Gödel's theorem

The divorce of Mathematics and Computer Science

Reinhard Kahle

CENTRIA, Universidade Nova de Lisboa
Departamento de Matemática, Universidade de Coimbra

DI/CITI/CENTRIA Seminários Centenário Kurt Gödel

Overview

Syntax and Semantics

- For the understanding of Gödel's theorems it is essential to distinguish between *syntax* and *semantics*.
- This is sometimes complicated, because this distinction is normally not important in *Mathematics*; in general we can *identify* a syntactical expression with its semantic meaning.
- In fact, in *Computer Science* we usually distinguish quite well the syntactic level—the source code of a program—from the semantic one: the specification we expect to be fulfilled when the program is executed.

The language of arithmetic

Definition (The language of arithmetic)

- Logical symbols: $\{\neg, \wedge, \vee, \rightarrow, \forall, \exists, =\}$,
- Variables: $\{x, y, z, w, ...\}$.
- Constant: 0,
- Function symbol: $succ, +, \cdot, \ldots$,

Terms and formulas are build inductively from these symbols.

Example

- Terme: $succ(0)$, $0 + succ(x)$.
- Formulas: $\forall x\ \neg(succ(x) = 0)$, $\forall x\ \exists y\ x + y = 0$.

Semantics: the structure of the natural numbers

The *meaning* of a mathematical expression is given in a *structure*.

Definition (Structure)

A structure $\mathcal{M}$ is given in terms of **set theory** and consists of

- a non-empty set M (the universe),
- constants $c_1, \dots$ (elements of M),
- functions $f_1, \dots, f_n, \dots$ ($f_i : M^{k_i} \longrightarrow M$ if f_i has arity k_i).

Semantics: the structure of the natural numbers

The *meaning* of a mathematical expression is given in a *structure*.

Definition (Structure)

A structure $\mathcal{M}$ is given in terms of **set theory** and consists of

- a non-empty set M (the universe),
- constants $c_1, \dots$ (elements of M),
- functions $f_1, \dots, f_n, \dots$ ($f_i : M^{k_i} \to M$ if f_i has arity k_i).

Definition (The structure $\mathcal{N}$ of arithmetic)

$$\mathcal{N} = \langle \mathbb{N}, 0, succ, +, \cdot \rangle$$

Remark

Here, $+$ is not a symbol of the language, but a designation of the addition function, given as the following set:
$\{(0,0,0),(0,1,1),(0,2,2),\dots,(1,0,1),(1,1,2),\dots,(2,0,2),(2,1,3),\dots\}$

Semantics: Truth

- The relation between the syntactical expressions and the semantical objects is given by a *interpretation function*.
- Usually, this interpretation function is canonical. In fact, the choice of the designations $(0, +, \cdot)$ should be made in a way that the interpretation is canonical.
  Here, we indicate the function by colors:
  - symbols in blue are syntactical entities;
  - objects in green are sets (or elements of them).

Semantics: Truth

- The relation between the syntactical expressions and the semantical objects is given by a *interpretation function*.
- Usually, this interpretation function is canonical. In fact, the choice of the designations (0, $+$, $\cdot$) should be made in a way that the interpretation is canonical.
  Here, we indicate the function by colors:
  - symbols in blue are syntactical entities;
  - objects in green are sets (or elements of them).

Truth (Examples)

$$\mathcal{N} \models \bar{2} + \bar{2} = \bar{4} \quad \Leftrightarrow \quad (2, 2, 4) \in +$$

$$\mathcal{N} \models \exists x\ x \cdot x = x \quad \Leftrightarrow \quad \text{it exists an } n \in \mathbb{N} \text{ such that } (n, n, n) \in \cdot$$

$$\mathcal{N} \models \neg\phi \quad \Leftrightarrow \quad \text{it does } \textbf{not} \text{ hold } \mathcal{N} \models \phi$$

The last clause ensures for all ϕ: $\mathcal{N} \models \phi$ or $\mathcal{N} \models \neg\phi$.

Syntax: Peano Arithmetic

- As "syntax" we consider the construction of a mathematical theory based on axiom systems.
- Given the logical axioms and rules (*Modus Ponens* and *Generalization*) it is only needed to specifiy the *non-logical* axioms.

Definition (Peano Arithmetic, *PA*)

1. $\forall x\ \neg(succ(x) = 0)$,
2. $\forall x\ \forall y\ succ(x) = succ(y) \rightarrow x = y$,
3. $\forall x\ x + 0 = x$,
4. $\forall x\ \forall y\ x + succ(y) = succ(x + y)$,
5. $\forall x\ x \cdot 0 = 0$,
6. $\forall x\ x \cdot succ(y) = (x \cdot y) + x$,
7. $\phi(0) \wedge (\forall y\ \phi(y) \rightarrow \phi(succ(y))) \rightarrow \forall x\ \phi(x)$ for each formula ϕ.

Correctness

A deductive system T is *correct* with respect to a given structure $\mathcal{M}$, if

$$T \vdash \phi \quad \Longrightarrow \quad \mathcal{M} \models \phi.$$

Lemma

PA is correct with respect to $\mathcal{N}$.

The first incompleteness theorem

The question

1. Does PA proves all formulas which are true in $\mathcal{N}$?

$$\mathcal{N} \models \phi \quad \Longrightarrow \quad PA \vdash \phi \quad ?$$

The first incompleteness theorem

The question

1. Does PA proves all formulas which are true in $\mathcal{N}$?

$$\mathcal{N} \models \phi \quad \Longrightarrow \quad PA \vdash \phi \quad ?$$

2. Does PA proves for each formula ϕ,

$$\text{either} \quad PA \vdash \phi \quad \text{or} \quad PA \vdash \neg\phi \quad ?$$

Because of correctness and the fact that the structure $\mathcal{N}$ is by definition "complete", i.e., $\mathcal{N} \models \phi$ or $\mathcal{N} \models \neg\phi$ holds for all ϕ, (1) and (2) are equivalent.

The Gödel sentence

This sentence is not provable.

- If the sentence above is formalizable in *PA*, then *PA* is incomplete.
- Goal: To formalize "This sentence is not provable." in *PA*.
- Two tasks:
  1. To formalize the predicate "x is provable".
  2. To express the self-reference "This sentence...".
- For the first task: *Aritmetization* or *Gödelization* of the notion of provability.
- For the second task: *Diagonalization lemma*

The first incompleteness theorem (PA)

Theorem

- *(Gödelization): If PA is ω-consistent, then for every formula ϕ*

$$PA \vdash \phi \Leftrightarrow PA \vdash Bew_{PA}(\ulcorner \phi \urcorner).$$

- *(Diagonalization lemma): Let $\phi(x)$ be a formula of Peano Arithmetic with exactly one free variable. Then there exists a sentence ψ such that*

$$PA \vdash \psi \leftrightarrow \phi(\ulcorner \psi \urcorner).$$

- *(The first incompleteness theorem): If PA is ω-consistent, then there exists a formula ϕ such that*

$$PA \nvdash \phi \quad \text{and} \quad PA \nvdash \neg\phi.$$

Proof.

Let ϕ such that $\phi \leftrightarrow \neg Bew_{PA}(\ulcorner \phi \urcorner)$. □

The first incompleteness theorem (general case)

- To *complete* PA we could think of adding more axioms (which are true in the structure $\mathcal{N}$).
- But Gödel's argument is *generic*: Replacing the provability predicate $Bew_{PA}(x)$ by the predicate $Bew_T(x)$ which takes the new axioms in considereation, we can actually follow literally the original proof to establish the same result for T.
- The only condition is that the set of new axioms is *recursive* (see below).

Theorem (Gödel 1931, Rosser 1936)

For every consistent deduction system T, which is a recursive extension of PA, there exist a formula ϕ such that

$$T \nvdash \phi \quad \text{and} \quad T \nvdash \neg\phi.$$

The condition of recursiveness

- The Gödelization is a method to associate natural numbers with syntactic expression: $\phi \mapsto \ulcorner \phi \urcorner$.
- The provability predicate $Bew_{PA}(\ulcorner \phi \urcorner)$ expresses—within *PA*—that ϕ is provable in *PA*.
- The definition of Bew_{PA} within *PA* is possible since proofs are *inductively defined* (starting from axioms and closed under rules), and such inductive definitions are formalizable in *PA* by use of *recursive functions*.
- However, for the base case—the axioms—in the inductive definition of Bew_{PA} it is needed that the set of axioms is at most *recursive*.

Tarski's theorem

- Why Gödel's argument does not work in the same way for the structure $\mathcal{N}$ of the natural numbers and the sentence:

This sentence is not true in $\mathcal{N}$.

Tarski's theorem

- Why Gödel's argument does not work in the same way for the structure $\mathcal{N}$ of the natural numbers and the sentence:

This sentence is not true in $\mathcal{N}$.

- Here, we would need a relation, definable in $\mathcal{N}$, which expresses: " $\cdot$ is true in $\mathcal{N}$".

Tarski's theorem

- Why Gödel's argument does not work in the same way for the structure $\mathcal{N}$ of the natural numbers and the sentence:

This sentence is not true in $\mathcal{N}$.

- Here, we would need a relation, definable in $\mathcal{N}$, which expresses: " $\cdot$ is true in $\mathcal{N}$".

Theorem (Tarski)

Truth in $\mathcal{N}$ is not arithmetical definable.

Tarski's theorem

- Why Gödel's argument does not work in the same way for the structure $\mathcal{N}$ of the natural numbers and the sentence:

This sentence is not true in $\mathcal{N}$.

- Here, we would need a relation, definable in $\mathcal{N}$, which expresses: " $\cdot$ is true in $\mathcal{N}$".

Theorem (Tarski)

Truth in $\mathcal{N}$ is not arithmetical definable.

In other words:

The set $\{\phi \mid \mathcal{N} \models \phi\}$ is not recursive.

Tarski's theorem

- Why Gödel's argument does not work in the same way for the structure $\mathcal{N}$ of the natural numbers and the sentence:

This sentence is not true in $\mathcal{N}$.

- Here, we would need a relation, definable in $\mathcal{N}$, which expresses: " $\cdot$ is true in $\mathcal{N}$".

Theorem (Tarski)

Truth in $\mathcal{N}$ is not arithmetical definable.

In other words:

The set $\{\phi \mid \mathcal{N} \models \phi\}$ is not recursive.

This relates to the condition of recursiveness in Gödel's theorem:

A complete axiomatization of $\mathcal{N}$

The set of axioms $\{\phi \mid \mathcal{N} \models \phi\}$ is trivially complete, but it is not recursive.

The Entscheidungsproblem

Situation:

- There exist formulas ϕ such that $PA \nvdash \phi$ and $PA \nvdash \neg\phi$.

The Entscheidungsproblem

Situation:

- There exist formulas ϕ such that $PA \not\vdash \phi$ and $PA \not\vdash \neg\phi$.
- Let us call a formula ϕ for which we can prove $PA \vdash \phi$ or $PA \vdash \neg\phi$ *determinated*.

Question:

- Can we at least *separate* the determinated from the non-determinated formulas?

The Entscheidungsproblem

Situation:

- There exist formulas ϕ such that $PA \nvdash \phi$ and $PA \nvdash \neg\phi$.
- Let us call a formula ϕ for which we can prove $PA \vdash \phi$ or $PA \vdash \neg\phi$ *determinated*.

Question:

- Can we at least *separate* the determinated from the non-determinated formulas?
- This equivalent to the question to *decide* whether $PA \vdash \phi$ or not?

The Entscheidungsproblem

Situation:

- There exist formulas ϕ such that $PA \nvdash \phi$ and $PA \nvdash \neg\phi$.
- Let us call a formula ϕ for which we can prove $PA \vdash \phi$ or $PA \vdash \neg\phi$ *determinated*.

Question:

- Can we at least *separate* the determinated from the non-determinated formulas?
- This equivalent to the question to *decide* whether $PA \vdash \phi$ or not?

Answer:

- No.

Theorem (Church 1936 and Turing 1936)

There is no recursive function f such that

$$f(\phi) = \begin{cases} 0, & \textit{if } PA \vdash \phi \\ 1, & \textit{otherwise.} \end{cases}$$

Church's thesis

- The condition of *recursive function* in the undecidability theorem is essential.

Church's thesis

- The condition of *recursive function* in the undecidability theorem is essential.
- The significance of this theorem for computer science is based on the following fact:

Chruch's thesis

The recursive function correspond exactly to the functions which are *intuitively* computable.

- In fact, the recursive functions are equivalent to the functions computable on a *Turing machine*; they are *Turing complete*.

Church's thesis

- The condition of *recursive function* in the undecidability theorem is essential.
- The significance of this theorem for computer science is based on the following fact:

Chruch's thesis

The recursive function correspond exactly to the functions which are *intuitively* computable.

- In fact, the recursive functions are equivalent to the functions computable on a *Turing machine*; they are *Turing complete*.
- There is no model of computation known which exceeds the class of recursive function (also not quantum computing!).

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.
    1. If we consider structures as the fundamental entities, we have as main result that the set of mathematical truth is not recursive. But, "the world" is still divided in *true* and *false*. The question whether what we can prove as true or false is another issue.

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.
  1. If we consider structures as the fundamental entities, we have as main result that the set of mathematical truth is not recursive. But, “the world” is still divided in *true* and *false*. The question whether what we can prove as true or false is another issue.
  2. If we consider recursive axiomatizations as fundamental entities, the existence of sentences which we can neither prove nor disprove are taken as “real” such that the principle of bivalence is rejected.

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.
  1. If we consider structures as the fundamental entities, we have as main result that the set of mathematical truth is not recursive. But, "the world" is still divided in *true* and *false*. The question whether what we can prove as true or false is another issue.
  2. If we consider recursive axiomatizations as fundamental entities, the existence of sentences which we can neither prove nor disprove are taken as "real" such that the principle of bivalence is rejected.
- Since recursiveness corresponds to the computational capabilities of computers (Church's thesis), it is natural (maybe the only possible way) for *Computer Science* to adopt the second position.

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.
  1. If we consider structures as the fundamental entities, we have as main result that the set of mathematical truth is not recursive. But, "the world" is still divided in *true* and *false*. The question whether what we can prove as true or false is another issue.
  2. If we consider recursive axiomatizations as fundamental entities, the existence of sentences which we can neither prove nor disprove are taken as "real" such that the principle of bivalence is rejected.
- Since recursiveness corresponds to the computational capabilities of computers (Church's thesis), it is natural (maybe the only possible way) for *Computer Science* to adopt the second position.
- However, the *mathematical* self-conception follows the first position.

The divorce

- The first incompleteness theorem refers to the relation of recursive axiomatizations and mathematical structures.
  1. If we consider structures as the fundamental entities, we have as main result that the set of mathematical truth is not recursive. But, "the world" is still divided in *true* and *false*. The question whether what we can prove as true or false is another issue.
  2. If we consider recursive axiomatizations as fundamental entities, the existence of sentences which we can neither prove nor disprove are taken as "real" such that the principle of bivalence is rejected.
- Since recursiveness corresponds to the computational capabilities of computers (Church's thesis), it is natural (maybe the only possible way) for *Computer Science* to adopt the second position.
- However, the *mathematical* self-conception follows the first position.

Here, mathematics and computer science separate:
While mathematics investigates *non-recursive structures*,
computer science deals with *recursive sets*.

A rough comparison

Mathematics	Computer Science
non-recursive sets	recursive sets
undecidable	(semi-)decidable
structures	axiomatic systems
semantics	syntax

A rough comparison

Mathematics	Computer Science
non-recursive sets	recursive sets
undecidable	(semi-)decidable
structures	axiomatic systems
semantics	syntax

An analogy within Computer Science:

Recursive	Feasible
recursive functions	polytime functions

The second incompleteness theorem

Theorem (Gödel 1931, Rosser 1936)

Any consistent deduction system T, which is a recursive extension of PA, can not prove its own consistency.

- The consistency Con_T of T is formalizable by $\neg Bew_T(\ulcorner 0 = 1 \urcorner)$.
- For the theorem it is shown: $T \nvdash Con_T$.

The consistency of PA

- Thus, $PA \nvdash Con_{PA}$.

The consistency of PA

- Thus, $PA \nvdash Con_{PA}$.
- The use of PA is a restriction imposed by philosophical reasons.

The consistency of PA

- Thus, $PA \nvdash Con_{PA}$.
- The use of PA is a restriction imposed by philosophical reasons.
- The unprovability of consistency is relativized to the methods we are allowed to use.

The consistency of PA

- Thus, $PA \nvdash Con_{PA}$.
- The use of PA is a restriction imposed by philosophical reasons.
- The unprovability of consistency is relativized to the methods we are allowed to use.

Classical Geometry

- There is an analogy to the classical construction problems of geometry:
- It makes part of the specification that we can use only *compass* and *ruler*!
- There exist other geometric instruments which, for instance, allow to trisection of an angle.

The consistency of PA

- Thus, $PA \nvdash Con_{PA}$.
- The use of PA is a restriction imposed by philosophical reasons.
- The unprovability of consistency is relativized to the methods we are allowed to use.

- We can use *other* means to prove the consistency of PA:
  - Transfinite induction up to ϵ_0,
  - Functionals of higher types (introduced by Gödel in 1958).

The consistency of *PA*

- Thus, $PA \nvdash Con_{PA}$.
- The use of *PA* is a restriction imposed by philosophical reasons.
- The unprovability of consistency is relativized to the methods we are allowed to use.

- We can use *other* means to prove the consistency of *PA*:
  - Transfinite induction up to ϵ_0,
  - Functionals of higher types (introduced by Gödel in 1958).
- These alternative means, however, are (in a certain sense) non-recursive, and therefore outside the scope of Computer Science.

The consistency of *PA*

- Thus, $PA \nvdash Con_{PA}$.
- The use of *PA* is a restriction imposed by philosophical reasons.
- The unprovability of consistency is relativized to the methods we are allowed to use.

- We can use *other* means to prove the consistency of *PA*:
  - Transfinite induction up to ϵ_0,
  - Functionals of higher types (introduced by Gödel in 1958).
- These alternative means, however, are (in a certain sense) non-recursive, and therefore outside the scope of Computer Science.
- Thus, we can read Gödel's second incompleteness theorem as:

> Computers cannot prove the consistency of *PA*.
> Mathematicians and/or philosophers might can...

Kurt Gödel 1906 – 1978

Kurt Gödel's achievement in modern logic . . . is a landmark which will remain visible far in space and time.

John von Neumann