

An Uncountable Chain in  $\{0, 1\}^{\mathbb{N}}$

# An Uncountable Chain in $\{0, 1\}^{\mathbb{N}}$

Part I: the proof I showed in class, with a bit more details.

# An Uncountable Chain in $\{0, 1\}^{\mathbb{N}}$

Part I: the proof I showed in class, with a bit more details.

**Definition.** Let  $X$  and  $Y$  be two partially ordered sets.

A function  $f : X \rightarrow Y$  is an *isomorphism* if

- $f$  is bijective,
- $x_1 \leq x_2$  if and only if  $f(x_1) \leq f(x_2)$ .

If such an  $f$  exists, we say  $X$  and  $Y$  are *isomorphic* and write  $X \cong Y$ .

# An Uncountable Chain in $\{0, 1\}^{\mathbb{N}}$

Part I: the proof I showed in class, with a bit more details.

**Definition.** Let  $X$  and  $Y$  be two partially ordered sets.

A function  $f : X \rightarrow Y$  is an *isomorphism* if

- $f$  is bijective,
- $x_1 \leq x_2$  if and only if  $f(x_1) \leq f(x_2)$ .

If such an  $f$  exists, we say  $X$  and  $Y$  are *isomorphic* and write  $X \cong Y$ .

Intuitive meaning:  $X$  and  $Y$  being isomorphic means that they look identical, differing only by the names of their elements.

01串和自然数的子集等势

**Observation 1.**  $(\{0, 1\}^{\mathbb{N}}, \leq)$  and  $(2^{\mathbb{N}}, \subseteq)$  are isomorphic.

**Observation 2.**  $(2^{\mathbb{N}}, \subseteq)$  and  $(2^{\mathbb{Q}}, \subseteq)$  are isomorphic.

**Observation 3.** If  $X$  and  $Y$  are isomorphic, then  $X$  has an uncountable chain if and only if  $Y$  has an uncountable chain.

**Theorem.**  $(2^{\mathbf{Q}}, \subseteq)$  has an uncountable chain.

**Proof.** For a real number  $x$ , define

$$B_x := \{q \in \mathbf{Q} \mid q < x\}.$$

Define  $C := \{B_x \mid x \in \mathbf{R}\}$ .

- $C$  is a chain. Any  $B_x, B_y$  are comparable. Indeed, if  $x \leq y$  then  $B_x \subseteq B_y$ .
- $C$  is uncountable. Indeed, the function  $f : \mathbf{R} \rightarrow C$  defined by  $f(x) = B_x$  is injective.

**Theorem.**  $(2^{\mathbf{Q}}, \subseteq)$  has an uncountable chain.

**Proof.** For a real number  $x$ , define

$$B_x := \{q \in \mathbf{Q} \mid q < x\}.$$

Define  $C := \{B_x \mid x \in \mathbf{R}\}$ .

- $C$  is a chain. Any  $B_x, B_y$  are comparable. Indeed, if  $x \leq y$  then  $B_x \subseteq B_y$ .
- $C$  is uncountable. Indeed, the function  $f : \mathbf{R} \rightarrow C$  defined by  $f(x) = B_x$  is injective.

**Corollary.**  $(\{0, 1\}^{\mathbf{N}}, \leq)$  has an uncountable chain.

Okay, maybe this was a bit mysterious...

Let's give a (longer) proof that actually shows how the elements of the chain are constructed.



Okay, maybe this was a bit mysterious...

Let's give a (longer) proof that actually shows how the elements of the chain are constructed.

We'll define a function  $f$  that takes as input an infinite bit sequence  $\mathbf{a} \in \{0, 1\}^{\mathbb{N}}$  and outputs an infinite bit sequence  $f(\mathbf{a}) \in \{0, 1\}^{\mathbb{N}}$  such that

1.  $f$  is an injection.
2. All output elements  $f(\mathbf{a})$  are comparable.

Point 1 will ensure the set of outputs is uncountable,  
Point 2 will ensure it is a chain.

# Example of our procedure

input sequence



01101001...

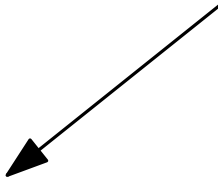
# Example of our procedure

input sequence



01101001...

output sequence



\*\*\*\*\*...

# Example of our procedure

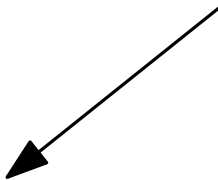
input sequence



01101001...

output sequence

just infinitely many \* in the beginning



\*\*\*\*\*...

01101001...

\*\*\*\*\*.....

read first bit of input



01101001...

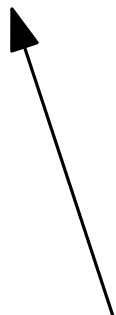
\*\*\*\*\*...

read first bit of input



01101001...

\* \* \* \* \* . . .



put it here

read first bit of input



01101001...

0\*\*\*\*\*...



put it here



read first bit of input



01101001...

0\*\*\*\*\*...

read first bit of input



01101001...

Rule 1: Read bit of input. In output,  
replace first \* by that bit.

0\*\*\*\*\*...

read first bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

0\*\*\*\*\*...

read first bit of input




01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.


0\*\*\*\*\*...



01101001...

## Rule 2:

- [illegible]



01101001...

## Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

[illegible]

01101001...

## Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

[illegible]

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

[illegible]



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

## Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

[illegible]

01101001...

## Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

[illegible]

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

0010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*0...

read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

0010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*0...

read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

0010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*0...

read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

00101010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*0...

read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

00101010\*010\*010\*010\*010\*010\*010\*010\*010\*010\*0...

read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

00101010\*0101010\*0101010\*0101010\*0101010\*0...



read next bit of input



01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

...

00101010\*0101010\*0101010\*0101010\*0101010\*0...

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

...  
00101010\*0101010\*0101010\*0101010\*0101010\*0...

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

...

0010101000101010\*0101010\*0101010\*0101010\*0...

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

0010101000101010\*0101010\*0101010\*0101010\*0...

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

001010100010101000101010\*010101000101010\*0...

read next bit of input

01101001...

Rule 1: Read bit of input. In output, replace first \* by that bit.

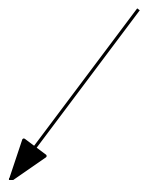
Rule 2:

- If that bit is 0, replace every other \* by 0, starting with the first \*.
- If that bit is 1, replace every other \* by 1, starting with the second \*.

001010100010101000101010\*010101000101010\*0...  
...

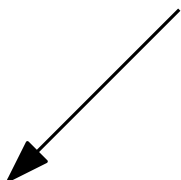
AND SO ON FOREVER

input **a**



01101001...

output  $f(\mathbf{a})$



001010100010101000101010\*010101000101010\*0...

**Claim.** This procedure is injective and produces a chain.

**Proof.** Let **a** and **b** be two different input sequences.

Let  $i$  be the first coordinate where  $a_i \neq b_i$ .

Let's assume  $a_i = 0$ ,  $b_i = 1$ .

Let's run the previous procedure on **a** and **b** and stop just before it reads the  $i^{\text{th}}$  bit.



Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast$$

$$f(\mathbf{b}) = \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast$$

Input:

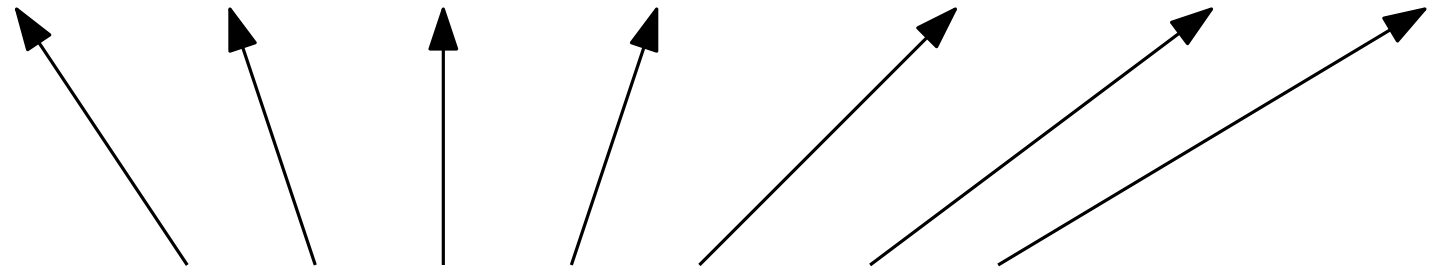
$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast \dots$$

$$f(\mathbf{b}) = \dots \ast \dots \ast \dots \ast \dots \ast \dots \ast \dots$$



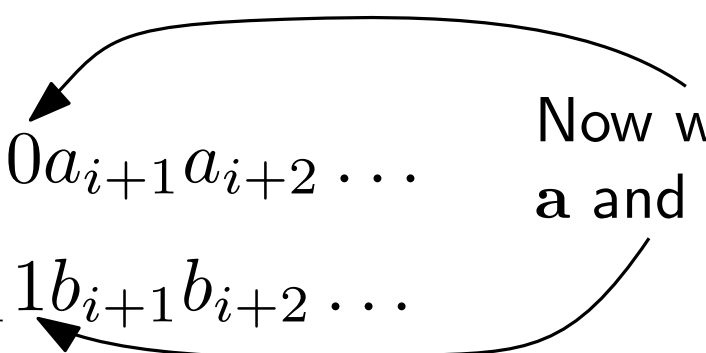
These parts of  $f(\mathbf{a})$  and  $f(\mathbf{b})$  consist of 0's and 1's. They are equal, because the parts of  $\mathbf{a}$  and  $\mathbf{b}$  read so far as identical.

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Now we read the next bit of  
**a** and **b**



Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots * \dots * \dots * \dots * \dots * \dots * \dots$$

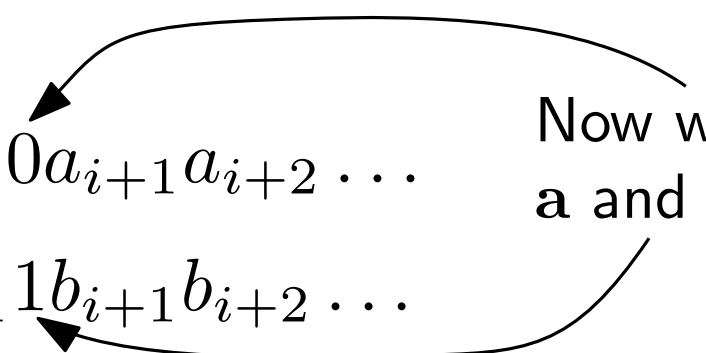
$$f(\mathbf{b}) = \dots * \dots * \dots * \dots * \dots * \dots * \dots$$

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Now we read the next bit of  
**a** and **b**



Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots 0 \dots * \dots * \dots * \dots * \dots * \dots$$

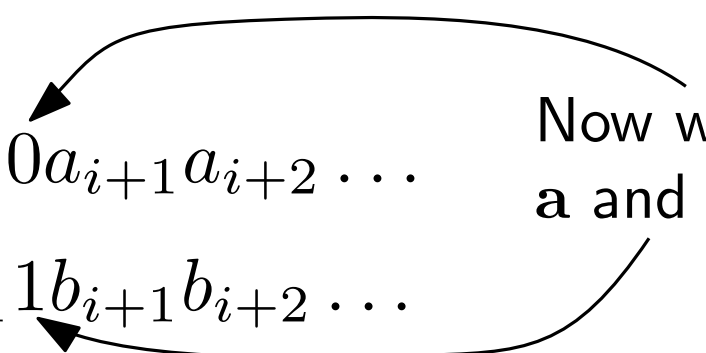
$$f(\mathbf{b}) = \dots 1 \dots * \dots * \dots * \dots * \dots * \dots$$

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Now we read the next bit of  
**a** and **b**



Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots 0 \dots 0 \dots * \dots 0 \dots * \dots 0 \dots$$

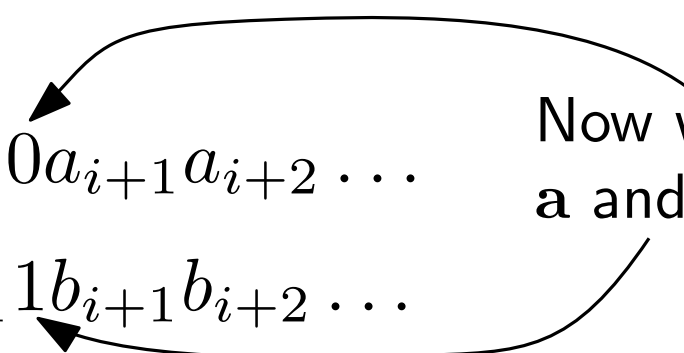
$$f(\mathbf{b}) = \dots 1 \dots * \dots 1 \dots * \dots 1 \dots * \dots$$

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Now we read the next bit of  
**a** and **b**



Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots 0 \dots 0 \dots * \dots 0 \dots * \dots 0 \dots$$

$$f(\mathbf{b}) = \dots 1 \dots * \dots 1 \dots * \dots 1 \dots * \dots$$

Whatever happens from now on, it is clear that

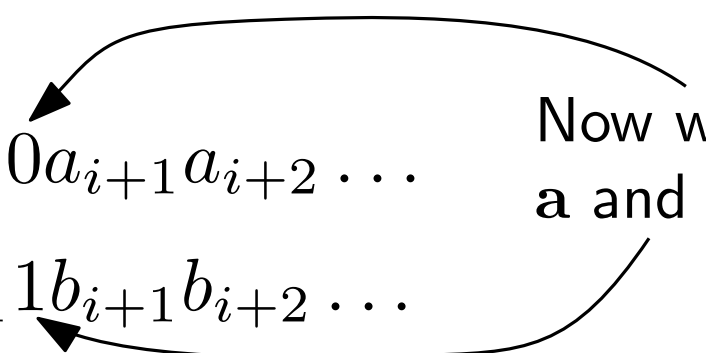
$$f(\mathbf{a}) < f(\mathbf{b}).$$

Input:

$$\mathbf{a} = a_1 a_2 \dots a_{i-1} 0 a_{i+1} a_{i+2} \dots$$

$$\mathbf{b} = a_1 a_2 \dots a_{i-1} 1 b_{i+1} b_{i+2} \dots$$

Now we read the next bit of  
**a** and **b**



Input, just before reading bit  $i$ :

$$f(\mathbf{a}) = \dots 0 \dots 0 \dots * \dots 0 \dots * \dots 0 \dots$$

$$f(\mathbf{b}) = \dots 1 \dots * \dots 1 \dots * \dots 1 \dots * \dots$$

Whatever happens from now on, it is clear that

$$f(\mathbf{a}) < f(\mathbf{b}).$$

So  $f$  is injective and  $\text{Im}(f)$  is a chain.