TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Lucas Waclawczyk

Decidability of Logical Theories

Proseminar Theoretical Computer Science // Dresden, May 16, 2020

Can we mechanize mathematics?

Inhalt

First-Order Logic

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y\; \Big[R_1(p,q) \wedge \big(\neg (R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p) \big) \Big]$$

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y \left[R_1(p,q) \wedge \left(\neg(R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p) \right) \right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y \;\left[R_1(p,q) \wedge \Big(\neg\big(R_1(x,1) \wedge R_1(y,1)\big) \vee R_2(x,y,p) \Big) \right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y\ \left[R_1(p,q) \wedge \left(\neg(R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p)\right)\right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$
- Boolean operators: $\wedge, \vee, \neg$

Syntax

$$\forall q \, \exists p \, \forall x \, \forall y \; \Big[R_1(p, q) \wedge \big(\neg (R_1(x, 1) \wedge R_1(y, 1)) \vee R_2(x, y, p) \big) \Big]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$
- Boolean operators: $\wedge, \vee, \neg$
- Relation symbols: $R_1, R_2, \ldots$

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y \;\left[R_1(p,q) \wedge \left(\neg (R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p) \right) \right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$
- Boolean operators: $\wedge, \vee, \neg$
- Relation symbols: $R_1, R_2, \ldots$
- Special characters: $[,], (,)$

Syntax

$$\forall q\, \exists p\, \forall x\, \forall y \; \left[R_1(p,q) \land \left(\neg(R_1(x,1) \land R_1(y,1)) \lor R_2(x,y,p) \right) \right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$
- Boolean operators: $\land, \lor, \neg$
- Relation symbols: $R_1, R_2, \ldots$
- Special characters: [,], (,)

Simplified here:

- mostly "sentences" (no free variables)

Syntax

$$\forall q\,\exists p\,\forall x\,\forall y\;\left[R_1(p,q)\wedge\left(\neg(R_1(x,1)\wedge R_1(y,1))\vee R_2(x,y,p)\right)\right]$$

Formulas consist of:

- Quantifiers: $\forall$, $\exists$
- Variables: $p, q, x, y, \ldots$
- Boolean operators: $\wedge, \vee, \neg$
- Relation symbols: $R_1, R_2, \ldots$
- Special characters: [,], (,)

Simplified here:

- mostly "sentences" (no free variables)
- only prenex normal form (all quantifiers on the left)

Semantics

$$\forall q\, \exists p\, \forall x\, \forall y\ \left[R_1(p,q) \wedge \left(\neg(R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p)\right)\right]$$

Semantics

$$\forall q\, \exists p\, \forall x\, \forall y\ \left[R_1(p,q) \wedge \left(\neg\left(R_1(x,1) \wedge R_1(y,1)\right) \vee R_2(x,y,p)\right)\right]$$

$$\overset{!}{=}\ \text{"There are infinitely many prime numbers."}$$

Semantics

$$\forall q\, \exists p\, \forall x\, \forall y \left[R_1(p,q) \wedge \left(\neg (R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p) \right) \right]$$

- *universe* $\mathcal{U}$
  - possible values for variables
  - here $\mathbb{N}$

Semantics

$$\forall q\, \exists p\, \forall x\, \forall y\ \left[R_1(p,q) \wedge \left(\neg(R_1(x,1) \wedge R_1(y,1)) \vee R_2(x,y,p)\right)\right]$$

$$= \quad \forall q\, \exists p\, \forall x,y\ [p > q \wedge (x,y > 1 \rightarrow xy \neq p)]$$

- *universe* $\mathcal{U}$
  - possible values for variables
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe + assignment of relations
  - here $(\mathbb{N},\ >_2,\ (\times \neq)_3)$

Semantics

$$\forall q\, \exists p\, \forall x,y\ \ [p > q \wedge (x,y > 1 \rightarrow xy \neq p)]$$

$$\text{vs.}\ \ [(x + x = y)\ \vee\ (x \geq y)]$$

- *universe* $\mathcal{U}$
  - possible values for variables
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe + assignment of relations
  - here $(\mathbb{N},\ >_2,\ (\times \neq)_3)$
- *language* $\mathrm{L}(\mathcal{M})$
  - sentences that make sense in $\mathcal{M}$
  - here $\mathrm{L}(\mathbb{N},\ >_2,\ (\times \neq)_3)$

Semantics

$$\forall q\, \exists p\, \forall x, y\ [p > q \wedge (x, y > 1 \rightarrow xy \neq p)]$$

$$\text{vs. } \forall y\, \exists x\ [\neg(xx \neq y)]$$

- *universe* $\mathcal{U}$
  - possible values for variables
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe + assignment of relations
  - here $(\mathbb{N}, >_2, (\times \neq)_3)$
- *language* $\mathrm{L}(\mathcal{M})$
  - sentences that make sense in $\mathcal{M}$
  - here $\mathrm{L}(\mathbb{N}, >_2, (\times \neq)_3)$
- *theory* $\mathrm{Th}\,(\mathcal{M})$
  - true sentences formed with $\mathcal{M}$
  - here $\mathrm{Th}\,(\mathbb{N}, >_2, (\times \neq)_3)$

Semantics

$$\forall q\ \exists p\ \forall x,y\ \ [p > q \wedge (x,y > 1 \rightarrow xy \neq p)]$$

$=$ "There are infinitely many prime numbers."

- *universe* $\mathcal{U}$
  - possible values for variables
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe + assignment of relations
  - here $(\mathbb{N},\ >_2,\ (\times \neq)_3)$
- *language* $\mathrm{L}(\mathcal{M})$
  - sentences that make sense in $\mathcal{M}$
  - here $\mathrm{L}(\mathbb{N},\ >_2,\ (\times \neq)_3)$
- *theory* $\mathrm{Th}\,(\mathcal{M})$
  - true sentences formed with $\mathcal{M}$
  - here $\mathrm{Th}\,(\mathbb{N},\ >_2,\ (\times \neq)_3)$

What Is Decidability?

- here: for logic (there is a more general definition)
- let $\mathcal{M}$ be a model, $\varphi \in \mathrm{L}(\mathcal{M})$

$$\mathrm{Th}\,(\mathcal{M})\ \text{decidable} \ :=$$

$$\text{there is an algorithm that decides whether } \varphi \text{ is true in } \mathcal{M}$$

Th $(\mathbb{N}, +)$ – A Decidable Theory

Theorem 1

Theorem

Th $(\mathbb{N}, +)$ *is decidable.*

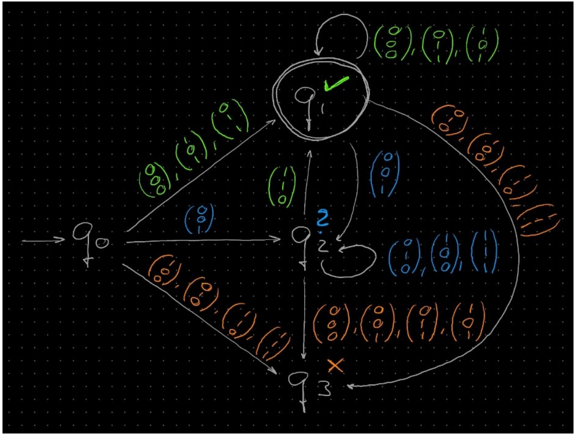
Theorem 1

Theorem

Th $(\mathbb{N}, +)$ *is decidable.*

i.e., there is an algorithm that can decide,
whether a sentence $\varphi \in \mathrm{L}(\mathbb{N}, +)$ is true or false.

Review: Automata



Example automaton that accepts all correct binary additions

Proof of Theorem 1

Idea: Construct an automaton that accepts an (almost) empty input iff the given sentence is true.

Let $i \in \mathbb{N} \setminus \{0\}$ and define $\Sigma_i := \{0, 1\}^i$ and $\Sigma_0 := \{()\}$.
An example for a word in Σ_3:

$$\begin{pmatrix}0\\1\\1\end{pmatrix} \begin{pmatrix}0\\0\\1\end{pmatrix} \begin{pmatrix}1\\1\\0\end{pmatrix} \quad \sim \quad \begin{pmatrix}1\\5\\6\end{pmatrix}$$

Proof of Theorem 1

Now, let

- $i \in \{0, \ldots, l\}$
- $\varphi = Q_1 x_1 \ldots Q_l x_l\, [\psi(x_1, \ldots, x_l)] \in \mathrm{L}(\mathbb{N}, +)$ where $Q_1, \ldots, Q_l \in \{\forall, \exists\}$

Proof of Theorem 1

Now, let

- $i \in \{0, \ldots, l\}$
- $\varphi = Q_1 x_1 \ldots Q_l x_l \, [\psi(x_1, \ldots, x_l)] \in \mathrm{L}(\mathbb{N}, +)$ where $Q_1, \ldots, Q_l \in \{\forall, \exists\}$
- $\varphi_i := Q_{i+1} x_{i+1} \ldots Q_l x_l \, [\psi(x_1, \ldots, x_l)]$

  $\Rightarrow \varphi_0 = \varphi$

  $\Rightarrow \varphi_l := \psi$

Proof of Theorem 1

Now, let

- $i \in \{0, \dots, l\}$
- $\varphi = Q_1 x_1 \dots Q_l x_l \, [\psi(x_1, \dots, x_l)] \in \mathrm{L}(\mathbb{N}, +)$ where $Q_1, \dots, Q_l \in \{\forall, \exists\}$
- $\varphi_i := Q_{i+1} x_{i+1} \dots Q_l x_l \, [\psi(x_1, \dots, x_l)]$
  $\Rightarrow \varphi_0 = \varphi$
  $\Rightarrow \varphi_l := \psi$
- $\varphi_i(a_1, \dots, a_i) := Q_{i+1} x_{i+1} \dots Q_l x_l \, [\psi(a_1, \dots, a_i, x_{i+1}, \dots, x_l)]$

Proof of Theorem 1

Now, let

- $i \in \{0, \ldots, l\}$
- $\varphi = Q_1 x_1 \ldots Q_l x_l \, [\psi(x_1, \ldots, x_l)] \in \mathrm{L}(\mathbb{N}, +)$ where $Q_1, \ldots, Q_l \in \{\forall, \exists\}$
- $\varphi_i := Q_{i+1} x_{i+1} \ldots Q_l x_l \, [\psi(x_1, \ldots, x_l)]$
  $\Rightarrow \varphi_0 = \varphi$
  $\Rightarrow \varphi_l := \psi$
- $\varphi_i(a_1, \ldots, a_i) := Q_{i+1} x_{i+1} \ldots Q_l x_l \, [\psi(a_1, \ldots, a_i, x_{i+1}, \ldots, x_l)]$

$\Longrightarrow$ φ_l is only a Boolean expression.

Proof of Theorem 1

Construct an automaton A_l that behaves like $\varphi_l = \psi$, meaning A_l accepts exactly the tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true:

- Take one addition automaton for each addition term in φ_l
- Combine them:
  - automaton product for $\wedge$
  - automaton union for $\vee$
  - automaton complement for $\neg$

  in a way that they behave like φ_l.

Proof of Theorem 1

Construct an automaton A_l that behaves like $\varphi_l = \psi$, meaning A_l accepts exactly the tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true:

- Take one addition automaton for each addition term in φ_l
- Combine them:
  - automaton product for $\wedge$
  - automaton union for $\vee$
  - automaton complement for $\neg$

  in a way that they behave like φ_l.

Proof of Theorem 1

Construct an automaton A_l that behaves like $\varphi_l = \psi$, meaning A_l accepts exactly the tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true:

- Take one addition automaton for each addition term in φ_l
- Combine them:
  - automaton product for $\wedge$
  - automaton union for $\vee$
  - automaton complement for $\neg$

  in a way that they behave like φ_l.

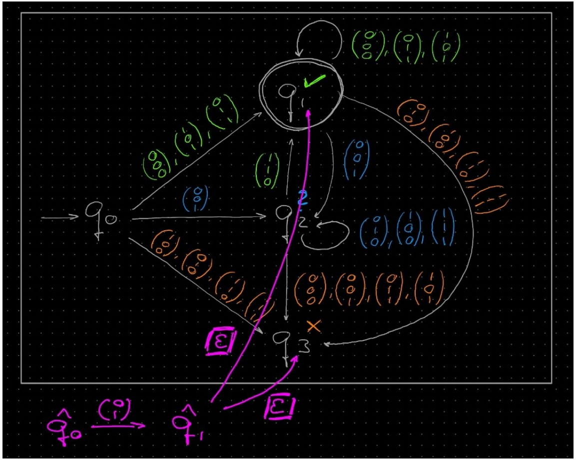
Important: There is an algorithm that constructs A_l from φ_l

Proof of Theorem 1

If $Q_i = \exists$, construct automaton A_i from A_{i+1} by

- copying A_{i+1}
- adding a new start state and one state for each character in Σ_i
- making A_i guess the right a_{i+1} non-deterministically

Proof of Theorem 1



Example construction of non-deterministic guessing

Proof of Theorem 1

If $Q_i = \exists$, construct automaton A_i from A_{i+1} by

- copying A_{i+1}
- adding a new start state and one state for each character in Σ_i
- making A_i guess the right a_{i+1} non-deterministically

If else $Q_i = \forall$, use complementation twice ($\forall x_i \varphi_{i+1} = \neg \exists x_i \neg \varphi_{i+1}$)

Proof of Theorem 1

If $Q_i = \exists$, construct automaton A_i from A_{i+1} by

- copying A_{i+1}
- adding a new start state and one state for each character in Σ_i
- making A_i guess the right a_{i+1} non-deterministically

If else $Q_i = \forall$, use complementation twice ($\forall x_i \varphi_{i+1} = \neg \exists x_i \neg \varphi_{i+1}$)

$\Longrightarrow \quad A_i$ accepts input $(a_1, \ldots, a_i) \in \mathbb{N}^i \quad \Leftrightarrow \quad \varphi_i$ is true

Proof of Theorem 1

If $Q_i = \exists$, construct automaton A_i from A_{i+1} by

- copying A_{i+1}
- adding a new start state and one state for each character in Σ_i
- making A_i guess the right a_{i+1} non-deterministically

If else $Q_i = \forall$, use complementation twice ($\forall x_i \varphi_{i+1} = \neg \exists x_i \neg \varphi_{i+1}$)

$\Longrightarrow \qquad A_i$ accepts input $(a_1, \ldots, a_i) \in \mathbb{N}^i \quad \Leftrightarrow \quad \varphi_i$ is true

$\Longrightarrow \qquad A_0$ accepts input $() \quad \Leftrightarrow \quad \varphi_0 = \varphi$ is true

Proof of Theorem 1

If $Q_i = \exists$, construct automaton A_i from A_{i+1} by

- copying A_{i+1}
- adding a new start state and one state for each character in Σ_i
- making A_i guess the right a_{i+1} non-deterministically

If else $Q_i = \forall$, use complementation twice $(\forall x_i \varphi_{i+1} = \neg\exists x_i \neg\varphi_{i+1})$

$\Longrightarrow \quad A_i$ accepts input $(a_1, \ldots, a_i) \in \mathbb{N}^i \quad \Leftrightarrow \quad \varphi_i$ is true

$\Longrightarrow \quad A_0$ accepts input $() \quad \Leftrightarrow \quad \varphi_0 = \varphi$ is true

Let the algorithm return "$\varphi \in \mathrm{Th}(\mathbb{N}, +)$" $\quad \Leftrightarrow \quad A_0$ accepts input $()$

Th $(\mathbb{N}, +, \times)$ – An Undecidable Theory

Theorem 2

Theorem

$Th\left(\mathbb{N}, +, \times\right)$ is undecidable.

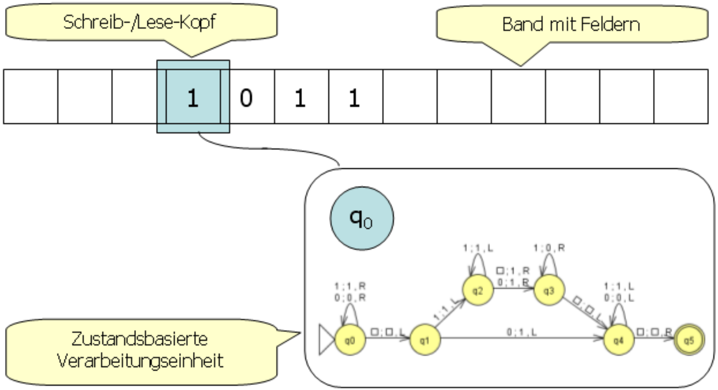
Theorem 2

Theorem

Th $(\mathbb{N}, +, \times)$ *is undecidable.*

i.e., there is no algorithm that can decide,
whether a sentence $\varphi \in$ L$(\mathbb{N}, +)$ is true or false.

Turing Machines



Example turing machine
source: [2]

Proof Idea for Theorem 2

- **The word problem for Turing machines is undecidable.** ([5], Satz 5.6)

Proof Idea for Theorem 2

- The word problem for Turing machines is undecidable. ([5], Satz 5.6)
- There is a mapping reduction that translates
  - a Turing machine M and a string w

Proof Idea for Theorem 2

- The word problem for Turing machines is undecidable. ([5], Satz 5.6)
- There is a mapping reduction that translates
  - a Turing machine M and a string w
  - to a formula $\varphi_{M,w}(x) \in L(\mathbb{N}, +, \times)$ that contains only one free variable x, such that

Proof Idea for Theorem 2

- The word problem for Turing machines is undecidable. ([5], Satz 5.6)
- There is a mapping reduction that translates
  - a Turing machine M and a string w
  - to a formula $\varphi_{M,w}(x) \in \mathrm{L}(\mathbb{N}, +, \times)$ that contains only one free variable x, such that
  - $\varphi_{M,w}(x)$ is true $\Leftrightarrow$ M accepts w (with the computation history suitably encoded in x)

Proof Idea for Theorem 2

- The word problem for Turing machines is undecidable. ([5], Satz 5.6)
- There is a mapping reduction that translates
  - a Turing machine M and a string w
  - to a formula $\varphi_{M,w}(x) \in \mathrm{L}(\mathbb{N}, +, \times)$ that contains only one free variable x, such that
  - $\varphi_{M,w}(x)$ is true $\Leftrightarrow$ M accepts w (with the computation history suitably encoded in x)

Assume $\mathrm{Th}\,(\mathbb{N}, +, \times)$ is decidable.

$\Longrightarrow$ The formulas $\exists x\, [\varphi_{M,w}(x)] \in \mathrm{Th}\,(\mathbb{N}, +, \times)$ are decidable.

Proof Idea for Theorem 2

- The word problem for Turing machines is undecidable. ([5], Satz 5.6)
- There is a mapping reduction that translates
  - a Turing machine M and a string w
  - to a formula $\varphi_{M,w}(x) \in L(\mathbb{N}, +, \times)$ that contains only one free variable x, such that
  - $\varphi_{M,w}(x)$ is true $\Leftrightarrow$ M accepts w (with the computation history suitably encoded in x)

Assume $\mathrm{Th}\,(\mathbb{N}, +, \times)$ is decidable.
$\Longrightarrow$ The formulas $\exists x\, [\varphi_{M,w}(x)] \in \mathrm{Th}\,(\mathbb{N}, +, \times)$ are decidable.
$\Longrightarrow$ The word problem for Turing machines is decidable.

Proof Idea for Theorem 2

- **The word problem for Turing machines is undecidable.** ([5], Satz 5.6) $\lightning$
- There is a mapping reduction that translates
  - a Turing machine M and a string w
  - to a formula $\varphi_{M,w}(x) \in \mathrm{L}(\mathbb{N}, +, \times)$ that contains only one free variable x, such that
  - $\varphi_{M,w}(x)$ is true $\Leftrightarrow$ M accepts w (with the computation history suitably encoded in x)

Assume $\mathrm{Th}\,(\mathbb{N}, +, \times)$ is decidable.

$\Longrightarrow$ The formulas $\exists x\,[\varphi_{M,w}(x)] \in \mathrm{Th}\,(\mathbb{N}, +, \times)$ are decidable.

$\Longrightarrow$ The word problem for Turing machines is decidable.

$\Longrightarrow$ $\lightning$

Thinking further...

Is there a true, unprovable sentence?

Thinking further...

Is there a true, unprovable sentence?

A proof is a series of implications and can be written as a string over some alphabet (here: L($\mathbb{N}, +, \times$)). Assumptions:

A_1 Proofs can be checked by a machine.

A_2 Provable statements are true.

Thinking further...

Is there a true, unprovable sentence?

A proof is a series of implications and can be written as a string over some alphabet (here: $L(\mathbb{N}, +, \times)$). Assumptions:

A_1 Proofs can be checked by a machine.

A_2 Provable statements are true.

Lemmas:

1. The provable statements of $\mathrm{Th}\,(\mathbb{N}, +, \times)$ are Turing recognizable.
   Proof idea: Just try all possible (suitably encoded) proofs.

Thinking further...

Is there a true, unprovable sentence?

A proof is a series of implications and can be written as a string over some alphabet (here: L($\mathbb{N}, +, \times$)). Assumptions:

A_1 Proofs can be checked by a machine.

A_2 Provable statements are true.

Lemmas:

1. The provable statements of Th $(\mathbb{N}, +, \times)$ are Turing recognizable.
   Proof idea: Just try all possible (suitably encoded) proofs.
2. There is a (true) statement in Th $(\mathbb{N}, +, \times)$ that is not provable.
   Proof idea: Contradiction to Theorem 2 by using 1.

Gödel’s Incompleteness Theorem

A True, Unprovable Statement

Theorem

We can construct a true statement in Th $(\mathbb{N}, +, \times)$, *that is not provable.*

A True, Unprovable Statement

Theorem

We can construct a true statement in Th $(\mathbb{N}, +, \times)$*, that is not provable.*

Proof: Let M be a Turing machine that operates as follows.

- Delete the input
- Look for proof of $\varphi := \neg\exists x[\varphi_{M,0}(x)] \in \text{Th}\,(\mathbb{N}, +, \times)$
- Accept input $\Leftrightarrow$ proof for φ has been found

A True, Unprovable Statement

Theorem

We can construct a true statement in Th $(\mathbb{N}, +, \times)$, that is not provable.

Proof: Let M be a Turing machine that operates as follows.

- Delete the input
- Look for proof of $\varphi := \neg\exists x[\varphi_{M,0}(x)] \in \mathrm{Th}\,(\mathbb{N}, +, \times)$
- Accept input $\Leftrightarrow$ proof for φ has been found

$\Longrightarrow$ φ is the wanted statement.

Gödel’s Method

- Construct the statement "This statement cannot be proved by the axioms."

Gödel’s Method

- Construct the statement "This statement cannot be proved by the axioms."
- Argue against just adding this statement to the axiom.

Gödel's Method

- Construct the statement "This statement cannot be proved by the axioms."
- Argue against just adding this statement to the axiom.

$\Longrightarrow$ Incompleteness ☺

Conclusion

Conclusion

$\Longrightarrow$ There are (very simple) indecidable logical theories.

Conclusion

$\Longrightarrow$ There are (very simple) indecidable logical theories.

$\Longrightarrow$ Mathematics cannot be mechanized.

Conclusion

$\Longrightarrow$ There are (very simple) indecidable logical theories.

$\Longrightarrow$ Mathematics cannot be mechanized.

$\Longrightarrow$ No sound logical system can be complete.

References

References

[1] Martin Alessandro - Own work, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=75082873`

[2] `https://www.inf-schule.de/grenzen/berechenbarkeit/turingmaschine/station_turingmaschine`

[3] Michael Sipser: Introduction to the Theory of Computation. Thomson Course Technology, 2006

[4] `https://www.youtube.com/watch?v=04ndIDcDSGc`

[5] Prof. Dr. Franz Baader: Skript Theoretische Informatik und Logik (Sommersemester 2020), TU Dresden