TECHNISCHE UNIVERSITÄT DRESDEN

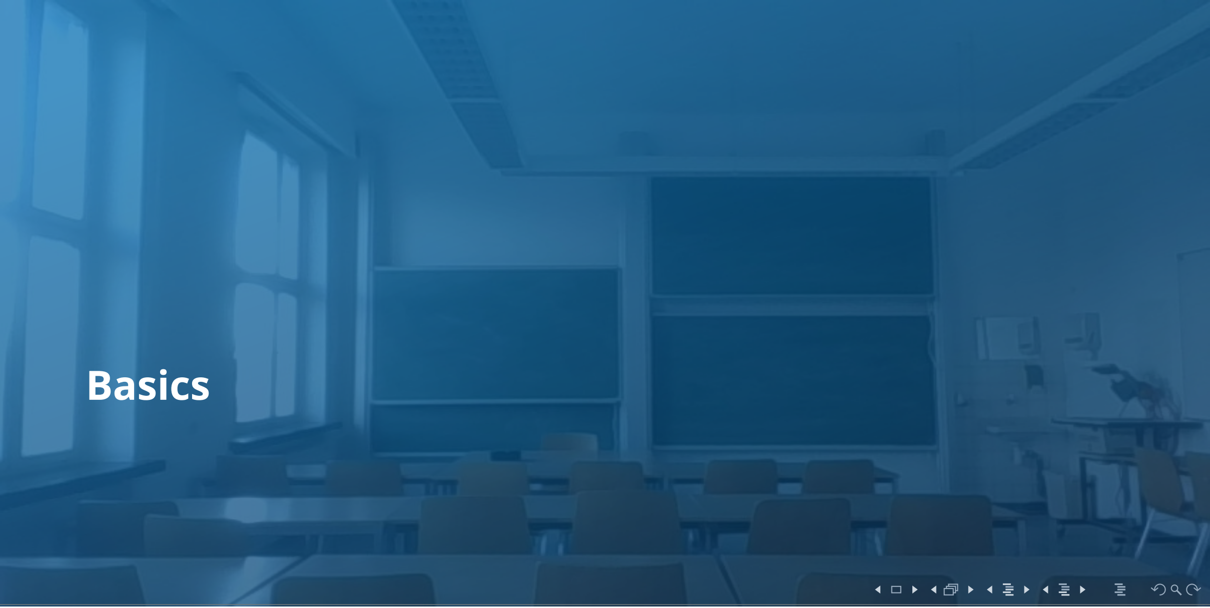
DRESDEN concept

Lucas Waclawczyk

# **Decidability** of Logical Theories

Proseminar Theoretical Computer Science // Dresden, May 11, 2020

# Inhalt



# Basics

# First-Order Logic

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

# First-Order Logic

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

Formulas consist of:

- Quantors: $\forall$, $\exists$

# First-Order Logic

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

Formulas consist of:

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

# First-Order Logic

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

---

Formulas consist of:

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

# First-Order Logic

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

---

Formulas consist of:

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

# First-Order Logic

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

---

Formulas consist of:

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

# First-Order Logic

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

---

Formulas consist of:

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

Simplified here:

- mostly "sentences"

# First-Order Logic

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

---

Formulas consist of:

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

Simplified here:

- mostly "sentences"
- only prenex normal form

# First-Order Logic

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

# First-Order Logic

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

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

# First-Order Logic

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

---

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

# First-Order Logic

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

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

---

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

# First-Order Logic

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

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

---

- *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* $L(\mathcal{M})$
  - sentences that make sense in $\mathcal{M}$
  - here $L(\mathbb{N},\ >_2,\ (\times \neq)_3)$

# First-Order Logic

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

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

---

- *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 $\mathrm{Th}\,(\mathbb{N},\ >_2,\ (\times \neq)_3)$

# First-Order Logic

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

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

---

- *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?

For logic (there is a more general definition):

- $\mathcal{M}$ model, $\varphi \in \mathrm{L}(\mathcal{M})$
- $\mathrm{Th}(\mathcal{M})$ decidable $:=$ there is an algorithm that decides whether $\varphi$ is true in $\mathcal{M}$

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

# Theorem 1

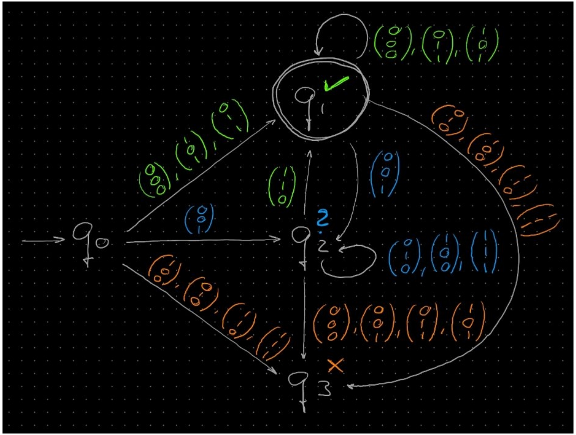
Th $(\mathbb{N}, +)$ is decidable

# Theorem 1

$\mathrm{Th}(\mathbb{N},+)$ 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

Let $i \in \mathbb{N} \setminus \{0\}$ and define $\Sigma_i := \{0,1\}^i$ and $\Sigma_0 := \{()\}$.
An example for a word in $\Sigma_2$:

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

# Proof of Theorem 1

Now, let

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

# Proof of Theorem 1

Now, let

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

$\implies$ $\varphi_l$ is only a Boolean expression.

# Proof of Theorem 1

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

  in a way that they behave like $\varphi_l$, i. e. that the resulting automaton accepts tuples $(a_1, \dots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \dots, a_l)$ is true.

# Proof of Theorem 1

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

  in a way that they behave like $\varphi_l$, i. e. that the resulting automaton accepts tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true.

We call that automaton $A_l$.

# Proof of Theorem 1

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

  in a way that they behave like $\varphi_l$, i. e. that the resulting automaton accepts tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true.

We call that automaton $A_l$.

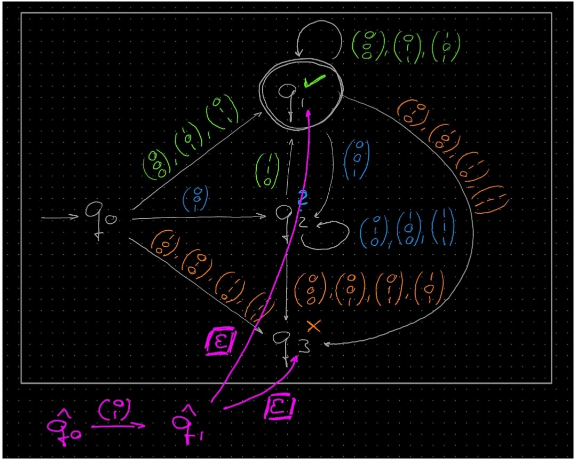
**Important:** There is an algorithm that constructs $A_l$ from $\varphi_l = \psi$.

# 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 $\Sigma_i$
- 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 $\Sigma_i$
- 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 $\Sigma_i$
- 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

# 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 $\Sigma_i$
- 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

# 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 $\Sigma_i$
- 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

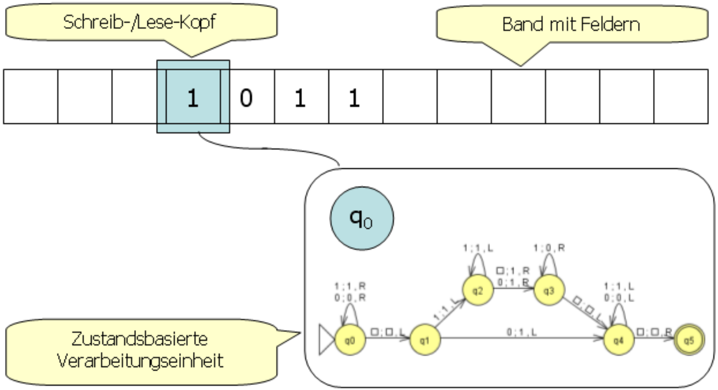
$\mathrm{Th}\,(\mathbb{N}, +, \times)$ is undecidable

# Theorem 2

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.

# Proof Idea for Theorem 2

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

# Proof Idea for Theorem 2

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

# Proof Idea for Theorem 2

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

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

# Proof Idea for Theorem 2

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

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

# Proof Idea for Theorem 2

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

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

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

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

$\Longrightarrow$ $\lightning$

# Thinking further...

Assumptions:

$A_1$ Proofs can be checked by a machine.

$A_2$ Provable statements are true.

# Thinking further...

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...

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.
2. There is a true statement in $\mathrm{Th}(\mathbb{N}, +, \times)$ that is not provable.
   **Proof idea:** Contradiction to Theorem 2 by using 1.

# Gödel’s Incompleteness Theorem

# A True, Unprovable Statement

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

# A True, Unprovable Statement

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

**Construction:** Let $M$ be a Turing machine that operates as follows.

- Delete the input
- Look for proof of $\neg\exists x[\varphi_{M,0}] \in \text{Th}(\mathbb{N}, +, \times)$
- Accept on proof, reject if no proof can be found

# A True, Unprovable Statement

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

**Construction:** Let $M$ be a Turing machine that operates as follows.

- Delete the input
- Look for proof of $\neg\exists x[\varphi_{M,0}] \in \mathrm{Th}\,(\mathbb{N}, +, \times)$
- Accept on proof, reject if no proof can be found

$\Longrightarrow \quad \neg\exists x[\varphi_{M,0}]$ 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=O4ndIDcDSGc

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