is a decidable property of $S, \vdash$ if for every n number $\vdash P(n)$ or $\vdash \neg P(n)$

T If $P(z)$ is decidable in $S, \vdash$ and $S', \vdash'$ is a consistent extension then:
$\vdash P(n) \leftrightarrow \vdash' P(n)$.

P $\rightarrow$ is obvious and $\neg \vdash P(n) \rightarrow \vdash \neg P(n) \rightarrow \vdash' \neg P(n) \rightarrow \neg \vdash' P(n)$.
Indeed, the first $\rightarrow$ is due to the decidability, the second to the extension and the third to the consistency of S'. The two ends in negative form give $\leftarrow$.

T If the same conditions hold for $F(y, z)$ in an $S, \vdash$ as in the first theorem plus:
For any $S', \vdash'$ system's decidable $D(z)$ property, there is a $P(z)$ decidable property in S, that gives exactly $D(z)$. That is, $\vdash' D(n) \leftrightarrow \vdash P(n)$
Then, in any consistent S' extension of S, $F(z, z)$ is not decidable.

P Suppose $F(z, z)$ were decidable in S'. Then $\neg F(z, z)$ is decidable too.
So it is represented in S, by a decidable $P_k(z)$ and so by the decidable $F(k, z)$ too.
Thus: $\vdash' \neg F(n, n) \leftrightarrow \vdash P_k(n) \leftrightarrow \vdash F(k, n) \leftrightarrow \vdash' F(k, n)$.
With $n = k$ we have $\vdash' \neg F(k, k) \leftrightarrow \vdash' F(k, k)$.
By the consistency of $\vdash'$, both sides must be false, which contradicts that $F(z, z)$ were decidable in S'. Observe that here the argument is very different!
Not only did k depend on the S' extension too, but it was merely an indirect hypothetical case. It didn't give an undecidable case of $F(z, z)$.

R As I mentioned, this self reference used in the first and this last theorem resembles the liar paradox. Indeed, $F(y, z)$ was able to claim something about itself.
There is a different, so called Berry's paradox as follows:
What is the smallest number, not definable by less than hundred letters in English?
This sentence is less then hundred letters, so it is defining a number exactly the way it claims to be impossible for that number.
Newer arguments were able to imitate this paradox and lead to a statement that is true among the naturals but not provable in an S system. If the natural numbers obey the S system, then this statement is undecidable because its negative is not provable either. Indeed if it were then it had to be true among the naturals due to the obeying, but its false due to the opposite being true.

D This argument also uses the derivabilities of some P property cases. But in a sense, these will be the total opposites of the previous ones. There we regarded all cases of most likely to be infinite many cases. Here we only want properties that are derivable for only one case. This simply means that $\vdash \forall y [P(y) \leftrightarrow y = n]$ or in short : $\vdash P = n$ so the P property defines the n number.
We also regard the lengths of the properties denoted as $| P |$. More exactly, we also assume that this being less than a value can be expressed in the language of S.
Luckily, we don't need these two expressibilities separately and thus an actual Gödel numbering, merely the existence of a property P that defines y and is shorter than x.
So let the formula that expresses this be $E(x, y) = \exists P \ (\vdash P = y \text{ and } | P | < x)$.
Then we can easily find the formula that defines the first y that doesn't satisfy this $E(x, y)$ with a fix x value as $F(x, y) = \forall z [z < y \rightarrow E(x, z)]$ and $\neg E(x, y)$.
This $F(x, y)$ is a function. It defines a unique y for every x because for any x length there are big enough numbers that can be defined under x length and there is a first among these.

T There are m , n numbers that $\forall y [F(m, y) \leftrightarrow y = n]$ is a true statement among the naturals, but not derivable in S.

P For better understanding, lets examine first, instead of $F(x, y)$ an $R(m, n)$ actual relation with the same meaning but using instead of $\vdash P = k$ the $P = k$ as actual truth among the naturals, that is properties that are true for only one number. Then:

$$
\begin{array}{rllll}
R(m, n): & \exists P & \text{that} & P = 1 & \text{and} \quad | P | < m \\
 & \exists P & \text{that} & P = 2 & \text{and} \quad | P | < m \\
 & . & & & \\
 & . & & & \\
 & . & & & \\
 & \exists P & \text{that} & P = n - 1 & \text{and} \quad | P | < m \\
\neg & \exists P & \text{that} & P = n & \text{and} \quad | P | < m
\end{array}
$$

It's quite plausible that R is not effective because the $P = k$ claims aren't either.
Apart from this, R is not explicitly expressible either. Indeed if it were an $F(x, y)$ that is exactly true as $R(m, n)$, then putting an m value into F , the $F(m, y)$ property always defines a fix n. For example at $m = 10$ it defines the $y = n$ so that:

There is under 10 symbols long definition of 1
There is under 10 symbols long definition of 2

.
.
.

There is under 10 symbols long definition of $n-1$ But
There is no under 10 symbols long definition of n .

With big enough m value instead of 10, this becomes contradictory because m becomes bigger than the length of $F(m, y)$, so it will define n under m length.
If instead of $P = k$ being true, we use as we started $\vdash P = k$, then this has two benefits:
First, it is now effective, so the corresponding R' is effective too.
Secondly, if as we assumed, this R' is explicit too as an $F(x, y)$, then this doesn't lead to contradiction, merely to the fact that $F(m, y)$ will determine a $y = n$ for big enough m so that this determination is not a derivable determination in the S system.
So $\forall y [F(m, y) \leftrightarrow y = n]$ is a true but not derivable statement of S.

R The first, simplest self reference argument needed the consistency of the S system.
This second one needed that the naturals obey S.
These are the two ends of the line! Indeed if a system is inconsistent then every statement is theorem so undecidability is impossible. Also, a model obeying S means that all statements of S are true in the model. The logical consequences follow in all models, so the derivable statements that is [S] are also true. But contradictory statements can't be both true in a model, so an inconsistent S can't have model.

The first argument's $F(y, z)$ was obtained by Gödel as a $\exists x\, G(x, y, z)$.
So $\vdash P_m(n) \leftrightarrow \vdash \exists x\, G(x, m, n)$.
The meaning of x is that Gödel sequenced not only the properties but the derivations too as $D_1, D_2, \ldots$ and so $G(k, m, n)$ means that D_k derives $P_m(n)$.
This is also abbreviated as $\vdash_{D_k} P_m(n)$.
The claimed equivalence is established by the six implications in the following line, starting and ending with $\vdash P_m(n)$ and having $\vdash \exists x\, G(x, m, n)$ in the middle:

$$\vdash P_m(n) \rightarrow \exists k \vdash_{D_k} P_m(n) \rightarrow \exists k \vdash G(k, m, n) \rightarrow \vdash \exists x\, G(x, m, n)$$

$$\rightarrow \exists k \neg \vdash \neg G(k, m, n) \rightarrow \exists k \vdash_{D_k} P_m(n) \rightarrow \vdash P_m(n)$$

The first and last implications are true by the definition of $\vdash_{D_k} P_n(n)$.
The third is a logical rule that name implies existence.
The fourth is assumed in general for any $P(x)$ as: $\vdash \exists x\, P(x) \rightarrow \exists k \neg \vdash \neg P(k)$
The second is defining G in one direction as : $\vdash_{D_k} P_m(n) \rightarrow \vdash G(k, m, n)$
The fifth is the reverse if stated negatively as: $\neg \vdash_{D_k} P_m(n) \rightarrow \vdash \neg G(k, m, n)$
So, these last two are the crucial and hard ones through a Gödel numbering.
This splintering of the equivalence into a G and $\neg$ G implication reminds us of the mentioned decidability representation in the Robinson system. Indeed this was the root of that much smarter method, discovered a bit later.

Also, the reason why only the use of the derivations allows a decision about G, is understandable from Kleene's T primitive relation. There, the z full computation sequence allowed the decidability, in fact primitiveness. Here, in an axiom system, the derivations are the full sequences of "computations".
The strangest is the claimed assumption for all properties. In negative form it is:
$\forall k \vdash \neg P(k) \rightarrow \neg \vdash \exists x\, P(x)$ and with using $\neg P$ as P, it is simply:
$\forall k \vdash P(k) \rightarrow \neg \vdash \exists x \neg P(x)$
If for every k we can derive P (k), then clearly these cases are also true among the naturals. So in their model, the $\forall x\, P(x)$ statement is true too. But that doesn't mean it's derivable. If the naturals are indeed a model then the following chain is true:
$\vdash \forall x\, P(x) \rightarrow \forall k \vdash P(k) \rightarrow \forall x\, P(x)$ is true among the naturals $\rightarrow$
$\exists x \neg P(x)$ is false among the naturals $\rightarrow \neg \vdash \exists x \neg P(x)$
In general, without the assumption of the naturals being a model the chain is:
$\vdash \forall x\, P(x) \rightarrow \forall x\, P(x)$ is true in every model $\rightarrow$
$\rightarrow \exists x \neg P(x)$ is false in every model $\rightarrow \neg \vdash \exists x \neg P(x)$
So our assumption is weaker than the assumption of the naturals being a model, but more than what is true in general. Obviously some P property is derivable for all k and so our assumption implies non derivable statements and thus consistency too.
The usual name of the assumption is thus ω-consistency, because it claimed non derivability from infinite many cases.
Rosser was able to avoid ω-consistency and use only simple consistency.
Before explaining that, I repeat the Gödel method with only two variable G (x , y).
Indeed, we only need the self referring y = z situation anyway.

D Let a theory contain the natural numbers, 1 , 2 , 3 , . . . among its names and let $P_1(y)$, $P_2(y)$, . . . be the properties, while D_1 , D_2 , . . . be the derivations.
$\vdash A$ denotes that A is a theorem, while $\vdash_{D_k} A$ that D_k is the derivation of A.

T Gödel: If there is a G (x , y) explicit formula, so that:

1.) $\vdash_{D_k} P_n(n) \rightarrow \vdash G(k, n)$

2.) $\neg \vdash_{D_k} P_n(n) \rightarrow \vdash \neg G(k , n)$

3.) $\vdash \neg G(1 , n)$, $\vdash \neg G(2 , n)$, ... $\rightarrow \neg \vdash \exists x\, G(x , n)$

and $\exists x\, G(x , y) = P_g(y)$ and $\neg P_g(y) = \neg \exists x\, G(x , y) = P_{\overline{g}}(y)$,

then $\exists x\, G(x , \overline{g})$ is an undecidable statement.

So, neither $\exists x\, G(x , \overline{g}) = P_g(\overline{g})$ nor $\neg \exists x\, G(x , \overline{g}) = P_{\overline{g}}(\overline{g})$ is theorem.

P By 1.) $\vdash P_n(n) \rightarrow \vdash_{D_k} P_n(n) \rightarrow \vdash G(k,n) \rightarrow \vdash \exists x\, G(x,n) = \vdash P_g(n)$

So with $n = \bar{g}$: $\vdash P_{\bar{g}}(\bar{g}) \rightarrow \vdash P_g(\bar{g})$

Since $P_g(\bar{g})$ is the negative of $P_{\bar{g}}(\bar{g})$, thus if the theory is consistent, then,

$P_{\bar{g}}(\bar{g}) = \neg \exists x\, G(x, \bar{g})$ can't be theorem. So:

$\neg \vdash_{D_1} P_{\bar{g}}(\bar{g}) \quad , \quad \neg \vdash_{D_2} P_{\bar{g}}(\bar{g}) \quad , \ . \ . \ .$

By 2.) for each: $\vdash \neg G(1, \bar{g})$, $\vdash \neg G(2, \bar{g})$, . . .

By 3.) : $\neg \vdash \exists x\, G(x, \bar{g})$, so $\exists x\, G(x, \bar{g})$ can't be theorem either.

T Rosser: With $G(x, y)$ of previous theorem but assuming only 1.) , 2.)
and instead of 3.) : There is an $R(x, y)$ explicit formula too, so that:

3.) $\vdash_{D_k} \neg P_n(n) \quad = \quad \vdash_{D_k} P_{\bar{n}}(n) \quad \rightarrow \quad \vdash R(k, n)$
4.) $\neg \vdash_{D_k} \neg P_n(n) \quad = \quad \neg \vdash_{D_k} P_{\bar{n}}(n) \quad \rightarrow \quad \vdash \neg R(k, n)$

and: $\exists x \{ G(x, y) \wedge \neg \exists z [z \leq x \wedge R(z, y)] \} = P_r(y)$ and $\neg P_r(y) = P_{\bar{r}}(y)$,
then $\exists x \{ G(x, \bar{r}) \wedge \neg \exists z [z \leq x \wedge R(z, \bar{r})] \} = P_r(\bar{r})$ is undecidable.

P By Gödel 1.) again: $\vdash P_n(n) \rightarrow \vdash_{D_k} P_n(n) \rightarrow \vdash G(k, n)$
By consistency: $\vdash P_n(n) \rightarrow$

$\neg \vdash_{D_1} \neg P_n(n) \quad , \quad \neg \vdash_{D_2} \neg P_n(n) \ , \ . \ . \ . \ . \ , \quad \neg \vdash_{D_k} \neg P_n(n) \ , \ . \ . \ .$

So by 4.) : $\underbrace{\vdash \neg R(1, n) \ , \ . \ . \ . \ . \ , \ \vdash \neg R(k, n)}_{\vdash \neg \exists z [z \leq k \ \wedge \ R(z, n)]} \ , \ . \ . \ .$

Thus with $\vdash G(k, n)$ we get $\vdash \exists\ x \{ \qquad \} = P_r(n)$

So with $n = \bar{r}$: $\vdash P_{\bar{r}}(\bar{r}) \rightarrow \vdash P_r(\bar{r})$ so again from consistency: $\neg \vdash P_{\bar{r}}(\bar{r})$

Now we show that $\neg \vdash P_r(\bar{r})$ too, so indeed, $P_r(\bar{r})$, $P_{\bar{r}}(\bar{r})$ are undecidable pair.

Suppose, $\vdash_{D_k} P_r(\bar{r})$ were! Then by 3.) applied to $n = \bar{r}$ $\vdash R(k, \bar{r})$ were too.

From this, by the sheer meaning of $\leq$ we get:

$\vdash \neg \exists x \{ k \leq x \wedge \neg \exists z [z \leq x \wedge R(z, \bar{r})] \}$

From $\neg \vdash P_{\bar{r}}(\bar{r})$ we also get $\neg \vdash_{D_k} P_{\bar{r}}(\bar{r})$ so by Gödel 2.) :

$$\underbrace{\vdash \neg G(1, \bar{r}), \ldots, \vdash \neg G(k, \bar{r})}_{\vdash \neg \exists x \{ x \leq k \wedge G(x, \bar{r}) \}}, \ldots$$

The last two $\vdash \neg \exists x \{ \qquad \}$ can be combined without k, so:

$\vdash \exists x \{ G(x, \bar{r}) \wedge \neg \exists z [\quad] \} = \vdash P_{\bar{r}}(\bar{r})$ contradicting that $\neg \vdash P_{\bar{r}}(\bar{r})$

R The Rosser formula avoided the infinite many cases of ω-consistency but did it achieve finiteness? This is a legitimate question, because we used twice a new arguable step, namely dotted conditions, with arbitrary many members.
In both cases it was:

$\vdash \neg P(1), \vdash \neg P(2), \ldots, \vdash \neg P(k) \quad \rightarrow \quad \vdash \neg \exists x \{ x \leq k \wedge P(x) \}$

Or with $\forall$ instead of $\exists$ $\quad \rightarrow \quad \vdash \forall x \{ \neg x \leq k \vee \neg P(x) \}$

That is, $\quad \rightarrow \quad \vdash \forall x \{ x \leq k \rightarrow \neg P(x) \}$

So using $\neg P(x) = P(x)$

$\vdash P(1), \vdash P(2), \ldots, \vdash P(k) \quad \rightarrow \quad \vdash \forall x \{ x \leq k \rightarrow P(x) \}$

This sounds really obvious and indeed can be obtained step by step from trivial assumptions:
First of all, $x \leq y \leftrightarrow x = y \vee x \leq y - 1$ Then,

$\forall x \{ x \leq k \rightarrow P(x) \} \leftrightarrow \forall x \{ (x = k \vee x \leq k - 1) \rightarrow P(x) \} \leftrightarrow$

$\forall x \{ [x = k \rightarrow P(x)] \wedge [x \leq k - 1 \rightarrow P(x)] \} \leftrightarrow$

$\forall x \{ P(k) \wedge [\quad] \} \leftrightarrow P(k) \wedge \forall x \{ x \leq k - 1 \rightarrow P(x) \}$

So using again the same for $\{ \quad \}$ repeatedly, we'll get:

$\forall x \{ x \leq k \rightarrow P(x) \} \leftrightarrow P(k) \wedge P(k-1) \wedge \ldots \wedge P(1)$

8. Collection arguments

The Gödel numbering was a tool to obtain self referring statements. But in the actual methods, we used properties that is formulas. Formulas collect objects. This can be by their truths or derivabilities. It can involve effectivity or not. But in the end thus the Gödel numbering will order objects namely numbers to certain collected objects like sets of numbers or sets of tuples. We also mentioned that Kleene used numbering of the partial functions to prove his T relation.
This feature of assigning objects to sets of objects is a much bigger picture. It is not only the origin of Set Theory but it is also behind its unhappy ending, the unsolvability of the Continuum Hypothesis.
The biggest unspoken truth is that Effectivity is related to this!
It's already an unspoken fact that Effectivity is not finished either!
It is a foggy uncleared field with artificially tied up loose ends!
The elementary version of a connection is not denied. It merely means that self reference is in the form of the so called "diagonalization". So lets start with this open mystery. To be even more dramatic I just want you to look at the following picture:

		1	2	3	4	5	6	7	8	9	10	11	12	. . .
S_1	=	Y	Y	N	Y	N	N	N	Y	Y	Y	N	Y	. . .
S_2	=	Y	N	N	Y	Y	Y	N	Y	Y	N	N	N	. . .
S_3	=	Y	Y	N	Y	Y	N	N	N	Y	N	Y	Y	. . .
	⋮													
S_+	=	Y	N	N	Y	Y	Y	N	Y	N	N	N	Y	. . .
S_-	=	N	Y	Y	N	N	N	Y	N	Y	Y	Y	N	. . .

The letters Y and N stand for yes and no. The first line of the naturals means that the yes and no-s under them stand for some kind of questions about the naturals. Namely the different lines are randomly chosen answers. These are continuing with dots, showing that they are infinite many. The indexed S symbols combine the whole line. What the indexing really involves is still not revealed by the picture.
The crucial last two S_+ and S_- lines are manufactured from the infinite table. It's not apparent at once how, but looking at it longer, it can be easily recognized. Indeed:
S_+ is the collection of all the diagonal answers, while S_- is the exact opposites, so it is the "anti diagonal" sequence. The "diagonal" method is actually using more this anti diagonal sequence though the other diagonal can be important too.
In the original Cantor method, the diagonal was unimportant.
In this original application the yes and no-s are merely telling if a natural is chosen to be in a set or not. So clearly every possible yes no sequence defines a subset of

the naturals. In short the S_m lines are possible subsets. The method of indexing, that is the order of the chosen subsets is unimportant. The crucial point of Cantor was that such sequenced list can't be complete! We can't list all possible subsets.
We can "clearly" visualize that subsets can be picked from the naturals and so we also assume that a collection of all these possible pickings is a set too. We can even imagine that some of these are assigned with a natural and so we can list these.
Nothing suggests by intuition that picking a sequence from the possible sequences of pickings would exhaust all of them. But also nothing seems to forbid such lucky full sequencing of all the sequences. It may sound as a self reference, but it's only using a concept for itself on a higher level. Sequences of sequences is not really self reference. Plus here, it is the totality that really is the issue, namely the impossibility.
The real meaning behind this, is the simple intuition that there are much more subsets in a set than elements. There are much more sets of naturals than naturals themselves. If all subsets could be assigned with different elements, then the subsets weren't really more, only subjectively.
A better visualization of all this is the following:
Let a , b , c , . . . be elements of a U universe. The sequencing here doesn't mean actual sequencability merely a beginning from the universe.
Similarly A , B , C , . . . are the subsets of U, again merely explaining the notation, not claiming sequencability.
Now lets use [ ] for a fix ordering of sets to elements. So [a] , [b] , [c] , . . . denote the sets obtained by [ ]. A simplest example of [ ] would order to every element the set of the single element, that is [s] = { s }. Using this ordering we only obtained the singular subsets of the U universe. So all the really important subsets of U are unindexed or unassigned to elements. Using the natural numbers as elements we can even better visualize this pathetic [ ] as :
[1] = { 1 } , [2] = { 2 } , [3] = { 3 } , . . .
Using the naturals is also useful to show how to "improve" this [ ] ordering.
Indeed, we can keep all these singular subsets assigned and yet "free up" elements to be used for other subsets, by merely using a subsequence of the naturals for these singular subsets. For example lets use the odd numbers only:
[1] = { 1 } , [3] = { 2 } , [5] = { 3 } , [7] = { 4 } , . . .
Then the even numbers can be used for other more important subsets.
This principle can be used again and again. So we can always stretch an infinite set to leave out elements and thus use those as new assignments or indexes. In fact we might regard this argument as a proof that eventually all infinites can be indexed by any infinite U starting universe. So all infinites are the same, merely infinite.
But this argument is false, as Cantor realized it first in history, by showing that all the subsets of the naturals can not be indexed. So the one by one stretchability as argument is subjective and faulty. Of course, only the precise rules of Logic will later capture the exact error in the subjective argument. This shows that the Set Theoretical arguments induced Logic too until the two fields finally combined in perfect unity.
But now, just to stick to Cantor's original argument, we merely have to regard the anti diagonal subset formed from a sequence of already assigned subsets.
This anti diagonal subset simply can't be among the sequence. Indeed, every line will have the diagonal yes or no altered. Thus if the diagonal natural was element, it becomes a non element or if it wasn't element then it becomes element. So these diagonal naturals will give an alteration from all lines!
Observe the irony that the faulty subjective one by one increasability, was refuted by a similar one by one example. Namely for every sequence we can give a subset missing. Clearly it doesn't mean that only one is missing. Most of the subsets must be missing!

But the argument is only producing one and that's enough because a full listing would be still just a listing and thus not full at all! So the self reference paradox is detectable in the reasoning that proves a solid, intuitively plausible fact too.
Most amazingly, the anti diagonal argument becomes a clear cut self reference argument if we generalize the same intuitive meaning from the naturals to any U set:
If a , b , c , . . . are some elements of a U set, then
[a] , [b] , . . . cannot contain all subsets of U. The proof is amazingly simple:
Lets observe that an s element is either <u>in</u> its own assignment [s] or not.
The collection of all those s that are <u>in</u> their own assignments is: $\{ s ; s \in [s] \}$.
This could be assigned to any t, that is $[t] = \{ s ; s \in [s] \}$.
If $t \in [t]$ then t is in the collection, so indeed, $t \in [t]$ again.
If $t \notin [t]$ then t is not satisfying $s \in [s]$, so $t \notin [t]$ again.
Now comes the big surprise! Lets look at the complement set of $\{ s ; s \in [s] \}$, that is:
$U - \{ s ; s \in [s] \} = \neg \{ s ; s \in [s] \}$. This contains all the elements that are not in their own assignments, that is $\{ s ; s \notin [s] \}$. This can't be assigned! Indeed, suppose $[t] = \{ s ; s \notin [s] \}$ were.
If $t \notin [t]$ then t satisfies $s \notin [s]$ and so $t \in [t]$.
If $t \in [t]$ then t doesn't satisfy $s \notin [s]$ and so $t \notin [t]$.
Both assumptions lead to its opposite, so the assumption of $[t] = \{ s ; s \notin [s] \}$ is contradictory already. So all the subsets of a U set cannot be assigned to elements.
As I said earlier, the new step by step missing subset, destroyed the naïve, step by step stretchability argument for all subsets to be assigned "eventually". And yet even this argument can be generalized correctly. But first lets see how we can stretch further.
For example the 1 , 2 , 3 , . . . sequence, into a whole sequence of sequences:

$$\begin{array}{ccccccc}
1 & 2 & 4 & 7 & 11 & 16 & \dots \\
 & 3 & 5 & 8 & 12 & 17 & \dots \\
 & & 6 & 9 & 13 & 18 & \dots \\
 & & & 10 & 14 & 19 & \dots \\
 & & & & 15 & 20 & \dots \\
 & & & & & 21 & \dots
\end{array}$$

I simply used longer and longer sub segments later and later, vertically and thus, split the single sequence of 1 , 2 , 3 , . . . into infinite many horizontally.
Now, we can put these after each other:

1 , 2 , 4 , 7 , 11 , 16 , . . . , 3 , 5 , 8 , 12 , 17 , . . . , 10 , 14 , 19 , . . .

In short, we got an infinitely multiplied stretch.
We might think that that's all folks. But, it's amazingly simple to go further.
Indeed, imagine that we do this same infinite replication, but only starting from the odd numbers. Then, all the evens are at our disposal to make new continuations.
These so called well orderings of a single S set can be always increased longened.
And yet, there has to be a full set of all possible stretches from the natural numbers.
Amazingly, the totality of these possible stretches cannot be assigned to the elements of the original set either. This follows from a similar, but even more abstract argument than the earlier $s \notin [s]$ choice.
Though stretches are obviously made from an original set, the stretches themselves, without specifying of what objects they are made from, continue each other too:

```
●
● ●
● ● ●
.
.
● ● ● . . . . . . .
● ● ● . . . . . . . ●
● ● ● . . . . . . . ● ●
● ● ● . . . . . . . ● ● ●
.
.
● ● ● . . . . . . . ● ● ●. . . . . .
.
.
● ● ● . . . . . . . ● ● ●. . . . . . . ● ● ●. . . . . .    . . . . . . . . . . . . .
.
.
.
```

Now assigning objects to these, at once defines a stretch from the objects, step by step. So if the set of all stretches were assignable to the objects from which they were made, then that would create a single stretch, made from the set of objects. Then of course, we could use only part of the set, like the odds or evens in it, and thus, use the rest of them to continue with longer stretches. But this would contradict the claim that we already collected all possible stretches.
There was a little nuance that we left foggy! We said, we listed all increasing stretches made from our set. But the "made from" can mean using all elements or merely using some elements. The above argument showing that the possible stretches are more than the set, because all the stretches can not be assigned to elements, was using the sub stretches too. In fact it was vital because the one by one increase of the stretches must start from a single element, to give an actual stretch for any assignment. If we had started with ● ● ● . . . . . . , that is the first unstretched use of all element of the naturals, then we couldn't conclude that all the stretches made using all naturals are more than the naturals. Not directly anyway, only using the substretches too and then observe that the proper substretches are clearly the same as the naturals themselves, so the proper full stretches must be more than the naturals. Proper substretch here means any substretch that is shorter than the minimal full stretch and thus must use only finite many naturals. This argument is less trivial but remains true in general:
At any S set, the proper substretches as set is similar exactly to the minimal full stretch. Indeed, the proper substretches are all shorter than the minimal full stretch by definition and so their total can't be longer than the minimal full. But the total cant be shorter either, otherwise it were itself a proper substretch obtained in the minimal full, and then we could obtain a new proper substretch contradicting that we collected all.
After this crucial minimal full stretch come most of the stretches, because they are more than the S set. They can't be indexed by S elements. In fact they must be the next infinity bigger than S because all smaller infinities are their beginnings and thus merely stretchings made from S.
Since as we saw, the subsets of S can't be indexed either by S, thus the obvious question is whether they are already this next infinity. If so, then all the subsets of an S could be indexed with stretches of S. This was Cantor's belief, the Continuum Hypothesis. But as it turned out, it is undecidable from the axioms of Set Theory.

The full assignment problems are a deep and unsolvable puzzle.
So the major method of incompleteness, the Gödel numbering, is directly relating to a most concrete unsolvability. Only a fool can't see that thus, effectivity is also part of a bigger puzzle. The fact that the different effective methods, like derivation systems, recursive functions, lambda calculus, verifiable collections, self altering finite systems, all give the same effective sets, show that there is something outside of these particular roads. This was called as Church thesis, referring to the existence of an objective essence of effectivity. For a while, it was emphasized that it is outside math. Later, falsely, the Turing machines were over valued as the simplest form, or computers in general as the widest form behind it. But the truth is that math itself is beyond Logic and axiom systems as we know them today. Set Theory was the big step in the self consciousness of math, so it will need the reform of Set Theory to go ahead. Effectivity is relating to Set Theory and directly to the Continuum Hypothesis.
The imperfection of Set Theory was clear from the start.
Indeed, the $\{x\ ;\ A\ (x)\ \}$ heuristic explicit collection principle failed through the Russell paradox. The same logic that showed how an assignment of all the S subsets to S elements is impossible, shows that collection of certain properties, without restricting them to S sets, is straight out contradictory. Indeed, sets were envisioned as gradually built from each other. A set can't be element of itself. So, $x \in x$ is never true or $x \notin x$ is true for all sets. Thus, collecting all these sets, would simply collect the whole universe of all sets. Accepting this universe as a set, has a problem in itself, because then we could build new sets from it and thus, not in it. But, this is merely a soft conceptual contradiction yet. Using $x \notin x$ as the particular exact $A\ (x)$ collecting property, however will lead to direct contradiction. Indeed
If $U = \{x\ ;\ x \notin x\ \}$ then again, both $U \in U$ and $U \notin U$ lead to contradiction:
$U \notin U \ \rightarrow\ U$ satisfies $x \notin x \ \rightarrow\ U \in \{x\ ;\ x \notin x\ \} = U$
$U \in U \ \rightarrow\ U$ doesn't satisfy $x \notin x \ \rightarrow\ U \notin \{x\ ;\ x \notin x\ \} = U$
This Russell paradox forced the splintering of the collection principle into several allowable collections. Seemingly, this went smoothly, that is a pretty obvious system cut out the contradictory collections. Formally, this corresponds to the induction axioms for arithmetic. Not in its meaning, rather in its spirit as something restricted by the language and yet, optimal in some sense.
The true corresponding axiom to induction was achieved in Set Theory by the above mentioned stretchings or well orderings of sets. This was exactified by using collection, plus a new axiom, the Axiom of Choice. Instead of picking new and new elements for the stretch types, we can pre-pick the next elements for all subsets, and then a start element will define a growth. This was a real success story, but didn't help in solving the mentioned full assignment problem.
The Axiom of Choice could also be called as random collection, because it picks elements without any rule or specification. But this "randomness" feature has still not been connected to the actual subject of random sequences. This in itself shows that something is still missing. Even more fundamentally, the Axiom of Choice is actually an extended axiom of Logic. Indeed, in Logic, as I mentioned before, we can use concrete cases to infer $\exists$ and from $\exists$, we can use hypothetical variables to play the roles of the claimed existing objects. The Axiom of Choice allows to use a whole hypothetical set that picks elements from claimed existing objects. This is the most beautiful axiom of mathematics, a perfect link between Logic and Set Theory.
The three types of collections, the explicit, that failed by the Russell paradox, the random, that became the Axiom of Choice and the effective, that again failed as the solution to effectivity, are somehow fragments of a missing bigger principle.

The oversimplified vision of the set comparisons is using the concept of equivalence instead of assignments. This is very natural for finite sets. If for example we want to know who are more in a ball room girls or boys, then all we have to ask is form pairs. Whichever is left without pairs is the more. Every new pairing will lead to the same result. But infinites are different! They can be stretched so they can be also equivalent to their subsets:
1 , 2 , 3 , 4 , 5 , ... is equivalent to 1 , 3 , 5 , 7 , ... or 2 , 4 , 6 , 8 , ...
So among infinites the demonstration that one can be ordered uniquely element by element to a subset of the other does not mean yet that the first is a bigger infinite. It merely shows that it is an at least as big infinite as the other. Indeed, a new one to one ordering might be perfect, not to a subset but to the whole set.
Among infinites to show that one is bigger, we have to demonstrate the usually trivial at least as big part and then as a crucial new part, we have to demonstrate the actual impossibility of a hypothetical full assignment. The first trivial part is usually not even mentioned. For example, it's obvious that the subsets are at least as big a set as the original U set because the singular subsets at once give an equivalence of the elements "into" a subset of subsets.
The crucial negative proof of impossible equivalence from the full set of subsets, then is sufficient for assignments, that is different but not unique orderings. The [ ] assignment of subsets to elements for example could be repetitive, that is giving same subsets for different elements. This would still make an equivalence possible if all subsets could be assigned. Indeed we could simply keep only one of the elements assigned to every subset and then all the subsets were equivalent to a subset of U.
Strangely in this innocent argument, we still used the Axiom of Choice too.
Even stranger is the fact that these possible "irrelevant" repetitions of the assignments, are not really an avoidable "nuance" in the real practice of orderings! I already mentioned this about the repetitions of Gödel numberings. Kleene's Fix point theorem expresses specifically this, but his other theorems contain it too. For indexing or coding, this can also be called a redundancy of codes. We see this in nature as different DNA triplets for same amino acids or in languages as alternative words. What never mentioned is that even the purely equivalence "tricks", that show that a seemingly bigger set is not really bigger, also carry this redundancy burden.
The simplest example is Cantor's classical method of showing that the fractions are merely the same infinity as the naturals. This is surprising since the fractions "cover" the whole line densely. And yet we can list them by increasing total of the numerator and denominator:

$$\frac{1}{1}, \underbrace{\frac{1}{2}, \frac{2}{1}}_{3}, \underbrace{\frac{1}{3}, \frac{2}{2}, \frac{3}{1}}_{4}, \underbrace{\frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1}}_{5}, \frac{1}{5}, \ldots$$

Now, after the big surprise of this simple method, we never stop and think about the hidden extra feature that not only we listed all fractions but we listed them with infinite repetitions. Indeed, the same valued expansions are all listed as different.
So the Set Theoretical equivalence arguments are related to Effectivity on a deeper level than we know it today. But it's time to turn to Effectivity directly.
So lets return to the first page of this section and regard the crucial yes/no table with a new meaning! The lines are now not merely possible answers to a collection, rather given by different Effective methods of collections. This raises a fundamental

question about our intentional meaning of Effectivity. Should it mean both yes and no answers or only yes. In my view, only yes! That's the meaning I used in this book!
To collect elements effectively we need only an exact method of verification case by case. The lucky finds of the cases don't have to be provided by a method, so the collected objects merely have to exist. But the collection is meant as total, that is gathering all verifiable existences. So in a sense these are "every" objects and indeed this quantor applies to them in the collection rules as axioms. The proper existences or secondary or hidden existences of the collections are lucky finds that also appear in the verifications but we don't collect them. We saw that collecting these too, the verification process is actually a decision process, so our effectivity becomes the much stronger yes/no decidability. In spite of all this, I stick to my view that Effectivity as an objective concept is the wider verifiable object collection, that doesn't have to relate to the decidability of the verification process. This leaves the door open to new verification methods, that are not directly decidable. With this vision, decidability is a mere coincidence. A collection, so that the uncollected object are also collectable. So a decider is merely two complementing recognizers.
The functional approaches and even computability are in contradiction to this view.
The idiotic expressions of "recursively enumerable" or "computably enumerable" are perfect examples of how the basic meaning of Effective becomes as secondary. Then these smart ass "concept twisters" admit it with a kind of smirk or internal wisdom, that actually these enumerabilities should be the real issue. Of course an admitted lie is still a lie. But lets return to our yes/no table! This also shows my point.
So suppose only the yes answers are provided by some methods! Then we can ask whether we have a system that gives all such methods. If so, then we can list these.
A certain new hidden decidability seems to lurk behind this again, like above behind the individual case by case verifications. But it's less dangerous here. Indeed, listing the effective methods as forms doesn't have to be verifiable to be proper. Every form is okay! It cant be contradictory. The worst that can happen is that it doesn't collect anything. Accepting the empty set, it is still a collection.
A whole new ball game is when we assign the indices to the listed collection methods! The expression of indexing or Gödel numbering, refers to something much stronger than actual indexing. We assume that from the listed order and thus from the indices, the actual collections can be obtained. The proof of this is through the existence of some universal method. This of course must have an extra variable, the index. So this $R(i, n)$ recognizer can give again only yes answers, but it gives exactly the right ones for every listed S_m, that is $S_m(n) =$ yes if and only if $R(m, n) =$ yes.
Recognizing the identity of two numbers is of course effective and so $R(n, n) =$ yes is an effective method. Thus it must be in our list of effective number recognizers.
This is exactly the diagonal S_+ and nothing forbids it to be in the list.
But S_- can't be there, so S_+ is an important one in our list because it definitely has a complement, namely S_- not in the list. So we have an example for what we claim, that most effective sets don't have effective complement.
Now, finally we can have a third meaning of the yes/no table:
We regard only decider methods that is having both yes and no be given by methods.
Such list can again be three kind. Theoretical, that is as pairings with naturals. Indexing by some forms. And finally indexing from which the yes/no answers are recreatable. In each of these three, we can aim for a full list, that is containing all deciders. In Cantor's original anti diagonal method a theoretical full list of all subsets was refuted by S_-. Above, for the effective collections, or recognizers, the full list was not refuted in any sense and only a non effective complement was produced.

Here, for the decidable sets or deciders, only a third kind of full list is refuted by S_- .
Of course non full list of deciders can be in all three meanings.
The mentioned fact of the apparent but false definability of Effectivity from decidability, also means that such non full decider classes are used only. But these approaches, then always progress to the real, independent listings of effectivities, or denumerabilities as they call it. Strangely then they never ask why this proper Gödel numbering is needed. Why not the indexing of the special deciders. For example, we can easily index all primitive recursive functions. Then all partial recursive functions can be obtained from primitive and so they could be indexed through those at once.
But this would be only a second class listing in our above sense. Formal but not recreatable. Indeed the $\exists y$ or μ operations are used externally after the numbering of the primitive recursive operations. We couldn't verify the obtainable cases from the indices. This hidden indexing problem only becomes apparent in Kleene's treatment because in Gödel's original version the derivations are used.

9. Program Arguments

The Gödel numbering as a needed tool, tends to rob the nice openness of effective collections and turn them into totally mechanical generations. Mathematical ingenuity is reduced in theory to derivability. So not surprisingly, a machine approach should not even regard free choices and instead be mechanical from the start. An other fact is, that the freedom in an effective collection also disappears. It is only the flexibility of the verifications that is essential. And yet, for machines too, to achieve these flexibilities, and thus full collectability, the freedom of choice still remains, only in a hidden form. This hidden form is that machines can't be object generators directly, they must be object alterators first. The input, the object we give to the machine is the freedom of choice. So, we go a level deeper than collection, and this level is alteration. Machines are object alterators! This implies two additional details:
One is that the object must have inner finite structure. This replaces the abstract atoms, $1, 2, 3, \ldots$ that only relate by $\lhd$. Now this $\lhd$ is unnecessary, it comes out from the inner structure. Simplest translation of $\lhd$ to inner structure could be using strokes like $| | | = 3$. The fact that we needed tuples of numbers to define even effective properties about single numbers, now translates to the fact that we need at least one other symbol for inner structures. So instead of a stroke plus something else, we usually start with the $0, 1$ binary digits to form strings. If we use more than two symbols, then on the other hand, these usually are regarded as an alphabet and the strings as words.
The other consequence of the alteration is that the machine that alters the finite object, the words or strings, is itself altering. It must have some internal working too.
These two consequences, then imply the crucial new feature of machines to become computers. The specific new feature is that the objects that are altered, can be used to influence the alteration of the machine itself.
Normal mechanical machines use the objects blindly. A lathe is merely carving its objects. Even early calculating machines were like this. The numerical data was operated by the machine goals. Mechanical calculators did addition and multiplication.
A first improvement could be to store the data in some containers and also a whole sequence of operations, that we have to perform. In this sequence, we would not enter the data manually, rather the machine could reach it from the containers. It could also change the data in these containers by the calculated results. An even more flexible version could be a machine that can be programmed to do different operation sequences.

The big idea of John Von Neumann and Alan Turing was that this sequence program could be regarded as data itself. Just as the data was obtained from addressed data containers, the program itself can have step numbers. Then, we could equip the machine with the ability to jump into any point in its program sequence. But the big idea is not merely to do such jump to a concretely given step number. Rather, to jump to a value that is located in a data container. This way, the machine can decide its own jumps according to some calculated values. This means continually changing jumps. This is a new level of flexibility. The computer is not merely an object altering machine, it is a self altering finite system through its data.
In computers, the input data replaces the freedom of choices of the derivation systems. But now, this input data can be directly connected to the operation of the machine too. Indeed, we can influence or program the machine through the data, determining the jumps. So, part of the input data is a program for the running of the machine. The other fix program of the machine itself, should not even be called program, rather just the machine rules.
The effective collections are now merely optional consequences of this new deeper level of programmable self alteration. The two basic methods are to collect inputs or outputs. This sounds obvious but it hides a crucial further artificial application of machine features. Indeed, the machine as data alterator only defines directly the input, as the initial data. But this can be anything so only the concept is defined not a particular set. The continuous data alterations would go on forever in theory. To stop or halt by some conditions of the altered data or the state of the machine is merely an external decision. Clearly we could imagine that we don't stop rather continue by some rules. Then the stop or end or halt condition may come about repeatedly. We might even think that this behavior of the machine how it repeats up to infinity in time is an important feature, needed to define collections. But not so! We only have to regard operations up to any first occurrence of conditions. These initial operations are already complex enough. The finite time behaviors give all Effectivities. In fact it's enough to regard machine states as conditions, that is initial operations up to a first machine state occurrence. We don't have to involve the altered data in defining the halt state. This seems contradictory to an envisioned machine goal but remember that we can program any data conditions to the halt state anyway.
This definition of halt, hides the crucial fact that even though a machine can have only finite many states, if we define all possible state changes, then some of these are not cyclic not even predictable. Namely a first appearance of some states is already not predictable. The reason for this is that the altered data is involved in the state alterations too. So though the states are directly limited in number, the widening data is not. With the concept of "halt" the output is defined as the data at halt state.
Now the input or output collections are defined too.
For inputs it means simply all those inputs that lead to any output at all.
For outputs it means trying all possible inputs but collecting only the obtained outputs. Both qualify as Effective from a verification view. Indeed any finite long data alteration sequence can be verified or refuted to obey the machine rules and reach the halt state. Merely collecting the beginnings or ends of these is then our choice. Such verification or refutation process for finite data sequences as inputs should be a machine itself, namely a decider that is actually two machines with complement input sets. So our earlier claim about obtaining all effective collections from decidable ones is again demonstrated in abstract.
A more surprising claim is that both of these collection methods can be totally mechanical, not relying on the lucky finds of finite data alteration sequences. So the subjective choice can be eliminated. After all that was the whole machine goal.

This relies on the arbitrary long data width. The method is simple! We try out all possible inputs! Of course a halt might take arbitrary long to reach, so we can't wait for every trial literally! Instead we try the inputs only for few steps of our machine alteration. We store the result and try new inputs. Then we always return to try more steps but also go ahead and try longer and longer inputs. Combing these together or "dovetailing" as the joints in cabinet making, we can accomplish the test of all inputs for all lengths of alterations.
None of these arguments used the inputs as programs yet. That crucial view is needed for the existence of a universal machine.
Also, though only the inputs were envisioned as programs, the outputs can again be inputs and so be programs too. In short, all data can be regarded as potential program. So object collections are actually program collections. This is a vital new view!
The effective collections all lead to Gödel numberings, a seemingly artificial assignment of numbers to collections. Now with machines as computers, all this changes. The objects are naturally programs, and thus describe machines and so effective collections of programs themselves.
If our program representation of the objects grasp all effective sets, then we can merely regard the effective sets as the possible [a] , [b] , [c] , . . . sets.
But even this hides a problem. Why should all objects that is input strings give a program? Well, if an s string is meaningless as a program, then s shouldn't collect any objects, so [s] = 0 = empty set. But there could be meaningful p programs that collect nothing too, so [p] = 0 too. This formal coinciding of empty collections as variants of real empty collections with meaningless programs, shows that the emptiness will have a special importance. The variants for the other collections, that is [p] = [p'] is the more general important issue.
These variant programs can also be called as the mentioned redundancy.
Here, with effective collections, of course, we don't aim for all subsets, only for the intuitively effective ones. In fact, we already claimed that the major point here is exactly the opposite of fullness. Indeed, most [s] effective sets have no effective complement. In short, there is no t that $\neg$ [s] = t.
And yet, the simplest effective sets, like the triplets of the basic operations or the composites, have effective complements. Indeed, $x + y \neq z$, or x being not composite, that is being prime, are easily recognizable too.
So why do we claim this naturalness of non effective complements, when we can't even give easy examples for it? All this becomes clear soon!
The only obvious non effective set that we can create as start is the same as we used in Cantor's argument, that is {s ; s $\notin$ [s] }. There, it was enough to show that the assignments can't be total, that is for all set of objects. But now, this is not the issue.
We want an effective set without effective complement. So, {s ; s $\notin$ [s] } is only a good example if its opposite, that is {s ; s $\in$ [s] } is effective, that is a [t].
This would follow from the more general fact that the (p , q) pairs for which q $\in$ [p] is effective. So, {(p , q) ; q $\in$ [p] } = [r]. This r is a program that defines a machine that recognizes all these (p , q) pairs. Well, we said that machines can be replaced by program inputs, but this means two things in detail. Firstly, that a single r machine can do that for all p machines and that this replacement is faithful as collection. This means that a fix r can recognize if a p collects q. The replaced machine can be arbitrary complex, because the p can be arbitrary long. The universal machine that imitates all machines with p programs can be even more faithful in the details of object alterations, but we don't care about that right now. All that matters for us is that r can recognize the collections.
When we'll explain the machines in detail, we'll show the actual universal machine.

From the existence of $[\,r\,] = \{(p\,,q)\,;\,q \in [\,p\,]\,\}$ then it's obvious that $\{s\,;\,s \in [\,s\,]\,\}$ is also a $[\,t\,]$, because in the inputs $(s\,,s)$ is recognizable or rather verifiable in finite steps, if we look for it. As we see, our application of the Cantor type, non effective complement, relies on the extensions of inputs.
In the followings, we won't need this anymore and so the $a\,,b\,,c\,,\,.\,.\,.$ objects don't have to relate to each other in these manners. They could be single natural numbers, tuples, strings, words, or even formulas of a language.
We'll assume only two abstract claims about the effective sets, that is the ones that have index or program, that is the ones that can be obtained as $[\,p\,]$.
For the first, lets imagine that we have a sequence of effective $S_1\,,\,S_2\,.\,.\,.$ sets.
This simply means that they are $[\,s_1\,]\,,[\,s_2\,]\,,\,.\,.\,.\,.$ This raises the question whether a single program could collect them all. This doesn't depend merely on the sets.
Indeed, even if we have only one fix or constant C set or nothing for the S sets, that is a $C\,,0\,,C\,,C\,,0\,,0\,,0\,,C\,,0\,,\,.\,.\,.$ sequence of sets, if the order is random, then together they are not effective. A generation of the order should be given by an a object, namely according to whether it is in a B set or not. So:

$$\{\,a \in B \rightarrow C\,\} = \begin{cases} C \text{ if } a \in B \\ 0 \text{ if } a \notin B \end{cases}$$

This $\{\,a \in B \rightarrow C\,\}$ set is actually an $S\,(a)$ set function of a, but it depends on the given $B\,,C$ sets too. If $B = [\,b\,]$, $C = [\,c\,]$ are given effectively, then $a\,,b\,,c$ determine completely $S\,(a)$ and effectively too. Thus, we claim that there is an indexing function for this, abbreviated as $(a\,,b\,,c)$. So:

$$[\,(a\,,b\,,c)\,] = S\,(a) = \{\,a \in [\,b\,] \rightarrow [\,c\,]\,\} = \begin{cases} [\,c\,] \text{ if } a \in [\,b\,] \\ 0 \text{ if } a \notin [\,b\,] \end{cases}$$

Our second assumption is a reverse collection indexing for the a variable in $(a\,,b\,,c)$.
Namely, we collect those values for which $(a\,,b\,,c)$ falls into a given E set.
That is, the set: $\{a\ ;\ (a\,,b\,,c) \in E\,\}$. Of course, this should be effective only if E is effective too, that is given as $E = [\,e\,]$. Then, the reverse collection has an index that we abbreviate as $\mathrm{inv}\,(e\,,b\,,c)$. So, $[\,\mathrm{inv}\,(e\,,b\,,c)\,] = \{\,a\ ;\ (a\,,b\,,c) \in [\,e\,]\,\}$ or
$a \in [\,\mathrm{inv}\,(e\,,b\,,c)\,] \leftrightarrow (a\,,b\,,c) \in [\,e\,]$.
We'll use this inverse index for a very special $[\,e\,] = E$.
An s is an "emptiness object" if $[\,s\,] = 0$.
E is an "emptiness container" if all emptiness objects are in E, that is:
$[\,s\,] = 0 \rightarrow s \in E$.
The U set of all objects is of course a trivial emptiness container. In fact, even leaving out some obviously non emptiness elements is fairly trivial.
So as "non trivial", we require that $U - E = \neg E$ should contain not only a c object, but all its variants too. That is: $c \in \neg E$ and $[\,c'] = [\,c\,] \rightarrow c' \in \neg E$ too.
Now we can claim our grand theorem:

T If there is an effective $E = [\,e\,]$ non trivial emptiness container,
then every effective set has effective complement,
that is, for every b, there is a d, that $\neg\,[\,b\,] = [\,d\,]$.

P Let c be the claimed existing element with all its variants in $\neg\,[\,e\,]$ and a , b be arbitrary objects. Then:

$$[\,(a, b, c)\,] = \begin{cases} [\,c\,] & \text{if } a \in [\,b\,] \\ 0 & \text{if } a \notin [\,b\,] \end{cases}$$

So, $a \notin [\,b\,] \rightarrow [\,(a, b, c)\,] = 0 \rightarrow (a, b, c) \in [\,e\,]$

But the reverse is true too, because:

$(a, b, c) \in [\,e\,] \rightarrow [\,(a, b, c)\,] \neq [\,c\,] \rightarrow a \notin [\,b\,]$

Indeed, both $\rightarrow$ can be seen easily in its negative forms:

$[\,(a, b, c)\,] = [\,c\,] \rightarrow (a, b, c)$ is a variant of $c \rightarrow (a, b, c) \in \neg[\,e\,]$

And $a \in [\,b\,] \rightarrow [\,(a, b, c)\,] = [\,c\,]$ by definition of $[\,(a, b, c)\,]$

So in short:

$$\begin{tikzcd} a \notin [\,b\,] \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & (a, b, c) \in [\,e\,] \arrow[d, leftrightarrow] \\ a \in \neg[\,b\,] \arrow[r, leftrightarrow] & a \in [\,\mathrm{inv}(e, b, c)\,] \end{tikzcd}$$

So $\neg[\,b\,] = [\,\mathrm{inv}(e, b, c)\,] = [\,d\,]$

If we wouldn't know that there is non effective complement from the Cantor $s \notin s$ argument and the universal machine, then we should expect by this theorem, to find a non trivial emptiness container and then find the effective complements for all sets. But knowing that we do have non effective complement and we even promised a lot of them, we should restate the theorem into its negative form:

T' If there is effective set so that its complement is not effective,
then there is no effective, non trivial emptiness container.

The reverse of this claim is:

T'' If there is no effective, non trivial emptiness container,
then there is effective set so that its complement is not effective.

A much better claim is:

T''' If there is no effective, non trivial emptiness container,
then there is an infinite multitude of effective sets that all have complements,
that are not effective.

And then of course T' + T'' together means:

T'''' If there is a single effective set, with non effective complement,
then there is a multitude of these.

We'll prove T''' but this T'''' is the real gem.
It shows that there is an abstract world behind effectivities that we still don't quite understand. The effective non trivial emptiness container is the simplest fact that implies this spreading of the non effective complements.
Abstraction will be pushed even further, because behind the heuristic idea for the multitude of non effective complements, the proof itself will be almost trivial.
The heuristic idea is the following: We'll collect programs by their operations but not in the details of such operation, rather by merely the collected [p] sets.
So we state some $\mathcal{P}$ property about sets and check if [p] is such.
If yes, we collect p itself. This could be called operative collection, but we need effective operative collection, which means that $\mathcal{P}$ must be effective.
This means that we introduce a third level of effectivity.
The objects having the program meanings was the first. The sets being possible effective collections is the second. And now, $\mathcal{P}$ properties of sets is the third.
The first two levels are related through the fundamental [p] collection method, which we assumed. Luckily, this third level won't be used in the abstract proof.
In spite of this, the effectivity of a $\mathcal{P}$ (S) is usually very trivial.
For example, if $\mathcal{P}$ (S) claims that there are twin primes in S, then this means that we have a way to represent the objects as numbers, that is $\lhd$ is determined. Then of course, addition, multiplication and primality is meaningful. This is clearly effective, that is verifiable. Being twins, that is two apart is easy too. So, $\mathcal{P}$ (S) is effective.
But this is truly effective only if S is an [s] effective collection itself.
Then, the two twin primes as objects can be verified by the program that collects S with adding the primality and twin verification. So the collection of those p where $\mathcal{P}$ ([p]) , that is [p] contains twins primes, is an effective set.
And practically, this is it! That's enough to guarantee that the complement, that is the set that contains those p programs for which [p] don't contain twin primes, is not effective. I said "practically", because a minute additional condition is needed.
Namely, that there has to be some p that satisfies $\mathcal{P}$ ([p]) and also some that doesn't. So we shouldn't take empty or full properties, collecting the whole U.
The real strong condition was hidden in the operativeness! Indeed, observe that by simply collecting the p-s through how [p] behaves, means that we always collect all variants. Indeed, if [p'] = [p] then they both satisfy $\mathcal{P}$. Most amazingly, our proof only relies on this feature of containing all variants. First observe, that trivially, if a collection of programs is closed by variants, then the complement is automatically closed too. Indeed, both $\mathcal{P}$ and $\neg\mathcal{P}$ collect all variants. This symmetricalness can be continued by putting the minor conditions as both $\mathcal{P}$ and $\neg\mathcal{P}$ being non empty.
So, a better symmetrical way of saying our claim too about the non effective complement is that if we split the U universe of programs in two sets, P and $\neg$P that are closed by variants and are not empty, then they both can not be effective.
The real advantage of this symmetrical form is that here we can choose one of the claimed existing elements in the two splits to be special kind. Indeed, every object must occur in one half of the split. Not surprisingly, the special objects will be the emptiness programs. They all must be in P or $\neg$P. So, the one that contains them is an emptiness container. But, the other has another program with all its variants, so the emptiness container is non trivial too. So, if it were effective, then by the T theorem,

all effective sets had affective complements. Or by T’’ this emptiness container can’t be effective.

Using the practical method of effective $\mathcal{P}$-s we can produce a whole arsenal of effective sets without effective complements. Having a particular element in [p] is the simplest effective property about [p]. And so, the set of programs that collect the p-s for which [p] doesn’t contain a particular number, is non effective. There is no program that can collect all those p programs whose collections avoid a fix number. Another way of saying this is that though the collection of p-s whose program avoids a fix number is operative, it is not effectively operative. So operative collection by a $\mathcal{P}$ can only be effective in one half of $\mathcal{P}$ or $\neg\,\mathcal{P}$. But this doesn’t mean either/or, because they can be both non effective. So, this “or” was exclusion.

For example, to have only finite many primes in [p], is a $\mathcal{P}$ property that is not effective, because we cannot verify it, but its opposite $\neg\,\mathcal{P}$, having infinite many primes is not verifiable either from finite concrete cases.

This grand result with the multitude of non effective complements, is Rice theorem!

It explains why it seems easier to find effective sets with effective complements, in spite of more having non effective complement.

Namely, because we define these “simple” sets formally, not operationally.

If we collect p programs by how they look, then we end up easily with effective complements. We have to run the p and see how it collects itself, then we always get non effective complements.

<u>10. Turing Machines</u>

We already revealed that machines are object alterators. Even the fact, that this relies on inner structure, so the objects are strings of symbols or words. We also mentioned the heuristic concept of programs as inputs that influence the jumps or cycles of the machines. Before real computers were built, Turing tried to simplify to the bone the concept of such machines. We mentioned that numbers can be imagined as repeated strokes but to separate them and thus represent more of them, that is tuples, we need a second symbol at least. So the 0 , 1 binary digits is the minimal alphabet. Using them as actual binary digits for the numbers already, is much more economical than repeated 1-s. For example, instead of twenty one digits we only need five, because:

$21 = 16 + 4 + 1 = 1 \bullet 2^4 + 0 \bullet 2^3 + 1 \bullet 2^2 + 0 \bullet 2^1 + 1 \bullet 2^0 = 1\;0\;1\;0\;1.$

Of course if we use this, then we need a third symbol for separation. The minimality of the alphabet is not an issue at real computers. The point is that we only need finite many basic symbols. On the other hand the electronic representations also implied a certain binary representation of all symbols because the early memories relied on magnetized or non magnetized states of tiny little rings. Those days are gone, and today the elementary bits of information are not sitting at specific locations, rather are circulating with a certain fix clock speed. At one location tens of thousands of bits are continually and yet can be grasped in exact groups. So the binary image of computers went through a whole evolution with changing meanings.

To regard the Turing machines as some kind of origin of computers is totally false!

Real computers have evolved on their own ways and Turing’s ideas only helped the motivations not the innovations. More surprisingly, Turing machines are neither the abstract form of Effectivity, not even machines in general. The opposite claim has different levels of stupidity or false oversimplification behind it. Some just use Turing Machine as a metaphysical label to justify all kinds of alternative mathematical

rubbish even contacts with aliens. Much more dangerous is the enthusiastic computability trend, that tries to use Turing Machines as the well defined abstract platform of a new age mathematics. There are two main branches and they couldn't agree on a common lingo yet. So the first is calling itself Algorithmic Information Theory. The other regards computability not only as a new preferred name of Effectivity but also defining the hidden truth of the Church Thesis. You can call anything in any way you want. You can even emphasize new aspects. The lie starts when you try not to tell real problems. Covering up might be for the sake of simpler explanations, but the explanations don't become simple. They only create parrots.
Two special form of lies in these two branches are the falsification of not only Turing but Gödel too. Gödel failed to express his views, so became the patsy in their hands.
The most absurd situation is that from all this, I look like someone who tries to lessen the greatness of Gödel or Turing, when in fact I want to show how much more they contain than the oversimplified formal results. Of course, now lets stick to Turing!
Lets forget about machines as data alterators! The real purpose of this, is to make even more precise the concept of verification. Effectivity is verifiable collection of objects. But mathematics was the most exact science already, before New Math started with Logic and Set Theory or Effectivity. Simply because already before these, the verifications of proofs were exact. Nobody knew how the great proofs are born, what brings about creativity, but once these gems are found then the healthy formalism puts them into exact lines of arguments. Most amazingly a second line of secret ability is possessed by all of us. Lets not venture into now whether the first line, creativity is in everybody or not! Maybe this second definitely apriori gift is even more important. So, without Logic, all people can already verify. With Logic, then we can see that behind this intuitive ability, lies a totally rigid line from simple axioms to theorems. In fact a crucial feature of this Logic is indirectness, which allows to assume the opposite of a C claim and then derive a contradiction A and $\neg A$. The two have their own separate derivations. If the chain leading from $\neg C$ to A is $\neg C \rightarrow \;...\; \rightarrow A$, then it also means $\neg C \rightarrow A$ as theorem. Similarly we have $\neg C \rightarrow \neg A$ as theorem. Implications of course can be reversed negatively, so we have $A \rightarrow C$ and $\neg A \rightarrow C$ as theorems. These imply $(A \text{ or } \neg A) \rightarrow C$, and
$(A \text{ or } \neg A)$ is an axiom of Logic, so the indirectness can be avoided formally.
The actual original chains have to be detailed step by step and they hide the creativity and also the details to be verified by our attention or a machine. For a machine, a totally indirect and unified chain to simple "false" is the "easiest" to verify, because it starts with the negative target and ends with a fix form. The derivation steps are still secondary existences, lucky or ingenious finds. But from a statement, we and a machine too, can list all possible consequences. Then such full list has no secondary existence. The input is the only existence, the only assumed condition. The verification will exactly tell merely from the input if we have a theorem or not. This sounds like a contradiction according to our earlier claims that the theorems are not decidable because the non theorems are not effectively collectable. This is so because I intentionally abused the word verification. It is true that every full derivation set from an input can be verified whether did or did not lead to contradiction, but this is only an effective collection for the successes that is contradictions. A non theorem negated, will create an infinite consequence set that contains no contradiction. But we can not tell this from finite many cases. The contradiction search is a totally verifiable method but not a decidable method, because involves longer and longer decisions. Most amazingly this failure is actually not only the root of the Incompleteness Theorem, but also of the Completeness Theorem. Indeed, the failing contradiction search is actually creating a model for the assumed negated statement.

From all this, it is clear that a successful representation of Effectivity in general also should use such full verification, without secondary existences. No subjective creativity, that is lucky finds. From the input a totally mechanical process should create all the consequences or alterations. The goal of course can be anything not a contradiction but any property of the generated alterations. But this property must be complex enough not to be found always. This halt condition is then effectively recognizable and so the inputs leading to it are also an Effective set. But the negative, the non halt is not effectively recognizable and so the non halting inputs are usually not Effective either. So is there an abstract object alteration method, that combines all intuitively acceptable scenarios? Not to our present knowledge! But the Turing Machines were never even targeting this goal, and this is the main point to see first!
The Turing Machines are coming from a totally different point. From space and time, from very classical mathematics and then offer an almost accidental window to define Effectivity too. Already our earlier vision of inputs as words or strings should be abandoned for something much bigger.
An infinite set containing only finite many possible symbols, the alphabet as elements is impossible because a set can't have more same elements. This is a fundamental feature of sets that the elements collapse. This is in stark contradiction with the everyday practice of math where we repeat things in slightly different meaning. So quite simply $\{ 1 , 1 \} = \{ 1 \}$. Set Theory conquers this problem with artificial methods. It creates more complex sets and then repeating old elements of elements is acceptable. So $\{ 1 , 2 \}$ can be combined with the element 1 as $\{ \{ 1 , 2 \} , 1 \}$. In fact already $\{ 1 \}$ can be as $\{ \{ 1 \} , 1 \}$. What these tricks really hide is that if a set has structure, that is relationships among the elements, then putting finite many variants into the elements, they won't collapse into the mere finite set because the relations are hidden parameters that distinguish the finite variants. As a simplest example we can put letters into the infinite many unit squares of the coordinate plane.
In fact, looking with a magnifying glass, all children realize that the colored pictures of magazines merely contain the same three colored dots. A painter on the contrary uses his brush with amazing variety to create the spots. But, while the painter has to create the whole painting spot by spot, the printing machines create all points at once.
Now, imagine to combine the worst of both worlds! Fix spots and created spot by spot! So we have a squared or maybe triangularized or hexagonalized plane or structured set in general. We put colors or any fix symbols into the elements. The ordering of the set allows this. This also defines the neighboring elements and so after every symboling of an element like coloring of a spot we also have a finite option of new neighboring spots. So we have two alphabets. One for the symbols, like the fix colors and one for the possible moves to neighboring spots. For example, with the red, blue, green basic colors only and a squared plane with the north, south, west, east moves only, we have the symbol alphabet: R, B, G and move alphabet: N, S, W, E.
So regarding a symbol writing plus a followed move as an elementary action, we'll have exactly twelve such: (R , N) , (R , S) , (R , W) , (R , E) , (B , N) , . . .
Writing colors and moving alternatively, we get a continuous coloring of the plane.
Of course, nothing forbids us to return to old spots and change their content. This suggests that instead of starting from an empty canvas, we can regard painting as an alteration of pictures. That's what Leonardo told his students, to stare at old walls and "see" the picture into them. Formally, even the empty canvas can be regarded as an alteration by simply accepting white as a fourth color. Or in general, for the locations as data containers and the symbols as data, we can accept "empty" as a symbol or data.
Leonardo's deeper interpretation of creation, as allowing the initial space to influence us, has a modern reverse twist, as regarding our finite alterations from the arbitrary

rich symbolings of a set. Indeed, we already saw that the simplest yes/no choices are actually defining the subsets. So any categorization or transitions among the symbolizations of sets is a step towards the holly grail, the Continuum Hypothesis.
But we will realize that the formal generalization to an alteration of the full space, and even this deep opportunity behind it, are very different from the necessary alteration of old alterations. This can be seen from defining our crucial goal informally first.
Using elementary actions still allows us to guide the changes from what we see from the outside. A fully internal method should decide the next actions relying on the last few in time and on a few symbols that were around in space. An extreme case would be merely using the last action and the symbol at the spot we moved to in this last action. In our above example we had twelve actions and three symbols, so we have exactly thirty six possible such previous situations. To tell exactly for each into which action to go is very narrow. To allow them to be chosen freely is hiding an external guidance. I don't reveal yet the crucial system, but we can see that the pre existing symbols are a necessity of such method trivially that is formally. But much more importantly, the re alterations will be crucial for a system to even "work".
To paint a picture by "elementary actions", that is finite many fix colors and spot by spot movements, is not that absurd. The painter can still be guided by his goal and move from the cat's tail to the head. But lets forget the two ends, creativity and totally internal determination! Instead, lets regard the middle! Effective alteration of the plane through elementary actions but guided from the outside. We could for example decide that around a center red spot we'll draw circles with increasing radii difference. We could draw a bullet that flies on a line with traces of the prime numbers. These all involve distances or coordinates and give a big freedom of actions even if we are effective. So, can these external effective designs be translated into rules that are determined by elementary transitions that we defined above ? The tricky part is the meaning of the word "translated". If we mean exactly producing the same alterations in time as an external method, then we can't succeed. But if we only mean the end result of an externally effective alteration sequence, then we can.
It's quite obvious that we can't simulate externally determined alterations exactly, by elementary transitions. Indeed, externally we can go forward without alterations of earlier changes and create patterns by locations or coordinates. Elementary transitions can only achieve such external orders by repeated alterations. The primality of a coordinate for example is given externally, but internally can only be established by trial divisions. So the history of the alteration sequence can not be simulated only a final coloration. Even this is quite unbelievable. Yet it's true! Internal methods can imitate all effectively obtainable full alterations!
This is the exact opposite of what we saw at the strange models. Those come about because the language can not express what we see from outside. Here this new inside language of Effectivity is enough to express even the outside ones.
This raises some very important questions about genetic determination. The fact that the living creatures all develop through specializations of identical initial cells, and the operational rules are stored in every cell, almost dictates that we are dealing with a similar situation. And yet any sane person must have a certain doubt that a kitten is merely an internally determined growth into the perfect final form. This is called the "morphological paradox" and it is indeed a paradox. It is the lowest form of mystical purposefulness imbedded into a seemingly perfect material determination. So it is the lowest battleground between idealism and materialism. Right now I only want to point to one factor. Even though the cells also die and recreate themselves, this is totally different from the above mentioned crucial re-alterations. The two must have some connection that we have no clue about yet at all!

So it's time to reveal the system of elementary transitions! It is amazingly simple! Instead of using only the possible elementary situations, that is the elementary actions combined with a possible last symbol at the last move, we allow an arbitrary large number of such and define a transition for each. A better visualization is the following: We place an arbitrary number of elementary actions, that is (,) pairs of symbols and moves into the plane, and then draw the transitions as arrows. The arrows are as many kind as many symbols we have. For example with the red, blue, green colors as symbols we can color the arrows. With other symbols we have to mark the arrows. Of course we can also use different solid, dotted wavy and so on arrows. The arrows depart from the second bracket of (,) to emphasize that the symbol read after that second move is determining the arrow type. So we have to depart all possible read symbol that is all possible kind of arrows. They can point to different other actions and we visualize them pointing to the first bracket, because the first part there the new symbol writing comes first. Then the second part the move. This "action web" can be followed through the arrows and the actions themselves continually. We never stop. The only freedom is where we start. The read symbols of course are provided by the actual space which has to be filled with symbols as start. We also have to start from a given spot. So the initial spot of the real space and the initial action point of the web will determine the eternal process of travel both in the action web and in the real space, re-coloring or re-symboling it at the same time.
Nothing forbids that from an action point we would go by one or maybe even more of our arrows back into the same point. Then we can simply omit these arrows:

(,) automatically includes
$$\begin{tikzcd}
(\quad , \quad) \arrow[loop above, dashed] \arrow[loop below]
\end{tikzcd}$$

Using instead of the plane, an infinite line of data containers and instead of colors, 0 or 1 data content, we have four possible actions:

$(0, \rightarrow)$	=	replace the data with 0 and move to the right.
$(1, \rightarrow)$	=	replace the data with 1 and move to the right.
$(0, \leftarrow)$	=	replace the data with 0 and move to the left.
$(1, \leftarrow)$	=	replace the data with 1 and move to the left.

An action web could be:

$$\begin{tikzcd}
 & (0, \rightarrow) \arrow[dl, dashed, bend right=60] \arrow[dr] & \\
(1, \rightarrow) \arrow[ur, dashed] \arrow[dr] & & (1, \rightarrow) \\
 & (0, \leftarrow) \arrow[ur, dashed] &
\end{tikzcd}$$

The dotted arrows apply if we read 0 and the full if we read 1.
The missing arrows again mean self returning ones, so both are like this for the last action and the full arrow for the lowest.
The simplest web is a single (1 , →).
Remember, this means that both arrows return to itself.
The resulting alteration is that wherever we start on the line, we write all 1-s towards the right up to infinity. It’s an unconditional writing sequence.
A more complex web might also contain such trivial in it. In fact our picture above did contain exactly this at the right end. So the other cycles all end up in this obvious, unless they avoid this one. Remember, that for every starting action in a web and starting point on the line, we enter into an eternal process in both. We might avoid some actions in a web and avoid some points on the line. What we avoid or get into, depends not only on the web and the symbols on the line, but also on the chosen initial points in the web and on the line.
If we only use a half line say towards the right, that is an infinite sequence of data containers, then they could be addressed instantly by the naturals. Or to put it in an other way, the naturals were a trivial coordination of our space. Then we could create natural, externally effective alterations.
Turing’s crucial idea was the exact opposite, to avoid addressing the data containers.
Instead, to create an alteration of the half line, letter by letter and always moving to the next letter forward or back. He even mentioned a childhood memory about his mother’s typewriter as the inspiration for this.
Of course a half line only as our space has the conflict with the basic two directional moves. Namely, what if we reach the left end of the line and the next action becomes one with the ← move? To avoid this we must have a special symbol located at left end, that when read can tell not to go left. We have an other problem too:
An infinite line or half line of data containers with all 0-s as initial content would mean that we regard the 0 as the empty symbol. Changing them to 1 is the real data but the 0 should be used among the 1-s to enhance the information too. After a finite alteration of course we can have only finite many 1-s and the rest of the line is all 0-s.
A question is where the changing data should start and end. We could cut off the two infinite 0 ends, that is regard between the first and last 1-s. Or in one directional line just regard the last 1. But actual alteration might have taken place beyond these points, with later 0-s as symbols written.
Not starting from empty initial line, rather having a finite input would suggest even more to have some kind of end markers. On the other hand, leaving the door open to an infinite complex initial line was an exciting opportunity to Turing from the start.
So as we see the infinite starting line with two directions, is very much the pure form of the basic idea. And yet the universal Effectivity lying behind it, initiated at once the pruning of the basic idea. Using one directional data line, end markers, finite inputs, going toward natural numbers.
As a start, lets just see how easy it is to restrict a web, to use only one half of the infinite data line. Lets introduce as a third symbol the • dot for the left end. We place it on our line somewhere and we start from the right of this. We want to keep this fix on the line and never be able to pass it by.
The elementary actions change from four to six in number because we have two new ones: (• , →) and (• , ←) .
By our goal we never want to use this second one. But merely avoiding it in a web is not enough. Now we need three kind of arrows. A new should be used if • was read. After this, that is through this new arrow, we always want to go into a (• , →) action, except from these actions themselves we always go into an old.

Indeed, if we read • as next symbol then we want to keep this symbol there and turn back to the right to an action involving only 0 or 1 writing.
So we can place some (• , →) new actions on the left of our web and create the new double lined arrows from each old actions into one of the new ones and one new arrow from each new actions into the old web:

$$\begin{tikzcd}
(\bullet, \rightarrow) \arrow[dr, Rightarrow] & & (0, \rightarrow) \arrow[dl, dashed, bend right] \arrow[dddrr] & & \\
& (1, \rightarrow) \arrow[ur, dashed] \arrow[dddrr] & & & \\
\vdots & & & & \\
& & & & (1, \rightarrow) \\
(\bullet, \rightarrow) \arrow[rrr, Rightarrow] & & & (0, \leftarrow) \arrow[ur, dashed] &
\end{tikzcd}$$

I only marked the returning double arrows from the new actions. It's quite visual how this web will turn back on the actual data line after any encounter of the • dot.
Also observe that we don't need the new arrows backwards from all actions. It's enough to define them from the (, ←) type ones. Indeed, we could never approach • from the left. That is, after an (, →) we should never read • anyway.
This shows that we could even use the same • as a second marker for the right end of data. This would only involve right encounters, so the other (, →) type of actions. But here a single rebound is not enough if we want to be able to move its position.
A much deeper problem is whether these restrictions narrow the obtainable effective alterations. My intention is toward the programs and universality, so these will at least suggest that the restrictions still leave wide enough Effectivity.
As a start I already admit that a second right end • will be used, that won't be a flexible end of data marker but rather a "kind of" fix one as the left end. Between the two dots as the initial segment of the data line will be the program for an arbitrary web. So this segment can be arbitrary long just as a web can be arbitrary complex. There is nothing surprising about this part. Under one fix web simulation, the initial program length remains the same but the actual program does alter. In fact the machines as alterators will be the crucial method of running the program. The idea of a program being a fix set of data is a nice fantasy but it doesn't work! Literally, because "work" means alteration. Of course we don't want to loose the program either! So a continuous restoration will be kept. The input data will be after the second dot to the right. The program will be simulating the target web, working on this input. The simulation will be in the pure form, that is without artificial halt conditions for ever. The input can be even infinite. The only deviation from the pure original form is that we use the one directional infinite line, that is we have a dot as a single left end dot for the webs too.
In a sense, we simply put the two dimensional webs on the one dimensional line between two dots. The crucial hard part in this is that a web can have arbitrary many actions. To locate all of them requires arbitrary large labels like numbers. This is true in two dimension too and we simply ignored it in our visual picture as locations. On the line they have to be marked exactly. Luckily, this doesn't mean arbitrary many basic symbols. In fact the simple repetitions of 0-s and 1-s will locate the different actions. We need both because it's not enough to locate the action, we have to store the possible addresses of the next actions too. So the 1-s will be the actual locations and the 0-s the addresses. After the 1-s will be five possible symbols for the five possible actions and after these, three groups of 0-s corresponding to the three

possible addresses depending on the read data as 0 , 1 or • . We'll also need a ? as cursor to mark the last used action. Thus, the three addresses after this ? will tell where to move this ? and thus also find the next action. This changing of ? is the hard part and it needs the use of a second ! cursor. Indeed we have to find as many 1-s as many 0-s are in the address. So the trick is to put a ? into the beginning of every location label, that is group of 1-s and put a ! into the beginning of our address. Then keep stepping forward with all ? among the 1-s step by step as with the single ! among the 0-s. The single surviving ? cursor will exactly be the next location.
So the initial program section of the line, without the cursors is:

• , 1 ( ) 0 . . 0 ( ) 0 . . 0 ( ) 0 . . 0 , 1 1 ( ) 0 . . 0 ( ) ……………….. ( ) •

Every comma starts a new action point with increasing many 1-s as label for it.
After these, comes the ( ) action symbol that can be five kind, corresponding to the five possible actions : (0 , →) , (0 , ←) , (1 , →) , (1 , ←) , (•, →)
Then comes three groups of 0-s separated by dummy use of action symbols.
The three groups correspond to the three possible arrows, that is the last read symbol.
The first can be for 0 the next for 1 and the last for • .
What is still missing from this program section is the cursors.
If we come from the data section, that is from the right of the program, with the last read data, that is 0 or 1 or • "in our pocket", then the location of the last used action symbol was already established and so we have a single ? cursor somewhere after that symbol. So it is in place of a first 0 in a first address.
By the way, we will also use the ? cursor on the data side. It will be where we read the last data in our pocket! So now we have to travel from that ? to the other one in the program. That's easy! We read the data and if it's not ? then we rewrite it unchanged and move left. When we finally encounter ? then we have to decide whether we are in the right address.
If we have 0 in our pocket then we are, so we replace ? with ! If we have 1 then we replace ? with 0 and look for the second group, that is we place ! after the first dummy ( ) and if we have • , then after the second one.
Now comes the mentioned trick:
We place a ? instead of every first 1-s, that is after every comma.
After this, we can move the ! to the right. This means changing it to 0, move to the right and write the ! there. If that next symbol is not a 0, then of course the address was already over and so we shouldn't change it rather "clean up" the ? location.
If we did move the ! to the right then we simply go through all ? from the beginning and move them to the right too. This of course again means putting 1 in place and putting ? into the next to the right. Now again, this shouldn't be done if we ran out of the 1-s, that is a ( ) action symbol is read. In this case the location was a shorter labeled one than what we look for, so we can simply avoid ? there any more.
This way, only the bigger locations will have ? in their labels when the ! finally is erased. After this comes the "clean up" when we go through again all ? but now we look for an action symbol as next. If found, then that will be the next action and we should place the ? after it. Still, we have to go through and where the next is not an action symbol and thus must be a 1 then we simply leave it, not move the ? there.
But how do we know if all ? in the program was cleaned up? Well, the program ends with • so encountering it tells it at once. The only problem is that we might have read this • as our left end data and replaced it with ? too. To solve this problem we should simply use a separate • for the program end and one for the left end data.

So, finding the ? data cursor, we can execute the five possible action symbols as actual actions, ending with the move. Then we read the symbol there and replace it with ? . Then we go back to the left to find the program cursor ? as we started.
This verbal account of the "meta program" that executes any action web, has to be itself put in form of an action web. The symbol alphabet contains the three assumed ones in the webs 0 , 1 , • and in addition the five action symbols, the comma and the two cursors ? , ! . The five action symbols that we above simply marked as ( ), now have to be distinguished as :
$\alpha = (0 , \rightarrow)$, $\beta = (0 , \leftarrow)$, $\gamma = (1 , \rightarrow)$, $\delta = (1 , \leftarrow)$, $\theta = (\bullet , \rightarrow)$.
So we have eleven symbols, thus also, eleven kind of arrows and twenty two actions.
The first step in our verbal method was the search for ? toward the left.
The translation of this into exact actions means that from our last action, the writing of ? among the data and move to the left, that is from (? , $\leftarrow$) the eleven possible arrows must lead into eleven actions that are all rewritings of the read symbol and left move again, except if ? is found. But that crucial ? encounter should cause three different actions depending on what was read as last data before (? , $\leftarrow$) . Namely:

$$\begin{tikzcd}
 & & (0,\leftarrow) \\
 & & (1,\leftarrow) \\
 & & (\bullet,\leftarrow) \\
 {} \arrow[r,"0"] & (?,\leftarrow) \arrow[uuur,"0"] \arrow[uur,"1"] \arrow[ur,"\bullet"] \arrow[r,"\alpha"] \arrow[ddr,"?"] & (\alpha,\leftarrow) \\
 & & \cdot \\
 & & (!,\rightarrow)
\end{tikzcd}$$

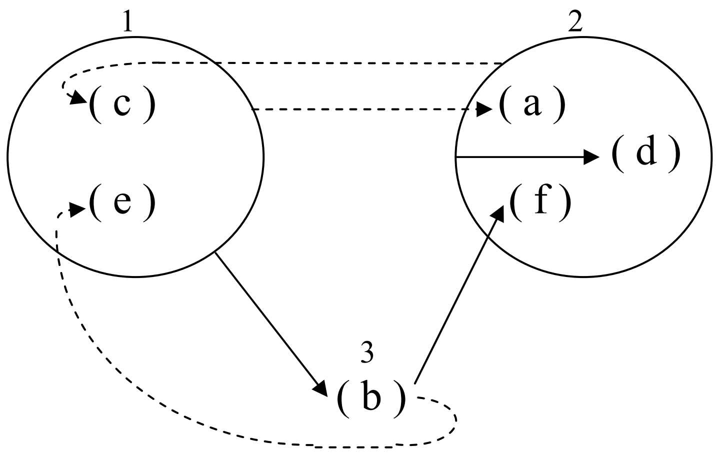
$$\begin{tikzcd}
 & & (0,\leftarrow) \\
 & & (1,\leftarrow) \\
 & & (\bullet,\leftarrow) \\
 {} \arrow[r,"1"] & (?,\leftarrow) \arrow[uuur,"0"] \arrow[uur,"1"] \arrow[ur,"\bullet"] \arrow[r,"\alpha"] \arrow[ddr,"?"] & (\alpha,\leftarrow) \\
 & & \cdot \\
 & & (0,\rightarrow)
\end{tikzcd}$$

$$\begin{tikzcd}
 & & (0,\leftarrow) \\
 & & (1,\leftarrow) \\
 & & (\bullet,\leftarrow) \\
 {} \arrow[r,"\bullet"] & (?,\leftarrow) \arrow[uuur,"0"] \arrow[uur,"1"] \arrow[ur,"\bullet"] \arrow[r,"\alpha"] \arrow[ddr,"?"] & (\alpha,\leftarrow) \\
 & & \cdot \\
 & & (0,\rightarrow)
\end{tikzcd}$$

Remember that the first address we'll use for 0 data, the second for 1 and the third for •. So in the first case we could put the new cursor ! in place of ? directly, while in the other two cases we had to replace the 0. In these cases we have to continue with right searches to pass by the used dummy action symbols say α and β.
These searches are actually new splits according to whether α is read or anything else which is 0 in the first group or β in the second. Once we find α and β, we write ! after this, that is in place of the first 0 of the second or third group.
After these, we continue the same way in the three alternative sequences:
First search toward the left for the beginning • and then turn back and place a ? after every comma until an other • is encountered. These again involve new splits.
Then turn back and search for !. When find it, move it to the right.
But the right symbol reading first splits this process again in two because it may be a non 0, in which case we counted all the 0-s and we proceed to the ? clean up.
If not, that is not the last 0 was counted then we write the new ! and start a new search for the beginning • again.
Then we go through every ? and move them to the right too. But again this splits according to the next symbol being a 1 or not. If not, then we encountered a too short location label so we don't place the ? any more.
The clean up also starts with going to the beginning and then go through all ?.
But now the split is according to whether the next symbol is an action symbol or not.
This in fact means a six way split. The five possible action symbols and anything else, which of course can only be a next 1. This then means a too big action location label, so we can stop writing ? . In the five possible action symbol cases we go first similarly, namely reposition the ? after the read action symbol. Then we continue the ? search till the end • . Then we can go in each of the five main splits till we find the ? among the data and then apply the actions correspondingly.
So as we see expressions like reading a data and "putting it in our pocket" actually meant the three main splits just as the reading of the action symbols meant a five split.
Beside these, we had many minor splits according to conditions for searches.
Once we start the whole process again, we don't have to repeat the splits forever, rather return to the beginning. The arrows return. All infinite paths are cut off.
The infinite effective action tree becomes a finite web.
So this exercise to show that the universal web simulation is itself a web became much more educational in an other sense. Namely we can see now that any effective process of decisions and alterations is merely one big cycle of split roads.
An infinite action tree with a fix number of branchings at every point, like with eleven in our case, means that we list all possible alternatives according to the possible read symbols. The points themselves are the actions. Starting from the beginning of the tree and with given symbols in a space, the actual sequence of actions is a path in the action tree. That is meaningful for any actions that are dependant on the last read data and for any tree and given data space. But our actions are duplets of new data writing and neighbor moves. So these actions themselves, have only from finite many options. This is so because the moves do not require the exact locations of the data places which would mean arbitrary large addresses. This finiteness of the tree points and branchings means that it makes sense to return to earlier points if we want to. If this is cutting off all continuations eventually then the tree becomes a finite web.
Now lets examine the data side! The use of neighbor moves instead of addresses means that the starting place must be given in the data space as the start of the action tree. Then, the data space defines a path in the action tree as the actual action sequence. Cutting the tree into a finite web doesn't change this. In fact the path can be totally unpredictable due to the randomness of the data space. So to define Effectivity

is now logical: We have to cut the data space to finite too! This simply means that we only have finite many occurrences of all symbols except one, which is thus the "empty" symbol too. For binary alphabet we have only finite many 1-s.
Remember, I said that Effectivity was not recognized because it was assumed as a mere finiteness directly. Well, now we split the concepts into two, the decision or action tree and the data space. And we reduced Effectivity to merely being finite for these two components. The reason it works is the tricky use of the neighbor moves in the actions plus the rewritings used in the actions too. Indeed, this allows that the finitely cut initial data space can grow arbitrarily and thus give bigger influence on the action tree to define the path, that is the action sequence. This action sequence and the accompanying changing sequence of the data space is the Raw Effectivity and not the altered space. Indeed usually we don't have a final altered space because the data kept changing infinitely. But we can contemplate at once, that a limit effectivity could exist too if the data all settle after finite changes. But much more important considerations are to be used from the Raw Effectivity first. Indeed this was merely the effectivity of verifications. It is not even apparent what we want to verify, the initial data or the action web. Best is to keep both doors open. What is fairly obvious is that the criteria of verification should only examine a beginning finite section of the infinite alteration sequence. So, actually this is the third finite cutting off. Indeed, why should we get involved in infinity properties about the whole sequence, that would go back to effectivity problems. In fact, we could also jump to the idea of merely using the singular alterations, that is define a final among these if it satisfies some conditions. But there is a much bigger jump to completely ignore the data and use the action tree! This simply means that at a chosen action point we don't put action at all, just stop!
At first this sounds absurd, because how can we ignore the data as result? Then we could ignore the initial data as well. Amazingly that is exactly true! We could translate the initial data into unconditional actions easily. Then the initial data line could always be regarded empty and had the only purpose of giving room for work.
The only reason we kept the door open to initial data is to define latter the programs and collected data easier directly, not extracted from the action sequence or web.
But for the verification conditions or halt conditions, we accept that they are built into the action web. This still may sound stupid, even after we see that it is possible. But the following argument explains it: If we wanted to decide the halt, according to the altered data, then we would have to enter into a new second examination of that. Then we should define a second action web. So why not combine the two at once. This means that we practically can even ignore the alteration sequences. Choosing action points in webs and using them as halt, can verify any wanted conditions about the data changes anyway. The input data is a mere convenience too, but they are the actual objects of effective collections. Namely a set of inputs is effective if they are the ones from which an action web reaches its halt point. The complement, that is the inputs from which the same web never reaches its halt point is usually not effective because the web defines an infinite action path, that is the web runs forever. Only an other halting web could make it effective too. This is the basic picture but the outputs are kept for three reasons. Firstly, real computers still aim for outputs as results. Secondly, the same effectively collectable inputs can be also obtained as simpler outputs. Finally, the Algorithmic Information Theory tapped into a new meaning of the alterations.
In this heuristic interpretation, the input is always a program that produces the output and the main question is if a short input can produce a long output. To produce a million 1-s is easy by telling exactly this. An other way of saying this is that the output is compressible, it doesn't really have much information. But there are always infinite many complex outputs that are incompressible.

Of course there is a basic contradiction in this whole idea in its form as I just introduced it. Namely with bigger and bigger computers or action webs, all big complex outputs can be produced by short inputs. The web can contain one million random actions of 0 and 1 writings, all initiated in a sequence by a single input.
The solution is to use a single universal web. Then such abuses of the webs are excluded eventually, that is for long enough outputs. So then the minimal inputs that produce an output are the compressibilities of it. This new use of the action webs for outputs, encountered other problems with the inputs too.
Also, this information approach merged with an older problem of randomness.
The older attack on randomness tried to reduce it to Effectivity but only succeeded with a heuristic recognition by Martin Löf that was further simplified by Solovay.
This purely relies on an effective collection of binary inputs as potential beginnings that must be avoided as infinite continuing occurrences in a random binary sequence.
This effective collection also has to have a finite total chance value.
Indeed infinite total would allow the collection of all beginnings too.
This definition of random sequences became identical to the information theoretical, first approached by Kolmogorov and then perfectionized by Chaitin. The obvious question of course is why a more complicated approach relying on special universal machines should give same sequences as the straight forward definition of Solovay.
Chaitin's answer is that the "hard yard" is not wasted because incompressibilities are the hidden reason behind undecidabilities and incompleteness.
An obvious resemblance to the already used Barry's paradox is obvious. Indeed Chaitin proved that all incompressible strings above a length are not provably such, in a fix system because a proof of incompressibility would lead to Berry's paradox.
This simply means that there are long truths beyond the scope of any finite system.
But Chaitin interprets this as the true hidden cause of incompleteness. He also regards randomness as the well defined fix property of sequences with complex beginnings.
Both are false oversimplifications of much deeper mysteries.
I went into all these complications just to indicate that the action webs and their applications can not be regarded as <u>the</u> solution of Effectivity. The next explanations show how Turing Machines are specializations of action webs, hiding more than revealing and giving the false impression of a perfect solution to Effectivity:
The obvious simplification in an action web's notation could be to combine actions that have same arrows into the same action point. We could simply circle these and apply the arrow from the surrounding circle. Of course having intermingled circles is not helping at all! An ideal situation would be if all the used circles were disjoint and the circled ones would have all their arrows common, going from their circle. Then, no arrow would have to be drawn from actions in circles. The common arrows from the circles of course still have to point to individual actions may be in circles.
A circle can go into an action in itself too, which also means that that action had the self arrow that we already omitted earlier as a simplification.
If an n symbol web can be circled like this, that is distributed into say m members of groups and left out ones, then that means exactly m n many arrows.
If we have more than that many actions, then some have no arrow going into them, so they can only be starting actions.
If we have less many actions than arrows, then some must accept multiple arrows.
With 0 , 1 symbols that is dotted and full arrows and using two circled groups, and one left out action we have six arrows. A possible simplified web is the following:



Observe that the notation actually means the followings:
(c) and (e) both go into (a) thru dotted arrows.
(c) and (e) both go into (b) thru full arrows.
(a) and (f) go into (d) through full arrow and (d) goes into itself too.
(a) and (f) and (d) all go thru dotted arrows into (c).
This notation of the combined groups and left out members on their own, can be put in a simple table of the transitions too, if we number all the members:

$$\begin{tikzcd} & 2 = a \\ 1 \arrow[ur, dashed] \arrow[dr] & \\ & 3 = b \end{tikzcd}$$

$$\begin{tikzcd} & 1 = c \\ 2 \arrow[ur, dashed] \arrow[dr] & \\ & 3 = b \end{tikzcd}$$

$$\begin{tikzcd} & 1 = e \\ 3 \arrow[ur, dashed] \arrow[dr] & \\ & 2 = f \end{tikzcd}$$

We assumed no point without incoming arrow and all different actions at entries.
If we have common entries then we can simply use same letters for those in this table.
Turing interpreted these numbered members as different states of a machine.
But this doesn't correspond to the actual splittings of an action tree.
The concept of states is misleading in many respects but shortens the notations.
The machine state interpretation means that the above = signs should be replaced with + signs. Indeed, the machine, reading a 0 goes from state 1 into state 2 and also executes the (a) action, which is a writing over the read 0 and a move.
The same 1 state when reading a 1, will go into state 3 and execute action (b).
The new states again split into states and accompanying actions by the read symbol.
As an example for my view against the state interpretation, I now introduce a pretty famous three state machine, namely the three state "busy beaver champion". This is the most complicated among the three state machines.
First I give it in state notation, which doesn't reveal its actions at all.
Then I show my preferred action web meaning in short version which can be read directly as the full web and then can even be "run" as an action tree.
Finally, I show what the changing data line is from an empty start with blank squares denoting 0 and grey ones 1 .

$$\begin{tikzcd} & 2 = (1 , \rightarrow) \\ 1 \arrow[ur, dashed] \arrow[dr] & \\ & 1 = (1 , \rightarrow) \end{tikzcd}$$

$$\begin{tikzcd} & 2 = (1 , \leftarrow) \\ 2 \arrow[ur, dashed] \arrow[dr] & \\ & 3 = (0 , \rightarrow) \end{tikzcd}$$

$$\begin{tikzcd} & 3 = (1 , \leftarrow) \\ 3 \arrow[ur, dashed] \arrow[dr] & \\ & 1 = (1 , \leftarrow) \end{tikzcd}$$