The mathematical and philosophical revolution launched by Godel's Incompleteness Theorem

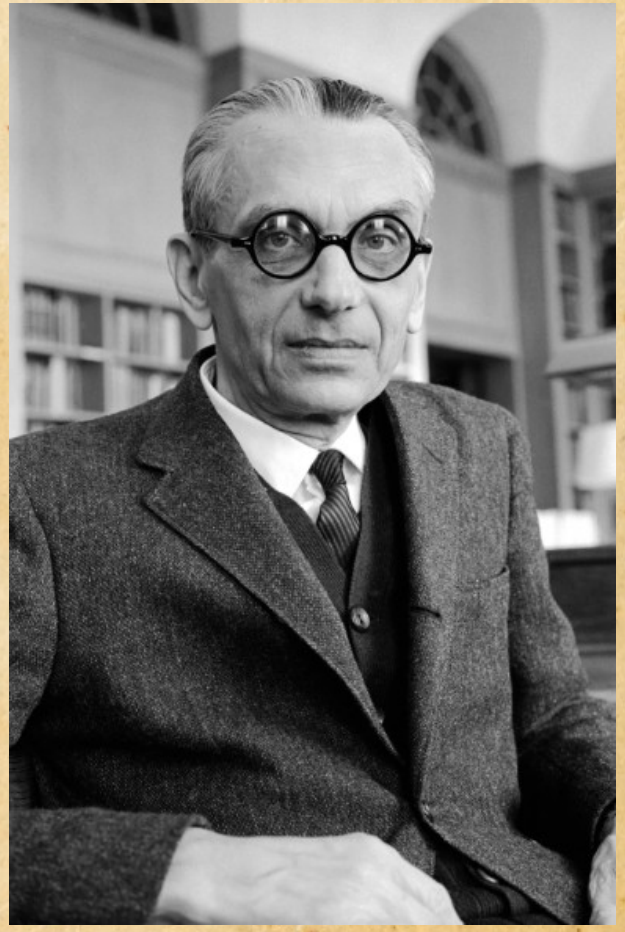
Gregory F. Johnson

Intellectual Fireworks of the early 20th Century

- Quantum mechanics
- Genetics and evolution
- Cosmology
- Technology: electrification, flight, ..

Kurt Godel

Foundations of mathematics take an unexpected and astonishing turn.



The intellectual hothouse of Vienna

Intellectual and philosophical ferment

- Logical positivism was widely influential
- “Scientific knowledge is the only factual knowledge. All traditional metaphysical doctrines are to be rejected as meaningless.”
  - Empirical verifiability
  - Anti-metaphysics
  - Unification of all scientific disciplines
    - “Botany is basically applied physics”
    - Search for The Theory of Everything
  -

A quiet young man listens

Mathematical Platonism

- Godel took a course from the philosopher Heinrich Gomperz
- Pivotal, transformative moment in his life
- Introduced Mathematical Platonism
  - “Mathematical objects, though not visible or accessible via physical experimentation, nonetheless have a mode of reality in their own right."
  - The act of mathematics is one of discovery, not invention.

Godel's worldview

- Godel characterized himself as follows:
  - "Baptized Lutheran (but not member of any religious congregation). My belief is theistic, not pantheistic, following Leibniz rather than Spinoza.
- Viewed afterlife as a logical necessity
- Theistic rationalism:
  - Philosophical and theological perspectives centered on the idea of a completely rational universe where nothing happens without a cause.

Godel's worldview

Kurt Godel's philosophical priors were outside of the mainstream.

They enabled him to imagine possibilities beyond the epistemic horizon of the mathematical establishment of his time.

- What if there are mathematical truths that cannot be proved?

Georges Lemaître similarly had philosophical priors that allowed him to imagine the unimaginable to more conventional physicists:

- What if there was in fact a beginning to the universe?

Mathematical Platonism

We may speculate:

Godel sensed a vast mathematical realm that lay beyond containment or capture.

It is eternal and untamed.

It is inexhaustible, and beckons us ever deeper into its beautiful mysteries.

The search for mathematical rigor

- What constitutes truth in mathematics?
- Can mathematics be completely defined and ‘boxed in’ within algorithmic containers?
- Rudolf Carnap: Mathematicians choose a linguistic framework based on its utility; "truth" is a matter of agreed-upon rules (conventions)

The journey to formalism

- Gottfried Frege created an opus of mathematical rigor, “Grundgesetze der Arithmetik (Basic Laws of Arithmetic)”

- On the eve of publication, a note from Bertrand Russell:

  Frege’s system contained a fatal inconsistency

Russell's Paradox

- "Let S be the set of all sets that do not contain themselves as elements."
  - Does S contain itself as an element?
  - If "yes" then "no"
  - If "no" then "yes"
- Colloquially:
  - "Smith is the town barber, and he shaves every man in the town who does not shave himself."
  - Who shaves Smith?

Further developments

- Russell and Whitehead developed an alternative that they intended to avoid inconsistency: The Principia Mathematica.

- They were unable to prove that their system was free of the inconsistency issues of The Grundgesetze.

Hilbert's Problems

- The 1900 Paris Conference of the International Congress of Mathematicians
- Hilbert defines ten problems as a roadmap for mathematics of the 20th century
  - (Later expanded to 23)

Hilbert’s Second Problem

- "When we are engaged in investigating the foundations of a science, **we must set up a system of axioms** which contains an **exact and complete description** of the relations subsisting between the elementary ideas of that science.”

- “But above all I wish to designate the following as **the most important** among the numerous questions which can be asked with regard to the axioms: To **prove that they are not contradictory**, that is, that a definite number of logical steps based upon them can never lead to contradictory results.”

Godel's First Incompleteness Theorem

- Any formal system that is:
  - consistent,
  - effectively axiomatized,
  - strong enough to express elementary arithmetic
- is incomplete.

  There are arithmetical statements that the system can neither prove nor refute.

  This responds to the first part of Hilbert's problem.

Godel's Second Incompleteness Theorem

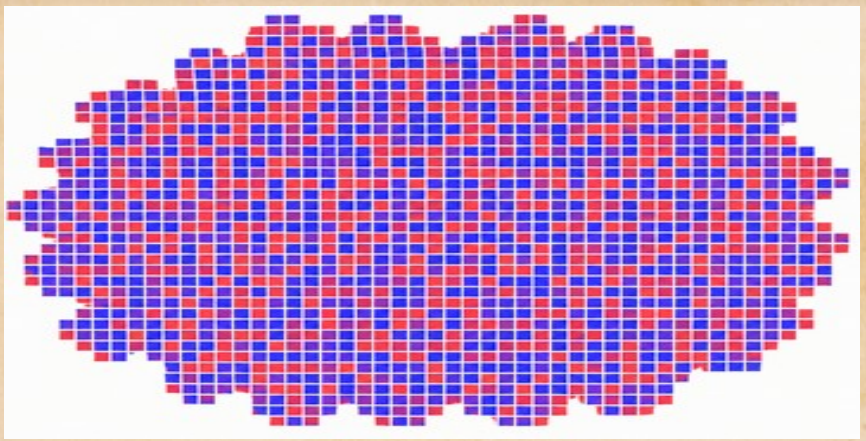
- Any formal system that is:
  - consistent,
  - effectively axiomatized,
  - strong enough to express elementary arithmetic
- **Cannot prove its own consistency.**

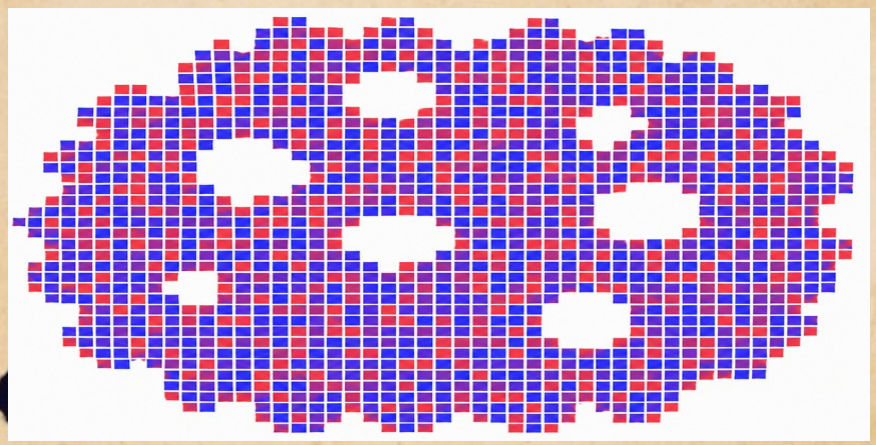
This responds to the second part of Hilbert's problem

The setting for formal logic

- Three components:
- **A mathematical entity of interest**
  - For example: Complex numbers, real-valued functions, vector spaces, etc.
- **A set of “strings”**
  - defined syntax
  - Algorithmically comprehensible
  - “starter” strings (“axioms”)
  - Rules for combining strings to create new ones (“rules of inference”)
- **A mapping between the two**
  - Manipulate the strings, perhaps algorithmically, to reason about the mathematical object

Truth versus provability





Semantics: Mathematical truth and falsehood

- All statements are either true or false

Syntax: string manipulation

- Initial set of ‘axiom’ strings
- New strings via ‘rules of inference’
- Syntactic ‘provability’
- Possible for neither “A” nor “not A”?

The setting for incompleteness

Mathematical entity:

- Natural numbers (non-negative integers)
- Operations: addition, multiplication
- Relations: equality, ordering relation (“n < m”)

Formal language:

- First-order predicate logic
- Peano axioms for arithmetic

Godel's Hall of Mirrors

- The formal language describing natural numbers is encoded into natural numbers

- The machinery of the formal language is used to reason about those numbers that represent the objects of the formal language

Deeper into the hall of mirrors

- The goal:
  - Construct a formula:

    “There exists a formula with Godel number M which possesses the property that the formula has no proof.”

  - Oh by the way:

    “The Godel number of the above formula is M.”

  - The formula would become a witness of the assertion it makes!

Godel numbers

- For intuition, Godel numbers in two steps:
  - Formula: x+5
  - String: “x+5”
  - Number: 89_12_22

  - Map printable ASCII characters into the range [0 .. 99]

Godel numbers

- Map strings into numbers in two ways:

```
>>> eval('111+222')
333

>>> gnum('111+222')
18181812191919
# 18_18_18_12_19_19_19
```

Godel number is huge; much larger than eval version.

Godel numbers

- Godel number was much larger.

  Is the other way around possible??

```
>>> eval('10**12')
1000000000000


>>> gnum(‘10**12’)
181711111819
```

Yes! (depending on the language features allowed)

Godel numbers

Is it possible to find an expression where the two are exactly equal?

Answering this curious question is the key..

Expressiveness of Peano Arithmetic

- Unproven claim: Peano arithmetic is powerful enough to do primitive recursion.

  I.e., algorithms that have simple “for loops”

  PA can be used to “code” numeric operations on formulas-as-numbers

Arithmetic on formulas

- Use Peano arithmetic to encode formula manipulation:
- Subst(N, M):
  - Subst returns a number Z
    - N is interpreted as the Godel number of formula F_x with a free variable x
    - Substitute M for all instances of ‘x’ in F_x
    - Z is the Godel number of the resulting formula

Operation of Substitute

>>> 'x+5'

Its Godel number: 89_12_22

>>> '6'

Its Godel number: 23

>>> Substitute(89_12_22, 23)

23_12_22

Corresponding string:

'6+5'

Lambda calculus

- A functional notation invented by Church in the 1930's
  - Basis of programming language Lisp
  - A functional sub-language in many modern programming languages (python, C#, C++, etc.)

```
>>> lambda x: x*x
>>> lambda x: x + 47
>>> (lambda x: x*x)  (12)
144
```

Function evaluation with lambda

```
>>> (lambda x: x*x) (6)
```

- To evaluate:
  - Take the body of the lambda term
  - Splice the argument into the body
  - Evaluate the result:

```
(lambda x: x*x + 10) (6)
            x*x + 10
            6*6 + 10
46
```

Applying a function to itself

- A lambda function for self-application:

  >>> lambda f: f(f)

  Self-apply a function that takes **anything**, and returns 47

  >>> (lambda f: f(f)) (lambda n: 47)

  47

  >>> identity_fn = lambda x:x

  >>> f = (lambda f: f(f)) (identity_fn)

  f is itself, the identity function..

Down the rabbit hole

- What happens if we apply self_app to itself??

```
>>> (lambda x: x(x))    ( lambda x: x(x) )
                x(x)
              x          (      x      )
            /    \            /   \
        ------------     ------------
    (lambda x: x(x))  ( (lambda x: x(x)) )
```

We get the exact original back. Evaluate that…

We get the exact original back. Evaluate that…

We get the exact original back. Evaluate that…

We get the exact original back. Evaluate that…

……

Self-application in arithmetic

- Apply the standard lambda calculus divergent expression to Godel numbers:

  Subst(x, x)

  This is a formula with one free variable.

  Its Godel number:

  52_86_67_84_85_09_89_13_89_10

  (I.e., 528667848509891389 10)

  “Subst(x, x)” is suggestive of “lamba x: x(x)”

Self-application in arithmetic

As suggested by the lambda term

(lambda x: x (x)) (lambda x: x (x))

- Replace ‘x’ by the Godel number of our term:

Subst(“Subst(x,x)”, “Subst(x,x)”)

Self-application in arithmetic

Consider “Subst(Subst(x,x), Subst(x, x))”

Do the arithmetic:

In first argument“Subst(x,x)”

replace “x” by the second argument “Subst(x, x)”

“Subst(Subst(x, x), Subst(x, x))”

The Y Combinator

An arithmetic formula that evaluates to its own Godel number!

- A final addition to conclude the argument:

lambda x: **F**(x x)

The Y Combinator

Applying this term to itself as before:

(lambda x: **F**(x x)) (lambda x: **F**(x x))

Evaluates to: **F** ((lambda x: **F**(x x)) (lambda x: **F**(x x)))

Evaluates to: **F**(**F** ((lambda x: **F**(x x)) (lambda x: **F**(x x))))

Evaluates to: **F**(**F**(**F** ((lambda x: **F**(x x)) (lambda x: **F**(x x)))))

….

F(**F**(**F**(...**F**((lambda x. **F**(x x)) (lambda x. **F**(x x)))…))))

….

Infinite tree of function applications

-
-

The Fixed Point Lemma: An intuitive look

Let “T” be the tree rooted at the topmost node

Consider the expression “f(T)”.

This is an exact copy of the original tree “T”!

For any formula F with a single free variable, it is possible to build a formula T such that

T ←→ F(godel_num(T))

The Godel formula

"There exists a formula with Godel number M which possesses the property that the formula has no proof."

Replace M by "Subst(x,x)"

"There exists a formula with Godel number Subst(x,x) which possesses the property that the formula has no proof."

Let us abbreviate the above as:

F(Subst(x,x))

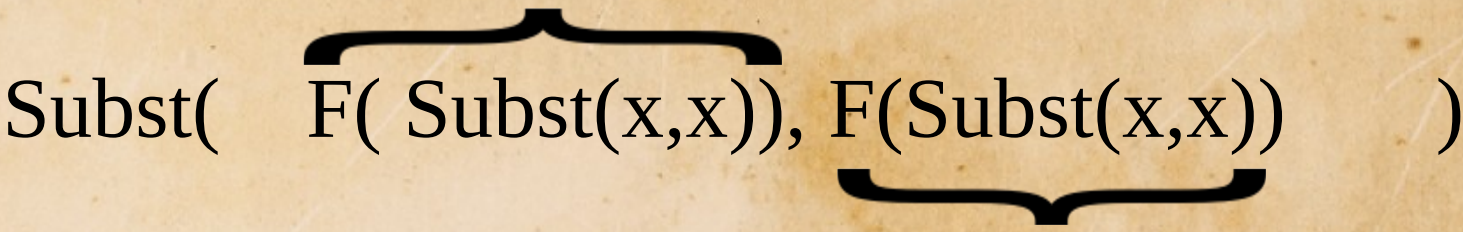
The Godel formula

- Starting with F(Subst(x,x))
- As before, replace “x” with a copy of the above:

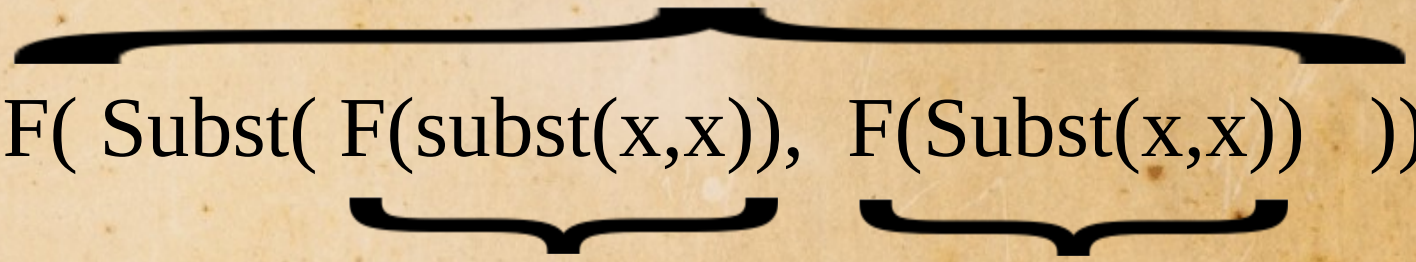
  F(Subst(F(Subst(x,x)), F(subst(y,y))))

The Godel formula

Now, start with the inner expression, which is our “M” in the abbreviation F(M):



Apply the outer Subst operation:



The Godel Formula

F(Subst(F(subst(x,x)), F(Subst(x,x))))

The magic has happened! The inner M evaluates to the entire outer expression.

Our original expression: F(M)

The inner expression: M

The evaluation of the inner expression: F(M)

The Godel formula

- So, the original goal has been achieved.

  We have a formula with an embedded expression M:

  > “There exists a formula with Godel number M which possesses the property that the formula has no proof.”

  We were able to create the required expression M which evaluates to a copy of the entire formula.

Conclusion

- Kurt Godel listened carefully to philosophical arguments
- He concluded that they were deficient for capturing the grandeur of mind and existence
- Using the power and rigor of mathematics he was able to make a profound philosophical counter-argument