TECHNISCHE UNIVERSITÄT DRESDEN

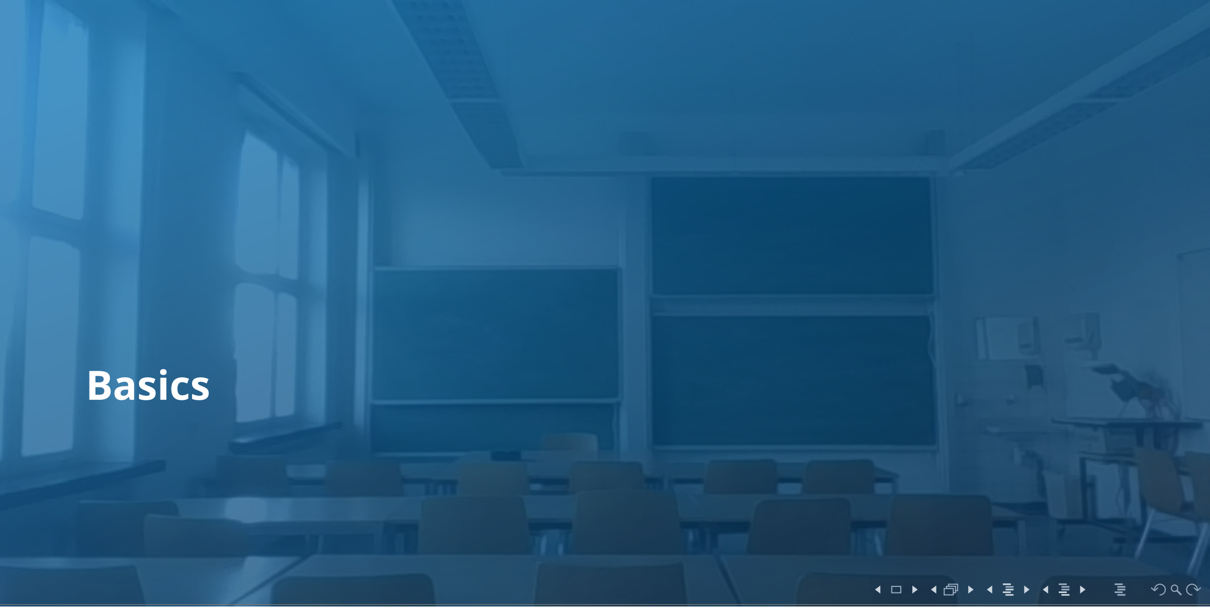
DRESDEN concept

Lucas Waclawczyk

Decidability of Logical Theories

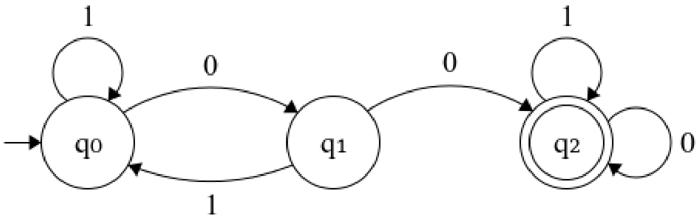
Proseminar Theoretical Computer Science // Dresden, May 4, 2020

Inhalt



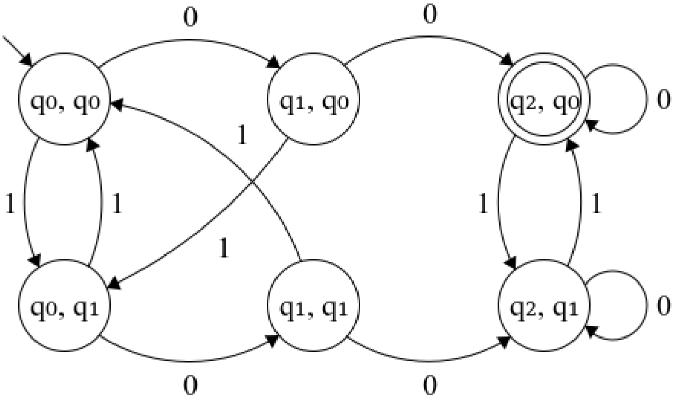
Basics

Automata



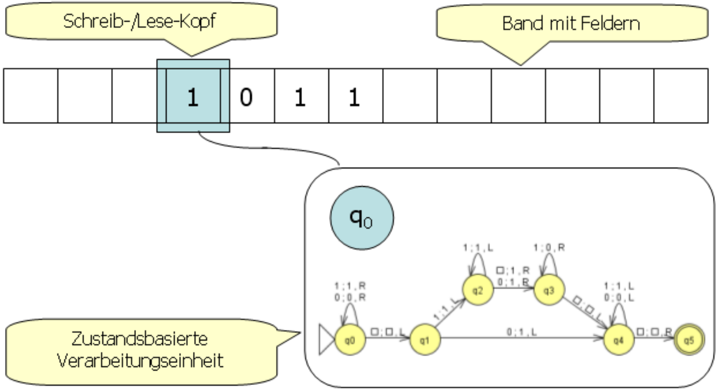
Example automaton that accepts all binary strings containing "00"
source: [1]

Automata



Example automaton that accepts all binary strings containing an even number of "1"
source: [1]

Turing Machines



Example turing machine
source: [2]

First-Order Logic

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

First-Order Logic

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

"There are infinitely many prime numbers."

First-Order Logic

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

"There are infinitely many prime numbers."

- *universe* $\mathcal{U}$
  - set of assignable variable values
  - here $\mathbb{N}$

First-Order Logic

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

$$= \quad \forall q\, \exists p\, \forall x, y\ \Big[R_1(p, q) \land \Big(\big(R_1(x, 1) \land R_1(y, 1)\big) \to R_2(x, y, p)\Big)\Big]$$

"There are infinitely many prime numbers."

- *universe* $\mathcal{U}$
  - set of assignable variable values
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe with assignment of relations
  - here $(\mathbb{N},\ >_2,\ (\times \neq)_3)$

First-Order Logic

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

"There are infinitely many prime numbers."

- *universe* $\mathcal{U}$
  - set of assignable variable values
  - here $\mathbb{N}$
- *model* $\mathcal{M}$
  - universe with assignment of relations
  - here $(\mathbb{N}, >_2, (\times \neq)_3)$

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

First-Order Logic

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

"There are infinitely many prime numbers."

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

What Is Decidability?

Generally:

- M, N sets, $\varphi \in M$
- N decidable $:=$ there is an algorithm that decides whether $\varphi \in N$

What Is Decidability?

Generally:

- M, N sets, $\varphi \in M$
- N decidable $:=$ there is an algorithm that decides whether $\varphi \in N$

For logic:

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

Proof of Theorem 1

Let $i \in \mathbb{N} \setminus \{0\}$ and define

$$\Sigma_i \quad := \quad \left\{ \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \right\} \quad \subset \quad \{0,1\}^i$$

and $\Sigma_0 := \{()\}$. An example for a word in Σ_2:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \sim \quad \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

Proof of Theorem 1

Now, let

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

Proof of Theorem 1

Now, let

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

Also, for $a_1, \ldots, a_i \in \mathbb{N}$, let $\varphi_i(a_1, \ldots, a_i)$ be φ_i with all occurrences of x_j replaced by a_j for $j \in \{1, \ldots, l\}$.

$$\Longrightarrow \qquad \varphi_l \text{ is only a Boolean expression.}$$

Proof of Theorem 1

Take

- an automaton that accepts simple addition ($a + b = c$)
- an automaton that accepts boolean "and" expressions ($p \land q$)
- an automaton that accepts boolean "or" expressions ($p \lor q$)
- an automaton that accepts boolean "not" expressions ($\neg p$)

Proof of Theorem 1

Take

- an automaton that accepts simple addition ($a + b = c$)
- an automaton that accepts boolean "and" expressions ($p \wedge q$)
- an automaton that accepts boolean "or" expressions ($p \vee q$)
- an automaton that accepts boolean "not" expressions ($\neg p$)

Combine them (closure, union, intersection, complementation) to get the automaton A_l that accepts tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true.

Proof of Theorem 1

Take

- an automaton that accepts simple addition ($a + b = c$)
- an automaton that accepts boolean "and" expressions ($p \wedge q$)
- an automaton that accepts boolean "or" expressions ($p \vee q$)
- an automaton that accepts boolean "not" expressions ($\neg p$)

Combine them (closure, union, intersection, complementation) to get the automaton A_l that accepts tuples $(a_1, \ldots, a_l) \in \mathbb{N}^l$ for which $\varphi_l(a_1, \ldots, a_l)$ is true.

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 all states
- adding a new start state
- making A_i guess the right a_{i+1} indeterministically

Proof of Theorem 1

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

- copying all states
- adding a new start state
- making A_i guess the right a_{i+1} indeterministically

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 all states
- adding a new start state
- making A_i guess the right a_{i+1} indeterministically

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

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

Proof of Theorem 1

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

- copying all states
- adding a new start state
- making A_i guess the right a_{i+1} indeterministically

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 all states
- adding a new start state
- making A_i guess the right a_{i+1} indeterministically

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

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

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 $\varphi_{M,w} \in \mathrm{Th}\,(\mathbb{N}, +, \times)$ 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 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 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

We can construct a true statement in 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 $\text{Th}\,(\mathbb{N}, +, \times)$, 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

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