NTNU
Norwegian University of
Science and Technology

# Axioms, algorithms and Hilbert's Entscheidungsproblem

Jan Stovicek
Department of Mathematical Sciences

September 9th, 2008

# Outline

# Outline

The Decision Problem

Formal Languages and Theories

Incompleteness

Undecidability

# The Decision Problem

**Problem (Hilbert's Entscheidungsproblem, 1928)**

*Is there an effective procedure (an algorithm) which, given a set of axioms*

# The Decision Problem

**Problem (Hilbert's Entscheidungsproblem, 1928)**

*Is there an effective procedure (an algorithm) which, given a set of axioms and a mathematical proposition,*

# The Decision Problem

**Problem (Hilbert's Entscheidungsproblem, 1928)**

*Is there an effective procedure (an algorithm) which, given a set of axioms and a mathematical proposition, decides whether it is or is not provable from the axioms?*

# The Decision Problem

**Problem (Hilbert's Entscheidungsproblem, 1928)**

*Is there an effective procedure (an algorithm) which, given a set of axioms and a mathematical proposition, decides whether it is or is not provable from the axioms?*

*From: David Hilbert and Wilhelm Ackermann,
Foundations of Theoretical Logic (Grundzüge der theoretischen Logik), 1928.*

NTNU
Norwegian University of
Science and Technology

# The Idea Behind

If we have a reasonable mathematical structure, such as the arithmetics on natural numbers, then:

# The Idea Behind

If we have a reasonable mathematical structure, such as the arithmetics on natural numbers, then:

1. Give a complete axiomatic description;

# The Idea Behind

If we have a reasonable mathematical structure, such as the arithmetics on natural numbers, then:

1. Give a complete axiomatic description;
2. Use the decision procedure (algorithm) to prove or disprove mathematical statements mechanically.

# The Idea Behind

If we have a reasonable mathematical structure, such as the arithmetics on natural numbers, then:

1. Give a complete axiomatic description;
2. Use the decision procedure (algorithm) to prove or disprove mathematical statements mechanically.

Hilbert & Ackermann:

> *We want to make it clear that for the solution of the decision problem a process would be given . . . , even though the difficulties of the process would make practical use illusory . . .*

NTNU
Norwegian University of Science and Technology

# Obstacles to Automated Proving

— Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.

# Obstacles to Automated Proving

— Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.

— Undecidability: There are problems which cannot be solved by any algorithm.

# Obstacles to Automated Proving

- Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.
- Undecidability: There are problems which cannot be solved by any algorithm.
- Inefficiency: Even if we have an algorithm, it may be far too slow.

# Obstacles to Automated Proving

- — Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.
- — Undecidability: There are problems which cannot be solved by any algorithm.
- — Inefficiency: Even if we have an algorithm, it may be far too slow.

*Remarks.*

1. The important results on incompleteness and undecidability come from 1930's

NTNU
Norwegian University of Science and Technology

# Obstacles to Automated Proving

- — Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.
- — Undecidability: There are problems which cannot be solved by any algorithm.
- — Inefficiency: Even if we have an algorithm, it may be far too slow.

*Remarks.*

1. The important results on incompleteness and undecidability come from 1930's — well before the first real computers were constructed!

NTNU
Norwegian University of Science and Technology

# Obstacles to Automated Proving

— Incompleteness: For some very basic mathematical structures, there is no reasonable complete description.

— Undecidability: There are problems which cannot be solved by any algorithm.

— Inefficiency: Even if we have an algorithm, it may be far too slow.

*Remarks.*

1. The important results on incompleteness and undecidability come from 1930's — well before the first real computers were constructed!
2. Despite the problems, there are computer programs designed for automated proving.

# Outline

The Decision Problem

Formal Languages and Theories

Incompleteness

Undecidability

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) $\text{not}\,(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

NTNU
Norwegian University of Science and Technology

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) $\text{not}\,(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

NTNU
Norwegian University of Science and Technology

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

(P5) $x \cdot 0 = 0$;

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x+1=0)$;

(P2) $x+1=y+1 \implies x=y$;

(P3) $x+0=x$;

(P4) $x+(y+1)=(x+y)+1$;

(P5) $x \cdot 0=0$;

(P6) $x \cdot (y+1)=x \cdot y+x$;

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

(P5) $x \cdot 0 = 0$;

(P6) $x \cdot (y + 1) = x \cdot y + x$;

(P7) for any formula $\varphi(x)$ in Peano Arithmetics, we have an axiom

NTNU
Norwegian University of Science and Technology

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

(P5) $x \cdot 0 = 0$;

(P6) $x \cdot (y + 1) = x \cdot y + x$;

(P7) for any formula $\varphi(x)$ in Peano Arithmetics, we have an axiom $(\varphi(0)$ and $(\forall x)(\varphi(x) \Longrightarrow \varphi(x + 1))) \implies (\forall x)(\varphi(x))$.

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

(P5) $x \cdot 0 = 0$;

(P6) $x \cdot (y + 1) = x \cdot y + x$;

(P7) for any formula $\varphi(x)$ in Peano Arithmetics, we have an axiom $(\varphi(0)$ and $(\forall x)(\varphi(x) \implies \varphi(x + 1))) \implies (\forall x)(\varphi(x))$.

NTNU
Norwegian University of Science and Technology

# Example: Peano Arithmetic

An attempt to axiomatically describe natural numbers and their arithmetics.

Axioms:

(P1) not $(\exists x)(x + 1 = 0)$;

(P2) $x + 1 = y + 1 \implies x = y$;

(P3) $x + 0 = x$;

(P4) $x + (y + 1) = (x + y) + 1$;

(P5) $x \cdot 0 = 0$;

(P6) $x \cdot (y + 1) = x \cdot y + x$;

(P7) for any formula $\varphi(x)$ in Peano Arithmetics, we have an axiom $(\varphi(0)$ and $(\forall x)(\varphi(x) \implies \varphi(x + 1))) \implies (\forall x)(\varphi(x))$.

# Formal Languages

We have the following components:

# Formal Languages

We have the following components:

— Variables: $x, y, z, \ldots$

# Formal Languages

We have the following components:

— Variables: $x, y, z, \ldots$

— Constants: $0, 1$

# Formal Languages

We have the following components:

— Variables: $x, y, z, \dots$

— Constants: $0, 1$

— Operation symbols: $+, \cdot$

# Formal Languages

We have the following components:

- Variables: $x, y, z, \ldots$
- Constants: $0, 1$
- Operation symbols: $+, \cdot$
- Terms: 0, 1, $x$, $x + y$, $x \cdot y$.

# Formal Languages

We have the following components:

- Variables: $x, y, z, \dots$
- Constants: $0, 1$
- Operation symbols: $+, \cdot$
- Terms: 0, 1, $x$, $x + y$, $x \cdot y$.
  If $t_1$ and $t_2$ are terms, so are $t_1 + t_2$ and $t_1 \cdot t_2$.

# Formal Languages

We have the following components:

- Variables: $x, y, z, \dots$
- Constants: $0, 1$
- Operation symbols: $+, \cdot$
- Terms: 0, 1, $x$, $x + y$, $x \cdot y$.
  If $t_1$ and $t_2$ are terms, so are $t_1 + t_2$ and $t_1 \cdot t_2$.
  Therefore, $(x + y) \cdot z, (1 + x) \cdot y + z, \dots$ are terms.

# Formal Languages

We have the following components:

— Variables: $x, y, z, \ldots$

— Constants: $0, 1$

— Operation symbols: $+, \cdot$

— Terms: 0, 1, $x$, $x + y$, $x \cdot y$.
If $t_1$ and $t_2$ are terms, so are $t_1 + t_2$ and $t_1 \cdot t_2$.
Therefore, $(x + y) \cdot z, (1 + x) \cdot y + z, \ldots$ are terms.

— Relation symbol: $=$

NTNU
Norwegian University of
Science and Technology

# Formal Languages (continued)

Formulas, sentences:

# Formal Languages (continued)

Formulas, sentences:

— Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.

# Formal Languages (continued)

Formulas, sentences:

— Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: **and** , **or** , **not** .

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: and , or , not .
- Quantifiers: $(\forall x), (\exists x)$.

NTNU
Norwegian University of Science and Technology

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: and , or , not .
- Quantifiers: $(\forall x), (\exists x)$.
  Example. $\psi(x)$ : not $(\exists y)(\exists z)(y < x$ and $z < x$ and $x = y \cdot z)$

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: and , or , not .
- Quantifiers: $(\forall x), (\exists x)$.
  Example. $\psi(x)$ : not $(\exists y)(\exists z)(y < x$ and $z < x$ and $x = y \cdot z)$
- Sentences: Formulas without free variables.

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: and , or , not .
- Quantifiers: $(\forall x), (\exists x)$.
  Example. $\psi(x) : \text{not}\,(\exists y)(\exists z)(y < x \text{ and } z < x \text{ and } x = y \cdot z)$
- Sentences: Formulas without free variables.

$$
\begin{aligned}
&(\forall x)( \\
&\quad \text{not}\,(x = 0 \text{ or } x = 1) \implies \\
&\quad (\exists y)(\exists z)(\psi(y) \text{ and } \psi(z) \text{ and } x + x = y + z) \\
&)
\end{aligned}
$$

NTNU
Norwegian University of
Science and Technology

# Formal Languages (continued)

Formulas, sentences:

- Atomic formulas: $t_1 = t_2$, where $t_1, t_2$ are terms.
  Example. $\varphi(x, y) : x + y = x \cdot y$
- Logical operators: and , or , not .
- Quantifiers: $(\forall x), (\exists x)$.
  Example. $\psi(x) : \text{not}\,(\exists y)(\exists z)(y < x \text{ and } z < x \text{ and } x = y \cdot z)$
- Sentences: Formulas without free variables.

$$
\begin{aligned}
&(\forall x)( \\
&\quad \text{not}\,(x = 0 \text{ or } x = 1) \implies \\
&\quad (\exists y)(\exists z)(\psi(y) \text{ and } \psi(z) \text{ and } x + x = y + z) \\
&)
\end{aligned}
$$

(Goldbach's Conjecture).

# Theories

A theory is a formal language together with a set of axioms.

# Theories

A theory is a formal language together with a set of axioms.

Proofs in a theory:

# Theories

A theory is a formal language together with a set of axioms.

Proofs in a theory:
A proof of a sentence $\xi$ in our language is a sequence

$$P: \quad \xi_1, \xi_2, \dots, \xi_k, \dots, \xi_n = \xi$$

of formulas

# Theories

A theory is a formal language together with a set of axioms.

Proofs in a theory:
A proof of a sentence $\xi$ in our language is a sequence

$$P: \quad \xi_1, \xi_2, \ldots, \xi_k, \ldots, \xi_n = \xi$$

of formulas such that each $\xi_k$ is

— an axiom, or

# Theories

A theory is a formal language together with a set of axioms.

Proofs in a theory:
A proof of a sentence $\xi$ in our language is a sequence

$$P: \quad \xi_1, \xi_2, \ldots, \xi_k, \ldots, \xi_n = \xi$$

of formulas such that each $\xi_k$ is

- — an axiom, or
- — logically follows from $\xi_0, \ldots, \xi_{k-1}$

# Theories

A theory is a formal language together with a set of axioms.

Proofs in a theory:
A proof of a sentence $\xi$ in our language is a sequence

$$P: \quad \xi_1, \xi_2, \dots, \xi_k, \dots, \xi_n = \xi$$

of formulas such that each $\xi_k$ is

— an axiom, or

— logically follows from $\xi_0, \dots, \xi_{k-1}$
(using substitution, modus ponens, generalization, logical axioms).

NTNU
Norwegian University of Science and Technology

# Outline

# Consistency and Completeness

If we have a theory describing some structure,

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers,

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent.

NTNU
Norwegian University of Science and Technology

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$,

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$, there must not be a proof for “not $\xi$”!

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$, there must not be a proof for “not $\xi$”!
   We will assume that Peano Arithmetics is consistent

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$, there must not be a proof for "not $\xi$"!
   We will assume that Peano Arithmetics is consistent (cheating in a sense!).

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$, there must not be a proof for “not $\xi$”!
   We will assume that Peano Arithmetics is consistent (cheating in a sense!).
2. Complete.

# Consistency and Completeness

If we have a theory describing some structure, such as the arithmetics of natural numbers, we would like it to be:

1. Consistent. If there is a proof in the theory for a sentence $\xi$, there must not be a proof for “not $\xi$”!
   We will assume that Peano Arithmetics is consistent (cheating in a sense!).
2. Complete. If $\xi$ is a sentence, there should be a proof for either $\xi$ or “not $\xi$”.

# Gödel’s Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics*

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved,*

NTNU
Norwegian University of Science and Technology

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved, but it is a true statement about natural numbers.*

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved, but it is a true statement about natural numbers. In particular, Peano Arithmetics is incomplete.*

NTNU
Norwegian University of
Science and Technology

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved, but it is a true statement about natural numbers. In particular, Peano Arithmetics is incomplete.*

*In fact, any consistent effective axiomatic description of the arithmetics of natural numbers is incomplete.*

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved, but it is a true statement about natural numbers. In particular, Peano Arithmetics is incomplete.*

*In fact, any consistent effective axiomatic description of the arithmetics of natural numbers is incomplete.*

Effective axiomatic description: There is an algorithm which determines whether a given formula is an axiom.

# Gödel's Incompleteness Theorem

**Theorem (Kurt Gödel, 1931)**

*There is a sentence $\xi$, the so called Gödel sentence, in Peano Arithmetics which cannot be proved, but it is a true statement about natural numbers. In particular, Peano Arithmetics is incomplete.*

*In fact, any consistent effective axiomatic description of the arithmetics of natural numbers is incomplete.*

Effective axiomatic description: There is an algorithm which determines whether a given formula is an axiom.
Recall: $(P7)$ $(\varphi(0)$ and $(\forall x)(\varphi(x) \Longrightarrow \varphi(x+1))) \implies (\forall x)(\varphi(x))$.

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number.

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$).

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

- Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.
- Then, roughly said we construct a sentence $\xi$,

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.

— Then, roughly said we construct a sentence $\xi$, which “says” that there is no proof for $\xi$ in Peano Arithmetics

# Idea behind the Theorem

- Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.
- Then, roughly said we construct a sentence $\xi$, which "says" that there is no proof for $\xi$ in Peano Arithmetics (Liar Paradox).

# Idea behind the Theorem

- Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.
- Then, roughly said we construct a sentence $\xi$, which “says” that there is no proof for $\xi$ in Peano Arithmetics (Liar Paradox).
- There can be no proof for $\xi$ in Peano Arithmetics,

# Idea behind the Theorem

— Every formula $\varphi(x_1, \dots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.

— Then, roughly said we construct a sentence $\xi$, which “says” that there is no proof for $\xi$ in Peano Arithmetics (Liar Paradox).

— There can be no proof for $\xi$ in Peano Arithmetics, but $\xi$ is a true statement about natural numbers.

NTNU
Norwegian University of
Science and Technology

# Idea behind the Theorem

— Every formula $\varphi(x_1, \ldots, x_n)$ in Peano Arithmetics can be represented by a number. We will denote this number $G_\varphi$ (the Gödel number of $\varphi$). So can be represented proofs.

— Then, roughly said we construct a sentence $\xi$, which “says” that there is no proof for $\xi$ in Peano Arithmetics (Liar Paradox).

— There can be no proof for $\xi$ in Peano Arithmetics, but $\xi$ is a true statement about natural numbers.

— The second part is newer with a different proof, it can be essentially found in a nice book by Alfred Tarski.

# Summary

Any axiomatic description of the arithmetics of natural numbers which is

# Summary

Any axiomatic description of the arithmetics of natural numbers which is

1. consistent,

# Summary

Any axiomatic description of the arithmetics of natural numbers which is

1. consistent, and
2. effectively described,

# Summary

Any axiomatic description of the arithmetics of natural numbers which is

1. consistent, and
2. effectively described,

is necessarily incomplete.

NTNU
Norwegian University of Science and Technology

# Outline



NTNU
Norwegian University of
Science and Technology

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory,*

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics,*

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics, and a sentence $\xi$ in the theory*

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics, and a sentence $\xi$ in the theory decides, whether $\xi$ is or is not provable from the axioms.*

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics, and a sentence $\xi$ in the theory decides, whether $\xi$ is or is not provable from the axioms.*

In other words:

— We have given up finding a complete axiomatic description for natural numbers.

NTNU
Norwegian University of Science and Technology

# The Decision Problem Revisited

**Problem (Hilbert's Entscheidungsproblem)**

*Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics, and a sentence $\xi$ in the theory decides, whether $\xi$ is or is not provable from the axioms.*

In other words:

— We have given up finding a complete axiomatic description for natural numbers.

— However, we still want an algorithm for automated proving for the descriptions we have at our disposal.

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

Different computational models:

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

Different computational models:

1. A program in C, Java, Pascal or similar

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

Different computational models:

1. A program in C, Java, Pascal or similar which gets an input file and can write its output to an output file.

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

Different computational models:

1. A program in C, Java, Pascal or similar which gets an input file and can write its output to an output file.
2. A Turing machine (after Alan Turing).

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

Different computational models:

1. A program in C, Java, Pascal or similar which gets an input file and can write its output to an output file.
2. A Turing machine (after Alan Turing).
3. Lambda calculus (by Alonzo Church).

# What Precisely is an Algorithm?

If we want to prove anything about existence of algorithms we need to have a definition of an algorithm.

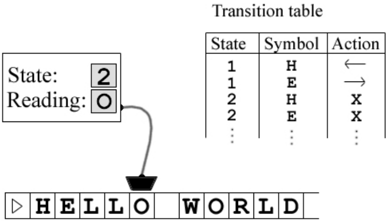
Different computational models:

1. A program in C, Java, Pascal or similar which gets an input file and can write its output to an output file.
2. A Turing machine (after Alan Turing).
3. Lambda calculus (by Alonzo Church).

All the models above have the same computational strength.

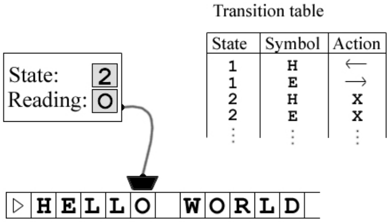
# Church-Turing Thesis

A Turing machine:



# Church-Turing Thesis

A Turing machine:



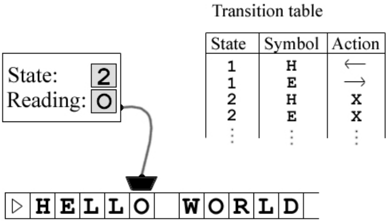

## Church-Turing Thesis

The intuitive notion of an “algorithm” is, formally, a Turing machine

NTNU
Norwegian University of Science and Technology

# Church-Turing Thesis

A Turing machine:

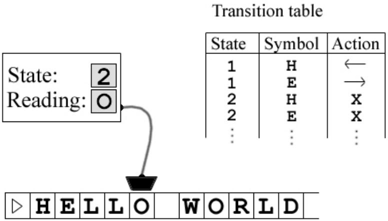



## Church-Turing Thesis

The intuitive notion of an “algorithm” is, formally, a Turing machine which finishes its computation in finite time given any input

# Church-Turing Thesis

A Turing machine:




## Church-Turing Thesis

The intuitive notion of an “algorithm” is, formally, a Turing machine which finishes its computation in finite time given any input (= halts on each input).

# The Halting Problem

**Problem (The Halting Problem)**

*Is there an algorithm (program)* $\mathrm{Halt}(P, F)$

# The Halting Problem

**Problem (The Halting Problem)**

*Is there an algorithm (program)* $\mathrm{Halt}(P, F)$ *which, given a source code* $P$ *of another program and its input file* $F$*,*

# The Halting Problem

**Problem (The Halting Problem)**

*Is there an algorithm (program)* $\mathrm{Halt}(P, F)$ *which, given a source code* $P$ *of another program and its input file* $F$*, decides whether* $P$ *halts on the input* $F$*?*

# The Halting Problem

**Problem (The Halting Problem)**

*Is there an algorithm (program)* $\mathrm{Halt}(P, F)$ *which, given a source code* $P$ *of another program and its input file* $F$*, decides whether* $P$ *halts on the input* $F$*?*

**Theorem (Alan Turing, 1936)**

*There is no such algorithm.*

NTNU
Norwegian University of
Science and Technology

# The Halting Problem

**Problem (The Halting Problem)**

*Is there an algorithm (program)* $\mathrm{Halt}(P, F)$ *which, given a source code* $P$ *of another program and its input file* $F$*, decides whether* $P$ *halts on the input* $F$*?*

**Theorem (Alan Turing, 1936)**

*There is no such algorithm. Therefore, the halting problem is undecidable.*

NTNU
Norwegian University of
Science and Technology

# Idea behind the Theorem

— Suppose we have such a program Halt($P$, $F$).

# Idea behind the Theorem

- Suppose we have such a program Halt($P$, $F$).
- Then define a program

Diag($F$) {<br>
  $x$: *if* Halt($F$, $F$) *then go to x;*<br>
}

# Idea behind the Theorem

— Suppose we have such a program $\mathrm{Halt}(P, F)$.

— Then define a program

$$
\begin{array}{l}
\mathrm{Diag}(F)\ \{ \\
\quad x: \ \ \textit{if}\ \mathrm{Halt}(F, F)\ \textit{then go to } x; \\
\}
\end{array}
$$

— What does then $\mathrm{Halt}(\mathrm{Diag}, \mathrm{Diag})$ return?

NTNU
Norwegian University of
Science and Technology

# Undecidability

**Theorem (Alan Turing, Alonzo Church, 1936)**

*There is no algorithm which, given a sentence $\xi$ in Peano Arithmetics, would decide whether or not $\xi$ is provable from the axioms.*

# Undecidability

**Theorem (Alan Turing, Alonzo Church, 1936)**

*There is no algorithm which, given a sentence $\xi$ in Peano Arithmetics, would decide whether or not $\xi$ is provable from the axioms.*

*The same holds for any consistent axiomatic description of the artihmetics of natural numbers.*

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

— A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics

# Idea behind the Theorem

— A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.

# Idea behind the Theorem

- A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.
- Here, $\mathbf{n} = \underbrace{1 + 1 + \cdots + 1}_{n\ times}$.

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

— A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.

— Here, $\mathbf{n} = \underbrace{1 + 1 + \cdots + 1}_{n \textit{ times}}$.

— Key point: A function $f : \mathbb{N} \to \mathbb{N}$ is definable if and only if it is computable by a program.

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

- A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.
- Here, $\mathbf{n} = \underbrace{1 + 1 + \cdots + 1}_{n \text{ times}}$.
- Key point: A function $f : \mathbb{N} \to \mathbb{N}$ is definable if and only if it is computable by a program.
- With similar arguments as in Gödel's theorem, we show that

$$f(x) = \begin{cases} 1 & \textit{if } x = G_\varphi \textit{ for a provable formula } \varphi \\ 0 & \textit{otherwise} \end{cases}$$

is not definable.

# Idea behind the Theorem

- A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.
- Here, $\mathbf{n} = \underbrace{1 + 1 + \cdots + 1}_{n \text{ times}}$.
- Key point: A function $f : \mathbb{N} \to \mathbb{N}$ is definable if and only if it is computable by a program.
- With similar arguments as in Gödel's theorem, we show that

$$f(x) = \begin{cases} 1 & \textit{if } x = G_\varphi \textit{ for a provable formula } \varphi \\ 0 & \textit{otherwise} \end{cases}$$

is not definable. Hence, it is not computable by a program.

NTNU
Norwegian University of Science and Technology

# Idea behind the Theorem

- A function $f : \mathbb{N} \to \mathbb{N}$ is definable if there is a formula $\varphi(x, y)$ in Peano Arithmetics such that $f(n) = k$ if and only if $\varphi(\mathbf{n}, \mathbf{k})$ is provable.
- Here, $\mathbf{n} = \underbrace{1 + 1 + \cdots + 1}_{n \text{ times}}$.
- Key point: A function $f : \mathbb{N} \to \mathbb{N}$ is definable if and only if it is computable by a program.
- With similar arguments as in Gödel's theorem, we show that

$$f(x) = \begin{cases} 1 & \text{if } x = G_\varphi \text{ for a provable formula } \varphi \\ 0 & \text{otherwise} \end{cases}$$

  is not definable. Hence, it is not computable by a program.
- The second part can be found in the book by Tarski.

NTNU
Norwegian University of Science and Technology