CS251

Great Ideas in *Theoretical* Computer Science

Limits of Human Reasoning

Last Time

Diagonalize against **the set of all decidable languages:**

$\overline{\text{SELF-ACCEPTS}}$ ($\overline{\text{SA}}$) is undecidable.

$$\overline{\text{SELF-ACCEPTS}} \leq \text{SELF-ACCEPTS} \leq \text{ACCEPTS}$$

$$\leq \text{HALTS} \leq \text{SAT} \leq \text{NEQ}$$

Undecidable problems
not involving Turing Machines

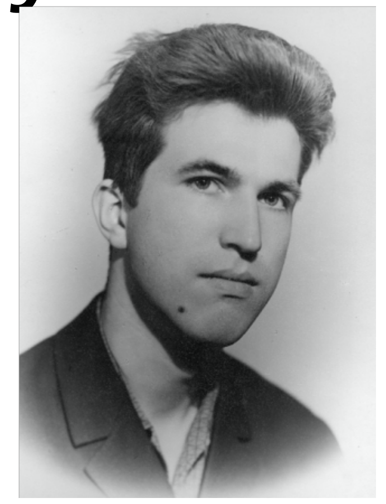
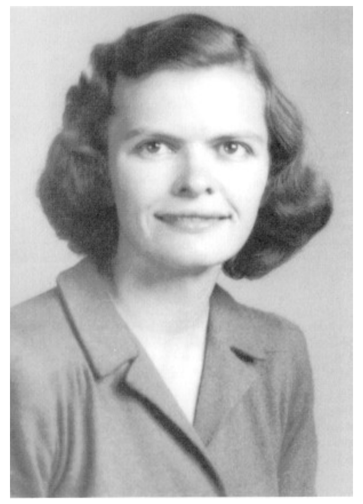
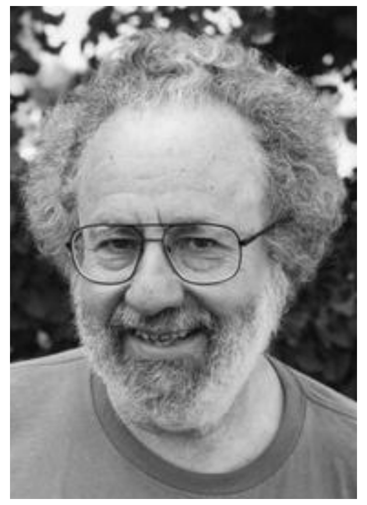
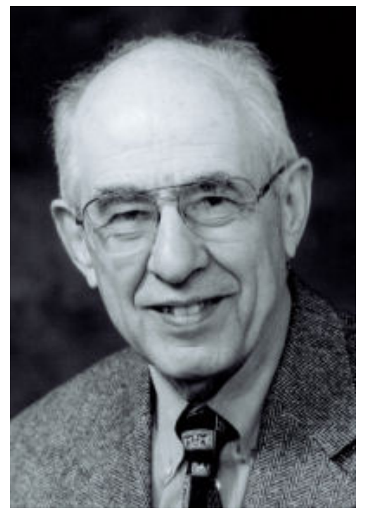
Hilbert's 10th Problem

Determining if a given multivariate polynomial with integral coefficients has an integer root.

e.g. $5xy^2z + 8yz^3 + 100x^{99}$

Undecidable!

Proved in 1970 by Matiyasevich-Robinson-Davis-Putnam.

Does it have a ***real*** root? **Decidable!**

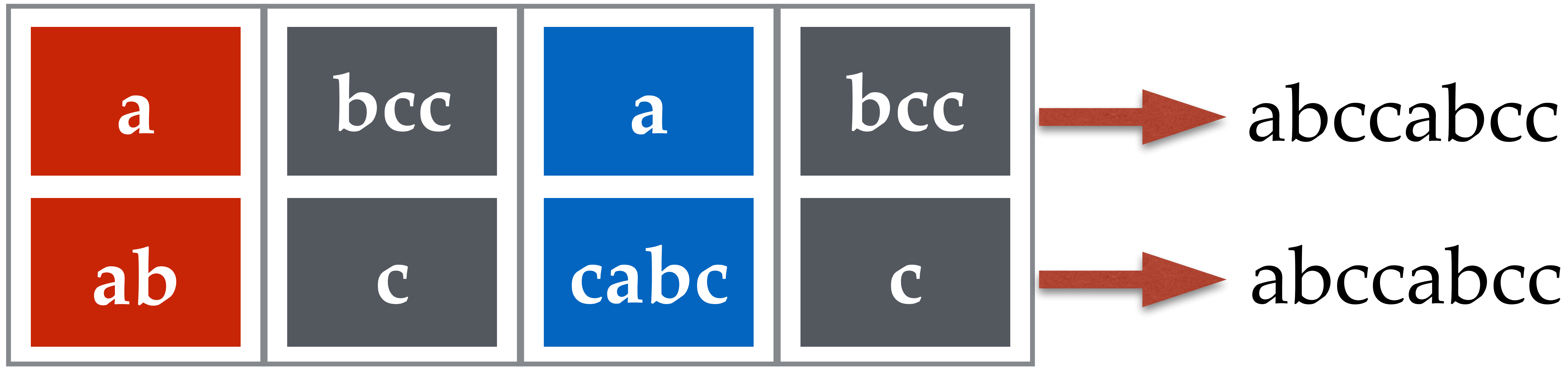
Proved in 1951 by Tarski.

Does it have a ***rational*** root? **No one knows!**

Post's Correspondence Problem (PCP)

Input: A finite collection of "dominoes" having strings written on each half.

Output: Accept if it is possible to match the strings.



Undecidable!

Proved in 1946 by Post.

Wang Tiles

Input: A finite collection of "Wang Tiles" (squares)
with colors on the edges.

Output: Accept iff it is possible to make an infinite grid
from copies of the given squares,
where touching sides must color-match.

Undecidable!

Proved in 1966 by Berger.

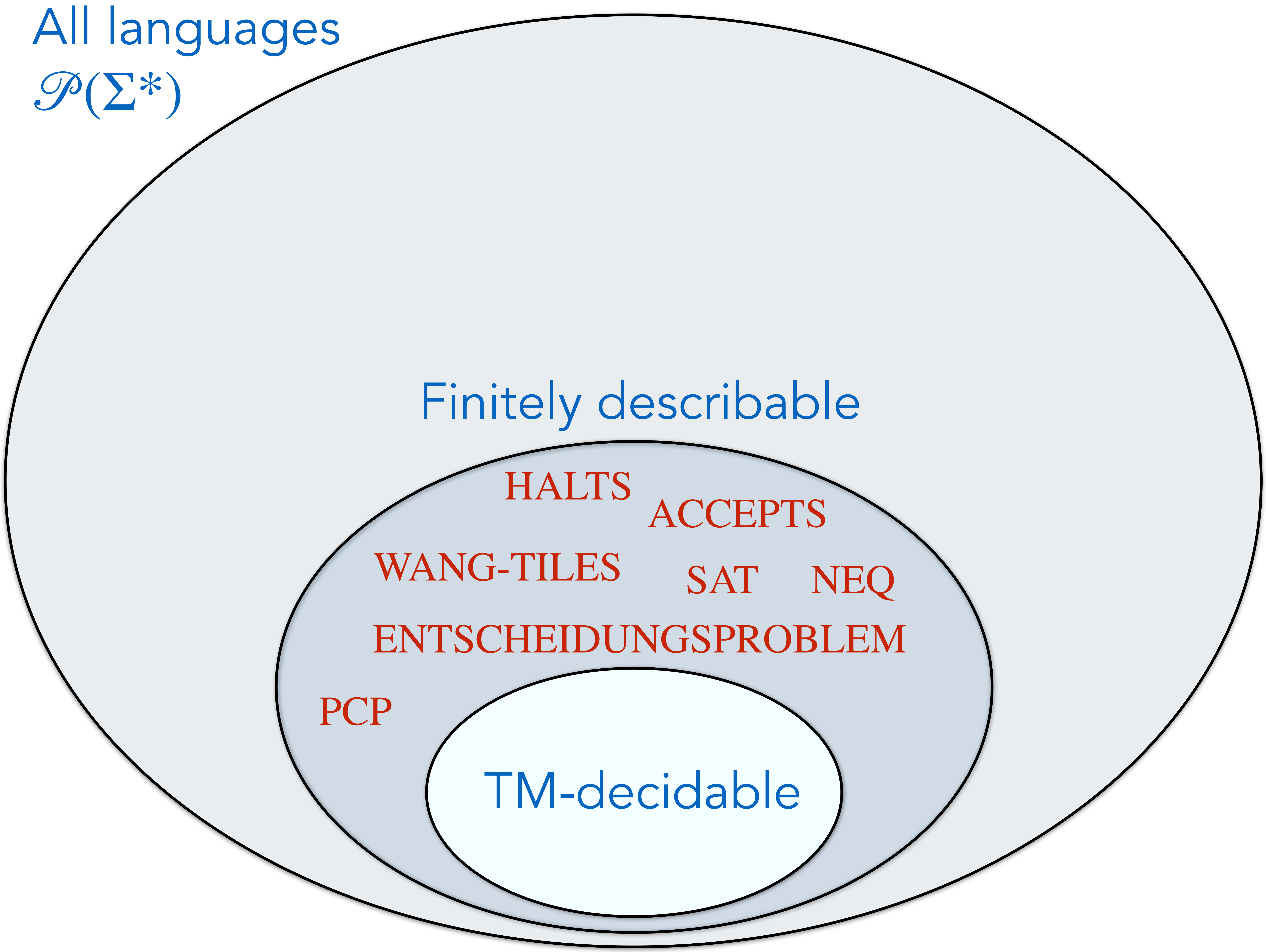
Mortal Matrices

Input: Two 21x21 matrices of integers U and V.

Output: Accept iff it is possible to multiply U and V
(multiple times in any order)
to get to the 0 matrix.

Undecidable!

Proved in 2007 by Halava, Harju, Hirvensalo.


All languages
$\mathscr{P}(\Sigma^*)$
Finitely describable
HALTS
ACCEPTS
WANG-TILES
SAT
NEQ
ENTSCHEIDUNGSPROBLEM
PCP
TM-decidable

Completed:

Formally define computation.

Understand the limits of computation.

Next:

Understand (logical) human reasoning and its limits.

Good Old Regular Mathematics
(GORM)

GORM: Good Old Regular Mathematics

Real World

Something of interest

Applications

Abstract World

Mathematical Model

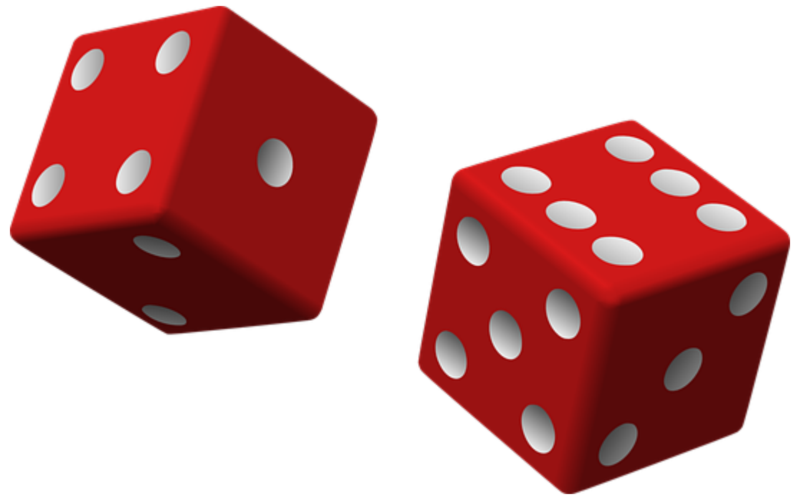
logic

New Knowledge

GORM: Good Old Regular Mathematics

Real World

Gambling



Abstract World

Probability Theory

New Knowledge

Applications

GORM: Good Old Regular Mathematics

Real World

Farming

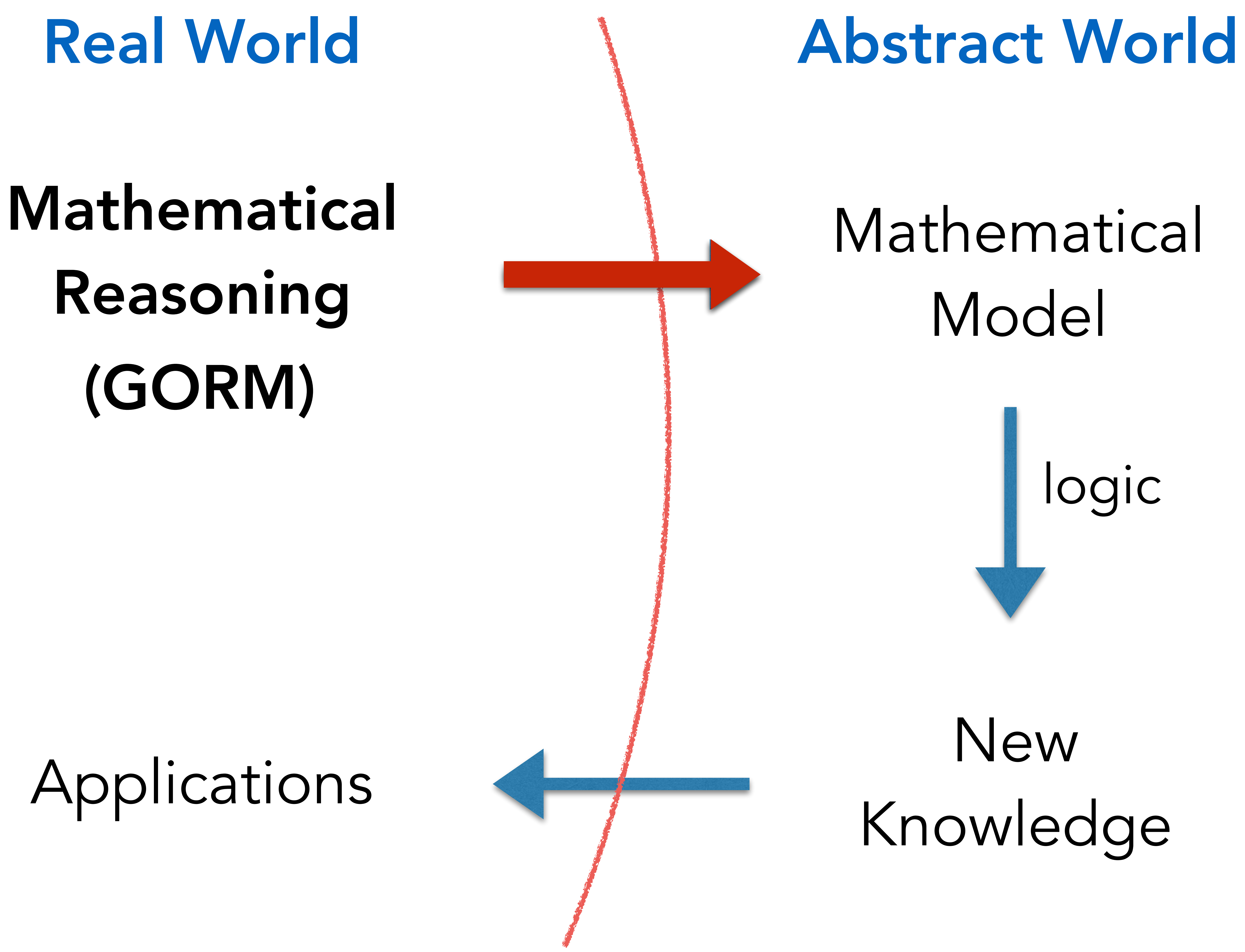
Applications

Abstract World

Plane Geometry

New Knowledge

FORM: Formalization of GORM



FORM: Formalization of GORM

Real World

Mathematical Statement

Axiom

Deduction Rule

Proof

Truth

Abstract World

Mathematical
Model

Elements of Mathematical Reasoning (Informal)

Statement: A well-formed sentence with a truth value.

Axiom: An obviously true statement.

Deduction Rule: A rule allowing you to derive new true statements from other true statements.

Proof: A chain of deductions, starting from axioms, and ending at the statement.

knowledge ≡ proof

Hope: **truth** ≡ provable

Part 1:

Essential components of FORM

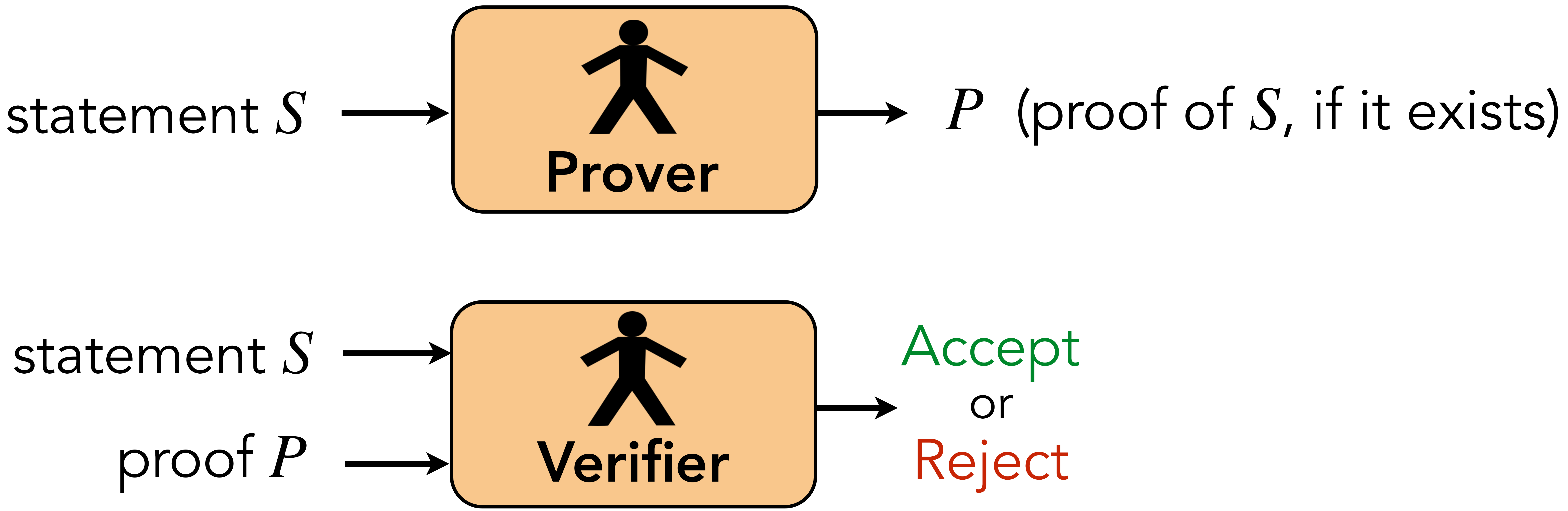
Part 2:

Proving interesting properties of FORM
(and therefore GORM)

Part 1:

Essential components of FORM

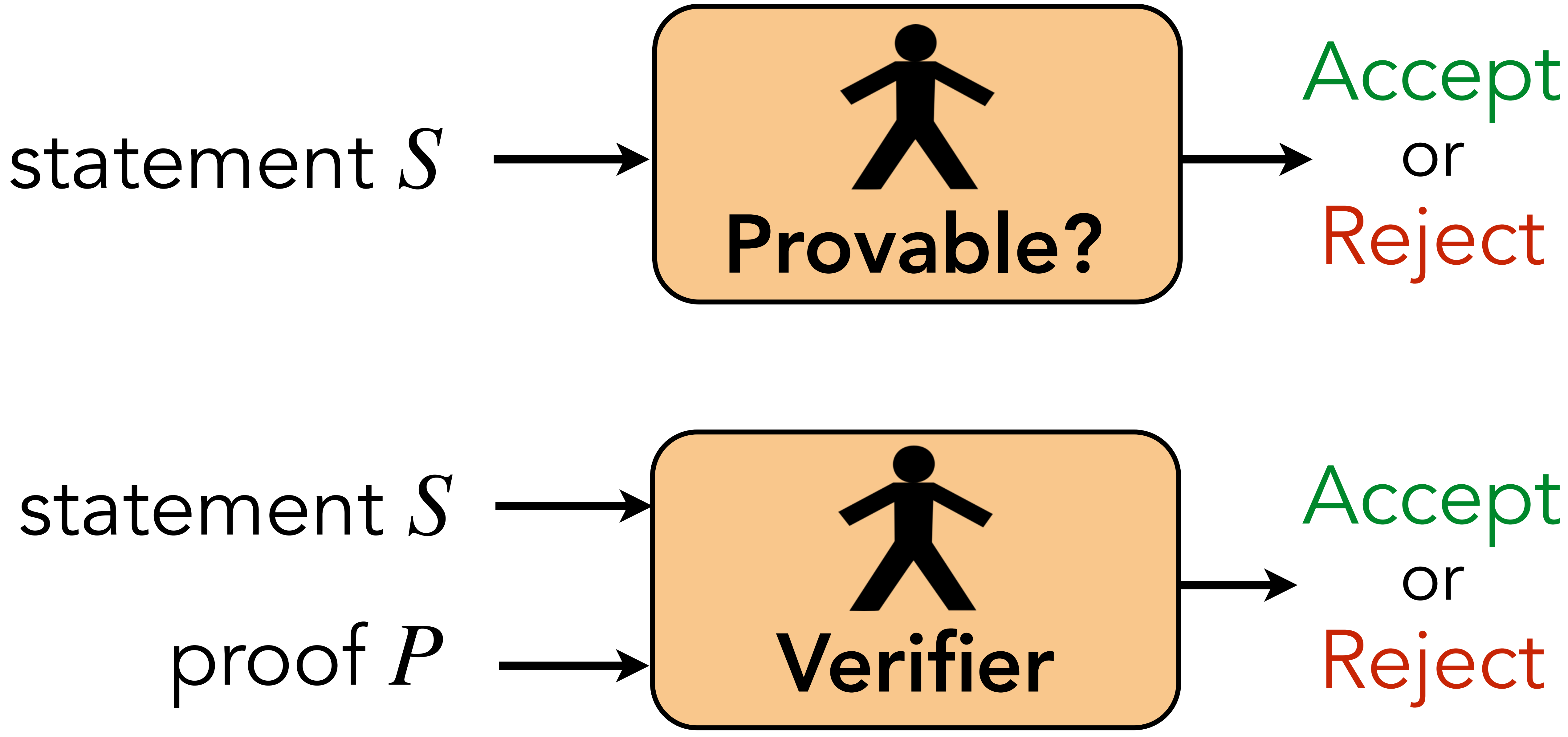
GORM is a computational process



N Statements and proofs are represented as **finite-length strings**.

N Want verification to be (efficiently) decidable.

GORM is a computational process



N Statements and proofs are represented as **finite-length strings**.

N Want verification to be (efficiently) decidable.

FORM: Formalization of GORM

FORM is a mathematical model (formalization) of GORM such that:

- For every statement S in GORM with a truth value,
  there is a precise representation of S in FORM (denoted by $\langle S \rangle$).

- For every argument P in GORM,
  there is a precise representation of P in FORM (denoted by $\langle P \rangle$).

- FORM specifies a decider TM V (called a *verifier*) such that
  $V(\langle S \rangle, \langle P \rangle)$ accepts iff P is a proof of S.

S is **provable** means $\exists w \in \Sigma^*$ such that $V(\langle S \rangle, w)$ accepts.

From Verifier to Prover

Let V be the verifier. We can build a prover from it:

def Prover($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** w

def isProvable($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

Is there an agreed upon FORM that we can work with?

The Church-Turing Thesis of Mathematics

GORM-to-ZFC Thesis

Church-Turing Thesis (GORA-to-TM Thesis):

TM is the right model for an algorithm.

Every algorithm compiles down to a TM.

GORM-to-ZFC Thesis:

The *Zermelo–Fraenkel-Choice (ZFC) axiomatic system* is the right model for GORM.

Every GORM-statement compiles down to a ZFC-statement.

Every GORM-proof compiles down to a ZFC-proof.

Part 1:

Essential components of FORM

Part 2:

Proving interesting properties of FORM
(and therefore GORM)

Part 2:

Proving interesting properties of FORM (and therefore GORM)

Properties you want from FORM

1. Consistency

For every statement S, **at most** one of S or $\neg S$ is provable.

Not consistent $\implies$ Any statement is provable.

Want to prove T.
AFSOC $\neg T$.
Derive (prove) S.
Derive (prove) $\neg S$.
Contradiction. $\blacksquare$

2. Soundness

If S is provable, then S is true.

(false statements are not provable)

Soundness $\implies$ Consistency.

3. Completeness

For every statement S, **at least** one of S or $\neg S$ is provable.

Properties you want from an axiomatic system

1. **Consistency**

   For every statement S, **at most** one of S or $\neg S$ is provable.

2. **Soundness**

   If S is provable, then S is true.

   (false statements are not provable)

3. **Completeness**

   For every statement S, **at least** one of S or $\neg S$ is provable.

N **Sound** & **Complete** $\Longrightarrow$ **truth** $\equiv$ **provable**.

Hilbert's Program

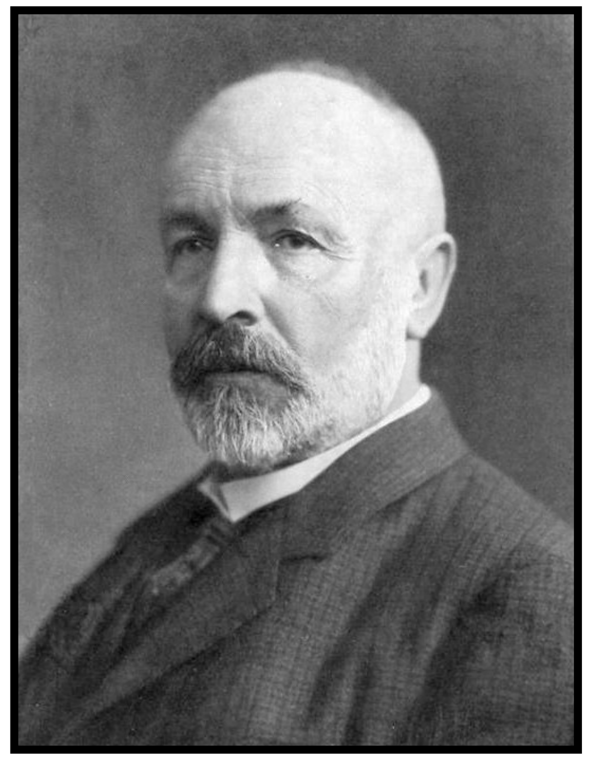
- Formalize GORM. (Create FORM)

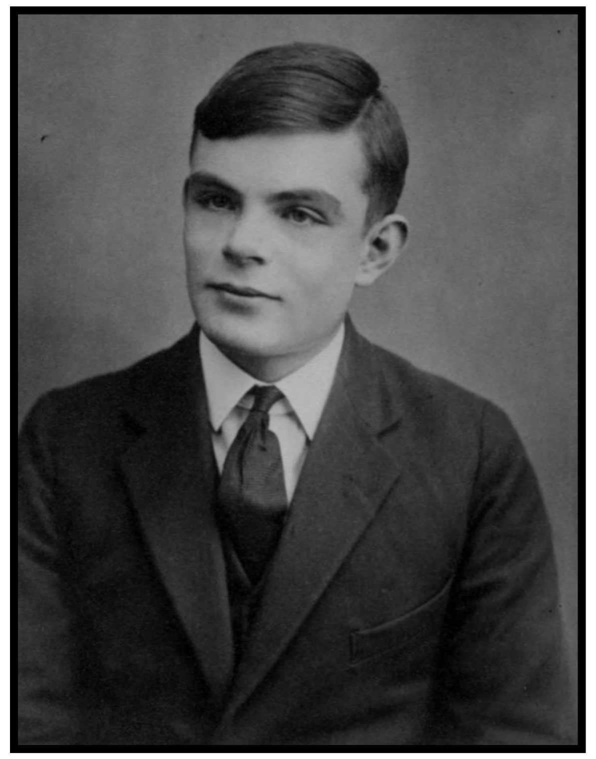
- Prove FORM is **complete**.

- Prove FORM is **consistent**.

Puzzle: Why not prove **soundness**?

? Is Hilbert's Program achievable?





finite
vs
infinite → limits of computation → limits of human reasoning

Contrapositive:

"Satisfactory" FORM

→ Compute the uncomputable

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

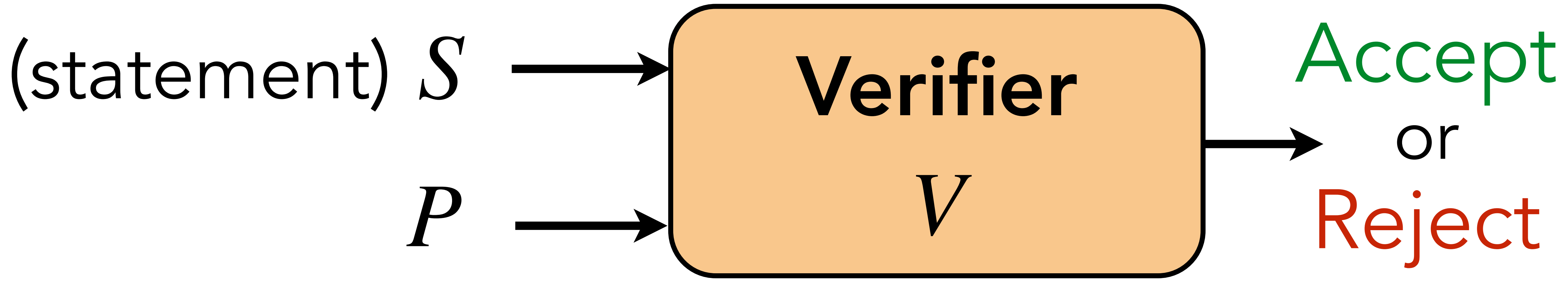
2nd Incompleteness Theorem

The Setting

The FORM we are using: **ZFC axiomatic system**.

GORM-to-ZFC Thesis:

For every GORM-statement and GORM-proof,
there is a corresponding ZFC-statement and ZFC-proof.



(is P a proof of S?)

Terminology:

S is **provable**: there is a proof of S in ZFC

S is **independent**: Neither S nor $\neg S$ is provable

Incompleteness: $\exists$ an independent S

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

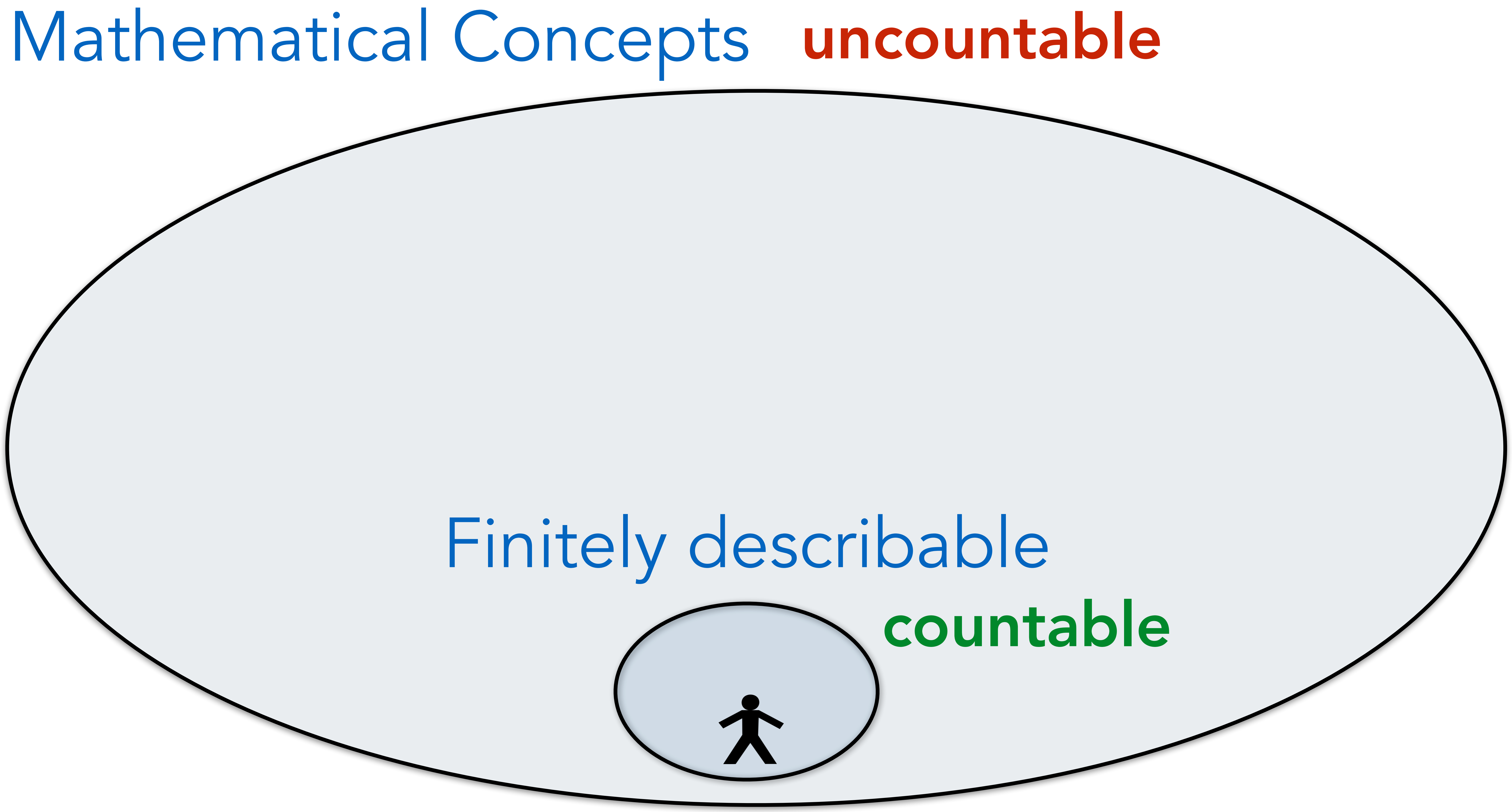
1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

2nd Incompleteness Theorem

0th Incompleteness Theorem

The 0th Incompleteness Theorem



Observation: We can't hope to reason about everything!

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

2nd Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #1)

? Can ZFC be both **sound** & **complete**?

Assume it is. Then **truth** ≡ **provable**.

We can then compute/decide *any* truth!!!

def Resolve($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

1st Incompleteness Theorem (Soundness version #1)

? Can ZFC be both **sound** & **complete**?

Assume it is. Then **truth** ≡ **provable**.

We can then compute/decide *any* truth!!!

def isTrue($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\text{HALTS}}(\langle M, x \rangle)$:
  return isTrue($\langle$"$M(x)$ halts"$\rangle$)

1st Incompleteness Theorem (Soundness version #1)

Can ZFC be both **sound** & **complete**?

Assume it is. Then **truth** ≡ **provable**.

We can then compute/decide *any* truth!!!

def isTrue($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\mathrm{SAT}}(\langle M \rangle)$:
  return isTrue($\langle$"$\exists x$ such that $M(x)$ accepts"$\rangle$)

1st Incompleteness Theorem (Soundness version #1)

? Can ZFC be both **sound** & **complete**?

Assume it is. Then **truth** ≡ **provable**.

We can then compute/decide *any* truth!!!

def isTrue($\langle S \rangle$):
    for $k = 1, 2, 3, \ldots$
        for every string w of length k:
            if $V(\langle S \rangle, w)$ **accepts**: **return** True
            if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\overline{\mathrm{SA}}}(\langle M \rangle)$: **# complement of SELF-ACCEPTS**
    return isTrue($\langle$"M does not self-accept"$\rangle$)

1st Incompleteness Theorem (Soundness version #1)

def Resolve($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\overline{\mathrm{SA}}}(\langle M \rangle)$:
  return Resolve($\langle$"M does not self-accept"$\rangle$)

Theorem:

ZFC cannot be both sound and complete.

I.e. if ZFC is sound, then it is **incomplete**.

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

2nd Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Soundness version #2)

? Assume ZFC is sound. What is an explicit statement S independent of ZFC?

```
def Resolve(⟨S⟩):
    for k = 1, 2, 3, ...
        for every string w of length k:
            if V(⟨S⟩, w) accepts:  return True
            if V(⟨¬S⟩, w) accepts:  return False
```

```
def M_SA‾(⟨M⟩) :
    return Resolve(⟨"M does not self-accept"⟩)
```

Observations:

- $M_{\overline{\text{SA}}}$ is not a correct decider for $\overline{\text{SA}}$.
- So for some input (for some M), $M_{\overline{\text{SA}}}$ doesn't give the right answer.
- So $\exists M$ such that Resolve(⟨"M does not self-accept"⟩) doesn't give the right answer.
- So $\exists M$ such that "M does not self-accept" is independent!

1st Incompleteness Theorem (Soundness version #2)

? Assume ZFC is sound. What is an explicit statement S independent of ZFC?

def Resolve($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\overline{\mathrm{SA}}}(\langle M \rangle)$:
  return Resolve($\langle$"M does not self-accept"$\rangle$)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
M_1	$\boxed{0}$	∞	1	$\cdots$
M_2	1	$\boxed{1}$	1	$\cdots$
M_3	1	0	$\boxed{\infty}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$
$\overline{\mathrm{SA}}$	1	0	1	$\cdots$

1st Incompleteness Theorem (Soundness version #2)

? Assume ZFC is sound. What is an explicit statement S independent of ZFC?

```
def Resolve(⟨S⟩):
    for k = 1, 2, 3, ...
        for every string w of length k:
            if V(⟨S⟩, w) accepts:  return True
            if V(⟨¬S⟩, w) accepts:  return False
```

```
def M_SA‾(⟨M⟩) :
    return Resolve(⟨"M does not self-accept"⟩)
```

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
M_1	$\boxed{0}$	∞	1	$\cdots$
M_2	1	$\boxed{1}$	1	$\cdots$
M_3	1	0	$\boxed{\infty}$	$\cdots$
				$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\overline{\mathrm{SA}}$	1	0	1	$\cdots$

1st Incompleteness Theorem (Soundness version #2)

? Assume ZFC is sound. What is an explicit statement S independent of ZFC?

```
def Resolve(⟨S⟩):
    for k = 1, 2, 3, ...
        for every string w of length k:
            if V(⟨S⟩, w) accepts:  return True
            if V(⟨¬S⟩, w) accepts:  return False
```

```
def M_SA-bar(⟨M⟩) :
    return Resolve(⟨"M does not self-accept"⟩)
```

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_{\overline{SA}}\rangle$	$\cdots$
M_1	$\boxed{0}$	∞	1	0	$\cdots$
M_2	1	$\boxed{1}$	1	∞	$\cdots$
M_3	1	0	$\boxed{\infty}$	0	$\cdots$
$M_{\overline{SA}}$	1	0	1	$\boxed{\infty}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\overline{SA}$	1	0	1		$\cdots$

Conclusions:

- $M_{\overline{SA}}$ does not give the right answer when the input is $\langle M_{\overline{SA}}\rangle$.
- $I =$ "$M_{\overline{SA}}$ does not self-accept" is independent of ZFC.

1st Incompleteness Theorem (Soundness version #2)

def Resolve($\langle S \rangle$):
    for $k = 1, 2, 3, \ldots$
        for every string w of length k:
            if $V(\langle S \rangle, w)$ **accepts**: **return** True
            if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\overline{\mathrm{SA}}}(\langle M \rangle)$:
    return Resolve($\langle$"M does not self-accept"$\rangle$)

<u>**Definition:**</u> I = "$M_{\overline{\mathrm{SA}}}$ does not self-accept"

<u>**Theorem:**</u> If ZFC is sound,
"$M_{\overline{\mathrm{SA}}}(\langle M_{\overline{\mathrm{SA}}} \rangle)$ does not accept" is **independent** of ZFC.

1st Incompleteness Theorem (Soundness version #2)

```
def Resolve(⟨S⟩):
    for k = 1, 2, 3, ...
        for every string w of length k:
            if V(⟨S⟩, w) accepts: return True
            if V(⟨¬S⟩, w) accepts: return False
```

```
def M_{\overline{SA}}(⟨M⟩) :
    return Resolve(⟨"M does not self-accept"⟩)
```

Definition: I = "$M_{\overline{\mathrm{SA}}}$ does not self-accept"

Theorem: ZFC is sound $\rightarrow$ I is **independent** of ZFC.

Theorem: ZFC is sound $\rightarrow$ I.

? Can we replace "ZFC sound" with "ZFC consistent"?

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

2nd Incompleteness Theorem

1st Incompleteness Theorem (Consistency version)

1st Incompleteness Theorem (Soundness version #2)

def Resolve($\langle S \rangle$):
  for $k = 1, 2, 3, \ldots$
    for every string w of length k:
      if $V(\langle S \rangle, w)$ **accepts**: **return** True
      if $V(\langle \neg S \rangle, w)$ **accepts**: **return** False

def $M_{\overline{\mathrm{SA}}}(\langle M \rangle)$:
  return Resolve($\langle$"M does not self-accept"$\rangle$)

Definition: I = "$M_{\overline{\mathrm{SA}}}$ does not self-accept"

Theorem: ZFC is consistent $\rightarrow$ I is **independent** of ZFC.

Theorem: ZFC is consistent $\rightarrow$ I.

The Plan

0th Incompleteness Theorem

1st Incompleteness Theorem (Soundness version #1)

1st Incompleteness Theorem (Soundness version #2)

1st Incompleteness Theorem (Consistency version)

2nd Incompleteness Theorem

2nd Incompleteness Theorem

Reductions for provability

Reductions for provability

Proving S **reduces** to proving T means: "$T \to S$" is provable.

If S **reduces** to T then: T provable $\Longrightarrow$ S provable

S unprovable $\Longrightarrow$ T unprovable

! Can use reductions to expand the landscape of unprovability.

2nd Incompleteness Theorem

Proving S **reduces** to proving T means: "$T \rightarrow S$" is provable.

If S **reduces** to T then: T provable $\Longrightarrow$ S provable

S unprovable $\Longrightarrow$ T unprovable

I = "$M_{\overline{\mathrm{SA}}}$ does not self-accept" is unprovable.

So any T, such that I reduces to T, is unprovable.

Theorem: ZFC is consistent $\rightarrow$ I.

By GORM-to-ZFC Thesis

$\exists$ GORM-proof of theorem $\longrightarrow$ $\exists$ ZFC-proof of theorem

2nd Incompleteness Theorem

Theorem:

If ZFC is consistent, "ZFC is consistent" is not provable.


WTF
WHAT JUST HAPPENED?!?!
memegenerator.net

Hilbert's Program

- Formalize GORM. (Create FORM)

- Prove FORM is **complete**.

- Prove FORM is **consistent**.


ARENA
UFC
UFC
UFC

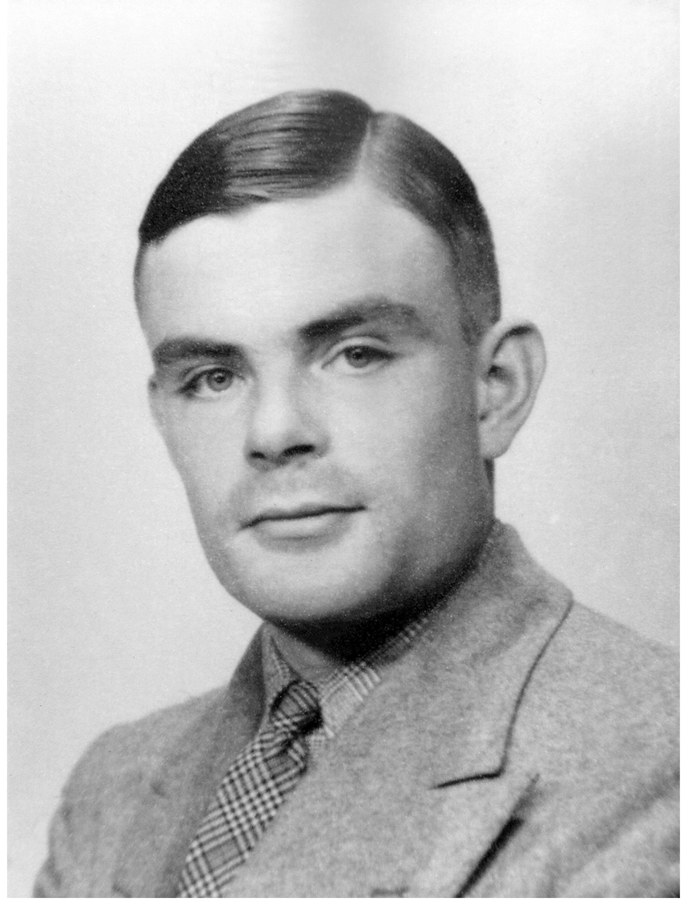
Hilbert's Program

- Formalize GORM. (Create FORM)

- Prove FORM is **complete**.

- Prove FORM is **consistent**.

- Prove isProvable is **decidable**.



isProvable is undecidable!

That's all for PART 1 of CS251

- Foundations of math.

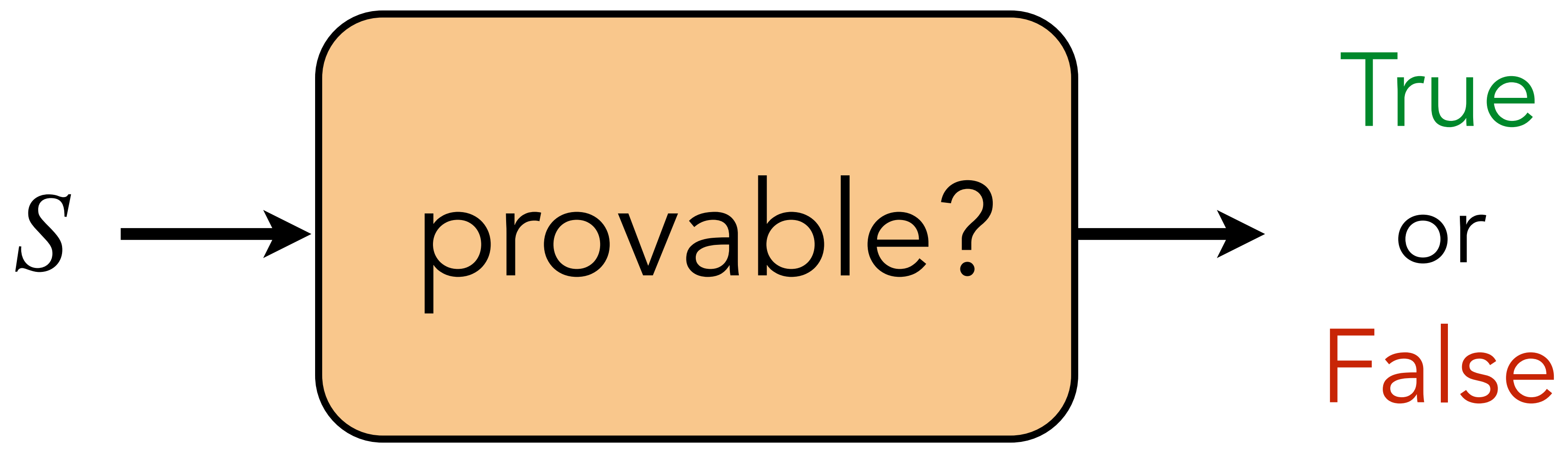
- Birth of computer science.

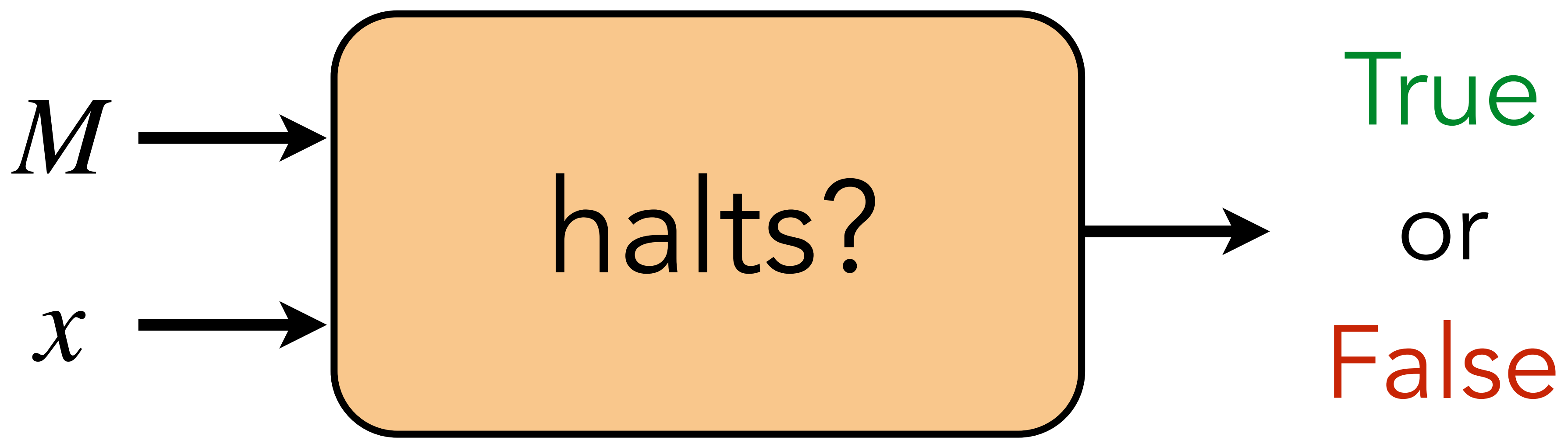
- Formalizing computation.

- Uncountability → Uncomputability → Unprovability.

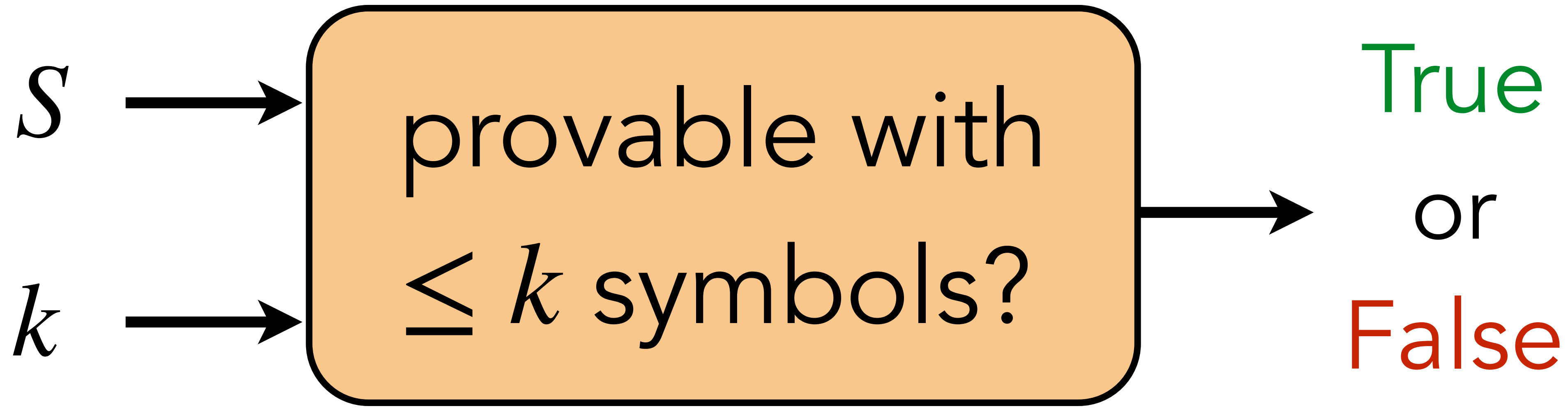
What is next?

Yes, the following (and many more) are uncomputable:





What about:



These are computable! But are they **practically computable**?

What is next?

We enter the land of **complexity theory** and the famous "**P** vs **NP**".