

Gödel's Incompleteness Theorems

Jan Sprenger

Vienna University of Technology
Institute for Analysis and Scientific Computing

July 10th, 2009

Overview and Outline of Talk

Aim: Present idea of Gödel's incompleteness theorems

Structure of talk:

- Historical Outset: What's the point?
- formal systems: language and logical rules
- sketch of proof

Historical Outset

- Beginning of 20th century: foundational crisis in mathematics (e.g. Russels contradiction in Cantor's set theory)
- Leads to discussion: What to do?
- One answer (by Hilbert): formal system that works according to strict rules
- One big problem: self-reference – supposedly eliminated with formal systems, as formal language does not allow it
- Gödel: Yes it does.

formal systems: idea

Idea: Formulate mathematics in a strictly formal way and language that does not allow 'self-referecne' and ambiguous formulations.

Requirements:

- Formal language allowing formulation of entire mathematics
- Logical rules for formal proofs that are 'finite'
- Consistent, 'unanimously accepted' system of axioms from which theorems are deduced
- Completeness: All correct statements can be proved
- Consistency: No wrong statements can be proved

formal languages

- limited (yet infinite) alphabet: $f_i^n, R_i^n, v_i, c_i, \forall, \exists, \wedge, \vee, \rightarrow, \leftrightarrow, \neg$
- Strict grammatical rules, stating exactly what constitutes admissible 'terms' and 'formulas'
- Terms: Basic building block (comparable to words), e.g. $c_1, f_1^2 v_2 c_3, \dots$
- Formulas: Final constructions (sentences), consisting of terms t_i and symbols, e.g. $R_1^2 t_1 t_2, \forall v_1 R_1^2 v_1 c_1, \dots$
- This can be explicitly strictly stated, so it is perfectly clear what is a term, what a formula and what is nonsense
- Avoids ambiguities, but notation is 'unhandy'

Axioms

- Initial set of Formulas considered to be true
- Some well-known axiomatic systems: Peano arithmetic, Euclid geometry.
- Usually want to have as little axioms as possible (but all that are necessary)
- crucial: set of axiom has to be sound, such that no contradiction can be derived

logical rules

- Only allowed rules: instances of tautologies, modus ponens and a few rules for quantifiers
- Tautologies: logical constructions that are always true, e.g. $A \rightarrow (B \rightarrow A)$, ...
- Modus ponens: $(A \rightarrow B)$ and A imply B
- quantifiers: e.g. $\forall v_i(A \rightarrow B)$ implies $\forall v_i A \rightarrow \forall v_i B$, ...

formal proofs

- Definition: A formal proof for the formula φ_n in a given formal system is a string of formulas $\varphi_i, 0 \leq i \leq n$ in its language, such that each φ_i is either an axiom or the result of applying a rule to formulas $\varphi_0, \dots, \varphi_k$ which appear in the sequence before φ_i .
- Employs only finite means
- Usually incredibly lengthy and quite complicated even for 'simple' concepts
- No ambiguities, 'everyone' can check whether something is a formal proof or not (even a computer program)

the theorems

Theorem (1st incompleteness theorem)

No sound formal system encompassing arithmetics is complete.

Theorem (2nd incompleteness theorem)

No sound formal system encompassing arithmetics can prove its own soundness.

sketch of proof

Recall: Self-reference is bad.

- Encode formal language in natural numbers, rules and notions like 'term' in 'natural' functions (arithmetisation)
- Represent working with the system inside the system, by computations on natural numbers (representation)
- Construct a 'self-referring' fixpoint (diagonalisation)

Arithmetisation I

- Idea: Turn talking about a system into computations on numbers
- Famous example: cartesian plane
- Proceeds recursively, defining first numbers for symbols, then for terms, then for formula
- Uniquely encode symbols (with 'tags'), use Cantor pairing (injective mapping $\mathbb{N}^2 \rightarrow \mathbb{N}$).
- Use prime encoding, relying on Fundamental Theorem of Algebra, to uniquely reflect terms and formulas.
- You can now uniquely assign a number to each and every possible formula, term and symbol in the language.

Arithmetisation II

- Make sure you can represent notions like 'modus ponens', 'Axiom', and, finally, 'Proof' within the system.
- Heavily relies on primitive recursion.
- Most important function here: $Prf(n, m) = 1$ iff n is a proof of the formula m .
- You can now represent anything the formal system does by a calculation on natural numbers.

Representation

- Recall: Working with the system can be represented by a calculation on natural numbers.
- However: System itself encompasses natural numbers!
- Therefore possible: Proving within the system assertions about the system.
- Important here: Use formula $pf(n) := \exists m Prf(m, n) = 1$.
- Finally get: A formula φ is provable in the system iff $pf(gn(\varphi))$ is provable.

diagonalisation

- Construct a fixpoint that uses self-reference.
- Use diagonalisation theorem: for every formula ψ with one free variable there is a sentence φ such that $\varphi \leftrightarrow \psi(gn(\varphi))$ is provable within the formal system.
- Central: system must allow representation of all primitive recursive functions (peano arithmetic does this)
- Gödel used (without lemma) $\neg pf(gn(x))$ as ψ .
- Yields: $\gamma \leftrightarrow \neg pf(gn(\gamma))$ in the system.

proof of the first theorem

- Fixpoint (easier version): $\gamma \leftrightarrow pf(gn(\neg\gamma))$.
- Recall: Representation gives φ is provable iff $pf(gn(\varphi))$ is provable.
- Assuming γ is provable, we get $\neg\gamma$ is provable, and vice versa.
- Therefore, if the system is sound, γ is not decidable.

the second theorem

- Fixpoint: $\gamma \leftrightarrow \neg pf(gn(\gamma))$.
- Informally, a direct consequence of the first theorem.
- Remember: If the formal system is sound, γ can not be proved.
- Arithmetisation&Representation: $Snd \rightarrow \neg pf(gn(\gamma))$ is provable.
- Therefore, if the system can prove that it is sound, it can not be sound.