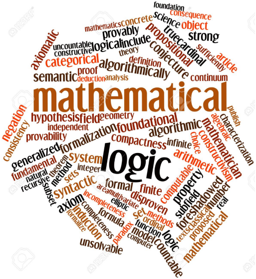
1-3 常用的证明方法

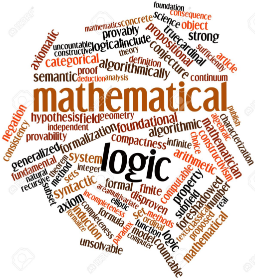
魏恒峰

hfwei@nju.edu.cn

2017 年 10 月 23 日


foundation
consequence
science
object
strong
mathematics
concrete
provably
axiomatic
uncountable
constructive
logical
include
theory
propositional
truecardinal
sufficiently
article
categorical
algorithmically
definition
conjecture
semantic
proof
deduction
analysis
continuum
mathematical
hypothesis
field
geometry
negation
consistency
independent
provability
foundational
algorithmic
mathematician
algebraic
characterization
publish
formalization
compactness
infinite
choice
constructivism
generalized
fundamental
natural
recursive
theorem
method
system
integer
logic
arithmetic
sets
syntactic
subset
formal
finite
computable
property
axiom
incompleteness
multivariate
elliptic
disproven
subfield
foreshadowed
number
real
completeness
formula
set
methods
ordinal
function
model
computer
paradox
induction
formalize
unsolvable
countable
proposed
classical
mathematical


数理逻辑
证明及其限度


mathematical
logic
foundational
algorithmic
formalization
compactness
arithmetic
mathematician
hypothesis
field
geometry
axiomatic
categorical
algorithmically
semantic
proof
propositional
conjecture
provably
logical
include
negation
consistency
provability
generalized
fundamental
system
syntactic
axiom
formal
finite
disproven
unsolvable
induction
formula
property
computable
foreshadowed
number
countable


Engines of Logic
逻辑的引擎

习题选讲

UD (第五章)	反证法 (Contradiction)
UD (第十七章)	数学归纳法 (Mathematical Induction)
ES (第二十四章)	鸽笼原理 (Pigeonhole Principle)

习题选讲

UD (第五章)	反证法 (Contradiction)
UD (第十七章)	数学归纳法 (Mathematical Induction)
ES (第二十四章)	鸽笼原理 (Pigeonhole Principle)

UD 题目 17.14：第二数学归纳法

使用 (第一) 数学归纳法证明第二数学归纳法。

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

UD 题目 17.14：第二数学归纳法

使用 (第一) 数学归纳法证明第二数学归纳法。

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

UD 题目 17.14：第二数学归纳法

使用 (第一) 数学归纳法证明：

Theorem (Second Principle of Mathematical Induction)

For an integer n, let $Q(n)$ denote an assertion. Suppose that

(i) $Q(1)$ *is true and*

(ii) *for all positive integers n, if $Q(1), \cdots, Q(n)$ are true, then $Q(n+1)$ is true.*

Then $Q(n)$ holds for all positive integers n.

Theorem (第二数学归纳法)

$$\Big[Q(1) \wedge \forall n \in \mathbb{N}^+ \Big((Q(1) \wedge \cdots \wedge Q(n)) \to Q(n+1) \Big) \Big] \to \forall n \in \mathbb{N}^+ Q(n).$$

Theorem (第二数学归纳法)

$$\Big[Q(1) \land \forall n \in \mathbb{N}^+ \Big((Q(1) \land \cdots \land Q(n)) \to Q(n+1) \Big) \Big] \to \forall n \in \mathbb{N}^+ Q(n).$$

Theorem ((第一) 数学归纳法)

$$\Big[P(1) \land \forall n \in \mathbb{N}^+ (P(n) \to P(n+1)) \Big] \to \forall n \in \mathbb{N}^+ P(n).$$

“标准” 证明示例。

$$P(n) \triangleq Q(1) \wedge \cdots \wedge Q(n)$$

用(第一) 数学归纳法证明 $P(n)$ 对一切正整数都成立。

"标准" 证明示例。

$$P(n) \triangleq Q(1) \land \cdots \land Q(n)$$

用(第一) 数学归纳法证明 $P(n)$ 对一切正整数都成立。

Proof.

By mathematical induction on $\mathbb{N}^+$.

Basis Step $P(1)$

Inductive Step $P(n) \to P(n+1)$

Therefore, $P(n)$ holds for all positive integers. $\square$

"标准" 证明示例。

$$P(n) \triangleq Q(1) \land \cdots \land Q(n)$$

用(第一) 数学归纳法证明 $P(n)$ 对一切正整数都成立。

Proof.

By mathematical induction on $\mathbb{N}^+$.

$$\begin{aligned} \text{Basis Step} \quad & P(1) \\ \text{Inductive Hypothesis} \quad & P(n) \\ \text{Inductive Step} \quad & P(n) \to P(n+1) \end{aligned}$$

Therefore, $P(n)$ holds for all positive integers. □

Proof.

能不能"算一算"呢?

$$P(n) \triangleq Q(1) \wedge \cdots \wedge Q(n)$$

□

Let us calculate [calculemus].

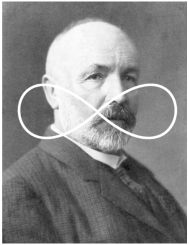
数学归纳法

(第一) 数学归纳法与第二数学归纳法等价。

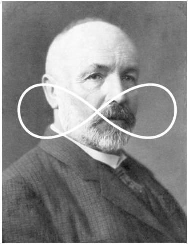
数学归纳法

(第一) 数学归纳法与第二数学归纳法等价。

Q : 为什么第二数学归纳法也被称为"强"(strong) 数学归纳法?



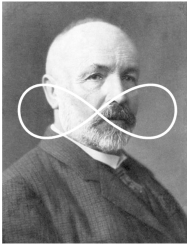
Georg Cantor (1845 – 1918)



Georg Cantor (1845 – 1918)

Leopold Kronecker

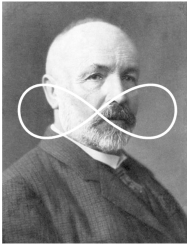
(1823 – 1891)



Georg Cantor (1845 – 1918)

Leopold Kronecker
(1823 – 1891)

Henri Poincaré
(1854 – 1912)



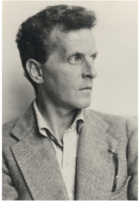
Georg Cantor (1845 – 1918)

Leopold Kronecker

(1823 – 1891)

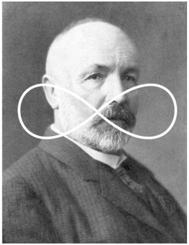
Henri Poincaré

(1854 – 1912)



Ludwig Wittgenstein

(1889 – 1951)

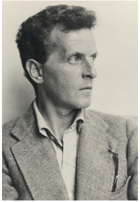


Georg Cantor (1845 – 1918)

David Hilbert (1862 – 1943)

Leopold Kronecker (1823 – 1891)

Henri Poincaré (1854 – 1912)



Ludwig Wittgenstein (1889 – 1951)

From his paradise that Cantor with us unfolded, we hold our breath in awe; knowing, we shall not be expelled.

— David Hilbert

没有人能把我们从 *Cantor* 创造的乐园中驱逐出去。

From his paradise that Cantor with us unfolded, we hold our breath in awe; knowing, we shall not be expelled.

— David Hilbert

没有人能把我们从 *Cantor* 创造的乐园中驱逐出去。

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Theorem (Cantor Theorem)

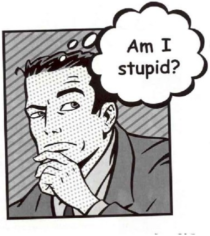
Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.



Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Understanding this problem:

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Understanding this problem:

2^A $A = \{1, 2, 3\}$,

$$2^A = \Big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\Big\}$$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Understanding this problem:

2^A $A = \{1, 2, 3\}$,

$$2^A = \Big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\Big\}$$

Onto

$$\forall B \in 2^A \ \exists a \in A \ (f(a) = B).$$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Understanding this problem:

2^A $A = \{1, 2, 3\}$,

$$2^A = \Big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\Big\}$$

Onto

$$\forall B \in 2^A \; \exists a \in A \; (f(a) = B).$$

Not Onto

$$\exists B \in 2^A \; \neg\Big(\exists a \in A \; (f(a) = B)\Big).$$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Proof.

$\square$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Proof.

- Constructive proof:

$$B = \{x \in A \mid x \notin f(x)\}.$$

□

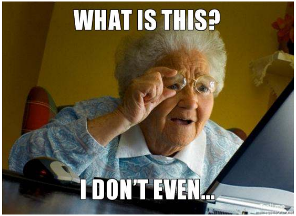
Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Proof.

- Constructive proof:

$$B = \{x \in A \mid x \notin f(x)\}.$$



□

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

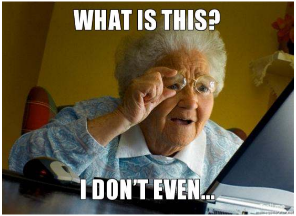
Proof.

- Constructive proof:

$$B = \{x \in A \mid x \notin f(x)\}.$$

- By contradiction:

$$\exists a \in A : f(a) = B.$$



Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

Proof.

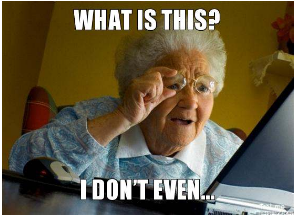
- Constructive proof:

$$B = \{x \in A \mid x \notin f(x)\}.$$

- By contradiction:

$$\exists a \in A : f(a) = B.$$

$$Q : a \in B \; (= f(a))?$$



Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

对角线论证 (Cantor's diagonal argument) (以下仅适用于可数集合 A).

a	$f(a)$					
	1	2	3	4	5	$\cdots$
1	1	1	0	0	1	$\cdots$
2	0	0	0	0	0	$\cdots$
3	1	0	0	1	0	$\cdots$
4	1	1	1	1	1	$\cdots$
5	0	1	0	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$

$\square$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

对角线论证 (Cantor's diagonal argument) (以下仅适用于可数集合 A).

a	$f(a)$					
	1	2	3	4	5	$\cdots$
1	1	1	0	0	1	$\cdots$
2	0	0	0	0	0	$\cdots$
3	1	0	0	1	0	$\cdots$
4	1	1	1	1	1	$\cdots$
5	0	1	0	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$

Theorem (Cantor Theorem)

Let A be a set.
If $f : A \to 2^A$, then f is not onto.

对角线论证 (Cantor's diagonal argument) (以下仅适用于可数集合 A).

a	$f(a)$					
	1	2	3	4	5	$\cdots$
1	1	1	0	0	1	$\cdots$
2	0	0	0	0	0	$\cdots$
3	1	0	0	1	0	$\cdots$
4	1	1	1	1	1	$\cdots$
5	0	1	0	1	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$

$$B = \{0, 1, 1, 0, 1\}$$

$\square$

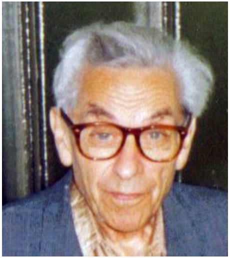
补充思考题

存在性证明 (Existence Proof)

1. **构造性证明 (Constructive proof)**
2. **反证法 (By contradiction)**

存在性证明 (Existence Proof)

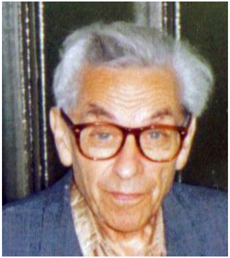
1. **构造性证明** (Constructive proof)
2. **反证法** (By contradiction)
3. 概率法 (Probabilistic Method)



Paul Erdős (1913 – 1996)

存在性证明 (Existence Proof)

1. **构造性证明** (Constructive proof)
2. **反证法** (By contradiction)
3. 概率法 (Probabilistic Method)



Paul Erdős (1913 – 1996)

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

$\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$ (UD: Theorem 5.2)

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

$$\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q} \qquad \text{(UD: Theorem 5.2)}$$

Proof.

$$\sqrt{2}^{\sqrt{2}}$$

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

$$\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q} \qquad \text{(UD: Theorem 5.2)}$$

Proof.

$$\sqrt{2}^{\sqrt{2}}$$

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}}$$

□

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

$$\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q} \qquad \text{(UD: Theorem 5.2)}$$

Proof.

$$\sqrt{2}^{\sqrt{2}}$$

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}}$$

□

Q : 这是构造性证明吗?

Theorem (Dov Jarden (1953))

$$\exists a, b \in \mathbb{R} \setminus \mathbb{Q} : a^b \in \mathbb{Q}.$$

$$\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q} \qquad \text{(UD: Theorem 5.2)}$$

Proof.

$$\sqrt{2}^{\sqrt{2}}$$

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}}$$

□

Q : 这是构造性证明吗？这是反证法吗？

Lossless Compression Algorithms

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Understanding this problem:

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Understanding this problem:

File f: a string of bits of a finite length $|f|$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Understanding this problem:

File f: a string of bits of a finite length $|f|$

Compression Alg.: a function $\mathcal{C}$

$$\mathcal{C} : F \to F$$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Understanding this problem:

File f: a string of bits of a finite length $|f|$

Compression Alg.: a function $\mathcal{C}$

$$\mathcal{C} : F \to F$$

Lossless: f is injective

$$\mathcal{C}(f_1) = \mathcal{C}(f_2) \implies f_1 = f_2$$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

Understanding this problem:

File f: a string of bits of a finite length $|f|$

Compression Alg.: a function $\mathcal{C}$

$$\mathcal{C} : F \to F$$

Lossless: f is injective

$$\mathcal{C}(f_1) = \mathcal{C}(f_2) \implies f_1 = f_2$$

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Proof.

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Proof.

- By contradiction

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Proof.

- By contradiction

$$\exists \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \land (\forall f \in F : |\mathcal{C}(f)| \leq |f|)$$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Proof.

- By contradiction

$$\exists \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \land (\forall f \in F : |\mathcal{C}(f)| \leq |f|)$$

M : # of bits of a shortest file f such that $(|\mathcal{C}(f)| = N) < (|f| = M)$

Theorem ("No Free Lunch" Theorem)

Any lossless (file) compression algorithm that makes some files shorter must necessarily make some files longer.

$$\boxed{\forall \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \to (\exists f \in F : |\mathcal{C}(f)| > |f|)}$$

Proof.

- By contradiction

$$\exists \mathcal{C} : (\exists f \in F : |\mathcal{C}(f)| < |f|) \land (\forall f \in F : |\mathcal{C}(f)| \leq |f|)$$

M : # of bits of a shortest file f such that $(|\mathcal{C}(f)| = N) < (|f| = M)$

- By the pigeonhole principle

$$2^N + 1 \textit{ vs. } 2^N$$

□

Longest Monotone Subsequence

Example (ES 24.8： Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Example (ES 24.8： Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Understanding this problem:

Example (ES 24.8: Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Understanding this problem:

Subsequence *vs.* substring

Example (ES 24.8: Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Understanding this problem:

Subsequence *vs.* substring

Monotone increasing *vs.* decreasing

Example (ES 24.8: Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Understanding this problem:

Subsequence *vs.* substring

Monotone increasing *vs.* decreasing strictly *vs.* non-strictly

Example (ES 24.8: Longest Monotone Subsequence)

Write a computer program that takes as its input a sequence of distinct integers and returns as its output the length of a longest monotone subsequence.

Understanding this problem:

Subsequence *vs.* substring

Monotone increasing *vs.* decreasing strictly *vs.* non-strictly

Longest existence? uniqueness?

ES 24.8： Longest Increasing Subsequence

- Given an integer array $A[1 \dots n]$
- To find (the length L of) a longest increasing subseqence.

$$5, 2, 8, 6, 3, 6, 9, 7 \implies 2, 3, 6, 9$$

学生反馈： 这道题为什么放在 “Pigeonhole Principle” 这一章?

学生反馈： 这道题为什么放在 “Pigeonhole Principle” 这一章？

学生反馈： 这道题为什么放在 "Pigeonhole Principle" 这一章?

Theorem (Erdős-Szekeres Theorem)

Let n be a positive integer. Every sequence of n^2+1 distinct integers must contain a monotone subsequence of length $n+1$.

Q：这道题与数学归纳法有什么关系？

Q : 这道题与数学归纳法有什么关系?

B.S. $P(1)$

I.H. $P(n)$

I.S. $P(n) \to P(n+1)$

$P(n)$ 是什么?

$L(i)$: the length of an LIS *ending at* i.

$L(i)$: the length of an LIS *ending at* i.

$$L = \max_{1 \le i \le n} L(i)$$

$L(i)$: the length of an LIS *ending at* i.

$$L = \max_{1 \le i \le n} L(i)$$

$$L(1) = 1$$
$$L(i) = 1 + \max\{L(j) : j < i \land A[j] < A[i]\}$$

$L(i)$: the length of an LIS *ending at* i.

$$L = \max_{1 \le i \le n} L(i)$$

$$L(1) = 1$$
$$L(i) = 1 + \max\{L(j) : j < i \land A[j] < A[i]\}$$


Thank You!