

Lecture 1: Introduction to Computability

Models of Computation

<https://clegra.github.io/moc.html>

Clemens Grabmayer

Ph.D. Program, Advanced Courses Period
Gran Sasso Science Institute
L'Aquila, Italy

July 6, 2026

Course overview

Monday, July 6 10.30 – 12.30	Tuesday, July 7 10.30 – 12.30	Wednesday, July 8 10.30 – 12.30	Thursday, July 9 10.30 – 12.30	Friday, July 10
<i>intro</i>	<i>classic models</i>			<i>additional models</i>
Introduction to Computability	Machine Models	Recursive Functions	Lambda Calculus	
computation and decision problems, from logic to computability, overview of models of computation relevance of MoCs	Post Machines, typical features, Turing's analysis of human computers, Turing machines, basic recursion theory	primitive recursive functions, Gödel–Herbrand recursive functions, partial recursive funct's, partial recursive = Turing-computable, Church's Thesis	λ -terms, β -reduction, λ -definable functions, partial recursive = λ -definable = Turing computable	
	<i>imperative programming</i>	<i>algebraic programming</i>	<i>functional programming</i>	
				14.30 – 16.30
				Three more Models of Computation
				Post's Correspondence Problem, Interaction-Nets, Fractran
				comparing computational power

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability
- ▶ some models of computation

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability
- ▶ some models of computation
- ▶ example relevance: calculator

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability
- ▶ some models of computation
- ▶ example relevance: calculator
- ▶ fields for which models of computation are important

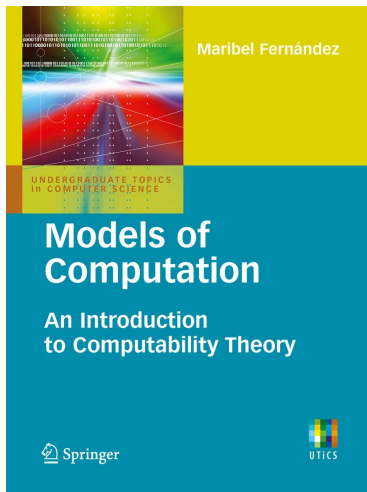
Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability
- ▶ some models of computation
- ▶ example relevance: calculator
- ▶ fields for which models of computation are important
- ▶ recommended reading

Today

- ▶ What is computation?
 - ▶ questions where the answer may depend on computation
 - ▶ algorithm examples
 - ▶ unsolvable problems
- ▶ from logic to computability
- ▶ some models of computation
- ▶ example relevance: calculator
- ▶ fields for which models of computation are important
- ▶ recommended reading
- ▶ references

Book



Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes.

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896.

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

A: ??

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

A: ??

Q: Will the rise in sea level until 2100 (worldwide yearly average) be more than 0.5 m?

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

A: ??

Q: Will the rise in sea level until 2100 (worldwide yearly average) be more than 0.5 m?

A: ?

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

A: ??

Q: Will the rise in sea level until 2100 (worldwide yearly average) be more than 0.5 m?

A: ? A 2010 Dutch study (KNMI) projected 0.47 m.

Q's where the A's depend (somehow) on computation

Q: Is $2^{20} > 1\,000\,000$?

A: Yes. (Check by computing $2^{20} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{20} = 1\,048\,576$).

Q: When was the last leap year before 1903?

A: 1896. (Use the rule: leap years are divisible by 4, but not divisible by 100 with the exception of those divisible by 400.)

Q: Given a day d in the second week of July 2026, will the sunshine percentage in L'Aquila on day d exceed 70%?

A: ??

Q: Will the rise in sea level until 2100 (worldwide yearly average) be more than 0.5 m?

A: ? A 2010 Dutch study (KNMI) projected 0.47 m.
In the meantime already ~ 1 m is being projected.

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

A: ? (... to be given in a moment...)

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

A: ? (... to be given in a moment...)

Q: Is $((p \rightarrow q) \rightarrow p) \rightarrow p$ a tautology of propositional calculus?

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

A: ? (... to be given in a moment...)

Q: Is $((p \rightarrow q) \rightarrow p) \rightarrow p$ a tautology of propositional calculus?

A: Yes (Peirce's law).

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

A: ? (... to be given in a moment...)

Q: Is $((p \rightarrow q) \rightarrow p) \rightarrow p$ a tautology of propositional calculus?

A: Yes (Peirce's law).

Q: Given a formula ϕ of propositional logic, is ϕ a tautology?

Q's where the A's depend (somehow) on computation

Q: What number comes next in the series 10, 9, 60, 90, 70, 66?

A: ?

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?

A: ? (... to be given in a moment...)

Q: Is $((p \rightarrow q) \rightarrow p) \rightarrow p$ a tautology of propositional calculus?

A: Yes (Peirce's law).

Q: Given a formula ϕ of propositional logic, is ϕ a tautology?

A: Yes, if the truth table for ϕ contains (in the row for ϕ) only "T"; no otherwise.

(Comput.) Yes-or-no-questions / Decision problems

Example

Tautology Problem for the propositional calculus

Instance: A formula ϕ of propositional logic.

Question: Is ϕ a tautology?

(Comput.) Yes-or-no-questions / Decision problems

Example

Tautology Problem for the propositional calculus

Instance: A formula ϕ of propositional logic.

Question: Is ϕ a tautology?

Suppose $A \subseteq E$, where E a set of finitely describable objects.

A **decision method for A in E** is a method by which, given an element $a \in E$, we can **decide** in a **finite number** of **steps** whether or not $a \in A$.

(Comput.) Yes-or-no-questions / Decision problems

Example

Tautology Problem for the propositional calculus

Instance: A formula ϕ of propositional logic.

Question: Is ϕ a tautology?

Suppose $A \subseteq E$, where E a set of finitely describable objects.

A **decision method for A in E** is a method by which, given an element $a \in E$, we can **decide** in a **finite number** of **steps** whether or not $a \in A$.

Decision problem for A in E : Find a decision method for A in E , or show that no such method can exist.

(Comput.) Yes-or-no-questions / Decision problems

Example

Tautology Problem for the propositional calculus

Instance: A formula ϕ of propositional logic.

Question: Is ϕ a tautology?

Suppose $A \subseteq E$, where E a set of finitely describable objects.

A **decision method for A in E** is a method by which, given an element $a \in E$, we can **decide** in a **finite number** of **steps** whether or not $a \in A$.

Decision problem for A in E : Find a decision method for A in E , or show that no such method can exist.

The decision problem for A in E is **solvable** (the set A in E is **(effectively) calculable**) if there exists a decision method for A in E .

(Comput.) What-questions / Computation Problems

Example

Computing the greatest common divisor

Instance: a pair $\langle a, b \rangle$ of numbers $a, b \in \mathbb{N}$ with $a, b > 0$.

Question: What is $\text{gcd}(a, b)$, the greatest common divisor of a and b ?

(Comput.) What-questions / Computation Problems

Example

Computing the greatest common divisor

Instance: a pair $\langle a, b \rangle$ of numbers $a, b \in \mathbb{N}$ with $a, b > 0$.

Question: What is $\text{gcd}(a, b)$, the greatest common divisor of a and b ?

Suppose $F : A \rightarrow B$ is a mapping, where the elements of A, B are finitely describable objects.

A **computation method** for F is a method by which, given an element $a \in A$, we can **obtain solution** $F(a)$ in a **finite number** of **steps**.

(Comput.) What-questions / Computation Problems

Example

Computing the greatest common divisor

Instance: a pair $\langle a, b \rangle$ of numbers $a, b \in \mathbb{N}$ with $a, b > 0$.

Question: What is $\text{gcd}(a, b)$, the greatest common divisor of a and b ?

Suppose $F : A \rightarrow B$ is a mapping, where the elements of A, B are finitely describable objects.

A **computation method** for F is a method by which, given an element $a \in A$, we can **obtain solution** $F(a)$ in a **finite number** of **steps**.

Computation problem for F : Find a computation method for F , or show that no such method can exist.

(Comput.) What-questions / Computation Problems

Example

Computing the greatest common divisor

Instance: a pair $\langle a, b \rangle$ of numbers $a, b \in \mathbb{N}$ with $a, b > 0$.

Question: What is $\text{gcd}(a, b)$, the greatest common divisor of a and b ?

Suppose $F : A \rightarrow B$ is a mapping, where the elements of A, B are finitely describable objects.

A **computation method** for F is a method by which, given an element $a \in A$, we can **obtain solution** $F(a)$ in a **finite number** of **steps**.

Computation problem for F : Find a computation method for F , or show that no such method can exist.

A mapping F is **calculable** if there exists a computation method for F .

Representing function

Let $P(a_1, \dots, a_n)$ be an n -ary number-theoretic predicate.

The **representing function** f of P :

$$f(a_1, \dots, a_n) := \begin{cases} 1 & \dots P(a_1, \dots, a_n) \text{ is true} \\ 0 & \dots P(a_1, \dots, a_n) \text{ is false} \end{cases}$$

Hence:

A **decision procedure** can be handled as a **computation procedure** f by taking '0' for 'yes', and '1' for 'no'.

Decision / Computation procedures (steps)

What is a **computation method** (**procedure**) more precisely, with respect to its **steps**?

Decision / Computation procedures (steps)

- What is a **computation method** (**procedure**) more precisely, with respect to its **steps**?
- A **mechanical, algorithmic computation procedure** that:
 - ▶ can be carried out by a **machine M** (ideal, not limited by resource problems, mechanical breakdown, etc.).

Decision / Computation procedures (steps)

What is a **computation method (procedure)** more precisely, with respect to its **steps**?

- A **mechanical, algorithmic computation procedure** that:
 - ▶ can be carried out by a **machine M** (ideal, not limited by resource problems, mechanical breakdown, etc.).
 - ▶ for computing a function F on an argument a ,
 - ▶ a is placed on the input device of the M ,
 - ▶ which then produces $F(a)$ after **finitely many steps**.

Decision / Computation procedures (steps)

What is a **computation method** (**procedure**) more precisely, with respect to its **steps**?

- A **mechanical, algorithmic computation procedure** that:
 - ▶ can be carried out by a **machine** M (ideal, not limited by resource problems, mechanical breakdown, etc.).
 - ▶ for computing a function F on an argument a ,
 - ▶ a is placed on the input device of the M ,
 - ▶ which then produces $F(a)$ after **finitely many steps**.
 - ▶ for computing a function F ,
 - ▶ the **machine** M that is chosen for obtaining $F(a)$ may **not** be **different** for **different** arguments a

Decision / Computation procedures (steps)

- What is a **computation method** (**procedure**) more precisely, with respect to its **steps**?
- A **mechanical, algorithmic computation procedure** that:
 - ▶ can be carried out by a **machine** M (ideal, not limited by resource problems, mechanical breakdown, etc.).
 - ▶ for computing a function F on an argument a ,
 - ▶ a is placed on the input device of the M ,
 - ▶ which then produces $F(a)$ after **finitely many steps**.
 - ▶ for computing a function F ,
 - ▶ the **machine** M that is chosen for obtaining $F(a)$ may **not** be **different** for **different** arguments a
 - Similar for a **decision methods**.

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(i.e. solvable for $x, y \in \mathbb{Z}$?)

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Using **Euclid's algorithm** we calculate $\gcd(15, 9)$:

$$15 : 9 = 1 \text{ rem } 6$$

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Using **Euclid's algorithm** we calculate $\gcd(15, 9)$:

$$\begin{array}{rclcl} 15 & : & 9 & = & 1 & \text{rem } 6 \\ 9 & : & 6 & = & 1 & \text{rem } 3 \end{array}$$

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Using **Euclid's algorithm** we calculate $\gcd(15, 9)$:

$$\begin{array}{rclcl} 15 & : & 9 & = & 1 & \text{rem } 6 \\ 9 & : & 6 & = & 1 & \text{rem } 3 \\ 6 & : & 3 & = & 2 & \text{rem } 0 \end{array}$$

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Using **Euclid's algorithm** we calculate $\gcd(15, 9)$:

$$\begin{array}{rclclcl} 15 & : & 9 & = & 1 & \text{rem} & 6 \\ 9 & : & 6 & = & 1 & \text{rem} & 3 \\ 6 & : & 3 & = & 2 & \text{rem} & 0 \end{array}$$

We find: $\gcd(15, 9) = 3$.

Solvability by an effective procedure

Q: Is the diophantine equation $15x + 9y + 12 = 0$ solvable?
(I.e. solvable for $x, y \in \mathbb{Z}$?)

From elementary number theory we know:

$$ax + by + c = 0 \text{ solvable in } \mathbb{Z} \iff \gcd(a, b) \mid c \quad (*)$$

Using **Euclid's algorithm** we calculate $\gcd(15, 9)$:

$$\begin{array}{rclcl} 15 & : & 9 & = & 1 & \text{rem } 6 \\ 9 & : & 6 & = & 1 & \text{rem } 3 \\ 6 & : & 3 & = & 2 & \text{rem } 0 \end{array}$$

We find: $\gcd(15, 9) = 3$.

Due to $3 \mid 12$ and $(*)$ we conclude:

A: **Yes.** (Infinitely many solutions, e.g. $x = 4$ and $y = -8$.)

Not effectively calculable

Examples (Shoenfield)

- ▶ methods that involve chance procedures: tossing a coin

Not effectively calculable

Examples (Shoenfield)

- ▶ methods that involve chance procedures: tossing a coin
- ▶ methods involving magic: asking a fortune teller

Not effectively calculable

Examples (Shoenfield)

- ▶ methods that involve chance procedures: tossing a coin
- ▶ methods involving magic: asking a fortune teller
- ▶ methods that require (unformalised, unmechanised) insight

Effectively calculable?

Example

Hilbert's 10th Problem

Instance: An equation $p(x_1, \dots, x_n) = 0$, where
 p a polynomial with integer coefficients.

Question: Is the equation solvable for $x_1, \dots, x_n \in \mathbb{Z}$?

Instances based on quadratic polynomials are of the form
 $ax^2 + bxy + cy^2 + dx + ey + f = 0$ with $a, b, c, d, e, f \in \mathbb{Z}$.

Effectively calculable? – No!

Example

Hilbert's 10th Problem

Instance: An equation $p(x_1, \dots, x_n) = 0$, where
 p a polynomial with integer coefficients.

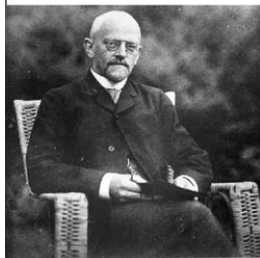
Question: Is the equation solvable for $x_1, \dots, x_n \in \mathbb{Z}$?

Instances based on quadratic polynomials are of the form
 $ax^2 + bxy + cy^2 + dx + ey + f = 0$ with $a, b, c, d, e, f \in \mathbb{Z}$.

Theorem (Matijasevic, 1970)

Hilbert's 10th Problem is unsolvable.

David Hilbert (1862–1943)



Hilbert

Problem (Entscheidungsproblem, 1928)

Is there a method for deciding, given a formula ϕ of the predicate calculus, whether or not ϕ is a tautology?

Timeline: From logic to computability

1900 Hilbert's 23 Problems in mathematics

Timeline: From logic to computability

1900 Hilbert's 23 Problems in mathematics

1921 Schönfinkel: Combinatory logic

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus
- 1931 Gödel: incompleteness theorems for first-order arithmetic

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus
- 1931 Gödel: incompleteness theorems for first-order arithmetic
- 1932 Church: λ -calculus

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus
- 1931 Gödel: incompleteness theorems for first-order arithmetic
- 1932 Church: λ -calculus
- 1933/34 Herbrand/Gödel: general recursive functions

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus
- 1931 Gödel: incompleteness theorems for first-order arithmetic
- 1932 Church: λ -calculus
- 1933/34 Herbrand/Gödel: general recursive functions
- 1936 Church/Kleene: λ -definable \sim general recursive
Church Thesis: 'effectively calculable' be defined as either
Church shows: the 'Entscheidungsproblem' is unsolvable

Timeline: From logic to computability

- 1900 Hilbert's 23 Problems in mathematics
- 1921 Schönfinkel: Combinatory logic
- 1928 Hilbert/Ackermann: formulate completeness/decision problems for the predicate calculus (the latter called 'Entscheidungsproblem')
- 1929 Presburger: completeness/decidability of theory of addition on \mathbb{Z}
- 1930 Gödel: completeness theorem of predicate calculus
- 1931 Gödel: incompleteness theorems for first-order arithmetic
- 1932 Church: λ -calculus
- 1933/34 Herbrand/Gödel: general recursive functions
- 1936 Church/Kleene: λ -definable \sim general recursive
Church Thesis: 'effectively calculable' be defined as either
Church shows: the 'Entscheidungsproblem' is unsolvable
- 1937 Post: machine model; Church's thesis as 'working hypothesis'
Turing: convincing analysis of a 'human computer'
leading to the 'Turing machine'

Calculable functions?

Questions/Exercises

- 1 Suppose $P(a, b)$ is a calculable predicate.
Why does $(\exists x)P(a, x)$ not have to be calculable?

Calculable functions?

Questions/Exercises

- 1 Suppose $P(a, b)$ is a calculable predicate.
Why does $(\exists x)P(a, x)$ not have to be calculable?
- 2 Let $f : \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$n \mapsto \begin{cases} 0 & \dots n = 0 \ \& \text{Goldbach's conjecture is false} \\ 1 & \dots n = 0 \ \& \text{Goldbach's conjecture is true} \\ n + 1 & \dots n > 0 \end{cases}$$

Is f calculable?

Calculable functions?

Questions/Exercises

- 1 Suppose $P(a, b)$ is a calculable predicate.
Why does $(\exists x)P(a, x)$ not have to be calculable?
- 2 Let $f : \mathbb{N} \rightarrow \mathbb{N}$ defined by

$$n \mapsto \begin{cases} 0 & \dots n = 0 \ \& \text{Goldbach's conjecture is false} \\ 1 & \dots n = 0 \ \& \text{Goldbach's conjecture is true} \\ n + 1 & \dots n > 0 \end{cases}$$

Is f calculable?

- 3 Can computation problems for mappings $F : \mathbb{N}^n \rightarrow \mathbb{N}^m$ always be represented by decision problems?

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
	hypercomputation	<i>speculative</i>
	quantum computing bio-computing reversible computing	<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
	Combinatory Logic	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
	Combinatory Logic λ -calculus	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system)	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
		<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks		<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
		<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
	hypercomputation	<i>speculative</i>
		<i>physics-/biology- inspired</i>

Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
	hypercomputation	<i>speculative</i>
	quantum computing	<i>physics-/biology- inspired</i>

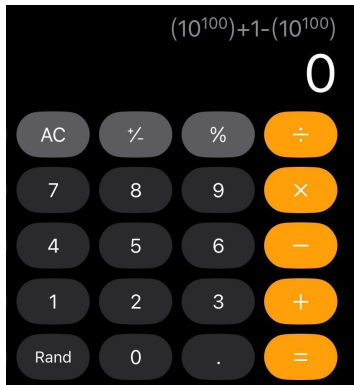
Some Models of Computation

machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
	hypercomputation	<i>speculative</i>
	quantum computing bio-computing	<i>physics-/biology- inspired</i>

Some Models of Computation

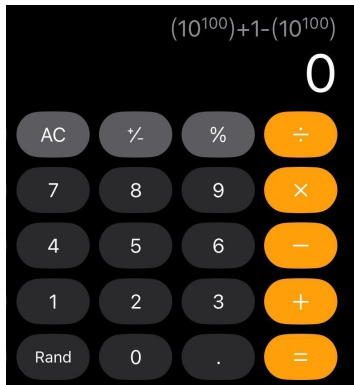
machine model	mathematical model	sort
Turing machine Post machine register machine	Combinatory Logic λ -calculus Herbrand–Gödel recursive functions partial-recursive/ μ -recursive functions Post canonical system (tag system) Post's Correspondence Problem Markov algorithms Lindenmayer systems	<i>classical</i>
	Fractran	<i>less well known</i>
cellular automata neural networks	term rewrite systems interaction nets logic-based models of computation concurrency and process algebra ζ -calculus evolutionary programming/genetic algorithms abstract state machines	<i>modern</i>
	hypercomputation	<i>speculative</i>
	quantum computing bio-computing reversible computing	<i>physics-/biology- inspired</i>

Example MoC relevance: Calculator (1/5)

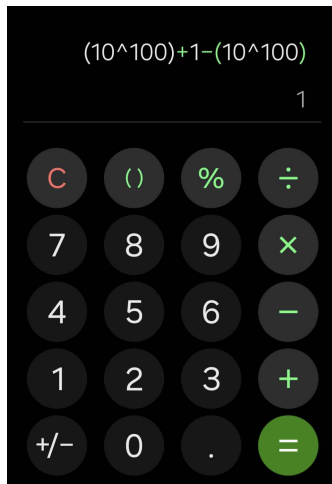


iOS

Example MoC relevance: Calculator (1/5)

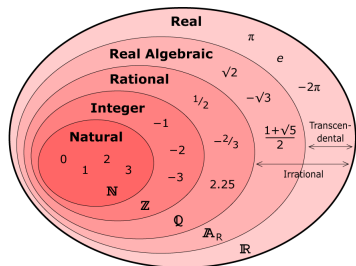


iOS



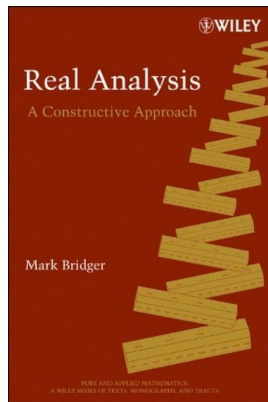
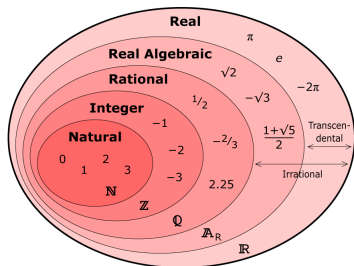
Android

Calculator (2/5): constructive real numbers



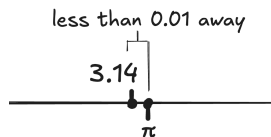
subclasses of real numbers \mathbb{R}

Calculator (2/5): constructive real numbers



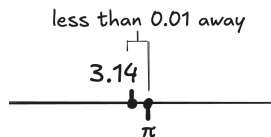
subclasses of real numbers \mathbb{R}

Calculator (3/5): constructive real numbers



approximating π within 0.01

Calculator (3/5): constructive real numbers



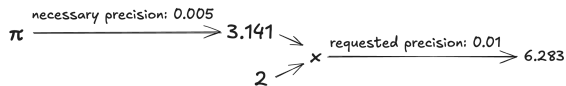
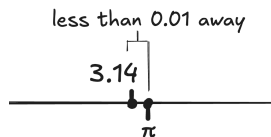
approximating π within 0.01

Definition

A real number $x \in \mathbb{R}$ is **constructive** if:

- ▶ there exists a program P_x that for every bound $0 < \delta \in \mathbb{Q}$ returns a **rational** approximation $P_x(\delta) \in \mathbb{Q}$ of x with $|x - P_x(\delta)| < \delta$.

Calculator (3/5): constructive real numbers



approximating π within 0.01

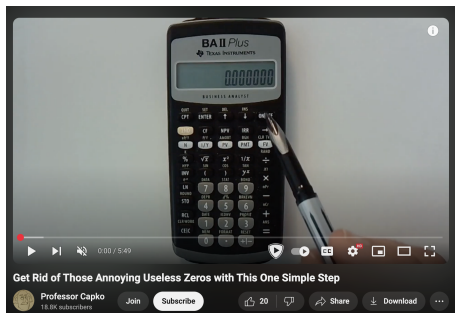
approximating 2π within 0.01

Definition

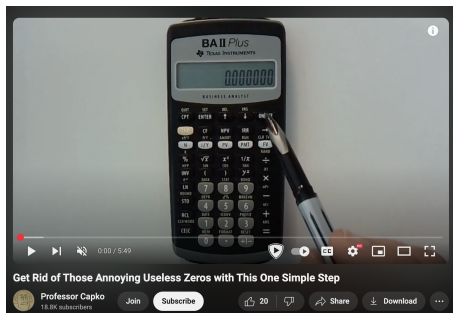
A real number $x \in \mathbb{R}$ is **constructive** if:

- ▶ there exists a program P_x that for every bound $0 < \delta \in \mathbb{Q}$ returns a **rational** approximation $P_x(\delta) \in \mathbb{Q}$ of x with $|x - P_x(\delta)| < \delta$.

Calculator (4/5): constructive real numbers

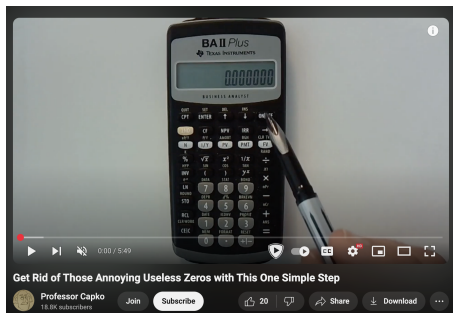


Calculator (4/5): constructive real numbers



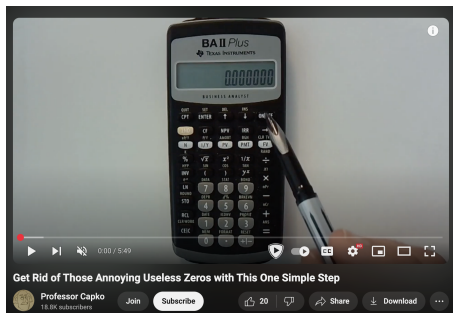
- ▶ How to recognize that 2 constructive reals x and y are the same?

Calculator (4/5): constructive real numbers



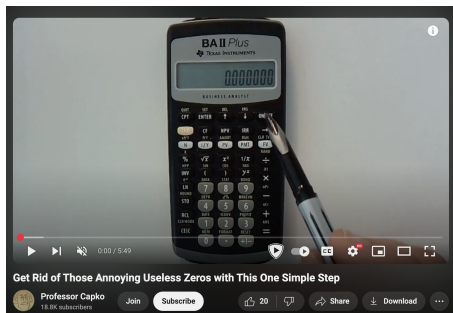
- ▶ How to recognize that 2 constructive reals x and y are the same?
- ▶ Does there exist an program *Compare* that given P_x and P_y decides whether $x = y$?

Calculator (4/5): constructive real numbers



- ▶ How to recognize that 2 constructive reals x and y are the same?
- ▶ Does there exist an program *Compare* that given P_x and P_y decides whether $x = y$?
- ▶ **No!** This problem is **undecidable**.

Calculator (4/5): constructive real numbers



Undecidable problem

Article Talk

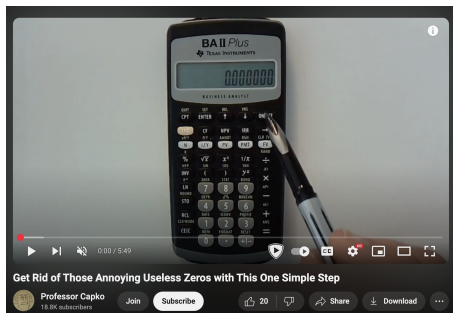


! This article needs additional citations for verification.
(July 2019) [Learn more](#)

In [computability theory](#) and [computational complexity theory](#), an **undecidable problem** is a [decision problem](#) for which it is proved to be impossible to construct an [algorithm](#) that always leads to a correct yes-or-no answer. The [halting problem](#) is an example: it can be proven that there is no algorithm that correctly determines whether an arbitrary program eventually halts when run.^[1]

- ▶ How to recognize that 2 constructive reals x and y are the same?
- ▶ Does there exist an program *Compare* that given P_x and P_y decides whether $x = y$?
- ▶ **No!** This problem is **undecidable**.

Calculator (4/5): constructive real numbers



Undecidable problem

Article Talk



! This article needs additional citations for verification.
(July 2019) [Learn more](#)

In [computability theory](#) and [computational complexity theory](#), an **undecidable problem** is a [decision problem](#) for which it is proved to be impossible to construct an [algorithm](#) that always leads to a correct yes-or-no answer. The [halting problem](#) is an example: it can be proven that there is no algorithm that correctly determines whether an arbitrary program eventually halts when run.^[1]

- ▶ How to recognize that 2 constructive reals x and y are the same?
- ▶ Does there exist an program *Compare* that given P_x and P_y decides whether $x = y$?
- ▶ **No!** This problem is **undecidable**.
- ▶ **Therefore** $x - y = 0$ can not always be decided.

Calculator (5/5): Böhm's full precision calculator



- ▶ Hans-Jürgen Böhm's Android full precision calculator

Calculator (5/5): Böhm's full precision calculator



- ▶ Hans-Jürgen Böhm's Android full precision calculator
- ▶ uses products of:
 - ▶ full-precision rational arithmetic,
 - ▶ either of:
 - (a) symbolic representations of π , e , and natural numbers, such \sqrt{x} , e^x , $\ln(x)$, $\log_{10}(x)$, $\sin(\pi x)$, $\tan(\pi x)$ for $x \in \mathbb{Q}$.
 - (b) constructive real numbers

Calculator (5/5): Böhm's full precision calculator



Rational

Can only represent fractions
Exact and easy to work with

RRA

Can represent any computable real
Inexact and impossible to check equality

- ▶ Hans-Jürgen Böhm's Android full precision calculator
- ▶ uses products of:
 - ▶ full-precision rational arithmetic,
 - ▶ either of:
 - (a) symbolic representations of π , e , and natural numbers, such \sqrt{x} , e^x , $\ln(x)$, $\log_{10}(x)$, $\sin(\pi x)$, $\tan(\pi x)$ for $x \in \mathbb{Q}$.
 - (b) constructive real numbers

Calculator (5/5): Böhm's full precision calculator



Rational

Can only represent fractions
Exact and easy to work with

RRA

Can represent any computable real
Inexact and impossible to check equality

- ▶ Hans-Jürgen Böhm's Android full precision calculator
- ▶ uses products of:
 - ▶ full-precision rational arithmetic,
 - ▶ either of:
 - (a) symbolic representations of π , e , and natural numbers, such \sqrt{x} , e^x , $\ln(x)$, $\log_{10}(x)$, $\sin(\pi x)$, $\tan(\pi x)$ for $x \in \mathbb{Q}$.
 - (b) constructive real numbers
- ▶ **Equality of products** with symbolic representations **can be decided!**
(But not equality of products with at least one constructive real number)

Calculator (5/5): Böhm's full precision calculator



Rational

Can only represent fractions
Exact and easy to work with

RRA

Can represent any computable real
Inexact and impossible to check equality

- ▶ Hans-Jürgen Böhm's Android full precision calculator
- ▶ uses products of:
 - ▶ full-precision rational arithmetic,
 - ▶ either of:
 - (a) symbolic representations of π , e , and natural numbers, such \sqrt{x} , e^x , $\ln(x)$, $\log_{10}(x)$, $\sin(\pi x)$, $\tan(\pi x)$ for $x \in \mathbb{Q}$.
 - (b) constructive real numbers
- ▶ **Equality of products** with symbolic representations **can be decided!**
(But not equality of products with at least one constructive real number)
- ▶ **Credits:** tech-blogger **Chad Nauseam** ([link](#)) for post
"A calculator app? Anyone could make that." ([link](#)) [2].

Some fields in which MoC's are important (I)

Complexity theory

- ▶ recognize problems as being **decidable**
- ▶ study the **computational complexity** of **decidable** problems (classification of problems into hierarchies)

Some fields in which MoC's are important (I)

Complexity theory

- ▶ recognize problems as being **decidable**
- ▶ study the **computational complexity** of **decidable** problems (classification of problems into hierarchies)

Recursion theory

- ▶ a **theory of computability** for sets and functions on \mathbb{N} (including **degrees of unsolvability** of **decidable** problems)

Some fields in which MoC's are important (I)

Complexity theory

- ▶ recognize problems as being **decidable**
- ▶ study the **computational complexity** of **decidable** problems (classification of problems into hierarchies)

Recursion theory

- ▶ a **theory of computability** for sets and functions on \mathbb{N} (including **degrees of unsolvability** of **decidable** problems)

Logic and Philosophy

- ▶ MoC's important for studying **un-/decidability** of logical theories

Some fields in which MoC's are important (I)

Complexity theory

- ▶ recognize problems as being **decidable**
- ▶ study the **computational complexity** of **decidable** problems (classification of problems into hierarchies)

Recursion theory

- ▶ a **theory of computability** for sets and functions on \mathbb{N} (including **degrees of unsolvability** of **decidable** problems)

Logic and Philosophy

- ▶ MoC's important for studying **un-/decidability** of logical theories

Rewriting

- ▶ **study in a systematic way** the operational and denotational aspects of MoC's like λ -calculus, CL, string rewriting, term rewriting, interaction nets

Some fields in which MoC's are important (II)

Computer Science

- ▶ e.g. **functional programming**: using/implementing λ -calculus

Some fields in which MoC's are important (II)

Computer Science

- ▶ e.g. **functional programming**: using/implementing λ -calculus

Neuro-psychology, Cognitive Modelling

- ▶ e.g. developing formal **platforms** for studying **human cognition**

Some fields in which MoC's are important (II)

Computer Science

- ▶ e.g. **functional programming**: using/implementing λ -calculus

Neuro-psychology, Cognitive Modelling

- ▶ e.g. developing formal **platforms** for studying **human cognition**

Artificial Intelligence

- ▶ use knowledge of human mind to model it in an artificial system
- ▶ modeling by machines to better understand the human mind
- ▶ **understand the inherent complexity of problems (un-/decidable?)**

Some fields in which MoC's are important (II)

Computer Science

- ▶ e.g. **functional programming**: using/implementing λ -calculus

Neuro-psychology, Cognitive Modelling

- ▶ e.g. developing formal **platforms** for studying **human cognition**

Artificial Intelligence

- ▶ use knowledge of human mind to model it in an artificial system
- ▶ modeling by machines to better understand the human mind
- ▶ **understand the inherent complexity of problems (un-/decidable?)**

Linguistics

- ▶ e.g. **formal calculi** for discovering the **structure of human languages** related to subclasses in the **Chomsky hierarchy**

Recommended reading

① Post machine: Page 1 + first paragraph on page 2 of:

- ▶ Emil Post: *Finite Combinatory Processes – Formulation 1*, Journal of Symbolic Logic (1936), [3], <https://www.wolframscience.com/prizes/tm23/images/Post.pdf>.

Recommended reading

- 1 **Post machine:** Page 1 + first paragraph on page 2 of:
 - ▶ Emil Post: *Finite Combinatory Processes – Formulation 1*, Journal of Symbolic Logic (1936), [3], <https://www.wolframscience.com/prizes/tm23/images/Post.pdf>.
- 2 **Turing machine motivation:** Turing's analysis of a human computer:
Part I of Section 9, pp. 249–252 of:
 - ▶ Alan M. Turing's: *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society (1936), [4], <http://www.wolframscience.com/prizes/tm23/images/Turing.pdf>.

Course overview

Monday, July 6 10.30 – 12.30	Tuesday, July 7 10.30 – 12.30	Wednesday, July 8 10.30 – 12.30	Thursday, July 9 10.30 – 12.30	Friday, July 10
<i>intro</i>	<i>classic models</i>			<i>additional models</i>
Introduction to Computability	Machine Models	Recursive Functions	Lambda Calculus	
computation and decision problems, from logic to computability, overview of models of computation relevance of MoCs	Post Machines, typical features, Turing's analysis of human computers, Turing machines, basic recursion theory	primitive recursive functions, Gödel–Herbrand recursive functions, partial recursive funct's, partial recursive = Turing-computable, Church's Thesis	λ -terms, β -reduction, λ -definable functions, partial recursive = λ -definable = Turing computable	
	<i>imperative programming</i>	<i>algebraic programming</i>	<i>functional programming</i>	
				14.30 – 16.30
				Three more Models of Computation
				Post's Correspondence Problem, Interaction-Nets, Fractran comparing computational power

References I



Maribel Fernández.

Models of Computation (An Introduction to Computability Theory).

Springer, Dordrecht Heidelberg London New York, 2009.



Chad Nauseam.

“A calculator app? Anyone could make that.”.

<https://chadnauseam.com/coding/random/calculator-app>, 2025.

Accessed: 29 June 2025.



Emil Leon Post.

Finite Combinatory Processes – Formulation 1.

Journal of Symbolic Logic, 1(3):103–105, 1936.

<https://www.wolframscience.com/prizes/tm23/images/Post.pdf>.

References II



Alan M. Turing.

On Computable Numbers, with an Application to the Entscheidungsproblem.

Proceedings of the London Mathematical Society,
42(2):230–265, 1936.

[http://www.wolframscience.com/prizes/tm23/
images/Turing.pdf](http://www.wolframscience.com/prizes/tm23/images/Turing.pdf).